

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

Felipe Martins Pinheiro Silva

**ANÁLISE DE PARCELAMENTO DE DÉBITOS INSCRITOS EM DÍVIDA ATIVA DA  
UNIÃO ATRAVÉS DE MACHINE LEARNING**

Belo Horizonte

2022

Felipe Martins Pinheiro Silva

**ANÁLISE DE PARCELAMENTO DE DÉBITOS INSCRITOS EM DÍVIDA ATIVA DA  
UNIÃO ATRAVÉS DE MACHINE LEARNING**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de Da-  
dos e Big Data como requisito parcial à obtenção  
do título de especialista.

Belo Horizonte

2022

Felipe Martins Pinheiro Silva

## **ANÁLISE DE PARCELAMENTO DE DÉBITOS INSCRITOS EM DÍVIDA ATIVA DA UNIÃO ATRAVÉS DE MACHINE LEARNING**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de Da-  
dos e Big Data como requisito parcial à obtenção  
do título de especialista.

Professor 1

---

Professor 1 – PUC Minas

Professor 2

---

Professor 2 – PUC Minas

Professor 3

---

Professor 3 – PUC Minas

Belo Horizonte, 30 de maio de 2022

À minha esposa, Natália, por ter me motivado,  
e pela paciência nos momentos que estive au-  
sente em virtude desse trabalho, sobretudo por  
estarmos nos primeiros meses de vida da  
nossa amada filha, Catarina.

## RESUMO

O presente trabalho é uma tentativa de identificação de padrões nos débitos inscritos em Dívida Ativa da União que foram parcelados pelos contribuintes, com o objetivo de gerar um modelo preditivo. Para isso, foram utilizadas técnicas de ciência de dados, em especial, de *machine learning*. As bases de dados originais foram extraídas de bancos de dados públicos fornecidos pela RFB e PGFN, quais sejam, respectivamente, base CNPJ e base de débitos inscritos em DAU não previdenciários. Todo o trabalho, desde a coleta, passando pelo processamento, tratamento, análise exploratória, até a aplicação dos algoritmos de *machine learning*, foi realizado na plataforma *Jupyter Notebook*, onde utilizou-se a linguagem *Python*. Ao se aplicar *machine learning*, verificou-se tratar-se de uma tarefa de aprendizado supervisionado, do tipo classificação. No processamento lidou-se com classes desbalanceadas, pois apenas 23% dos débitos encontravam-se parcelados, o que exigiu ajustes no algoritmo classificador. Por fim, o modelo apresentou resultados apenas razoáveis, possivelmente em virtude do desbalanceamento, além da insuficiência de dados.

Palavras-chave: Ciência de Dados. *Machine Learning*. *Python*. Aprendizado Supervisionado. Classificação. Classes Desbalanceadas. Administração Tributária. Dívida Ativa. Parcelamento.

## SUMÁRIO

<b>1. Introdução.....</b>	<b>7</b>
<b>1.1. Contextualização.....</b>	<b>8</b>
<b>1.2. O problema proposto.....</b>	<b>12</b>
<b>2. Coleta de Dados .....</b>	<b>15</b>
<b>3. Processamento/Tratamento de Dados .....</b>	<b>19</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>42</b>
<b>5. Criação de Modelos de Machine Learning .....</b>	<b>70</b>
<b>5.1. Teoria.....</b>	<b>70</b>
<b>5.2. Prática .....</b>	<b>78</b>
<b>6. Apresentação dos Resultados .....</b>	<b>87</b>
<b>6.1. Avaliação dos resultados .....</b>	<b>87</b>
<b>6.2. Conclusão .....</b>	<b>93</b>
<b>7. Links.....</b>	<b>94</b>
<b>REFERÊNCIAS.....</b>	<b>96</b>

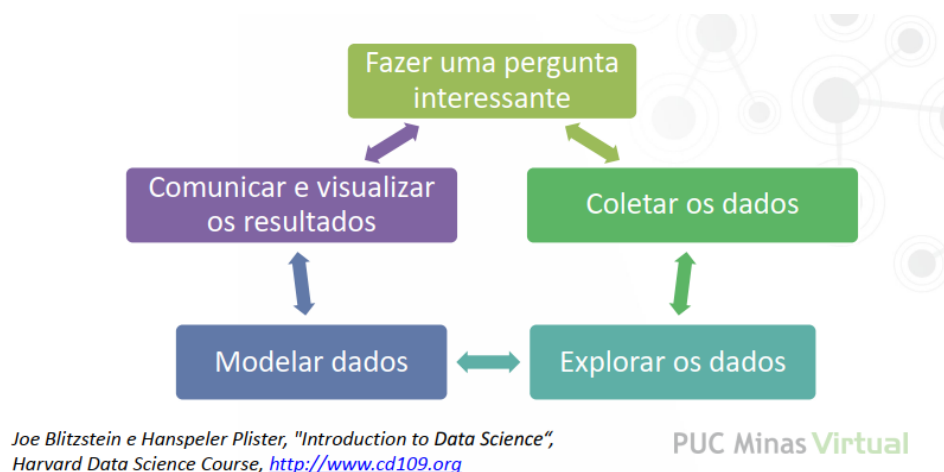
## 1. INTRODUÇÃO

O presente trabalho consiste em tentar identificar padrões nos débitos inscritos em Dívida Ativa da União que foram parcelados pelos contribuintes, com o objetivo de levantar estatísticas e gerar um modelo preditivo. Para isso, serão utilizadas técnicas de ciência de dados, em especial, de *machine learning* (ou aprendizado de máquina) aplicada em linguagem *Python*, num contexto de *big data*.

Esperamos que este trabalho propicie ao leitor um melhor entendimento não apenas da administração tributária brasileira, mas, principalmente, do universo da ciência de dados, suas diversas técnicas de manipulação e análise de dados que podem ser replicadas em qualquer área de estudo.

Existem vários modelos de processos que evoluíram ao longo tempo e continuam evoluindo. Apresenta-se na Figura 1, o processo de ciência de dados proposto por Blitztein e Plister, citado pelo professor Paula, da PUC Minas, e ensinado no presente curso de pós-graduação e que servirá de estrutura geral deste trabalho.

**Figura 1 – O processo de Ciência de Dados**



Fonte: Blitztein; Plister, apud Paula (2019).

O capítulo 1 (introdução) está relacionado ao item “fazer uma pergunta interessante”. Nele, faremos uma contextualização do assunto e proporemos um problema.

No capítulo 2 trataremos da coleta de dados. Será revelado onde os dados originais foram obtidos, o formato e estrutura dos *datasets* (ou conjunto de dados)<sup>1</sup>, o relacionamento entre estes, além de informações sobre os campos selecionados.

No capítulo 3 detalharemos como se deu o processamento e tratamento dos dados obtidos. Falaremos sobre os processos de seleção, limpeza, enriquecimento e codificação.

No capítulo 4 faremos a análise e exploração dos dados obtidos, utilizando-se de estatística descritiva.

No capítulo 5, o principal deste trabalho, faremos a modelagem dos dados, utilizando-se de algoritmos de *machine learning*.

No capítulo 6, o principal deste trabalho, faremos a apresentação dos resultados obtidos no capítulo anterior, além da conclusão.

Por fim, no capítulo 7, disponibilizamos os *links* para os arquivos utilizados, além do vídeo de apresentação.

### **1.1. Contextualização**

Poucas bases de dados podem ser consideradas tão massivas e tão fiel ao termo “Big Data” quanto os dados tributários. Sua base cadastral, de débitos, declarações, transações comerciais, financeiras, aduaneiras, de praticamente todas as pessoas e empresas de uma determinada população, certamente se encaixam nas características de big data, especialmente em sua expressão mais evidente: volume.

---

<sup>1</sup> Em linhas gerais, um *dataset* é um conjunto de dados (ou valores) dispostos em formas de tabela, onde as linhas são os registros e as colunas são os campos (ou variáveis) que apresentam as características (*features*) dos registros.



Ressalte-se que o volume não é a única dimensão a ser considerada. Há ainda uma “variedade imensa de dados, não estruturados, dentro e fora das empresas (coletados nas mídias sociais, por exemplo), que precisam ser validados (terem veracidade para serem usados) e tratados em velocidade adequada para terem valor para o negócio” (TAURION, 2013). Volume, variedade, velocidade, veracidade e valor são os chamados 5 V’s.

A TechAmerica Foundation, citada por Gandomi e Haider, define *big data* como “grandes volumes de dados de alta velocidade, complexos e variáveis que requerem técnicas e tecnologias avançadas para permitir a captura, armazenamento, distribuição, gerenciamento e análise das informações”. (TechAmerica Foundation apud GANDOMI; HAIDER, 2012, tradução nossa) <sup>2</sup>.

Por sua vez, *machine learning* (ou aprendizado de máquina) é uma das formas de tratamento e análise dessas informações. Ficamos com a ótima definição de Heller (2019), cuja comparação com a computação tradicional de um sistema baseado em regras, deixa bastante evidenciado seu conceito:

Machine learning é um ramo da inteligência artificial que inclui métodos ou algoritmos para criar modelos automaticamente a partir de dados. Ao contrário de um sistema que executa uma tarefa seguindo regras explícitas, um sistema de aprendizado de máquina aprende com a experiência. Enquanto um sistema baseado em regras executará uma tarefa da mesma maneira todas as vezes (para melhor ou para pior), o desempenho de um sistema de aprendizado de máquina pode ser melhorado por meio de treinamento, expondo o algoritmo a mais dados. (HELLER, 2019, tradução nossa) <sup>3</sup>.

Outro importante conceito sobre o tema é Mineração de Dados (*Data Mining*). Segundo a Gartner Group:

---

<sup>2</sup> *Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information.*

<sup>3</sup> *Machine learning is a branch of artificial intelligence that includes methods, or algorithms, for automatically creating models from data. Unlike a system that performs a task by following explicit rules, a machine learning system learns from experience. Whereas a rule-based system will perform a task the same way every time (for better or worse), the performance of a machine learning system can be improved through training, by exposing the algorithm to more data.*

Mineração de Dados (Data Mining) é o processo de descoberta de novas e significativas correlações, padrões e tendências em grandes volumes de dados, por meio do uso de técnicas e reconhecimento de padrões, estatística e outras ferramentas matemáticas. Para encontrar padrões, o processo de Data Mining utiliza técnicas de Machine Learning (Aprendizado de Máquina). (Gartner Group apud ESCOVEDO; KOSHIYAMA, 2020).

Ou seja, *big data*, *machine learning* e *data mining* são técnicas que estão intrinsecamente relacionadas e serão conjugadas nesse trabalho, para extração e análise de uma base de dados de alto volume.

Voltando à análise de dados tributários, considerando o exposto, é possível afirmar que sim, estes são passíveis de aplicação das técnicas de *big data*. No entanto, no Brasil, esbarramos no sigilo fiscal.

Segundo Carvalho (2019), “os direitos e garantias fundamentais dos cidadãos previstos na legislação brasileira, permitem uma ampla argumentação no sentido de se resguardar os dados, a privacidade e a intimidade dos contribuintes. Por outro lado, há uma enorme necessidade de se ampliar os mecanismos e ferramentas de combate aos crimes contra a ordem tributária, como a sonegação fiscal.”

Após fazer um levantamento da legislação brasileira e experiências no âmbito da Receita Federal, além da utilização de *big data* em outros países/organizações (compartilhamento automático de dados pela OCDE e mineração de dados pela IRS/EUA – lá, como aqui, também há críticas e preocupações quanto à privacidade de dados), o autor conclui que, no âmbito interno da administração tributária brasileira, não apenas é possível, como ainda há um enorme potencial a ser explorado.

De toda sorte, há exceções ao sigilo fiscal. Dentre outras, não são vedadas as informações relativas às inscrições na Dívida Ativa da Fazenda Pública (BRASIL, 1966), que são de competência da PGFN<sup>4</sup>, além de informações do Cadastro Nacional de Pessoas Jurídicas – CNPJ (BRASIL, 2016), que é de competência da RFB<sup>5</sup>. Ou

---

<sup>4</sup> PGFN – Procuradoria Geral da Fazenda Nacional.

<sup>5</sup> RFB – Secretaria Especial da Receita Federal do Brasil (ou simplesmente Receita Federal).

seja, são dados públicos. E é justamente em cima deles que será realizado o presente trabalho. Cruzaremos as informações dos débitos inscritos em DAU com a base de dados cadastrais do CNPJ, e as analisaremos.

Conforme já comentado, o enfoque desse trabalho serão os débitos parcelados. Infelizmente não abordaremos o mais importante instituto de extinção da dívida: o pagamento. Seria muito mais enriquecedor poder fazer um levantamento de regularização de dívida tributária considerando todos os seus institutos, pagamento, parcelamento e eventuais outros. Mas os dados obtidos não nos permitem. Isso porque após o pagamento, assim como após qualquer outro ato de extinção da dívida, a PGFN exclui a informação do débito de sua base de dados pública. Como o parcelamento ativo não extingue o débito, apenas o deixa com sua exigibilidade suspensa, ele permanece na referida base de dados.

Ressalte-se que a PGFN já utiliza *machine learning* para otimizar a cobrança da dívida ativa da União. A ferramenta ajuda a prever a probabilidade de recuperação dos débitos, o que permite ao órgão poder classificá-los e traçar estratégias de cobrança apropriadas para cada classe. (INSTITUTO INNOVARE, 2020).

E essa é uma prática cada vez mais adotada em vários setores. Consultando-se sites de bancos, financeiras, empresas de cobrança, etc, muitos afirmam que também já estão utilizando algoritmos de *big data/machine learning*. Isso lhes permite fazer análises preditivas, conhecer melhor o cliente, precificar melhor a carteira, calcular riscos de créditos, reduzir inadimplência, reduzir fraude, gerenciar melhor os riscos, etc.

De forma alguma o presente trabalho se propõe a ter a importância e a utilidade dos citados exemplos. Primeiramente, porque o autor é um principiante na área. Depois, porque é um trabalho sem qualquer vínculo oficial junto aos referidos órgãos. Além disso, os dados públicos, que são a nossa matéria-prima, apesar de volumosos, são restritos em termos de diversidade de informações que seriam extremamente úteis para o problema proposto desse trabalho. Assim, diante das limitações, a nossa proposta é fazer o máximo possível com os dados obtidos.

## 1.2. O problema proposto

De forma bem objetiva, os problemas propostos são:

- 1) Qual é o perfil dos débitos parcelados na dívida ativa da União?
- 2) É possível prever se um débito será parcelado com os dados obtidos?

Para termos uma melhor visualização dos problemas propostos faremos um levantamento utilizando-se da técnica dos 5-Ws (*University of Nebraska-Lincoln*, 2022).

### **(Why?) Por que esse problema é importante?**

Trazemos um caso relativo à área da administração tributária, em especial, de seu macroprocesso de cobrança. Tentaremos entender um pouco melhor qual é o perfil dos débitos e dos contribuintes que procuram regularizar seus débitos inscritos em Dívida Ativa da União através de parcelamento.

Apesar de não podermos analisar o pagamento, o parcelamento é uma das principais formas de regularização. Em seu relatório anual, publicado em 2022 e referente a 2021, a PGFN informou, em relação à sua arrecadação por estratégia, que 39,1% (R\$ 12,4 bilhões de um total de R\$ 31,7 bilhões) corresponde a benefício fiscal. Como veremos mais adiante, se refere a parcelamento. É a maior rubrica. (PGFN, 2022, p. 13).

Nesse mesmo relatório, a PGFN cita a consolidação da transação tributária. Trata-se de diversos programas de regularização tributária, dentre eles o parcelamento, iniciados em 2020, destinados a devedores com reduzida capacidade de pagamento ou que tiveram suas finanças prejudicadas pela pandemia. Somente em 2021 foram 876 mil acordos, referente a um montante de R\$ 221 bilhões. Até o

presente momento, alguns destes programas continuam com pedidos de adesões em aberto.

Fugindo da área tributária, é muito comum ouvirmos falar que o parcelamento faz parte da cultura do brasileiro. Segundo Júnior (2017), uma das origens seria a época de hiperinflação, quando se obter um parcelamento a juros baixos era uma forma de garantir um bom desconto.

Uma pesquisa realizada pelo SPC em 2015, citada pela FECOMERCIO-SP (2015), levantou que 79% dos consumidores brasileiros costumam parcelar suas compras.

Assim, o parcelamento é um fenômeno importante, e merece ser estudado.

### **(Who?) Quem iremos analisar? Qual a origem dos dados?**

Iremos analisar os débitos dos contribuintes inscritos em DAU. Os dados provêm de dois órgãos governamentais: PGFN e RFB. O primeiro fornecerá a base de dados dos débitos de pessoas físicas e jurídicas, enquanto o segundo fornecerá a base cadastral de pessoa jurídica. Faremos o cruzamento destas duas bases e analisaremos.

### **(What?): Quais os objetivos com essa análise? O que iremos analisar?**

Em atenção às duas perguntas iniciais do problema proposto, os objetivos deste trabalho são: 1) fazer uma análise exploratória dos dados e, 2) criar um modelo de *machine learning* que consiga fazer a predição se um determinado débito será ou não parcelado.

Isso poderá permitir prever, entre outras possibilidades, qual será o nível de adesão a um parcelamento, e qual será o valor arrecadado. Ao se identificar o perfil do contribuinte em relação ao parcelamento, permitirá ao órgão classificá-los e tomar ações específicas, personalizadas.

Permitirá ainda descobrir o perfil dos contribuintes da outra classe: aqueles que não parcelam, que, como veremos, é amplamente majoritário. Isso poderá gerar insights que permitam repensar as estratégias de cobrança, aperfeiçoamento de sistemas e até políticas de parcelamento.

Serão basicamente dois grupos de dados analisados: débito e cadastro. O primeiro contém informações sobre valor, tributo, situação, etc. O segundo contém informações sobre atividade econômica, natureza jurídica, porte, etc.

**(Where?): Trata dos aspectos geográficos e logísticos de sua análise.**

As bases de dados originais, bem como a análise realizada, contemplam 100% dos débitos inscritos em dívida ativa da União, de todas as unidades federativas do país.

**(When?): Qual o período está sendo analisado?**

O período analisado é o 1º trimestre/2022, que é a última base de dados disponibilizada pela PGFN.

## 2. COLETA DE DADOS

Os dados originais foram coletados na internet, nos sites da RFB e PGFN, sendo os dados cadastrais da base CNPJ no primeiro, enquanto os dados dos débitos no segundo. Foram baixadas 3 bases de dados:

- 1) **CNPJ Empresa** (CNPJ 8 dígitos) – Contém os dados cadastrais gerais da empresa, tais como CNPJ, natureza jurídica, porte, razão social, capital social, e outros.

Link: <https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/cadastros/consultas/dados-publicos-cnpj>

A base de dados está dividida em 10 arquivos, cujo texto dos links de cada um é Dados Abertos CNPJ EMPRESA XX (onde XX varia de 01 a 10).

Data do download: 21/02/2022

Tamanho: 871 MB

- 2) **CNPJ Estabelecimento** (CNPJ 14 dígitos) – Contém os dados cadastrais específicos de um determinado estabelecimento da empresa, tais como CNPJ, matriz/filial, situação cadastral, data de início de atividade, CNAE fiscal (código da atividade econômica), endereço, e outros.

Link: <https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/cadastros/consultas/dados-publicos-cnpj>

A base de dados está dividida em 10 arquivos, cujo texto dos links de cada um é Dados Abertos CNPJ ESTABELECIMENTO XX (onde XX varia de 01 a 10).

Data do download: 21/02/2022

Tamanho: 3,48 GB

- 3) **Débitos inscritos em Dívida Ativa da União no 1º trimestre/2022** – Contém os dados dos débitos no 1º trimestre/2022, tais como CPF/CNPJ, nome do devedor, número de inscrição, situação da inscrição, receita (tributo), data da inscrição, valor, e outros.

Link: [https://dadosabertos.pgfn.gov.br/2022\\_trimestre\\_01/Dados\\_abertos\\_Nao\\_Previdenciario.zip](https://dadosabertos.pgfn.gov.br/2022_trimestre_01/Dados_abertos_Nao_Previdenciario.zip)

Data do download: 22/05/2022

Tamanho: 772 MB

A seguir, detalha-se como se deu a construção do *dataste* final, que é onde será aplicado o algoritmo de *machine learning*.

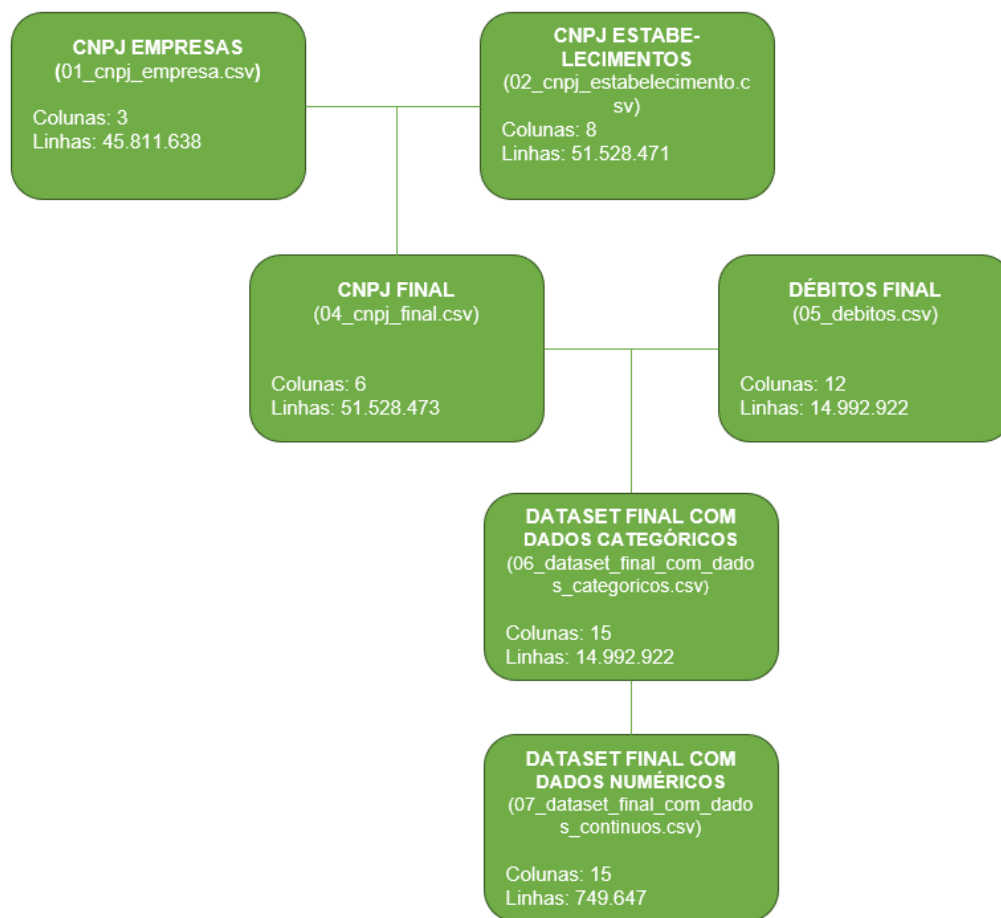
Primeiramente, os dois primeiros arquivos (CNPJ Empresa e CNPJ Estabelecimento) foram fundidos em um só utilizando-se o campo CNPJ de 8 dígitos (que é comum a ambos) como campo-chave. Nesse processo os campos (colunas) que não serão utilizados neste trabalho foram descartados a fim de reduzir o custo computacional. A esta nova base de dados deu-se o nome de “CNPJ FINAL”.

Em seguida, trabalhou-se a base de débitos, criando-se alguns campos novos, e descartando outros que eram desinteressantes para o trabalho. A esta nova base de dados deu-se o nome de “DÉBITOS FINAL”.

Depois, fundiu-se as duas bases anteriormente criadas para formar o DATASET FINAL com dados categóricos (será utilizado na análise exploratória), que, por sua vez, deu origem ao DATASET FINAL com dados contínuos (será utilizado em *machine learning*). Ver Figura 2.

**Figura 2 – Relacionamento entre os datastes para construção do dataset final**





Fonte: elaboração própria.

Conforme ilustrado na Figura 2, o DATASET FINAL possui 15 colunas (também chamadas de campos ou features) e 749.647 linhas (também chamadas de registros). No Quadro 1 detalha-se quais são esses campos, o *dataset* de origem, além do tipo de variável<sup>6</sup>.

QUADRO 1				
DATASET FINAL – COLUNAS/CAMPOS				
Num	Dataset de origem	Nome da coluna/campo	Descrição	Tipo de variável
1	3	TIPO_PESSOA	Pessoa física ou jurídica? (PF ou PJ)	A

<sup>6</sup> Os tipos de variáveis serão conceituados no capítulo 3, alínea g.

2	3	SITUACAO_INSCRICAO **	Situação da inscrição (ativa ajuizada, ativa em cobrança, ativa parcelada, etc)	A
3	3	TIPO_SITUACAO_INSCRICAO **	Agrupamento do item anterior (em cobrança, em negociação, benefício fiscal, suspenso por decisão judicial, garantia)	A
4	3	RECEITA_PRINCIPAL	Tributo (IRPF, IRPJ, CSLL, PIS, COFINS, Simples Nacional, etc)	A
5	3	INDICADOR_AJUIZADO	O débito estava ajuizado? (S ou N)	A
6	3	HA_OUTROS_DEBITOS *	Há outros débitos inscritos? (S ou N)	A
7	3	INSC_>1_ANO *	Débito inscrito há mais de um ano? (S ou N)	A
8	3	VALOR_CONSOLIDADO	Valor do débito	B
9	3	VALOR_FAIXA *	Faixa de valor do débito (de F1 a F10)	C
10	2	SIT_CADAST	Situação cadastral do CNPJ (ativa, suspensa, inapta, baixada)	A
11	2	CNAE_PRINC	Código da atividade econômica (serviço, comércio, indústria, agricultura, etc)	A
12	1	NAT_JURID	Código da Natureza jurídica (empresário individual, sociedade empresária limitada, etc)	A
13	1	PORTE	Porte (microempresa, empresa de pequeno porte, etc.)	A
14	2	ATIV_>10_ANOS *	Empresa com mais de 10 anos de atividade? (S ou N)	A
15	3	PARCELADO * ⊗	Situação final do débito. O débito estava parcelado no 1º trimestre/2022? (S ou N)	A

Fonte: elaboração própria.

1 - CNPJ Empresa

2 - CNPJ Estabelecimento

3 - Débitos inscritos em Dívida Ativa da União no 1º trimestre/2022

A – Variável categórica nominal

B – Variável quantitativa contínua

C – Variável quantitativa discreta

\* Campos criados através de enriquecimento/feature engineering (não são dados originais)

\*\* Campos utilizados apenas na análise exploratória. Foram excluídos da aplicação em machine learning.

⊗ Campo alvo/target – campo que se deseja prever através de *machine learning*

### 3. PROCESSAMENTO/TRATAMENTO DE DADOS

No capítulo anterior informou-se a origem dos dados, o relacionamento entre as diversas bases de dados coletadas para a formação do *dataset* final, além de uma apresentação dos seus campos. No presente capítulo, detalharemos como esse processamento foi realizado. Os dados brutos coletados ainda não estão prontos para serem submetidos aos algoritmos de *machine learning*. É necessário seu pré-processamento ou tratamento. Existem diversas técnicas para sua realização. Neste trabalho, adotaremos as seguintes:

- . Limpeza: está relacionada à qualidade dos dados. Consiste em tratar dados ausentes, inconsistentes (ruidosos), irrelevantes, duplicados ou redundantes.

- . Enriquecimento: agregar outra base de dados para criar novos campos que não constavam na base de dados inicial.

- . Codificação: correção/alteração do formato/tipo dos dados para se adequar ao que o algoritmo de machine learning exige.

- . Feature Engineering: em linhas gerais, consiste em técnicas para escolha dos melhores campos (features), como, p. exemplo, utilizando-se de correlação com a variável que se deseja prever. Consiste ainda em criar novos campos, promover mudanças, conversão de dados categóricos em contínuos, etc.

Todo este trabalho foi realizado na plataforma *Jupyter Notebook*<sup>7</sup>, utilizando-se da linguagem *Python*<sup>8</sup> (versão 3.0), cujos arquivos *ipynb*<sup>9</sup> utilizados estão relacionados na Quadro 2, e estão disponibilizados para *download* (ver capítulo 7 - *Links*). A

---

<sup>7</sup> *Jupyter Notebook*: é um ambiente de programação que executa o código apresentado. Suporta diversas linguagens de programação.

<sup>8</sup> *Python*: é uma linguagem de programação das mais utilizadas em *machine learning*.

<sup>9</sup> *ipynb*: é a extensão do arquivo notebook em *Python* gerado pelo *Jupyter Notebook*.

principal biblioteca *Python* utilizada foi a *Pandas*<sup>10</sup>, adotada em todos os notebooks. Diversas outras bibliotecas também foram utilizadas, mas de forma pontual algumas serão oportunamente apresentadas. Ao longo deste trabalho os principais códigos de programação na referida linguagem serão exibidos logo abaixo da ação tomada, conforme quadro abaixo (para não ficar exaustivo, evitaremos repetições dos mesmos comandos):

In:	Linha de exemplo de entrada na linguagem Python
Out:	Linha de exemplo de saída na linguagem Python

O Quadro 2 está dividida em 3 colunas: arquivo de entrada, processamento e arquivo de saída. No arquivo de entrada especificamos quais arquivos de base de dados foram utilizados no *notebook* correspondente, que gerará um novo arquivo de saída, após seu processamento. Observe que nos *notebooks* 01, 02, e 05 os arquivos de entrada são as bases de dados brutas coletadas na internet (conforme detalhado no capítulo 2), enquanto nos demais os arquivos de entrada são arquivos de saída gerados em outros *notebooks*.

<b>QUADRO 2</b> <b>ARQUIVOS E NOTEBOOKS UTILIZADOS</b>		
<b>ARQUIVO DE ENTRADA</b>	<b>PROCESSAMENTO (NOTEBOOK PYTHON)</b>	<b>ARQUIVO DE SAÍDA</b>
Baixado internet	01_cnpj_empresas.ipynb	01_cnpj_empresa.csv
Baixado internet	02_cnpj_estabelecimentos.ipynb	02_cnpj_estabelecimento.csv
01_cnpj_empresa.csv; 02_cnpj_estabelecimento.csv	03_merge_empresa_estabelecimento.ipynb	03_merge_empresa_estabelecimento.csv
03_merge_empresa_estabelecimento.csv	04_cnpj_final.ipynb	04_cnpj_final.csv
Baixado internet	05_debitos.ipynb	05_debitos.csv

---

<sup>10</sup> *Pandas*: é uma biblioteca *Python* com diversos recursos para manipulação e análise de dados.

04_cnpj_final.csv; 05_debitos.csv	06_merge_debitos_cnpj.ipynb	06_dataset_final_com_dados_categoricos.csv
06_dataset_final_com_dados_categoricos.csv	07_conv_dados_contínuos.ipynb	07_dataset_final_com_dados_contínuos.csv
06_dataset_final_com_dados_categoricos.csv; 07_dataset_final_com_dados_contínuos.csv	08_analise_exploratoria.ypynb	N/A
07_dataset_final_com_dados_contínuos.csv	09_machine_learning.ipynb	N/A

Fonte: elaboração própria.

Obs.: Os notebooks de números 01 a 07 se referem ao processamento/tratamento dos dados e serão detalhados neste capítulo 3.

Obs.: Os notebooks de números 08, 09 não geraram arquivos de saída, e sim gráficos e informações finais que serão detalhadas nos capítulos 4 e 5.

A seguir detalharemos os notebooks referentes ao processamento/tratamento dos dados (notebooks 01 a 07).

#### a) **Notebook 01 cnpj\_empresas.ipynb**

Este notebook faz a importação da base de dados CNPJ Empresas, baixados da internet, que está dividido em 10 arquivos (df0 a df9). Faz a concatenação de todos estes em um único dataset e, ao final, exporta o arquivo como '01\_cnpj\_empresa.csv'.

##### ▪ **Importando a referida base de dados:**

Este comando é feito 10 vezes, um para cada arquivo importado (abaixo, representamos apenas um). Observe que na importação já selecionamos as colunas desejadas, de índices 0, 2 e 5, que se referem a 'CNPJ\_BASICO', 'NAT\_JURID' e 'PORTE'. As demais colunas foram descartadas por entendermos serem irrelevantes.

In:	<pre>import pandas as pd df0 = pd.read_csv("CNPJ_empresa/K3241.K03200Y0.D10510.EMPRESV.zip", sep= ';', encoding='ISO-8859-1', usecols=[0, 2, 5], names=['CNPJ_BASICO', 'NAT_JURID', 'PORTE'], dtype={'CNPJ_BASICO':object, 'NAT_JURID':object, 'PORTE':object})</pre>
Out:	

- **Concatenando os 10 datasets importados em um único ('df\_princ'):**

In:	<code>df_princ = pd.concat([df0, df1, df2, df3, df4, df5, df6, df7, df8, df9])</code>
Out:	

- **Consultando a quantidade de linhas/registros no dataset 'df\_princ':**

In:	<code>df_princ.count()</code>
Out:	<pre> CNPJ_BASICO    45811638 NAT_JURID      45811638 PORTE          45752207 dtype: int64 </pre>

Observa-se que o dataset possui 45.811.638 linhas, sendo que o campo 'PORTE' não possui todas as linhas preenchidas, ou seja, estão ausentes. Problemas de dados ausentes serão resolvidos no notebook '06\_merge\_debitos\_cnpj.ipynb'.

Por fim, exporta-se o dataset `df_princ` como arquivo `01_cnpj_empresa.csv`.

In:	<code>df_princ.to_csv("01_cnpj_empresa.csv")</code>
Out:	

## b) **Notebook 02\_cnpj\_estabelecimentos.ipynb**

Este notebook faz a importação da base de dados CNPJ Estabelecimentos, baixados da internet, que também está dividido em 10 arquivos (df0 a df9). Faz a concatenação de todos estes em um único dataset e, ao final, exporta o arquivo como `02_cnpj_estabelecimento.csv`.

- **Importando-se a base de dados:**

Assim como no notebook anterior, este comando é feito 10 vezes, um para cada arquivo importado. Da mesma forma, observe que na importação já selecionamos as colunas desejadas, de índices 0, 1, 2, 4, 5, 10, 11, 19, que se referem a 'CNPJ\_BASICO', 'CNPJ\_ORDEM', 'CNPJ\_DV', 'NOME\_FANTASIA', 'SIT\_CADAST',

'DATA\_INICIO\_ATIV', 'CNAE\_PRINC', 'UF'. As demais colunas foram descartadas por entendermos serem irrelevantes pra este trabalho.

In:	<pre>import pandas as pd df0 = pd.read_csv("CNPJ_estabelecimento/K3241.K03200Y0.D20212.ESTA- BELE.zip", sep= ';', encoding='ISO-8859-1', usecols=[0, 1, 2, 4, 5, 10, 11, 19], names=['CNPJ_BASICO', 'CNPJ_ORDEM', 'CNPJ_DV', 'NOME_FANTA- SIA', 'SIT_CADAST', 'DATA_INICIO_ATIV', 'CNAE_PRINC', 'UF'], dtype={"CNPJ_BASICO": "category", "CNPJ_ORDEM": "category", "CNPJ_DV": "category"})</pre>
Out:	

- **Concatenando-se os 10 datasets importados em um único ('df\_cnpj\_estab):**

In:	<pre>df_cnpj_estab = pd.concat([df0, df1, df2, df3, df4, df5, df6, df7, df8, df9])</pre>
Out:	

- **Consultando-se a quantidade de linhas/registros no dataset 'df\_princ':**

In:	<pre>df_cnpj_estab.count()</pre>
Out:	<pre>CNPJ_BASICO      51528471 CNPJ_ORDEM       51528471 CNPJ_DV          51528471 NOME_FANTASIA    30754574 SIT_CADAST       51528471 DATA_INICIO_ATIV 51528471 CNAE_PRINC       51528471 UF              51528471 dtype: int64</pre>

Observa-se que o dataset possui 8 colunas e 51.528.471 linhas, sendo que o campo 'NOME\_FANTASIA' não possui todas as linhas preenchidas, ou seja, estão ausentes. Como esta coluna será descartada antes do dataset final, não será um problema.

Por fim, exporta-se o dataset 'df\_cnpj\_estab' como arquivo '02\_cnpj\_estabelecimento.csv'.

In:	<pre>df_princ.to_csv("02_cnpj_estabelecimento.csv")</pre>
Out:	

c) **Notebook 03\_merge\_empresa\_estabelecimento.ipynb**

Este notebook faz a fusão dos arquivos criados nos itens a e b. Ao final, exporta o arquivo como 03\_merge\_empresa\_estabelecimento.csv.

▪ **Fundindo-se/mesclando-se as duas bases utilizando-se do campo 'CNPJ\_BASICO', que é comum a ambos:**

Aqui cabe uma análise de **dados duplicados**. A razão de ter mais registros é porque aquele possui dados referentes à matriz e de todas as filiais de uma mesma empresa, enquanto este possui apenas um registro por empresa. Assim, como a base determinante é 'CNPJ\_estabelecimento' (how = 'left'), os registros que serão importados da base CNPJ\_empresas (colunas 'NAT\_JURID' e 'PORTE') ficarão duplicados. Entendemos que isso não comprometerá o trabalho porque os débitos estão vinculados aos CNPJ's dos estabelecimentos e cada um tem sua particularidade. Afinal, pelo menos teoricamente, é possível que uma matriz seja uma boa pagadora e uma filial não. O algoritmo de ML tratará cada estabelecimento isoladamente, apesar de compartilharem duas informações ('NAT\_JURID' e 'PORTE') idênticas.

In:	<code>df_cnpj_merge = pd.merge(df_estab, df_empresa, how = 'left', on = 'CNPJ_BASICO')</code>
Out:	

O resultado é um novo dataset com 51.528.473 registros e 8 colunas.

In:	<code>df_cnpj_merge.count()</code>
Out:	<pre> CNPJ_BASICO      51528473 CNPJ_ORDEM       51528473 CNPJ_DV          51528473 SIT_CADAST       51528473 DATA_INICIO_ATIV 51528473 CNAE_PRINC       51528473 NAT_JURID        48505659 PORTE            48446228 dtype: int64 </pre>

d) **Notebook 04\_cnpj\_final.ipynb**



Aqui faremos alguns ajustes e enriquecimentos no dataset criado no item anterior, como criação de novos campos, e exclusão de outros. No final, exportaremos o arquivo como '04\_cnpj\_final.csv'.

- **Redução de valores dos campos 'CNAE\_PRINC':**

Este campo possui 1343 valores (códigos) diferentes. Isso pode dificultar a análise exploratória e aplicação adequada dos algoritmos de *machine learning*. Observa-se que seu padrão de valores é um código composto por 7 dígitos. Analisando-se a estrutura desse código, observa-se que se o reduzirmos para 2 dígitos, será suficiente para identificarmos satisfatoriamente sua atividade econômica, dentro dos objetivos deste trabalho.

In:	# AJUSTANDO CNAE_PRINC df_cnpj_final['CNAE_PRINC'] = df_cnpj_final['CNAE_PRINC'].str[:2]
Out:	

- **Criação do campo 'ATIV\_>10\_ANOS':**

A ideia é converter um dado do tipo data (que tem pouca utilidade em machine learning) em um dado do tipo booleano<sup>11</sup>, cujos valores são S (sim) ou N (não). Criado a partir do campo original 'DATA\_INICIO\_ATIV' através das seguintes etapas: 1) conversão de seu formato para 'datetime64[ns]'; 2) em seguida, criação de um campo temporário de nome 'ANO\_INICIO\_ATIV', que converte a data em ano; 3) por fim, criação do campo 'ATIV\_>10\_ANOS' onde anos anteriores ou iguais a 2010 recebem valor 'S', caso contrário, 'N'. Escolheu-se o ano 2010 para corte dos 10 anos porque a base de dados dos débitos é do início de 2021. Ressalte-se que foi necessária a importação da biblioteca *NUMPY*<sup>12</sup> para esta última etapa.

---

<sup>11</sup> Booleano: é um tipo de dado que possui dois valores, que podem ser considerados como 0 ou 1, falso ou verdadeiro

<sup>12</sup> Numpy: é uma biblioteca para a linguagem Python com uma grande coleção de funções matemáticas.

In:	<pre># CRIANDO NOVO CAMPO COM ANO ATIVIDADE MAIOR DO QUE 10 ANOS df_cnpj_final['DATA_INICIO_ATIV'] = pd.to_datetime(df_cnpj_final['DATA_INICIO_ATIV'], format="%Y/%m/%d", errors = 'coerce')  df_cnpj_final['ANO_INICIO_ATIV'] = df_cnpj_final['DATA_INICIO_ATIV'].dt.year  import numpy as np df_cnpj_final['ATIV_&gt;10_ANOS'] = np.where(df_cnpj_final.ANO_INICIO_ATIV&lt;=2010, 'S', 'N')</pre>
Out:	

#### ▪ Criação de um novo campo CNPJ:

Até este momento, o CNPJ do estabelecimento está dividido em 3 colunas: CNPJ\_BASICO (8 caracteres), CNPJ\_ORDEM (4 caracteres), CNPJ\_DV (2 caracteres). Como será necessário fundir este dataset com o dataset dos débitos, e o campo CNPJ neste possui 14 caracteres, é necessário fundir as referidas 3 colunas em uma, para que os campos fiquem iguais. Além disso, será necessário incluir as pontuações do padrão CNPJ (99.999.999/9999-99). Assim, fizemos o seguinte: 1) criamos um novo campo temporário CNPJ\_BASICO\_2, onde incluiu-se as pontuações no campo de 8 caracteres; 2) Em seguida, criamos mais um campo CNPJ\_BASICO\_3, juntando-se as 3 colunas, devidamente separadas por "/" e "-".

In:	<pre># CRIANDO NOVO CAMPO CNPJ COM PONTUAÇÃO df_cnpj_final['CNPJ_BASICO_2'] = df_cnpj_final['CNPJ_BASICO'].str[0:2]+"."+df_cnpj_final['CNPJ_BASICO'].str[2:5]+"."+df_cnpj_final['CNPJ_BASICO'].str[5:9]  df_cnpj_final['CNPJ_BASICO_3'] = df_cnpj_final['CNPJ_BASICO_2'].str[0:]+"/"+df_cnpj_final['CNPJ_ORDEM'].str[0:]+-"+df_cnpj_final['CNPJ_DV'].str[0:]</pre>
Out:	

Por fim, excluimos todos os campos desnecessários, reordenamos as colunas e renomeamos o título da coluna 'CNPJ\_BASICO\_3' para 'CNPJ\_BASICO'.

In:	<pre># EXCLUINDO COLUNAS DESNECESSÁRIAS df_cnpj_final = df_cnpj_final.drop(columns=['DATA_INICIO_ATIV', 'ANO_INICIO_ATIV'])  df_cnpj_final = df_cnpj_final.drop(columns=['CNPJ_BASICO', 'CNPJ_ORDEM', 'CNPJ_DV', 'CNPJ_BASICO_2'])</pre>
-----	--

	<pre># REORDENANDO COLUNAS df_cnpj_final = df_cnpj_final[['CNPJ_BASICO_3', 'SIT_CADAST', 'CNAE_PRINC', 'NAT_JURID', 'PORTE', 'ATIV_&gt;10_ANOS']]  # RENOMEANDO COLUNA df_cnpj_final = df_cnpj_final.rename(columns = {'CNPJ_BASICO_3': 'CNPJ_BASICO'})</pre>
Out:	

O resultado desse ajuste é um dataset com 51.528.473 registros (não houve mudança em relação ao notebook anterior) e 6 colunas (2 colunas a menos, que já fundimos três colunas em uma, e substituímos outra).

In:	<code>df_cnpj_merge.count()</code>
Out:	<pre>&lt;class 'pandas.core.frame.DataFrame'&gt; RangeIndex: 51528473 entries, 0 to 51528472 Data columns (total 6 columns): #   Column      Dtype ---  - 0    CNPJ_BASICO  object 1    SIT_CADAST  category 2    CNAE_PRINC  object 3    NAT_JURID   category 4    PORTE       category 5    ATIV_&gt;10_ANOS object dtypes: category(3), object(3) memory usage: 1.3+ GB</pre>

#### e) Notebook 05 debitos.ipynb

Os notebooks anteriores foram referentes à base de dados CNPJ. Neste trabalho, trataremos a base de dados de débitos. Primeiramente, importaremos o arquivo original disponibilizado pela PGFN, referente ao 1º trimestre/2022, onde faremos diversos ajustes, os quais detalharemos a seguir. Aqui criaremos a variável mais importante deste trabalho, que é a variável PARCELADO. Por fim, exportaremos o dataset final como '05\_debitos.csv'.

#### ▪ **Importando base de dados de débitos:**

O arquivo original é baixado compactado, e possui diversos arquivos, separados por unidade federativa (ou seja, 27 arquivos). A biblioteca zipfile importa todos os

arquivos sem a necessidade de descompactação. Neste momento, já é realizada a seleção das colunas necessárias. Em seguida, concatena-se todos os arquivos importados. Por fim, reseta-se o índice das linhas, pois a concatenação gerou índices duplicados.

In:	<pre># IMPORTANDO DATASET DE DEBITOS DA PGFN DO 1º TRIMESTRE/2022 # Importando dataset de uma pasta compactada com vários arquivos  import pandas as pd from zipfile import ZipFile  with ZipFile ("pgfn\Dados_abertos_Nao_Previdenciario_1trim2022.zip") as zipfiles:     filelist = zipfiles.namelist()[1:]      data = [pd.read_csv(zipfiles.open(file_name), usecols=[0, 1, 2, 6, 7, 8, 9, 10, 11, 12], dtype={"NUMERO_INSCRICAO":"category"}, sep= ';', encoding='unicode_escape') for file_name in filelist]  # Concatenando-se os arquivos importados df = pd.concat(data)  # Resetando o índice das linhas df = df.reset_index(drop=True)</pre>
Out:	

- **Removendo-se o tipo de devedor diferente de principal:**

Cada débito é único, mas podem ter vários devedores, que são classificados em principal, solidário e corresponsável, que estão registrados em linhas diferentes, como se fossem outros débitos. Ou seja, estes diversos devedores acabam **duplicando** os registros dos débitos. Como sempre há pelo menos um devedor principal, os demais tipos serão removidos.

In:	<pre># REMOVENDO TODAS AS LINHAS DA COLUNA 'TIPO_DEVEDOR' QUE SÃO DIFERENTES DE 'PRINCIPAL' df = df.drop(df.loc[df['TIPO_DEVEDOR']!='PRINCIPAL'].index)</pre>
Out:	

- **Substituindo-se os valores do campo 'TIPO\_PESSOA' por PF e PJ:**

Os valores desse campo encontram-se por extenso, ou seja, pessoa física e pessoa jurídica. Como é desnecessário, e a fim de reduzir o custo computacional, serão substituídos por suas famosas siglas.

In:	# SUBSTITUINDO O CAMPO TIPO_PESSOA POR PJ E PF df['TIPO_PESSOA'] = df['TIPO_PESSOA'].replace(['Pessoa jurídica', 'Pessoa física'], ['PJ', 'PF'])
Out:	

#### ▪ Criação do campo 'INSC\_>1\_ANO':

Em um dos *insights* desse trabalho levantou-se a hipótese de que quanto mais antigo o débito, menor é a probabilidade de sua regularização/parcelamento. Assim, criamos um novo campo booleano que assinala 'S' quando um débito for maior que um ano e 'N' quando não. Para isso, usamos o campo original 'DATA\_INSCRICAO' como referência, e criamos um campo temporário 'ANO\_INSCRICAO'. Ressalte-se que o período base de referência é 1º trimestre/2022. Assim, considerou-se maiores do que um ano todos os débitos inscritos até 2019.

In:	# CRIANDO NOVO CAMPO CALCULADO DO TEMPO DE INSCRIÇÃO DA DÍVIDA (MAIOR QUE 1 ANO) UTILIZANDO DATA_INSCRICAO import numpy as np  df['ANO_INSCRICAO'] = df['DATA_INSCRICAO'].dt.year df['INSC_>1_ANO'] = np.where(df.ANO_INSCRICAO<=2019, 'S', 'N')
Out:	

#### ▪ Criação do campo 'HA\_OUTROS\_DEBITOS':

Em outro *insight* levantou-se a hipótese de que quanto maior a quantidade de débitos, mais improvável é a sua chance de regularização. Para tanto, criou-se um campo temporário 'CONTADOR\_NOME' que conta quantos registros há para um determinado CPF ou CNPJ. Em seguida, criou-se o novo campo 'HA\_OUTROS\_DEBITOS' que assinala 'S' quando o contador for maior do que 1, e 'N' quando não.

In:	# CRIANDO CAMPO QUE VERIFICA SE HÁ MAIS DE UM DÉBITO (MAIS DE UMA LINHA PARA O MESMO CPF_CNPJ) df['CONTADOR_NOME'] = df.groupby('CPF_CNPJ')['CPF_CNPJ'].transform('count')
-----	---

	<pre>import numpy as np df['HA_OUTROS_DEBITOS'] = np.where(df.CONTADOR_NOME&gt;1, 'S', 'N')</pre>
Out:	

#### ▪ Criação do campo 'VALOR\_FAIXA':

De modo similar ao item anterior, levantamos a hipótese de que quanto maior o valor da dívida, menos provável é a sua regularização. O campo original 'VALOR\_CONSOLIDADO' informa o valor exato do débito. No entanto, entendemos que criar faixas de valores (de 1 a 10, ver TABELA 1) tornaria essa informação analiticamente mais relevante. Assim, com base no referido campo original, criou-se campo 'VALOR\_FAIXA'. Manteremos os dois campos no dataset, pois ambos serão úteis na análise exploratória e machine learning.

TABELA 1	
FAIXA DE VALORES CAMPO 'VALOR_FAIXA'	
FAIXA	VALOR R\$
1	Até 1.000,00
2	De 1.000,01 a 5.000,00
3	De 5.000,01 a 10.000,00
4	De 10.000,01 a 20.000,00
5	De 20.000,01 a 50.000,00
6	De 50.000,01 a 100.000,00
7	De 100.000,01 a 500.000,00
8	De 500.000,01 a 1.000.000,00
9	De 1.000.000,01 a 10.000.000,00
10	Acima de 10.000.000,01

Fonte: elaboração própria.

In:	<pre># CRIANDO NOVO CAMPO CALCULADO DA FAIXA DE VALOR  import sys  df['VALOR_FAIXA']=pd.cut(      df['VALOR_CONSOLIDADO'],      bins=[0, 1000, 5000, 10000, 20000, 50000, 100000, 500000, 1000000, 10000000, sys.maxsize],      labels=['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']</pre>
-----	--

	)
Out:	

#### ▪ Limpando o campo 'RECEITA\_PRINCIPAL':

Este campo se refere ao nome do tributo. No entanto, está bastante extenso, sem qualquer necessidade, pois está acompanhado do texto “receita da dívida ativa”. Para piorar, está completamente despadronizado, pois algumas vezes está abreviado ou com pontuação diferente. Assim, utilizou-se a função *replace* para remover esse texto e suas diversas variações, de modo a restar apenas o nome irreduzível do tributo.

In:	<pre># LIMPANDO A COLUNA RECEITA PRINCIPAL (REMOVER TEXTO 'Receita da dívida ativa' e variações)  df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('Receita da dívida ativa - ', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace(' - Re- ceita da dívida ativa', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('Rec. dív. ativa - ', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('Rec. dí- vida ativa - ', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('Receita da dív. ativa - ', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.re- place('DIV.ATIVA-', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('DIV ATIVA-', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('R D ATIVA - ', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('R D Ativa - ', '') df["RECEITA_PRINCIPAL"] = df["RECEITA_PRINCIPAL"].str.replace('R D Ativa - ', '')</pre>
Out:	

#### ▪ Reduzindo a quantidade de valores do campo 'RECEITA\_PRINCIPAL':

Ainda no campo 'RECEITA\_PRINCIPAL', verificamos que ele possui mais de 100 variedades de valores (tributos) distintos, a maioria com baixa frequência. Da mesma forma como já procedemos anteriormente para CNAE\_PRINC, reduziremos essa quantidade, pois, como já justificado, pode dificultar a análise exploratória e distorcer os algoritmos de *machine learning*. Assim, identificamos os 12 tributos de maior

frequência – os quais mantivemos seus nomes – e agrupamos os demais sob a rubrica 'OUTROS'.

In:	<pre># REDUÇÃO DE VALORES DO CAMPO 'RECEITA_PRINCIPAL'  def replace(x):     if x != "SIMPLES NACIONAL" and x != "IRPF" and x != "COFINS" and x     != "IRPJ" and x != "CSLL" and x != "PIS" and x != "Multa - CLT" and x !=     "Multa Isolada" and x != "SIMPLES" and x != "SPU" and x != "IRRF" and x     != "SIMP NAC - MEI":         return 'OUTROS'     else:         return str(x)  df['RECEITA_PRINCIPAL'] = df['RECEITA_PRINCIPAL'].apply(replace)</pre>
Out:	

#### ▪ Criação do campo 'PARCELADO':

Esse é o principal campo de todo o *dataset*. É o campo alvo (target) que desejamos que os algoritmos de *machine learning* aprendam a prever. Observando-se o *dataset*, descobriu-se que o valor 'Benefício Fiscal' encontrado no campo 'TIPO\_SITUACAO\_INSCRICAO' se refere aos débitos parcelados, cuja descrição detalhada se encontra em outro campo, SITUACAO\_INSCRICAO (SISPAR é um sistema de parcelamento da PGFN). Assim, criou-se o campo booleano 'PARCELADO' que assinala 'S' quando valor 'Benefício Fiscal' é encontrado no referido campo, e 'N' quando não.

In:	<pre># IDENTIFICANDO VALOR QUE CORRESPONDE AOS DEBITOS PARCELADOS  df.groupby('TIPO_SITUACAO_INSCRICAO')['SITUACAO_INSCRICAO'].value_counts()</pre>
Out:	<pre>TIPO_SITUACAO_INSCRICAO Benefício Fiscal          3452760  SITUACAO_INSCRICAO ATIVA NAO AJUIZAVEL NEGOCIADA NO SISPAR    2644713 ATIVA AJUIZADA NEGOCIADA NO SISPAR         632234 ATIVA AJUIZADA PARCELADA LEI 12996/14       58913 ATIVA AJUIZADA PARC LEI 11941/09 ART 1...   36758 ATIVA AJUIZADA PARC LEI 11941/09 ART 3...   31301 ... (ADAPATADO PARA ENQUADRAMENTO NA TABELA)</pre>
In:	<pre># CRIANDO NOVO CAMPO CALCULADO: PARCELADO (S/N) import numpy as np  df['PARCELADO'] = np.where(df.TIPO_SITUACAO_INSCRICAO=='Benefício Fis- cal', 'S', 'N')</pre>



Out:	
------	--

#### ▪ Excluindo-se colunas desnecessárias:

Após os tratamentos acima, as colunas que foram utilizadas para gerar outras mais apropriadas não são mais necessárias, e já podem ser excluídas a fim de evitar **redundância** de informação.

In:	# EXCLUINDO COLUNAS DESNECESSÁRIAS  df = df.drop(columns=['DATA_INSCRICAO', 'ANO_INSCRICAO', 'CONTADOR_NOME', 'TIPO_DEVEDOR'])
Out:	

#### ▪ Resultado final do Notebook 05\_debitos.ipynb:

O resultado é um *dataset* com 14.992.922 linhas e 12 colunas, exportado como '05\_debitos.csv'.

In:	df.info()
Out:	<pre>&lt;class 'pandas.core.frame.DataFrame'&gt; Int64Index: 14992922 entries, 1 to 19651576 Data columns (total 12 columns): #   Column                                Dtype ---  - 0   CPF_CNPJ                             object 1   TIPO_PESSOA                           object 2   NUMERO_INSCRICAO                      object 3   TIPO_SITUACAO_INSCRICAO               object 4   SITUACAO_INSCRICAO                    object 5   RECEITA_PRINCIPAL                     object 6   INDICADOR_AJUIZADO                    object 7   VALOR_CONSOLIDADO                     float64 8   INSC_&gt;1_ANO                           object 9   HA_OUTROS_DEBITOS                     object 10  VALOR_FAIXA                           category 11  PARCELADO                             object dtypes: category(1), float64(1), object(10) memory usage: 1.4+ GB</pre>

#### f) Notebook 06 merge debitos cnpj.ipynb

Concluídos os *datasets* de CNPJ e débitos, agora os fundiremos. Ao final, exportaremos o arquivo `06_dataset_final_com_dados_categoricos.csv`, que será utilizado para análise exploratória (notebook 8 / capítulo 4).

#### ▪ Importando os arquivos:

Importando arquivos gerados nos 2 itens anteriores.

In:	<pre># IMPORTANDO ARQUIVO 04_cnpj_final.csv df_cnpj_final = pd.read_csv("04_cnpj_final.csv", dtype={"CNPJ_BASICO": "category", "NAT_JURID": "category", "PORTE": "category"})  # IMPORTANDO ARQUIVO 05_debitos.csv df_debitos = pd.read_csv("05_debitos.csv", nrows=1000000)</pre>
Out:	

#### ▪ Realizando a fusão:

Será utilizado o campo 'CNPJ\_BASICO' do *dataset* CNPJ e 'CPF\_CNPJ' do *dataset* débitos como chave, pois possuem exatamente o mesmo dado (já no mesmo formato). Como a função *merge* exige que os campos tenham os mesmos nomes, renomeou-se o primeiro. Conforme já justificado em caso similar anteriormente, determinamos que o *dataset* dos débitos seja predominante ("*how='left'*"), ou seja, os registros do *dataset* CNPJ que não encontrarem correspondência não serão recuperados.

In:	<pre># RENOMEANDO COLUNA df_cnpj_final = df_cnpj_final.rename(columns = {'CNPJ_BASICO': 'CPF_CNPJ'})  # FUSÃO DOS DATASET DÉBITOS E CNPJ_FINAL df_debitos_cnpj = pd.merge(df_debitos, df_cnpj_final, how='left', on='CPF_CNPJ')</pre>
Out:	

#### ▪ Excluindo campos desnecessários:

Uma vez realizadas todas as fusões, os campos 'CPF\_CNPJ', e 'NUMERO\_INSCRICAO' não são mais necessários e serão excluídos. Eles fazem referências a registros únicos, ou seja, não são características genéricas, que podem ser

aplicadas em qualquer débito. Assim, além de não terem utilidade na análise exploratória, comprometerão os algoritmos de machine learning.

In:	# EXCLUINDO COLUNAS DESNECESSÁRIAS df_debitos_cnpj = df_debitos_cnpj.drop(columns=['CPF_CNPJ', 'NUMERO_INSCRICAO'])
Out:	

#### ▪ Informações do dataset final:

Observe que o dataset final possui 15 colunas e 14.992.922 linhas, com dois campos *numéricos* (VALOR\_CONSOLIDADO e VALOR\_FAIXA) e todos os demais do tipo *object* (categórico). Foi exportado como 06\_dataset\_final\_com\_dados\_categoricos.csv.

In:	df_debitos_cnpj.info()
Out:	<pre>&lt;class 'pandas.core.frame.DataFrame'&gt; Int64Index: 14992922 entries, 0 to 14992921 Data columns (total 15 columns): #   Column                Dtype ---  - 0   TIPO_PESSOA            object 1   TIPO_SITUACAO_INSCRICAO object 2   SITUACAO_INSCRICAO     object 3   RECEITA_PRINCIPAL      object 4   INDICADOR_AJUIZADO     object 5   HA_OUTROS_DEBITOS      object 6   INSC_&gt;1_ANO            object 7   VALOR_CONSOLIDADO      float64 8   VALOR_FAIXA            float64 9   SIT_CADAST             object 10  CNAE_PRINC              object 11  NAT_JURID              object 12  PORTE                  object 13  ATIV_&gt;10_ANOS          object 14  PARCELADO              object dtypes: float64(2), object(13) memory usage: 1.8+ GB</pre>

#### g) Notebook 07 conv dados continuos.ipynb

Por conter dados categóricos, o *dataset* do item anterior não poderá ser utilizado nos algoritmos de *machine learning* que iremos adotar, pois os principais algoritmos trabalham apenas com variáveis quantitativas. Assim, neste notebook faremos

sua conversão para dados quantitativos contínuos e discretos. Ao final, exportaremos como "07\_dataset\_final\_com\_dados\_continuos.csv".

Antes, um pouco de teoria. As variáveis (ou dados) dividem-se, basicamente, em quantitativas (ou numéricas) e qualitativas (ou categóricas). As quantitativas, por sua vez, subdividem-se em discretas (números inteiros) ou contínuas (números reais). Já as qualitativas subdividem-se em nominais (quando não é possível ordená-las, p. ex. nomes, cores, sexo) e ordinais (quando é possível ordená-las, p. ex. tamanho: pequeno, médio, grande). (MAYER, 2022).

Os algoritmos de ML exigem dados ordenados. Por óbvio, as variáveis quantitativas podem ser ordenadas, assim como as qualitativas ordinais. Mas como ordenaremos as qualitativas nominais? Esse é um problema recorrente em ML, especialmente nesse trabalho, onde a maioria dos nossos dados são categóricos. De toda sorte, existem técnicas nesse sentido. Uma delas é ordenar uma variável categórica de acordo com sua correlação com a variável alvo. Existem várias bibliotecas de códigos, chamados de *encoders*, que realizam esse trabalho automaticamente. Utilizaremos o *encoder Weight of Evidence* (WOE), que é baseado no Teorema de Bayes. Ela relaciona o valor de um campo com seu resultado em relação à variável alvo (PARCELADO) e cria um número real, passível de ordenação. A título de exemplo, o campo 'CNAE\_PRINC' possui 81 valores diferentes, que, originalmente, não obedecem a qualquer ordenação lógica, já que são apenas códigos de atividade econômica. O *encoder* utilizado cria um novo código ordenando esse campo de acordo com sua influência sobre a variável alvo.

É necessário ainda **normalizar** os dados, ou seja, deixá-los na mesma escala. Por exemplo, um valor da coluna VALOR\_CONSOLIDADO possui valores que vão até bilhões de reais, enquanto há outras colunas que variam entre 0 e 1. Isso poderá induzir o algoritmo a dar mais importância para os maiores valores e comprometer o resultado. Assim, mesmo as variáveis que já são contínuas também serão trabalhadas através de normalização.

Como vimos no item anterior, o arquivo “06\_dataset\_final\_com\_dados\_categoricos.csv” possui quase 15 milhões de linhas. É uma quantidade massiva de dados que dificulta muito o processamento em um simples computador caseiro (alguns notebooks levam horas para serem processados). Assim, teremos que reduzir a quantidade de linhas.

Essa técnica é chamada de **redução de numerosidade**. Ela permite que os dados sejam substituídos ou estimados por representação de dados menores. Existem métodos paramétricos (utiliza-se apenas os parâmetros do modelo ao invés dos dados reais) ou não-paramétricos (agrupamento, amostragens e o uso de histogramas) (GOMES, 2019). No nosso caso, utilizaremos o método de amostragem estratificada, que mantém a proporção das classes da variável PARCELADO.

Após alguns testes optamos por reduzir este *dataset* em 95%. Apesar de drástico, está no limite da capacidade de processamento da máquina utilizada. De qualquer forma, ainda serão 750 mil linhas! Adotamos o algoritmo *StratifiedShuffleSplit* da biblioteca Scikit-learn<sup>13</sup> para realizá-la.

Em síntese, dividimos este notebook em 5 partes:

- 1) Preparo inicial: importação do dataset e redução da quantidade de linhas com amostragem estratificada (*StratifiedShuffleSplit*).
- 2) Normalização das variáveis contínuas: utilizaremos o método *MinMaxScaler*.
- 3) Conversão das variáveis categóricas binárias (que possuem 2 variações de valores): utilizaremos o método *replace* para substituir S e N por 1 e 0.

---

<sup>13</sup> Scikit-learn: é a biblioteca de *machine learning* mais popular para *Python*.

- 4) Conversão das variáveis categóricas multicasse (com muitas variações de valores): utilizaremos o *encoder* Weight of Evidence (WOE).
- 5) Procedimentos finais: ajustes, conferências e exportação do dataset final reduzido e convertido em variáveis quantitativas.

#### ▪ **Importando o dataset:**

Importaremos o dataset gerado no item anterior.

In:	# IMPORTANDO O DATASET df = pd.read_csv("06_dataset_final_com_dados_categoricos.csv")
Out:	

#### ▪ **Reduzindo a amostra de dados (redução de numerosidade):**

In:	# Dividindo o dataset split = StratifiedShuffleSplit(test_size=0.05) for x, y in split.split(df, df['PARCELADO']): df1 = df.iloc[x] # 95% dos dados df2 = df.iloc[y] # 5% dos dados. Utilizar essa.
Out:	
In:	# Resultado da divisão print("dataset original: ",df.shape) print("dataset 95%: ",df1.shape) print("dataset 5% - nossa amostra: ",df2.shape)
Out:	dataset original: (14992922, 15) dataset 95%: (14243275, 15) dataset 5% - nossa amostra: (749647, 15)

#### ▪ **Checando se há valores nulos (NaN):**

Antes de realizarmos as conversões, faz-se necessário checar se há valores nulos no dataset. Com o método *isna* verificamos se existem valores NaN (*Not a Number*). Assim, verificamos que todos os campos que foram originados das bases de dados de CNPJ possuem muitos valores nulos (138.293 para ser mais exato – é uma quantidade considerável: representa 18% do dataset). Observamos que o campo 'TIPO\_PESSOA' possui 138.205 linhas para PF, ou seja, é quase a mesma quantidade. É claro que como esses campos vieram da base CNPJ, que se referem apenas a PJ, não haveria valores correspondentes para os registros PF, por isso existem

esses valores nulos. Não nos resta outra alternativa senão substituir os valores nulos por zero.

In:	# CHECANDO SE HÁ DADOS NULOS / NAN df2.isna().sum()	
Out:	<pre> TIPO_PESSOA                0 TIPO_SITUACAO_INSCRICAO    0 SITUACAO_INSCRICAO         0 RECEITA_PRINCIPAL          0 INDICADOR_AJUIZADO         0 HA_OUTROS_DEBITOS          0 INSC_&gt;1_ANO                0 VALOR_CONSOLIDADO          0 VALOR_FAIXA                0 SIT_CADAST                 138293 CNAE_PRINC                 138293 NAT_JURID                  138305 PORTE                      138305 ATIV_&gt;10_ANOS              138293 PARCELADO                  0 dtype: int64 </pre>	
In:	# SUBSTITUINDO DADOS NULOS / NAN POR ZERO df.fillna(value = 0, inplace = True)	
Out:		

#### ▪ Normalizando variáveis contínuas:

In:	<pre> from sklearn.preprocessing import MinMaxScaler  variaveis_continuas = ['VALOR_FAIXA', 'VALOR_CONSOLIDADO']  scaler = MinMaxScaler()  df2[variaveis_continuas] = scaler.fit_transform(df2[variaveis_continuas]) </pre>	
Out:		

#### ▪ Conversão das variáveis categóricas binárias:

In:	<pre> df2['TIPO_PESSOA'] = df2['TIPO_PESSOA'].replace(['PJ', 'PF'], ['1', '0']) df2['INDICADOR_AJUIZADO'] = df2['INDICADOR_AJUIZADO'].re- place(['SIM', 'NAO'], ['1', '0']) df2['HA_OUTROS_DEBITOS'] = df2['HA_OUTROS_DEBITOS'].re- place(['S', 'N'], ['1', '0']) df2['INSC_&gt;1_ANO'] = df2['INSC_&gt;1_ANO'].replace(['S', 'N'], ['1', '0']) df2['ATIV_&gt;10_ANOS'] = df2['ATIV_&gt;10_ANOS'].replace(['S', 'N'], ['1', '0']) df2['PARCELADO'] = df2['PARCELADO'].replace(['S', 'N'], ['1', '0']) df2.head() </pre>	
Out:		

### ▪ Conversão das variáveis categóricas multiclasse:

Utilizamos este método para os valores das colunas 'TIPO\_SITUACAO\_INSCRICAO', 'SITUACAO\_INSCRICAO', 'RECEITA\_PRINCIPAL', 'SIT\_CADAST', 'CNAE\_PRINC', 'NAT\_JURID', e 'PORTE'. Adicionamos o sufixo COD\_ ao título da coluna alterada. No código abaixo reproduzimos a conversão de apenas um dos campos, já que os demais são exatamente iguais.

In:	<pre> pip install category_encoders from category_encoders import WOEEncoder  y = df2['PARCELADO']  X = df2['TIPO_SITUACAO_INSCRICAO'] encoder = WOEEncoder(cols=['TIPO_SITUACAO_INSCRICAO']).fit(X, y)  df2['COD_TIPO_SIT_INSC'] = encoder.transform(X)  df2[['TIPO_SITUACAO_INSCRICAO', 'COD_TIPO_SIT_INSC']].groupby('TIPO_SITUACAO_INSCRICAO').agg(lambda x: list(set(x))).reset_index() </pre>
Out:	

### ▪ Informações do *dataset* final:

Observe que o *dataset* final possui as mesmas 15 colunas do *dataset* do *notebook* anterior. No entanto, dessa vez, foi reduzido em 95%, resultando em 749.647 linhas. Outra diferença é que enquanto naquele todos os campos estavam como *object* (categórico), neste os convertemos para números inteiros (int64) ou reais (float64).

In:	df2.info()
Out:	<pre> &lt;class 'pandas.core.frame.DataFrame'&gt; Int64Index: 749647 entries, 1419707 to 10318597 Data columns (total 15 columns): #   Column                Non-Null Count  Dtype ---  - 0   TIPO_PESSOA            749647 non-null int64 1   COD_TIPO_SIT_INSC      749647 non-null float64 2   COD_SIT_INSC           749647 non-null float64 3   COD_RECEITA            749647 non-null float64 4   INDICADOR_AJUIZADO     749647 non-null int64 5   HA_OUTROS_DEBITOS      749647 non-null int64 6   INSC_&gt;1_ANO            749647 non-null int64 </pre>



7	VALOR_CONSOLIDADO	749647	non-null	float64
8	VALOR_FAIXA	749647	non-null	float64
9	COD_SIT_CADAST	749647	non-null	float64
10	COD_CNAE_PRINC	749647	non-null	float64
11	COD_NAT_JURID	749647	non-null	float64
12	COD_PORTE	749647	non-null	float64
13	ATIV_>10_ANOS	749647	non-null	int64
14	PARCELADO	749647	non-null	int64

dtypes: float64(9), int64(6)  
memory usage: 107.6 MB

## 4. ANÁLISE E EXPLORAÇÃO DOS DADOS

Passada a etapa de coleta, processamento e tratamento dos dados, devemos realizar sua análise exploratória. Aqui realizaremos sínteses estatísticas e técnicas de visualização para tentar entender melhor os dados.

É importante que essa etapa seja anterior à aplicação de *machine learning*, pois permitirá: avaliar a qualidade dos dados; entender sua natureza; além de propiciar *insights* e observação de tendências que poderão ser úteis na criação do modelo.

### a) PARTE 1 – PREPARO DOS DADOS

#### ▪ Importando as bibliotecas necessárias:

In:	<pre>import pandas as pd import numpy as np import seaborn as sns import matplotlib.pyplot as plt from matplotlib import style style.use('ggplot')</pre>
Out:	

#### ▪ Carregando a base de dados:

Ressalte-se que nesta análise exploratória estamos utilizando o dataset completo, com quase 15 milhões de linhas. O dataset amostral (criado no notebook anterior) será utilizado apenas para aplicação de machine learning.

In:	<code>df=pd.read_csv('06_dataset_final_com_dados_categoricos.csv')</code>
Out:	

#### ▪ Ajustando e analisando tipo de dados:

Observa-se que temos 15 variáveis e 14.992.922 registros no dataset. Quanto ao tipo de variáveis, podemos dividi-las em:

- Numéricas/quantitativas: VALOR\_FAIXA, VALOR\_CONSOLIDADO

-Categóricas/qualitativas: TIPO\_PESSOA, TIPO\_SITUACAO\_INSCRICAO, SITUACAO\_INSCRICAO, RECEITA\_PRINCIPAL, INDICADOR\_AJUIZADO, HA\_OUTROS\_DEBITOS, INSC\_>1\_ANO, SIT\_CADAST, CNAE\_PRINC, NAT\_JURID, PORTE, ATIV\_>10\_ANOS, PARCELADO.

Conforme já exposto, nossa variável alvo (ou target ou resposta) é PARCELADO. Muitas das análises realizadas neste capítulo será justamente a distribuição dos valores (não ou sim) desta variável com as demais.

In:	<pre># CONVERTENDO TIPO DE DADOS PARA OBJECT df["SIT_CADAST"] = df["SIT_CADAST"].astype("object") df["CNAE_PRINC"] = df["CNAE_PRINC"].astype("object") df["NAT_JURID"] = df["NAT_JURID"].astype("object") df["PORTE"] = df["PORTE"].astype("object") df.info()</pre>
Out:	<pre>&lt;class 'pandas.core.frame.DataFrame'&gt; RangeIndex: 14992922 entries, 0 to 14992921 Data columns (total 15 columns): #   Column                                Dtype ---  - 0   TIPO_PESSOA                           object 1   TIPO_SITUACAO_INSCRICAO               object 2   SITUACAO_INSCRICAO                   object 3   RECEITA_PRINCIPAL                     object 4   INDICADOR_AJUIZADO                     object 5   HA_OUTROS_DEBITOS                     object 6   INSC_&gt;1_ANO                           object 7   VALOR_CONSOLIDADO                     float64 8   VALOR_FAIXA                           float64 9   SIT_CADAST                             object 10  CNAE_PRINC                             object 11  NAT_JURID                             object 12  PORTE                                 object 13  ATIV_&gt;10_ANOS                         object 14  PARCELADO                             object dtypes: float64(2), object(13) memory usage: 1.7+ GB</pre>

## b) PARTE 2 – Análise de cada variável individualmente

### ▪ Estatística Descritiva - Variáveis numéricas:

Quando necessário o python retorna números em notação científica<sup>14</sup>, como no resultado abaixo. Isso ocorre quando há números muito grandes ou muito pequenos. É uma forma de compactá-los.

Das informações abaixo, para a coluna VALOR\_CONSOLIDADO, destaca-se o valor médio (*mean*), mediano (50%), e máximo (*max*) de um débito, que são R\$145.173,70, R\$ 7.003,00 e R\$ 19.019.090.000,00, respectivamente.

In:	df.describe()																												
Out:	<table> <tr> <th></th><th>VALOR_CONSOLIDADO</th><th>VALOR_FAIXA</th></tr> <tr> <td>count</td><td>1.499292e+07</td><td>1.499292e+07</td></tr> <tr> <td>mean</td><td>1.451737e+05</td><td>3.581026e+00</td></tr> <tr> <td>std</td><td>9.254073e+06</td><td>1.876638e+00</td></tr> <tr> <td>min</td><td>0.000000e+00</td><td>1.000000e+00</td></tr> <tr> <td>25%</td><td>2.610460e+03</td><td>2.000000e+00</td></tr> <tr> <td>50%</td><td>7.003000e+03</td><td>3.000000e+00</td></tr> <tr> <td>75%</td><td>2.757519e+04</td><td>5.000000e+00</td></tr> <tr> <td>max</td><td>1.901909e+10</td><td>1.000000e+01</td></tr> </table>			VALOR_CONSOLIDADO	VALOR_FAIXA	count	1.499292e+07	1.499292e+07	mean	1.451737e+05	3.581026e+00	std	9.254073e+06	1.876638e+00	min	0.000000e+00	1.000000e+00	25%	2.610460e+03	2.000000e+00	50%	7.003000e+03	3.000000e+00	75%	2.757519e+04	5.000000e+00	max	1.901909e+10	1.000000e+01
	VALOR_CONSOLIDADO	VALOR_FAIXA																											
count	1.499292e+07	1.499292e+07																											
mean	1.451737e+05	3.581026e+00																											
std	9.254073e+06	1.876638e+00																											
min	0.000000e+00	1.000000e+00																											
25%	2.610460e+03	2.000000e+00																											
50%	7.003000e+03	3.000000e+00																											
75%	2.757519e+04	5.000000e+00																											
max	1.901909e+10	1.000000e+01																											

#### ▪ Estatística descritiva - Variáveis categóricas:

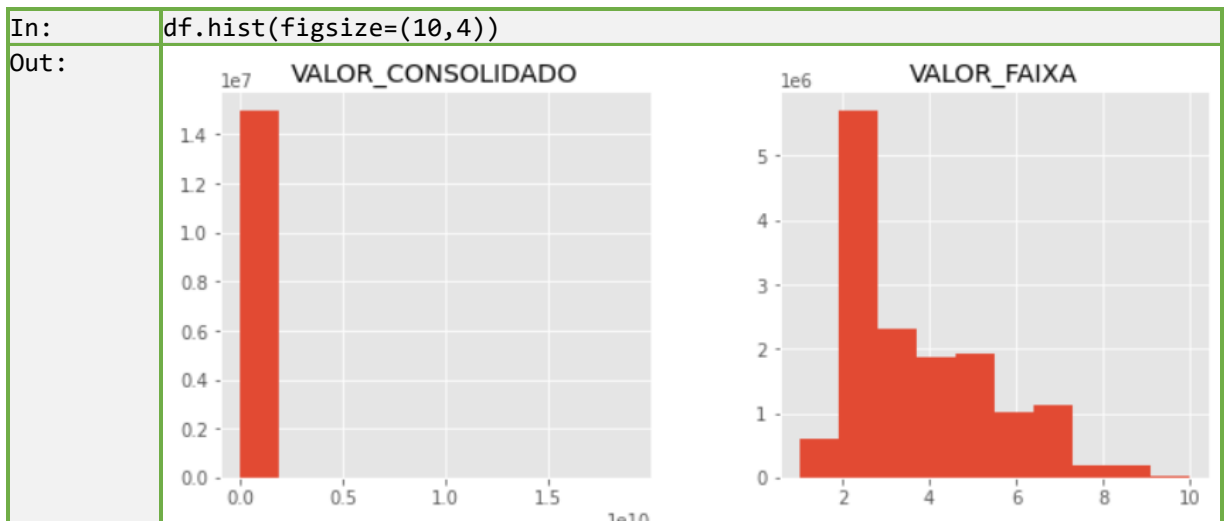
Para as variáveis categóricas, deixaremos para fazer mais destaques nos tópicos seguintes, quando as aprofundaremos.

In:	df.describe(include=['O'])																																															
Out:	<table><tr><th></th><th>TIPO_PESSOA</th><th>TIPO_SITUACAO_INSCRICAO</th><th>SITUACAO_INSCRICAO</th><th>RECEITA_PRINCIPAL</th><th>INDICADOR_AJUIZADO</th></tr><tr><td>count</td><td>14992922</td><td>14992922</td><td>14992922</td><td>14992922</td><td>14992922</td></tr><tr><td>unique</td><td>2</td><td>5</td><td>117</td><td>13</td><td>2</td></tr><tr><td>top</td><td>PJ</td><td>Em cobrança</td><td>ATIVA EM COBRANCA</td><td>SIMPLES NACIONAL</td><td>NAO</td></tr><tr><td>freq</td><td>12223696</td><td>11476828</td><td>5029325</td><td>2662456</td><td>10107954</td></tr></table>									TIPO_PESSOA	TIPO_SITUACAO_INSCRICAO	SITUACAO_INSCRICAO	RECEITA_PRINCIPAL	INDICADOR_AJUIZADO	count	14992922	14992922	14992922	14992922	14992922	unique	2	5	117	13	2	top	PJ	Em cobrança	ATIVA EM COBRANCA	SIMPLES NACIONAL	NAO	freq	12223696	11476828	5029325	2662456	10107954										
		TIPO_PESSOA	TIPO_SITUACAO_INSCRICAO	SITUACAO_INSCRICAO	RECEITA_PRINCIPAL	INDICADOR_AJUIZADO																																										
	count	14992922	14992922	14992922	14992922	14992922																																										
	unique	2	5	117	13	2																																										
	top	PJ	Em cobrança	ATIVA EM COBRANCA	SIMPLES NACIONAL	NAO																																										
	freq	12223696	11476828	5029325	2662456	10107954																																										
	<table><tr><th>HA_OUTROS_DEBITOS</th><th>INSC_&gt;1_ANO</th><th>SIT_CADAST</th><th>CNAE_PRINC</th><th>NAT_JURID</th><th>PORTE</th><th>ATIV_&gt;10_ANOS</th><th>PARCELADO</th></tr><tr><td>14992922</td><td>14992922</td><td>12221987.0</td><td>12221987.0</td><td>12221724.0</td><td>12221718.0</td><td>12221987</td><td>14992922</td></tr><tr><td>2</td><td>2</td><td>5.0</td><td>80.0</td><td>72.0</td><td>3.0</td><td>2</td><td>2</td></tr><tr><td>S</td><td>S</td><td>2.0</td><td>47.0</td><td>2062.0</td><td>1.0</td><td>S</td><td>N</td></tr><tr><td>12987108</td><td>8999041</td><td>7698079.0</td><td>2477111.0</td><td>6846692.0</td><td>6865172.0</td><td>8583880</td><td>11540162</td></tr></table>								HA_OUTROS_DEBITOS	INSC_>1_ANO	SIT_CADAST	CNAE_PRINC	NAT_JURID	PORTE	ATIV_>10_ANOS	PARCELADO	14992922	14992922	12221987.0	12221987.0	12221724.0	12221718.0	12221987	14992922	2	2	5.0	80.0	72.0	3.0	2	2	S	S	2.0	47.0	2062.0	1.0	S	N	12987108	8999041	7698079.0	2477111.0	6846692.0	6865172.0	8583880	11540162
	HA_OUTROS_DEBITOS	INSC_>1_ANO	SIT_CADAST	CNAE_PRINC	NAT_JURID	PORTE	ATIV_>10_ANOS	PARCELADO																																								
	14992922	14992922	12221987.0	12221987.0	12221724.0	12221718.0	12221987	14992922																																								
	2	2	5.0	80.0	72.0	3.0	2	2																																								
S	S	2.0	47.0	2062.0	1.0	S	N																																									
12987108	8999041	7698079.0	2477111.0	6846692.0	6865172.0	8583880	11540162																																									

<sup>14</sup> A notação científica é usada para representar números muito grandes ou pequenos de uma forma compacta e compreensível. Por exemplo, 6.000.000 pode ser representado como 6e+6. Da mesma forma, 0,000006 pode ser representado como 6e-6. (DELFSTACK, 2021).

- **Distribuição das variáveis (apenas para dados contínuos):**

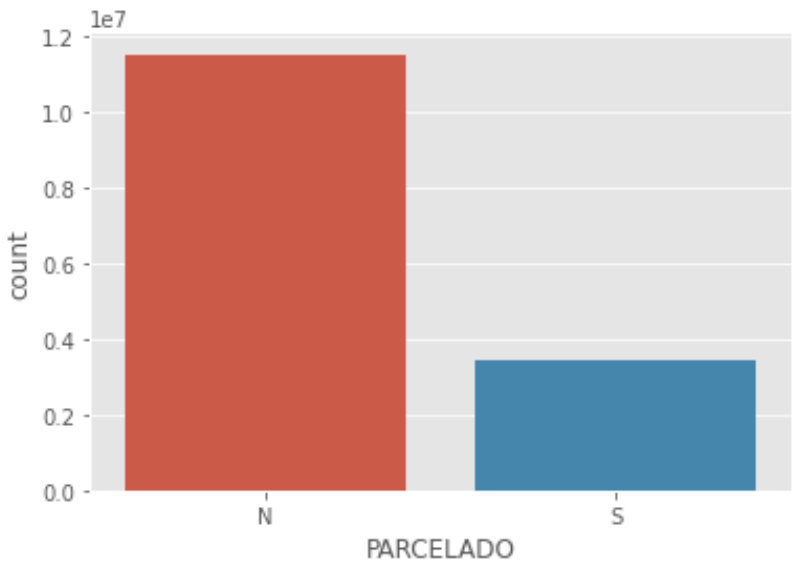
Ambos os campos com dados contínuos (VALOR\_CONSOLIDADO e VALOR\_FAIXA) se referem ao mesmo dado: valor do débito. O valor consolidado é o valor exato, enquanto valor faixa classifica todos os valores em 10 faixas. Curiosamente, apresentam resultados diferentes no histograma. Apesar de ambos demonstrarem uma distribuição assimétrica à esquerda, o que indica que a maior parte dos débitos é de baixo valor, o campo valor consolidado apresenta essa distribuição muito mais acentuada. Ou seja, a classificação em faixa que realizamos não ficou proporcional à distribuição dos valores originais. Isso porque criamos, propositadamente, várias faixas para valores mais baixos, enquanto deixamos uma vasta gama de valores mais elevados em uma menor quantidade de faixas. Assim, essa classificação em faixas foi eficaz porque conseguiu estratificar melhor os valores, o que permite uma melhor análise, conforme faremos adiante.



- **Análise da variável alvo PARCELADO:**

Observe que 23% dos débitos encontram-se parcelados (PARCELADO='S'). Ou seja, temos um caso claro de classes desbalanceadas. Isso afetará a aplicação dos algoritmos de machine learning. Detalharemos isso no capítulo 5.

In:	<code># Distribuição das classes da variável</code>
-----	---

	<code>df['PARCELADO'].value_counts()</code>						
Out:	<pre> N    11540162 S      3452760 Name: PARCELADO, dtype: int64 </pre>						
In:	<pre> # Porcentagem de débitos parcelados (taxa de parcelamento) df['PARCELADO_num'].sum()/df['PARCELADO'].count() </pre>						
Out:	<code>0.23029266743333954</code>						
In:	<pre> # Plotagem do gráfico sns.countplot(data=df, x='PARCELADO') </pre>						
Out:	 <table border="1"> <thead> <tr> <th>PARCELADO</th> <th>count</th> </tr> </thead> <tbody> <tr> <td>N</td> <td>11540162</td> </tr> <tr> <td>S</td> <td>3452760</td> </tr> </tbody> </table>	PARCELADO	count	N	11540162	S	3452760
PARCELADO	count						
N	11540162						
S	3452760						

c) **PARTE 3 – Distribuições das variáveis em relação à variável alvo**

▪ **Variável VALOR\_CONSOLIDADO:**

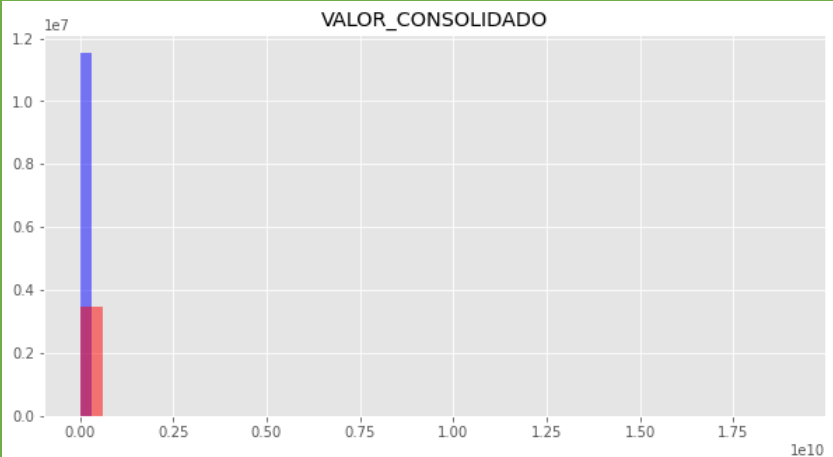
Observa-se que o montante total da dívida em estoque na nossa base de dados é de R\$ 2.176.578.087.492,30 (R\$ 2,176 trilhões). A título de curiosidade, em seu já citado relatório, a PGFN afirma que, em 2021, o montante em estoque de débitos tributários não previdenciários se encontrava em R\$ 1,941 trilhões (PGFN, 2022, p. 16).

Observa-se ainda que do referido montante, R\$ 290 bilhões (13,3%) encontram-se parcelados. Ou seja, a taxa de parcelamento do montante é menor do que a taxa do quantitativo de débitos, que, como vimos no item anterior, foi de 23%. Isso indica que o parcelamento de valores menores é predominante.

In:	<code># VALOR TOTAL DA CARTEIRA (MONTANTE TOTAL)</code>
-----	---

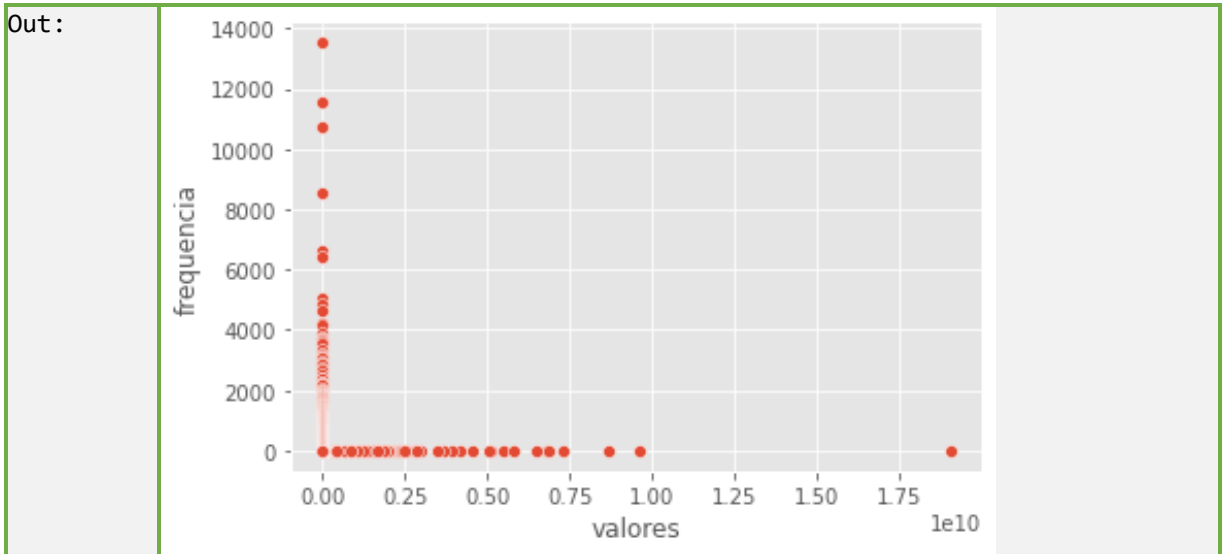
	<pre>soma_valor = df['VALOR_CONSOLIDADO'].sum() float_formatter = "R\$ {:,.2f}".format float_formatter(soma_valor)</pre>
Out:	'R\$ 2,176,578,087,492.30'
In:	<pre># DISTRIBUIÇÃO DO MONTANTE X PARCELAMENTO pivot_valor_total = pd.pivot_table(df, index=['PARCELADO'], values='VA- LOR_CONSOLIDADO', aggfunc = 'sum') pivot_valor_total.style.format({'VALOR_CONSOLIDADO': 'R\$ {:,.2f}'})</pre>
Out:	<pre>      VALOR_CONSOLIDADO PARCELADO N      R\$ 1,886,265,794,791.40 S      R\$ 290,312,292,700.90</pre>

Plotamos o gráfico de distribuição das classes de parcelamento versus valor consolidado, onde a barra azul se refere ao montante não parcelado, e a vermelha o parcelado.

In:	<pre># DISTRIBUIÇÃO DAS CLASSES DE PARCELAMENTO X VALOR_CONSOLIDADO x_cont=['VALOR_CONSOLIDADO'] fig, ax = plt.subplots(1, figsize=(10, 5)); df[df.PARCELADO == "N"][x_cont].hist( bins=30, color="blue", alpha=0.5, ax=ax); df[df.PARCELADO == "S"][x_cont].hist( bins=30, color="red", alpha=0.5, ax=ax);</pre>
Out:	

Plotamos ainda gráfico de frequência, onde é possível verificar que a maior frequência de valores de débitos é de pequenos valores.

In:	<pre># GRÁFICO VALORES X FREQUÊNCIA VALOR_CONSOLIDADO sns.scatterplot(data=contador_valor, x='valores', y='frequencia');</pre>
-----	--



Por fim, plotamos o gráfico box-plot, onde é possível observar que ambas as situações (parcelado ou não parcelado) encontram-se mal distribuídos. Observa-se ainda a concentração dos débitos parcelados em menores valores, embora apresente outliers<sup>15</sup> com valores elevados.

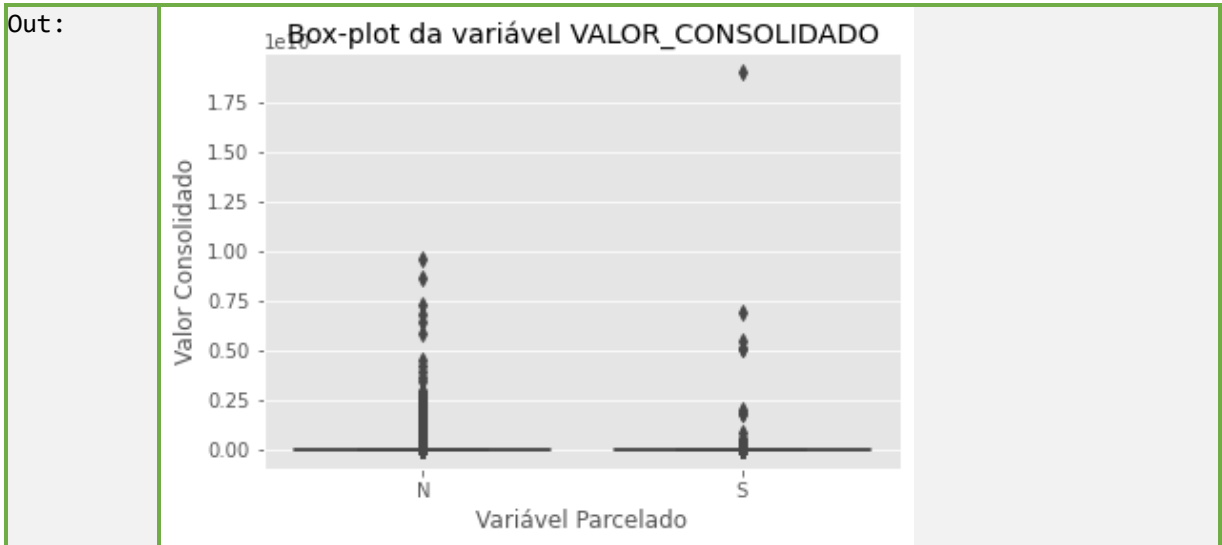
In:

```
# BOX-PLOT VALOR_CONSOLIDADO X PARCELADO
sns.boxplot(x="PARCELADO", y="VALOR_CONSOLIDADO", data=df);
plt.title('Box-plot da variável VALOR_CONSOLIDADO');
plt.xlabel('Variável Parcelado');
plt.ylabel('Valor Consolidado');
df.groupby(['PARCELADO'])['VALOR_CONSOLIDADO'].mean()
```

---

<sup>15</sup> “Um outlier é uma observação que se diferencia tanto das demais observações que levanta suspeitas de que aquela observação foi gerada por um mecanismo distinto” (HAWKINS apud SALES, 2018).





▪ **Variável FAIXA DE VALOR (VALOR\_FAIXA):**

Observa-se que a faixa 2 (de R\$ 1.000 a R\$ 5.000) corresponde por 38% dos débitos. Somada às faixas 3, 4 e 5 (valores de 1.000 a 50.000 reais) respondem por 78% dos débitos. Em observação ao gráfico da taxa de parcelamento, observa-se que, em linhas gerais, quanto maior a faixa, menor é a taxa de parcelamento. A última faixa (faixa 10 – acima de R\$ 10 milhões) possui uma taxa de apenas 3,5%, em comparação a uma taxa média de 11,5% para as faixas 2, 3 e 4.

In:	<pre># CONTAGEM DE VALORES COM PERCENTUAL df_cont = df frequencia = df_cont['VALOR_FAIXA'].value_counts() percentual = df_cont['VALOR_FAIXA'].value_counts(normalize = True)*100 dist_freq_perc = pd.DataFrame({'Frequência': frequencia, 'Porcentagem(%)': percentual}) dist_freq_perc</pre>
Out:	<pre>Frequência  Porcentagem(%) 2.0    5709103    38.078660 3.0    2318277    15.462478 5.0    1930306    12.874784 4.0    1880276    12.541093 7.0    1133213     7.558321 6.0    1017555     6.786903 1.0     594826     3.967379 8.0     194991     1.300554 9.0     191943     1.280224 10.0     22430     0.149604</pre>
In:	<pre># Distribuição das classes x VALOR_FAIXA plt.figure(figsize=(8,4)) sns.countplot(data=df, x = 'VALOR_FAIXA', hue = 'PARCELADO');</pre>



#### ▪ Variável TIPO\_SITUACAO\_INSCRICAO

Observa-se que 76,5% dos débitos estão em cobrança, seguido por 23% de débitos que estão com benefício fiscal. Estas duas situações respondem por 99,5% dos débitos. As demais situações se dividem entre garantia, suspenso por decisão judicial e em negociação (de parcelamento). Como já vimos, benefício fiscal se refere aos débitos parcelados, ou seja, não há análise em relação à taxa de parcelamento, já que são informações redundantes.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)
-----	--

Out:			
		<b>Frequência</b>	<b>Porcentagem(%)</b>
	<b>Em cobrança</b>	11476828	76.548307
	<b>Benefício Fiscal</b>	3452760	23.029267
	<b>Garantia</b>	39092	0.260736
	<b>Suspensão por decisão judicial</b>	22229	0.148263
	<b>Em negociação</b>	2013	0.013426

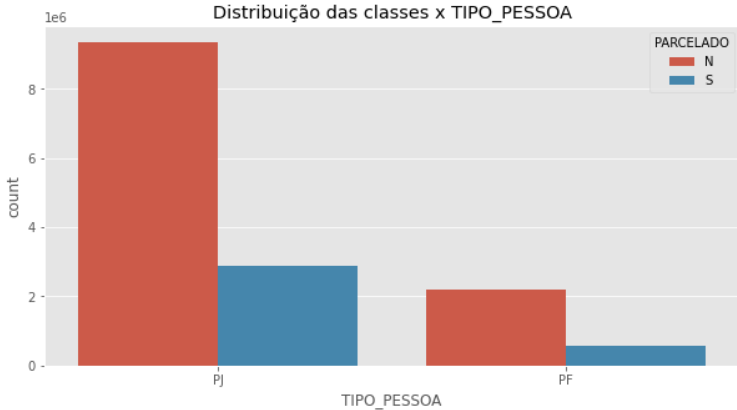
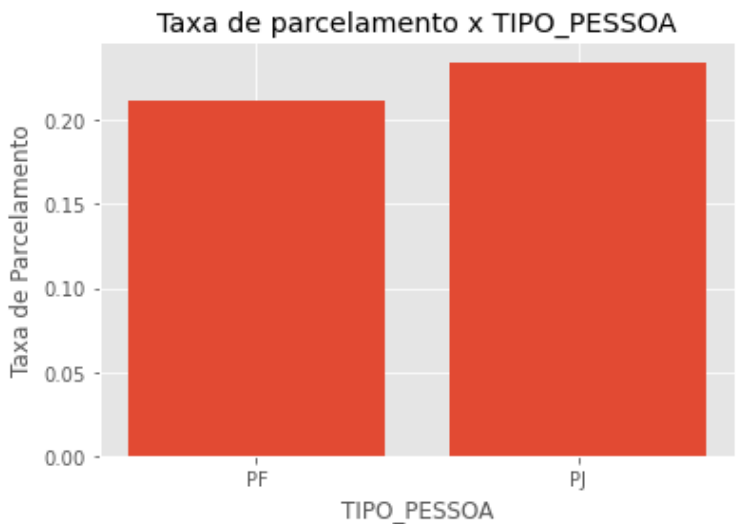
▪ **Variável Situação da Inscrição do Débito (SITUACAO\_INSCRICAO):**

Esta variável é um subtipo da variável do item anterior, ou seja, é um detalhamento da situação do débito. Possui 117 valores/situações distintas. No entanto, observa-se que as 5 situações mais frequentes correspondem por 87% dos casos, com destaque para ATIVA EM COBRANCA (33,5%). Assim, como no item anterior, a situação está diretamente relacionada à taxa de parcelamento, ou seja, também são informações redundantes, não cabendo análise.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)		
Out:		<b>Frequência</b>	<b>Porcentagem(%)</b>
	ATIVA EM COBRANCA	5029325	33.544662
	ATIVA AJUIZADA	3712324	24.760510
	ATIVA NAO AJUIZAVEL NEGOCIADA NO SISPAR	2644713	17.639744
	ATIVA A SER AJUIZADA	1272596	8.487979
	ATIVA NAO PRIORIZADA PARA AJUIZAMENTO	835908	5.575351
	...	...	...
	ATIVA PARCELADA ART 8 MP 303/06 E AJUIZAMENTO A SER CANCELADO	1	0.000007
	ATIVA COM AJUIZAMENTO SUSPENSO PARA ANALISE DO ORGAO DE ORIGEM	1	0.000007
	ATIVA AJUIZADA OPCAO PAGAMENTO A VISTA MP470	1	0.000007
	ATIVA NAO AJUIZADA COM EXIBILIDADE SUSPENSA - ANALISE MP 449	1	0.000007
	ATIVA COM PARCELAMENTO SIMPLIFICADO E AJUIZAMENTO A SER CANCELADO	1	0.000007
	(incompleto por limitação de espaço)		

▪ **Variável TIPO DE PESSOA (TIPO\_PESSOA):**

Observa-se que 81,5% dos débitos são de pessoa jurídica, enquanto 18,5% são de pessoa física. Observa-se ainda uma taxa de parcelamento levemente mais alta para pessoa jurídica, da ordem de 23,4% contra 21,1%.

In:	<pre># CONTAGEM DE VALORES COM PERCENTUAL df_cont = df frequencia = df_cont['TIPO_PESSOA'].value_counts() percentual = df_cont['TIPO_PESSOA'].value_counts(normalize = True)*100 dist_freq_perc = pd.DataFrame({'Frequência': frequencia, 'Porcentagem(%)': percentual}) dist_freq_perc</pre>	
Out:	<pre>Frequência  Porcentagem(%) PJ      12223696      81.529778 PF       2769226      18.470222</pre>	
In:	<pre># DISTRIBUIÇÃO DAS CLASSES X TIPO_PESSOA # Distribuição das classes x TIPO_PESSOA plt.figure(figsize=(10,5)) sns.countplot(data=df, x = 'TIPO_PESSOA', hue = 'PARCELADO'); plt.title('Distribuição das classes x TIPO_PESSOA');</pre>	
Out:		
	<pre># TAXA DE PARCELAMENTO x TIPO_PESSOA (GRAF) x=df_plot['TIPO_PESSOA'] y=df_plot['PARCELADO_num'] plt.bar(x,y); plt.title('Taxa de parcelamento x TIPO_PESSOA'); plt.xlabel('TIPO_PESSOA'); plt.ylabel('Taxa de Parcelamento');</pre>	
		

Os próximos 5 itens são referentes às variáveis CNAE\_PRINC, NAT\_JURID, SIT\_CADAST, PORTE, ATIV\_>10\_ANOS, se aplicam exclusivamente aos débitos de pessoa jurídica. Assim, os percentuais consideram apenas esta amostra.

▪ **Variável CNAE Principal (CNAE\_PRINC):**

Esta variável apresenta 80 valores, ou códigos de atividade econômica. Lembre-se que originalmente eram 1.343. Mesmo com esse trabalho de simplificação realizado no notebook 04\_cnpj\_final.ipynb, ainda encontramos um resultado bem diversificado e pulverizado, à exceção dos 2 registros mais frequentes.

Assim, com base nos resultados do processamento desta variável (codificação abaixo), construímos a Tabela 2, consultando também a tabela CNAE, para interpretarmos os códigos. Limita-se aos primeiros 20 registros (de um total de 80), mas contempla 75,6% do total de frequência.

Destaca-se, de imediato, os dois primeiros, que totalizam 30% dos débitos. Ambas as atividades pertencem ao setor de comércio (CNAE Seção G), e possuem taxas de parcelamento próximas à média, o que é de se esperar, já que compõem uma boa fatia do total.

Em seguida, vemos diversas divisões CNAE que apresentam uma frequência de 1,2 a 5,3% do total. A maioria possui taxa de parcelamento próxima à média, mas destacamos duas com taxas acima de 30%: atividades de atenção à saúde humana (hospitais, consultórios, etc) com 35%; e atividades imobiliárias com 32,7%.

TABELA 2 – RESULTADOS DA VARIÁVEL CNAE					
POR ORDEM DECRESCENTE DE FREQUÊNCIA					
CNAE SEÇÃO	CNAE DIVISÃO	DESCRIÇÃO DA DIVISÃO	FREQ	FREQ %	TX PARC %
G	47	COMÉRCIO VAREJISTA	2477111	20,3%	22,9%
G	46	COMÉRCIO POR ATACADO, EXCETO VEÍCULOS AUTOMOTORES E MOTOCICLETAS	1195482	9,8%	24,1%
H	49	TRANSPORTE TERRESTRE	648631	5,3%	23,4%
F	41	CONSTRUÇÃO DE EDIFÍCIOS	549760	4,5%	22,2%

I	56	ALIMENTAÇÃO	531001	4,3%	23,8%
F	43	SERVIÇOS ESPECIALIZADOS PARA CONSTRUÇÃO	491643	4,0%	17,6%
G	45	COMÉRCIO E REPARAÇÃO DE VEÍCULOS AUTOMOTORES E MOTOCICLETAS	468311	3,8%	24,5%
N	82	SERVIÇOS DE ESCRITÓRIO, DE APOIO ADMINISTRATIVO E OUTROS SERVIÇOS PRESTADOS ÀS EMPRESAS	423724	3,5%	22,1%
Q	86	ATIVIDADES DE ATENÇÃO À SAÚDE HUMANA	367078	3,0%	35,0%
C	10	FABRICAÇÃO DE PRODUTOS ALIMENTÍCIOS	253291	2,1%	22,9%
C	25	FABRICAÇÃO DE PRODUTOS DE METAL, EXCETO MÁQUINAS E EQUIPAMENTOS	246672	2,0%	21,1%
P	85	EDUCAÇÃO	229963	1,9%	26,6%
C	14	CONFECÇÃO DE ARTIGOS DO VESTUÁRIO E ACESSÓRIOS	212009	1,7%	18,8%
M	71	SERVIÇOS DE ARQUITETURA E ENGENHARIA; TESTES E ANÁLISES TÉCNICAS	185159	1,5%	28,4%
N	81	SERVIÇOS PARA EDIFÍCIOS E ATIVIDADES PAISAGÍSTICAS	169074	1,4%	18,8%
L	68	ATIVIDADES IMOBILIÁRIAS	162426	1,3%	32,7%
M	73	PUBLICIDADE E PESQUISA DE MERCADO	161116	1,3%	23,4%
F	42	OBRAS DE INFRA-ESTRUTURA	160142	1,3%	22,6%
J	62	ATIVIDADES DOS SERVIÇOS DE TECNOLOGIA DA INFORMAÇÃO	155669	1,3%	28,2%
C	28	FABRICAÇÃO DE MÁQUINAS E EQUIPAMENTOS	151337	1,2%	22,0%

Fonte: elaboração própria.

Criamos ainda a Tabela 3, que ordena os dados por taxa de parcelamento. Observa-se vários casos de taxas bem superiores à média, mas possuem baixa frequência. Além das já citadas anteriormente, destacamos as 3 primeiras: atividades veterinárias com 43,3%; rádio e televisão com 43%; e jurídica, contabilidade e auditoria com 37,7%.

TABELA 3 – RESULTADOS DA VARIÁVEL CNAE POR ORDEM DECRESCENTE DE TAXA DE PARCELAMENTO					
CNAE SEÇÃO	CNAE DIVISÃO	DESCRIÇÃO DA DIVISÃO	FREQ	FREQ %	TX PARC %
M	75	ATIVIDADES VETERINÁRIAS	7107	0,1%	43,3%
J	60	ATIVIDADES DE RÁDIO E DE TELEVISÃO	20724	0,2%	43,0%
M	69	ATIVIDADES JURÍDICAS, DE CONTABILIDADE E DE AUDITORIA	133106	1,1%	37,7%
I	55	ALOJAMENTO	59798	0,5%	36,2%
O	84	ADMINISTRAÇÃO PÚBLICA, DEFESA E SEGURIDADE SOCIAL	11312	0,1%	36,1%
Q	86	ATIVIDADES DE ATENÇÃO À SAÚDE HUMANA	367078	3,0%	35,0%

E	39	DESCONTAMINAÇÃO E OUTROS SERVIÇOS DE GESTÃO DE RESÍDUOS	563	0,0%	33,0%
L	68	ATIVIDADES IMOBILIÁRIAS	162426	1,3%	32,7%
H	50	TRANSPORTE AQUAVIÁRIO	9319	0,1%	31,6%
K	66	ATIVIDADES AUXILIARES DOS SERVIÇOS FINANCEIROS, SEGUROS, PREVIDÊNCIA COMPLEMENTAR E PLANOS DE SAÚDE	123248	1,0%	30,7%
J	59	ATIVIDADES CINEMATOGRAFICAS, PRODUÇÃO DE VÍDEOS E DE PROGRAMAS DE TELEVISÃO; GRAVAÇÃO DE SOM E EDIÇÃO DE MÚSICA	32006	0,3%	30,0%
R	93	ATIVIDADES ESPORTIVAS E DE RECREAÇÃO E LAZER	55787	0,5%	30,0%
E	38	COLETA, TRATAMENTO E DISPOSIÇÃO DE RESÍDUOS; RECUPERAÇÃO DE MATERIAIS	30827	0,3%	29,2%
C	11	FABRICAÇÃO DE BEBIDAS	36213	0,3%	28,9%
M	72	PESQUISA E DESENVOLVIMENTO CIENTÍFICO	5398	0,0%	28,6%
M	71	SERVIÇOS DE ARQUITETURA E ENGENHARIA; TESTES E ANÁLISES TÉCNICAS	185159	1,5%	28,4%
J	62	ATIVIDADES DOS SERVIÇOS DE TECNOLOGIA DA INFORMAÇÃO	155669	1,3%	28,2%
R	90	ATIVIDADES ARTÍSTICAS, CRIATIVAS E DE ESPETÁCULOS	32983	0,3%	28,1%
M	70	ATIVIDADES DE SEDES DE EMPRESAS E DE CONSULTORIA EM GESTÃO EMPRESARIAL	114835	0,9%	28,0%
J	63	ATIVIDADES DE PRESTAÇÃO DE SERVIÇOS DE INFORMAÇÃO	40006	0,3%	27,7%

Fonte: elaboração própria.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)																																						
Out:	<table><thead><tr><th></th><th>Frequência</th><th>Porcentagem(%)</th></tr></thead><tbody><tr><td>47.0</td><td>2477111</td><td>20.267662</td></tr><tr><td>46.0</td><td>1195482</td><td>9.781405</td></tr><tr><td>49.0</td><td>648631</td><td>5.307083</td></tr><tr><td>41.0</td><td>549760</td><td>4.498123</td></tr><tr><td>56.0</td><td>531001</td><td>4.344637</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td>97.0</td><td>2837</td><td>0.023212</td></tr><tr><td>91.0</td><td>2753</td><td>0.022525</td></tr><tr><td>99.0</td><td>1390</td><td>0.011373</td></tr><tr><td>92.0</td><td>1104</td><td>0.009033</td></tr><tr><td>39.0</td><td>563</td><td>0.004606</td></tr></tbody></table> <p>80 rows × 2 columns</p>				Frequência	Porcentagem(%)	47.0	2477111	20.267662	46.0	1195482	9.781405	49.0	648631	5.307083	41.0	549760	4.498123	56.0	531001	4.344637	...	...	...	97.0	2837	0.023212	91.0	2753	0.022525	99.0	1390	0.011373	92.0	1104	0.009033	39.0	563	0.004606
	Frequência	Porcentagem(%)																																					
47.0	2477111	20.267662																																					
46.0	1195482	9.781405																																					
49.0	648631	5.307083																																					
41.0	549760	4.498123																																					
56.0	531001	4.344637																																					
...	...	...																																					
97.0	2837	0.023212																																					
91.0	2753	0.022525																																					
99.0	1390	0.011373																																					
92.0	1104	0.009033																																					
39.0	563	0.004606																																					
In:	# Taxa de parcelamento x CNAE_PRINC # Codificação suprimida (similar à do item anterior)																																						

Out:	CNAE_PRINC	PARCELADO_num
	58	75.0 0.432531
	44	60.0 0.430371
	52	69.0 0.377136
	40	55.0 0.361551
	65	84.0 0.361475
	...	...
	37	51.0 0.115368
	75	94.0 0.106555
	78	97.0 0.076489
	73	92.0 0.073370
	69	88.0 0.042430
	80 rows × 2 columns	

▪ **Variável Natureza Jurídica (NAT\_JURID):**

Similarmente ao item anterior, aqui também temos uma alta diversidade de valores, além de códigos externos para consultarmos. Assim, lançamos mão de uma tabela para auxiliar-nos na análise.

Na Tabela 4, que lista apenas 23 linhas (de um total de 72), observa-se que os 5 primeiros valores já correspondem a 97,3% de todos os casos. Destaque para sociedade empresária limitada (com 56%) e empresário individual (com 22,1%). Ambas apresentam taxa de parcelamento próxima à média.

TABELA 4 – RESULTADOS DA VARIÁVEL NAT_JURID				
POR ORDEM DECRESCENTE DE TAXA DE FREQUÊNCIA				
NAT_JURID	DESCRIÇÃO	FREQ	FREQ %	TX PARC %
2062	Sociedade Empresária Limitada	6846692	56,0%	23,4%
2135	Empresário Individual	2704032	22,1%	22,0%
2305	Empresa Individual de Responsabilidade Limitada (de Natureza Empresária)	1660180	13,6%	26,0%
2240	Sociedade Simples Limitada	423722	3,5%	25,0%
2054	Sociedade Anônima Fechada	261163	2,1%	22,3%
3999	Associação Privada	99626	0,8%	19,1%
2232	Sociedade Simples Pura	48386	0,4%	39,6%
2313	Empresa Individual de Responsabilidade Limitada (de Natureza Simples)	37499	0,3%	28,4%



2046	Sociedade Anônima Aberta	24142	0,2%	11,4%
2143	Cooperativa	23977	0,2%	18,8%
3271	Órgão de Direção Local de Partido Político	20339	0,2%	1,3%
2321	Sociedade Unipessoal de Advogados	12713	0,1%	34,0%
4014	Empresa Individual Imobiliária	7889	0,1%	16,4%
1244	Município	6973	0,1%	42,0%
3131	Entidade Sindical	6883	0,1%	20,2%
3220	Organização Religiosa	6073	0,0%	13,7%
3069	Fundação Privada	4732	0,0%	24,6%
2038	Sociedade de Economia Mista	4480	0,0%	31,1%
3085	Condomínio Edifício	4239	0,0%	17,0%
2070	Sociedade Empresária em Nome Coletivo	3437	0,0%	11,8%
2151	Consórcio de Sociedades	2795	0,0%	21,6%
2011	Empresa Pública	2656	0,0%	30,5%
1120	Autarquia Municipal	1247	0,0%	27,3%

Fonte: elaboração própria.

Na Tabela 5, reordenamos os dados por taxa de parcelamento, filtrando apenas taxas acima de 30%. Destaque para as três primeiras com taxas acima de 50%. Destarte, todas possuem baixíssima frequência.

Nesta tabela destaca-se ainda uma predominância de códigos relacionados ao grupo administração pública, que são todos aqueles que começam com 1.

Os demais grupos são: 2 - Entidades Empresariais; 3 - Entidades sem Fins Lucrativos; 4 - Pessoas Físicas (equiparadas à pessoa jurídica); 5 - Organizações Internacionais e Outras Instituições Extraterritoriais.

TABELA 5 – RESULTADOS DA VARIÁVEL NAT_JURID				
POR ORDEM DECRESCENTE DE TAXA DE PARCELAMENTO				
NAT_JURID	DESCRIÇÃO	FREQ	FREQ %	TX PARC %
2267	Sociedade Simples em Comandita Simples	26	0,0%	84,6%
2089	Sociedade Empresária em Comandita Simples	54	0,0%	61,1%
1279	Fundação Pública de Direito Privado Municipal	224	0,0%	54,0%
1023	Órgão Público do Poder Executivo Estadual ou do Distrito Federal	48	0,0%	43,8%
1210	Consórcio Público de Direito Público (Associação Pública)	252	0,0%	42,5%
1244	Município	6973	0,1%	42,0%

1236	Estado ou Distrito Federal	113	0,0%	40,7%
2232	Sociedade Simples Pura	48386	0,4%	39,6%
2321	Sociedade Unipessoal de Advogados	12713	0,1%	34,0%
1180	Órgão Público Autônomo Municipal	16	0,0%	31,3%
2038	Sociedade de Economia Mista	4480	0,0%	31,1%
2011	Empresa Pública	2656	0,0%	30,5%

Fonte: elaboração própria.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)																																									
Out:	<table><tr><th></th><th>Frequência</th><th>Porcentagem(%)</th></tr><tr><td>2062.0</td><td>6846692</td><td>56.020673</td></tr><tr><td>2135.0</td><td>2704032</td><td>22.124800</td></tr><tr><td>2305.0</td><td>1660180</td><td>13.583845</td></tr><tr><td>2240.0</td><td>423722</td><td>3.466958</td></tr><tr><td>2054.0</td><td>261163</td><td>2.136875</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td>3239.0</td><td>2</td><td>0.000016</td></tr><tr><td>5010.0</td><td>2</td><td>0.000016</td></tr><tr><td>1171.0</td><td>1</td><td>0.000008</td></tr><tr><td>2275.0</td><td>1</td><td>0.000008</td></tr><tr><td>1228.0</td><td>1</td><td>0.000008</td></tr><tr><td colspan="3">72 rows × 2 columns</td></tr></table>				Frequência	Porcentagem(%)	2062.0	6846692	56.020673	2135.0	2704032	22.124800	2305.0	1660180	13.583845	2240.0	423722	3.466958	2054.0	261163	2.136875	...	...	...	3239.0	2	0.000016	5010.0	2	0.000016	1171.0	1	0.000008	2275.0	1	0.000008	1228.0	1	0.000008	72 rows × 2 columns		
		Frequência	Porcentagem(%)																																							
	2062.0	6846692	56.020673																																							
	2135.0	2704032	22.124800																																							
	2305.0	1660180	13.583845																																							
	2240.0	423722	3.466958																																							
	2054.0	261163	2.136875																																							
	...	...	...																																							
	3239.0	2	0.000016																																							
	5010.0	2	0.000016																																							
	1171.0	1	0.000008																																							
	2275.0	1	0.000008																																							
	1228.0	1	0.000008																																							
	72 rows × 2 columns																																									
In:	# Taxa de parcelamento x NAT_JURID # Codificação suprimida (similar à do item anterior)																																									
Out:	<table><tr><th></th><th>NAT_JURID</th><th>PARCELADO_num</th></tr><tr><td>43</td><td>2267.0</td><td>0.846154</td></tr><tr><td>29</td><td>2089.0</td><td>0.611111</td></tr><tr><td>20</td><td>1279.0</td><td>0.540179</td></tr><tr><td>1</td><td>1023.0</td><td>0.437500</td></tr><tr><td>15</td><td>1210.0</td><td>0.424603</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td>13</td><td>1171.0</td><td>0.000000</td></tr><tr><td>12</td><td>1163.0</td><td>0.000000</td></tr><tr><td>9</td><td>1139.0</td><td>0.000000</td></tr><tr><td>5</td><td>1082.0</td><td>0.000000</td></tr><tr><td>71</td><td>8885.0</td><td>0.000000</td></tr></table>				NAT_JURID	PARCELADO_num	43	2267.0	0.846154	29	2089.0	0.611111	20	1279.0	0.540179	1	1023.0	0.437500	15	1210.0	0.424603	...	...	...	13	1171.0	0.000000	12	1163.0	0.000000	9	1139.0	0.000000	5	1082.0	0.000000	71	8885.0	0.000000			
		NAT_JURID	PARCELADO_num																																							
	43	2267.0	0.846154																																							
	29	2089.0	0.611111																																							
	20	1279.0	0.540179																																							
	1	1023.0	0.437500																																							
	15	1210.0	0.424603																																							
	...	...	...																																							
	13	1171.0	0.000000																																							
	12	1163.0	0.000000																																							
	9	1139.0	0.000000																																							
	5	1082.0	0.000000																																							
	71	8885.0	0.000000																																							

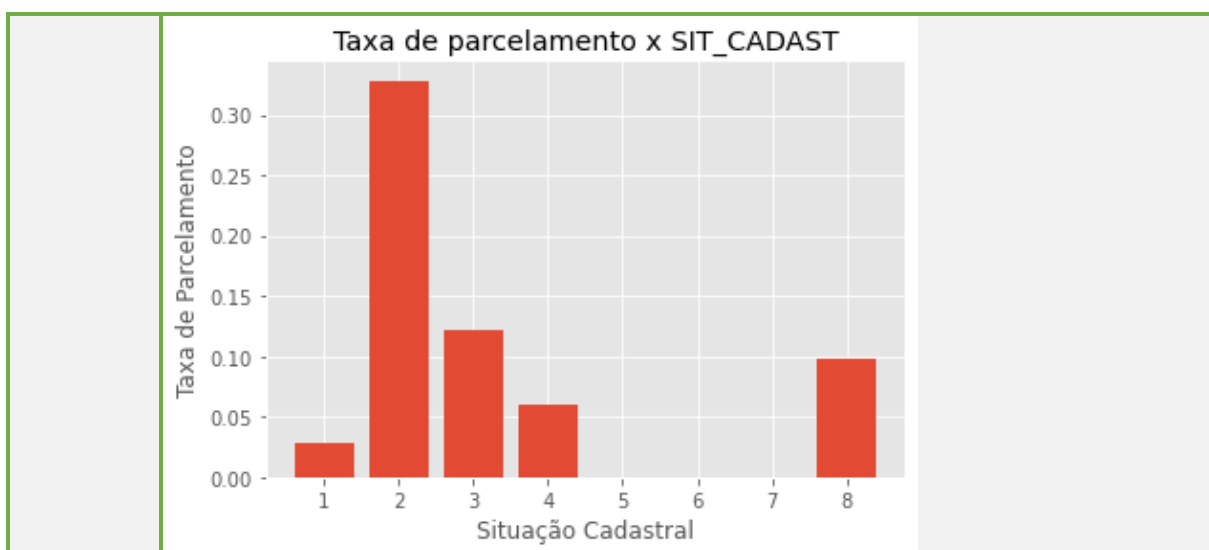
- **Variável Situação Cadastral do CNPJ (SIT\_CADAST):**

Os débitos cujos titulares estão com situação cadastral ativa correspondem por 63% do total, seguida por inapta, com 22%. A primeira apresenta uma taxa de parcelamento de 33%, ou seja, bem acima da média, enquanto a segunda apenas 6%, o que parece ser determinante para baixar a média geral.

QUADRO 3	
CÓDIGO DA SITUAÇÃO CADASTRAL	
CÓDIGO	DESCRIÇÃO
1	NULA
2	ATIVA
3	SUSPENSA
4	INAPTA
8	BAIXADA

Fonte: BRASIL. Receita Federal. Dados Públicos CNPJ.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)																			
Out:	Frequência	Porcentagem(%)																		
	2.0 7698079	62.985495																		
	4.0 2667775	21.827670																		
	8.0 1786107	14.613884																		
	3.0 69331	0.567265																		
	1.0 695	0.005686																		
In:	# Distribuição das classes x SIT_CADAST' # Codificação suprimida (similar à do item anterior)																			
Out:	<p>Distribuição das classes x SIT_CADAST</p> <table border="1"> <thead> <tr> <th>SIT_CADAST</th> <th>PARCELADO N (count)</th> <th>PARCELADO S (count)</th> </tr> </thead> <tbody> <tr> <td>4.0</td> <td>~2.5e6</td> <td>~0.2e6</td> </tr> <tr> <td>1.0</td> <td>~0.1e6</td> <td>~0.1e6</td> </tr> <tr> <td>2.0</td> <td>~5.2e6</td> <td>~2.5e6</td> </tr> <tr> <td>3.0</td> <td>~0.1e6</td> <td>~0.1e6</td> </tr> <tr> <td>8.0</td> <td>~1.5e6</td> <td>~0.2e6</td> </tr> </tbody> </table>		SIT_CADAST	PARCELADO N (count)	PARCELADO S (count)	4.0	~2.5e6	~0.2e6	1.0	~0.1e6	~0.1e6	2.0	~5.2e6	~2.5e6	3.0	~0.1e6	~0.1e6	8.0	~1.5e6	~0.2e6
SIT_CADAST	PARCELADO N (count)	PARCELADO S (count)																		
4.0	~2.5e6	~0.2e6																		
1.0	~0.1e6	~0.1e6																		
2.0	~5.2e6	~2.5e6																		
3.0	~0.1e6	~0.1e6																		
8.0	~1.5e6	~0.2e6																		
	# Taxa de parcelamento x SIT_CADAST # Codificação suprimida (similar à do item anterior)																			



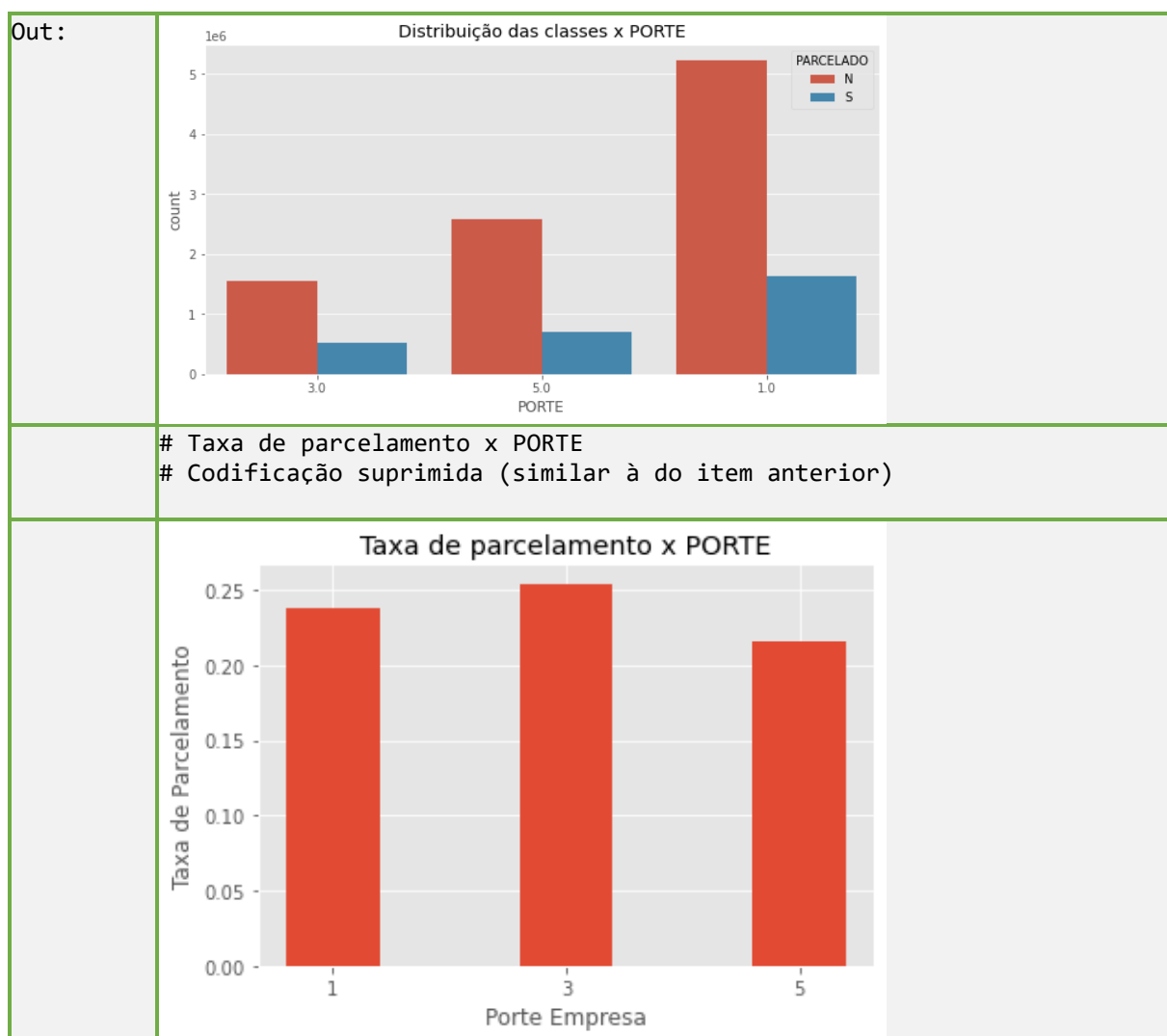
▪ **Variável Porte do CNPJ (PORTE):**

Os débitos das micro-empresas respondem por 56% do total, seguida por demais (27%) e EPP (17%). Quanto à taxa de parcelamento, temos 23,8%, 21,5% e 25,4%, respectivamente, ou seja, todas próximas à taxa média. Assim, o porte da empresa não parece ser tão determinante para a taxa de parcelamento.

QUADRO 4	
CÓDIGO DO PORTE DO CNPJ	
CÓDIGO	DESCRIÇÃO
0	NÃO INFORMADO
1	MICRO EMPRESA
3	EMPRESA DE PEQUENO PORTE (EPP)
5	DEMAIS

Fonte: BRASIL. Receita Federal. Dados Públicos CNPJ.

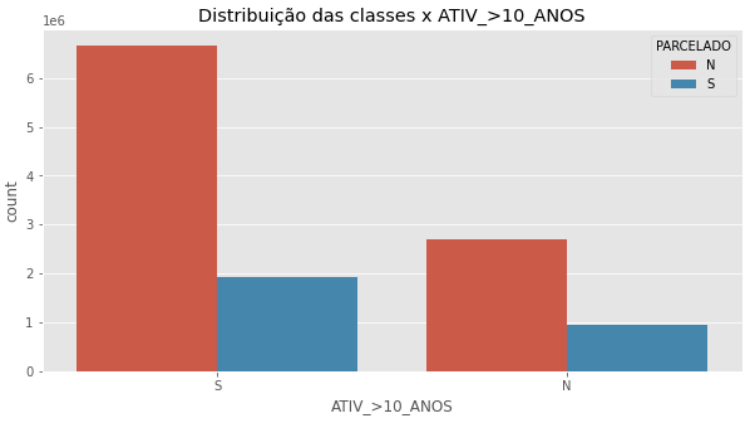

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)	
Out:	Frequência	Porcentagem(%)
	1.0 6865172	56.171906
	5.0 3286435	26.890123
	3.0 2070111	16.937971
In:	# DISTRIBUIÇÃO DAS CLASSES X PORTE # Codificação suprimida (similar à do item anterior)	



▪ **Variável Atividade Maior do que 10 Anos (ATIV\_>10\_ANOS):**

Observa-se que 70% dos débitos de pessoa jurídica são de empresas com mais de 10 anos de atividade. Ou seja, são empresas mais estabelecidas e estáveis, o que se pressupunha, no entendimento do autor, maior capacidade de pagamento e adimplência. No entanto, aqui se verifica o contrário. Empresas com menos de 10 anos de atividade apresentaram uma taxa de parcelamento de 26%, enquanto as com mais de 10 anos apresentam uma taxa de 22,3%, mais próxima à média geral.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)	
Out:	Frequência	Porcentagem(%)
	S 8583880	70.233097
	N 3638107	29.766903

In:	# DISTRIBUIÇÃO DAS CLASSES X ATIV_>10_ANOS # Codificação suprimida (similar à do item anterior)	
Out:		
	# Taxa de parcelamento x ATIV_>10_ANOS # Codificação suprimida (similar à do item anterior)	
		

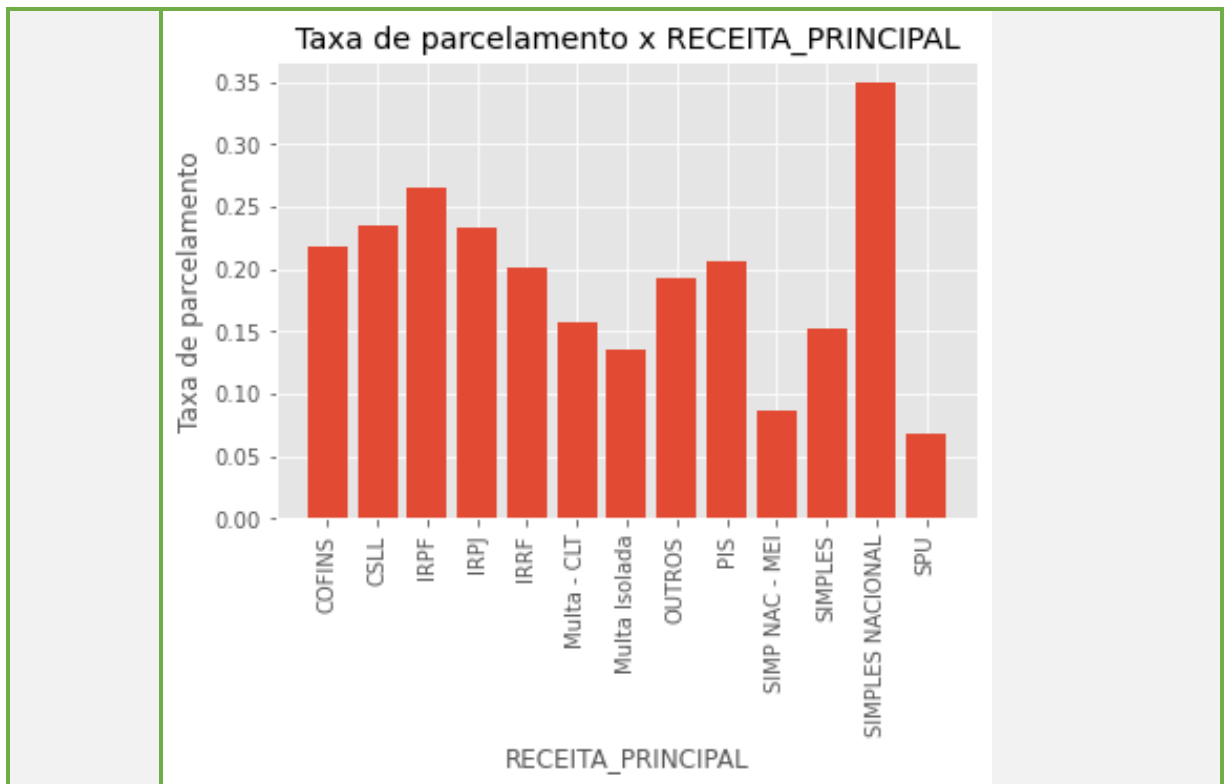
Nos últimos cinco itens analisamos as variáveis exclusivas para pessoa jurídica. A partir deste ponto, os próximos itens se referem aos dois tipos de pessoa.

▪ **Variável Receita Principal (RECEITA\_PRINCIPAL):**

Esta variável se refere ao tributo. Observa-se que o SIMPLES NACIONAL corresponde a 17,7% do total de débitos. Se somarmos ao SIMPLES (Federal, programa anterior, que foi substituído) e SIMPLES NAC - MEI, temos 23,1%. São seguidos por IRPF (12%), IRPJ (11%), COFINS (10,7%), CSLL (10,5%), e PIS (6,9%).

Quanto à taxa de parcelamento, novamente o SIMPLES NACIONAL apresenta a maior taxa, com 35%, seguido por IRPF (26,4%), CSLL (23,4%), e IRPJ (23,4%). Assim, se verifica que SIMPLES NACIONAL se destaca tanto pela alta frequência, como pela alta taxa de parcelamento, bem superior aos demais tributos.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)																																												
Out:	<table><thead><tr><th></th><th>Frequência</th><th>Porcentagem(%)</th></tr></thead><tbody><tr><td><b>SIMPLES NACIONAL</b></td><td>2662456</td><td>17.758086</td></tr><tr><td><b>IRPF</b></td><td>1810681</td><td>12.076905</td></tr><tr><td><b>IRPJ</b></td><td>1651907</td><td>11.017912</td></tr><tr><td><b>OUTROS</b></td><td>1644457</td><td>10.968222</td></tr><tr><td><b>COFINS</b></td><td>1612494</td><td>10.755035</td></tr><tr><td><b>CSLL</b></td><td>1587306</td><td>10.587036</td></tr><tr><td><b>PIS</b></td><td>1042228</td><td>6.951467</td></tr><tr><td><b>Multa Isolada</b></td><td>809705</td><td>5.400582</td></tr><tr><td><b>Multa - CLT</b></td><td>767020</td><td>5.115881</td></tr><tr><td><b>SIMP NAC - MEI</b></td><td>552523</td><td>3.685226</td></tr><tr><td><b>IRRF</b></td><td>317248</td><td>2.115985</td></tr><tr><td><b>SPU</b></td><td>278586</td><td>1.858117</td></tr><tr><td><b>SIMPLES</b></td><td>256311</td><td>1.709547</td></tr></tbody></table>				Frequência	Porcentagem(%)	<b>SIMPLES NACIONAL</b>	2662456	17.758086	<b>IRPF</b>	1810681	12.076905	<b>IRPJ</b>	1651907	11.017912	<b>OUTROS</b>	1644457	10.968222	<b>COFINS</b>	1612494	10.755035	<b>CSLL</b>	1587306	10.587036	<b>PIS</b>	1042228	6.951467	<b>Multa Isolada</b>	809705	5.400582	<b>Multa - CLT</b>	767020	5.115881	<b>SIMP NAC - MEI</b>	552523	3.685226	<b>IRRF</b>	317248	2.115985	<b>SPU</b>	278586	1.858117	<b>SIMPLES</b>	256311	1.709547
	Frequência	Porcentagem(%)																																											
<b>SIMPLES NACIONAL</b>	2662456	17.758086																																											
<b>IRPF</b>	1810681	12.076905																																											
<b>IRPJ</b>	1651907	11.017912																																											
<b>OUTROS</b>	1644457	10.968222																																											
<b>COFINS</b>	1612494	10.755035																																											
<b>CSLL</b>	1587306	10.587036																																											
<b>PIS</b>	1042228	6.951467																																											
<b>Multa Isolada</b>	809705	5.400582																																											
<b>Multa - CLT</b>	767020	5.115881																																											
<b>SIMP NAC - MEI</b>	552523	3.685226																																											
<b>IRRF</b>	317248	2.115985																																											
<b>SPU</b>	278586	1.858117																																											
<b>SIMPLES</b>	256311	1.709547																																											
In:	# DISTRIBUIÇÃO DAS CLASSES X RECEITA_PRINCIPAL # Codificação suprimida (similar à do item anterior)																																												
Out:	<div><p>Distribuição das classes x RECEITA_PRINCIPAL</p><table><thead><tr><th>RECEITA_PRINCIPAL</th><th>PARCELADO N (count)</th><th>PARCELADO S (count)</th></tr></thead><tbody><tr><td>SIMPLES NACIONAL</td><td>1.75</td><td>0.90</td></tr><tr><td>Multa - CLT</td><td>0.65</td><td>0.10</td></tr><tr><td>OUTROS</td><td>1.30</td><td>0.30</td></tr><tr><td>COFINS</td><td>1.25</td><td>0.35</td></tr><tr><td>Multa Isolada</td><td>0.70</td><td>0.10</td></tr><tr><td>CSLL</td><td>1.20</td><td>0.35</td></tr><tr><td>IRPF</td><td>1.30</td><td>0.45</td></tr><tr><td>PIS</td><td>0.85</td><td>0.20</td></tr><tr><td>IRPJ</td><td>1.25</td><td>0.35</td></tr><tr><td>SIMPLES</td><td>0.20</td><td>0.05</td></tr><tr><td>SIMP NAC - MEI</td><td>0.50</td><td>0.05</td></tr><tr><td>IRRF</td><td>0.25</td><td>0.05</td></tr><tr><td>SPU</td><td>0.25</td><td>0.05</td></tr></tbody></table></div>			RECEITA_PRINCIPAL	PARCELADO N (count)	PARCELADO S (count)	SIMPLES NACIONAL	1.75	0.90	Multa - CLT	0.65	0.10	OUTROS	1.30	0.30	COFINS	1.25	0.35	Multa Isolada	0.70	0.10	CSLL	1.20	0.35	IRPF	1.30	0.45	PIS	0.85	0.20	IRPJ	1.25	0.35	SIMPLES	0.20	0.05	SIMP NAC - MEI	0.50	0.05	IRRF	0.25	0.05	SPU	0.25	0.05
RECEITA_PRINCIPAL	PARCELADO N (count)	PARCELADO S (count)																																											
SIMPLES NACIONAL	1.75	0.90																																											
Multa - CLT	0.65	0.10																																											
OUTROS	1.30	0.30																																											
COFINS	1.25	0.35																																											
Multa Isolada	0.70	0.10																																											
CSLL	1.20	0.35																																											
IRPF	1.30	0.45																																											
PIS	0.85	0.20																																											
IRPJ	1.25	0.35																																											
SIMPLES	0.20	0.05																																											
SIMP NAC - MEI	0.50	0.05																																											
IRRF	0.25	0.05																																											
SPU	0.25	0.05																																											
	# Taxa de parcelamento x RECEITA_PRINCIPAL # Codificação suprimida (similar à do item anterior)																																												

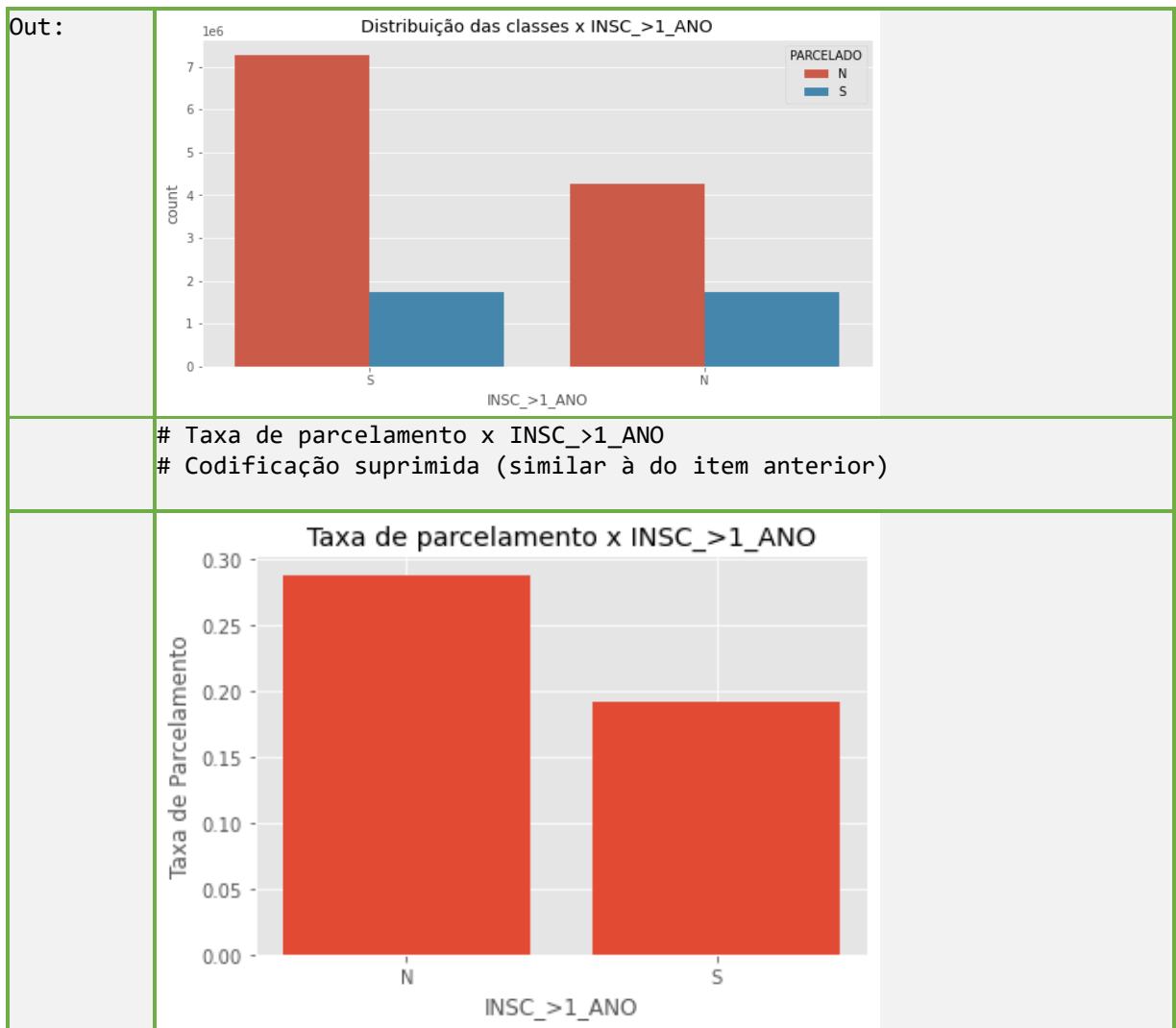


▪ **Variável Inscrição Maior do Que Um Ano (INSC\_>1\_ANO):**

Esta variável diz respeito à antiguidade da dívida. Observa-se que 60% dos débitos estão inscritos em DAU há mais de um ano. É de se esperar que quanto mais antiga uma dívida, menor é seu índice de regularização. E é exatamente o que se observa aqui. As dívidas de curto prazo apresentam uma taxa de parcelamento de 28,7%, enquanto para as de longo prazo a taxa é de apenas 19,2%. É uma diferença considerável, o que sugere que o credor deve reforçar a cobrança nas dívidas geradas há menos tempo.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)	
Out:	Frequência	Porcentagem(%)
	S 8999041	60.021929
	N 5993881	39.978071
In:	# DISTRIBUIÇÃO DAS CLASSES X INSC_>1_ANO # Codificação suprimida (similar à do item anterior)	



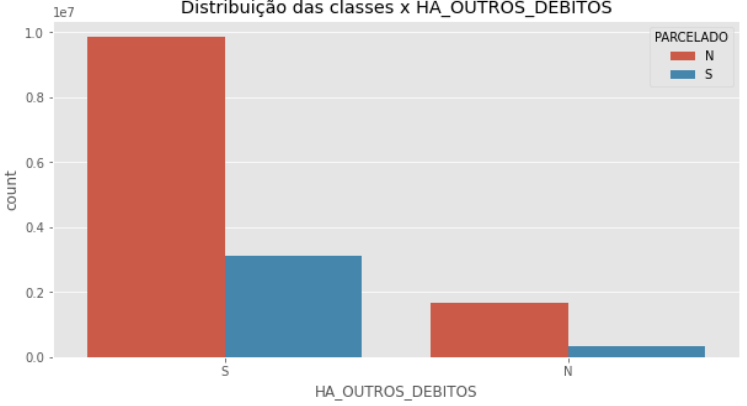
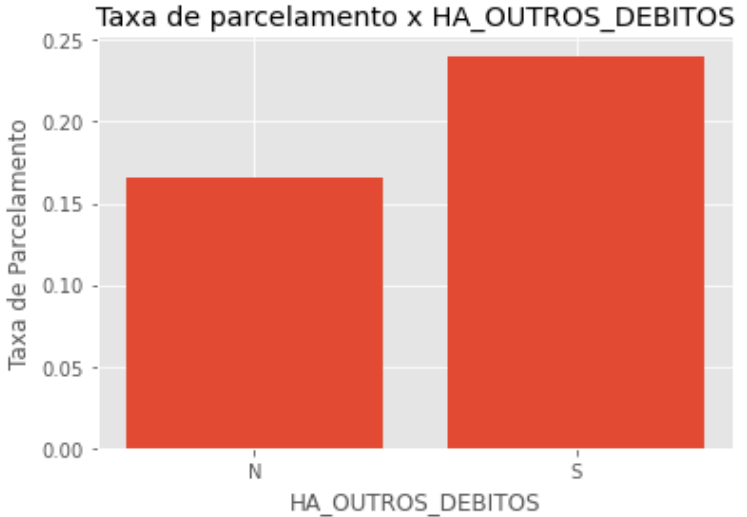


▪ **Variável Há Outros Débitos (HA\_OUTROS\_DEBITOS):**

Essa variável indica quando existe mais de um débito para o mesmo contribuinte. Observa-se que 86,6% dos débitos não são os únicos para um mesmo devedor.

Similar à análise do valor da dívida, para o autor, a quantidade de débitos também sugeriria que quanto maior sua quantidade, menor seu índice de regularização. No entanto, se observa o contrário. Quando há mais de um débito, a taxa de parcelamento é de 24%, contra 16,5%. Essa informação sugere que geralmente o contribuinte opta por parcelar em lote todos os seus débitos.

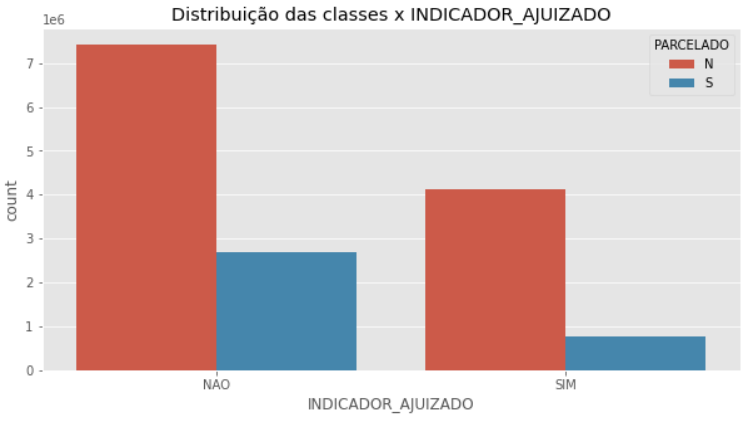
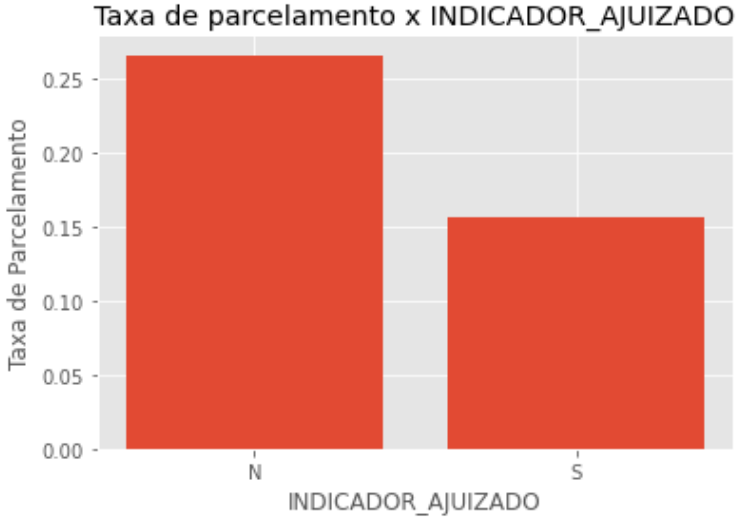
In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)
Out:	Frequência    Porcentagem(%)

	S	12987108	86.621594
	N	2005814	13.378406
In:	# DISTRIBUIÇÃO DAS CLASSES X HA_OUTROS_DEBITOS # Codificação suprimida (similar à do item anterior)		
Out:			
	# Taxa de parcelamento x HA_OUTROS_DEBITOS # Codificação suprimida (similar à do item anterior)		
			

▪ **Variável Indicador de Ajuizado (INDICADOR\_AJUIZADO):**

Aqui se observa que 67,4% dos débitos ainda não foram ajuizados. Ajuizamento significa que a PGFN ingressou na justiça para fins de cobrança judicial. Curiosamente, os débitos não ajuizados possuem uma taxa de parcelamento bem superior, de 26,5%, enquanto os ajuizados apenas 15,6%. Uma hipótese possível é que o ajuizamento está relacionado a outros fatores, tais como: antiguidade (INSC\_>1\_ANO), já que existe um prazo de cobrança amigável antes da cobrança judicial; e valor (VALOR\_FAIXA), já que existe um valor de corte de R\$ 20 mil para o ajuizamento. Para reforçar essa hipótese, veremos no item 4.d (mapa de calor) que,

de fato, o ajuizamento apresenta correlação com as referidas variáveis (embora os coeficientes encontrados sejam considerados fracos). Como já vimos, antiguidade e valor são inversamente proporcionais à taxa de parcelamento.

In:	# CONTAGEM DE VALORES COM PERCENTUAL # Codificação suprimida (similar à do item anterior)										
Out:	<table> <thead> <tr> <th></th><th>Frequência</th><th>Porcentagem(%)</th></tr> </thead> <tbody> <tr> <td>NAO</td><td>10107954</td><td>67.418172</td></tr> <tr> <td>SIM</td><td>4884968</td><td>32.581828</td></tr> </tbody> </table>		Frequência	Porcentagem(%)	NAO	10107954	67.418172	SIM	4884968	32.581828	
	Frequência	Porcentagem(%)									
NAO	10107954	67.418172									
SIM	4884968	32.581828									
In:	# DISTRIBUIÇÃO DAS CLASSES X INDICADOR_AJUIZADO # Codificação suprimida (similar à do item anterior)										
Out:											
	# Taxa de parcelamento x INDICADOR_AJUIZADO # Codificação suprimida (similar à do item anterior)										
											

d) **PARTE 4 – CORRELAÇÃO DE TODAS AS VARIÁVEIS ENTRE SI (MAPA DE CALOR / HEATMAP)**

Esse gráfico permite a análise simultânea da correlação<sup>16</sup> de todas as variáveis entre si. Ressalte-se que este gráfico se aplica somente a variáveis numéricas/contínuas, mas a maioria das nossas variáveis são categóricas. Assim, utilizamos o dataset “07\_dataset\_final\_com\_dados\_contínuos.csv”, onde realizamos a conversão dos dados categóricos para contínuos. No entanto, esta conversão foi realizada ou para valores binários (0 e 1) ou para múltiplos valores em relação à variável alvo. Ou seja, é uma conversão que poderá gerar distorções nesse gráfico, especialmente quando se tratar de comparação entre variáveis que foram convertidas com métodos diferentes.

De qualquer forma, é válida a análise de correlação em relação à variável alvo (PARCELADO). Assim, verifica-se que a maior correlação se deu com a variável COD\_SIT\_CADASTRAL (0.26). Arredondando-se para 0.3, pode-se considerar como correlação fraca.

Observa-se ainda correlação negativa com quatro variáveis, as quais destacam-se INDICADOR\_AJUIZADO (-0.12) e INSC\_>1\_ANO (-0.11). Curiosamente, dentre as variáveis binárias, estas foram a que apresentaram a maior diferença entre suas classes nos seus respectivos gráficos de taxa de parcelamento (N=26,5% x S=15,6%) e (N=28,7% x S=19%).

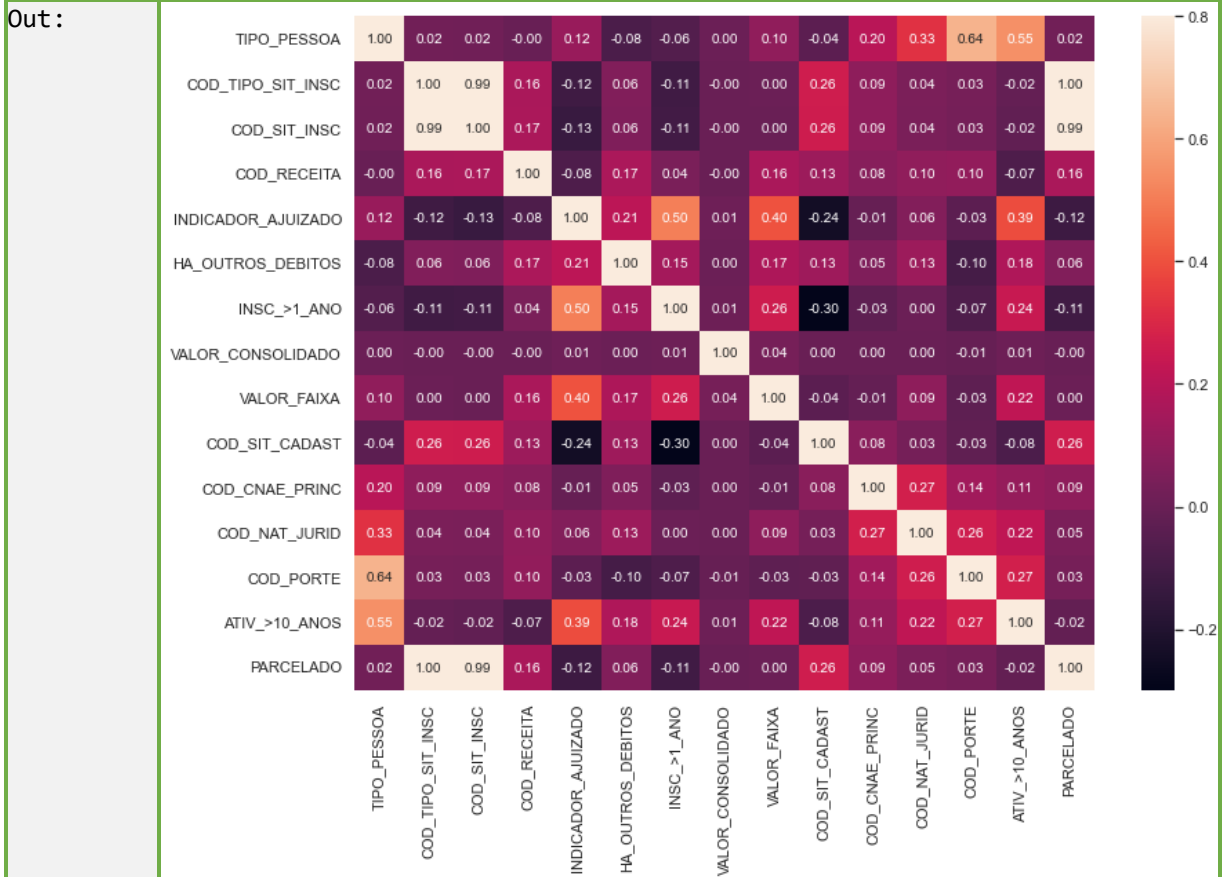
Entre as variáveis binárias, TIPO\_PESSOA e ATIV\_>10\_ANOS apresentaram as menores correlações em relação à variável alvo.

Por fim, destaca-se a correlação da variável INDICADOR\_AJUIZADO com a variável INSC\_>1\_ANO (0.50: moderada). É uma confirmação que, de fato, há correlação entre estas variáveis conforme já apontado anteriormente.

In:	# IMPORTANDO O DATASET df2 = pd.read_csv("07_dataset_final_com_dados_contínuos.csv")
Out:	

<sup>16</sup> Correlação: os coeficientes variam de -1.00 a 1.00, variando de correlação perfeita negativa a correlação perfeita positiva. Zero significa que não há correlação. A partir de +/-0.7 é considerada correlação forte. De +/-0.5 a +/-0.7 correlação moderada. De +/-0.3 a +/-0.5 correlação fraca. De 0 a +/- 0.3 correlação desprezível. (WIKIPEDIA, 2018).

```
In: # APLICANDO-SE O HEATMAP
    corr = df2.corr()
    sns.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white'})
    plt.subplots(figsize=(12, 9))
    sns.heatmap(corr, vmax=.8,annot_kws={'size': 10}, annot=True,
    fmt='.2f');
```



```
In: # Verificando-se a correlação da variável alvo com as demais
    corr_list = corr['PARCELADO'].sort_values(axis=0, ascending=False).iloc[1:]
    corr_list
```

Out:	COD_TIPO_SIT_INSC	0.999333
	COD_SIT_INSC	0.994037
	COD_SIT_CADAST	0.260968
	COD_RECEITA	0.163776
	COD_CNAE_PRINC	0.094466
	HA_OUTROS_DEBITOS	0.059807
	COD_NAT_JURID	0.045565
	COD_PORTE	0.034213
	TIPO_PESSOA	0.021379
	VALOR_FAIXA	0.002001
	VALOR_CONSOLIDADO	-0.003720
	ATIV_>10_ANOS	-0.016432
	INSC_>1_ANO	-0.112324
	INDICADOR_AJUIZADO	-0.119889
	Name: PARCELADO, dtype: float64	

## 5. CRIAÇÃO DE MODELOS DE MACHINE LEARNING

Uma vez realizadas todas as etapas preparatórias, neste capítulo, o principal desse trabalho, após uma breve abordagem teórica, finalmente aplicaremos os algoritmos de *machine learning* para criação de um modelo preditivo dos nossos dados. Assim, criaremos um modelo que, baseado nos atributos selecionados tentarão prever se o débito será ou não parcelado.

### 5.1. Teoria

No início deste trabalho conceituamos brevemente *big data*, *data mining* e *machine learning*. Neste capítulo aprofundaremos os conceitos deste último para, em seguida, aplicarmos no nosso caso prático.

Segundo Facelli e outros (2022) *machine learning* é uma evolução ou subárea da inteligência artificial, que se expandiu na década de 1970. Esta exigia uma maior necessidade de ação humana.

Em virtude da crescente complexidade dos problemas a serem computacionalmente tratados, a velocidade e o volume de dados motivaram o desenvolvimento de ferramentas computacionais mais sofisticadas, mais independentes da intervenção humana. Surgiu, assim, o *machine learning*. Desde então, tem sido largamente utilizado em diversos setores, tais como sistemas de recomendação de acordo com os padrões do usuário (p. ex. indicações de filmes pela Netflix, recomendação de compras de produtos em sites de *e-commerce*), legendas automáticas do Youtube, reconhecimento facial, filtragem de e-mails (p. ex. *spam*), detecção de fraudes por bancos e operadores de cartões de crédito, auxílio no diagnóstico de doenças por meio de análise de dados clínicos, e muito mais.

A autora afirma ainda que as aplicações em ML buscam por modelos capazes de representar o conhecimento através de um conjunto de dados. Em geral, esses conjuntos de dados são estruturados em formato tabular, uma matriz atributo-valor,

em que cada linha representa um objeto e cada coluna representa um atributo (característica ou variável). Estes podem ser divididos em atributos preditivos e atributo alvo, cujo valor rotula o objeto com uma classe ou valor numérico. No nosso caso, o atributo alvo é a variável PARCELADO, e o objeto rotulado é uma classe (S ou N / 1 ou 0).

Como veremos a no tópico a seguir, nem todos os conjuntos de dados possuem atributo alvo. Quando possuem, estão relacionados às tarefas de aprendizado supervisionado. Quando não possuem, estão relacionados às tarefas de aprendizado não-supervisionado.

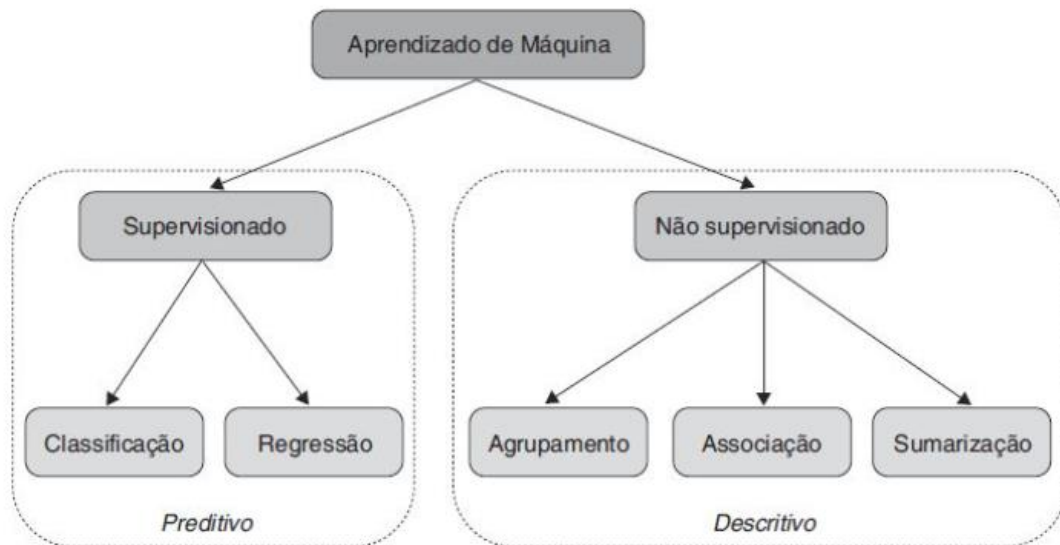
### **5.1.1. Tarefas de Aprendizado**

Ainda segundo Facelli e outros (2022) as tarefas de aprendizado de máquina podem ser divididas em preditivas e descritivas. As primeiras são utilizadas em algoritmos de aprendizado supervisionado, enquanto as segundas de aprendizado não-supervisionado. Estas são as duas principais classificações em *machine learning*. Há ainda outros tipos, mas não é o caso de nos aprofundarmos aqui.

As tarefas preditivas (aprendizado supervisionado) são aplicadas a conjuntos de dados de treinamento rotulados (ou conhecidos) para induzir um modelo preditivo capaz de prever o valor de seu atributo alvo. Subdivide-se em técnicas de classificação (quando o valor do rótulo a ser predito é discreto) e regressão (quando o valor é contínuo).

Já as tarefas de descrição (aprendizado não-supervisionado), os algoritmos analisam dados não rotulados (ou desconhecidos) e extraem padrões. Por sua vez, subdivide-se em agrupamento, associação e sumarização.

## **Figura 3 – Hierarquia Clássica de Aprendizado**



Fonte: Faceli (2022)

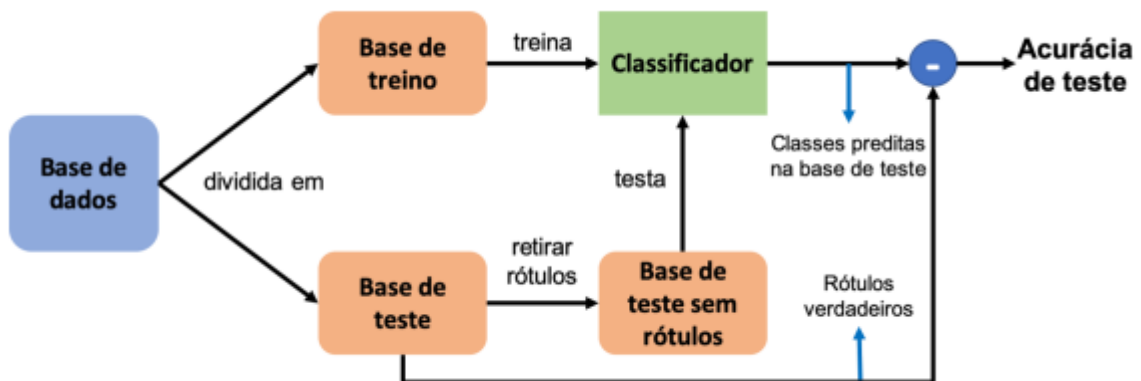
Aplicando-se a classificação acima no nosso trabalho, uma vez que temos dados rotulados (valor do débito, tributo, atividade econômica, etc..) e que desejamos, com base neles, prever se um débito será parcelado (atributo alvo), conclui-se que estamos diante de uma tarefa de **aprendizado supervisionado**. E como o valor do atributo alvo é 1 (parcelado) ou 0 (não-parcelado), ou seja, um dado discreto (número inteiro), temos um caso de **classificação**.

### 5.1.2. Algoritmos de Classificação

De acordo com Escovedo e Koshiyama, os algoritmos de classificação são uns dos mais importantes e mais utilizados em machine learning. Para sua aplicação devemos particionar a base de dados em treino e teste. Na base de treinamento o algoritmo classificador tenta “aprender” como os dados rotulados (ou features) se relacionam com o atributo alvo. Em seguida, este aprendizado é submetido a um teste com os dados de teste (já previamente excluídos dos resultados do atributo alvo) a fim de tentar prevê-los. Ao final, compara-se o resultado previsto com os resultados reais da base de teste e mede-se sua taxa de acertos. Uma das principais métrica é taxa de acurácia. (ESCOVEDO; KOSHIYAMA, 2020).

**Figura 4 - Resumo de um problema de Classificação**





Fonte: ESCOVEDO; KOSHIYAMA, 2020.

As duas principais formas de partição são Hold-out e Cross-validation. Na técnica de **Hold-out** o dataset é dividido em duas partes, uma para treino e outra para teste. Primeiramente trabalha-se no treinamento e ao final compara-se com a de teste. É de fácil aplicação e rápido processamento. No entanto, com essa divisão há risco de as amostragens não representarem todo o conjunto de dados e apresentarem resultados inconsistentes, decorrentes, por exemplo, de overfitting<sup>17</sup>. Utilizaremos essa técnica de partição para processar todos os algoritmos de machine learning, a fim de escolher o melhor avaliado.

Já a **Cross-validation** (validação cruzada) divide o dataset em no mínimo 2 partições ou dobras (folds), embora normalmente se utilize 5 ou 10. Em seguida as partições são mutuamente treinadas e testadas, em revezamento. Por utilizar toda a base, tende a ter uma maior representatividade dos dados e apresentar melhores resultados. Em contrapartida, exige maior capacidade de processamento. Utilizaremos esta técnica apenas no algoritmo escolhido, tanto na fase de *tuning*, quanto na versão final do modelo.

A seguir apresentamos sinteticamente diversos algoritmos de classificação. Entre parênteses seu nome correspondente na biblioteca scikit-learn. É importante

---

<sup>17</sup> "Dizemos que ocorreu overfitting quando o classificador se ajustou em excesso ao conjunto de treinamento". (ESCOVEDO; KOSHIYAMA, 2020)

ressaltar que não existe algoritmo melhor do que outro. Depende muito dos dados utilizados e qual o objetivo do seu trabalho. Assim, é indispensável conhecer bem o problema levantado, os dados disponíveis, e escolher adequadamente o algoritmo.

#### **5.1.2.1. Árvore de Decisão (DecisionTreeClassifier)**

É considerado um dos algoritmos mais simples e didáticos, bem como um dos mais antigos. Podem ser usados em problemas de classificação e regressão. Basicamente, usa amostras das características dos dados para criar regras de decisão no formato de árvore. (ESCOVEDO; KOSHIYAMA, 2020).

Um de suas desvantagens é sua sensibilidade, quando uma alteração nos dados pode gerar resultados bem diferentes. Ademais, pode não apresentar a melhor acurácia em relação a modelos mais recentes.

#### **5.1.2.1. KNN (KNeighborsClassifier)**

A ideia principal do KNN (k-Nearest Neighbours ou, em português, k-Vizinhos Mais Próximos) é considerar que os exemplos vizinhos são similares ao exemplo cuja informação se deseja inferir. Também é válida para problemas de classificação e regressão. (ESCOVEDO; KOSHIYAMA, 2020).

Para Harrison (2020) esse algoritmo classifica com base em distância para algum número (k) de amostras de treinamento. Essa família de algoritmos é chamada de aprendizado baseado em instâncias, ou seja, não há modelo para aprender, pois assume que a distância é suficiente para inferência.

Suas vantagens são fácil aplicação, por ter poucos parâmetros, além de seus bons resultados. Por outro lado, demanda alto custo computacional.

#### **5.1.2.3. Naïve Bayes (GaussianNB)**

Naive Bayes é um classificador probabilístico que assume independência entre as características (atributos/features) dos dados. É bastante utilizado para classificação de texto, como a identificação de spam. Uma vantagem desse algoritmo é que ele pode treinar um modelo com um pequeno número de amostras, além de ser capaz de trabalhar com dados com muitas características. Possui ainda um baixo custo computacional. Por outro lado, às vezes pode ser necessário levantar as interações entre características, o que ele não oferece. (HARRISON, 2020).

#### **5.1.2.4. Support Vector Machine (SVM)**

Support Vector Machine (SVM), ou Máquina de Vetor de Suporte, é um dos algoritmos mais efetivos para classificação. Exigem poucos ajustes, tendem a apresentar boa acurácia e conseguem modelar fronteiras de decisão complexas e não lineares. Além disso, são menos propensos a overfitting se comparados com outros métodos. Pode ser utilizado em reconhecimento de voz e manuscritos, text mining, entre outras. (ESCOVEDO; KOSHIYAMA, 2020).

No entanto, exige alto custo computacional. Por isso, no nosso caso, como temos uma base de dados de grande volume, não iremos adotá-lo.

#### **5.1.2.5. Random Forest (RandomForestClassifier)**

Conforme o nome sugere, ele cria uma floresta aleatória. Essa floresta é uma combinação (ensemble) de árvores de decisão, com o objetivo de obter uma predição com maior acurácia e mais estabilidade. Também pode ser usada tanto em classificação como regressão. Sua maior limitação é que uma quantidade grande de árvores pode tornar o algoritmo lento (DONGES apud SILVA, 2018).

Outra característica interessante é sua capacidade para medir a importância relativa de cada feature para a predição. Por este motivo ela será usada em nosso trabalho para escolha das melhores e descarte das piores. Esse descarte é importante para evitar overfitting e melhorar desempenho.

#### **5.1.2.6. XGBoost (XGBClassifier)**

XGBoost também se utiliza de combinações de árvores. Ele cria uma árvore fraca e então cria subsequentes para reduzir os erros residuais. Ele tenta capturar e abordar quaisquer padrões nos erros até que pareçam ser aleatórios. (HARRISON, 2020).

É um dos algoritmos mais populares, versáteis, e que apresenta os melhores resultados. Por outro, também tem alto custo computacional.

#### **5.1.2.7. Regressão Logística (LogisticRegression)**

Como o nome sugere, este algoritmo estima probabilidades usando uma função de regressão logística. Embora tenha regressão no nome, é usado para classificação. É de fácil aplicação, interpretação, e muito eficiente em treinar.

#### ***5.1.2. Classes Desbalanceadas***

Um dos problemas que foram apontados na análise exploratória foi o desbalanceamento das classes da variável alvo: 23% de valor 1 (parcelado) contra 77% de valor 0 (não-parcelado).

Apesar de muito comuns em machine learning, o desbalanceamento é um problema que merece uma atenção cuidadosa. Isso porque a maioria dos algoritmos

tendem a ter um excelente resultado em relação à classe majoritária em detrimento da minoritária. Assim, para alguns algoritmos, devemos realizar alguns ajustes.

Basicamente, há duas técnicas de balanceamento:

- a) **Balanceamento em termos de dados:** antes de ser submetido ao algoritmo classificador, o dataset deve sofrer um ajuste de modo a balancear suas classes. Por sua vez, pode ser feito de duas formas:
  - i. Undersampling: consiste em reduzir a classe majoritária para se igualar à minoritária.
  - ii. Oversampling: é o inverso, consiste em aumentar a minoritária para se igualar à majoritária.
- b) **Balanceamento em termos de algoritmo:** o balanceamento é realizado pelo próprio algoritmo classificador.

Após alguns testes, observamos que ambas as técnicas trouxeram resultados parecidos. Assim, por praticidade, optamos pelo balanceamento em termos de algoritmo. Dos algoritmos selecionados para teste, apenas regressão logística, árvore de decisão, random forest e XGBoost possuem parâmetros de balanceamento, sendo que apenas os três primeiros têm opção de ajuste automático (`class_weight='balanced'`). Já o XGBoost deve ser calculado. Seu desenvolver sugere utilizar a razão da classe majoritária sobre a minoritária, o que, no nosso caso, resulta num índice de 3.3.

Outra consequência importante decorrente do desbalanceamento é que a principal métrica de avaliação de resultado, qual seja, a acurácia, fica totalmente prejudicada. Mesmo realizando o balanceamento, por quaisquer técnicas, o problema persiste. Assim, devemos buscar outras formas de avaliação. Conforme será justificado no próximo capítulo, a **F1-score** é melhor métrica de avaliação nestes casos.

## 5.2. Prática

A prática será realizada no notebook 09\_machine\_learning.ipynb, sendo que os principais comandos serão reproduzidos aqui. Dos 7 algoritmos apresentados, apenas o SVM não será testado, em virtude da sua alta demanda de processamento.

Em síntese, teremos as seguintes etapas:

- a) Preparo inicial: importação das bibliotecas e do arquivo '07\_dataset\_final\_com\_dados\_contínuos.csv', além de algumas verificações e ajustes;
- b) Seleção das features (variáveis) mais importantes;
- c) Treinamento dos diversos algoritmos de machine learning e escolha do melhor;
- d) Tuning do modelo escolhido;
- e) Preparação da versão final do modelo;

### a) Preparo inicial

```
In: # Importando as bibliotecas necessárias

from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score,
precision_score, recall_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

	<pre> from sklearn.metrics import matthews_corrcoef import pickle from joblib import dump import pandas as pd import numpy as np import seaborn as sns import matplotlib.pyplot as plt from matplotlib import style style.use('ggplot') </pre>
Out:	
In:	<pre> # Importando o dataset df=pd.read_csv('07_dataset_final_com_dados_contínuos.csv') </pre>
Out:	

## b) Seleção das features (variáveis) mais importantes

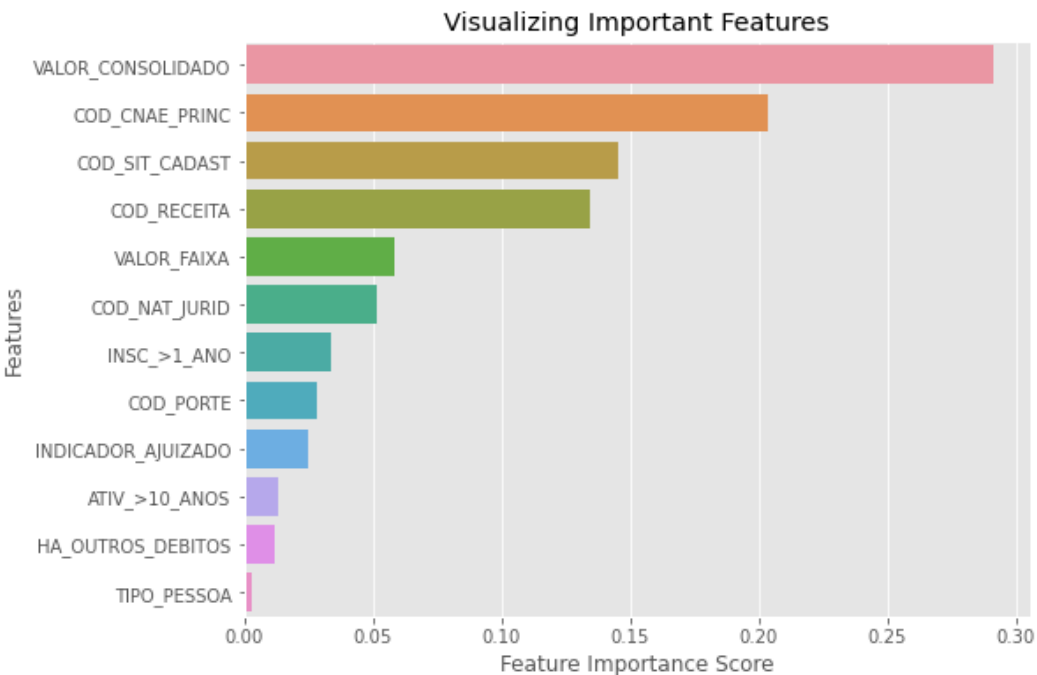
A seleção das features é parte de um processo denominado redimensionamento. Este consiste em mais uma técnica de redução do dataset não apenas para melhorar a performance de processamento, mas também a própria qualidade do modelo, já que a alta dimensionalidade pode superajustá-lo ao treinamento (ALURA, 2022).

Para divisão dos dados em treino e teste utilizaremos a função *train\_test\_split*, que divide o dataset em duas partes. Este instanciamento será aproveitado também no próximo passo (alínea C, para treinamento de diversos algoritmos candidatos).

Treinaremos um modelo de machine learning especificamente para essa seleção das *features*. Segundo Clésio (2015), um dos algoritmos mais indicados para este fim é o Random Forest, o qual adotamos.

Conforme gráfico abaixo, se observa que 4 features apresentam importância menor do que 2%. Assim, as excluiremos. São elas: HA\_OUTROS\_DEBITOS, ATIV\_>10\_ANOS, TIPO\_PESSOA, VALOR\_FAIXA.

In:	<pre> # Separando os atributos e a classe do dataset em X e y: X = df X = df.drop('PARCELADO', axis=1) #removendo a variável target de X y = df['PARCELADO'] # variável alvo/target  # Particionando em conjuntos de treino e teste mantendo a proporção de classes </pre>
-----	--

	<pre> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, stratify=y)  # Criando e treinando classificador para visualizar as features mais im- portantes para o modelo clf_feature_import = RandomForestClassifier(n_estimators=100, ran- dom_state=12) clf_feature_import = clf_feature_import.fit(X_train,y_train)  # Visualizando graficamente as features mais importantes feature_imp = pd.Series(clf.feature_importances_,index=previsores.co- lumn).sort_values(ascending=False)  plt.figure(figsize=(8,6)) sns.barplot(x=feature_imp, y=feature_imp.index) plt.xlabel('Feature Importance Score') plt.ylabel('Features') plt.title("Visualizing Important Features") plt.show() </pre>																										
Out:	 <table border="1"> <thead> <tr> <th>Features</th> <th>Feature Importance Score</th> </tr> </thead> <tbody> <tr> <td>VALOR_CONSOLIDADO</td> <td>0.29</td> </tr> <tr> <td>COD_CNAE_PRINC</td> <td>0.20</td> </tr> <tr> <td>COD_SIT_CADAST</td> <td>0.15</td> </tr> <tr> <td>COD_RECEITA</td> <td>0.13</td> </tr> <tr> <td>VALOR_FAIXA</td> <td>0.06</td> </tr> <tr> <td>COD_NAT_JURID</td> <td>0.05</td> </tr> <tr> <td>INSC_&gt;1_ANO</td> <td>0.03</td> </tr> <tr> <td>COD_PORTE</td> <td>0.03</td> </tr> <tr> <td>INDICADOR_AJUIZADO</td> <td>0.02</td> </tr> <tr> <td>ATIV_&gt;10_ANOS</td> <td>0.01</td> </tr> <tr> <td>HA_OUTROS_DEBITOS</td> <td>0.01</td> </tr> <tr> <td>TIPO_PESSOA</td> <td>0.005</td> </tr> </tbody> </table>	Features	Feature Importance Score	VALOR_CONSOLIDADO	0.29	COD_CNAE_PRINC	0.20	COD_SIT_CADAST	0.15	COD_RECEITA	0.13	VALOR_FAIXA	0.06	COD_NAT_JURID	0.05	INSC_>1_ANO	0.03	COD_PORTE	0.03	INDICADOR_AJUIZADO	0.02	ATIV_>10_ANOS	0.01	HA_OUTROS_DEBITOS	0.01	TIPO_PESSOA	0.005
Features	Feature Importance Score																										
VALOR_CONSOLIDADO	0.29																										
COD_CNAE_PRINC	0.20																										
COD_SIT_CADAST	0.15																										
COD_RECEITA	0.13																										
VALOR_FAIXA	0.06																										
COD_NAT_JURID	0.05																										
INSC_>1_ANO	0.03																										
COD_PORTE	0.03																										
INDICADOR_AJUIZADO	0.02																										
ATIV_>10_ANOS	0.01																										
HA_OUTROS_DEBITOS	0.01																										
TIPO_PESSOA	0.005																										
In:	<pre> # Selecionando as features de maior importância (acima de 2%): features_selected = [] for feature,importance in feature_imp.iteritems():     if importance &gt; 0.02:         print(f'{feature}: {round(importance * 100)}%')         features_selected.append(feature) </pre>																										
Out:	<pre> VALOR_CONSOLIDADO: 29% COD_CNAE_PRINC: 20% COD_SIT_CADAST: 15% COD_RECEITA: 13% VALOR_FAIXA: 6% COD_NAT_JURID: 5% INSC_&gt;1_ANO: 3% COD_PORTE: 3% INDICADOR_AJUIZADO: 3% </pre>																										



### c) Treinando os algoritmos candidatos de Machine Learning

Aqui realizaremos, em um único código, o treinamento dos nossos 6 algoritmos de *machine learning* candidatos, e os resultados de diversos indicadores serão retornados.

Como já levantado, nosso *dataset* possui classes desbalanceadas, já que os débitos parcelados correspondem a apenas 23% do total. Dos referidos 6 modelos, apenas dois (KNN e GaussianNB) não tem parâmetros nativos para realizar o balanceamento. Assim, utilizaremos esse recurso apenas nos modelos que possuem esse recurso. Nos algoritmos de regressão logística, árvore de decisão e random forest, o parâmetro permite um balanceamento automático. Já no XGBoost ele deve ser especificado manualmente utilizando-se um índice que deve ser a razão entre a classe majoritária sobre a minoritária. No nosso caso:  $77 / 23 = 3.3$ .

Por fim, utilizaremos o recurso de loop do Python para percorrer cada um dos algoritmos, treinar cada modelo, medir os escores dos indicadores acurácia, precisão, recall, f1, mcc e roc\_auc, e calcular sua a média final por modelo (ver tabela de saída do código).

In:	<pre># Redividindo o dataset, desta vez apenas as features selecionadas X = df[features_selected] # Atualizando X apenas para features selecionadas y = df['PARCELADO'] # y permanece o mesmo</pre>
Out:	
In:	<pre># Testando diversos algoritmos candidatos de Machine Learning  print('TESTANDO ALGORITMOS COM BALANCEAMENTO') print('MODELO:', 'ACCURACY', 'PRECISIO', 'RECALL ', 'F1 ', 'MCC ', 'ROC_AUC ', 'MEDIA GERAL')  clf_choose_model = [] clf_choose_model.append(('LRe', LogisticRegression(class_weight='balanced'))), clf_choose_model.append(('Gau', GaussianNB())), clf_choose_model.append(('DTr', DecisionTreeClassifier(class_weight='balanced'))), clf_choose_model.append(('RFo', RandomForestClassifier(class_weight='balanced'))), clf_choose_model.append(('KNN', KNeighborsClassifier())),</pre>

	<pre> clf_choose_model.append(('XGB', XGBClassifier(scale_pos_weight=3.3)))  resultado_accuracy = [] resultado_precision = [] resultado_recall = [] resultado_f1 = [] resultado_mcc = [] resultado_roc_auc = []  resultado_geral = [] nomes = []  # Percorrendo cada um dos modelos, treinando-o e avaliando-o for nome, modelo in clf_choose_model:     modelo = modelo.fit(X_train,y_train)     y_pred_choose_model = modelo.predict(X_test)     ttsplit_result_accuracy = accuracy_score(y_test, y_pred_choose_model)     ttsplit_result_precision = precision_score(y_test, y_pred_choose_model)     ttsplit_result_recall = recall_score(y_test, y_pred_choose_model)     ttsplit_result_f1 = f1_score(y_test, y_pred_choose_model)     ttsplit_result_mcc = matthews_corrcoef(y_test, y_pred_choose_model)     ttsplit_result_roc_auc = roc_auc_score(y_test, y_pred_choose_model)      ttsplit_result_geral = (ttsplit_result_accuracy + ttsplit_result_precision + ttsplit_result_recall + ttsplit_result_f1 + ttsplit_result_mcc + ttsplit_result_roc_auc)/6      resultado_accuracy.append(ttsplit_result_accuracy)     resultado_precision.append(ttsplit_result_precision)     resultado_recall.append(ttsplit_result_recall)     resultado_f1.append(ttsplit_result_f1)     resultado_mcc.append(ttsplit_result_mcc)     resultado_roc_auc.append(ttsplit_result_roc_auc)      resultado_geral.append(ttsplit_result_geral)     nomes.append(nome)  # retornando a média das validações, bem como seu desvio padrão texto = "%s : %f %f %f %f %f %f %f" % (nome, ttsplit_result_accuracy.mean(), ttsplit_result_precision.mean(), ttsplit_result_recall.mean(), ttsplit_result_f1.mean(), ttsplit_result_mcc.mean(), ttsplit_result_roc_auc.mean(), ttsplit_result_geral.mean()) print(texto) </pre>
Out:	<pre> TESTANDO ALGORITMOS COM BALANCEAMENTO MODELO: ACCURACY PRECISIO RECALL  F1      MCC      ROC_AUC  MEDIA GE- RAL LRe   : 0.596118 0.336733 0.777311 0.469903 0.269333 0.659608 0.518168 Gau   : 0.549030 0.316148 0.823882 0.456951 0.249824 0.645338 0.506862 DTr   : 0.652478 0.347128 0.577937 0.433738 0.218830 0.626359 0.476078 RFo   : 0.666951 0.364226 0.598471 0.452850 0.248201 0.642956 0.495609 KNN   : 0.748869 0.426418 0.262164 0.324700 0.189186 0.578327 0.421611 XGB   : 0.633949 0.359562 0.754634 0.487055 0.296963 0.676237 0.534733 </pre>

- **Comparando-se os resultados dos algoritmos candidatos:**

Conforme já explanado anteriormente, não existe algoritmo melhor do que o outro. É necessário saber qual o que melhor se encaixa nos seus dados, com o objetivo desejado. Escolhemos os 6 escores acima porque entendemos que seriam o que nos proporcionariam essa informação.

Um dos principais escores em *machine learning*, senão o principal, é a acurácia. No entanto, no nosso caso de classes desbalanceadas, ele fica um pouco prejudicado, pois pode ser muito bom em prever a classe majoritária, camuflando o mal desempenho da minoritária. Alguns escores são conhecidos por serem inversamente proporcionais, como é o caso da acurácia e o recall (*trade-off*). Assim, considerando-se a especificidade do nosso trabalho, daremos atenção especial a três: `f1_score`, `mcc` e `roc_auc_score`. Estes são mais adequados para se analisar datasets com classes desbalanceadas.

Observa-se que o algoritmo XGBoost obteve os melhores resultados, com um escore médio geral de 0.53, seguido por regressão logística com 0.51. O pior resultado tivemos com KNN, com 0.42.

O XGBoost não teve a melhor acurácia, com 0.63, e sim o KNN, com 0.74. Por outro lado, teve os melhores resultados em `f1_score` de 0.48, `mcc_score` de 0.29 e `roc_auc_score` de 0.67. Assim, este é o algoritmo escolhido.

#### **d) Fazendo o *tuning* do modelo**

Uma vez escolhido o algoritmo, passaremos a tentar otimizá-lo alterando-se seus hiperparâmetros, utilizando as configurações de parâmetros que o próprio algoritmo dispõe. Como são muitos hiperparâmetros e cada um possui diversas possibilidades de ajuste, utilizaremos a função `RandomizeSearchCV` para realizar algumas combinações e retornar a que apresenta o melhor escore. No item anterior justificamos que `f1_score`, `mcc_score` e `roc_auc_score` são os três melhores para avaliar classes desbalanceadas. Mas como a referida função permite escolher apenas um, optamos pelo `f1_score`. Assim, definimos este como escore a ser otimizado (`scoring='f1'`).

Uma alternativa bastante comum e até mais utilizada é o GridSearchCV, mas este exige mais recurso computacional.

O resultado encontrado foi: Melhor parâmetro: {'scale\_pos\_weight': 3.3, 'objective': 'reg:logistic', 'min\_split\_loss': 0.5, 'min\_child\_weight': 1, 'max\_depth': 8, 'max\_delta\_step': 10, 'learning\_rate': 0.1, 'booster': 'dart'}, com um f1\_score de 0.48. Observa-se que é o mesmo score encontrado no item anterior (c) Treinando os algoritmos candidatos de Machine Learning).

In:	<pre> # Fazendo o tuning do modelo # Definindo um dicionário que recebe as listas de parâmetros e valores # Para visualizar todos os parâmetros, acesse (https://xgboost.readthedocs.io/en/stable/parameter.html)  parametros_grid = {'objective':['reg:squarederror', 'reg:logistic', 'binary:logistic', 'binary:logitraw', 'binary:hinge'],                     'booster':['gbtree', 'dart'],                     'scale_pos_weight':[1, 2, 3.3],                     'learning_rate':[0.1, 0.3, 0.5],                     'min_split_loss':[0, 0.1, 0.3, 0.5],                     'max_depth':[4, 6, 8],                     'min_child_weight':[1, 2],                     'max_delta_step':[0,5,10]                     }  # Criando o modelo clf_grid = XGBClassifier()  # Definindo K kfold = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)  # Testando diferentes combinações com os parâmetros grid = RandomizedSearchCV(estimator=clf_grid, param_distributions=parametros_grid, cv=kfold, scoring='f1') grid.fit(X, y)  # Print do resultado print("Grid scores on development set:") means = grid.cv_results_['mean_test_score'].round(5) stds = grid.cv_results_['std_test_score'].round(5)  for mean, std, params in zip(means, stds, grid.cv_results_['params']):     print(f'mean:{mean},std:{std},params:{params}') print() print(f'Melhor parâmetro:{grid.best_params_}, Score:{grid.best_score_}') </pre>
Out:	<pre> Melhor parâmetro: {'scale_pos_weight': 3.3, 'objective': 'reg:logistic', 'min_split_loss': 0.5, 'min_child_weight': 1, 'max_depth': 8, 'max_delta_step': 10, 'learning_rate': 0.1, 'booster': 'dart'}, Score:0.48477567696982043 </pre>

### e) Preparando a Versão Final do Modelo:

De posse do resultado do *tuning*, aplicaremos os parâmetros encontrados no algoritmo escolhido. Logo em seguida, faremos outras avaliações.

Ressalte-se que, desta vez, para treinamento do modelo, não dividiremos em treino e teste. Utilizaremos todo o dataset, já excluído das *features* não selecionadas (`clf = clf.fit(X, y)`).

A divisão em treino e teste ocorrerá apenas para realização da predição através da função *cross\_val\_predict*, que é uma função de validação cruzada com predição. No caso, optamos por dividir em 5 partes (`n_splits=5`). A variável `y_pred` receberá o resultado dessa predição.

In:	<pre># Aplicando algoritmo com os melhores hiperparâmetros  # Separando os atributos e a classe do dataset em X e y: X = df[features_selected] # Reatualizando X apenas para selecionados y = df['PARCELADO'] # y permanece o mesmo  # Instanciando o algoritmo classificador escolhido (XGBoost) clf = XGBClassifier(scale_pos_weight=3.3, objective='reg:logistic', min_split_loss=0.5, min_child_weight=1, max_depth=8, max_delta_step=10, learning_rate=0.1, booster='dart')  # Treinando o algoritmo classificador clf = clf.fit(X, y)  # Realizando a predição do algoritmo utilizando-se de cross-validation y_pred = cross_val_predict(clf, X, y, cv=5)</pre>
Out:	

In:	<pre># Métricas de avaliação das predições  print('Acurácia:',accuracy_score(y, y_pred)) print('Precision score:',precision_score(y, y_pred)) print('Recall score:',recall_score(y, y_pred)) print('F1 score:',f1_score(y, y_pred)) print('MCC score:',matthews_corrcoef(y, y_pred)) print('ROCAUC score:',roc_auc_score(y, y_pred))</pre>
Out:	<pre>Acurácia: 0.6289213456466843 Precision score: 0.3563565688900503 Recall score: 0.7583150870607862 F1 score: 0.48486127931911865</pre>

	MCC score: 0.29352794437581625 ROCAUC score: 0.6742612594169738
--	--

## 6. APRESENTAÇÃO DOS RESULTADOS

Neste capítulo, faremos um breve resumo do trabalho, avaliaremos os resultados obtidos no capítulo anterior e finalmente, a conclusão.

Uma boa forma de sintetizar o que foi feito neste trabalho é utilizar o modelo de fluxo de trabalho de ciência de dados proposto por Vasandani (2019), conforme Figura 5. A autora sugere que, ao se iniciar um projeto, primeiramente devemos nos focar nos seus objetivos, para depois trabalhar para alcançá-los.

**Figura 5 - Fluxo de Trabalho de Ciência de Dados**

**Data Science Workflow Canvas\***  
Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!

Title:		
<b>1 Problem Statement</b> What problem are you trying to solve? What larger issues do the problem address?  Qual o perfil débitos parcelados na dívida ativa da União?  Como prever se um débito será parcelado?	<b>2 Outcomes/Predictions</b> What prediction(s) are you trying to make? Identify applicable predictor (X) and/or target (y) variables.  Variáveis de predição/features (X): valor do débito, tributo, situação, tipo de pessoa, tempo do débito, ajuizamento, ramo de atividade, natureza jurídica, porte, tempo de atividade  Variável alvo (y): parcelado (S/N)	<b>3 Data Acquisition</b> Where are you sourcing your data from? Is there enough data? Can you work with it?  Os dados são extraídos de 2 órgãos governamentais: RFB e PGFN.  Não há dados públicos suficientes para um bom modelo.  Os dados são extremamente volumosos, exigiu muito tempo de processamento. Mas foi possível trabalhá-los.
<b>4 Modeling</b> What models are appropriate to use given your outcomes?  Algoritmos supervisionados de machine learning.  Após testes com diversos algoritmos, optamos por XGBoost.	<b>5 Model Evaluation</b> How can you evaluate your model's performance?  Métricas de avaliação do modelo: - accuracy_score - precision_score - recall_score - f1_score - roc_auc_score	<b>6 Data Preparation</b> What do you need to do to your data in order to run your model and achieve your outcomes?  Fusão das bases de débitos e dados cadastrais de CNPJ.  Adaptação/criação de variáveis.  Conversão para dados contínuos.  Encontrar a melhor performance do modelo de predição através de diversos testes.

**✓ Activation**  
When you finish filling out the canvas above, now you can begin implementing your data science workflow in roughly this order.

1 Problem Statement → 2 Data Acquisition → 3 Data Prep → 4 Modeling → 5 Outcomes/Preds → 6 Model Eval

\* Note: This canvas is intended to be used as a starting point for your data science projects. Data science workflows are typically nonlinear.

Conceptualized by Jasmine Vasandani using notes from General Assembly's Data Science Immersive. Format inspired by Business Model Canvas.

### 6.1. Avaliação dos resultados

Como vimos, optamos pela escolha do algoritmo XGBoost para construção final do modelo. Agora, vamos analisá-lo através das métricas escolhidas.

TABELA 6 – AVALIAÇÃO DO MODELO	
Acurácia	0.6289213456466843
Precision score	0.3563565688900503
Recall score	0.7583150870607862
F1 score	0.48486127931911865
MCC score	0.29352794437581625
ROCAUC score	0.6742612594169738

Fonte: elaboração própria

**a) Acurácia:**

Acurácia é taxa de acertos de predição de todas as classes sobre o total da amostra. Nosso modelo apresentou uma taxa de acurácia de 0.62, o que pode ser considerado um resultado razoável.

No entanto, apesar de ser uma das principais métricas de avaliação, a acurácia não é indicada para avaliar dados desbalanceados, como é o nosso caso. Isso porque ela pode ser muito eficaz em prever os dados da classe majoritária (classe 0, não parcelado), e péssima para prever a classe minoritária (classe 1, parcelado). Assim, teremos que analisar a acurácia junto com as demais métricas.

**b) Precisão**

Precisão é a razão entre os verdadeiros positivos (tp) e a soma dos verdadeiros positivos (tp) mais falso positivo (fp). Sua fórmula é  $[ tp / (tp + fp) ]$ . Ou seja, aqui estamos mensurando apenas classe positiva (ou classe 1, parcelado). Ao contrário do resultado anterior, essa métrica apresentou um péssimo resultado: 35%.

**c) Recall**

Recall ou revocação ou sensibilidade pode ser entendida como o inverso da precisão: dos que eram realmente positivos quantos foram acertados? É dada pela seguinte fórmula  $[ tp / (tp + fn) ]$ . Aqui encontramos um bom resultado: 75%.



O objetivo desta métrica é detectar os positivos mesmo renunciando à precisão. Normalmente, quanto maior o recall, menor a precisão, e vice-versa.

São comuns os exemplos de importância desse indicador na área médica, onde prever erradamente como falso positivo é muito menos problemático do que um falso negativo. Pode custar vidas.

No nosso caso, entendo que acurácia e recall são de igual importância no nosso trabalho. Assim, vale a pena procurar uma taxa de maior equilíbrio entre ambas. E é nesse ponto que a próxima métrica, a F1-score, pode nos ajudar.

**d) F1-score**

Também conhecida como F-score ou F-measure, F1-score é a média harmônica entre a precisão e o recall. Sua fórmula é  $[ F1 = 2 * (precision * recall) / (precision + recall) ]$ .

Como adiantando, é muito útil quando consideramos precisão e recall de igual importância. Assim, foi escolhida como nossa principal métrica. Por esse motivo foi o indicador utilizado para processamento do *tuning*. Portanto, precisão e o recall estão equilibrados em seu melhor ponto possível.

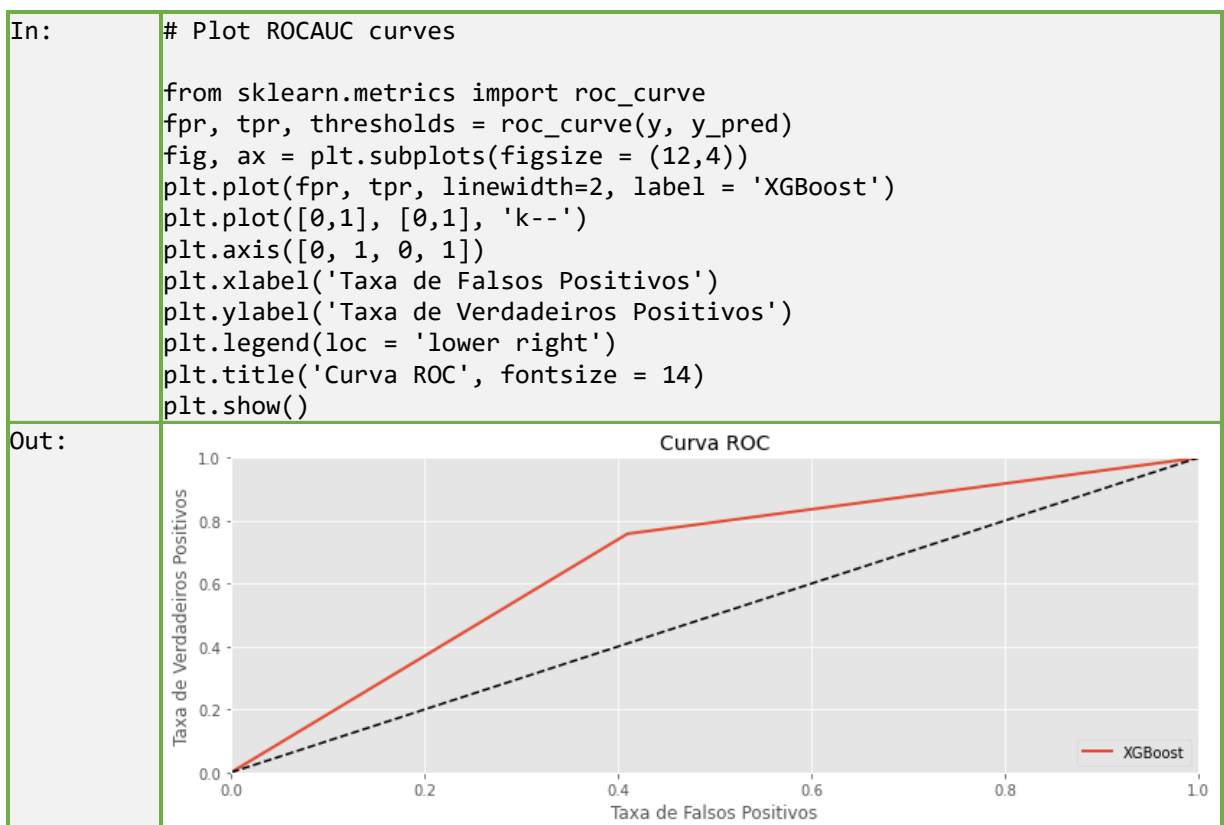
**e) MCC**

Outra métrica indicada para dados desbalanceados é o MCC, sigla em inglês para Matthews Correlation Coefficient. Utiliza todas as possibilidades da matriz de confusão (tn, fp, fn e tp – serão apresentados a seguir) para calcular seu índice, que varia de -1 a 1. Se maior do que 0, tem mais acertos do que erros, e vice-versa. Ou seja, quanto mais próximo de 1, melhor (GUALBERTO, 2020). O índice encontrado no nosso trabalho foi **0,29**, ou seja, tivemos mais acertos do que erros, embora ainda seja um índice apenas regular.

**f) ROC-AUC**

Sigla para Area Under the Receiver Operating Characteristic Curve. Segundo Gerola (2020), esta métrica visa responder à seguinte pergunta: qual é a chance de um exemplo positivo ter um score (previsão) maior do que um negativo? Devemos utilizar quando for mais importante medir os positivos acima dos negativos do que prever a probabilidade real do evento.

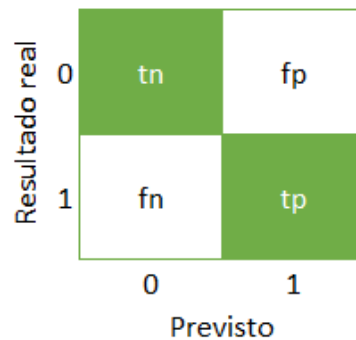
A curva ROC é medida em AUC (Area Under the Curve). O valor mínimo deve ser 0.5. Se for menor significa que o modelo está pior do que se fosse aleatório. No nosso modelo obtivemos uma taxa também razoável de **0.67**. Plotamos o gráfico abaixo que dá uma boa ideia de como nosso modelo ficou longe do ideal. A linha pontilhada se refere ao valor mínimo 0.5. Quanto mais próxima do canto superior esquerdo, melhor o resultado.



#### g) Matriz de Confusão

Matriz de confusão é uma tabela que nos permite visualizar o resultado real versus previsto em seus quadrantes (no caso de 2 classes): verdadeiros negativos (tn), falsos positivos (fp), falsos negativos (fn), e verdadeiros positivos (tp). O ideal é que a diagonal de verdadeiros (tn e tp) concentre o máximo de valores, ao passo que a diagonal de falsos (fn e fp) o mínimo.

**Figura 6 – Matriz de Confusão com 2 Classes**

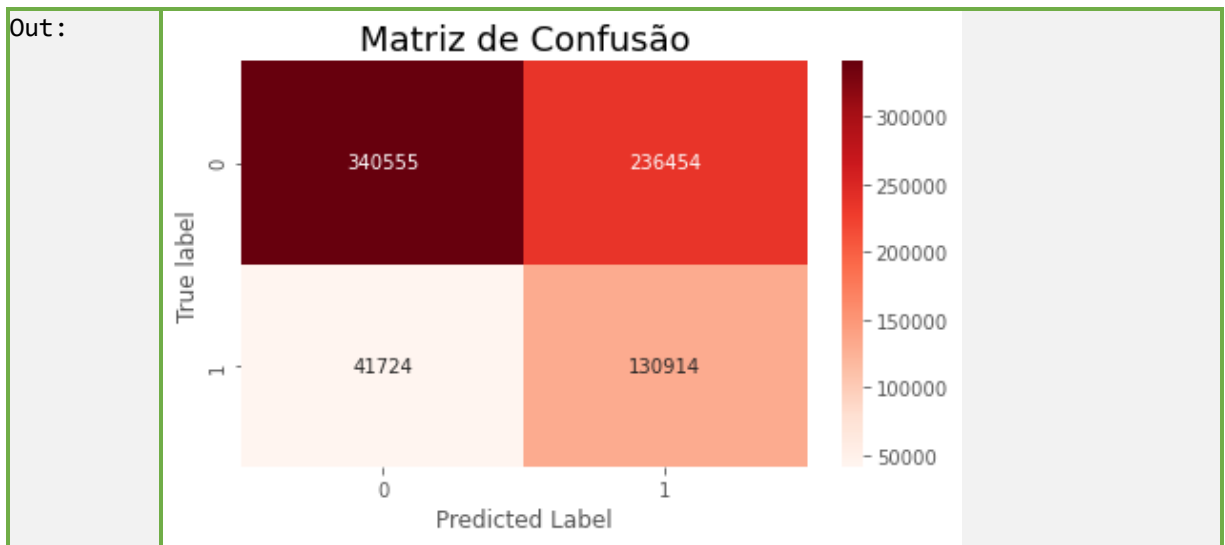


Fonte: elaboração própria

Observe que nossa tabela não atende ao ideal. Até tivemos um bom resultado para tn, fn e tp. No entanto, tivemos um péssimo resultado de fp (falso positivo). Estes foram previstos como classe 1 (parcelado), mas o resultado real foi classe 0 (não-parcelado). Foi bem superior ao tp (quase o dobro). Isso é reflexo da baixa precisão e de uma relativamente alto recall para a classe 1. No entanto, como já ajustamos o F1-score, entendemos que esse é o melhor balanceamento possível do modelo.

```
In: # Plotando a Matriz de confusão

fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y, y_pred), annot=True,
            ax=ax, fmt='d', cmap='Reds')
ax.set_title("Matriz de Confusão", fontsize=18)
ax.set_ylabel("True label")
ax.set_xlabel("Predicted Label")
plt.tight_layout()
```



#### h) Especificidade

De acordo com Scudilio (2020), especificidade é a proporção dos verdadeiros negativos (tn) entre todas as observações que realmente são negativas no seu conjunto de dados. Representa a capacidade de um modelo em prever a classe negativa. Sua fórmula é  $[ tn / (tn + fp) ]$ . Aplicando-se essa fórmula, encontramos uma taxa de **0.72**, o que é um resultado razoável.

#### i) Relatório de Classificação

Esse relatório apresenta um relatório completo com os indicadores precision, recall e f1-score para cada uma das classes. Observe que estes são os mesmos índices para classe 1 que já demonstramos. A novidade são os referidos indicadores para a classe 0. Observe que o resultado desta é bem superior àquela.

In:	# Relatório de Classificação print(metrics.classification_report(y, y_pred))				
Out:		precision	recall	f1-score	support
	0	0.89	0.59	0.71	577009
	1	0.36	0.76	0.48	172638
	accuracy			0.63	749647
	macro avg	0.62	0.67	0.60	749647
	weighted avg	0.77	0.63	0.66	749647

## 6.2. Conclusão

Antes de iniciar este trabalho, ao perceber o massivo volume de dados existente, imaginamos que o resultado seria extremamente satisfatório. No entanto, vimos que as métricas de avaliação são claras que o modelo é apenas razoável.

Pode ser que estes dados tenham sido mal trabalhados, ou, simplesmente, pode ser que sejam insuficientes para que os algoritmos de *machine learning* encontrem padrões satisfatórios que permitam uma predição de melhor qualidade. Por insuficientes, não me refiro ao volume, e sim às features disponíveis.

Como já afirmado no capítulo inicial, os dados tributários são sigilosos, de modo que tivemos que trabalhar com exceções ao sigilo, que, obviamente, são apenas parte dos dados que os órgãos que mantêm estes dados possuem.

Os dados que talvez ajudassem a melhorar o modelo são:

- a) Quanto ao débito: situação anterior do débito antes de ser parcelado; forma de constituição do débito (se foi declarado espontaneamente ou lançado de ofício); se houve litígio administrativo;
- b) Quanto ao contribuinte: histórico de situação fiscal (é um devedor contumaz?); dívida em outros órgãos (como RFB); renda e patrimônio do contribuinte (a fim de calcular sua capacidade de pagamento).

De qualquer forma, temos uma análise exploratória de dados e um modelo preditivo baseado exclusivamente nos dados públicos disponíveis. Resgatando-se o problema proposto, quais sejam: compreender o perfil dos débitos parcelados na dívida ativa da União; e prever se um débito será parcelado com os dados obtidos, entendemos que o primeiro foi satisfatoriamente respondido, enquanto o último, como vimos, obtivemos um resultado apenas razoável.


## 7. LINKS

7.1. Link para o vídeo: <https://www.youtube.com/watch?v=XfjSaJG98zo>



7.2. Link para download do dataset final (e demais arquivos) no Dropbox:

<https://www.dropbox.com/sh/xcr520qelx32013/AAA-FXouG-Aew8jFrDen1ijha?dl=0>



[Acessar](#) [Registrar-se](#)

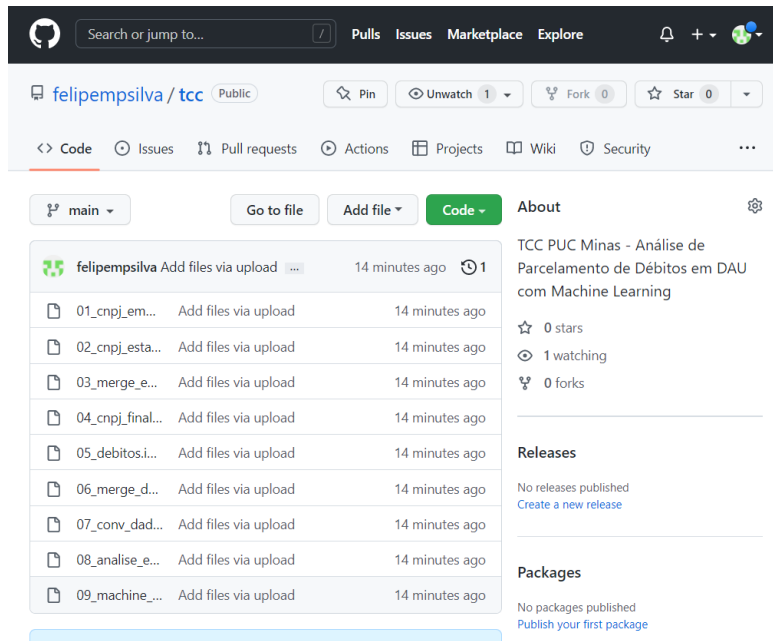
Arquivos públicos

[Salvar no Dropbox](#) [Baixar](#)

Nome	Modificado	Tamanho	
01_cnpj_empresas.ipynb	30/05/2022 04:26	9,25 KB	
02_cnpj_estabelecimentos.ipynb	30/05/2022 04:30	48,98 KB	
03_merge_empresa_estabelecimento.ipynb	23/04/2022 04:19	20,31 KB	
04_cnpj_final.ipynb	30/05/2022 04:40	54,61 KB	
05_debitos.ipynb	30/05/2022 04:52	119,12 KB	
06_merge_debitos_cnpj.ipynb	30/05/2022 05:00	66,47 KB	
07_conv_dados_continuos.ipynb	30/05/2022 05:43	111,84 KB	
07_dataset_final_com_dados_continuos.csv	26/05/2022 02:15	145,4 MB	
08_analise_exploratoria.ipynb	30/05/2022 09:25	1,21 MB	
09_machine_learning-com_ensemble_xgb_knn.ipynb	30/05/2022 11:50	238,51 KB	
09_machine_learning.ipynb	30/05/2022 14:39	152,16 KB	

7.3. Link para o repositório de notebooks: <https://github.com/felipempsilva/tcc>

Não foi possível fazer o upload do dataset final no GitHub, pois este limita arquivos a 25 MB.



## REFERÊNCIAS

ALURA. **Machine Learning: lidando com dados de muitas dimensões**. Disponível em: <<https://www.alura.com.br/conteudo/reducao-dimensionalidade>>. Acesso em: 29 mai. 2022.

BRASIL. Decreto nº 8.777, de 11 de maio de 2016. Institui a Política de Dados Abertos do Poder Executivo Federal. **Diário Oficial da União**, Brasília, 12 mai. 2016. Disponível em: <[http://www.planalto.gov.br/ccivil\\_03/\\_ato2015-2018/2016/decreto/d8777.htm](http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2016/decreto/d8777.htm)>. Acesso em: 27 mai. 2021.

BRASIL. Lei nº 5.172, de 25 de outubro de 1966. Dispõe sobre o Sistema Tributário Nacional e institui normas gerais de direito tributário aplicáveis à União, Estados e Municípios. **Diário Oficial da União**, Brasília, 27 out. 1966. Disponível em: <[http://www.planalto.gov.br/ccivil\\_03/leis/l5172compilado.htm](http://www.planalto.gov.br/ccivil_03/leis/l5172compilado.htm)>. Acesso em: 27 mai. 2021.

BRASIL. Procuradoria-Geral da Fazenda Nacional (PGFN). **Dados Abertos**. Disponível em: <<https://www.gov.br/pgfn/pt-br/assuntos/divida-ativa-da-uniao/dados-abertos/dados-abertos>>. Acesso em: 22 mai. 2022.

BRASIL. Procuradoria-Geral da Fazenda Nacional (PGFN). **PGFN em números 2022 – dados de 2021**. Disponível em: <[https://www.gov.br/pgfn/pt-br/acesso-a-informacao/institucional/pgfn-em-numeros/pgfn\\_em\\_numeros25042022-compresed.pdf](https://www.gov.br/pgfn/pt-br/acesso-a-informacao/institucional/pgfn-em-numeros/pgfn_em_numeros25042022-compresed.pdf)>. Acesso em: 29 mai. 2022.

BRASIL. Secretaria Especial da Receita Federal do Brasil (RFB). **Dados Públicos CNPJ**. Disponível em: <<https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/cadastros/consultas/dados-publicos-cnpj>>. Acesso em 21 fev. 2022.

CARVALHO, Rafael Neves. **A Utilização de Ferramentas de Análise de Informações Do Tipo “Biga Data” No Âmbito Da Receita Federal do Brasil**. Brasília: 2019.

CLÉSIO, Flávio. **Sete técnicas para redução da dimensionalidade**. 13 jun 2015. Disponível em: <<https://mineracaodedados.wordpress.com/2015/06/13/7-tecnicas-para-reducao-da-dimensionalidade>>. Acesso em: 29 mai. 2022.

DELFTSTACK. **Notação científica em Python**. 18 jul 2021. Disponível: <<https://www.delftstack.com/pt/howto/python/scientific-notation-python/>>. Acesso em: 29 mai. 2022.

ESCOVEDO, Tatiana; KOSHIYAMA, Adriano. **Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise**. Casa do Código, 2020. E-book.

FACELI, Katti ... [et al.]. **Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina**. Rio de Janeiro: LTC, 2022. E-book.



FECOMERCIO-SP. **Pesquisa mostra que 79% dos brasileiros preferem compras parceladas.** 02 ago 2015. <https://www.fecomercio.com.br/noticia/pesquisa-mostra-que-79-dos-brasileiros-preferem-compras-parceladas>

GANDOMI, Amir; HAIDER, Murtaza. **Beyond the hype: Big data concepts, methods, and analytics.** 2014. Disponível em: < <https://www.sciencedirect.com/science/article/pii/S0268401214001066>>. Acesso em: 26 mai. 2021.

GEROLA, Letícia. **Qual métrica eu uso?** 22 jun 2020. Disponível em: <<https://medium.com/joguei-os-dados/que-m%C3%A9trica-eu-uso-818bb4422e37>>. Acesso em: 29 mai. 2022.

GOMES, Pedro César Tebaldi. **Conheça as Etapas do Pré-Processamento de dados.** 13 dez 2019. Disponível em: <<https://www.datageeks.com.br/pre-processamento-de-dados/>>. Acesso em: 29 mai. 2022.

GUALBERTO, Arnaldo. **O que não te contam sobre métricas de classificação binária.** 29 jul 2020. Disponível em: <<https://medium.com/@arnaldog12/o-que-n%C3%A3o-te-contam-sobre-m%C3%A9tricas-de-classifica%C3%A7%C3%A3o-bin%C3%A1ria-d1834e385402>>. Acesso em: 29 mai. 2022.

HELLER, Martin. **What is machine learning? Intelligence derived from data.** 15 mai 2019. Disponível em: <<https://www.infoworld.com/article/3214424/what-is-machine-learning-intelligence-derived-from-data.html>>. Acesso em 29 mai. 2022.

INSTITUTO INNOVARE. **PGFN usa algoritmos de machine learning para otimizar cobrança da dívida ativa da União.** 27 nov 2020. Disponível em: <<https://www.premioinnovare.com.br/noticias/pgfn-usa-algoritmos-de-machine-learning-para-otimizar-cobranca-da-divida-ativa-da-uniao/47>>. Acesso em: 29 mai. 2022.

HARRISON, Matt. **Machine Learning – Guia de Referência Rápida: Trabalhando com dados estruturados em Python.** São Paulo: Novatec: 2020. E-book.

JUNIOR, Valter Police. **Quando vale a pena parcelar uma compra?** 27 jun 2017. Disponível em: <<https://epocanegocios.globo.com/colunas/Seu-Planejamento-Financeiro/noticia/2017/06/quando-vale-pena-parcelar-uma-compra.html>>. Acesso em: 29 mai. 2022.

MAYER, Fernando de Pol. **Análise exploratória de dados.** Disponível em: <<https://www.inf.ufsc.br/~andre.zibetti/probabilidade/aed.html>>. Acesso em: 29 mai. 2022.

PAULA, Hugo Bastos de. **Aprendizado de Máquina.** Pontifícia Universidade Católica de Minas Gerais. Belo Horizonte: 2019.

SALLES, Rodrigo. **Outlier: o ponto fora da curva**. 15 mai 2018. Disponível em: <<https://medium.com/ensina-ai/outlier-o-ponto-fora-da-curva-1f28f3d9c23>>. Acesso em: 29 mai. 2022.

SCUDILIO, Juliana. **Qual a melhor métrica para avaliar os modelos de Machine Learning?** 26 jul 2020. Disponível em: <<https://www.flai.com.br/juscudilio/qual-a-melhor-metrica-para-avaliar-os-modelos-de-machine-learning>>. Acesso em: 29 mai. 2022.

SILVA, Josenildo. **Aprendendo em uma Floresta Aleatória**. 12 mar 2018. Disponível em: <<https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleat%C3%B3ria-3545f6babdf8>>. Acesso em: 29 mai. 2022.

TAURION, Cezar. **Big Data**. Rio de Janeiro: Brasport, 2013. ePub.

UNIVERSITY OF NEBRASKA-LINCOLN. **Remember the 5 W's**. Disponível em: <<https://its.unl.edu/bestpractices/remember-5-ws>>. Acesso em: 29 mai. 2022.

VASANDANI, Jasmine. **A Data Science Workflow Canvas to Kickstart Your Projects**. 5 mai 2019. Disponível em: <<https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0>>. Acesso em: 29 mai. 2022.

WIKIPEDIA. **Coeficiente de correlação de Pearson**. 10 out 2018. Disponível em: <[https://pt.wikipedia.org/wiki/Coeficiente\\_de\\_correla%C3%A7%C3%A3o\\_de\\_Pearson](https://pt.wikipedia.org/wiki/Coeficiente_de_correla%C3%A7%C3%A3o_de_Pearson)>. Acesso em: 29 mai. 2022.