

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

Carla Fabiana Mont-Morency Gondim Rodrigues

USO REGULAMENTADO DE ÁGUA BRUTA

Belo Horizonte

2021

Carla Fabiana Mont-Morency Gondim Rodrigues

USO REGULAMENTADO DE ÁGUA BRUTA

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

Belo Horizonte
2021

SUMÁRIO

<u>1. Introdução</u>	4
<u>1.1. Contextualização</u>	4
<u>1.2. O problema proposto</u>	4
<u>2. Coleta de Dados</u>	4
<u>3. Processamento/Tratamento de Dados</u>	5
<u>4. Análise e Exploração dos Dados</u>	5
<u>5. Criação de Modelos de Machine Learning</u>	5
<u>6. Apresentação dos Resultados</u>	5
<u>7. Links</u>	6
APÊNDICE	7

1. Introdução

.1.1. Contextualização

No Brasil a caatinga abrange grande parte da Região Nordeste. O clima semiárido predomina, e no estado do Ceará corresponde a 86,8% do território. No estado não há rios perenes, o que demanda um grande esforço para o gerenciamento dos recursos hídricos. A pluviosidade é sazonal e concentra-se no primeiro semestre, na chamada quadra chuvosa. Diversos reservatórios foram construídos ao longo dos anos para armazenar água durante esse período e utilizar nos meses de baixa pluviosidade. Trechos dos leitos dos principais rios são perenizados através desses açudes.

Além dos mananciais superficiais, também há significativa contribuição de águas subterrâneas. Diversos poços são utilizados para exploração de aquíferos.

Esses mananciais são interligados por adutoras, canais e túneis formando uma grande malha d'água que permite ao estado manter suas atividades.

Em conjunto com outros fatores, há o acompanhamento da pluviometria, e também os monitoramentos qualitativo e quantitativo das águas dos principais reservatórios. Atualmente são monitorados 184 açudes e mais outros tipos de mananciais, através de medições e controle por balanço hídrico.

Para o gerenciamento de todo esse sistema, destacam-se alguns de seus principais instrumentos: Outorga, Fiscalização e Cobrança, além da participação os usuários através de suas respetivas instâncias.

A concessão das outorgas tem por objetivo regularizar o uso assim como as obras e serviços de interferência hídrica.

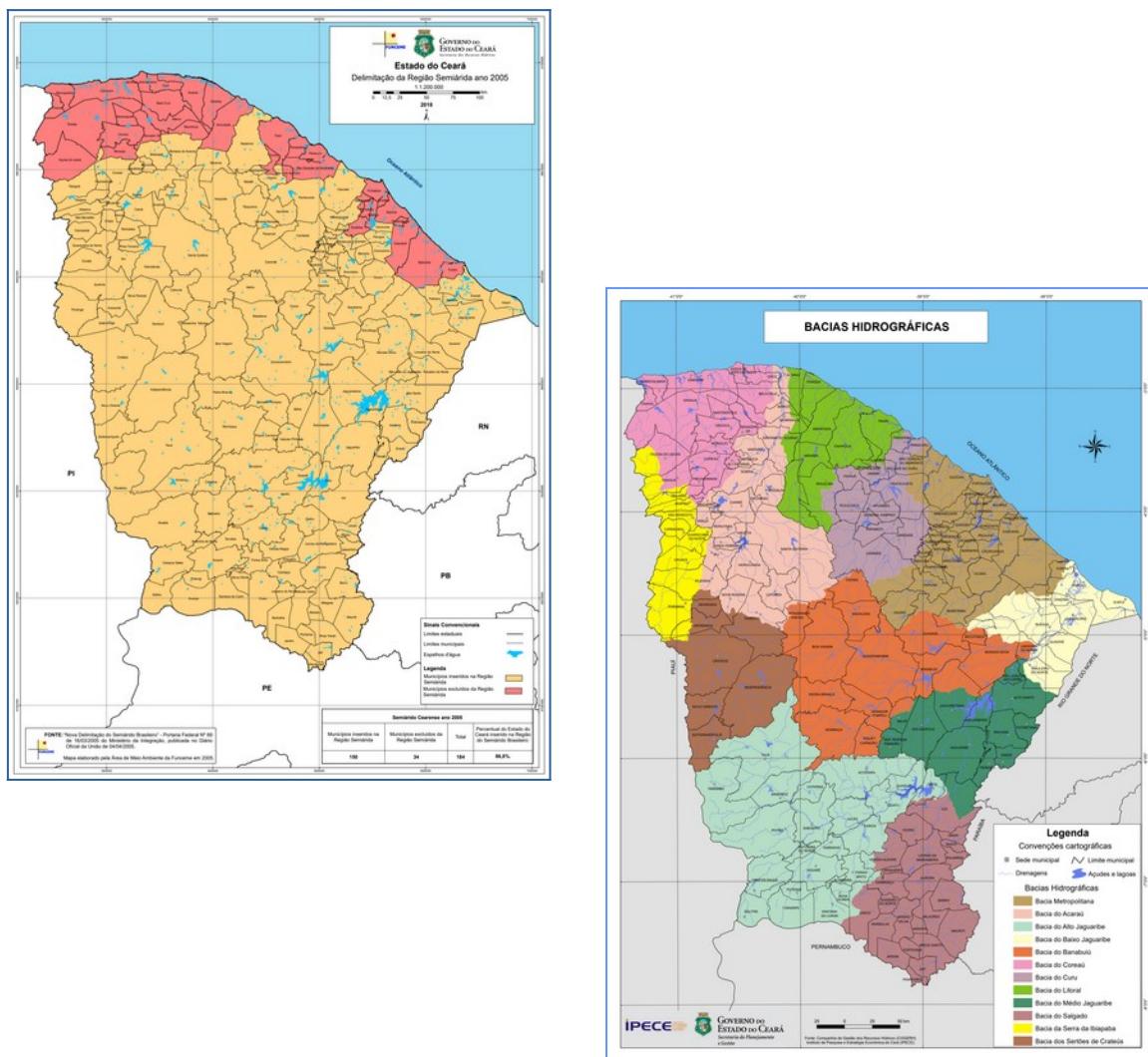
Anualmente ocorrem, após o final da quadra chuvosa, quando é observado o aporte obtido nos açudes monitorados, a alocação negociada das águas

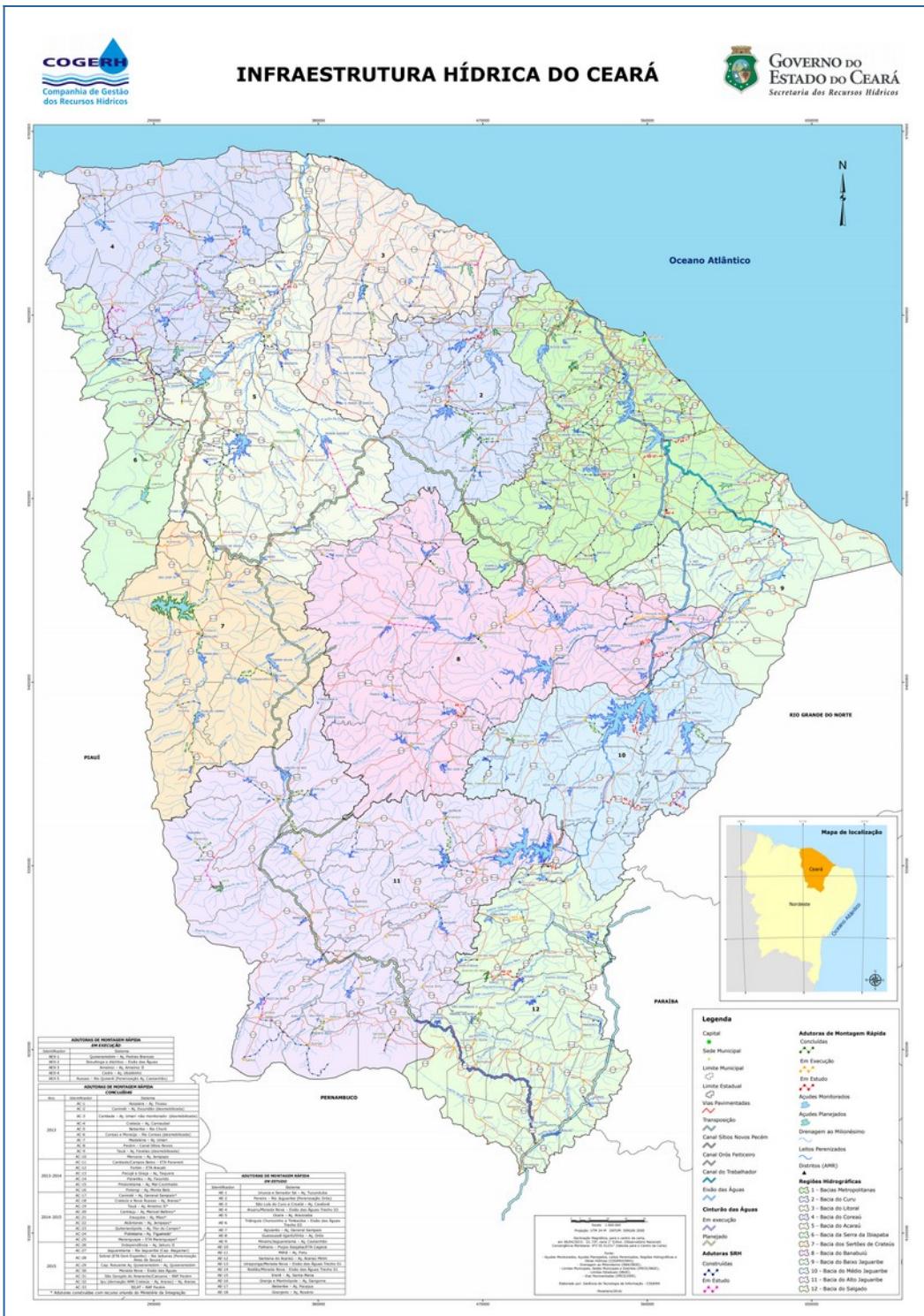
acumuladas, frequentemente no mês de juto, cuja operação dos reservatórios é definida pelos respectivos usuários.

O Rio Jaguaribe é o mais extenso, cortar o estado de sul a norte, e nele ficam os 02 maiores reservatórios: Orós e Castanhão.

No estado não há reservatórios com objetivos hidrelétricos, sendo o açude Castanhão o de maior capacidade de acumulação, com 6.700.000.000,00 m³.

O estado do Ceará que tem 148.886,00 km² e 184 municípios está dividido em 12 regiões hidrográficas.





.1.2. O problema proposto

Nesse projeto foi suprimida pessoa da área de negócio, pois recai sobre a presente estudante de Ciências de Dados e Big Data.

(Why?)

A regularização do uso da água é de suma importância para decisões referentes ao uso dos recursos hídricos. Busca-se obter um panorama .É importante conhecer a situação para direcionar os esforços de regularização, e consequentemente dar suporte aos tomadores de decisão no gerenciamento dos recursos hídricos. Entender a dinâmica da demanda para melhor gerenciar a oferta, inclusive quanto as transferências hídricas entre as regiões. Esse entendimento também recai sobre o gerenciamento das solicitações de obras e serviços de interferência hídrica, assim como de projetos de infraestrutura. Observar as demandas por município e predizer as tendências poderá colaborar no planejamento da regularização do uso dos recursos hídricos, mas esse é uma etapa posterior ao retrato da situação atual.

(Who?)

Os dados analisados são de uso regular de recursos hídricos do Estado do Ceará e também dados gerais dos municípios cearenses.

(What?):

O objetivo é ter uma visão do que já foi realizado, mas principalmente do que há para realizar, visualizar a situação atual para favorecer o planejamento de ações futuras. As vazões e volumes anuais outorgados em relação a população, a extensão territorial e as atividades produtivas nos municípios cearenses.

(Where?):

A região de interesse é o estado do Ceará, Região Nordeste do Brasil,

(When?):

Ano de 2020.

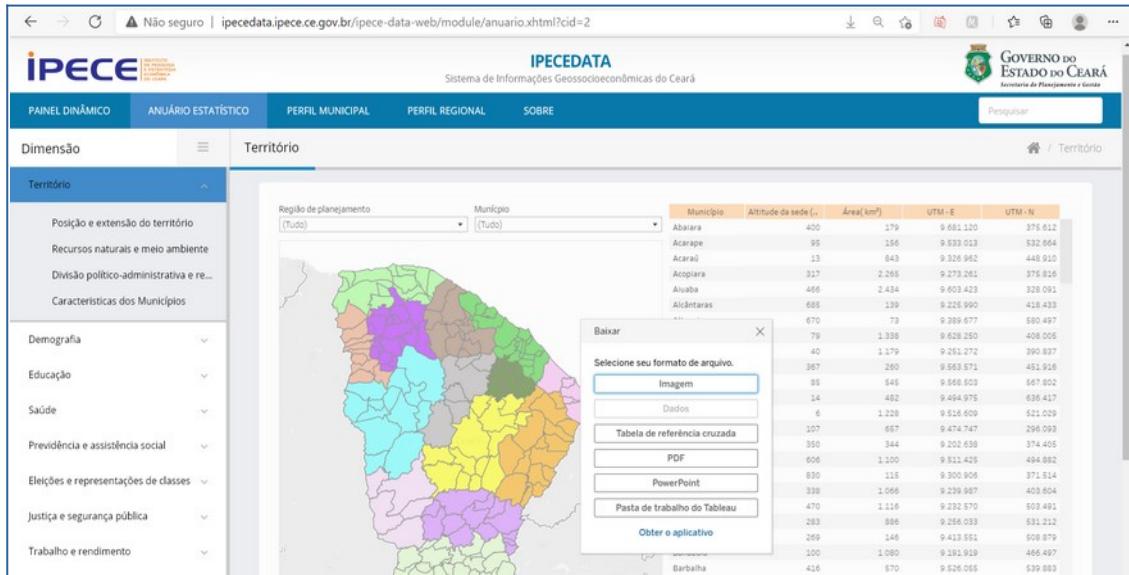
2. Coleta de Dados

A coleta teve como ponto de partida os dados referentes ao direito de uso da água bruta e buscou-se em outras fontes dados disponíveis que oferecer componentes para enriquecer a análise.

Os datasets foram obtidos de 05 fontes, ANA, IBGE, IPECE, COGERH e PUC, considerando também os dados georreferenciados.

Fontes	Data	Local
COGERH	2021	http://atlas.cogerh.com.br/
IPECE	2021	https://www.ipece.ce.gov.br/sistemas/
ANA	08/03/2021	http://www.snhrh.gov.br/
IBGE	2021	https://www.ibge.gov.br/explica/codigos-dos-municios.php#CE
PUC	2020	https://pucminas.instructure.com/courses/1768/files/378913?module_item_id=192541

Datasets baixados a partir da base do IPECE



A melhor opção no momento foi baixar pdf e excel, pois as licenças para uso pleno do Tableau não estavam disponíveis e isso estava atrasando.

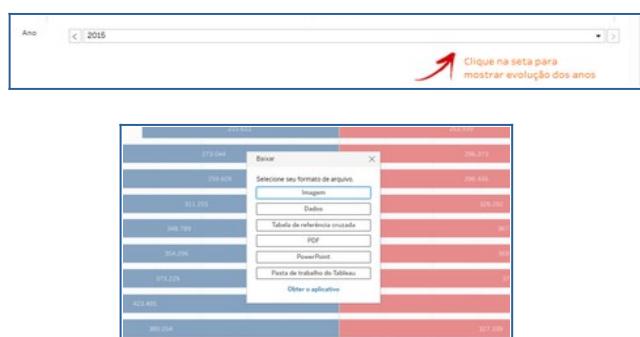
Dessa tabela, os dados de interesse no momento são as colunas “Município” e “Área”. Essa fonte é a mais atualizada, pois os municípios tiveram suas áreas revisadas em 2019. Talvez seja mais proveitosa.



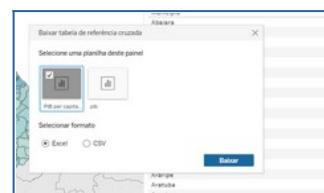
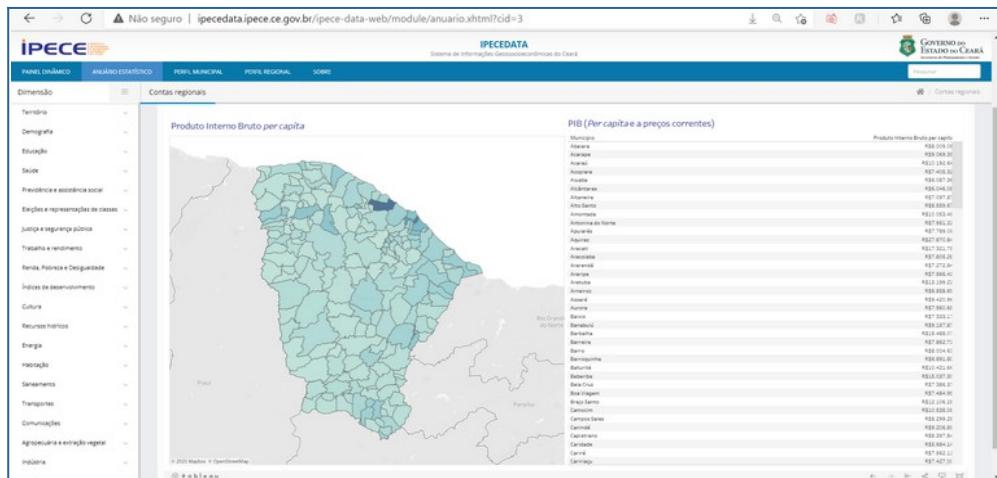
Os dados selecionados foram os mais recentes disponíveis : ano 2015

Esses dados são sobre população masculina e feminina e despertaram uma curiosidade: será que tem mais outorgas entre homens ou mulheres? Bom, há muitas empresas, nas quais o nome da empresa, é o mesmo nome do proprietário, em outras usam muitos nomes de fantasia, e para obter o nome do proprietário envolveria um aprofundamento na pesquisa. Há uma certa preocupação em não desobedecer a LGPD. Talvez essa curiosidade seja respondida em um próximo projeto.

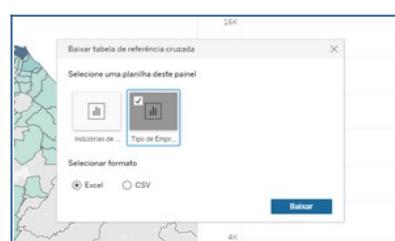
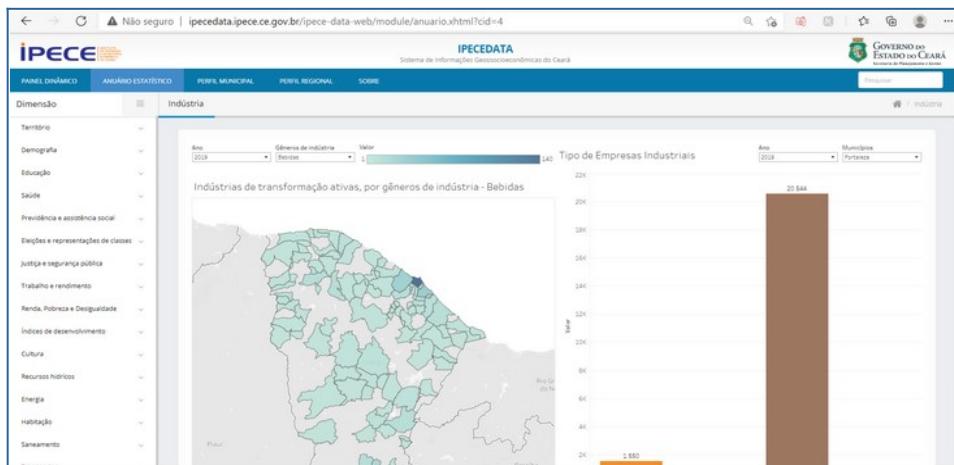
Os dados selecionados foram os mais recentes disponíveis : ano 2015 e foram baixados em formato PDF e Excel.



Temos aqui o PIB per capita por município



Como os dados mais recentes são de 2017, foram as 02 opções, caso decida fazer uma simulação de 2019.



Ambas foram baixadas para verificar o que armazenam, e foi constatado que os dados das planilhas são iguais, repetidos, provavelmente ainda não tem dados mais novos disponíveis para atualização.

No IPECEDATA, para obter os dados é preciso baixar uma planilha por município, nesse caso talvez em um outro local possam estar disponíveis já unificados.

Buscando dados no IBGE, a princípio pareceram muito pulverizados.

The screenshot shows a computer screen displaying the official website of the Instituto Brasileiro de Geografia e Estatística (IBGE). The URL is https://www.ibge.gov.br/estatisticas/downloads-estatisticas.html. The page has a blue header bar with the IBGE logo and navigation links for 'CORONAVÍRUS (COVID-19)', 'Simplificado', 'Participe', 'Acesso à Informação', 'Legislação', and 'Canais'. Below the header, there are tabs for 'Estatísticas', 'Geociências', 'Cidades e Estados', 'Agência de Notícias', 'Nossos sites', and 'Acesso à Informação'. A breadcrumb trail indicates the user is in the 'Estatísticas' section. The main content area is titled 'Downloads' and contains a message: 'Aqui você pode baixar conteúdos das nossas pesquisas estruturais, censos, entre outras, na área de estatísticas.' Below this message is a list of three items:

- » [Acesso_a_internet_e_posse_cellular](#)
- » [acesso_ao_cadastro_unico_2014](#)
- » [Artigos_e_Apresentacoes](#)

Below this list, there is a large, detailed file tree structure for the 'Pib_Municípios' dataset, showing years from 2002 to 2018, along with sub-folders for 'base', 'ods', and 'xlsx' files, including 'indice.txt' and 'tabelas_completas.ods'.

De volta ao site do IPECE foi encontrada seção com anuários estatísticos, de onde foram baixados alguns datasets do anuário mais recente.

Publicações anteriores disponíveis:

Anuário 2017	Anuário 2016



3. Processamento/Tratamento de Dados

O dataset com os dados de outorgas de direito de uso de água bruta, é a principal base, contem campos com dados de usuários, que foram previamente suprimidos (nome, razão social, endereço, e-mail, telefones, cpf, cnpj), assim como os dados da concessão (processo, ato, data de registro na base, código do registro). Abriga os dados referentes ao uso da água, como volume e outros inerentes a cada finalidade de uso.

Os dados de direito de uso de água bruta são oriundos da exportação do Cadastro Nacional de Recursos Hídricos (CNARH), da Agência Nacional de Águas (ANA), alimentado pela Companhia de Gestão de Recursos Hídricos (COGERH) pertencente ao Sistema de Recursos Hídricos da Secretaria de Recursos Hídricos do Ceará, subtraindo-se campos que possam servir como identificadores dos usuários, visando cumprir o que preza a Lei Geral de Proteção de Dados (LGPD).

As informações relacionadas ao direito de uso de água bruta são publicizadas em obediência a Lei de Transparência, no site da COGERH, inclusive georreferenciadas.

A base da ANA é alimentada predominantemente por dados da base da COGERH e, a partir das atualizações nos registros ocorridas de 2017 em diante, inclusive várias retroativas, com ID's exclusivos para cada ponto de captação de água georreferenciado. A grande diferença envolve maior detalhamento das captações de águas subterrâneas.

Os demais datasets, contêm campos com dados dos municípios, foram baixados do IPECE e agrupados em diretórios conforme classificação original nos sites de origem para posterior aprofundamento da triagem.

Os datasets foram abertos com Libreoffice e salvos com nomenclatura conforme o conteúdo que continham em nova pasta identificada como ‘renomeados’.

PUC > 13 - TCC > tcc_enfim > agropecuaria			
Nome	Data de modificação	Tipo	Tamanho
AGROPECUÁRIA renomeados	07/03/2021 15:30	Pasta de arquivos	
18.1.1	25/02/2021 00:02	Planilha do Micros...	74 KB
18.5.1	25/02/2021 00:01	Planilha do Micros...	68 KB
18.5.2	25/02/2021 00:01	Planilha do Micros...	60 KB
18.5.3	25/02/2021 00:01	Planilha do Micros...	78 KB
18.7.1	25/02/2021 00:00	Planilha do Micros...	36 KB
18.7.2	25/02/2021 00:00	Planilha do Micros...	186 KB

Posteriormente, os dados dos municípios obtidos dos datasets oriundos do IPECE, CAR, IBGE, foram classificados e agrupados conforme afinidade com as principais finalidades de uso da água bruta adotadas no estado do Ceará, ou seja: abastecimento humano, dessedentação animal, aquicultura, irrigação, indústria, serviço e comércio, e demais usos.

Termologia temática	
Água bruta	Água sem nenhum tratamento qualitativo.
Captiação	Retirada de água do manancial
Direito de Uso	Outorga para fins de uso de água bruta
Espelho d'água	Área da superfície superior de um polígono de um corpo hídrico
Leito	Calhas dos rios, riachos, córregos (águas correntes)
Litros diários por habitante L/hab.dia	Consumo médio de um habitante por dia
Manancial	Rios, riachos, córregos, lagoas, açudes(represas),canais, adutoras, poços e fontes (nascentes)
Manejo	Forma de utilização da água quanto aos horários, tempo de captação e intervalos
Massa de água	Água acumulada
Métodos	Técnicas produtivas
Outorga	Autorização formal para uso de recursos hídricos ou Execução de obras/serviços de interferência hídrica
Uso consuntivo	Consumo com captação de água de água
Uso não consuntivo	Sem consumo expressivo de água, 'sem' captação de água (Ex.: peixe em gaiola)
Vazão (l/s)	Volume de um fluido por tempo, nesse caso volume de água em litros por segundo
Vazão máxima (l/s)	Máxima vazão instantânea de captação em litros por segundo
Vigência	Período de validade da outorga (anual)
Volume anual (m ³ /ano)	Volume de água bruta para uso anual em metro cúbicos
Volume máximo (m ³ /ano)	Máximo volume anual em determinado local (Ex.: município)

Pré seleção de relações conforme dados disponíveis:

Relação	Datasets
Gerais	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls
Volume anual e pib municipal	Extração cnarh.xls 25.2.1_PIB PREÇOS CORRENTES ANUAIS.xls municípios_ce_ibge.xls
Volume anual e população	municípios_ce_ibge.xls
Finalidade de uso – Irrigação	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls
Área irrigada por município	Extração cnarh.xls
Área irrigada por município < área do municípios	Extração cnarh.xls Área dos municípios e coordenadas.xls
Volume anual e área irrigada	Extração cnarh.xls
Volume anual e produção agrícola Dados agrícolas	Extração cnarh.xls
Área irrigada por município e área imóveis rurais	Extração cnarh.xls 18.1.1 ESTRUTURA FUNDIÁRIA
Finalidade de uso – Dessedentação animal	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls
Volume anual e nº de animais (bovino, caprino, suíno, equino, aves)	Extração cnarh.xls
Nº total de Bovinos e vacas ordenhadas, produção de leite corte	18.5.1 PRODUTOS DE ORIGEM ANIMAL LEITE
Nº total de Aves e produção de ovos (galinha, codorna), abate	18.5.3 PRODUTOS DE ORIGEM ANIMAL OVOS
Corte dados	
Finalidade de uso - Aquicultura	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls
Volume anual e área de viveiros (peixe,camarão) ornamentais	
Volume anual e produção kg (peixe, camarão) dados da produção excluem gaiolas ?	
Espelho d'água e ??? buscar dataset ou descartar	
Finalidade de uso - Abastecimento humano	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls
População abastecida e população do município	Extração cnarh.xls BUSCAR
Volume anual per capita para população abastecida E população do município	Extração cnarh.xls
Finalidade de uso - Industrial	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls
Volume anual por nº de empresas ativas	
Finalidade de uso - Serviço e Comércio	
Volume anual por município (total,subterrânea,superficial)	Extração cnarh.xls

Planilhas descartadas:

Datasets descartados	Justificativa
18.5.2 PRODUTOS DE ORIGEM ANIMAL MEL DE ABELHA.xls	Não há dados, com essa classificação, relacionados ao uso de água bruta
18.7.1 PESCA DO CONTINENTAL AÇUDES DNOCS	Os dados são gerais do Estado, não são especificados quais açudes ou distribuídos por municípios
19.2.1 INDÚSTRIA EXTRATIVA MINERAL EMPRESAS ATIVAS	A extração mineral (predominantemente de areia Em calhas de rios para a construção civil) é classificada como serviço de interferência hídrica, sem o uso não consumutivo. Nos datasets não há campos que os mensurem essa atividade.
Indústrias de transformação ativas, por gêneros de indústria BEBIDAS POR MUNICÍPIO	Os datasets estão separados por cada tipo de gênero e trazem apenas o quantitativo de estabelecimentos. O dataset 19.1.1 DADOS GERAIS EMPRESAS ATIVAS é apresenta esses dados reunidos.
Tipo de Empresas Industriais FORTALEZA	São 184 municípios para baixar os dados, mas esses contêm apenas o quantitativo de estabelecimentos. O dataset 19.1.1 DADOS GERAIS EMPRESAS ATIVAS é apresenta esses dados reunidos.
Piramide REFERÊNCIA NÃO IDENTIFICADA	Contém dados sobre a população, mas a referência não ficou clara.

É preciso definir a melhor janela temporal para esse estudo, isso dependerá dos dados disponíveis nos datasets, para tal todos foram colocados num mesmo diretório, “renomeados”

Dataset renomeados conforme finalidade de uso associada

Finalidade associada	Tema na fonte	Nome do arquivo	Período Referencial	Fonte
Irrigação	Agropecuária	18.1.1 ESTRUTURA FUNDIÁRIA	2005	IPECE
Dessedentação animal	Agropecuária	18.5.1 PRODUTOS DE ORIGEM ANIMAL LEITE	2013 2016	IPECE
Dessedentação animal	Agropecuária	18.5.3 PRODUTOS DE ORIGEM ANIMAL OVOS	2013-2015	IPECE
Aquicultura	Agropecuária	18.7.2 PESCA AQUICULTURA	2013-2016	IPECE
Irrigação	Área	Área dos municípios e coordenadas		IPECE
Serviço e comércio	Comércio interno	20.1.1 COMÉRCIO INTERNO ESTABELECIMENTOS	2014-2016	IPECE
Indústria	Indústria	19.1.1 DADOS GERAIS EMPRESAS ATIVAS	2014-2016	IPECE
Indústria	Indústria	19.2.1 INDÚSTRIA DE TRANSFORMAÇÃO EMPRESAS ATIVAS POR GÊNERO	2014-2016	IPECE
Geral	Indústria	19.4.1 INDÚSTRIA DA CONSTRUÇÃO CIVIL EMPRESAS ATIVAS	2014-2016	IPECE
Serviço e comércio	Serviços	21.1.1 EMPRESAS DE SERVIÇOS POR ATIVIDADE	2014-2016	IPECE
Serviço e comércio	Turismo	22.2.2 OFERTA HOTELEIRA MEIOS DE HOSPEDAGEM	2014-2016	IPECE
Abastecimento humano	População	MUNICIPIOS_CE_IBGE	2.019,00	IBGE
Irrigação	População	MUNICIPIOS_CE_IBGE	2.019,00	IBGE
Geral	População	MUNICIPIOS_CE_IBGE	2.019,00	IBGE
Todos	Direito de uso	direito de uso	2001-2020	ANA

Os dados obtidos no IPECE são do Anuário Estatístico do Ceará 2017.

Os anos de 2014, 2015 e 2016 foram escolhidos por serem mais recente e por estarem presentes na maioria dos datasets obtidos.

O dataset “18.1.1 ESTRUTURA FUNDIÁRIA”, referente ao ano 2005, permaneceu pois constava no Anuário Estatístico do ano de 2017, sugerindo que não ocorreram grandes alterações ou que eram os dados mais recentes a época de sua edição.

O dataset “municípios_ce_ibge”, referente ao ano 2019 permaneceu, pois seus dados não costumam sofrer variações extremas.

Os dados sobre os municípios oriundos do IPECE estavam dispersos em muitos arquivos, então foram reunidos num mesmo arquivo.

As planilhas de cada grupo foram incorporadas nas abas e estas renomeadas conforme seus conteúdos num dataset unificado, como exemplo “AGROPECUÁRIA UNIFICADA.xls”

The screenshot shows a LibreOffice Calc spreadsheet titled "AGROPECUÁRIA UNIFICADA.xls". The main title is "ANUÁRIO ESTATÍSTICO DO CEARÁ - 2017". Below it, section 18.7 PESCA is shown, specifically Table 18.7.2: "Produção e valor da produção da aquicultura, por tipo de produto, segundo os municípios - Ceará - 2013-2016". The table has columns for Municípios (Ceará, Acaraípe, Acaraú, Alto Santo) and various fish species (Carpa, Pintado, Tambaqui, Tilápia, etc.). The data shows production volumes in kilograms for each species across the four municipalities.

Municípios	Quantidade (kg)							
	Carpa	Pintado, cachara, cachapira e pintachara, surubim	Tambaqui	Tilápia				Camarão
Ceará	5.500	6.800	2.800	6.000	13.500	30.634.375	900	54.315 33.949.805 1
Acarape	-	-	-	-	-	-	-	-
Acaraú	-	-	-	-	-	-	-	4.552.000
Alto Santo	-	-	-	-	-	3.183.969	-	16.900

Esse procedimento foi descartado assim como a planilha exemplo acima.

Para uma maior eficiência, optou-se, como ensaio, por utilizar a ferramenta PowerBI, onde as planilhas foram adicionadas a um relatório previamente denominado “USO DE ÁGUA CE”, num diretório local.

Arquivo → Novo → Salvar Como → “USO DE ÁGUA CE.”



Através do comando “Obter dados” na Página Inicial, as planilhas foram inseridas.



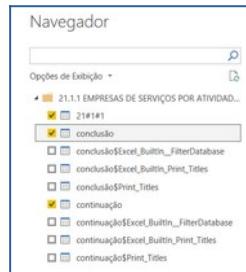
Não foi realizada nenhuma transformação durante o carregamento das planilhas, essa atividade será realizada posteriormente.

Municípios	Produção de ovos
Abaiara	33

Essas planilhas foram renomeadas conforme conteúdo. Durante o processo de renomear no PowerBI, percebeu-se que essas planilhas geradas ao comando “Carregar”, que estão a esquerda eram réplicas das planilhas a direita, então foram excluídas.



Ao carregar o arquivo “21.1.1 EMPRESAS DE SERVIÇOS POR ATIVIDADE”, apenas os assinalados foram carregados, os demais eram réplicas.



Os arquivos “direito de uso.csv” e “MUNICÍPIOS CE.xls” foram previamente salvos como “excel 93-2003” para evitar tratamentos, considerando que é um ensaio ainda, nos campos de texto em palavras com caracteres da língua portuguesa como “ç” ou “~”, por exemplo.

Da planilha “direito de uso”, que originalmente possui 14.901 registros e 228 campos, antes de realizar o upload para o PowerBI, foram excluídas as colunas:

TABELA DE CAMPOS EXCLUÍDOS – ensaio	
Nome da coluna	Conteúdo
EMP_NM_EMPREENDIMENTO	Nome do empreendimento, predominantemente igual a coluna “emp_nm_usuario”, ou seja da pessoa física ou jurídica
EMP_NM_USUARIO	Nome do titular da outorga de direito de uso da água, pessoa física ou jurídica.
EMP_NU_CPFNPJ	CPF e CNPJ, cadastro de pessoa física e cadastro nacional de pessoa jurídica, junto a receita federal
INT_CD_REGLA	Código no regla -sigla Sistema de cadastro da ANA, o Ceará tem um sistema próprio e não utiliza, logo estava completamente vazio
EMP_DS_EMAILRESPONSAVEL	Endereço eletrônico do responsável
EMP_NU_CEPENDERECHO	Código de endereçamento postal -cep nos correios do endereço de correspondência do responsável
EMP_DS_LOGRADOURO	Nome do logradouro para correspondência do responsável
EMP_DS_COMPLEMENTOENDERECO	Complemento do endereço de correspondência do responsável
EMP_CD_IBGEMUNCORRESPONDENCIA	Código do inibge do município do endereço de correspondência do responsável
EMP_NU_LOGRADOURO	Número no logradouro do endereço de correspondência do responsável
EMP_NU_CAIXAPOSTAL	Número da caixa postal junto aos correios de correspondência do responsável
EMP_DS_BAIRRO	Bairro do endereço de correspondência do responsável
EMP_NU_DDD	Código de área do telefone de contato do responsável
EMP_NU_TELEFONE	Telefone de contato do responsável
EMP SG_UF	Estado do endereço de correspondência do responsável
EMP_NM_MUNICIPIO	Município do endereço de correspondência do responsável

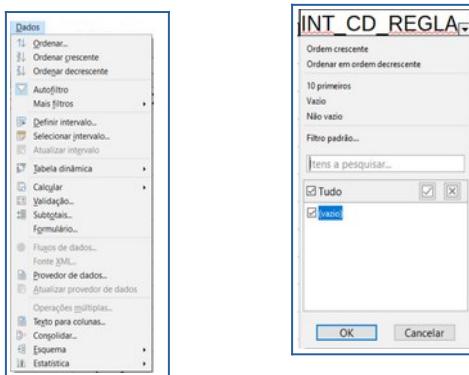
Esse dataset é o que estará disponível para os trabalhos, após a exclusão dos dados acima listados e a inclusão das colunas “CPF ou CNPJ”, “data inicial” e “data final”, conforme descrito a seguir.

Para a planilha “direito de uso” usar como separador apenas o “ponto e vírgula”, pois os dados têm diversos formatos, além de ser o adotado no CNARH.

A coluna “EMP_NU_CPFCNPJ” trazia os números de CPF e CNPJ, que seriam eliminados, mas há um possível interesse em contabilizar pessoas físicas e jurídicas futuramente, então esses dados passaram por um tratamento antes disso. Foi aplicado uma condicionante simples, utilizando o LibreOffice Calc. Uma nova coluna denominada “CPF ou CNPJ” foi adicionada a planilha. Com a coluna AB eliminada, o vínculo se perderia, então seu conteúdo da coluna AC foi copiado e colado como um texto sem formatação desvinculado em uma nova coluna HW.

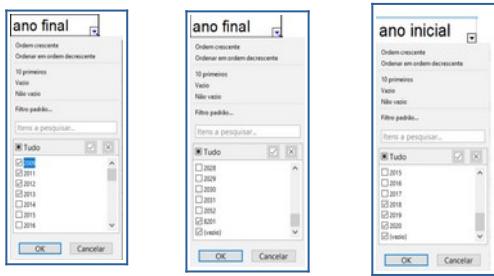
Esse procedimento foi dispensado, pois o dataset “direito de uso” já traz informações mais específicas, pois ser pessoa física ou jurídica, não está associado a nenhuma finalidade de uso, ambas podem obter qualquer tipo de outorga.

Ao utilizar o LibreOffice Calc para abrir o dataset, através da aplicação do comando “Autofiltro” em “Dados” foi observado que na coluna “INT_CD_REGLA” todos as células estavam vazios.



Também foram excluídos os registros fora da janela temporal desse estudo, ou seja, as outorgas não vigentes no ano de 2014, 2015, 2016, as com data final anterior a 2013 e as com data inicial posterior a 2017. A princípio, os dados de 2017 foram mantidos pois a maioria dos datasets oriundos do IPECE retratam o cenário anterior próximo a concessão das outorgas desse ano.

Após algumas tentativas fracassadas de carregar a planilha completa no PowerBI, por apresentar “erro....”, a planilha foi fracionada no Libre Office Calc com aplicação de filtros. As datas foram aplicados o comando “texto para colunas”, em seguida separado o ano final e o ano inicial e excluídos os assinalados:

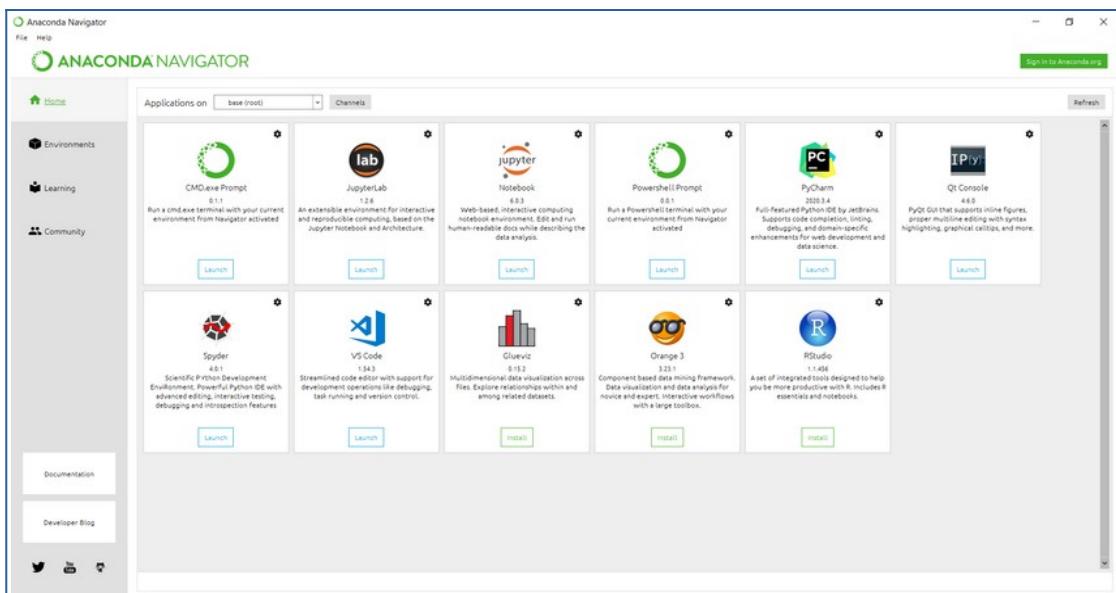


Ainda no Calc duas colunas foram adicionadas “ano inicial” e “ano final”.

Nesse momento ficou claro que uma mudança de ferramenta era necessária, ou melhor, a combinação delas, tendo em vista a possível acessibilidade por outras pessoas.

Como esse Dataset costuma receber novos registros, revisões, enfim, atualizações, sem uma periodicidade uniforme, optou-se por utilizar desenvolver um script permitindo sua substituição por novas versões.

Para elaboração desse script, foi utilizada a linguagem Python na plataforma Anaconda no editor Jupyter Notebook, pois já contém muitas bibliotecas, o que agiliza essa atividade e mais adiante as de aprendizado de máquina.



Bibliotecas iniciais requeridas para aplicação de scripts em Python: pandas, xlrd, matplotlib, scikit-learn, plotnine, seaborn e notebook.

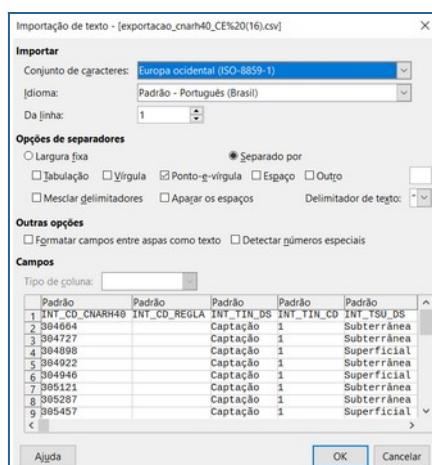
Verificando as bibliotecas já instaladas com o comando “pip freeze”:

```
In [2]: pip freeze
    pep8==1.7.1
    pexpect==4.8.0
    pickleshare==0.7.5
    Pillow==7.0.0
    pkginfo==1.5.0.1
    pluggy==0.13.1
    ply==3.11
    prometheus-client==0.7.1
    prompt-toolkit==3.0.3
    psutil==5.6.7
    py==1.8.1
    pycodestyle==2.5.0
    pycosat==0.6.3
    pycparser==2.19
    pycrypto==2.6.1
    pycurl==7.43.0.5
    pydocstyle==4.0.1
    pyflakes==2.1.1
    Pygments==2.5.2
    pylint==2.4.4
```

Foi percebida a ausência da biblioteca “plotnine”, que a seguir foi instalada com o comando “pip install plotnine”

```
In [3]: pip install plotnine
Collecting plotnine
  Using cached plotnine-0.7.1-py3-none-any.whl (4.4 MB)
Collecting mizani>=0.7.1
  Using cached mizani-0.7.2-py3-none-any.whl (62 kB)
Requirement already satisfied: numpy>=1.16.0 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (1.18.1)
Requirement already satisfied: scipy>=1.2.0 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (1.4.1)
Collecting statsmodels>=0.11.1
  Using cached statsmodels-0.12.2-cp37-none-win_amd64.whl (9.3 MB)
Requirement already satisfied: pandas>=1.1.0 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (1.2.3)
Collecting descartes>=1.1.0
  Using cached descartes-1.1.0-py3-none-any.whl (5.8 kB)
Requirement already satisfied: matplotlib>=3.1.1 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (3.1.3)
Requirement already satisfied: patsy>=0.5.1 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (0.5.1)
Collecting palettable
  Using cached palettable-3.3.0-py2.py3-none-any.whl (111 kB)
Requirement already satisfied: pytz>=2017.3 in c:\users\55859\anaconda3\lib\site-packages (from pandas>=1.1.0->plotnine) (2019.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\55859\anaconda3\lib\site-packages (from pandas>=1.1.0->plotnine) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\55859\anaconda3\lib\site-packages (from matplotlib>=3.1.1->plotnine) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\55859\anaconda3\lib\site-packages (from matplotlib>=3.1.1->plotnine) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\55859\anaconda3\lib\site-packages (from matplotlib>=3.1.1->plotnine) (2.4.6)
Requirement already satisfied: six in c:\users\55859\anaconda3\lib\site-packages (from patsy>=0.5.1->plotnine) (1.14.0)
Requirement already satisfied: setuptools in c:\users\55859\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=3.1.1->plotnine) (45.2.0.post20200210)
Installing collected packages: palettable, mizani, statsmodels, descartes, plotnine
Successfully installed descartes-1.1.0 mizani-0.7.2 palettable-3.3.0 plotnine-0.7.1 statsmodels-0.12.2
Note: you may need to restart the kernel to use updated packages.
```

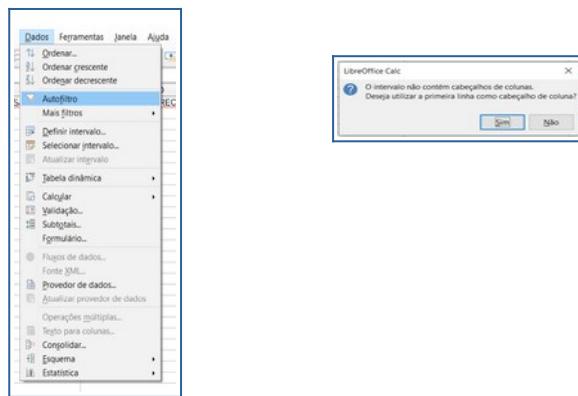
O dataset original foi baixado, em seguida aberto no Calc do pacote do LibreOffice, onde foram observadas as colunas trazidas



A base de dados do CNARH, já passou por algumas alterações em suas versões, assim como passou por unificação a outras bases e, com a introdução de novos campos que os registros mais antigos não continham, surgiram algumas lacunas que não tiveram seus dados atualizados com a formatação padronizada.

Nessa base de dados, a ingestão de dados pode ser via carga de dados, ETL, e também via web com preenchimento direto dos registros.

Em uma breve inspeção, ainda no Calc, com aplicação de “Autofiltro”, verificou-se que os dados estavam em sua coluna correspondente.



Na coluna “DATA_EXTRACAO”, onde não é pertinente a presença de “vazio” e todos os valores devem ser iguais, pois é a data de extração do dataset a partir da base de origem.

Após na checagem visual, comparando os dados ao cabeçalhos até a última coluna, chegou-se a conclusão que estavam nas colunas corretas.

O dataset foi salvo como uma planilha Excel do pacote Microsoft Office para prevenir uma possível incompatibilidade no uso do PowerBI, também da Microsoft, sendo renomeado como “direito_de_uso.xls” no mesmo diretório do Jupyter Notebook “direito_de_uso.ipynb”.



Importou-se a biblioteca pandas como “pd”, uma forma mais abreviada para indicar de qual biblioteca será retirada a função.

```
import pandas as pd
```

A seguir o arquivo “direito_de_uso.xls”, identificando as colunas com dados expressos em datas que serão utilizadas para delimitar a janela temporal, ou seja, as datas de início e final da vigência da outorga.

```
#CARREGANDO DATASET E IDENTIFICANDO AS COLUNAS DATA
direito_de_uso=pd.read_excel("direito_de_uso.xls",parse_dates=['OUT_DT_OUTORGAFINAL','OUT_DT_OUTORGAINICIAL','INT_DT_REGISTRO','A'])
```

Para verificar o carregamento, serão exibidas apenas as primeiras 05 linhas.

Removendo colunas com dados pessoais e/ou de contato que não são necessários nessa análise e atribuindo a esse DataFrame que é devolvido pelo método “drop” na mesma variável “direito_de_uso” .

COLUNAS A REMOVER I			
EMP_NU_CPFNPJ	EMP_NU_CEPENDERECO	EMP_NU_LOGRADOURO	EMP_NU_TELEFONE
EMP_NM_EMPREENDIMENTO	EMP_CD_IBGEMUNCORR	EMP_NU_CAIXAPOSTAL	EMP_SG_UF
EMP_NM_USUARIO	EMP_DS_LOGRADOURO	EMP_DS_BAIRRO	EMP_NM_MUNICIPIO
EMP_DS_EMAILRESPONSAVEL	EMP_DS_COMPLEMENTO	EMP_NU_DDD	

O conteúdo desses campos pode ser consultado na tabela de códigos campos do CNARH/ANA

```
#REMOVER COLUNAS COM INFORMAÇÕES PESSOAIS E DE CONTATO E REDEFINIR
```

```
direito_de_uso = direito_de_uso.drop(columns=['EMP_NU_CPFNPJ', 'EMP_NM_EMPREENDIMENTO', 'EMP_NM_USUARIO', 'EMP_DS_EMAILRESPONSAVEL'])
```

```
#VERIFICANDO A CARGA EM 05 LINHAS POR DEFAULT
```

```
direito_de_uso.head()
```

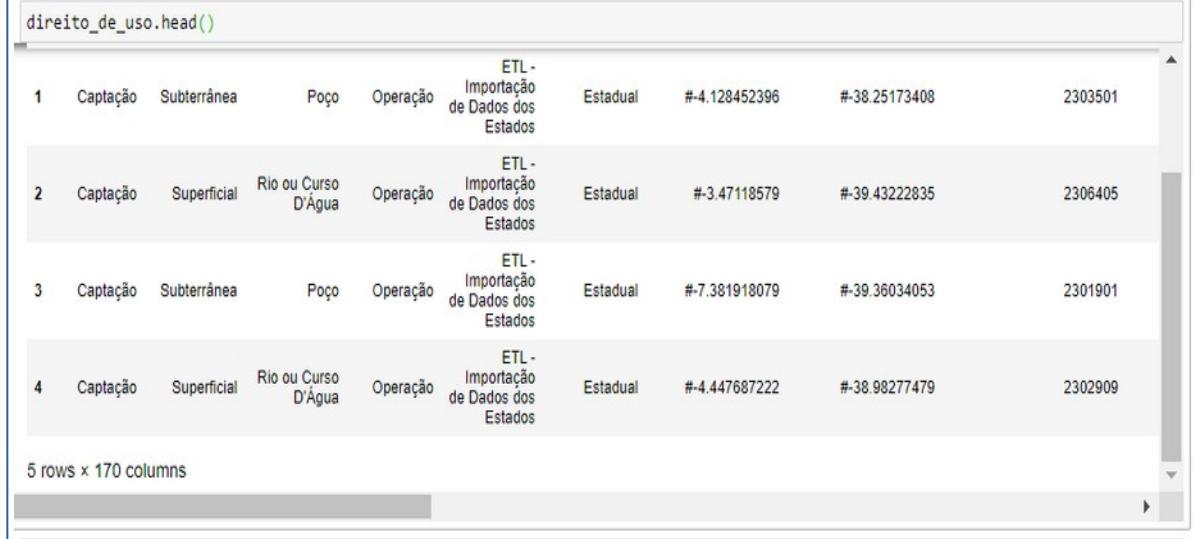
	INT_CD_CNARH40	INT_CD_REGLA	INT_TIN_DS	INT_TIN_CD	INT_TSU_DS	INT_TSU_CD	INT_TCH_DS	INT_TSI_DS	INT_TSI_CD	INT_TOD_DS	...	AMA_Q'
0	304664	NaN	Captação	1	Subterrânea	2	Poço	Operação	3.0	ETL - Importação de Dados dos Estados	...	
1	304727	NaN	Captação	1	Subterrânea	2	Poço	Operação	3.0	ETL - Importação de Dados dos Estados	...	
2	304898	NaN	Captação	1	Superficial	1	Rio ou Curso D'Água	Operação	3.0	ETL - Importação de Dados dos Estados	...	
3	304922	NaN	Captação	1	Subterrânea	2	Poço	Operação	3.0	ETL - Importação de Dados dos Estados	...	
4	304946	NaN	Captação	1	Superficial	1	Rio ou Curso D'Água	Operação	3.0	ETL - Importação de Dados dos Estados	...	

5 rows × 213 columns

Executou-se o reconhecimento de colunas com dados de códigos e IDs, que não serão utilizados nessa análise, para remoção e atribuiu-se novamente a esse DataFrame que é devolvido pelo método “drop” na mesma variável “direito_de_uso”.

Observe que ‘CD’ na nomenclatura dos cabeçalhos é um indicativo de que provavelmente refere-se a ‘código’ e ‘NU’ a um número identificador. A confirmação se deu com a aplicação de filtros numa amostra.

COLUNAS A REMOVER II			
INT_CD_CNARH40	ETP_MPE_CD	OUT_NU_PROCESSO	EFL_TTE_CD
INT_CD_REGLA	SIR_TSI_CD	OUT_NU_ATO	ITC_TUM_CD
INT_TIN_CD	SIR_TCT_CD	FIN_TFN_CD	ASB_TNP_CD
INT_TSU_CD	FTE_TCO_CD	TTC_TCU_CD	ASB_AQP_CD
INT_TSI_CD	FOH_TOH_CD	FSE_TES_CD	ASB_TPN_CD
INT_NU_CNARH	FOU_TOU_CD	FIE_TPS_CD	ABS_TCA_CD
INT_NU_SIAGAS	TUC_TEC_CD	FAH_TAH_CD	TST_TTB_CD
INT_CD_DECLARACAO	TUC_CD	FPE_TPE_CD	TST_TMI_CD
INT_CD_ORIGEM	ESC_TET_CD	CNA_CD_CNAE	ING_CD_COMITEFEDERAL
OUT_TPO_CD	CTE_TSC_CD	ING_CD_OTTOBACIA TRECHO	ING_CD_COMITEESTADUAL
OUT_TSP_CD	CTE_TCA_CD	FPE_CNA_CD	

#REMOVER COLUNAS COM INFORMAÇÕES DE CÓDIGOS DE CAMPOS JÁ EXISTENTES EM OUTRAS COLUNAS E DE ID's DE OUTRAS BASES DE DADOS									
direito_de_uso = direito_de_uso.drop(columns=['INT_CD_CNARH40', 'INT_CD_REGLA', 'INT_TIN_CD', 'INT_TSU_CD', 'INT_TSI_CD', 'INT_NU_CNARH'])									
<ipython-input-12-133a2a2a2a>									
direito_de_uso.head()									
									

Verificando informações sobre o DataFrame

#INFORMAÇÕES DO DATAFRAME
direito_de_uso.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 170 entries, INT_TIN_DS to OTO_DS_OUTROUSO
dtypes: datetime64[ns](4), float64(120), int64(1), object(45)
memory usage: 19.3+ MB

Resumo da quantidade de valores nulos por coluna

```
#VALORES NULOS POR COLUNA
direito_de_uso.isnull().sum()
```

```
INT_TIN_DS          0
INT_TSU_DS          0
INT_TCH_DS          0
INT_TSI_DS         282
INT_TOD_DS          0
...
ING_NM_COMITEFEDERAL 14901
ING_NM_COMITEESTADUAL 0
ING_CS_CONAMA       2419
OTO_NM_OUTROUSO     14901
OTO_DS_OUTROUSO      14620
Length: 170, dtype: int64
```

Resumo do percentual de dados ausentes por coluna

```
#VERIFICANDO AS %DADOS AUSENTES POR COLUNA
direito_de_uso.isna().mean()
```

```
INT_TIN_DS        0.000000
INT_TSU_DS        0.000000
INT_TCH_DS        0.000000
INT_TSI_DS        0.018925
INT_TOD_DS        0.000000
...
ING_NM_COMITEFEDERAL 1.000000
ING_NM_COMITEESTADUAL 0.000000
ING_CS_CONAMA     0.162338
OTO_NM_OUTROUSO    1.000000
OTO_DS_OUTROUSO    0.981142
Length: 170, dtype: float64
```

Para descartar colunas nas quais mais de 99% dos valores estão ausentes, ou seja, até no máximo 149 valores preenchidos de 14901 registros.

```
#DESCARTAR AS COLUNAS COM MAIS DE 99% DE VALORES AUSENTES FICARIA
direito_de_uso.dropna(thresh=len(direito_de_uso)*0.01, axis=1)
```

	INT_TIN_DS	INT_TSU_DS	INT_TCH_DS	INT_TSI_DS	INT_TOD_DS	INT_TDM_DS	INT_NU_LATITUDE	INT_NU_LONGITUDE	ING_NU_IBGEMUNICIPIO	INC
0	Captação	Subterrânea		Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.975406649	#-37.98052819	2311801
1	Captação	Subterrânea		Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.128452396	#-38.25173408	2303501
2	Captação	Superficial	Rio ou Curso D'Água		Operação	ETL - Importação de Dados dos Estados	Estadual	#-3.47118579	#-39.43222835	2306405
3	Captação	Subterrânea		Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-7.381918079	#-39.36034053	2301901
4	Captação	Superficial	Rio ou Curso D'Água		Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.447687222	#-38.98277479	2302909
...
14896	Lançamento	Superficial	Rio ou Curso D'Água	Operação	ETL - Importação de Dados dos Estados	Estadual	#-3.779782794	#-38.6383851645	2303709	
14897	Captação	Subterrânea		Poço	Operação	CNARH 40	Estadual	#-3.8276944444	#-41.0688944444	2313401
14898	Captação	Superficial	Rio ou Curso D'Água		Operação	ETL - Importação de Dados dos Estados	Estadual	#-6.0207875374	#-38.6775517217	2306900
14899	Captação	Subterrânea		Poço	Operação	CNARH 40	Estadual	#-7.2147777778	#-39.2693833333	2307304
14900	Captação	Superficial	Rio ou Curso D'Água		Operação	ETL - Importação de Dados dos Estados	Estadual	#-5.1566937072	#-38.0529319643	2307601

14901 rows × 112 columns

Nessa ação, seriam eliminadas 59 colunas, mas os dados preenchidos poderiam estar concentrados nos períodos de maior interesse, lembrando que novas colunas foram incorporadas à base ao longo dos anos.

Momentaneamente o percentual foi elevado para 100%, para mais adiante será dada continuidade no uso desse método.

O valor '0,00001' foi adotado no método pois implica dizer que o número de dados preenchidos por coluna é inferior a uma linha ($14.901 \times 0,00001 = 0,14901$)

#DESCARTAR AS COLUNAS COM MAIS DE 100% DE VALORES AUSENTES direito_de_uso=direito_de_uso.dropna(thresh=len(direito_de_uso)*0.00001, axis=1)										
direito_de_uso.head()										
	INT_TIN_DS	INT_TSU_DS	INT_TCH_DS	INT_TSI_DS	INT_TOD_DS	INT_TDM_DS	INT_NU_LATITUDE	INT_NU_LONGITUDE	ING_NU_IBGEMUNICIPIO	ING_SC
0	Captação	Subterrânea		Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.975406649	#-37.98052819	2311801
1	Captação	Subterrânea		Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.128452396	#-38.25173408	2303501
2	Captação	Superficial	Rio ou Curso D'Água		Operação	ETL - Importação de Dados dos Estados	Estadual	#-3.47118579	#-39.43222835	2306405
3	Captação	Subterrânea		Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-7.381918079	#-39.36034053	2301901
4	Captação	Superficial	Rio ou Curso D'Água		Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.447687222	#-38.98277479	2302909

5 rows x 137 columns

Mais 33 colunas seriam eliminadas nessa operação mantendo-se 137 colunas. Para consolidar a ação, o DataFrame "direito_de_uso" foi redefinido.

```
#DESCARTAR AS COLUNAS COM 100% DE VALORES AUSENTES
direito_de_uso=direito_de_uso.dropna(thresh=len(direito_de_uso)*0.00001, axis=1)
```

Verificando colunas ativas:

```
#VERIFICANDO LISTA DE COLUNAS ATIVAS
direito_de_uso.info(1)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Data columns (total 137 columns):
 #   Column           Dtype  
 --- 
 0   INT_TIN_DS       object  
 1   INT_TSU_DS       object  
 2   INT_TCH_DS       object  
 3   INT_TSI_DS       object  
 4   INT_TOD_DS       object  
 5   INT_TDM_DS       object  
 6   INT_NU_LATITUDE  object  
 7   INT_NU_LONGITUDE object  
 8   ING_NU_IBGEMUNICIPIO int64  
 9   ING_SG_UFMUNICIPIO object  
 10  ING_NM_MUNICIPIO  object  
 11  INT_NM_CORPOHIDRICO object  
 12  INT_NM_CORPOHIDRICOALTERADO object  
 13  INT_DS_ORGAO     object  
 ...  ...
```

A qualidade da água não será abordada nessa análise, que tem foco em aspectos mais relacionados aos dados quantitativos da água. As colunas com dados sobre a qualidade da água, que não serão utilizados nessa análise, foram removidas e atribuiu-se novamente a esse DataFrame que é devolvido pelo método “drop” na mesma variável “direito_de_uso”.

Observe que ‘AMA’ na nomenclatura dos cabeçalhos é um indicativo de que refere-se a parâmetros de qualidade da água, podendo essas colunas serem eliminadas sem necessidade de nova verificação de conteúdo.

COLUNAS A REMOVER III			
AMA_DT_COLETA	AMA_QT_PH	AMA_QT_CLORETO	AMA_QT_NITRITOS
AMA_DT_ANALISE	AMA_QT_CARBONATO	AMA_QT_DUREZATOTAL	AMA_QT_POTASSIO
AMA_NU_CONDUTIVIDADEELETTRICA	AMA_QT_TEMPERATURA	AMA_QT_FERROTOTAL	AMA_QT_SODIO
AMA_QT_COLIFORMESTOTAIS	AMA_QT_BICARBONATO	AMA_QT_FLUORETOS	AMA_QT_SULFATO
AMA_QT_COLIFORMESFECIAIS	AMA_QT_CALCIO	AMA_QT_NITRATOS	AMA_QT_MAGNESIO
AMA_QT_STD			

direito_de_uso.head(5)										
	INT_TIN_DS	INT_TSU_DS	INT_TCH_DS	INT_TSI_DS	INT_TOD_DS	INT_TDM_DS	INT_NU_LATITUDE	INT_NU_LONGITUDE	ING_NU_IBGEMUNICIPIO	ING_SG
0	Captação	Subterrânea	Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.975406649	#-37.98052819	2311801	
1	Captação	Subterrânea	Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.128452396	#-38.25173408	2303501	
2	Captação	Superficial	Rio ou Curso D'Água	Operação	ETL - Importação de Dados dos Estados	Estadual	#-3.47118579	#-39.43222835	2306405	
3	Captação	Subterrânea	Poço	Operação	ETL - Importação de Dados dos Estados	Estadual	#-7.381918079	#-39.36034053	2301901	
4	Captação	Superficial	Rio ou Curso D'Água	Operação	ETL - Importação de Dados dos Estados	Estadual	#-4.447687222	#-38.98277479	2302909	

5 rows x 116 columns

Mais 21 colunas retiradas, permanecendo 116 colunas.

As características construtivas dos poços, assim como os dados dos testes de vazão neles aplicados não serão abordados nessa análise, que tem foco mais relacionado a demanda em detrimento da oferta dos mananciais.

Nesse sentido, tais colunas foram removidas e atribuiu-se novamente a esse DataFrame que é devolvido pelo método “drop” na mesma variável “direito_de_uso”.

Observe que ‘ABS’ na nomenclatura dos cabeçalhos é um indicativo de que refere-se aos dados construtivos, já ‘TST’ diz respeito aos testes de vazão realizados, não houve necessidade de nova verificação de conteúdo.

COLUNAS A REMOVER IV			
ASB_DT_INSTALACAO	ASB_NU_TOPO	ASB_NU_ALTURABOCATUBO	TST_NU_ND
ASB_TNP_DS	ASB_NU_BASE	ASB_NU_COTATERRENO	TST_NU_NE
ASB_NU_DIAMETROPERFURACAO	ASB_TPN_DS	TST_DT	TST_VZ_ESTABILIZACAO
ASB_NU_DIAMETROFILTRO	ABS_TCA_DS	TST_TTB_DS	TST_TMI_DS
ASB_AOP_DS	ASB_NU_PROFUNDIDADEDEFINIDA	TST_DS_TEMPODURACAO	TST_NU_PERMEABILIDADE

Mais 20 colunas retiradas, permanecendo 96 colunas.

Foi aplicado o método ‘info(1)’ no intuito de observar o DataFrame e definir mais colunas com potencial de serem eliminadas.

```
direito_de_uso.info(1)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Data columns (total 96 non-null columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   INT_TIN_DS       14901 non-null   object  
 1   INT_TSU_DS       14901 non-null   object  
 2   INT_TCH_DS       14901 non-null   object  
 3   INT_TS1_DS       14619 non-null   object  
 4   INT_TOD_DS       14901 non-null   object  
 5   INT_TDM_DS       14901 non-null   object  
 6   INT_NU_LATITUDE  14901 non-null   object  
 7   INT_NU_LONGITUDE 14901 non-null   object  
 8   INT_NU_MUNICIPIO 14901 non-null   float64 
 9   ING_NU_MUNICIPIO 14901 non-null   object  
 10  ING_NM_MUNICIPIO 14901 non-null   object  
 11  INT_NM_CORPOHIDRICO 6075 non-null   object  
 12  INT_NM_CORPOHIDRICOALTERADO 3 non-null    object  
 13  INT_DS_ORGAO    14901 non-null   object  
 14  INT_DT_REGISTRO 14901 non-null   datetime64[ns] 
 15  INT_DS_OPCIONAL 428 non-null    object  
 16  OUT_TP_OUTORGIA 13941 non-null   object  
 17  OUT_TP_SITUACAOUTORGIA 14901 non-null   object  
 18  OUT_TP_OUTORGAFINAL 14651 non-null   object  
 19  OUT_DT_OUTORGAINICIAL 14671 non-null   datetime64[ns] 
 20  OUT_TP_ATO     11738 non-null   object  
 21  DAD_QT_VAZAODIAJAN 14374 non-null   object  
 22  DAD_QT_VAZAODIAFEV 14373 non-null   float64 
 23  DAD_QT_VAZAODIAMAR 14375 non-null   float64 
 24  DAD_QT_VAZAODIAABR 14390 non-null   float64 
 25  DAD_QT_VAZAODIAMAI 14645 non-null   float64 
 26  DAD_QT_VAZAODIAJUN 14653 non-null   float64 
 27  DAD_QT_VAZAODIAJUL 14660 non-null   float64 
 28  DAD_QT_VAZAODIAAGO 14725 non-null   float64 
 29  DAD_QT_VAZAODIASET 14831 non-null   float64 
 30  DAD_QT_VAZAODIAOUT 14846 non-null   float64 
 31  DAD_QT_VAZAODIANOV 14844 non-null   float64 
 32  DAD_QT_VAZAODIADEZ 14827 non-null   float64 
 33  DAD_QT_HORASJAN 14374 non-null   float64 
 34  DAD_QT_HORASFEV 14373 non-null   float64 
 35  DAD_QT_HORASMAR 14375 non-null   float64 
 36  DAD_QT_HORASMAIO 14396 non-null   float64 
 37  DAD_QT_HORASMAI 14645 non-null   float64 
 38  DAD_QT_HORASJUN 14575 non-null   float64 
 39  DAD_QT_HORASJUL 14669 non-null   float64 
 40  DAD_QT_HORASAGO 14755 non-null   float64 
 41  DAD_QT_HORASET 14831 non-null   float64 
 42  DAD_QT_HORASOUT 14846 non-null   float64 
 43  DAD_QT_HORASNOV 14844 non-null   float64 
 44  DAD_QT_HORASDEZ 14827 non-null   float64 
 45  DAD_QT_DIAJAN 14374 non-null   float64
```

```

46 DAD_QT_DIAFEV 14373 non-null float64
47 DAD_QT_DIAMAR 14375 non-null float64
48 DAD_QT_DIAABR 14390 non-null float64
49 DAD_QT_DIAMAI 14645 non-null float64
50 DAD_QT_DIAJUN 14653 non-null float64
51 DAD_QT_DIAJUL 14669 non-null float64
52 DAD_QT_DIAAGO 14755 non-null float64
53 DAD_QT_DIASET 14831 non-null float64
54 DAD_QT_DIAOUT 14846 non-null float64
55 DAD_QT_DIANOV 14844 non-null float64
56 DAD_QT_DIADEZ 14827 non-null float64
57 INT_QT_VAZAOMAXIMA 14900 non-null float64
58 INT_QT_VAZAOMEDIA 14852 non-null float64
59 INT_QT_VOLUMΕANUAL 14852 non-null float64
60 FIN_TFN_DS 14899 non-null object
61 FES_NU_PROFUNDIDADEMEDIATLTANQUE 5 non-null float64
62 FES_NU_AREATOTALTANQUE 105 non-null float64
63 TTC_TCU_DS 191 non-null object
64 FIE_TPS_DS 1 non-null object
65 IUS_NU_ALTURARES 8 non-null float64
66 IUS_NU_AREARESMAX 159 non-null float64
67 IUS_NU_VOLUMERES 778 non-null float64
68 IUS_NU_COEFICIENTE_RETORNO 11143 non-null float64
69 IUS_NM_ENTIDADECONCEDENTE 4 non-null float64
70 IUS_NU_CONCESSAO 36 non-null object
71 IUS_DT_FINALCONCESSAO 77 non-null datetime64[ns]
72 ETP_MPE_DS 1 non-null object
73 ETP_NU_QUANTIDADEMAMENSAL 1 non-null float64
74 SIR_TSI_DS 4242 non-null object
75 SIR_TCT_DS 3837 non-null object
76 SIR_NU_AREAIRRIGADA 3941 non-null float64
77 FIA_NU_POPULACAOATENDIDA 883 non-null float64
78 FRE_NU_VOLUMENA 1 non-null float64
79 HTE_NU_QUANTIDADE 3 non-null float64
80 TUC_TEC_DS 3 non-null object
81 TUC_DS 3 non-null object
82 CTE_TSC_DS 199 non-null object
83 CTE_TCA_DS 199 non-null object
84 CTE_NU_CABECAS 199 non-null float64
85 EFL_NU_DBORUTO 90 non-null float64
86 EFL_NU_DBOTRATADO 90 non-null float64
87 EFL_NU_NITROGENIOBRUTO 1 non-null float64
88 EFL_TTE_DS 1 non-null object
89 ITC_TUM_DS 451 non-null object
90 ITC_NU_PRODUCAOANUAL 451 non-null float64
91 CNA_DS 451 non-null object
92 DATA_EXTRACAO 14901 non-null datetime64[ns]
93 ING_NM_COMITEESTADUAL 14901 non-null object
94 ING_CS_CONAMA 12482 non-null float64
95 OTO_DS_OUTROUSO 281 non-null object
dtypes: datetime64[ns](4), float64(56), int64(1), object(35)
memory usage: 10.9+ MB

```

São irrelevantes para essa análise: a forma como foi realizada a ingestão de dados na base original, a data de alimentação dos registros na base, a nomenclatura do corpo hídrico, o tipo de ato formal empregado para regularização do uso, a profundidade dos tanques de aquicultura (para os cálculos de demanda é adotado um valor padrão), os dados da concessionária de água, os dados da entidade concedente, a discriminação dos produtos obtidos (espécies cultivadas, minerais extraídos), os de manejo adotados (irrigação, criação animal), a produção industrial anual pretendida por produto, a classificação nacional da atividade produtiva, a categoria junto ao CONAMA, os detalhes descritivos da finalidade de uso quando tipificadas como outros usos, os comitês estaduais a que estão associados os usuários, a dominialidade dos mananciais, e os tipos de mananciais.

As colunas cujos dados de preenchimento são iguais em todos os registro também devem ser eliminados pois não geram impacto, são eles: a sigla estadual, a data da extração do registro da base, e o órgão que alimentou a base de dados.

COLUNAS A REMOVER V			
INT_NM_CORPOHIDRICO	SIR_TSI_DS	OUT_TP_ATO	EFL_NU_DBOTRATADO
IUS_NM_ENTIDADECONCEDENTE	INT_TOD_DS	ING_CS_CONAMA	EFL_NU_DBOBRUTO
IUS_DT_FINALCONCESSAO	INT_TDM_DS	FIE_TPS_DS	DATA_EXTRACAO
EFL_NU_NITROGENIOBRUTO	INT_TCH_DS	ETP_MPE_DS	INT_DT_REGISTRO
S_NU_PROFUNDIDADEMEDIANAO	SIR_TCT_DS	EFL_TTE_DS	OTO_DS_OUTROUSO
INT_NM_CORPOHIDRICOALTERADO	CTE_TSC_DS	INT_DS_ORGAO	IUS_NU_CONCESSAO
ING_NM_COMITEESTADUAL	CNA_DS	INT_DS_OPCIONAL	ING_SG_UFMUNICIPIO
ITC_NU_PRODUCAOANUAL	ITC_TUM_DS		

#REMover COLUNAS COM INFORMAçõES IRRELEVANTES E COM TODOS OS ELEMENTOS IGUAIS
direito_de_uso = direito_de_uso.drop(columns=['INT_NM_CORPOHIDRICO', 'INT_NM_CORPOHIDRICOALTERADO', 'INT_DT_REGISTRO', 'INT_DS_OF'])
< ----- >
direito_de_uso.head()
INT_TIN_DS INT_TSU_DS INT_TSI_DS INT_NU_LATITUDE INT_NU_LONGITUDE ING_NU_IBGEMUNICIPIO ING_NM_MUNICIPIO OUT_TP_OUTORGIA OUT_TP_
0 Captação Subterrânea Operação #-4.975406649 #-37.98052819 2311801 RUSSAS Direito de Uso
1 Captação Subterrânea Operação #-4.128452396 #-38.25173408 2303501 CASCAVEL Direito de Uso
2 Captação Superficial Operação #-3.47118579 #-39.43222835 2306405 ITAPIPOCA Direito de Uso
3 Captação Subterrânea Operação #-7.381918079 #-39.36034053 2301901 BARBALHA Direito de Uso
4 Captação Superficial Operação #-4.447687222 #-38.98277479 2302909 CAPISTRANO Direito de Uso

5 rows × 66 columns

Mais 30 colunas retiradas, permanecendo 66 colunas.

Foi aplicado o método ‘select(1)’ no intuito de observar no DataFrame as colunas do tipo “object”, ou seja não numéricas, e detectar o potencial de eliminação de linhas.

#SELEcionando COLUNAS DO TIPO OBJECT
direito_de_uso.select_dtypes(include='object')
< ----- >
INT_TIN_DS INT_TSU_DS INT_TSI_DS INT_NU_LATITUDE INT_NU_LONGITUDE ING_NM_MUNICIPIO OUT_TP_OUTORGIA OUT_TP_SITUACAOOUTORGIA
0 Captação Subterrânea Operação #-4.975406649 #-37.98052819 RUSSAS Direito de Uso Outorgado
1 Captação Subterrânea Operação #-4.128452396 #-38.25173408 CASCAVEL Direito de Uso Outorgado
2 Captação Superficial Operação #-3.47118579 #-39.43222835 ITAPIPOCA Direito de Uso Outorgado
3 Captação Subterrânea Operação #-7.381918079 #-39.36034053 BARBALHA Direito de Uso Outorgado
4 Captação Superficial Operação #-4.447687222 #-38.98277479 CAPISTRANO Direito de Uso Outorgado
...
14896 Lançamento Superficial Operação #-3.779782794 #-38.6383851645 CAUCAIA Direito de Uso Outorgado
14897 Captação Subterrânea Operação #-3.8276944444 #-41.0688944444 TIANGUÁ Direito de Uso Outorgado
14898 Captação Superficial Operação #-6.0207875374 #-38.6775517217 JAGUARIBE Direito de Uso Outorgado
14899 Captação Subterrânea Operação #-7.2147777778 #-39.2693833333 JUAZEIRO DO NORTE Direito de Uso Outorgado
14900 Captação Superficial Operação #-5.1566937072 #-38.0529319643 LIMOEIRO DO NORTE Direito de Uso Outorgado

14901 rows × 15 columns

Verificando as colunas para uma visão geral.

```
#VISÃO GERAL DO CONTEÚDO DAS COLUNAS OBJETO COM OS ELEMENTOS AGRUPADOS
a = direito_de_uso.groupby(['INT_TIN_DS']).INT_TIN_DS.count()
b = direito_de_uso.groupby(['INT_TSU_DS']).INT_TSU_DS.count()
c = direito_de_uso.groupby(['INT_TSI_DS']).INT_TSI_DS.count()
d = direito_de_uso.groupby(['ING_NM_MUNICIPIO']).ING_NM_MUNICIPIO.count()
e = direito_de_uso.groupby(['OUT_TP_OUTORGA']).OUT_TP_OUTORGA.count()
f = direito_de_uso.groupby(['OUT_TP_SITUACAOOUTORGA']).OUT_TP_SITUACAOOUTORGA.count()
g = direito_de_uso.groupby(['OUT_DT_OUTORGAFINAL']).OUT_DT_OUTORGAFINAL.count()
h = direito_de_uso.groupby(['DAD_QT_VAZAODIAJAN']).DAD_QT_VAZAODIAJAN.count()
i = direito_de_uso.groupby(['FIN_TFN_DS']).FIN_TFN_DS.count()
j = direito_de_uso.groupby(['TTC_TCU_DS']).TTC_TCU_DS.count()
k = direito_de_uso.groupby(['TUC_TEC_DS']).TUC_TEC_DS.count()
l = direito_de_uso.groupby(['TUC_DS']).TUC_DS.count()
m = direito_de_uso.groupby(['CTE_TCA_DS']).CTE_TCA_DS.count()

print(a,
      b,
      c,
      d,
      e,
      f,
      g,
      h,
      i,
      j,
      k,
      l,
      m)

INT_TIN_DS
Barragem          1
Captação        14793
Lançamento       107
Name: INT_TIN_DS, dtype: int64
INT_TSU_DS
Subterrânea     8649
Superficial     6252
Name: INT_TSU_DS, dtype: int64
INT_TSI_DS
Construção        9
Operação       14604
Projeto           6
Name: INT_TSI_DS, dtype: int64
ING_NM_MUNICIPIO
ABAIARA            16
ACARAPE             12
ACARAÚ              102
ACOPIARA            46
AIUABA              44
...
URUBURETAMA        22
```

Nessa análise não há interesse nas categorias ‘Barragem’ e ‘Lançamento’ em ‘INT_TIN_DS’.

```
#VERIFICANDO A COLUNA
direito_de_uso.groupby(['INT_TIN_DS']).INT_TIN_DS.count()

INT_TIN_DS
Barragem          1
Captação        14793
Lançamento       107
Name: INT_TIN_DS, dtype: int64
```

Selecionando as linhas para descarte por presença elementos nas colunas

```
#SELEÇÃO DE LINHAS PARA EXCLUSÃO POR IRRELEVÂNCIA DO GRUPO
linhas_INT_TIN_DS = direito_de_uso.loc[direito_de_uso['INT_TIN_DS'].isin(['Barragem','Lançamento'])]

linhas_INT_TIN_DS.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 108 entries, 117 to 14896
Data columns (total 66 columns):
```

Excluindo as 108 linhas selecionadas, mantendo-se 14793 linhas.

```
#EXCLUINDO AS LINHAS SELECIONADAS
direito_de_uso.drop(linhas_INT_TIN_DS.index, inplace=True)

direito_de_uso.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14793 entries, 0 to 14900
Data columns (total 66 columns):
```

Nessa análise não há interesse nas categorias 'Construção' e 'Projeto' em 'INT_TSU_DS'.

```
#SELEÇÃO DE LINHAS PARA EXCLUSÃO POR IRRELEVÂNCIA DO GRUPO
linhas_INT_TSI_DS = direito_de_uso.loc[direito_de_uso['INT_TSI_DS'].isin(['Construção','Projeto'])]

#EXCLUINDO AS LINHAS SELECIONADAS
direito_de_uso.drop(linhas_INT_TSI_DS.index, inplace=True)
|
```

```
direito_de_uso.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14779 entries, 0 to 14900
Data columns (total 66 columns): ..
```

Excluindo as 8 linhas selecionadas, mantendo-se 14779 linhas.

Nessa análise não há interesse nas categorias "Cadastro" , 'Preventiva' e 'Outra' em 'OUT_TP_OUTORGA'.

```
#VERIFICANDO A COLUNA
direito_de_uso.groupby(['OUT_TP_OUTORGA']).OUT_TP_OUTORGA.count()
|
```

```
#SELEÇÃO DE LINHAS PARA EXCLUSÃO POR IRRELEVÂNCIA DO GRUPO
linhas_OUT_TP_OUTORGA = direito_de_uso.loc[direito_de_uso['OUT_TP_OUTORGA'].isin( ['Cadastro','Preventiva','Outra'])]

#EXCLUINDO AS LINHAS SELECIONADAS
direito_de_uso.drop(linhas_OUT_TP_OUTORGA.index, inplace=True)
```

```
direito_de_uso.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14567 entries, 0 to 14900
Data columns (total 66 columns): ..
```

Excluindo as 8 linhas selecionadas, mantendo-se 14567 linhas.

Autorizado	1
Em Análise	24
Inválido	2
Outorgado	14541
Name:	OUT_TP_SITUACAOOUTORGA

Nessa análise não há interesse nas categorias 'Autorizado', 'Em Análise' e 'Inválido' em 'OUT_TP_SITUACAOOUTORGA'.

```
#VERIFICANDO A COLUNA
direito_de_uso.groupby(['OUT_TP_SITUACAOOUTORGA']).OUT_TP_SITUACAOOUTORGA.count()

#SELEÇÃO DE LINHAS PARA EXCLUSÃO POR IRRELEVÂNCIA DO GRUPO
linhas_OUT_TP_SITUACAOOUTORGA = direito_de_uso.loc[direito_de_uso['OUT_TP_SITUACAOOUTORGA'].isin(['Autorizado', 'Em Análise', 'Inativo'])]

#EXCLUINDO AS LINHAS SELECIONADAS
direito_de_uso.drop(linhas_OUT_TP_SITUACAOOUTORGA.index, inplace=True)

direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 66 columns):
 #   Column           Non-Null Count  Dtype  

```

Excluindo as 8 linhas selecionadas, mantendo-se 14541 linhas.

Após a exclusão de alguns grupos de elementos, as colunas passaram a ter todos seus elementos iguais. Além dessas, mais outras também foram percebidas como conteúdos irrelevantes para análise, podendo ser excluídas.

Essas colunas não foram retiradas antes, para que as linhas pertencentes aos grupos que excluídos, não permanecessem no DataFrame.

COLUNAS A REMOVER VII			
OUT_TP_OUTORG	INT_TIN_DS	TUC_TEC_DS	TUC_DS
OUT_TP_SITUACAOOUTORG	INT_TSI_DS	TTC_TCU_DS	

```
#REMOVENDO COLUNAS CUJOS ELEMENTOS, APÓS EXCLUSÃO DE GRUPOS, PASSARAM A SER TODOS IGUAIS E MAIS ALGUNS IRRELEVANTES
direito_de_uso = direito_de_uso.drop(columns=['INT_TIN_DS', 'INT_TSI_DS', 'OUT_TP_OUTORG', 'OUT_TP_SITUACAOOUTORG', 'TUC_TEC_DS'])

direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 59 columns):
```

Mais 7 colunas retiradas, permanecendo 59 colunas.

Definindo o Dataframe reserva que permanecerá com as coordenadas, pois ao longo dos anos os limites municipais foram alterados e novos municípios criados. A vigência das outorgas pode ser até de 35 anos. Além disso, o próprio sistema de coordenadas adotado foi substituído.

```
#DEFININDO O DATAFRAME QUE PERMANECERÁ COM AS COORDENADAS
direito_de_uso_coordenadas=direito_de_uso

direito_de_uso_coordenadas.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 59 columns):
```

Removendo as coordenadas, pois será agrupado por município e já tem-se uma coluna identificando nominalmente e outra por codificação do IBGE.

COLUNAS A REMOVER VII	
DAD_QT_VAZAODIAJAN	INT_NU_LONGITUDE

```
#REMOVENDO COLUNAS COORDENADAS
direito_de_uso = direito_de_uso.drop(columns=['INT_NU_LATITUDE', 'INT_NU_LONGITUDE'], )

direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 57 columns):
```

Mais 2 colunas retiradas, permanecendo 57 colunas.

Definindo o Dataframe reserva que permanecerá com o manejo (dias por mês e horas por dia) e com as vazões mensais, pois poderá ser utilizado para obter os volumes mensais, que variam ao longo do ano nos casos de irrigação e da aquicultura em tanques escavados em virtude do ciclo produtivo e também com a oferta pluvial. Esse aprofundamento não será observado nesse momento, os volumes anuais poderão embasar essa análise.

```
#DEFININDO O DATAFRAME QUE PERMANECERÁ COM AS VAZÕES MENSais
direito_de_uso_vazoes_mensais=direito_de_uso

direito_de_uso_vazoes_mensais.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
```

Esses dados apresentam ‘DAD’ na nomenclatura dos cabeçalhos, podendo essas colunas serem eliminadas sem necessidade verificação de conteúdo

```
#OBTENDO UMA LISTA COM AS COLUNAS TIPO OBJETO
w=direito_de_uso.select_dtypes(include='float64')
list(w.columns.values)
```

COLUNAS A REMOVER VIII			
DAD_QT_VAZAODIAJAN	DAD_QT_VAZAODIAOUT	DAD_QT_HORASJUL	DAD_QT_DIAABR
DAD_QT_VAZAODIAFEV	DAD_QT_VAZAODIANOV	DAD_QT_HORASAGO	DAD_QT_DIAMAI
DAD_QT_VAZAODIAMAR	DAD_QT_VAZAODIADEZ	DAD_QT_HORASSET	DAD_QT_DIAJUN
DAD_QT_VAZAODIAABR	DAD_QT_HORASJAN	DAD_QT_HORASOUT	DAD_QT_DIAJUL
DAD_QT_VAZAODIAMAI	DAD_QT_HORASFEV	DAD_QT_HORASNOM	DAD_QT_DIAAGO
DAD_QT_VAZAODIAJUN	DAD_QT_HORASMAR	DAD_QT_HORASDEZ	DAD_QT_DIASET
DAD_QT_VAZAODIAJUL	DAD_QT_HORASABR	DAD_QT_DIAJAN	DAD_QT_DIAOUT
DAD_QT_VAZAODIAAGO	DAD_QT_HORASMAI	DAD_QT_DIAFEV	DAD_QT_DIANOV
DAD_QT_VAZAODIASET	DAD_QT_HORASJUN	DAD_QT_DIAMAR	DAD_QT_DIADEZ

```
#REMOVENDO COLUNAS MANEJO E VAZÕES MENSais
direito_de_uso = direito_de_uso.drop(columns=['DAD_QT_VAZAODIAJAN',
'DAD_QT_VAZAODIAFEV',
'DAD_QT_VAZAODIAMAR',
'DAD_QT_VAZAODIAABR',
'DAD_QT_VAZAODIAMAI',
'DAD_QT_VAZAODIAJUN',
'DAD_QT_VAZAODIAJUL',
'DAD_QT_VAZAODIAAGO',
'DAD_QT_VAZAODIASET',
'DAD_QT_VAZAODIAOUT',
'DAD_QT_VAZAODIANOV',
'DAD_QT_VAZAODIADEZ',
'DAD_QT_HORASJAN'],
'DAD_QT_HORASFEV',
'DAD_QT_HORASMAR',
'DAD_QT_HORASABR',
'DAD_QT_HORASMAI',
'DAD_QT_HORASJUN',
'DAD_QT_HORASJUL',
'DAD_QT_HORASAGO',
'DAD_QT_HORASSET',
'DAD_QT_HORASOUT',
'DAD_QT_HORASNNOV',
'DAD_QT_HORASDEZ',
'DAD_QT_DIAJAN',
'DAD_QT_DIAFEV',
'DAD_QT_DIAMAR',
'DAD_QT_DIAABR',
'DAD_QT_DIAMAI',
'DAD_QT_DIAJUN',
'DAD_QT_DIAJUL',
'DAD_QT_DIAAGO',
'DAD_QT_DIASET',
'DAD_QT_DIAOUT',
'DAD_QT_DIANOV',
'DAD_QT_DIADEZ',], )
```

Mais 36 colunas retiradas, permanecendo 21 colunas.

Há uma coluna com o volume anual que poderá embasar essa análise, podendo ser dispensada duas com as vazões máxima e média, pois esse refinamento não será abordado.

COLUNAS A REMOVER IX	
INT_QT_VAZAOMAXIMA	INT_QT_VAZAOMEDIA

```
#REMOVENDO COLUNAS VAZÕES MÉDIAS E MÁXIMAS
direito_de_uso = direito_de_uso.drop(columns=['INT_QT_VAZAOMAXIMA',
'INT_QT_VAZAOMEDIA'], )
direito_de_uso.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 19 columns):
```

Mais 2 colunas retiradas, permanecendo 19 colunas.

Verificando as colunas

```
direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   INT_TSU_DS       14541 non-null   object  
 1   ING_NU_IBGEMUNICIPIO 14541 non-null   int64   
 2   ING_NM_MUNICIPIO    14541 non-null   object  
 3   OUT_DT_OUTORGAFINAL 14540 non-null   object  
 4   OUT_DT_OUTORGAINICIAL 14540 non-null   datetime64[ns] 
 5   INT_QT_VOLUMEANUAL  14493 non-null   float64 
 6   FIN_TFN_DS        14539 non-null   object  
 7   FES_NU_AREATOTALTANQUE 57 non-null    float64 
 8   IUS_NU_ALTURARES   7 non-null     float64 
 9   IUS_NU_AREARESMAX  158 non-null   float64 
 10  IUS_NU_VOLUMERES   772 non-null   float64 
 11  IUS_NU_COEFICIENTE_RETORNO 10983 non-null float64 
 12  ETP_NU_QUANTIDADEMAXMENSAL 1 non-null   float64 
 13  SIR_NU_AREAIRRIGADA  3919 non-null   float64 
 14  FIA_NU_POPULACAOATENDIDA 863 non-null   float64 
 15  FRE_NU_VOLUMENA    0 non-null    float64 
 16  HTE_NU_QUANTIDADE  3 non-null    float64 
 17  CTE_TCA_DS        197 non-null   object  
 18  CTE_NU_CABECAS    197 non-null   float64 
dtypes: datetime64[ns](1), float64(12), int64(1), object(5)
memory usage: 2.2+ MB
```

```
#OBSERVANDO OS GRUPOS DE ELEMENTOS NA COLUNA FINALIDADE DE DIREITO DE USO

direito_de_uso.groupby(['FIN_TFN_DS', ]).FIN_TFN_DS.count()

FIN_TFN_DS
70445                  1
115077.2                1
251850                  1
253440                  1
Abastecimento Público      3402
Aquicultura em Tanque Escavado 533
Aquicultura em Tanque Rede    54
Consumo Humano              5
Criação Animal               1156
Indústria                     1834
Irrigação                   5083
Mineração - Extração de AreiaCascalho em Leito de Rio 69
Mineração - Outros Processos Extrativos 1
Outras                      2396
Serviços                     2
Name: FIN_TFN_DS, dtype: int64
```

Para excluir mais algumas colunas, o valor '0,01' foi adotado no método pois implica dizer que o número de dados preenchidos por coluna é inferior a uma linha ($14.541 \times 0,01 = 145,41$).

Realizado esse contato de reconhecimento e observação do comportamento da planilha, partiu-se para a implementação dos *scripts*.

O dataset em questão costuma receber novos dados, alterações nos antigos dados, assim como também de novos campos são introduzidos e outros caem em desuso, então o script tenta lidar com essas peculiaridades dentro do que foi observado de maneira que outras versões possam ser utilizadas nas análises.

Na reparação inicial importou-se as bibliotecas e definindo de uma forma mais abreviada para indicar de qual biblioteca será retirada a função.

```
#IMPORTANDO AS BIBLIOTECAS
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import pickle
```

Carregando o arquivo Excel e gerando o Dataframe no Pandas

```
#CARREGANDO DATASET
direito_de_uso_original=pd.read_excel("direito_de_uso_.xlsx")
```

Verificando as informações do DF gerado

```
#VERIFICANDO AS INFORMAÇÕES DO DF ORIGINAL
print("direito_de_uso_original")
direito_de_uso_original.info()

direito_de_uso_original
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 228 entries, INT_CD_CNARH40 to OTO_DS_OUTROUSO
dtypes: float64(157), int64(6), object(65)
memory usage: 25.9+ MB
```

Verificando a presença de registros repetidos.

```
#VERIFICAR A PRESENÇA DE REGISTROS DUPLICADOS REPETIDOS
repetidos = direito_de_uso_original[direito_de_uso_original.duplicated(keep='first')]
```

```
print ('Registros repetidos:')
repetidos

Registros duplicados

INT_CD_CNARH40  INT_CD_REGLA  INT_TIN_DS  INT_TIN_CD  INT_TSU_DS  INT_TSU_CD  INT_TCH_DS  INT_TSI_DS  INT_TSI_CD  INT_TOD_DS ...  AMA_QT_
0 rows x 228 columns
```

Criando uma cópia do DF original.

```
#CRIANDO CÓPIA DO DF PRESERVANDO O ORIGINAL
direito_de_uso=direito_de_uso_original
```

Verificando as informações do DF gerado.

```
#VERIFICANDO AS INFORMAÇÕES DO DF CÓPIA DO ORIGINAL
print("DF direito_de_uso é a cópia de DF direito_de_uso_original")
direito_de_uso.info()
```

DF direito_de_uso é a cópia de DF direito_de_uso_original
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 228 entries, INT_CD_CNARH40 to OTO_DS_OUTROUSO
dtypes: float64(157), int64(6), object(65)
memory usage: 25.9+ MB

Eliminando colunas vazias.

```
#OBTENDO O PERCENTUAL IDEAL
Linhas=len(direito_de_uso_original)-1
ndig=len(str(Linhas))
perc=1/(10**ndig)

#DESCARTAR AS COLUNAS COM 100% DE VALORES AUSENTES
direito_de_uso=direito_de_uso_original.dropna(thresh=len(direito_de_uso_original)*perc, axis=1)

direito_de_uso.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 185 entries, INT_CD_CNARH40 to OTO_DS_OUTROUSO
dtypes: float64(114), int64(6), object(65)
memory usage: 21.0+ MB

Criando uma lista com o nome de todas as colunas.

```
#CRIANDO UMA LISTA COM OS NOMES DAS COLUNAS |
colunas_direito_de_uso=direito_de_uso.columns.values.tolist()

print(colunas_direito_de_uso)

['INT_CD_CNARH40', 'INT_TIN_DS', 'INT_TIN_CD', 'INT_TSU_DS', 'INT_TSU_CD', 'INT_TSI_DS', 'INT_TSI_CD', 'INT_TOD_DS', 'INT_TOM_DS', 'INT_NU_CNARH', 'INT_NU_SIAGAS', 'INT_NU_LATITUDE', 'INT_NU_LONGITUDE', 'ING_NU_IBGEMUNICIPIO', 'ING_SG_UFMUNICIPIO', 'ING_NM_MUNICIPIO', 'INT_NM_CORPOHIDRICO', 'INT_NM_CORPOHIDRICOALTERADO', 'INT_DS_ORGAO', 'INT_DT_REGISTRO', 'INT_CD_DCLARACAO', 'INT_DS_OPCIONAL', 'EMP_NM_EMPREENDIMENTO', 'EMP_NM_USUARIO', 'EMP_NU_CPFNPJ', 'EMP_DS_EMAILRESPONSABEL', 'EMP_NU_CEPENDERECO', 'EMP_CD_IBGEMUNCORRESPONDENCIA', 'EMP_DS_LOGRADOURO', 'EMP_DS_COMPLEMENTOENDERECO', 'EMP_NU_LOGRADOURO', 'EMP_NU_CAIAXAPOSTAL', 'EMP_DS_BAIRRO', 'EMP_NU_DDD', 'EMP_NU_TELEFONE', 'EMP_SG_UF', 'EMP_NM_MUNICIPIO', 'OUT_TP_OUTORGAA', 'OUT_TPO_CD', 'OUT_TP_SITUACAOOUTORGAA', 'OUT_TSP_CD', 'OUT_DT_OUTORGAFINAL', 'OUT_DT_OUTORGAINICIAL', 'OUT_NU_PROCESSO', 'OUT_TP_ATO', 'O_UT_NU_ATO', 'DAD_QT_VAZAODIAJAN', 'DAD_QT_VAZAODIAFEV', 'DAD_QT_VAZAODIAMAR', 'DAD_QT_VAZAODIAABR', 'DAD_QT_VAZAODIAmai', 'DAD_QT_VAZAODIAJUN', 'DAD_QT_VAZAODIAJUL', 'DAD_QT_VAZAODIAAGO', 'DAD_QT_VAZAODIASET', 'DAD_QT_VAZAODIAOUT', 'DAD_QT_VAZAODIANOV', 'DAD_QT_VAZAODIADEZ', 'DAD_QT_HORASJAN', 'DAD_QT_HORADEV', 'DAD_QT_HORASMAR', 'DAD_QT_HORASABR', 'DAD_QT_HORASMAI', 'DAD_QT_HORASJUN', 'DAD_QT_HORASAGO', 'DAD_QT_HORASSET', 'DAD_QT_HORASOUT', 'DAD_QT_HORASNOV', 'DAD_QT_HORASDEZ', 'DAD_QT_DIAZAN', 'DAD_QT_DIAFEV', 'DAD_QT_DIAMAR', 'DAD_QT_DIAABR', 'DAD_QT_DIAmai', 'DAD_QT_DIAJUN', 'DAD_QT_DIAJUL', 'DAD_QT_DIAAGO', 'DAD_QT_DIASET', 'DAD_QT_DIAOUT', 'DAD_QT_DIANOV', 'DAD_QT_DIADEZ', 'INT_QT_VAZAOMAXIMA', 'INT_QT_VAZAOMEDIA', 'INT_QT_VOLUMEANUAL', 'FIN_TFN_DS', 'FIN_TFN_CD', 'FES_NU_PROFUNDIDADEMEDIATANQUE', 'FES_NU_AREATOTALTANQUE', 'TTC_TCU_DS', 'TTC_TCU_CD', 'FIE_TPS_DS', 'FIE_TPS_CD', 'FPE_CNA_CD', 'IUS_NU_ALTURARES', 'IUS_NU_AREARESMAX', 'IUS_NU_VOLUMERES', 'IUS_NU_COEFICIENTE_RERNO', 'IUS_NM_ENTIDADEDECONCEDENTE', 'IUS_NU_CONCESSAO', 'IUS_DT_FINALCONCESSAO', 'ETP_MPE_DS', 'ETP_MPE_CD', 'ETP_NU_QUANTIDADEDEM_AXMENSAL', 'SIR_TSI_DS', 'SIR_TSI_CD', 'SIR_TCT_DS', 'SIR_TCT_CD', 'SIR_NU_AREAIRRIGADA', 'FTA_NU_POPULACAOATENDIDA', 'FRE_NU_VOLUMENA', 'HTE_NU_QUANTIDADEDE', 'TUC_TEC_DS', 'TUC_TEC_CD', 'TUC_DS', 'TUC_CD', 'CTE_TSC_DS', 'CTE_TSC_CD', 'CTE_TCA_DS', 'CTE_TCACD', 'CTE_NU_CABECAS', 'EFL_NU_DBOBRUTO', 'EFL_NU_DBOTRATADO', 'EFL_NU_NITROGENIOBRUTO', 'EFL_TTE_DS', 'EFL_TTE_CD', 'ITC_TUM_DS', 'ITC_NU_PRODUCAOANUAL', 'CNA_DS', 'CNA_CD_CNAE', 'ASB_DT_INSTALACAO', 'ASB_TNP_DS', 'ASB_TNP_CD', 'ASB_NU_DIAMETROPERFURACAO', 'ASB_NU_DIAMETROFILTRO', 'ASB_AOP_DS', 'ASB_AOP_CD', 'ASB_NU_TOPO', 'ASB_NU_BASE', 'ASB_TPN_DS', 'ASB_TPN_CD', 'ABS_TCA_DS', 'ABS_TCA_CD', 'ASB_NU_PROFUNDIDADEDEFINAL', 'ASB_NU_ALTURABOCATUBO', 'ASB_NU_COTATERRENO', 'TST_DT', 'TST_TTB_DS', 'TST_TTB_CD', 'TST_DS_TEMPODURACAO', 'TST_NU_ND', 'TST_NU_NE', 'TST_VZ_ESTABILIZACAO', 'TST_TMI_DS', 'TST_TMI_CD', 'TST_NU_PERMEABILIDADE', 'AMA_DT_COLETA', 'AMA_DT_ANALISE', 'AMA_NU_CONDUVIDADEELETTRICA', 'AMA_QT_TEMPERATURA', 'AMA_QT_STD', 'AMA_QT_PPH', 'AMA_QT_COLIFORMESTOTAIS', 'AMA_QT_COLIFORMESFECALIS', 'AMA_QT_BICARBONATO', 'AMA_QT_CALCIO', 'AMA_QT_CARBONATO', 'AMA_QT_CLORETO', 'AMA_QT_DUREZATOTAL', 'AMA_QT_FERROTOTAL', 'AMA_QT_FLUORETOS', 'AMA_QT_NITRATOS', 'AMA_QT_NITRITOS', 'AMA_QT_POTASSIO', 'AMA_QT_SODIO', 'AMA_QT_SULFATO', 'AMA_QT_MAGNESIO', 'DATA_EXTRACAO', 'ING_CD_OTTOBACIA_TRECHO', 'ING_CD_COMITEESTADUAL', 'ING_NM_COMITEESTADUAL', 'ING_CS_CONAMA', 'OTO_DS_OUTROUSO']
```

Transformando a lista de colunas em *string*.

```
#TRANSFORMANDO A LISTA DE COLUNAS EM STRING E EXCLUINDO CARACTERES
lista_col_texto=str(colunas_direito_de_uso)#Definindo como string
lista_col_texto=lista_col_texto.replace(', ','')#Excluindo vírgulas
lista_col_texto=lista_col_texto.replace("''", "")#Excluindo apóstrofos
lista_col_texto=lista_col_texto.replace("[", "")#Excluindo colchetes
lista_col_texto=lista_col_texto.replace("]", "")#Excluindo colchetes

lista_col_texto

'INT_CD_CNARH40 INT_TIN_DS INT_TIN_CD INT_TSU_DS INT_TSU_CD INT_TCH_DS INT_TSI_DS INT_TSI_CD INT_TOD_DS INT_TDM_DS INT_NU_CNARH
INT_NU_SIAGAS INT_NU_LATITUDE INT_NU_LONGITUDE ING_NU_IBGEMUNICIPIO ING_SG_UFMUNICIPIO ING_NM_MUNICIPIO INT_NM_CORPOHIDRICO INT
_NM_CORPOHIDRICOALTERADO INT_DS_ORGAO INT_DT_REGISTRO INT_CD_DECLARACAO INT_DS_OPCIONAL EMP_NM_EMPREENDIMENTO EMP_NM_USUARIO EM
P_NU_CPFNPJ EMP_DS_EMAILRESPONSAVEL EMP_NU_CEPENDERECHO EMP_CD_IBGEMUNCORRESPONDENCIA EMP_DS_LOGRADOURO EMP_DS_COMPLEMENTOENDER
ECO EMP_NU_LOGRADOURO EMP_NU_CAIXPASTAL EMP_DS_BAIRRO EMP_NU_DDD EMP_NU_TELEFONE EMP_SG_UF EMP_NM_MUNICIPIO OUT_TP_OUTORG
OUT_TPO_CD OUT_TP_SITUACAOOUTORGIA OUT_TSP_CD OUT_DT_OUTORGAFINAL OUT_DT_OUTORGAINICIAL OUT_NU_PROCESSO OUT_TP_ATO OUT_NU_ATO DAD_Q
T_VAZAODIAJAN DAD_QT_VAZAODIAFEV DAD_QT_VAZAODIAMAR DAD_QT_VAZAODIAABR DAD_QT_VAZAODIAJUN DAD_QT_VAZAODIAJUL
DAD_QT_VAZAODIAAGO DAD_QT_VAZAODIASET DAD_QT_VAZAODIAOUT DAD_QT_VAZAODIANOV DAD_QT_VAZAODIADEZ DAD_QT_HORASJUN DAD_QT_HORASFEV
DAD_QT_HORASMAR DAD_QT_HORASABR DAD_QT_HORASMAI DAD_QT_HORASJUN DAD_QT_HORASJUL DAD_QT_HORASAGO DAD_QT_HORASSET DAD_QT_HORASOUT
DAD_QT_HORASNOV DAD_QT_HORASDEZ DAD_QT_DIAJAN DAD_QT_DIAFEV DAD_QT_DIAMAR DAD_QT_DIAABR DAD_QT_DIAIMAI DAD_QT_DIAJUN DAD_QT_DIAJ
UL DAD_QT_DIAAGO DAD_QT_DIASET DAD_QT_DIAOUT DAD_QT_DIANOV DAD_QT_DIADEZ INT_QT_VAZAOMAXIMA INT_QT_VAZAOMEDIA INT_QT_VOLUMEANUA
L FIN_TFN_DS FIN_TFN_CD FES_NU_PROFUNDIDADEMEDIATANQUE FES_NU_AREATOTALTANQUE TTC_TCU_DS TTC_TCU_CD FIE_TPS_DS FIE_TPS_CD FPE_C
NA_CD_IUS_NU_ALTURARES IUS_NU_AREARESMAX IUS_NU_VOLUMERES IUS_NU_COEFICIENTE_RETORNO IUS_NM_ENTIDADEDECONCEDENTE IUS_NU_CONCESSAO
IUS_DT_FINALCONCESSAO ETP_MPE_DS ETP_MPE_CD ETP_NU_QUANTIDADEXMAXMENSAL SIR_TSI_DS SIR_TSI_CD SIR_TCT_DS SIR_TCT_CD SIR_NU_AREAI
RRIGADA FIA_NU_POPULACAOATENDIDA FRE_NU_VOLUMENA HTE_NU_QUANTIDADE TUC_TEC_DS TUC_TEC_CD TUC_DS TUC_CD CTE_TSC_DS CTE_TSC_CD CT
E_TCA_DS CTE_TCA_CD CTE_NU_CABECAS EFL_NU_DBOBRUTO EFL_NU_DBOTRATADO EFL_NU_NITROGENIOBRUTO EFL_TTE_DS EFL_TTE_CD ITC_TUM_DS IT
C_TUM_CD ITC_NU_PRODUCAOANUAL CNA_DS CNA_CD_CNAE ASB_DT_INSTALACAO ASB_TNP_DS ASB_NU_DIAMETROPURFACAO ASB_NU_DIAME
TROFILTR0 ASB_AQP_DS ASB_AQP_CD ASB_NU_TOPO ASB_NU_BASE ASB_TPN_DS ASB_TPN_CD ABS_TCA_DS ABS_TCA_CD ASB_NU_PROFUNDIDADEFINAL AS
B_NU_ALTURABOCATUBO ASB_NU_COTATERRENO TST_DT TST_TTB_DS TST_TTB_CD TST_DS_TEMPODURACAO TST_NU_ND TST_NU_NE TST_V2_ESTABILIZAC
O TST_TMI_DS TST_TMI_CD TST_NU_PERMEABILIDADE AMA_DT_COLETA AMA_DT_ANALISE AMA_NU_CONDUTIVIDADEELETTRICA AMA_QT_TEMPERATURA AMA_
QT_STD AMA_QT_PH AMA_QT_COLIFORMESTOTAIS AMA_QT_COLIFORMESFECAIS AMA_QT_BICARBONATO AMA_QT_CALCIO AMA_QT_CARBONATO AMA_QT_CLORE
TO AMA_QT_DUREZATOTAL AMA_QT_FERROTOTAL AMA_QT_FLUORETOS AMA_QT_NITRATOS AMA_QT_NITRITOS AMA_QT_POTASSIO AMA_QT_SODIO AMA_QT_SU
LFATO AMA_QT_MAGNESIO DATA_EXTRACAO ING_CD_OTTOBACIA TRECHO ING_CD_COMITEESTADUAL ING_NM_COMITEESTADUAL ING_CS_CONAMA OTO_DS_OU
TROUSO'
```

Filtrando e criando conjunto com as colunas com dados pessoais e de contato dos usuários.

```
#SEPARANDO AS COLUNAS COM OS DADOS DOS USUÁRIO DA LISTA DE COLUNAS DO DATASET, ELAS INICIAM COM 'EMP_'
usuario_filtro = lambda x: 'EMP_' in x # Define a função de filtro
dados_usuarios = filter(usuario_filtro, lista_col_texto.split()) # Filtra
conj_col_dados_usuarios = set(dados_usuarios) # Cria resultado em Conjunto

print(conj_col_dados_usuarios)

{'EMP_SG_UF', 'EMP_DS_COMPLEMENTOENDERECO', 'EMP_NU_CPFNPJ', 'EMP_NU_CEPENDERECHO', 'EMP_DS_EMAILRESPONSAVEL', 'EMP_NM_USUARIO', 'EMP_NM_EMPREENDIMENTO', 'EMP_NU_DDD', 'EMP_DS_LOGRADOURO', 'EMP_NM_MUNICIPIO', 'EMP_DS_BAIRRO', 'EMP_NU_CAIXPASTAL', 'EMP_CD_IBGEMUNCORRESPONDENCIA', 'EMP_NU_TELEFONE', 'EMP_NU_LOGRADOURO'}
```

Separando as colunas com dados de códigos.

```
#SEPARANDO AS COLUNAS COM OS CÓDIGOS DE CATEGORIA DE DADOS JÁ PRESENTES EM OUTRA COLUNA DO DATAFRAME,
#ELAS APRESENTAM 'CD' NA NOMENCLATURA
codigo_filtro = lambda x: 'CD' in x # Define a função de filtro
dados_codigo = filter(codigo_filtro, lista_col_texto.split()) # Filtra
conj_col_codigos = set(dados_codigo) # Cria resultado em Conjunto

print(conj_col_codigos)

{'ASB_TPN_CD', 'TST_TMI_CD', 'OUT_TSP_CD', 'SIR_TSI_CD', 'ETP_MPE_CD', 'TUC_CD', 'EMP_CD_IBGEMUNCORRESPONDENCIA', 'INT_CD_DECLARACAO', 'ABS_TCA_CD', 'SIR_TCT_CD', 'CTE_TSC_CD', 'ING_CD_COMITEESTADUAL', 'TTC_TCU_CD', 'TST_TTB_CD', 'FIN_TFN_CD', 'INT_TSU_CD', 'ASB_TNP_CD', 'OUT_TPO_CD', 'TUC_TEC_CD', 'CTE_TCA_CD', 'INT_CD_CNARH40', 'EFL_TTE_CD', 'INT_TSI_CD', 'INT_TIN_CD', 'CNA_CD_CNAE', 'FPE_CNA_CD', 'ING_CD_OTTOBACIA TRECHO', 'ITC_TUM_CD', 'FIE_TPS_CD', 'ASB_AQP_CD'}
```

Separando as colunas com dados de qualidade da água.

```
#SEPARANDO AS COLUNAS COM OS DADOS DE QUALIDADE DA ÁGUA DA LISTA DE COLUNAS DO DATASET, ELAS INICIAM COM 'AMA_'
qualidade_filtro = lambda x: 'AMA_' in x # Define a função de filtro
dados_qualidade = filter(qualidade_filtro, lista_col_texto.split()) # Filtra
conj_col_dados_qualidade = set(dados_qualidade) # Cria resultado em Conjunto

print(conj_col_dados_qualidade)

{'AMA_QT_COLIFORMESFECALIS', 'AMA_QT_MAGNESIO', 'AMA_QT_PH', 'AMA_QT_DUREZATOTAL', 'AMA_QT_STD', 'AMA_QT_SULFATO', 'AMA_QT_NITRATO', 'AMA_QT_CARBONATO', 'AMA_QT_BICARBONATO', 'AMA_QT_TEMPERATURA', 'AMA_QT_CLORETO', 'AMA_QT_POTASSIO', 'AMA_DT_ANALISE', 'AMA_NU_CONDUTIVIDADEELETTRICA', 'AMA_QT_NITRITOS', 'AMA_QT_COLIFORMESTOTAIS', 'AMA_QT_FERROTOTAL', 'AMA_QT_CALCIO', 'AMA_DT_COLETA', 'AMA_QT_FLUORETOS', 'AMA_QT_SODIO'}
```

Separando as colunas com dados de poços.

```
#SEPARANDO AS COLUNAS COM OS DADOS DOS POÇOS DA LISTA DE COLUNAS DO DATASET, ELAS INICIAM COM 'TST_' OU 'ASB_'
poco_filtro = lambda x: 'ASB_' in x or 'TST_' in x # Define a função de filtro
dados_poco = filter(poco_filtro, lista_col_texto.split()) # Filtra
conj_col_dados_poco = set(dados_poco) # Cria resultado em Conjunto

print(conj_col_dados_poco)

{'ASB_TPN_CD', 'TST_TMI_CD', 'ASB_NU_DIAMETROPERFURACAO', 'ASB_NU_DIAMETROFILTRO', 'TST_DS_TEMPODURACAO', 'ASB_NU_TOPO', 'TST_TMI_DS', 'ASB_NU_BASE', 'TST_DT', 'ASB_AQP_DS', 'TST_TTB_CD', 'ASB_NU_COTATERRENO', 'ASB_DT_INSTALACAO', 'ASB_TNP_CD', 'TST_NU_PERMEABILIDADE', 'ASB_TNP_DS', 'TST_NU_NE', 'ASB_NU_PROFUNDIDADEFINAL', 'ASB_TPN_DS', 'TST_TTB_DS', 'TST_NU_ND', 'ASB_NU_ALTURABOCATUBO', 'TST_VZ_ESTABILIZACAO', 'ASB_AQP_CD'}
```

Separando as colunas com dados de lançamento de efluentes.

```
#SEPARANDO AS COLUNAS COM OS DADOS SOBRE EFLUENTES DA LISTA DE COLUNAS DO DATASET, ELAS INICIAM COM 'EFL_'
efluente_filtro = lambda x: 'EFL_' in x # Define a função de filtro
dados_efluente = filter(efluente_filtro, lista_col_texto.split()) # Filtra
conj_col_dados_efluente = set(dados_efluente) # Cria resultado em Conjunto

print(conj_col_dados_efluente)

{'EFL_TTE_DS', 'EFL_NU_DBOTRATADO', 'EFL_NU_NITROGENIOBRUTO', 'EFL_TTE_CD', 'EFL_NU_DBOBRUTO'}
```

Separando as colunas com dados sobre comitês.

```
#SEPARANDO AS COLUNAS COM OS DADOS SOBRE COMITÉS DA LISTA DE COLUNAS DO DATASET, ELAS APRESENTAM 'COMITE'
comite_filtro = lambda x: 'COMITE' in x # Define a função de filtro
dados_comite = filter(comite_filtro, lista_col_texto.split()) # Filtra
conj_col_dados_comite = set(dados_comite) # Cria resultado em Conjunto

print(conj_col_dados_comite)

{'ING_NM_COMITEESTADUAL', 'ING_CD_COMITEESTADUAL'}
```

Separando as colunas com dados sobre ID's.

```
#SEPARANDO AS COLUNAS COM OS DADOS SOBRE ID'S DE INSTITUIÇÕES E SISTEMAS DE CADASTRO DA LISTA DE COLUNAS DO DATASET,
#ELAS APRESENTAM COM 'REGLA', 'CNARH', 'SIAGAS', 'PROCESSO', 'NI_ATO', 'CNA', 'CONAMA'
identificadores_filtro = lambda x: 'REGLA' in x or 'CNARH' in x or 'SIAGAS' in x or 'PROCESSO' in x or 'NU_ATO' in x or 'CNA' in x
dados_identificadores = filter(identificadores_filtro, lista_col_texto.split()) # Filtra
conj_col_dados_identificadores = set(dados_identificadores) # Cria resultado em Conjunto

print(conj_col_dados_identificadores)

{'FPE_CNA_CD', 'OUT_NU_ATO', 'INT_NU_CNARH', 'ING_CS_CONAMA', 'INT_CD_CNARH40', 'OUT_NU_PROCESSO', 'CNA_DS', 'INT_NU_SIAGAS', 'CNA_CD_CNAE'}
```

Unificando selecionados em conjuntos para remover .

```
print(conj_col_remover1)

{'EMP_NU_CPF_CNPJ', 'EMP_NM_EMPREENDIMENTO', 'EFL_NU_DBOTRATADO', 'EFL_NU_NITROGENIO_BRUTO', 'ASB_NU_DIAMETRO_PERFURACAO', 'OUT_NU_Processo', 'ASB_NU_TOPO', 'INT_CD_DECLARACAO', 'ABS_TCA_CD', 'AMA_QT_PH', 'AMA_QT_SULFATO', 'AMA_QT_STD', 'TST_TMI_DS', 'AMA_QT_CARBONATO', 'TST_DT', 'TTC_TCU_CD', 'AMA_QT_BICARBONATO', 'ASB_AQP_DS', 'ASB_NU_COTATERRENO', 'EMP_NU_CAIXA_POSTAL', 'FIN_TFN_CD', 'EMP_NU_LOGRADOURO', 'INT_TSU_CD', 'AMA_QT_TEMPERATURA', 'TST_NU_PERMEABILIDADE', 'EMP_DS_EMAIL_RESPONSAVEL', 'EMP_NM_USUARIO', 'AMA_QT_CLORETO', 'AMA_QT_POTASSIO', 'TUC_TEC_CD', 'TST_NU_NE', 'EMP_DS_LOGRADOURO', 'AMA_QT_COLIFORMESTOTAIS', 'ING_CD_OT_TOBACIA_TRECHO', 'AMA_QT_CALCIO', 'OUT_NU_ATO', 'AMA_QT_FLUORETOS', 'AMA_QT_SODIO', 'EMP_NM_MUNICIPIO', 'INT_NU_SIAGAS', 'ASB_AQP_CD', 'EMP_NU_CEPENDERECO', 'AMA_QT_COLIFORMES_FCAIS', 'ASB_TPN_CD', 'AMA_QT_MAGNESIO', 'TST_TMI_CD', 'OUT_TSP_CD', 'EFL_TTE_DS', 'SIR_TSI_CD', 'ETP_MPE_CD', 'ASB_NU_DIAMETRO_FILTRO', 'TST_DS_TEMP_DURACAO', 'TUC_CD', 'EMP_CD_IBGE_MUNICORRESPONDENCIA', 'SIR_TCT_CD', 'EMP_NU_DDD', 'AMA_QT_DUREZA_TOTAL', 'CTE_TSC_CD', 'AMA_QT_NITRATOS', 'ASB_NU_BASE', 'ING_CD_COMITE_ESTADUAL', 'ING_CS_CONAMA', 'TST_TTB_CD', 'EMP_DS_BAIRRO', 'ASB_DT_INSTALACAO', 'ASB_TNP_CD', 'EMP_DS_COMPLEMENTO_E_DERECHO', 'OUT_TPO_CD', 'INT_NU_CNARH', 'CTE_TCA_CD', 'ASB_TNP_DS', 'INT_CD_CNARH40', 'EFL_TTE_CD', 'AMA_DT_ANALISE', 'INT_TSI_CD', 'EFL_NU_DBORUTO', 'AMA_NU_CONDUTIVIDADE_ELETTRICA', 'AMA_QT_NITRITOS', 'INT_TIN_CD', 'ASB_NU_PROFUNDIDADEFINAL', 'CNA_DS', 'AMA_QT_FERRO_TOTAL', 'CNA_CD_CNAE', 'EMP_SG_UF', 'FPE_CNA_CD', 'ASB_TPN_DS', 'ITC_TUM_CD', 'TST_TTB_DS', 'TST_NU_ND', 'AMA_DT_COLETA', 'FIE_TPS_CD', 'ING_NM_OMITE_ESTADUAL', 'ASB_NU_ALTURA_BOCATUBO', 'TST_VZ_ESTABILIZACAO', 'EMP_NU_TELEFONE'}
```

Não foram usados os dados mensais nessa versão, pois nem foi dado foco a demanda anual.

```
#DEFININDO DF QUE PERMANECERÁ COM OS DADOS MENSais
direito_de_uso_mensal=direito_de_uso
```

Alguns os prints podem não estar conforme o script atualizado.

```
#VERIFICANDO A COLUNA
direito_de_uso.groupby(['OUT_TP_SITUACAO_OUTORGAS']).OUT_TP_SITUACAO_OUTORGAS.count()

#SELEÇÃO DE LINHAS PARA EXCLUSÃO POR IRRELEVÂNCIA DO GRUPO
linhas_OUT_TP_SITUACAO_OUTORGAS = direito_de_uso.loc[direito_de_uso['OUT_TP_SITUACAO_OUTORGAS'].isin(['Autorizado', 'Em Análise', 'Inativo'])]

#EXCLUINDO AS LINHAS SELECIONADAS
direito_de_uso.drop(linhas_OUT_TP_SITUACAO_OUTORGAS.index, inplace=True)

#REMOVENDO COLUNAS CUJOS ELEMENTOS, APÓS EXCLUSÃO DE GRUPOS, PASSARAM A SER TODOS IGUAIS E MAIS ALGUNS IRRELEVANTES
direito_de_uso = direito_de_uso.drop(columns=['INT_TIN_DS', 'INT_TSI_DS', 'OUT_TP_OUTORGAS', 'OUT_TP_SITUACAO_OUTORGAS', 'TUC_TEC_DS'])

#DEFININDO O DATAFRAME QUE PERMANECERÁ COM AS COORDENADAS
direito_de_uso_coordenadas=direito_de_uso
```

```
direito_de_uso_original
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 228 entries, INT_CD_CNARH40 to OTO_DS_OUTROUSO
dtypes: float64(157), int64(6), object(65)
memory usage: 25.9+ MB
conj col a remover1
conj col a remover2
```

Verificando as colunas e linhas que permaneceram.

```
direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   INT_TSU_DS      14541 non-null   object  
 1   INT_NU_LATITUDE 14541 non-null   object  
 2   INT_NU_LONGITUDE 14541 non-null   object  
 3   ING_NU_IBGEMUNICIPIO 14541 non-null   int64  
 4   ING_NM_MUNICIPIO 14541 non-null   object  
 5   OUT_DT_OUTORGAFINAL 14540 non-null   object  
 6   OUT_DT_OUTORGAINICIAL 14540 non-null   object  
 7   INT_QT_VOLUMEANUAL 14493 non-null   float64 
 8   FIN_TFN_DS       14539 non-null   object  
 9   FES_NU_AREATOTALTANQUE 57 non-null   float64 
10  SIR_NU_AREAIRRIGADA 3919 non-null   float64 
11  FIA_NU_POPULACAOATENDIDA 863 non-null   float64 
12  CTE_TCA_DS       197 non-null    object  
13  CTE_NU_CABECAS   197 non-null    float64 
dtypes: float64(5), int64(1), object(8)
memory usage: 1.7+ MB
```

Iniciando a manipulação de dados.

Identificando a presença de valores preenchidos com NaN (not a number).

```
direito_de_uso.isnull().sum()

INT_TSU_DS           0
INT_NU_LATITUDE      0
INT_NU_LONGITUDE     0
ING_NU_IBGEMUNICIPIO 0
ING_NM_MUNICIPIO     0
OUT_DT_OUTORGAFINAL 1
OUT_DT_OUTORGAINICIAL 1
INT_QT_VOLUMEANUAL   48
FIN_TFN_DS           2
FES_NU_AREATOTALTANQUE 14484
SIR_NU_AREAIRRIGADA   10622
FIA_NU_POPULACAOATENDIDA 13678
CTE_TCA_DS           14344
CTE_NU_CABECAS        14344
dtype: int64
```

direito_de_uso.head(15)							
NT_QT_VOLUMEANUAL	FIN_TFN_DS	FES_NU_AREATOTALTANQUE	SIR_NU_AREAIRRIGADA	FIA_NU_POPULACAOATENDIDA	CTE_TCA_DS	CTE_NU_CABECAS	
557760.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
0.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
16128.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
0.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
0.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
0.0	Indústria		NaN	NaN	NaN	NaN	NaN
0.0	Indústria		NaN	NaN	NaN	NaN	NaN
9600.0	Irrigação		NaN	NaN	NaN	NaN	NaN
0.0	Criação Animal		NaN	NaN	NaN	NaN	NaN
0.0	Criação Animal		NaN	NaN	NaN	NaN	NaN
139104.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
20160.0	Abastecimento Público		NaN	NaN	NaN	NaN	NaN
54000.0	Irrigação		NaN	NaN	NaN	NaN	NaN
85280.0	Irrigação		NaN	NaN	NaN	NaN	NaN
139104.0	Irrigação		NaN	NaN	NaN	NaN	NaN

Podem ser substituídos todos os valores NaN por 0.00, pois os campos de preenchimento obrigatório na base, referentes as colunas que permaneceram no DF não apresentam valores nessa condição.

```
#SUBSTITUINDO OS VALORES NAN DAS COLUNAS POR 0.00 (ZERO)
direito_de_uso.fillna(0.00,inplace=True)
```

```
#CONFERINDO SE AINDA HÁ PRESENÇA DE VALORES PREENCHIDOS COM NaN (Not a Number)
direito_de_uso.isnull().sum()
```

```
INT_TSU_DS          0  
INT_NU_LATITUDE    0  
INT_NU_LONGITUDE   0  
ING_NU_IBGEMUNICIPIO 0  
ING_NM_MUNICIPIO   0  
OUT_DT_OUTORGAFINAL 0  
OUT_DT_OUTORGAINICIAL 0  
INT_TQ_VOLUMEANUAL 0  
FIN_TFN_DS         0  
FES_NU_AREATOTALTANQUE 0  
SIR_NU_AREARRIGADA 0  
FIA_NU_POPULACAOATENDIDA 0  
CTE_TCA_DS         0  
CTE_NU_CABECAS    0  
dtype: int64
```

```
direito_de_uso.head()
```

IT_QT_VOLUMENACIONAL	FIN_TFN_DS	FES_NU_AREATOTALTANQUE	SIR_NU_AREAIRRIGADA	FIA_NU_POPULACAOATENDIDA	CTE_TCA_DS	CTE_NU_CABECAS
557760.0	Abastecimento Público	0.0	0.0	0.0	0.0	0.0
0.0	Abastecimento Público	0.0	0.0	0.0	0.0	0.0
16128.0	Abastecimento Público	0.0	0.0	0.0	0.0	0.0
0.0	Abastecimento Público	0.0	0.0	0.0	0.0	0.0
0.0	Abastecimento Público	0.0	0.0	0.0	0.0	0.0

Reducindo o consumo de memória com alteração do tipo de coluna para ‘category’

```
#VERIFICANDO A PRESENÇA DE CATEGORIAS NAS COLUNAS  
direito_de_uso["INT TSU DS"].value_counts()
```

```
Subterrânea    8479  
Superficial    6062  
Name: INT TSU DS, dtype: int64
```

```
#VERIFICANDO A PRESENÇA DE CATEGORIAS NAS COLUNAS  
direito de uso! "FIN TEN DS"].value_counts()
```

Irrigação	5083
Abastecimento Público	3402
Outras	2396
Indústria	1834
Criação Animal	1156
Aquicultura em Tanque Escavado	533
Mineração - Extração de AreiaCascalho em Leito de Rio	69
Aquicultura em Tanque Rede	54
Consumo Humano	5
0.0	2
Serviços	2
115077.2	1
70445	1
253440	1
Mineração - Outros Processos Extrativos	1
251850	1

```
#rEDUZINDO O CONSUMO DE MEMÓRIA COM ALTERAÇÃO PARA DO TIPO DA COLUNA PARA 'category'
direito_de_uso.INT_TSU_DS = direito_de_uso.INT_TSU_DS.astype('category')

direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   INT_TSU_DS       14541 non-null   category
 1   INT_NU_LATITUDE  14541 non-null   object  
 2   INT_NU_LONGITUDE 14541 non-null   object  
 3   ING_NU_IBGEMUNICIPIO 14541 non-null   int64  
 4   ING_NM_MUNICIPIO  14541 non-null   object  
 5   OUT_DT_OUTORGAFINAL 14541 non-null   object  
 6   OUT_DT_OUTORGAINICIAL 14541 non-null   object  
 7   INT_QT_VOLUMEANUAL  14541 non-null   float64 
 8   FIN_TFN_DS        14541 non-null   object  
 9   FES_NU_AREATOTALTANQUE 14541 non-null   float64 
 10  SIR_NU_AREAIRRIGADA 14541 non-null   float64 
 11  FIA_NU_POPULACAOATENDIDA 14541 non-null   float64 
 12  CTE_TCA_DS        14541 non-null   object  
 13  CTE_NU_CABECAS   14541 non-null   float64 
dtypes: category(1), float64(5), int64(1), object(7)
memory usage: 1.6+ MB
```

Mais ajustes e tentativas noutra versão.

Selecionando as colunas referentes a datas.

```
#SELECCIONANDO AS COLUNAS CONTENDO DATAS
#ESSAS COLUNAS SÃO IDENTIFICADAS COM 'DT_' OU 'DATA_' NA NOMENCLATURA
data_filtro = lambda x: 'DATA_' in x or 'DT_' in x # Define a função de filtro
data = filter(data_filtro, lista_col_texto.split()) # Filtra
lista_datas = list(data) # Cria resultado em lista

print(lista_datas)

['INT_DT_REGISTRO', 'OUT_DT_OUTORGAFINAL', 'OUT_DT_OUTORGAINICIAL', 'IUS_DT_FINALCONCESSAO', 'ASB_DT_INSTALACAO', 'AMA_DT_COLETA', 'AMA_DT_ANALISE', 'DATA_EXTRACAO']
```

Verificando a presença de colunas tipo data.

```
#VERIFICANDO A PRESENÇA DE COLUNAS TIPO DATAS
col_datas=direito_de_uso_original.select_dtypes(include="datetime")
list(col_datas.columns.values)

[]
```

Gerando o DataFrame de trabalho já atribuindo tipo data as colunas selecionadas, não houve alteração na memória utilizada (25,9 MB). Verificando a presença de colunas tipo data no DataFrame de trabalho.

```
#VERIFICANDO A PRESENÇA DE COLUNAS TIPO DATAS
col_datas=direito_de_uso.select_dtypes(include="datetime")
lista_datas_alteradas=list(col_datas.columns.values)

print(lista_datas_alteradas)

['INT_DT_REGISTRO', 'OUT_DT_OUTORGAINICIAL', 'IUS_DT_FINALCONCESSAO', 'DATA_EXTRACAO']
```

Verificando se restam colunas a alterar para tipo data.

```
#VERIFICANDO SE TODAS AS COLUNAS DA LISTA ESTÃO COMO DATA
col_data_inalteradas=list(set(lista_datas).difference(lista_datas_alteradas))
print(col_data_inalteradas)

[ASB_DT_INSTALACAO', 'OUT_DT_OUTORGAFINAL', 'AMA_DT_COLETA', 'AMA_DT_ANALISE']
```

Separando as colunas que contêm os dados pessoais e de contato dos usuários para removê-los.

```
#RETORNANDO UMA LISTA COM OS NOMES DE TODAS AS COLUNAS
direito_de_uso.columns.values.tolist()
```

```
['INT_CD_CNARH40',
 'INT_CD_REGLA',
 'INT_TIN_DS',
 'INT_TIN_CD',
 'INT_TSU_DS',
 'INT_TSU_CD',
 'INT_TCH_DS',
 'INT_TSI_DS',
 'INT_TSI_CD',
 'INT_TOD_DS',
 'INT_TDM_DS',
 'INT_NU_CNARH',
 'INT_NU_SIAGAS',
 'INT_NU_LATITUDE',
 'INT_NU_LONGITUDE',
 'ING_NU_IBGEMUNICIPIO',
 'ING_SG_UFMUNICIPIO',
 'ING_NM_MUNICIPIO',
 'ING_NM_CORPOHIDRICO',
 'INT_NM_CORPOHIDRICOALTERADO',
 'INT_DS_ORGAO',
 'INT_DT_REGISTRO',
 'INT_CD_DECLARACAO',
 'INT_CD_ORIGEM',
 'INT_DS_OPCIONAL',
 'EMP_NM_EMPREENDIMENTO',
 'EMP_NM_USUARIO',
 'EMP_NU_CPFNPJ',
 'EMP_DS_EMAILRESPONSABEL',
 'EMP_NU_CEPENDERECHO',
 'EMP_CD_IBGEMUNICORRESPONDENCIA',
 'EMP_DS_LOGRADOURO',
 'EMP_DS_COMPLEMENTOENDERECO',
 'EMP_NU_LOGRADOURO',
 'EMP_DS_BAIRRO',
 'EMP_NU_DDD',
 'EMP_NU_TELEFONE',
 'EMP_SG_UF',
 'EMP_NM_MUNICIPIO',
 'OUT_TP_OUTORGAFINAL',
 'OUT_TS_CD',
 'OUT_DT_OUTORGAFINAL',
 'OUT_DT_OUTORGAINICIAL',
 'OUT_NU_PROCESSO',
 'OUT_TP_ATO',
 'OUT_NU_ATO',
 'DAD_QT_VAZAODIAJAN',
 'DAD_QT_VAZAODIAFEV',
 'DAD_QT_VAZAODIAMAR',
 'DAD_QT_VAZAODIA',
 'DAD_QT_VAZAODIAJAI',
 'DAD_QT_VAZAODIAJUN',
 'DAD_QT_VAZAODIAIGO',
 'DAD_QT_VAZAODIASET',
 'DAD_QT_VAZAODIAOUT',
 'DAD_QT_VAZAODIANOV',
 'DAD_QT_VAZAODIADEZ',
 'DAD_QT_HORASJAN',
 'DAD_QT_HORASFEV',
 'DAD_QT_HORASABR',
 'DAD_QT_HORASMAI',
 'DAD_QT_HORASJUN',
 'DAD_QT_HORASAGO',
 'DAD_QT_HORASSET',
 'DAD_QT_HORASOUT',
 'DAD_QT_HORASN',
 'DAD_QT_HORASDEZ',
 'DAD_QT_DIAJAN',
 'DAD_QT_DIAFEV',
 'DAD_QT_DIAMAR',
 'DAD_QT_DIAABR',
 'DAD_QT_DIAIMAI',
 'DAD_QT_DIAJU',
 'DAD_QT_DIAJUL',
 'DAD_QT_DIAAGO',
 'DAD_QT_DIASET',
 'DAD_QT_DIAOUT',
 'DAD_QT_DIANOV',
 'DAD_QT_DIADEZ',
 'INT_QT_VAZAOMAXIMA',
 'INT_QT_VAZAOMEDIA',
 'INT_QT_VOLUMEAURAL',
 'FIN_TFN_DS',
 'FIN_TFN_CD',
 'FES_NU_PROFUNDIDADEMIDATANQUE',
 'FES_NU_AREATOTALTANQUE',
 'TTC_TCU_DS',
 'TTC_TCU_CD',
 'FSE_TES_DS',
 'FSE_TES_CD',
 'FIE_NU_POPULACAOFINALATENDIDA',
 'FIE_TPS_DS',
 'FIE_TPS_CD',
 'FAH_TA_CD',
 'FAH_NU_POTENCIAINSTALADA',
 'FAH_IC_APROVEITAMENTOAGUA',
 'FAH_NU_AREAINUNDADANA',
 'FAH_NU_VOLUMENA',
 'FPE_TPE_DS',
 'FPE_TPE_CD',
 'FPE_CNA_CD',
 'IUS_NU_ALTURARES',
 'IUS_NU_AREARESMAX',
 'IUS_NU_VOLUMERES',
 'IUS_NU_COEFICIENTE_RETRONO',
 'IUS_NM_ENTIDADECONCEDENTE',
 'IUS_NU_CONCESSAO',
 'IUS_DT_FINALCONCESSAO',
 'ETP_MPE_DS',
 'ETP_MPE_CD',
 'ETP_NU_QUANTIDADEDEMIXENSAL',
 'SIR_TSI_DS',
 'SIR_TSI_CD',
 'SIR_TCT_DS',
 'SIR_TCT_CD',
 'SIR_NU_AREIRRIGADA',
 'FIA_NU_POPULACAOATENDIDA',
 'FTE_NU_POTENCIAINSTALADA',
 'FTE_NU_PRODUCAOMIXENSALMEDIA',
 'FTE_TCO_DS',
 'FTE_TCO_CD',
 'FTE_TSR_CD',
 'PME_CD_MESOPERACAO',
 'PME_NU_PRODUCAOAMENSAL',
 'FEA_NU_PRODUCAOAMIXENSALAREA',
 'FEA_NU_PROPORCAOAGUAPOLPA',
 'FEA_PC_TEORUMIDADE',
 'FOH_TOH_DS',
 'FOH_TOH_CD',
 'FRE_NU_AREAINUNDADANA',
 'FRE_NU_VOLUMENA',
 'FOU_TOU_DS',
 'FOU_TOU_CD',
 'HTE_NU_QUANTIDADE',
 'TUC_TEC_DS',
 'TUC_TEC_CD',
 'TUC_DS',
 'TUC_CD',
 'FTR_AR_TOTALEMPREENDIMENTO',
 'FTR_NU_PROCESSOIPA',
 'ESC_NU_PRODUCAOPRETENDIDA',
 'ESC_TET_DS',
 'ESC_TET_CD',
 'CTE_TSC_DS',
 'CTE_TSC_CD',
 'CTE_TCA_DS',
 'CTE_NU_CABECAS',
 'EFL_NU_DBOBRUTO',
 'EFL_NU_DBOTRATADO',
 'EFL_NU_FOSFOROBRUTO',
 'EFL_NU_NITROGENIOTRATADO',
 'EFL_NU_TEMPERATURA',
 'EFL_TTE_DS',
 'EFL_TTE_CD',
 'ITC_TUM_DS',
 'ITC_TUM_CD',
 'ITC_NU_PRODUCAOANUAL',
 'CNA_DS',
 'CNA_CD_CNAE',
 'ASB_DT_INSTALACAO',
 'ASB_TNP_DS',
 'ASB_TNP_CD',
 'ASB_NU_DIAMETROPURIFACAO',
 'ASB_TPN_DS',
 'ASB_TCA_DS',
 'ASB_TCA_CD',
 'ASB_NU_PROFUNDADEFINAL',
 'ASB_NU_ALTURABOCATUBO',
 'ASB_NU_COTATERRENO',
 'TST_DT',
 'TST_TTB_DS',
 'TST_TTB_CD',
 'TST_DS_TEMPODURACAO',
 'TST_NU_ND',
 'TST_NU_NE',
 'TST_VZ_ESTABILIZACAO',
 'TST_TMI_DS',
 'TST_TMI_CD',
 'TST_NU_COEFICIENTEARMAZENAMENTO',
 'TST_NU_TRANSMISSIVIDADE',
 'TST_NU_CONDUITIVIDADEHIDRAULICA',
 'TST_NU_PERMEABILIDADE',
 'AMA_DT_COLETA',
 'AMA_DT_ANALISE',
 'AMA_NU_CONDUITIVIDADEELETTRICA',
 'AMA_QT_TEMPERATURA',
 'AMA_QT_STD',
 'AMA_QT_PH',
 'AMA_QT_COLIFORMESFECAIS',
 'AMA_QT_BICARBONATO',
 'AMA_QT_CALCIO',
 'AMA_QT_CARBONATO',
 'AMA_QT_CLORETO',
 'AMA_QT_DUREZATOTAL',
 'AMA_QT_FERROTOTAL',
 'AMA_QT_FLUORETOS',
 'AMA_QT_NITRATOS',
 'AMA_QT_NITRITOS',
 'AMA_QT_POTASSIO',
 'AMA_QT_SODIO',
 'AMA_QT_SULFATO',
 'AMA_QT_MAGNESIO',
 'DATA_EXTRACAO',
 'ING_CD_OTTOBACIA_TRECHO',
 'ING_CD_COMITEFEDERAL',
 'ING_NM_COMITEFEDERAL',
 'ING_CD_COMITEESTADUAL',
 'ING_NM_COMITEESTADUAL',
 'ING_CS_CONAMA',
 'OTO_NM_OUTROUSO',
 'OTO_DS_OUTROUSO']
```

```
#CRIANDO UMA LISTA COM OS NOMES DAS COLUNAS
colunas_direito_de_uso=direito_de_uso.columns.values.tolist()
```

```
print(colunas_direito_de_uso)
```

```
['INT_CD_CNARH40', 'INT_CD_REGLA', 'INT_TIN_DS', 'INT_TSU_DS', 'INT_TSU_CD', 'INT_TCH_DS', 'INT_TSI_DS', 'INT_TSI_CD', 'INT_TOD_DS', 'INT_TDM_DS', 'INT_NU_CNARH', 'INT_NU_SIAGAS', 'INT_NU_LATITUDE', 'INT_NU_LONGITUDE', 'ING_NU_IBGEMUNICIPIO', 'ING_SG_UFMUNICIPIO', 'ING_NM_MUNICIPIO', 'ING_NM_CORPOHIDRICO', 'INT_NM_CORPOHIDRICOALTERADO', 'INT_DS_ORGAO', 'INT_DT_REGISTRO', 'INT_CD_DECLARACAO', 'INT_CD_ORIGEM', 'INT_DS_OPCIONAL', 'EMP_NM_EMPREENDIMENTO', 'EMP_NM_USUARIO', 'EMP_NU_CPFNPJ', 'EMP_DS_EMAILRESPONSABEL', 'EMP_NU_CEPENDERECHO', 'EMP_CD_IBGEMUNICORRESPONDENCIA', 'EMP_DS_LOGRADOURO', 'EMP_DS_COMPLEMENTOENDERECO', 'EMP_NU_LOGRADOURO', 'EMP_NU_CAIAXPOSTAL', 'EMP_DS_BAIRRO', 'EMP_NU_DDD', 'EMP_NU_TELEFONE', 'EMP_SG_UF', 'EMP_NM_MUNICIPIO', 'OUT_TP_OUTORGAFINAL', 'OUT_TS_CD', 'OUT_DT_OUTORGAFINAL', 'OUT_DT_OUTORGAINICIAL', 'OUT_NU_PROCESSO', 'OUT_TP_ATO', 'OUT_NU_ATO', 'DAD_QT_VAZAODIAJAN', 'DAD_QT_VAZAODIAFEV', 'DAD_QT_VAZAODIAMAR', 'DAD_QT_VAZAODIA', 'DAD_QT_VAZAODIAJAI', 'DAD_QT_VAZAODIAJUN', 'DAD_QT_VAZAODIAIGO', 'DAD_QT_VAZAODIASET', 'DAD_QT_VAZAODIAOUT', 'DAD_QT_VAZAODIANOV', 'DAD_QT_VAZAODIADEZ', 'DAD_QT_HORASJAN', 'DAD_QT_HORASFEV', 'DAD_QT_HORASABR', 'DAD_QT_HORASMAI', 'DAD_QT_HORASJUN', 'DAD_QT_HORASAGO', 'DAD_QT_HORASSET', 'DAD_QT_HORASOUT', 'DAD_QT_HORASN', 'DAD_QT_HORASDEZ', 'DAD_QT_DIAJAN', 'DAD_QT_DIAFEV', 'DAD_QT_DIAMAR', 'DAD_QT_DIAABR', 'DAD_QT_DIAIMAI', 'DAD_QT_DIAJU', 'DAD_QT_DIAJUL', 'DAD_QT_DIAAGO', 'DAD_QT_DIASET', 'DAD_QT_DIAOUT', 'DAD_QT_DIANOV', 'DAD_QT_DIADEZ', 'INT_QT_VAZAOMAXIMA', 'INT_QT_VAZAOMEDIA', 'INT_QT_VOLUMEAURAL', 'FIN_TFN_DS', 'FIN_TFN_CD', 'FES_NU_PROFUNDADEMIDATANQUE', 'FES_NU_AREATOTALTANQUE', 'TTC_TCU_DS', 'TTC_TCU_CD', 'FSE_TES_DS', 'FSE_TES_CD', 'FIE_NU_POPULACAOFINALATENDIDA', 'FIE_TPS_DS', 'FIE_TPS_CD', 'FAH_TA_CD', 'FAH_NU_POTENCIAINSTALADA', 'FAH_IC_APROVEITAMENTOAGUA', 'FAH_NU_AREAINUNDADANA', 'FAH_NU_VOLUMENA', 'FPE_TPE_DS', 'FPE_TPE_CD', 'FPE_CNA_CD', 'IUS_NU_ALTURARES', 'IUS_NU_AREARESMAX', 'IUS_NU_VOLUMERES', 'IUS_NU_COEFICIENTE_RETRONO', 'IUS_NM_ENTIDADECONCEDENTE', 'IUS_NU_CONCESSAO', 'IUS_DT_FINALCONCESSAO', 'ETP_MPE_DS', 'ETP_MPE_CD', 'ETP_NU_QUANTIDADEDEMIXENSAL', 'SIR_TSI_DS', 'SIR_TSI_CD', 'SIR_TCT_DS', 'SIR_TCT_CD', 'SIR_NU_AREIRRIGADA', 'FIA_NU_POPULACAOATENDIDA', 'FTE_NU_POTENCIAINSTALADA', 'FTE_NU_PRODUCAOMIXENSALMEDIA', 'FTE_TCO_DS', 'FTE_TCO_CD', 'FTE_TSR_CD', 'PME_CD_MESOPERACAO', 'PME_NU_PRODUCAOAMENSAL', 'FEA_NU_PRODUCAOAMIXENSALAREA', 'FEA_NU_PROPORCAOAGUAPOLPA', 'FEA_PC_TEORUMIDADE', 'FOH_TOH_DS', 'FOH_TOH_CD', 'FRE_NU_AREAINUNDADANA', 'FRE_NU_VOLUMENA', 'FOU_TOU_DS', 'FOU_TOU_CD', 'HTE_NU_QUANTIDADE', 'TUC_TEC_DS', 'TUC_TEC_CD', 'TUC_DS', 'TUC_CD', 'FTR_AR_TOTALEMPREENDIMENTO', 'FTR_NU_PROCESSOIPA', 'ESC_NU_PRODUCAOPRETENDIDA', 'ESC_TET_DS', 'ESC_TET_CD', 'CTE_TSC_DS', 'CTE_TSC_CD', 'CTE_TCA_DS', 'CTE_NU_CABECAS', 'EFL_NU_DBOBRUTO', 'EFL_NU_DBOTRATADO', 'EFL_NU_FOSFOROBRUTO', 'EFL_NU_NITROGENIOTRATADO', 'EFL_NU_TEMPERATURA', 'EFL_TTE_DS', 'EFL_TTE_CD', 'ITC_TUM_DS', 'ITC_TUM_CD', 'ITC_NU_PRODUCAOANUAL', 'CNA_DS', 'CNA_CD_CNAE', 'ASB_DT_INSTALACAO', 'ASB_TNP_DS', 'ASB_TNP_CD', 'ASB_NU_DIAMETROPURIFACAO', 'ASB_TPN_DS', 'ASB_TCA_DS', 'ASB_TCA_CD', 'ASB_NU_PROFUNDADEFINAL', 'ASB_NU_ALTURABOCATUBO', 'ASB_NU_COTATERRENO', 'TST_DT', 'TST_TTB_DS', 'TST_TTB_CD', 'TST_DS_TEMPODURACAO', 'TST_NU_ND', 'TST_NU_NE', 'TST_VZ_ESTABILIZACAO', 'TST_TMI_DS', 'TST_TMI_CD', 'TST_NU_COEFICIENTEARMAZENAMENTO', 'TST_NU_TRANSMISSIVIDADE', 'TST_NU_CONDUITIVIDADEHIDRAULICA', 'TST_NU_PERMEABILIDADE', 'AMA_DT_COLETA', 'AMA_DT_ANALISE', 'AMA_NU_CONDUITIVIDADEELETTRICA', 'AMA_QT_TEMPERATURA', 'AMA_QT_STD', 'AMA_QT_PH', 'AMA_QT_COLIFORMESFECAIS', 'AMA_QT_BICARBONATO', 'AMA_QT_CALCIO', 'AMA_QT_CARBONATO', 'AMA_QT_CLORETO', 'AMA_QT_DUREZATOTAL', 'AMA_QT_FERROTOTAL', 'AMA_QT_FLUORETOS', 'AMA_QT_NITRATOS', 'AMA_QT_NITRITOS', 'AMA_QT_POTASSIO', 'AMA_QT_SODIO', 'AMA_QT_SULFATO', 'AMA_QT_MAGNESIO', 'DATA_EXTRACAO', 'ING_CD_OTTOBACIA_TRECHO', 'ING_CD_COMITEFEDERAL', 'ING_NM_COMITEFEDERAL', 'ING_CD_COMITEESTADUAL', 'ING_NM_COMITEESTADUAL', 'ING_CS_CONAMA', 'OTO_NM_OUTROUSO', 'OTO_DS_OUTROUSO']
```

```
#TRANSFORMANDO A LISTA DE COLUNAS DO DATASET EM STRING |
lista_col_texto=str(colunas_direito_de_uso)
```

```
: #FILTRANDO AS COLUNAS COM OS DADOS DOS USUÁRIO DA LISTA DE COLUNAS DO DATASET, ELAS INICIAM COM 'EMP_'
f_filtro = lambda x: 'EMP_' in x # Define a função de filtro
dados_usuarios = filter(f_filtro, lista_col_texto.split()) # Filtra
lista_col_dados_usuarios = list(dados_usuarios) # Cria resultado em lista|
```

```
: print(lista_col_dados_usuarios)
```

```
["'EMP_NM_EMPREENDIMENTO'", "'EMP_NM_USUARIO'", "'EMP_NU_CPFNPJ'", "'EMP_DS_EMAILRESPONSABEL'", "'EMP_NU_CEPENDERECHO'", "'EMP_CD_IBGEMUNICORRESPONDENCIA'", "'EMP_DS_LOGRADOURO'", "'EMP_DS_COMPLEMENTOENDERECO'", "'EMP_NU_LOGRADOURO'", "'EMP_NU_CAIAXPOSTAL'", "'EMP_DS_BAIRRO'", "'EMP_NU_DDD'", "'EMP_NU_TELEFONE'", "'EMP_SG_UF'", "'EMP_NM_MUNICIPIO']
```

```
#TRANSFORMANDO EM STRING PARA MANIPULAR
tx_lista_col_dados_usuarios=str(lista_col_dados_usuarios)
```

```
#ELIMINANDO AS VÍRGULAS 1
tx_lista_col_dados_usuarios_limpo= tx_lista_col_dados_usuarios.replace(",","")

print(tx_lista_col_dados_usuarios_limpo)

["'EMP_NM_EMPREENDIMENTO'" "'EMP_NM_USUARIO'" "'EMP_NU_CPFNPJ'" "'EMP_DS_EMAILRESPONSAVEL'" "'EMP_NU_CEPENDERECHO'" "'EMP_CD_IBGEMUNCORRESPONDENCIA'" "'EMP_DS_LOGRADOURO'" "'EMP_DS_COMPLEMENTOENDERECO'" "'EMP_NU_LOGRADOURO'" "'EMP_NU_CAIXAPOSTAL'" "'EMP_DS_BAIRRO'" "'EMP_NU_DDD'" "'EMP_NU_TELEFONE'" "'EMP_SG_UF'" "'EMP_NM_MUNICIPIO'"
```

```
#ELIMINANDO OS COLCHETES 2
tx_lista_col_dados_usuarios_limpo= tx_lista_col_dados_usuarios_limpo.replace("[", "")
```

```
tx_lista_col_dados_usuarios_limpo= tx_lista_col_dados_usuarios_limpo.replace("]", "")
```

```
print(tx_lista_col_dados_usuarios_limpo)

["'EMP_NM_EMPREENDIMENTO'" "'EMP_NM_USUARIO'" "'EMP_NU_CPFNPJ'" "'EMP_DS_EMAILRESPONSAVEL'" "'EMP_NU_CEPENDERECHO'" "'EMP_CD_IBGEMUNCORRESPONDENCIA'" "'EMP_DS_LOGRADOURO'" "'EMP_DS_COMPLEMENTOENDERECO'" "'EMP_NU_LOGRADOURO'" "'EMP_NU_CAIXAPOSTAL'" "'EMP_DS_BAIRRO'" "'EMP_NU_DDD'" "'EMP_NU_TELEFONE'" "'EMP_SG_UF'" "'EMP_NM_MUNICIPIO'"
```

```
: #ELIMINANDO OS APÓSTROFOS 3
tx_lista_col_dados_usuarios_limpo= tx_lista_col_dados_usuarios_limpo.replace("'", "")
```

```
: print(tx_lista_col_dados_usuarios_limpo)

"EMP_NM_EMPREENDIMENTO" "EMP_NM_USUARIO" "EMP_NU_CPFNPJ" "EMP_DS_EMAILRESPONSAVEL" "EMP_NU_CEPENDERECHO" "EMP_CD_IBGEMUNCORRESPONDENCIA" "EMP_DS_LOGRADOURO" "EMP_DS_COMPLEMENTOENDERECO" "EMP_NU_LOGRADOURO" "EMP_NU_CAIXAPOSTAL" "EMP_DS_BAIRRO" "EMP_NU_DDD" "EMP_NU_TELEFONE" "EMP_SG_UF" "EMP_NM_MUNICIPIO"
```

```
#ELIMINANDO AS ASPAS, DEIXANDO ESPAÇO 4
tx_lista_col_dados_usuarios_limpo= tx_lista_col_dados_usuarios_limpo.replace("'", ' ')
```

```
print(tx_lista_col_dados_usuarios_limpo)

EMP_NM_EMPREENDIMENTO EMP_NM_USUARIO EMP_NU_CPFNPJ EMP_DS_EMAILRESPONSAVEL EMP_NU_CEPENDERECHO EMP_CD_IBGEMUNCORRESPONDENCIA EMP_DS_LOGRADOURO EMP_DS_COMPLEMENTOENDERECO EMP_NU_LOGRADOURO EMP_NU_CAIXAPOSTAL EMP_DS_BAIRRO EMP_NU_DDD EMP_NU_TELEFONE EMP_SG_UF EMP_NM_MUNICIPIO
```

```
#GERANDO A LISTA DE COLUNAS DADOS SOBRE USUÁRIOS PARA REMOÇÃO
list_tx_lista_col_dados_usuarios_limpo= tx_lista_col_dados_usuarios_limpo.split()
```

```
list_tx_lista_col_dados_usuarios_limpo

['EMP_NM_EMPREENDIMENTO',
 'EMP_NM_USUARIO',
 'EMP_NU_CPFNPJ',
 'EMP_DS_EMAILRESPONSAVEL',
 'EMP_NU_CEPENDERECHO',
 'EMP_CD_IBGEMUNCORRESPONDENCIA',
 'EMP_DS_LOGRADOURO',
 'EMP_DS_COMPLEMENTOENDERECO',
 'EMP_NU_LOGRADOURO',
 'EMP_NU_CAIXAPOSTAL',
 'EMP_DS_BAIRRO',
 'EMP_NU_DDD',
 'EMP_NU_TELEFONE',
 'EMP_SG_UF',
 'EMP_NM_MUNICIPIO']
```

```
#REMOVER COLUNAS COM INFORMAÇÕES PESSOAIS E DE CONTATO E REDEFINIR
direito_de_uso = direito_de_uso.drop(columns=list_tx_lista_col_dados_usuarios_limpo)

direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 213 entries, INT_CD_CNARH40 to OTO_DS_OUTROUSO
dtypes: datetime64[ns](4), float64(153), int64(5), object(51)
memory usage: 24.2+ MB
```

As colunas contendo dados pessoais e de contato dos usuários foram excluídas, mantendo-se as demais colunas, ou seja, 213 colunas no DataFrame ‘direito_de_uso’, que foi redefinido.

Unificando colunas afins em que os elementos presentes não se sobreponem.

```
#REMOVER COLUNAS COM INFORMAÇÕES PESSOAIS E DE CONTATO E REDEFINIR
direito_de_uso_sem_usuario = direito_de_uso.drop(columns=list_tx_lista_col_dados_usuarios_limpo)

direito_de_uso_sem_usuario.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14901 entries, 0 to 14900
Columns: 213 entries, INT_CD_CNARH40 to OTO_DS_OUTROUSO
dtypes: datetime64[ns](4), float64(153), int64(5), object(51)
memory usage: 24.2+ MB
```

Depois de diversos ajustes, agora com um olhar inverso ao que vinha sendo realizado, pois a fartura de dados provocou uma tempestade de possibilidades que ofuscou a objetividade no reconhecimento de quais realmente eram mais relevantes para o objetivo principal, foi refeita essa etapa selecionando apenas colunas de interesse e descartando as demais.

```
direito_de_uso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14541 entries, 0 to 14900
Data columns (total 9 columns):
 #   Column           Non-Null Count Dtype
 ---  -- 
 0   INT_TSU_DS       14541 non-null  category
 1   INT_NU_LATITUDE  14541 non-null  object
 2   INT_NU_LONGITUDE 14541 non-null  object
 3   ING_NU_IBGEMUNICIPIO 14541 non-null  int64
 4   ING_NM_MUNICIPIO  14541 non-null  object
 5   OUT_DT_OUTORGAFINAL 14540 non-null  object
 6   OUT_DT_OUTORGAINICIAL 14540 non-null  object
 7   INT_QT_VOLUMEANUAL 14493 non-null  float64
 8   FIN_TFN_DS        14539 non-null  category
dtypes: category(2), float64(1), int64(1), object(5)
memory usage: 937.7+ KB
```

```

#IMPORTANDO AS BIBLIOTECAS
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import pickle
import pandas as pd
import datetime
import glob
import pydot
import graphviz
import os, sys
import sklearn

%matplotlib inline

from openpyxl import load_workbook
from unicodedata import normalize
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,export_graphviz
from sklearn.tree import DecisionTreeClassifier#for checking testing results
from sklearn.tree import plot_tree
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix#for visualizing tree
from sklearn import tree
from mlxtend.plotting import plot_decision_regions
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder#for train test splitting
from sklearn.model_selection import train_test_split#for decision tree object
from dtreeviz.trees import *

from ipywidgets import interactive
from IPython.display import SVG,display
from graphviz import Source

#CARREGANDO DATASET
direito_de_uso_original=pd.read_excel("exportacao_cnarh40_CE.xlsx")

#CRIANDO CÓPIA DO DF PRESERVANDO O ORIGINAL
direito_de_uso=direito_de_uso_original

#EXCLUINDO AS LINHAS SELECIONADAS
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['FIN_TFN_DS'].isnull()].index, inplace=True) # Finalidade de uso ausente
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['OUT_DT_OUTORGAFINAL'].isnull()].index, inplace=True) # Data final ausente
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['OUT_DT_OUTORGAINICIAL'].isnull()].index, inplace=True) # Data inicial ausente
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['INT_QT_VOLUMEANUAL'].isnull()].index, inplace=True) # Data inicial ausente
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['INT_QT_VOLUMEANUAL'] == 0].index, inplace=True) # Volume anual zero, pode
direito_de_uso.drop(direito_de_uso['INT_TIN_DS'] != 'Captação'].index, inplace=True)
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['OUT_TE_OUTORGAT'] != 'Direito de Uso'].index, inplace=True)
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['OUT_TP_SITUACAOOUTORG'] != 'Outorgado'].index, inplace=True)
direito_de_uso.drop(direito_de_uso.loc[direito_de_uso['INT_TSI_DS'] != 'Operação'].index, inplace=True)

#VERIFICAR A PRESENCA DE REGISTROS REPETIDOS NO DF DE TRABALHO
# verificar antes de excluir colunas para evitar exclusão de coincidentes
repetidos = direito_de_uso[direito_de_uso.duplicated()]

#EXCLUINDO OS REGISTROS REPETIDOS E DEIXANDO UM EXEMPLAR DOS REGISTRO DE CADA GRUPO DE CLONES
if (len(repetidos))>0:
    direito_de_uso.drop_duplicates(inplace=True)

# SIMPLIFICANDO OS CAMPOS DA COLINA COM MUNICÍPIOS
municipios_direito_de_uso = direito_de_uso['ING_NM_MUNICIPIO']
municipios_direito_de_uso = municipios_direito_de_uso.str.normalize('NFKD').str.encode('ascii', errors='ignore').str.decode('utf-8')
direito_de_uso.insert(0,'ING_NM_MUNICIPIO_clean', municipios_direito_de_uso)#Inserindo uma nova coluna na posição "1"

```



```

DF = direito_de_uso

LISTA_MUNICIPIO = []
LISTA_VOLUME = []
LISTA_VIGENCIA = []
LISTA_TIPO_MANANCIAL = []
LISTA_FINALIDADE_DE_USO = []
LISTA_LATITUDE = []
LISTA_LONGITUDE = []

for i in DF.index:
    ano_inicial = DF.loc[i,'ANO_INICIAL']
    ano_final = DF.loc[i,'ANO_FINAL']
    #passo_ano = 1
    vigencia = list(range(ano_inicial, ano_final,1))# Convertendo um intervalo em uma lista
    lista_municipio = []# Cria uma Lista vazia.
    lista_volume_anual = []# Cria uma Lista vazia.

    lista_tipo_manancial = []# Cria uma Lista vazia.
    lista_finalidade_de_uso = []# Cria uma Lista vazia.
    lista_latitude = []# Cria uma Lista vazia.
    lista_longitude = []# Cria uma Lista vazia.

    municipio = DF.loc[i,'MUNICIPIO']
    volume_anual = DF.loc[i,'VOLUME_ANUAL']

    tipo_manancial = DF.loc[i,'TIPO_MANANCIAL']
    finalidade_de_uso = DF.loc[i,'FINALIDADE_DE_USO']
    latitude = DF.loc[i,'LATITUDE']
    longitude = DF.loc[i,'LONGITUDE']

    anos_vigencia = vigencia
    for m in vigencia:
        ano = ano_inicial
        lista_municipio.append(municipio)
        lista_volume_anual.append(volume_anual)

        lista_tipo_manancial.append(tipo_manancial)
        lista_finalidade_de_uso.append(finalidade_de_uso)
        lista_latitude.append(latitude)
        lista_longitude.append(longitude)

    LISTA_VIGENCIA.extend(anos_vigencia)
    LISTA_MUNICIPIO.extend(lista_municipio)
    LISTA_VOLUME.extend(lista_volume_anual)

    LISTA_TIPO_MANANCIAL.extend(lista_tipo_manancial)
    LISTA_FINALIDADE_DE_USO.extend(lista_finalidade_de_uso)
    LISTA_LATITUDE.extend(lista_latitude)
    LISTA_LONGITUDE.extend(lista_longitude)

direito_de_uso_EXTENDIDA = pd.DataFrame(np.column_stack([LISTA_TIPO_MANANCIAL, LISTA_LATITUDE, LISTA_LONGITUDE, LISTA_MUNICIPIO,
columns=['TIPO_MANANCIAL', 'LATITUDE', 'LONGITUDE', 'MUNICIPIO', 'FINALIDADE_DE_USO', 'VOLUME_M3'])

# ALTERAÇÃO DO TIPO DA COLUNA
direito_de_uso_EXTENDIDA.TIPO_MANANCIAL = direito_de_uso_EXTENDIDA.TIPO_MANANCIAL.astype('category')
direito_de_uso_EXTENDIDA.FINALIDADE_DE_USO = direito_de_uso_EXTENDIDA.FINALIDADE_DE_USO.astype('category')
direito_de_uso_EXTENDIDA.VOLUME_M3 = direito_de_uso_EXTENDIDA.VOLUME_M3.astype('float32')
direito_de_uso_EXTENDIDA.ANOS_VIGENTES = direito_de_uso_EXTENDIDA.ANOS_VIGENTES.astype('int32')

# GUARDANDO UMA VERSÃO COM AS COORDENADAS
direito_de_uso_EXTENDIDA_coordenadas = direito_de_uso_EXTENDIDA

#RETIRAR COORDENADAS
direito_de_uso_EXTENDIDA = direito_de_uso_EXTENDIDA.drop(columns=['LATITUDE', 'LONGITUDE'])# Removendo as colunas

```

Agrupando os volumes de mesmos município, finalidade, ano vigente e tipo de manancial e somando-os.

```

# AGRUPANDO O VOLUME POR MUNICÍPIO, FINALIDADE , ANO E TIPO _MANANCIAL (SOMANDO)
soma_vol_ano_municipio = direito_de_uso_EXTENDIDA.groupby(['MUNICIPIO', 'FINALIDADE_DE_USO', 'ANOS_VIGENTES', 'TIPO_MANANCIAL'])

```

soma_vol_ano_municipio				
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	
ABAIARA	Abastecimento Humano	2005	Subterrânea	0.0
			Superficial	0.0
		2006	Subterrânea	0.0
			Superficial	0.0
VÁRZEA ALEGRE	Serviço e Comércio	2007	Subterrânea	0.0
			...	
		2028	Superficial	0.0
		2029	Subterrânea	0.0
		2030	Superficial	0.0
			Subterrânea	0.0
			Superficial	0.0

Name: VOLUME_M3, Length: 76544, dtype: float32

Convertendo a série obtida em dataframe

```

df_soma_vol_ano_municipio = pd.DataFrame([soma_vol_ano_municipio]) #Convertendo em DF

df_soma_vol_ano_municipio

```

MUNICIPIO	ABAJARA												VÁRZE	
FINALIDADE_DE_USO	Abastecimento Humano												Serviç	
ANOS_VIGENTES	2005	2006			2007			2008			2009			2026
TIPO_MANANCIAL	Subterrânea	Superficial	Subterrânea	Superficial	Subterrânea	Superficial	Subterrânea	Superficial	Subterrânea	Superficial	Subterrânea	Superficial	Subter	
VOLUME_M3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

1 rows x 76544 columns

Transpondo o dataframe obtido

VOLUME_M3				
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	
ABAIAARA	Abastecimento Humano	2005	Subterrânea	0.0
			Superficial	0.0
	2006		Subterrânea	0.0
			Superficial	0.0
	2007		Subterrânea	0.0
			Superficial	0.0
VÁRZEA ALEGRE	Serviço e Comércio
		2028	Superficial	0.0
		2029	Subterrânea	0.0
			Superficial	0.0
		2030	Subterrânea	0.0
			Superficial	0.0

Convertendo todos indexadores do segundo nível de indexação do dataframe em colunas.

```
df_soma_vol_ano_municipio_T = df_soma_vol_ano_municipio_T.reset_index(level=['MUNICIPIO', 'FINALIDADE_DE_USO', 'ANOS_VIGENTES', 'TIPO_MANANCIAL'])
```

df_soma_vol_ano_municipio_T					
	MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOLUME_M3
0	ABAIIARA	Abastecimento Humano	2005	Subterrânea	0.0
1	ABAIIARA	Abastecimento Humano	2005	Superficial	0.0
2	ABAIIARA	Abastecimento Humano	2006	Subterrânea	0.0
3	ABAIIARA	Abastecimento Humano	2006	Superficial	0.0
4	ABAIIARA	Abastecimento Humano	2007	Subterrânea	0.0
...
76539	VÁRZEA ALEGRE	Serviço e Comércio	2028	Superficial	0.0
76540	VÁRZEA ALEGRE	Serviço e Comércio	2029	Subterrânea	0.0
76541	VÁRZEA ALEGRE	Serviço e Comércio	2029	Superficial	0.0
76542	VÁRZEA ALEGRE	Serviço e Comércio	2030	Subterrânea	0.0
76543	VÁRZEA ALEGRE	Serviço e Comércio	2030	Superficial	0.0

Como há apenas dois tipos de manancial e os dados estão presentes em todos as classificações, podem ser convertidos em colunas.

Dividindo o Dataframe em dois usando como parâmetro as categorias do tipo de manancial que deverão tornar-se novas colunas de volume que é a única que é específica..

```
# DIVIDINDO O DF PELA COLUNA 'TIPO_MANANCIAL'
df_soma_vol_ano_municipio_T_superf = df_soma_vol_ano_municipio_T.loc[(df_soma_vol_ano_municipio_T['TIPO_MANANCIAL']) == 'Superficial']
df_soma_vol_ano_municipio_T_subter = df_soma_vol_ano_municipio_T.loc[(df_soma_vol_ano_municipio_T['TIPO_MANANCIAL']) == 'Subterrânea']
```

Conforme o esperado os Dataframes têm o mesmo número de linhas e colunas.

Dataframes gerados:

Tipo de manancial “Superficial”

df_soma_vol_ano_municipio_T_superf					
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOLUME_M3	
1	ABAIARA	Abastecimento Humano	2005	Superficial	0.0
3	ABAIARA	Abastecimento Humano	2006	Superficial	0.0
5	ABAIARA	Abastecimento Humano	2007	Superficial	0.0
7	ABAIARA	Abastecimento Humano	2008	Superficial	0.0
9	ABAIARA	Abastecimento Humano	2009	Superficial	0.0
...
76535	VÁRZEA ALEGRE	Serviço e Comércio	2026	Superficial	0.0
76537	VÁRZEA ALEGRE	Serviço e Comércio	2027	Superficial	0.0
76539	VÁRZEA ALEGRE	Serviço e Comércio	2028	Superficial	0.0
76541	VÁRZEA ALEGRE	Serviço e Comércio	2029	Superficial	0.0
76543	VÁRZEA ALEGRE	Serviço e Comércio	2030	Superficial	0.0

38272 rows × 5 columns

Tipo de manancial “Subterrânea”

df_soma_vol_ano_municipio_T_subter					
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOLUME_M3	
0	ABAIARA	Abastecimento Humano	2005	Subterrânea	0.0
2	ABAIARA	Abastecimento Humano	2006	Subterrânea	0.0
4	ABAIARA	Abastecimento Humano	2007	Subterrânea	0.0
6	ABAIARA	Abastecimento Humano	2008	Subterrânea	0.0
8	ABAIARA	Abastecimento Humano	2009	Subterrânea	0.0
...
76534	VÁRZEA ALEGRE	Serviço e Comércio	2026	Subterrânea	0.0
76536	VÁRZEA ALEGRE	Serviço e Comércio	2027	Subterrânea	0.0
76538	VÁRZEA ALEGRE	Serviço e Comércio	2028	Subterrânea	0.0
76540	VÁRZEA ALEGRE	Serviço e Comércio	2029	Subterrânea	0.0
76542	VÁRZEA ALEGRE	Serviço e Comércio	2030	Subterrânea	0.0

38272 rows × 5 columns

Renomeando as colunas 'Volume_m3'

```
#RENOMEAR COLUNAS 'VOLUME_M3' DOS DATAFRAMES 'SUPERFICIAL' E 'SUBTERRÂNEO'
df_soma_vol_ano_municipio_T_subter = df_soma_vol_ano_municipio_T_subter.rename(columns={'VOLUME_M3':'VOL_SUBTER_M3'})
df_soma_vol_ano_municipio_T_superf = df_soma_vol_ano_municipio_T_superf.rename(columns={'VOLUME_M3':'VOL_SUPERF_M3'})
```

Dataframes com colunas renomeadas:

df_soma_vol_ano_municipio_T_subter					
	MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOL_SUBTER_M3
0	ABAIARA	Abastecimento Humano	2005	Subterrânea	0.0
2	ABAIARA	Abastecimento Humano	2006	Subterrânea	0.0
4	ABAIARA	Abastecimento Humano	2007	Subterrânea	0.0
6	ABAIARA	Abastecimento Humano	2008	Subterrânea	0.0
8	ABAIARA	Abastecimento Humano	2009	Subterrânea	0.0
...
76534	VÁRZEA ALEGRE	Serviço e Comércio	2026	Subterrânea	0.0
76536	VÁRZEA ALEGRE	Serviço e Comércio	2027	Subterrânea	0.0
76538	VÁRZEA ALEGRE	Serviço e Comércio	2028	Subterrânea	0.0
76540	VÁRZEA ALEGRE	Serviço e Comércio	2029	Subterrânea	0.0
76542	VÁRZEA ALEGRE	Serviço e Comércio	2030	Subterrânea	0.0

38272 rows × 5 columns

df_soma_vol_ano_municipio_T_superf					
	MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOL_SUPERF_M3
1	ABAIARA	Abastecimento Humano	2005	Superficial	0.0
3	ABAIARA	Abastecimento Humano	2006	Superficial	0.0
5	ABAIARA	Abastecimento Humano	2007	Superficial	0.0
7	ABAIARA	Abastecimento Humano	2008	Superficial	0.0
9	ABAIARA	Abastecimento Humano	2009	Superficial	0.0
...
76535	VÁRZEA ALEGRE	Serviço e Comércio	2026	Superficial	0.0
76537	VÁRZEA ALEGRE	Serviço e Comércio	2027	Superficial	0.0
76539	VÁRZEA ALEGRE	Serviço e Comércio	2028	Superficial	0.0
76541	VÁRZEA ALEGRE	Serviço e Comércio	2029	Superficial	0.0
76543	VÁRZEA ALEGRE	Serviço e Comércio	2030	Superficial	0.0

38272 rows × 5 columns

Reunificando o Dataframe com as colunas renomeadas

```
# REUNIFICANDO O DF
LISTA_VOL_SUBTER_M3 = list(df_soma_vol_ano_municipio_T_subter['VOL_SUBTER_M3']) # Lista com os elementos que retornarão para o DF
DIREITO_DE_USO = df_soma_vol_ano_municipio_T_superf.assign(VOL_SUBTER_M3 = LISTA_VOL_SUBTER_M3)
```

DIREITO_DE_USO						
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOL_SUPERF_M3	VOL_SUBTER_M3	
1	ABAIARA	Abastecimento Humano	2005	Superficial	0.0	0.0
3	ABAIARA	Abastecimento Humano	2006	Superficial	0.0	0.0
5	ABAIARA	Abastecimento Humano	2007	Superficial	0.0	0.0
7	ABAIARA	Abastecimento Humano	2008	Superficial	0.0	0.0
9	ABAIARA	Abastecimento Humano	2009	Superficial	0.0	0.0
...
76535	VÁRZEA ALEGRE	Serviço e Comércio	2026	Superficial	0.0	0.0
76537	VÁRZEA ALEGRE	Serviço e Comércio	2027	Superficial	0.0	0.0
76539	VÁRZEA ALEGRE	Serviço e Comércio	2028	Superficial	0.0	0.0
76541	VÁRZEA ALEGRE	Serviço e Comércio	2029	Superficial	0.0	0.0
76543	VÁRZEA ALEGRE	Serviço e Comércio	2030	Superficial	0.0	0.0

38272 rows × 6 columns

Removendo coluna que perdeu o significado no Dataframe

#REMOVER COLUNAS SELECIONADAS					
DIREITO_DE_USO = DIREITO_DE_USO.drop(columns='TIPO_MANANCIAL')# Removendo as colunas					
DIREITO_DE_USO					
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	
1	ABAIARA	Abastecimento Humano	2005	0.0	0.0
3	ABAIARA	Abastecimento Humano	2006	0.0	0.0
5	ABAIARA	Abastecimento Humano	2007	0.0	0.0
7	ABAIARA	Abastecimento Humano	2008	0.0	0.0
9	ABAIARA	Abastecimento Humano	2009	0.0	0.0
...
76535	VÁRZEA ALEGRE	Serviço e Comércio	2026	0.0	0.0
76537	VÁRZEA ALEGRE	Serviço e Comércio	2027	0.0	0.0
76539	VÁRZEA ALEGRE	Serviço e Comércio	2028	0.0	0.0
76541	VÁRZEA ALEGRE	Serviço e Comércio	2029	0.0	0.0
76543	VÁRZEA ALEGRE	Serviço e Comércio	2030	0.0	0.0

38272 rows × 5 columns

Testando se as linhas da coluna 'VOL_SUBTER_M3' foram acopladas corretamente ao município, finalidade de uso e ano vigente correspondente.

###TESTANDO SE A NOVA COLUNA FOI ACOPLADA CORRETAMENTE					
DIREITO_DE_USO_teste = DIREITO_DE_USO.loc[76130 : 76150] # Selecionando trecho da tabela					
DIREITO_DE_USO_teste					
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	
76131	VÁRZEA ALEGRE	Abastecimento Humano	2006	934080.0	0.000000
76133	VÁRZEA ALEGRE	Abastecimento Humano	2007	974400.0	0.000000
76135	VÁRZEA ALEGRE	Abastecimento Humano	2008	974400.0	0.000000
76137	VÁRZEA ALEGRE	Abastecimento Humano	2009	974400.0	6482.399902
76139	VÁRZEA ALEGRE	Abastecimento Humano	2010	974400.0	11826.000000
76141	VÁRZEA ALEGRE	Abastecimento Humano	2011	974400.0	15858.000000
76143	VÁRZEA ALEGRE	Abastecimento Humano	2012	974400.0	24267.599609
76145	VÁRZEA ALEGRE	Abastecimento Humano	2013	974400.0	24267.599609
76147	VÁRZEA ALEGRE	Abastecimento Humano	2014	974400.0	24267.599609
76149	VÁRZEA ALEGRE	Abastecimento Humano	2015	974400.0	20235.599609

```
teste_2 = df_soma_vol_ano_municipio_T.query('FINALIDADE_DE_USO=="Abastecimento Humano" and MUNICIPIO == "VÁRZEA ALEGRE" and ANOS_VIGENTES > 2010')
```

MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOLUME_M3	
76146	VÁRZEA ALEGRE	Abastecimento Humano	2014	Subterrânea	24267.599609
76147	VÁRZEA ALEGRE	Abastecimento Humano	2014	Superficial	974400.000000

```
teste_1 = df_soma_vol_ano_municipio_T.query('FINALIDADE_DE_USO=="Abastecimento Humano" and MUNICIPIO == "VÁRZEA ALEGRE" and ANOS_VIGENTES < 2010')
```

MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOLUME_M3	
76136	VÁRZEA ALEGRE	Abastecimento Humano	2009	Subterrânea	6482.399902
76137	VÁRZEA ALEGRE	Abastecimento Humano	2009	Superficial	974400.000000

Inserindo coluna com a soma dos volumes superficial e subterrâneo

```
# INSERINDO COLUNA CO A SOMA DOS VOLUMES DE CADA LINHA
DIREITO_DE_USO['VOLUME_MUN_M3'] = DIREITO_DE_USO['VOL_SUPERF_M3'] + DIREITO_DE_USO['VOL_SUBTER_M3']
```

MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	TIPO_MANANCIAL	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3	
1	ABAIARA	Abastecimento Humano	2005	Superficial	0.0	0.0	0.0
3	ABAIARA	Abastecimento Humano	2006	Superficial	0.0	0.0	0.0
5	ABAIARA	Abastecimento Humano	2007	Superficial	0.0	0.0	0.0
7	ABAIARA	Abastecimento Humano	2008	Superficial	0.0	0.0	0.0
9	ABAIARA	Abastecimento Humano	2009	Superficial	0.0	0.0	0.0
...
76535	VÁRZEA ALEGRE	Serviço e Comércio	2026	Superficial	0.0	0.0	0.0
76537	VÁRZEA ALEGRE	Serviço e Comércio	2027	Superficial	0.0	0.0	0.0
76539	VÁRZEA ALEGRE	Serviço e Comércio	2028	Superficial	0.0	0.0	0.0
76541	VÁRZEA ALEGRE	Serviço e Comércio	2029	Superficial	0.0	0.0	0.0
76543	VÁRZEA ALEGRE	Serviço e Comércio	2030	Superficial	0.0	0.0	0.0

38272 rows x 7 columns

Para os trabalhos com *datasets* oriundos do IPECE, as colunas também foram renomeadas, pois estavam desconfiguradas e não transmitiam clareza, pois originalmente muitas células do cabeçalho estavam mescladas.

ANUÁRIO ESTATÍSTICO DO CEARÁ - 2017		Column2	Column3	Column4	Column5	Column6	Column7	Column8
AGROPECUÁRIA E EXTRAÇÃO VEGETAL								
18.5 PRODUTOS DE ORIGEM ANIMAL								
Tabela 18.5.1 Quantidade produzida e valor da produção								
Municípios	Vacas ordenhadas (cabeças)				Produção de leite			
					Quantidade produzida (mil l)			
		2013	2014	2015	2016	2013	2014	
Ceará	561.325	580949	548158	534479	455.452	498133	48	
Abaiara	1.619	1539	1495	1046	951	894		
Acarape	467	483	474	463	235	243		

ANUÁRIO ESTATÍSTICO DO CEARÁ - 2017																		
AGROPECUÁRIA E EXTRAÇÃO VEGETAL																		
18.5 PRODUTOS DE ORIGEM ANIMAL																		
Tabela 18.5.3 Quantidade produzida e valor da produção de ovos - Ceará - 2013-2015																		
Municípios			Produção de ovos						Produção de codorna									
			Ovos de galinha			Ovos de codorna			Ovos de galinha			Ovos de codorna						
			Quantidade produzida (mil dz)	Valor da produção (R\$ mil)			Quantidade produzida (mil dz)			Valor da produção (R\$ mil)								
			2013	2014	2015	2016	2013	2014	2015	2016	2013	2014	2015	2016	2013	2014	2015	2016
9	Ceará	135.129	136.802	144.122	161.557	479.608	565.401	636.278	682.422	1.507	1.597	16.641	13.168	1.721	2.064	16.523	13.628	
10	Abaiara	33	31	43	32	125	98	167	176	-	-	-	-	-	-	-	-	
11	Acarape	8	8	9	9	37	46	50	55	-	-	-	-	-	-	-	-	
12	Acarau	201	201	201	201	722	966	1.087	1.087	-	-	-	-	-	-	-	-	

Para cada planilha, uma planilha denominada “Tabela de campos IPECE” foi elaborada para elencar a equivalência entre a nomenclatura original e a mais adotada nesse testudo.

Em um primeiro momento, manualmente, as linhas do cabeçalho original foram excluídas e uma nova linha foi inserida como cabeçalho com a nomenclatura adotada. Também foi excluída a última linha que continha a fonte dos respectivos dados.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
Municípios	Ovos de galinha 2013 (mil dz)	Ovos de galinha 2014 (mil dz)	Ovos de galinha 2015 (mil dz)	Ovos de galinha 2016 (mil dz)	Ovos de galinha 2013 (R\$ mil)	Ovos de galinha 2014 (R\$ mil)	Ovos de galinha 2015 (R\$ mil)	Ovos de galinha 2016 (R\$ mil)	Ovos de codorna 2013 (mil dz)	Ovos de codorna 2014 (mil dz)	Ovos de codorna 2015 (mil dz)	Ovos de codorna 2016 (mil dz)	Ovos de codorna 2013 (R\$ mil)	Ovos de codorna 2014 (R\$ mil)	Ovos de codorna 2015 (R\$ mil)	Ovos de codorna 2016 (R\$ mil)	
2	Ceará	135.129	136.802	144.122	161.557	479.608	565.401	636.278	682.422	1.507	1.597	16.641	13.168	1.721	2.064	16.523	13.628
3	Abaiara	33	31	43	32	125	98	167	176	-	-	-	-	-	-	-	-
4	Acarape	8	8	9	9	37	46	50	55	-	-	-	-	-	-	-	-
5	Acarau	201	201	201	201	722	966	1.087	1.087	-	-	-	-	-	-	-	-
6	Acopiara	295	304	298	268	1.061	1.460	1.789	1.609	-	-	-	-	-	-	-	-

Cada *dataset* foi carregado no PowerBI e a tipologia de seus respectivos campos adequada conforme tabela, mas antes os seus excedentes foram removidos.

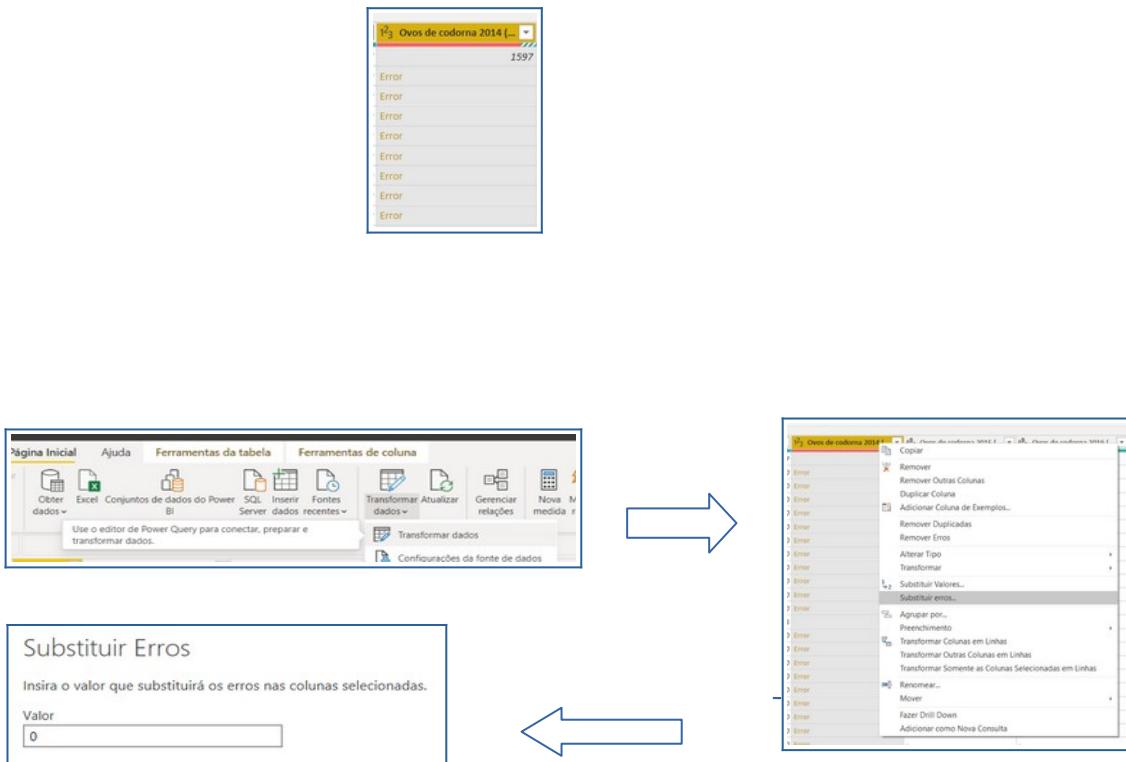
As colunas excedentes, ou seja, que não possuíam campos nas quais todas as linhas constavam “null” foram excluídas através do comando “Remover Colunas” .

Da mesma forma, as linhas excedentes, ou seja, que não possuíam registros e que todas as linhas constavam “null” foram excluídas através do comando “Manter Linhas”.

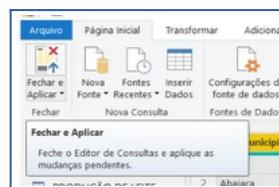
Ao concluir a transformação é preciso aplicar:

Em “Ferramentas da Coluna” foi adequado o tipo e o formato das de cada uma das colunas :

Dados das colunas referentes as codornas não puderam ser convertidos automaticamente, então foi necessário “Transformar dados” no editor Power Query.



Nas colunas “Ovos de codorna ... (mil dz)” e “Ovos de codorna (R\$ mil)” Os erros foram substituídos por zero, “0” e “0,00”, respectivamente.



Repetir os passos para todos os do IPECE e prosseguir registrando

Nesta etapa foi utilizado a plataforma Anaconda

Para desenvolvimento dos trabalhos em Python foi utilizado o Jupyter Notebook da Plataforma Anaconda.

Bibliotecas requeridas para aplicação de Scripts do Python no Power BI: pandas, xlrd, matplotlib, scikit-learn, plotnine, seaborn e notebook.

Verificando as bibliotecas já instaladas com o comando “pip freeze”:

```
In [2]: pip freeze
peps=1.7.1
pexpect==4.8.0
pickleshare==0.7.5
Pillow==7.0.0
piginfo==1.5.0.1
pluggy==0.13.1
ply==3.11
prometheus-client==0.7.1
prompt-toolkit==3.0.3
psutil==5.6.7
py==1.8.1
pycodestyle==2.5.0
pycosat==0.6.3
pycparser==2.19
pycrypto==2.6.1
pycurl==1.43.0.5
pydocstyle==4.0.1
pyflakes==2.1.1
Pygments==2.5.2
pylint==2.4.4
```

Foi percebida a ausência da biblioteca “plotnine”, que a seguir foi instalada com o comando “pip install plotnine”

```
In [3]: pip install plotnine
Collecting plotnine
  Using cached plotnine-0.7.1-py3-none-any.whl (4.4 MB)
Collecting mizani>=0.7.1
  Using cached mizani-0.7.2-py3-none-any.whl (62 kB)
Requirement already satisfied: numpy>=1.16.0 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (1.18.1)
Requirement already satisfied: scipy>=1.2.0 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (1.4.1)
Collecting statsmodels>=0.11.1
  Using cached statsmodels-0.12.2-cp37-none-win_amd64.whl (9.3 MB)
Requirement already satisfied: pandas>=1.1.0 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (1.2.3)
Collecting descartes>=1.1.0
  Using cached descartes-1.1.0-py3-none-any.whl (5.8 kB)
Requirement already satisfied: matplotlib>=3.1.1 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (3.1.3)
Requirement already satisfied: patsy>0.5.1 in c:\users\55859\anaconda3\lib\site-packages (from plotnine) (0.5.1)
Collecting palettable
  Using cached palettable-3.3.0-py2.py3-none-any.whl (111 kB)
Requirement already satisfied: pytz>=2017.3 in c:\users\55859\anaconda3\lib\site-packages (from pandas>=1.1.0->plotnine) (2019.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\55859\anaconda3\lib\site-packages (from pandas>=1.1.0->plotnine) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\55859\anaconda3\lib\site-packages (from matplotlib>=3.1.1->plotnine) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\55859\anaconda3\lib\site-packages (from matplotlib>=3.1.1->plotnine) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in c:\users\55859\anaconda3\lib\site-packages (from matplotlib>=3.1.1->plotnine) (2.4.6)
Requirement already satisfied: six in c:\users\55859\anaconda3\lib\site-packages (from patsy>0.5.1->plotnine) (1.14.0)
Requirement already satisfied: setuptools in c:\users\55859\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=3.1.1->plotnine) (45.2.0.post20200210)
Installing collected packages: palettable, mizani, statsmodels, descartes, plotnine
Successfully installed descartes-1.1.0 mizani-0.7.2 palettable-3.3.0 plotnine-0.7.1 statsmodels-0.12.2
Note: you may need to restart the kernel to use updated packages.
```

4. Análise e Exploração dos Dados

Considerando as peculiaridades da finalidade de uso ‘Abastecimento Humano’, que além de prioritária em relação ao Direito de uso de Água Bruta, também é essencial e, portanto, espera-se esteja presente em todos os municípios em todos os anos, ou que ao menos seja a mais presente.

Nesse sentido, foi obtido um Dataframe selecionando a finalidade de uso ‘Abastecimento Humano’ a partir do Dataframe ‘DIREITO_DE_USO’ (maiúsculo).

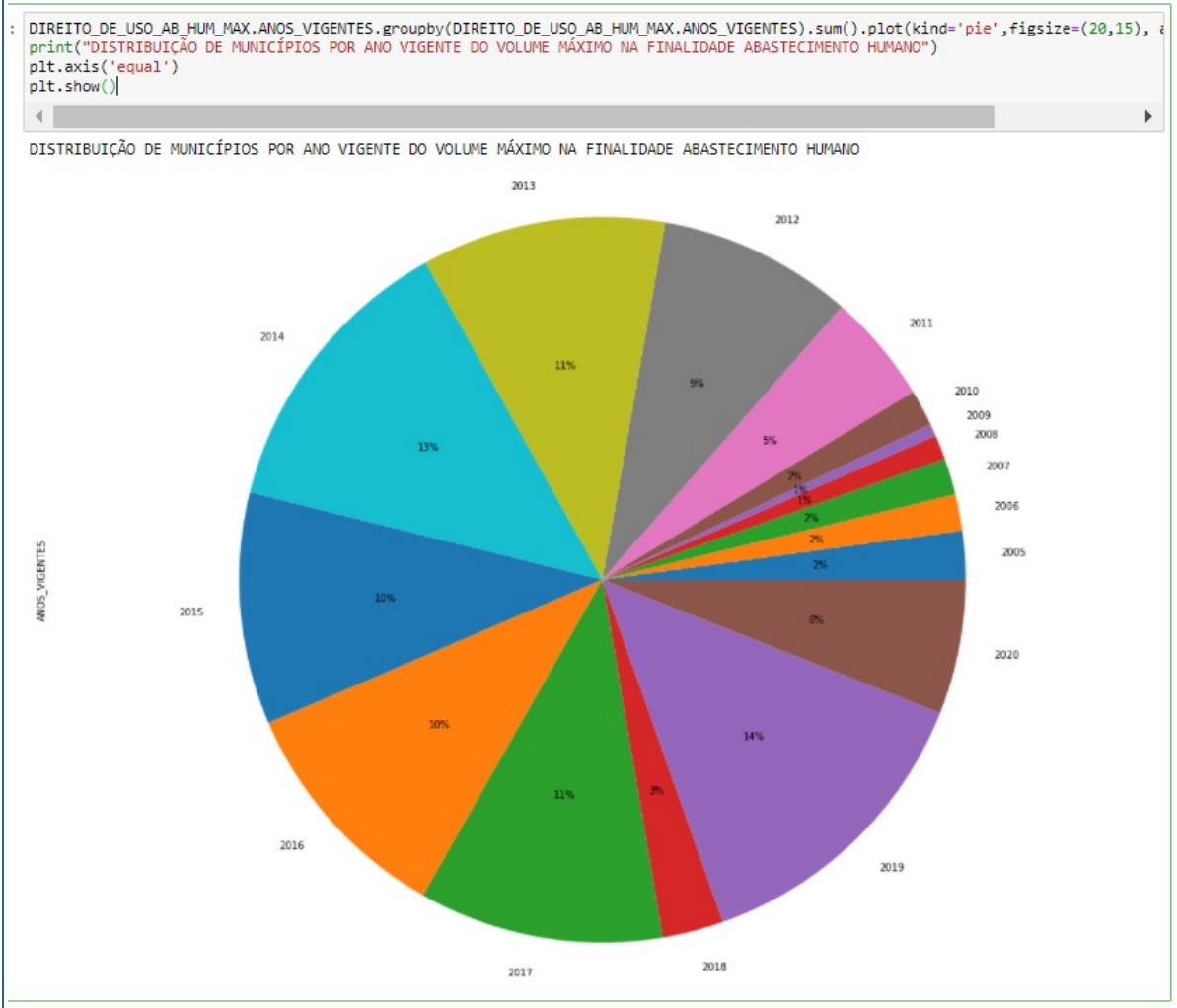
# OBTENDO O DATAFRAME COM A FINALIDADE DE USO 'ABASTECIMENTO HUMANO'						
DIREITO_DE_USO_AB_HUM = DIREITO_DE_USO.loc[(DIREITO_DE_USO['FINALIDADE_DE_USO']) == 'Abastecimento Humano']						
DIREITO_DE_USO_AB_HUM						
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3	
1	ABAIRARA	Abastecimento Humano	2005	0.00000	0.0	0.00000
3	ABAIRARA	Abastecimento Humano	2006	0.00000	0.0	0.00000
5	ABAIRARA	Abastecimento Humano	2007	0.00000	0.0	0.00000
7	ABAIRARA	Abastecimento Humano	2008	0.00000	0.0	0.00000
9	ABAIRARA	Abastecimento Humano	2009	0.00000	0.0	0.00000
...
76171	VÁRZEA ALEGRE	Abastecimento Humano	2026	386305.90625	0.0	386305.90625
76173	VÁRZEA ALEGRE	Abastecimento Humano	2027	0.00000	0.0	0.00000
76175	VÁRZEA ALEGRE	Abastecimento Humano	2028	0.00000	0.0	0.00000
76177	VÁRZEA ALEGRE	Abastecimento Humano	2029	0.00000	0.0	0.00000
76179	VÁRZEA ALEGRE	Abastecimento Humano	2030	0.00000	0.0	0.00000

4784 rows × 6 columns

Para cada município há um conjunto de volumes anuais vigentes. Foi identificado qual o valor do volume máximo dentro desse conjunto de cada município e o ano vigente correspondente.

Espera-se que esse valor seja compatível com a população e seu nível de consumo, lembrando que as populações tendem a aumentar a cada ano.

Como está a distribuição do número de municípios com volume máximo na finalidade abastecimento humano, ou seja, em quais anos há maior/menor concentração de municípios com volume máximo vigente?



Lembrando que os dados de direito de uso no dataset ‘exportacao_cncharh40_CE.xlsx’ contemplam as regularizações de uso realizadas até 2020, sendo que o ano da regularização é considerado o ano inicial de vigência da outorga independente do mês, e que o número de anos de vigência são variáveis. Os anos de vigência podem estender-se além de 2020.

O dataset ‘MUNICÍPIOS_CE_IBGE_original.xlsx’ traz a estimativa de população municipal em 2020, o que poderá nortear a elaboração de alguns parâmetros de consumo de água.

Considerando o ano de 2020 como referencial, como estão distribuídas as ocorrências do volume máximo na finalidade abastecimento humano por ano de vigência - município?



Nesse caso são 94 % dos 184 municípios do Estado apresentaram volume máximo anterior a 2020. Então, os volumes vigentes em 2020 estão inferiores aos de anos anteriores?

Considerando que já estamos em 2021, e tratar-se de abastecimento humano, supõe-se que esses volumes ‘deveriam’ estar iguais ou maiores que volume máximo de cada município.

Será um indício de carência de regularização? Alguma falha ou especificidade nos registros? Será que esses volumes máximos estavam compatíveis com os consumos estimados para suas populações?

Nesse sentido, buscou-se comparar com os volumes máximos obtidos com os volumes de 2020 para observar as diferenças.

Obtendo DF com volumes vigentes em 2020.

# OBTENDO O DATAFRAME COM ANO DE VIGÊNCIA '2020'						
DIREITO_DE_USO_AB_HUM_2020 = DIREITO_DE_USO_AB_HUM.loc[(DIREITO_DE_USO_AB_HUM['ANOS_VIGENTES']) == 2020]						
DIREITO_DE_USO_AB_HUM_2020						
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3	
31	ABAIIARA	Abastecimento Humano	2020	0.00000	1.313312e+05	1.313312e+05
447	ACARAPE	Abastecimento Humano	2020	0.00000	0.000000e+00	0.000000e+00
863	ACARAÚ	Abastecimento Humano	2020	0.00000	7.778588e+05	7.778588e+05
1279	ACOPIARA	Abastecimento Humano	2020	0.00000	1.304875e+06	1.304875e+06
1695	AIUABA	Abastecimento Humano	2020	0.00000	0.000000e+00	0.000000e+00
...
74495	URUBURETAMA	Abastecimento Humano	2020	0.00000	0.000000e+00	0.000000e+00
74911	URUOCA	Abastecimento Humano	2020	0.00000	3.052422e+05	3.052422e+05
75327	VARJOTA	Abastecimento Humano	2020	0.00000	0.000000e+00	0.000000e+00
75743	VIÇOSA DO CEARÁ	Abastecimento Humano	2020	0.00000	5.217310e+04	5.217310e+04
76159	VÁRZEA ALEGRE	Abastecimento Humano	2020	386305.90625	0.000000e+00	3.863059e+05

184 rows x 6 columns

Inserindo os volumes vigentes em 2020 no dataframe “DIREITO_DE_USO_AB_HUM_MAX”.

# INSERINDO SOMA VOLUMES 2020 NO DF						
lista_som_col_abhum2020 = list(DIREITO_DE_USO_AB_HUM_2020['VOLUME_MUN_M3']) # Lista com os elementos que serão inseridos no DF v						
DIREITO_DE_USO_AB_HUM_MAX = DIREITO_DE_USO_AB_HUM_MAX.assign(VOLUME_MUN_ABHUM2020_M3 = lista_som_col_abhum2020)						
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MAX_MUN_M3	VOLUME_MUN_ABHUM2020_M3
25	ABAIIARA	Abastecimento Humano	2017	0.000000e+00	1.313312e+05	1.313312e+05
421	ACARAPE	Abastecimento Humano	2007	1.209600e+05	0.000000e+00	1.209600e+05
851	ACARAÚ	Abastecimento Humano	2014	6.115200e+05	1.408826e+06	2.020346e+06
1263	ACOPIARA	Abastecimento Humano	2012	1.313424e+06	5.376000e+03	1.318800e+06
1685	AIUABA	Abastecimento Humano	2015	7.924089e+05	4.380000e+02	7.928469e+05
...
74477	URUBURETAMA	Abastecimento Humano	2011	3.037440e+06	0.000000e+00	3.037440e+06
74909	URUOCA	Abastecimento Humano	2019	0.000000e+00	3.052422e+05	3.052422e+05
75319	VARJOTA	Abastecimento Humano	2016	4.140864e+06	0.000000e+00	4.140864e+06
75731	VIÇOSA DO CEARÁ	Abastecimento Humano	2014	3.763200e+04	1.184754e+05	1.561074e+05
76143	VÁRZEA ALEGRE	Abastecimento Humano	2012	9.744000e+05	2.426760e+04	9.986676e+05

184 rows x 7 columns

A diferença absoluta entre os municípios revelou-se bastante elástica quanto a ordem de grandeza. Qual o tamanho dessa diferença, por exemplo, em relação aos volumes vigentes em 2020?

Exemplificando:

Município	A	B	C	D
Soma dos volumes vigentes em 2020 (m ³ /ano)	25.000,00	250,00	0,00	100.000.000,00
Máximo volume dentre as somas anuais dos volumes vigentes (m ³ /ano)	100.000,00	1.000,00	75,00	100.000.000,00
Diferença absoluta (m ³ /ano) (Vol_2020 - Vol_Max)	-75.000,00	-750,00	-75,00	0,00
Deficit ((Vol2020 - Vol Max)/ Vol Max)	-0,75	-0,75	-1,00	0,00

No exemplo, o município A possui diferença absoluta 100 vezes maior que o município B, mas o, por hora, deficit proporcional de ambos é de -75%.

O município C apresenta diferença absoluta 1.000 vezes menor que o município A. Já o seu, por hora, deficit proporcional, que é de -100%, é 33,3% maior que o do município A.

Obtendo e inserindo a coluna com o percentual da diferença entre os volumes máximos e vigentes em 2020 de cada município.

OBTENDO O PERCENTUAL EQUIVALENTE DA DIFERENÇA ABSOLUTA EM CADA MUNICÍPIO ENTRE VOLUME MÁXIMO E VOLUME VIGENTE EM 2020 EM RELAÇÃO AO VOLUME MÁXIMO DE 2020
serie_dif_vol_max_2020 = DIREITO_DE_USO_AB_HUM_MAX['VOLUME_MUN_ABHUM2020_M3'] - DIREITO_DE_USO_AB_HUM_MAX['VOLUME_MAX_MUN_M3'] #
per_lista_dif_vol_max_2020 = serie_dif_vol_max_2020 / DIREITO_DE_USO_AB_HUM_MAX['VOLUME_MAX_MUN_M3'] #
INSERINDO O PERCENTUAL NO DF
DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020'] = per_lista_dif_vol_max_2020
DIREITO_DE_USO_AB_HUM_MAX
DADE_DE_USO ANOS_VIGENTES VOL_SUPERF_M3 VOL_SUBTER_M3 VOLUME_MAX_MUN_M3 VOLUME_MUN_ABHUM2020_M3 DIF_PROP_VOL_MAX_VOL_2020
cimento Humano 2017 0.000000e+00 1.313312e+05 1.313312e+05 1.313312e+05 0.000000
cimento Humano 2007 1.209600e+05 0.000000e+00 1.209600e+05 0.000000e+00 -1.000000
cimento Humano 2014 6.115200e+05 1.408826e+06 2.020346e+06 7.778588e+05 -0.614987
cimento Humano 2012 1.313424e+06 5.376000e+03 1.318800e+06 1.304875e+06 -0.010559
cimento Humano 2015 7.924089e+05 4.380000e+02 7.928469e+05 0.000000e+00 -1.000000
...
cimento Humano 2011 3.037440e+06 0.000000e+00 3.037440e+06 0.000000e+00 -1.000000
cimento Humano 2019 0.000000e+00 3.052422e+05 3.052422e+05 3.052422e+05 0.000000
cimento Humano 2016 4.140864e+06 0.000000e+00 4.140864e+06 0.000000e+00 -1.000000
cimento Humano 2014 3.763200e+04 1.184754e+05 1.561074e+05 5.217310e+04 -0.665787
cimento Humano 2012 9.744000e+05 2.426760e+04 9.986676e+05 3.863059e+05 -0.613179

Verificando a ausência de valores nos cálculos executados para obter o percentual da diferença entre Volume Máximo x Volume 2020 - 'DIF_PROP_VOL_MAX_VOL_2020'. São esperadas 04 ocorrências, pois tem-se 04 municípios com Vazão Máxima zero, que podem ser substituídos por zero.

```
DIREITO_DE_USO_AB_HUM_MAX.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 25 to 76143
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MUNICIPIO        184 non-null    object  
 1   FINALIDADE_DE_USO 184 non-null    category
 2   ANOS_VIGENTES    184 non-null    int64   
 3   VOL_SUPERF_M3    184 non-null    float32 
 4   VOL_SUBTER_M3    184 non-null    float64  
 5   VOLUME_MAX_MUN_M3 184 non-null    float64  
 6   VOLUME_MUN_ABHUM2020_M3 184 non-null    float64  
 7   DIF_PROP_VOL_MAX_VOL_2020 180 non-null    float64  
dtypes: category(1), float32(1), float64(4), int64(1), object(1)
memory usage: 15.4+ KB
```

Alterando campos da coluna 'DIF_PROP_VOL_MAX_VOL_2020' com ausência de valor 'NaN' para zero.

```
DIREITO_DE_USO_AB_HUM_MAX.DIF_PROP_VOL_MAX_VOL_2020.fillna(0, inplace=True)

DIREITO_DE_USO_AB_HUM_MAX.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 25 to 76143
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MUNICIPIO        184 non-null    object  
 1   FINALIDADE_DE_USO 184 non-null    category
 2   ANOS_VIGENTES    184 non-null    int64   
 3   VOL_SUPERF_M3    184 non-null    float32 
 4   VOL_SUBTER_M3    184 non-null    float64  
 5   VOLUME_MAX_MUN_M3 184 non-null    float64  
 6   VOLUME_MUN_ABHUM2020_M3 184 non-null    float64  
 7   DIF_PROP_VOL_MAX_VOL_2020 184 non-null    float64  
dtypes: category(1), float32(1), float64(4), int64(1), object(1)
memory usage: 15.4+ KB
```

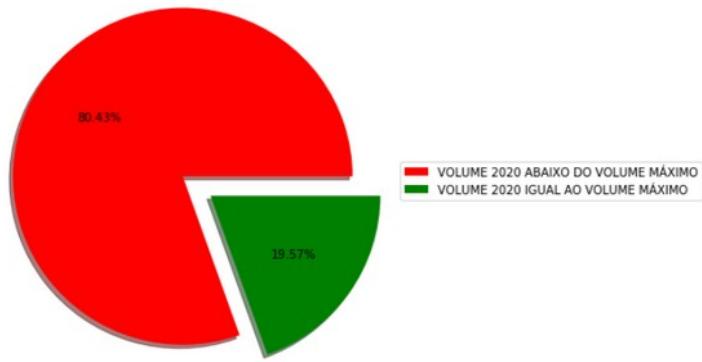
Como se dá o comportamento dessas diferenças ?

Incidência de municípios comparando volume vigente em 2020 aos volumes máximos históricos.

Lembrando que não há volume vigente em 2020 superior ao volume máximo, pois a janela histórica do volume máximo já inclui 2020.

```
#INCIDÊNCIA DE MUNICÍPIOS COMPARANDO OS VOLUMES VIGENTES EM 2020 AOS VOLUMES MÁXIMOS
vol2020_abaixo_vol_max = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020']) < 0])# Calcula o numero de municipios com volume menor que o maximo
vol2020_igual_vol_max = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020']) == 0])# Calcula o numero de municipios com volume igual ao maximo
mun_volume_2020_maximo = [vol2020_abaixo_vol_max, vol2020_igual_vol_max]
labels0 = ['VOLUME 2020 ABAIXO DO VOLUME MÁXIMO', 'VOLUME 2020 IGUAL AO VOLUME MÁXIMO',]
cores0 = ['r', 'g'] # Definindo as cores
explode0 = (0.1,0.1) # Define nível de separabilidade entre as partes, ordem do vetor representa as partes
plt.pie(mun_volume_2020_maximo, autopct='%.1f%%', shadow=True, explode=explode0, colors = cores0) # Define o formato de visualização
# plt.legend(labels0, loc=0)# Insere a Legenda e a localização da Legenda
ax0.legend(loc="center left",bbox_to_anchor=(1,0,0.5,1))# Insere a Legenda e a localização da Legenda
ax0.set_title("INCIDÊNCIA DE MUNICÍPIOS COMPARANDO OS VOLUMES VIGENTES EM 2020 EM RELAÇÃO AOS VOLUMES MÁXIMOS HISTÓRICOS")#Título do gráfico
plt.axis('equal')# Diz que o gráfico será plotado em círculo
plt.show()
```

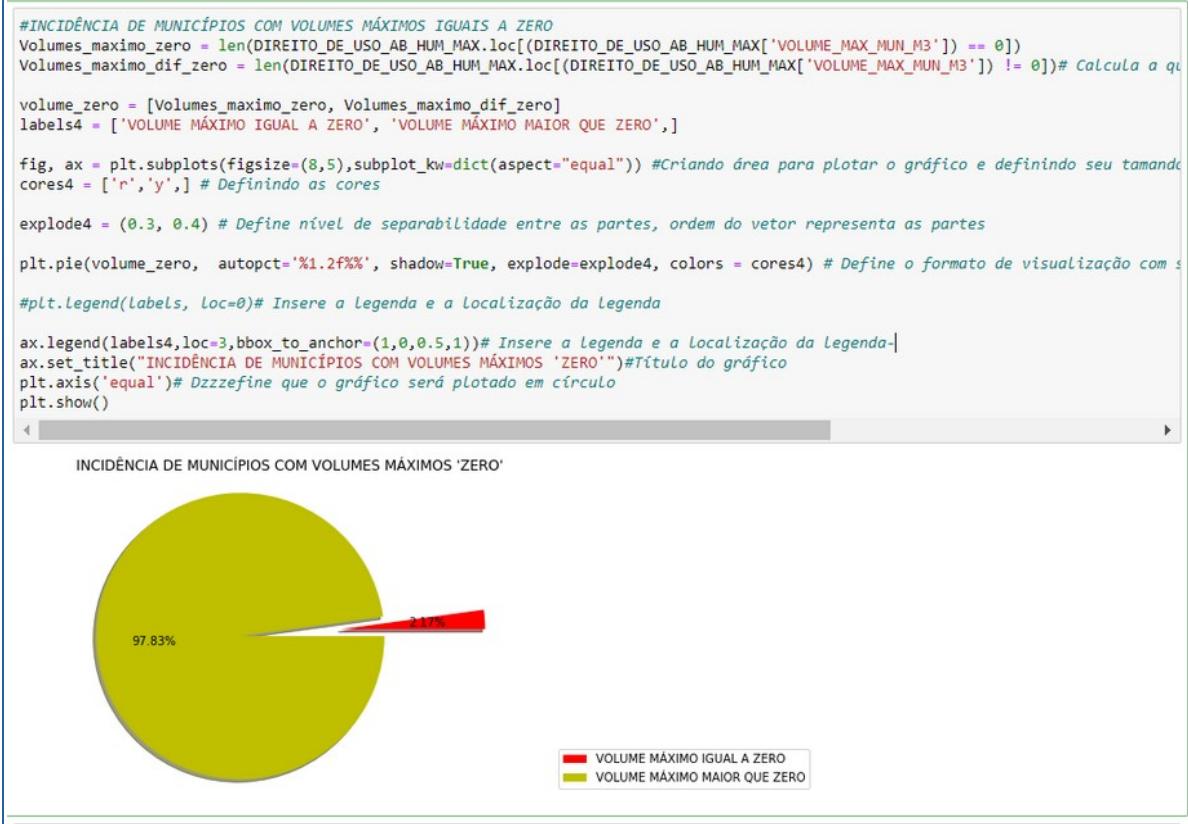
INCIDÊNCIA DE MUNICÍPIOS - VOLUMES VIGENTES EM 2020 EM RELAÇÃO AOS VOLUMES MÁXIMOS HISTÓRICOS



Um volume máximo ser igual ao volume vigente em 2020 é suficiente como indicativo de que a regularização do uso, na finalidade observada, está dentro do esperado?

Há algum município com volume máximo vigente igual a zero?

Esses municípios representam 2,17 % do total de municípios do Estado.



Em quais municípios há volume máximo igual a zero?

MUNICIPIO FINALIDADE_DE_USO ANOS_VIGENTES VOL_SUPERF_M3 VOL_SUBTER_M3 VOLUME_MAX_MUN_M3 VOLUME_MUN_ABHUM2020_M3						
3329	AMONTADA	Abastecimento Humano	2005	0.0	0.0	0.0
29953	IBIAPINA	Abastecimento Humano	2005	0.0	0.0	0.0
49505	MUCAMBO	Abastecimento Humano	2005	0.0	0.0	0.0
58241	PINDORETAMA	Abastecimento Humano	2005	0.0	0.0	0.0

O volume máximo vigente igual a zero é um indicativo de que nunca ocorreu regularização de uso de água bruta nesses municípios? Há alguma justificativa?

Observando dados estatísticos referentes à incidência de municípios e aos valores da diferença percentual volume máximo e volume 2020.

```
# OBSERVANDO DADOS ESTATÍSTICO REFERENTES À INCIDÊNCIA DE MUNICÍPIOS E AOS VALORES DA DIFERENÇA PERCENTUAL VOLUME MÁXIMO E VOLUME 2020
DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020'].describe()
```

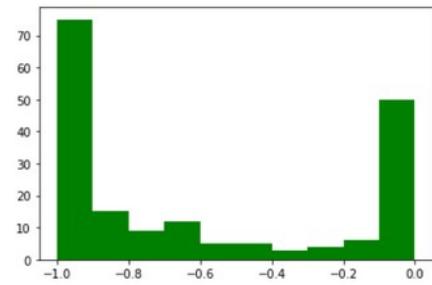
	count	mean	std	min	25%	50%	75%	max	Name:
	184.000000	-0.592801	0.416930	-1.000000	-0.994460	-0.764668	-0.024943	0.000000	DIF_PROP_VOL_MAX_VOL_2020, dtype: float64

Visualizando a distribuição da diferença percentual volume máximo e volume 2020.

```
# VISUALIZANDO A DISTRIBUIÇÃO DA DIFERENÇA PERCENTUAL VOLUME MÁXIMO E VOLUME 2020
print('DISTRIBUIÇÃO DA INCIÊNCIA DE MUNICÍPIOS POR FAIXA PROPORCIONAL DA DIFERENÇA ENTRE O VOLUME MÁXIMO ')
print('E O VOLUME 2020 EM RELAÇÃO AO VOLUME MÁXIMO HISTÓRICO')
plt.hist(DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020'], bins=10,color='g')
```

DISTRIBUIÇÃO DA INCIÊNCIA DE MUNICÍPIOS POR FAIXA PROPORCIONAL DA DIFERENÇA ENTRE O VOLUME MÁXIMO
E O VOLUME 2020 EM RELAÇÃO AO VOLUME MÁXIMO HISTÓRICO

```
(array([75., 15., 9., 12., 5., 3., 4., 6., 50.]),
 array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ]),
 <a list of 10 Patch objects>)
```



As maiores concentrações ocorrem nos extremos, ou seja, a maior entre -1,0 e 0,9, e a segunda maior entre -0,1 e 0,0.

Os percentuais das diferenças volumétricas anuais entre volume vigente 2020 e volume máximo vigente em relação ao volume máximo vigente podem variar de (-1,0) até (0,0), conforme abaixo:

Deficit	%	Diferença Percentual do Volume Vigente em 2020 Em relação ao Volume Máximo Vigente
-1,00	100	Máximo deficit de regularização
0,00	0	Mínimo deficit de regularização

Considerando os critérios iniciais ora observados, quer sejam, finalidade de uso abastecimento humano, soma do volume máximo anual vigente no município, observa-se que conforme a frequência de municípios, as faixas de deficit podem ser agrupadas em 04 níveis, lembrando que se a diferença é zero, mas o volume máximo é zero, certamente tem-se um problema.:

Faixas de deficit diferença percentual entre Volume máximo e Volume 2020			
	Volume máximo = 0,00		
	- 1,0	-----	- 0,9
	- 0,9	-----	- 0,1
	- 0,1	-----	0,0

Onde, 75 dos 184 municípios apresentam, entre 90% e 100%, de deficit de regularização no volume anual vigente, e em 46, entre 0% e 10%.

Em outras palavras, em pelo menos 75 municípios a soma de todos os volumes vigentes simultaneamente em 2020, estaria mais de 90% abaixo do valor máximo já alcançado em somas dos volumes vigentes simultaneamente num mesmo ano já registrados nesses municípios, tendo como finalidade de uso da água no abastecimento humano. Isso representa 39,89% dos municípios, e dessa fatia, metade está 100% abaixo do volume máximo vigente histórico. Há ainda, 04 municípios sem ocorrência de registro de volume anual na finalidade abastecimento humano em seus históricos.

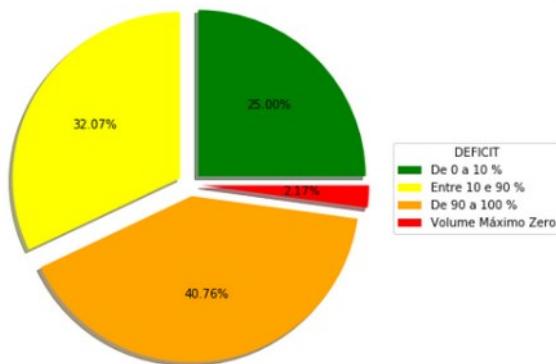
```

#INCIDÊNCIA DE MUNICÍPIOS POR FAIXA DE DEFICT EM VOLUMES VIGENTES EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS JÁ VIGENTES CONFORME RE

de_0_10 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020']) >= -0.1]) - Volumes_maximo_ze
de_90_100 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020']) <= -0.9])# Calcula a quanti
entre_10_90 = len(DIREITO_DE_USO_AB_HUM_MAX['DIF_PROP_VOL_MAX_VOL_2020'])] - de_0_10 - de_90_100 - Volumes_maximo_zero # Calcula o

deficit = [de_0_10, entre_10_90, de_90_100,Volumes_maximo_zero]# Faixas estabelecidas
labels3 = ['0 a 10 %', 'Entre 10 e 90 %', 'De 90 a 100 %', 'Volume Máximo Zero']#definindo textos da legendas
fig, ax = plt.subplots(figsize=(6,6),subplot_kw=dict(aspect="equal")) #Criando área para plotar o gráfico e definindo seu tamande
cores = ['green','yellow','orange','red'] # Definindo as cores
explode3 = (0.07,0.07,0.07,0.07) # Define nível de separabilidade entre as partes, ordem do vetor representa as partes
plt.pie(deficit, autopct='%1.2f%%', shadow=True, explode=explode3, colors = cores ) # Define o formato de visualização com saíde
ax.legend(labels3, title = "DEFICIT",loc="center left",bbox_to_anchor=(1,0,0.5,1))# Insere a legenda e a localização da Legenda
ax.set_title("MUNICÍPIOS POR FAIXA DE DEFICT EM VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS JÁ VIGENTES")#Títu
plt.axis('equal')# Dzzefine que o gráfico será plotado em círculo
plt.show()#Exibir o gráfico-

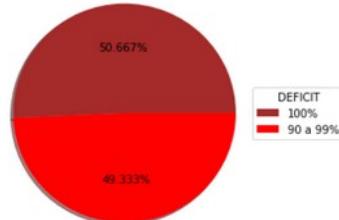
```



```

#DISTRIBUIÇÃO DOS MUNICÍPIOS DENTRO DA FATIA DE 90 A 100% DE DEFÍCIT DE VOLUME VOLUME VIGENTE EM 2020 COMPARADOS AOS VOLUME M
de_cem = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['DEF_PROP_VOL_MAX_VOL_2020']) ==-1.0])
dif_defict_90_10 = de_90_100 - de_cem
defict_90_10_valores =[de_cem, dif_defict_90_10]
defict_90_10 = ['100%', '90 a 99%']
cores2 = ['brown', 'red'] # Definindo as cores
fig2, ax2 = plt.subplots(figsize=(4,4), subplot_kw=dict(aspect="equal")) #Criando área para plotar o gráfico e definindo seu tam
plt.pie(defict_90_10_valores, autopct='%1.3f%%', shadow=True, colors = cores2) # Define o formato de visualização com saída em %
ax2.legend(defict_90_10, title = "DEFÍCIT", loc = "center left", bbox_to_anchor=(1,0,0.5,1))# Insere a Legenda e a localização de
ax2.set_title("DISTRIBUIÇÃO DOS MUNICÍPIOS DENTRO DA FATIA DE 90 A 100% DE DEFÍCIT DE VOLUME VOLUME VIGENTE EM 2020 COMPARADOS
plt.axis('equal')# Dizze que o gráfico será plotado em círculo
plt.show()#Exibir o gráfico-

```



Os volumes anuais vigentes são apropriados aos consumos e populações? Estariam superestimados? Ou subestimados?

Em 2020, não foi realizado o CENSO pelo IBGE, temos a população estimada para 2020 para cada município.

Nos projetos de engenharia voltados para o abastecimento público, o dimensionamento das bombas de captação, das adutoras, das estações de elevação, das redes de distribuição, estações de tratamento d'água, etc, são baseados na vazão que se pretende transportar através mesmas. Para determinar essa vazão, é definido um horizonte para projeto, no qual é vislumbrado um crescimento da população que será atendida pelo projeto. Para efeito de estimativa, geralmente assume-se que a população cresce numa taxa de 2% ao ano.

A partir da população estimada para 2020, será simulada a população no ano de ocorrência do ao volume máximo histórico de cada município.

Inserindo a população estimada para 2020 no df

'DIREITO_DE_USO_AB_HUM_MAX'

```
#INSERINDO A POPULAÇÃO ESTIMADA PARA 2020
DIREITO_DE_USO_AB_HUM_MAX.sort_values(by=['MUNICIPIO'], ascending=True, inplace=True)#Ordenando pelos municípios
lista_POP_ESTIMADA_2020 = list(IBGE_IPECE['POP_ESTIMADA_2020'])#Gerando uma lista a partir de uma coluna do df 'IBGE_IPECE'
DIREITO_DE_USO_AB_HUM_MAX.insert(8,'POP_ESTIMADA_2020',lista_POP_ESTIMADA_2020,True)#Inserindo uma lista no df 'DIREITO_DE_USO_AB_HUM_MAX'
```

VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MAX_MUN_M3	VOLUME_MUN_ABHUM2020_M3	DIF_PROP_VOL_MAX_VOL_2020	POP_ESTIMADA_2020
2017	0.000000e+00	1.313312e+05	1.313312e+05	1.313312e+05	0.000000	11853
2007	1.209600e+05	0.000000e+00	1.209600e+05	0.000000e+00	-1.000000	15036
2014	6.115200e+05	1.408826e+06	2.020346e+06	7.778588e+05	-0.614987	63104
2012	1.313424e+06	5.376000e+03	1.318800e+06	1.304875e+06	-0.010559	54481
2015	7.924089e+05	4.380000e+02	7.928469e+05	0.000000e+00	-1.000000	17493
...
2011	3.037440e+06	0.000000e+00	3.037440e+06	0.000000e+00	-1.000000	22040
2019	0.000000e+00	3.052422e+05	3.052422e+05	3.052422e+05	0.000000	13915
2016	4.140864e+06	0.000000e+00	4.140864e+06	0.000000e+00	-1.000000	18471
2012	9.744000e+05	2.426760e+04	9.986676e+05	3.863059e+05	-0.613179	40903
2014	3.763200e+04	1.184754e+05	1.561074e+05	5.217310e+04	-0.665787	61410

Estimando o consumo da população, retrocedida a partir de 2020 até o ano de volume máximo histórico.

```
#ESTIMANDO O CONSUMO DA POPULAÇÃO, RETROCEDIDA A PARTIR DE 2020, NO ANO DE VOLUME MÁXIMO HISTÓRICO
tx_aa = 0.02# taxa de crescimento 2% ao ano, retrocedendo fica negativa a 2% ao ano
dcres = 1 - tx_aa
for i in DIREITO_DE_USO_AB_HUM_MAX.index:
    anos_atras = 2020 - DIREITO_DE_USO_AB_HUM_MAX['ANOS_VIGENTES']# Quantidade de anos anteriores da ocorrência do volume máximo
    dcres = 1 - tx_aa # decréscimo anual
    pop_ano_max = DIREITO_DE_USO_AB_HUM_MAX['POP_ESTIMADA_2020'] * dcres ** anos_atras # Montante=Capital(1+taxa)^período
    vol_hab_dia_ano_max = DIREITO_DE_USO_AB_HUM_MAX['VOLUME_MAX_MUN_M3']*1000/(365*pop_ano_max)#1m³ = 1000 litros, ano = 365 dias
    DIREITO_DE_USO_AB_HUM_MAX['POP_EST_ANO_VOL_MAX'] = pop_ano_max.astype('int32') #Inserindo a coluna no df
    DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX'] = vol_hab_dia_ano_max #Inserindo a coluna no df
```

DIREITO_DE_USO_AB_HUM_MAX						
E_MAX_MUN_M3	VOLUME_MUN_ABHUM2020_M3	DIF_PROP_VOL_MAX_VOL_2020	POP_ESTIMADA_2020	POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	
1.313312e+05	1.313312e+05	0.000000	11853	11155	32.252883	
1.209600e+05	0.000000e+00	-1.000000	15036	11563	28.660094	
2.020346e+06	7.778588e+05	-0.614987	63104	55900	99.019230	
1.318800e-06	1.304875e+06	-0.010559	54481	46350	77.952922	
7.928469e+05	0.000000e+00	-1.000000	17493	15812	137.373136	
...	
3.037440e+06	0.000000e+00	-1.000000	22040	18375	452.864807	
3.052422e+05	3.052422e+05	0.000000	13915	13636	61.325685	
4.140864e+06	0.000000e+00	-1.000000	18471	17037	665.891378	
9.986676e+05	3.863059e+05	-0.613179	40903	34798	78.625664	
1.561074e+05	5.217310e+04	-0.665787	61410	54399	7.862035	

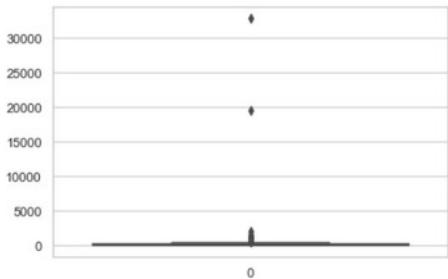
Visualizando a distribuição do consumo em litros diários por habitante no ano de volume máximo histórico.



A esmagadora concentração na primeira metade sugere presença de alguns 'outliers'.

Detectando anômalos

```
#GERANDO BOXPLOT DA DISTRIBUIÇÃO DO CONSUMO DIÁRIO POR HABITANTE NOS MUNICÍPIOS
lista_vol_hab_dia_ano_max = list(vol_hab_dia_ano_max)# Gerando lista
%matplotlib inline
sns.set(style="whitegrid", color_codes=True)#Definindo parâmetros
sns.boxplot(data=lista_vol_hab_dia_ano_max);#Gerando Boxplot
```



Identificando e verificando a incidência de ocorrência dos 'outliers' identificados

```
#IDENTIFICANDO ANÔMALOS
outliers_vol_hab_dia_ano_max = []# Criando lista para receber outliers

#Estabelecendo um limite superior e inferior e quaisquer valores que estiverem acima ou abaixo desses limites serão considerados
#Valores acima da linha superior serão todos os valores que forem maior que o valor da média mais o valor de 02 vezes o desvio padrão.
#0 Limite inferior será o valor da média menos 02 vezes o valor do desvio padrão.

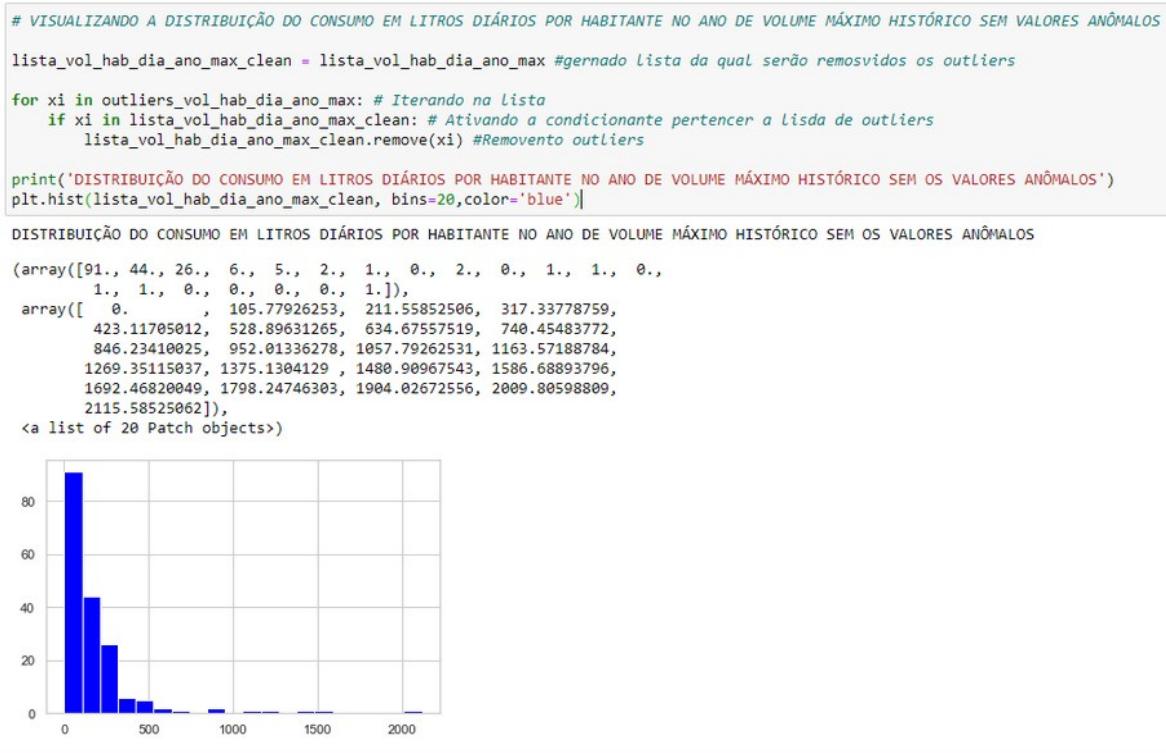
mean2 = np.mean(lista_vol_hab_dia_ano_max, axis=0) # Obtendo a média
sd2 = np.std(lista_vol_hab_dia_ano_max, axis=0) # Obtendo o desvio padrão

for z in lista_vol_hab_dia_ano_max: #Listando os valores abaixo da média em 02 vezes o desvio padrão
    if (z > mean2 + 2* sd2):
        outliers_vol_hab_dia_ano_max.append(z)
for z in lista_vol_hab_dia_ano_max: #Listando os valores acima da média em 02 vezes o desvio padrão
    if (z < mean2 - 2* sd2):
        outliers_vol_hab_dia_ano_max.append(z)
|
#OBSERVANDO A INCIDÊNCIA DOS OUTLIERS IDENTIFICADOS
print('OUTLIERS IDENTIFICADOS --> quantidade')
tout = 0
for out in outliers_vol_hab_dia_ano_max:
    nout = outliers_vol_hab_dia_ano_max.count(out)
    tout = tout + nout
    print(out,'-->', nout)
```

OUTLIERS IDENTIFICADOS --> quantidade
32941.4406228367 --> 1
19520.235634285764 --> 1

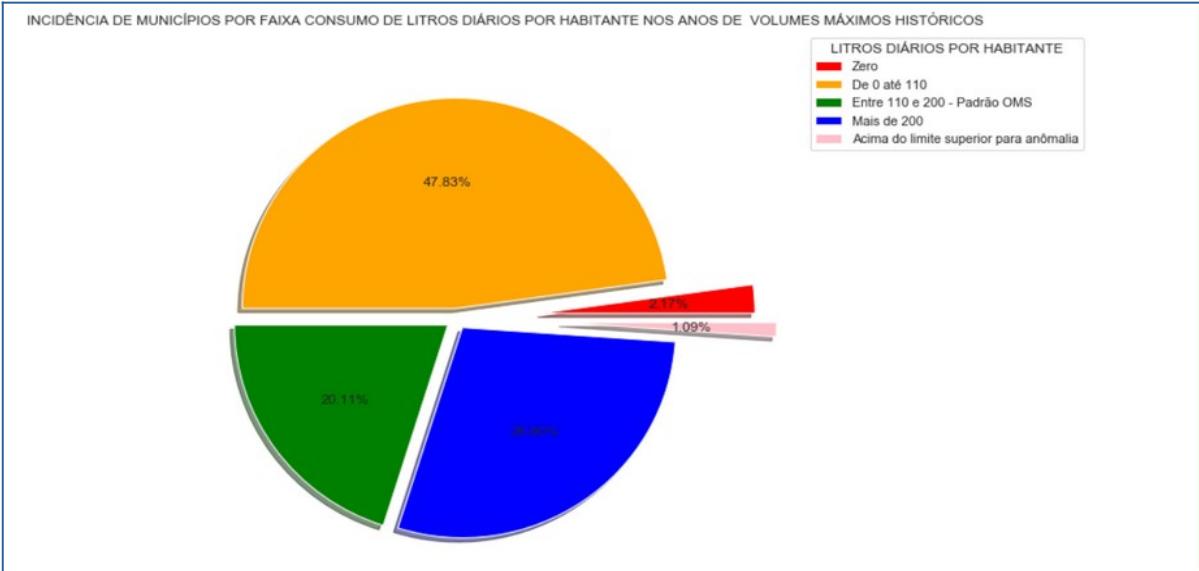
Visualizando a distribuição do consumo em litros diários por habitante no ano de volume máximo histórico sem valores anômalos.

Há maior concentração de ocorrências entre 0.0 e 105.77926253, uns 90 municípios e em seguida, outros 42 entre 105.77926253 e 211.55852506, somando aproximadamente 132 municípios.



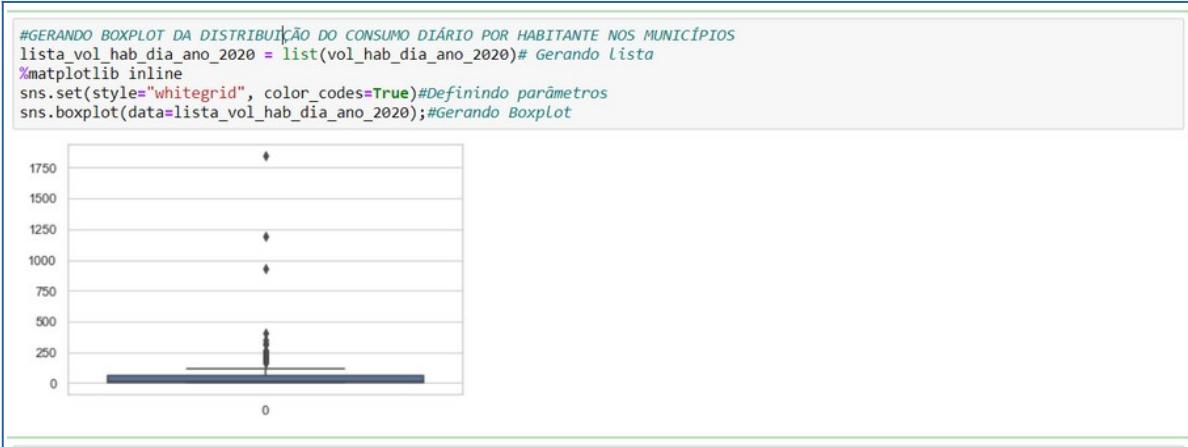
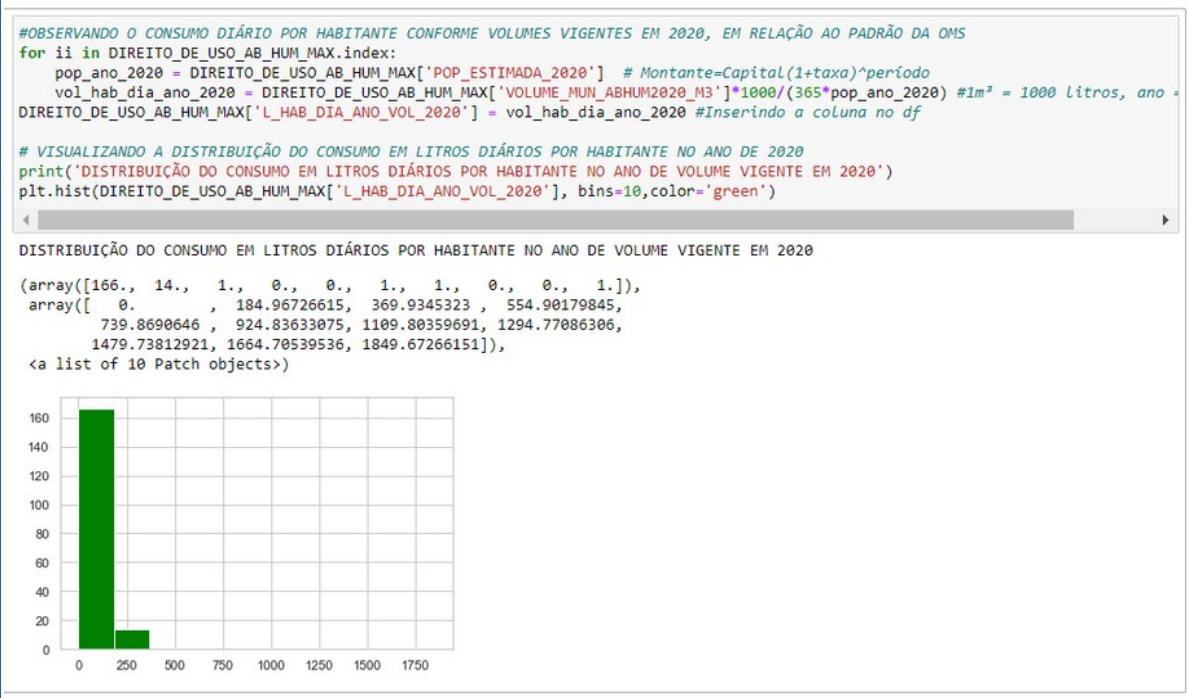
De acordo com a Organização das Nações Unidas, cada pessoa necessita de cerca de 110 litros de água por dia para atender as necessidades de consumo e higiene. No entanto, no Brasil, o consumo por pessoa pode chegar a mais de 200 litros/dia. Assim chamaremos de Padrão OMS-Brasil essa faixa de consumo entre 110 e 200 litros diários por habitantes.

```
#INCIDÊNCIA DE MUNICÍPIOS POR FAIXA CONSUMO NOS ANOS DE VOLUMES MÁXIMOS HISTÓRICOS
outmais = mean2 + 2* sd2 #Valores acima da linha superior serão todos os valores que forem maior que o valor da média mais o valor da desvio padrão multiplicado por 2
oms_zero = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX']) == 0])
oms_de_0_110 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX']) < 110]) - oms_zero # Calcula o número de municípios que consumem menos de 110 litros
oms_out_mais = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX']) > outmais])# Calcula a quantidade de municípios que consumem acima do limite superior para anomalia
oms_mais_200 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX']) > 200])-oms_out_mais # Calcula a quantidade de municípios que consumem mais de 200 litros
oms_entre_110_200 = len(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX']) - oms_zero - oms_de_0_110 - oms_mais_200 - oms_out_mais # Calcula a quantidade de municípios que consumem entre 110 e 200 litros
oms = [oms_zero, oms_de_0_110, oms_entre_110_200, oms_mais_200, oms_out_mais]# Faixas estabelecidas
labels9 = ['Zero', 'De 0 até 110', 'Entre 110 e 200 - Padrão OMS', 'Mais de 200', 'Acima do limite superior para anomalia'] #definindo as legendas
fig9, ax9 = plt.subplots(figsize=(9,9), subplot_kw=dict(aspect="equal")) #Criando área para plotar o gráfico e definindo seu tamanho
cores9 = ['red', 'orange', 'green', 'blue', 'pink'] # Definindo as cores
explode9 = (0.4,0.05, 0.05,0.05, 0.5) # Define nível de separabilidade entre as partes, ordem do vetor representa as partes
plt.pie(oms, autopct='%.1f%%', shadow=True, explode=explode9, colors = cores9 ) # Define o formato de visualização com saída em porcentagem
ax9.legend(labels9, title = "LITROS DIÁRIOS POR HABITANTE", loc=2, bbox_to_anchor=(1,0,0.5,1))# Insere a legenda e a localização da mesma
ax9.set_title("INCIDÊNCIA DE MUNICÍPIOS POR FAIXA CONSUMO DE LITROS DIÁRIOS POR HABITANTE NOS ANOS DE VOLUMES MÁXIMOS HISTÓRICOS")
plt.axis('equal')# Define que o gráfico será plotado em círculo
plt.show()#Exibir o gráfico
```



Considerando o volume máximo e população 2020, qual seria esse consumo diário por habitante em litros?

E quanto aos volumes vigentes em 2020, como está essa distribuição em relação ao padrão de consumo OMS-Brasil?



```

: #Estabelecendo um limite superior e inferior e quaisquer valores que estiverem acima ou abaixo desses limites serão considerados
#Valores acima da linha superior serão todos os valores que forem maior que o valor da média mais o valor de 02 vezes o desvio padrão
#O limite inferior será o valor da média menos 02 vezes o valor do desvio padrão
mean1 = np.mean(lista_vol_hab_dia_ano_2020, axis=0) # Obtendo a média
sd1 = np.std(lista_vol_hab_dia_ano_2020, axis=0) # Obtendo o desvio padrão
for z in lista_vol_hab_dia_ano_2020: #Listando os valores abaixo da média em 02 vezes o desvio padrão
    if (z > mean1 + 2* sd1):
        outliers_vol_hab_dia_ano_2020.append(z)
for z in lista_vol_hab_dia_ano_2020: #Listando os valores acima da média em 02 vezes o desvio padrão
    if (z < mean1 - 2* sd1):
        outliers_vol_hab_dia_ano_2020.append(z)

#OBSERVANDO A INCIDÊNCIA DOS OUTLIERS IDENTIFICADOS
print('OUTLIERS IDENTIFICADOS --> quantidade')
tout1 = 0
for out1 in outliers_vol_hab_dia_ano_2020:
    tout1 = outliers_vol_hab_dia_ano_2020.count(out1)
    tout1 = tout1 + tout1
    print(out1,'-->', tout1)

# VISUALIZANDO A DISTRIBUIÇÃO DO CONSUMO EM LITROS DIÁRIOS POR HABITANTE NO ANO DE VOLUME VIGENTE EM 2020 SEM VALORES ANÔMALOS
lista_vol_hab_dia_ano_2020_clean = lista_vol_hab_dia_ano_2020 #gerando lista da qual serão removidos os outliers
for xii in outliers_vol_hab_dia_ano_2020: # Iterando na lista
    if xii in lista_vol_hab_dia_ano_2020_clean: # Ativando a condicionante pertencer a lista de outliers
        lista_vol_hab_dia_ano_2020_clean.remove(xii) #Removendo outliers
print('DISTRIBUIÇÃO DO CONSUMO EM LITROS DIÁRIOS POR HABITANTE EM ANO 2020 SEM OS VALORES ANÔMALOS')
plt.hist(lista_vol_hab_dia_ano_2020_clean, bins=20,color='purple')

```

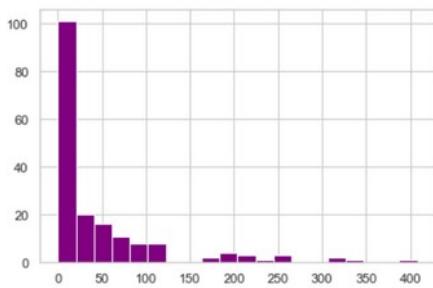
```

OUTLIERS IDENTIFICADOS --> quantidade
1849.6726615083692 --> 2
928.8381348841292 --> 2
1195.674256215232 --> 2
1849.6726615083692 --> 2
928.8381348841292 --> 2
1195.674256215232 --> 2
DISTRIBUIÇÃO DO CONSUMO EM LITROS DIÁRIOS POR HABITANTE EM ANO 2020 SEM OS VALORES ANÔMALOS

(array([101.,  20.,  16.,  11.,   8.,   8.,   0.,   0.,   2.,   4.,   3.,
       1.,   3.,   0.,   0.,   2.,   1.,   0.,   0.,   1.]),
 array([ 0.          , 20.40511439, 40.81022878, 61.21534317,
        81.62045756, 102.02557195, 122.43068635, 142.83580074,
       163.24091513, 183.64602952, 204.05114391, 224.4562583 ,
      244.86137269, 265.26648708, 285.67160147, 306.07671586,
     326.48183025, 346.88694465, 367.29205904, 387.69717343,
     408.10228782]),

<a list of 20 Patch objects>

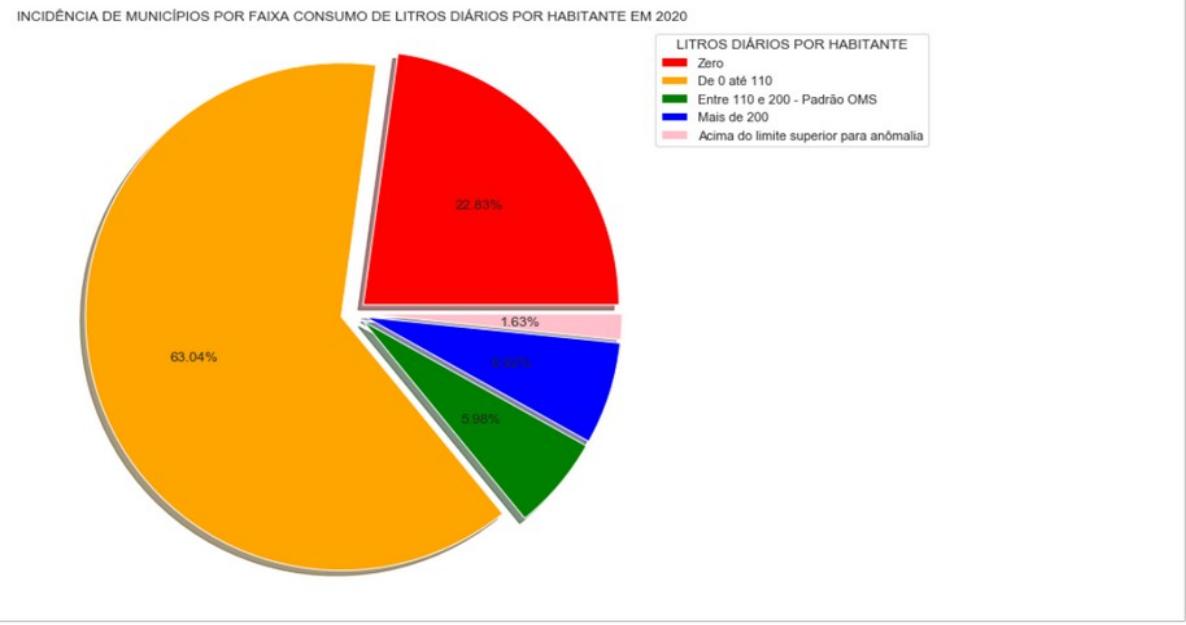
```



E quanto aos volumes vigentes em 2020, como está essa distribuição em relação ao padrão de consumo OMS-Brasil?

```
#INCIDÊNCIA DE MUNICÍPIOS POR FAIXA CONSUMO NO ANO DE 2020

outmais_1 = mean1 + 2* sd1 #Valores acima da linha superior serão todos os valores que forem maior que o valor da média mais o dobro da desvio padro
oms_zero_1 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_2020']) == 0])
oms_de_0_110_1 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_2020']) < 110]) - oms_zero_1 # Calcula o número de municípios com consumo entre 0 e 110 litros diáridos
oms_out_mais_1 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_2020']) > outmais_1])# Calcula o número de municípios com consumo acima da média mais o dobro da desvio padro
oms_mais_200_1 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_2020']) > 200])- oms_out_mais_1 # Calcula o número de municípios com consumo acima de 200 litros diáridos
oms_entre_110_200_1 = len(DIREITO_DE_USO_AB_HUM_MAX.loc[(DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_2020']) < 200]) - oms_zero_1 - oms_de_0_110_1 - oms_mais_200_1 - oms_out_mais_1 # Calcula o número de municípios com consumo entre 110 e 200 litros diáridos
oms1 = [oms_zero_1, oms_de_0_110_1, oms_entre_110_200_1, oms_mais_200_1, oms_out_mais_1]# Faixas estabelecidas
labels91 = ['Zero', 'De 0 até 110', 'Entre 110 e 200 - Padrão OMS', 'Mais de 200', 'Acima do limite superior para anormalia'] #definindo as faixas
fig91, ax91 = plt.subplots(figsize=(9,9), subplot_kw=dict(aspect="equal")) #Criando área para plotar o gráfico e definindo seu tamanho
cores91 = ['red','orange','green','blue','pink'] # Definindo as cores
explode91 = (0.05,0.05, 0.05,0.05, 0.05) # Define nível de separabilidade entre as partes, ordem do vetor representa as partes
plt.pie(oms1, autopct='%.1f%%', shadow=True, explode=explode91, colors = cores91 ) # Define o formato de visualização com saída
ax91.legend(labels91, title = "LITROS DIÁRIOS POR HABITANTE", loc=2, bbox_to_anchor=(1,0,0.5,1))# Insere a legenda e a localização
ax91.set_title("INCIDÊNCIA DE MUNICÍPIOS POR FAIXA CONSUMO DE LITROS DIÁRIOS POR HABITANTE EM 2020")#Título do gráfico
plt.axis('equal')# Define que o gráfico será plotado em círculo
plt.show()#Exibir o gráfico
```



Supondo o consumo mínimo da faixa padrão da OMS-Brasil e a população estimada para 2020, que volume anual seria necessário estar vigente para atender essa demanda na finalidade abastecimento humano em cada município?

Esse volume poderá compor a análises considerando outras finalidades de uso da água em conjunto.

#VOLUME VIGENTE EM 2020 CONSIDERANDO SUPONDO CONSUMO MÍNIMO DA FAIXA PADRÃO DA OMS E A POPULAÇÃO ESTIMADA PELO IBGE PARA 2020						
cons_min_oms = 110/1000 #Convertendo o Limite inferior do padrão OMS para consumo por habitanteem litros diário para m³ diários,						
for iii in DIREITO_DE_USO_AB_HUM_MAX.index:						
vol_ano_2020_min_oms = cons_min_oms *365 * DIREITO_DE_USO_AB_HUM_MAX['POP_ESTIMADA_2020']# Ano = 365 dias						
DIREITO_DE_USO_AB_HUM_MAX['VOL_2020_MIN_OMS'] = vol_ano_2020_min_oms #Inserindo a coluna no df						
DIREITO_DE_USO_AB_HUM_MAX						
DIF_PROP_VOL_MAX_VOL_2020	POP_ESTIMADA_2020	POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	
0.000000	11853	11155.948776	32.252883	30.356156	475897.95	
-1.000000	15036	11563.020645	28.660094	0.000000	603695.40	
-0.614987	63104	55900.197602	99.019230	33.771552	2533625.60	
-0.010559	54481	46350.420233	77.952922	65.619207	2187412.15	
-1.000000	17493	15812.286498	137.373136	0.000000	702343.95	
...
-1.000000	22040	18375.800677	452.864807	0.000000	884906.00	
0.000000	13915	13636.700000	61.325685	60.099171	558687.25	
-1.000000	18471	17037.062283	665.891378	0.000000	741610.65	
-0.613179	40903	34798.759913	78.625664	25.875177	1642255.45	
-0.665787	61410	54399.580609	7.862035	2.327634	2465611.50	

Supondo um indicador para o deficit de regularização dos municípios, iniciando pela finalidade de uso abastecimento humano, com o intuito de estabelecer prioridades nas ações de fortalecimento da regularização do uso da água.

São 03 critérios observados para compor a pontuação:

- A comparação do consumo diários por habitante, no ano de máximo volume considerando a população estimada da época, com consumo nos limites do padrão de referência da OMS-Brasil enriquecido com mais subdivisões.
- A comparação do consumo conforme volumes vigentes em 2020 pela estimativa populacional do mesmo ano com consumo nos limites do padrão de referência da OMS-Brasil, enriquecido com mais subdivisões.
- A comparação entre os consumos no ano de máximo volume vigentes e em volumes vigentes em 2020.

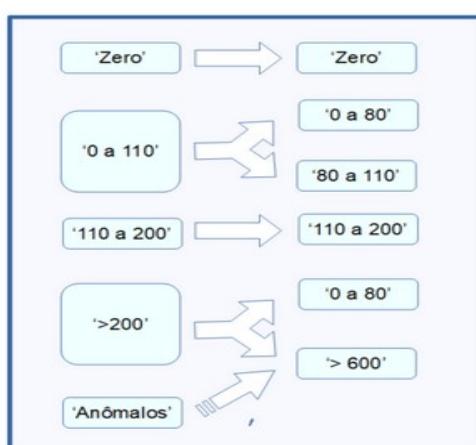
O volume vigente em 2020 pode ser também o consumo máximo do histórico. Espera-se que os consumos obtidos para 2020 não ultrapassem os consumos obtidos para o ano de ocorrência do volume máximo de cada município.

Preliminarmente, na exploração de dados foi percebida a presença de anomalias nos valores de consumo considerando os volumes por municípios, tanto no ano do consumo volume histórico quanto em 2020.

Aprofundando mais na exploração, as faixas de valores foram mais detalhadas visando a elaboração do indicador.

Os valores ‘Acima de 200’, ou seja, acima da faixa padrão OMS-Brasil de consumo diário, passam a ser divididos em valores acima ou abaixo do triplo desse valor, que corresponde a 600 litros diários por habitante. Esse valor é elevado e poderá ser revisto. Abaixo desse patamar foi imaginando um consumo per capita descuidado, com muito conforto hídrico, com sinais de desperdício, mas ainda praticável. Já os valores acima desse patamar sugerem a presença de anomalias entre os registros do município, uma vaga possibilidade de que os valores estejam superestimados para finalidade pode ter sido absorvido em virtude desse uso ser o predominante no registro, conforme algum entendimento adotado na época em que ocorreu.

Os valores ‘De 0 até 110’, ou seja, abaixo da faixa padrão OMS-Brasil de consumo diário, mas diferentes de zero, passam a ser divididos em valores acima ou abaixo de 40% do valor superior da faixa padrão, esse valor corresponde a 80 litros diários por habitante. Acima desse patamar estaria o consumo precário, mas praticado em regiões com escassez hídrica. Abaixo, estaria o consumo com seríssimas restrições, que implicam em problemas de questões sanitárias. Isso pode significar também que o volume de outra finalidade de uso



Definido que uma menor pontuação sugere maior necessidade de regularização, as diversas camadas foram comparadas umas com as outras imaginando uma resposta para a questão: que situação é pior, essa ou aquela? E assim os valores foram sendo atribuídos separadamente nas 03 comparações mencionadas e também considerando sua evolução, ou seja, o consumo em 2020 em relação ao consumo no ano de volume máximo está mais próximo do padrão OMS-Brasil? Estar dentro do padrão ou numa camada acima, em 2020 é mais vantajoso do que estar nessas mesmas camadas no ano de volume máximo.

O consumo em 2020 menor que do ano de volume máximo sugere uma redução na regularização e foi atribuído uma pontuação menor. Numa situação como essa, o consumo do ano de volume máximo também pode ter sido, por exemplo, uma anomalia que posteriormente foi sanada, por isso as pontuações não devem ser observadas apenas isoladamente.

Uma pontuação mais baixa foi atribuída aos consumos mais elevados, apesar deles parecerem atender muito bem aos padrões da OMS-Brasil, pois reflete a possível falha na qualidade dos dados, o que prejudica o planejamento das ações de regularização, pois exigirá maior aprofundamento para identificar as causas dessas anomalias, e consequentemente mais tempo, o que poderia retardar o início dos trabalhos de fomento a regularização.

LEGENDA	
Litros diários por habitante Ano do Máximo Volume Histórico Vigente	Lmax
Litros diários por habitante Ano de 2020	L2020
Pontuação 01	P01
Pontuação 02	P02
Pontuação 03	P03
Pontuação total	PT

PT | P01 + P02 + P03 |

Indicadores de indícios de déficit na regularização do uso da água			
Finalidade de uso: abastecimento humano			
Lmax	P01	L2020 <=> Lmax	P02
Acima 600	-1	Igual	1
200 a 600	1	Menor	0
110 a 200	0		
110 a 80	-1		
Abaixo 80	-2		
Zero	-3		

L2020
Acima 600
200 a 600
110 a 200
110 a 80
Abaixo 80
Zero

Simulando no Calc (LibreOffice) para ilustrar as opções de combinação de ocorrência de valores para os consumos diários máximo e em 2020, partindo dos consumos máximos. O objetivo é observar o comportamento hierárquico da pontuação total (PT).

Município	Lmax	L2020	L2020 <= Lmax	P01	P03	P02	PT
M1	700,00	650,00	<	-1	-1	0	-2
M2	700,00	700,00	=	-1	-1	1	-1
M3	700,00	500,00	<	-1	2	0	1
M4	700,00	150,00	<	-1	1	0	0
M5	700,00	90,00	<	-1	-1	0	-2
M6	700,00	50,00	<	-1	-2	0	-3
M7	700,00	Zero	<	-1	-3	0	-4
M8	500,00	450,00	<	1	2	0	3
M9	500,00	500,00	=	1	2	1	4
M10	500,00	150,00	<	1	1	0	2
M11	500,00	90,00	<	1	-1	0	0
M12	500,00	50,00	<	1	-2	0	-1
M13	500,00	Zero	<	1	-3	0	-2
M14	180,00	150,00	<	1	1	0	2
M15	180,00	180,00	=	0	1	1	2
M16	180,00	90,00	<	0	-1	0	-1
M17	180,00	50,00	<	0	-2	0	-2
M18	180,00	Zero	<	0	-3	0	-3
M19	100,00	100,00	=	-1	-1	1	-1
M20	100,00	90,00	<	-1	-1	0	-2
M21	100,00	50,00	<	-1	-2	0	-3
M22	100,00	Zero	<	-1	-3	0	-4
M23	60,00	60,00	=	-2	-2	1	-3
M24	60,00	50,00	<	-2	-2	0	-4

Ordenando para observar como ficará a classificação, percebe-se formação de 10 grupos hierárquicos conforme a pontuação obtida (4,3,2,1,0,-1,-2,-3,-4,-5). Os pontos atribuídos às situações nas 03 comparações mostrou-se coerente na pontuação total retornada. Os graus de dificuldade estão condizentes.

Vejamos os exemplos:

- Os municípios M9 e M8 têm mesmo Lmax, mas o L2020 de M8 é menor, o que sugere um declínio na regularização.
- O município M15 apresenta L2020 e Lmax menores que os valores de M3, mas não sofreu declínio e está no padrão OMS, enquanto o Lmax de M3 está na faixa de possibilidade de presença de anomalia.
- Por outro lado a situação de M23, também não apresentou declínio, mas nunca esteve nos padrões OMS, portanto sua pontuação está abaixo de M15.

- M26 nunca teve valores registrados, apesar de não apresentar declínio está abaixo de M25, que mesmo estando com zero em 2020, já teve valores registrados.

A existência de valores registrados, pode ser útil na regularização do uso, pois os dados desses usos que constam na base podem nortear os trabalhos dentro do município, ao passo que a ausência total dificulta a abordagem.

Há ainda que se considerar, nesses casos de ausência, a possibilidade de ocorrência, não com muita frequência, de captações em municípios vizinhos, pois os volumes anuais regularizados estão vinculados às coordenadas dos pontos de captação e não de uso. Nesses casos de anomalias e de ausência de registro é válido observar o comportamento dos municípios vizinhos, mais uma vez, a pontuação mais baixa se dá pela maior necessidade de aprofundamento na exploração dos dados devido a maior complexidade de fatores envolvidos.

Lembrando que nesse momento a observação refere-se apenas ao uso na finalidade abastecimento humano.

Município	Lmax	L2020	L2020 =< Lmax	P01	P03	P02	PT
M9	500,00	500,00	=	1	2	1	4
M8	500,00	450,00	<	1	2	0	3
M10	500,00	150,00	<	1	1	0	2
M14	180,00	150,00	<	1	1	0	2
M15	180,00	180,00	=	0	1	1	2
M3	700,00	500,00	<	-1	2	0	1
M4	700,00	150,00	<	-1	1	0	0
M11	500,00	90,00	<	1	-1	0	0
M2	700,00	700,00	=	-1	-1	1	-1
M12	500,00	50,00	<	1	-2	0	-1
M16	180,00	90,00	<	0	-1	0	-1
M19	100,00	100,00	=	-1	-1	1	-1
M1	700,00	650,00	<	-1	-1	0	-2
M5	700,00	90,00	<	-1	-1	0	-2
M13	500,00	Zero	<	1	-3	0	-2
M17	180,00	50,00	<	0	-2	0	-2
M20	100,00	90,00	<	-1	-1	0	-2
M6	700,00	50,00	<	-1	-2	0	-3
M18	180,00	Zero	<	0	-3	0	-3
M21	100,00	50,00	<	-1	-2	0	-3
M23	60,00	60,00	=	-2	-2	1	-3
M7	700,00	Zero	<	-1	-3	0	-4
M22	100,00	Zero	<	-1	-3	0	-4
M24	60,00	50,00	<	-2	-2	0	-4
M25	60,00	Zero	<	-2	-3	0	-5
M26	Zero	Zero	=	-3	-3	1	-5

Atribuindo a pontuação P01 para Lmax.

```
#ATRIBUINTE A PONTUAÇÃO 01 - LMAX
lista_P01 = []#Criando a lista para receber os valores p01
Lmax = DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_MAX']
for ip in Lmax:# Comparando os valores com as faixas definidas e atribuindo a pontuação correspondente
    if (ip > 200):
        if(ip > 600):
            p01 = -1
            lista_P01.append(p01)
        else:
            p01 = 1
            lista_P01.append(p01)
    elif (ip < 110):
        if (ip < 80):
            if (ip == 0):
                p01 = -3
                lista_P01.append(p01)
            else:
                p01 = -2
                lista_P01.append(p01)
        else:
            p01 = -1
            lista_P01.append(p01)
    else:
        p01 = 0
        lista_P01.append(p01)
DIREITO_DE_USO_AB_HUM_MAX['P01_LMAX'] = lista_P01
```

DIREITO_DE_USO_AB_HUM_MAX							
'OL_MAX_VOL_2020	POP_ESTIMADA_2020	POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	
0.000000	11853	11155	32.252883	30.356156	475897.95	-2	
-1.000000	15036	11563	28.660094	0.000000	603695.40	-2	
-0.614987	63104	55900	99.019230	33.771552	2533625.60	-1	
-0.010559	54481	46350	77.952922	65.619207	2187412.15	-2	
-1.000000	17493	15812	137.373136	0.000000	702343.95	0	
...	
-1.000000	22040	18375	452.864807	0.000000	884906.00	1	
0.000000	13915	13636	61.325685	60.099171	558687.25	-2	
-1.000000	18471	17037	665.891378	0.000000	741610.65	-1	
-0.613179	40903	34798	78.625664	25.875177	1642255.45	-2	
-0.665787	61410	54399	7.862035	2.327634	2465611.50	-2	

Atribuindo a pontuação P03 para L2020.

```
#ATRIBUINHO A PONTUAÇÃO 03 - L2020
lista_P03 = [] #Criando a lista para receber os valores p03
L2020 = DIREITO_DE_USO_AB_HUM_MAX['L_HAB_DIA_ANO_VOL_2020']
for ipp in L2020:# Comparando os valores com as faixas definidas e atribuindo a pontuação correspondente
    if (ipp > 200):
        if(ipp > 600):
            p03 = -1
            lista_P03.append(p03)
        else:
            p03 = 1
            lista_P03.append(p03)
    elif (ipp < 110):
        if (ipp < 80):
            if (ipp == 0):
                p03 = -3
                lista_P03.append(p03)
            else:
                p03 = -2
                lista_P03.append(p03)
        else:
            p03 = -1
            lista_P03.append(p03)
    else:
        p03 = 0
        lista_P03.append(p03)
DIREITO_DE_USO_AB_HUM_MAX['P03_2020'] = lista_P03
```

DIREITO_DE_USO_AB_HUM_MAX								
OL_2020	POP_ESTIMADA_2020	POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	
0.000000	11853	11155	32.252883	30.356156	475987.95	-2	-2	
1.000000	15036	11563	28.660094	0.000000	603695.40	-2	-3	
0.614987	63104	55900	99.019230	33.771552	2533625.60	-1	-2	
0.010559	54481	46350	77.952922	65.619207	2187412.15	-2	-2	
1.000000	17493	15812	137.373136	0.000000	702343.95	0	-3	
...	
1.000000	22040	18375	452.864807	0.000000	884906.00	1	-3	
0.000000	13915	13636	61.325685	60.099171	558687.25	-2	-2	
1.000000	18471	17037	665.891378	0.000000	741610.65	-1	-3	
0.613179	40903	34798	78.625664	25.875177	1642255.45	-2	-2	
0.665787	61410	54399	7.862035	2.327634	2465611.50	-2	-2	

Atribuindo a pontuação P02 para comparação entre L2020_LMAX

```

: #ATRIBUINDO A PONTUAÇÃO P02 - L2020 -LMAX
LMAX_L2020 = lmax - L2020
lista_P02 = []#Criando a Lista para receber os valores p02
for ipp in LMAX_L2020:# Comparando os valores com as faixas definidas e atribuindo a pontuação correspondente
    if ipp == 0:
        p02 = 1
        lista_P02.append(p02)
    else:
        p02 = 0
        lista_P02.append(p02)
DIREITO_DE_USO_AB_HUM_MAX['P02_L2020_LMAX'] = lista_P02

```

DIREITO_DE_USO_AB_HUM_MAX							
TIMADA_2020	POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	P02_L2020_LMAX
11853	11155	32.252883	30.356156	475897.95	-2	-2	0
15036	11563	28.660094	0.000000	603695.40	-2	-3	0
63104	55900	99.019230	33.771552	2533625.60	-1	-2	0
54481	46350	77.952922	65.619207	2187412.15	-2	-2	0
17493	15812	137.373136	0.000000	702343.95	0	-3	0
...
22040	18375	452.864807	0.000000	884906.00	1	-3	0
13915	13636	61.325685	60.099171	558687.25	-2	-2	0
18471	17037	665.891378	0.000000	741610.65	-1	-3	0
40903	34798	78.625664	25.875177	1642255.45	-2	-2	0
61410	54399	7.862035	2.327634	2465611.50	-2	-2	0

Totalizando a pontuação para abastecimento humano.

#TOTALIZANDO A PONTUAÇÃO PARA ABASTECIMENTO HUMANO							
DIREITO_DE_USO_AB_HUM_MAX							
POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	P02_L2020_LMAX	PT_AB_HUM
11155	32.252883	30.356156	475897.95	-2	-2	0	-4
11563	28.660094	0.000000	603695.40	-2	-3	0	-5
55900	99.019230	33.771552	2533625.60	-1	-2	0	-3
46350	77.952922	65.619207	2187412.15	-2	-2	0	-4
15812	137.373136	0.000000	702343.95	0	-3	0	-3
...
18375	452.864807	0.000000	884906.00	1	-3	0	-2
13636	61.325685	60.099171	558687.25	-2	-2	0	-4
17037	665.891378	0.000000	741610.65	-1	-3	0	-4
34798	78.625664	25.875177	1642255.45	-2	-2	0	-4
54399	7.862035	2.327634	2465611.50	-2	-2	0	-4

Atribuindo graduação aos valores a pontuação total obtida

```
#####
#ATRIBUINDO GRADUAÇÃO AOS VALORES A PONTUAÇÃO TOTAL OBTIDA
lista_nota_ab_hum= [] #Criando a Lista para receber os valores ntab
nota_ab_hum = DIREITO_DE_USO_AB_HUM_MAX['PT_AB_HUM']
for ipn in nota_ab_hum:
    if (ipn == -5):
        ntab = 10
        lista_nota_ab_hum.append(ntab)
    if (ipn == -4):
        ntab = 9
        lista_nota_ab_hum.append(ntab)
    if (ipn == -3):
        ntab = 8
        lista_nota_ab_hum.append(ntab)
    if (ipn == -2):
        ntab = 7
        lista_nota_ab_hum.append(ntab)
    if (ipn == -1):
        ntab = 6
        lista_nota_ab_hum.append(ntab)
    if (ipn == 0):
        ntab = 5
        lista_nota_ab_hum.append(ntab)
    if (ipn == 1):
        ntab = 4
        lista_nota_ab_hum.append(ntab)
    if (ipn == 2):
        ntab = 3
        lista_nota_ab_hum.append(ntab)
    if (ipn == 3):
        ntab = 2
        lista_nota_ab_hum.append(ntab)
    if (ipn == 4):
        ntab = 1
        lista_nota_ab_hum.append(ntab)
DIREITO_DE_USO_AB_HUM_MAX['REG_AB_HUM'] = lista_nota_ab_hum
```

DIREITO_DE_USO_AB_HUM_MAX									
VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	P02_L2020_LMAX	PT_AB_HUM	REG_AB_HUM	
11155	32.252883	30.358168	475897.95	-2	-2	0	-4	9	
11583	28.660094	0.000000	603695.40	-2	-3	0	-5	10	
55900	99.019230	33.771552	2633625.60	-1	-2	0	-3	8	
46350	77.952922	65.619207	2187412.15	-2	-2	0	-4	9	
15812	137.373138	0.000000	702343.95	0	-3	0	-3	8	
...	
18375	452.864807	0.000000	884906.00	1	-3	0	-2	7	
13836	81.325685	60.099171	558887.25	-2	-2	0	-4	9	
17037	665.891378	0.000000	741610.65	-1	-3	0	-4	9	
38233	11.803720	3.494609	1642265.45	-2	-2	0	-4	9	
52245	52.369737	17.234528	2466611.50	-2	-2	0	-4	9	

Inserindo o código IBGE do município para posterior uso na visualização.

#INSERINDO O CÓDIGO IBGE DO MUNICÍPIO								
lista_CD_IBGE = list(IBGE_IPECE['CD_IBGE'])#Gerando uma lista a partir de uma coluna do df 'CD_IBGE' DIREITO_DE_USO_AB_HUM_MAX.insert(1,'CD_IBGE',lista_CD_IBGE,True)#Inserindo uma lista no df 'DIREITO_DE_USO_AB_HUM_MAX'								
DIREITO_DE_USO_AB_HUM_MAX								
MUNICIPIO	CD_IBGE	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MAX_MUN_M3	VOLUME_MUN_ABHU	
25	ABAIARA	2300101	Abastecimento Humano	2017	0.000000e+00	1.313312e+05	1.313312e+05	1.3
421	ACARAPE	2300150	Abastecimento Humano	2007	1.209600e+05	0.000000e+00	1.209600e+05	0.0
851	ACARAU	2300200	Abastecimento Humano	2014	6.115200e+05	1.408826e+06	2.020346e+06	7.7
1263	ACOPIARA	2300309	Abastecimento Humano	2012	1.313424e+06	5.376000e+03	1.318800e+06	1.3
1685	AIUABA	2300408	Abastecimento Humano	2015	7.924089e+05	4.380000e+02	7.928489e+05	0.0
...
74477	URUBURETAMA	2313807	Abastecimento Humano	2011	3.037440e+06	0.000000e+00	3.037440e+06	0.0
74909	URUOCA	2313906	Abastecimento Humano	2019	0.000000e+00	3.052422e+05	3.052422e+05	3.0
75319	VARJOTA	2313955	Abastecimento Humano	2016	4.140884e+06	0.000000e+00	4.140884e+06	0.0
75727	VARZEA ALEGRE	2314003	Abastecimento Humano	2012	9.744000e+05	2.426760e+04	9.986676e+05	3.8
76147	VICOSA DO CEARA	2314102	Abastecimento Humano	2014	3.763200e+04	1.184754e+05	1.561074e+05	5.2
184 rows x 19 columns								

Reinserindo nomes dos municípios sem as alterações a partir de lista reservada previamente para o relacionamento entre tabelas em outra ferramenta.

#REINSERINDO NOMES DOS MUNICÍPIOS SEM AS ALTERAÇÕES A PARTIR DE LISTA RESERVADA PREVIAMENTE								
DIREITO_DE_USO_AB_HUM_MAX.insert(0,'Nome_Municipio',lista_nome_municipio ,True)#Inserindo uma lista no df 'DIREITO_DE_USO_AB_HUM'								
DIREITO_DE_USO_AB_HUM_MAX								
Nome_Municipio	MUNICIPIO	CD_IBGE	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MAX_MUN_M3	VOLUM
25	Abaiara	ABAIARA	2300101	Abastecimento Humano	2017	0.000000e+00	1.313312e+05	1.313312e+05
421	Acarape	ACARAPE	2300150	Abastecimento Humano	2007	1.209600e+05	0.000000e+00	1.209600e+05
851	Acaraú	ACARAÚ	2300200	Abastecimento Humano	2014	6.115200e+05	1.408826e+06	2.020346e+06
1263	Acopiara	ACOPIARA	2300309	Abastecimento Humano	2012	1.313424e+06	5.376000e+03	1.318800e+06
1685	Aluaba	AIUABA	2300408	Abastecimento Humano	2015	7.924089e+05	4.380000e+02	7.928489e+05
...
74477	Uruburetama	URUBURETAMA	2313807	Abastecimento Humano	2011	3.037440e+06	0.000000e+00	3.037440e+06
74909	Uruoca	URUOCA	2313906	Abastecimento Humano	2019	0.000000e+00	3.052422e+05	3.052422e+05
75319	Varjota	VARJOTA	2313955	Abastecimento Humano	2016	4.140884e+06	0.000000e+00	4.140884e+06
75731	Várzea Alegre	VICOSA DO CEARA	2314003	Abastecimento Humano	2014	3.763200e+04	1.184754e+05	1.561074e+05
76143	Vicoso do Ceará	VARZEA ALEGRE	2314102	Abastecimento Humano	2012	9.744000e+05	2.426760e+04	9.986676e+05
184 rows x 20 columns								

Exportando o dataframe para o formato Excel.

```
DIREITO_DE_USO_AB_HUM_MAX.to_excel('DIREITO_DE_USO_AB_HUM_MAX_.xlsx',encoding='utf-8',index=False)
```

Essa análise ficará mais compreensível se pudermos observar os valores expostos em mapas geográficos, pois é importante ver o que acontece nos municípios vizinhos também.

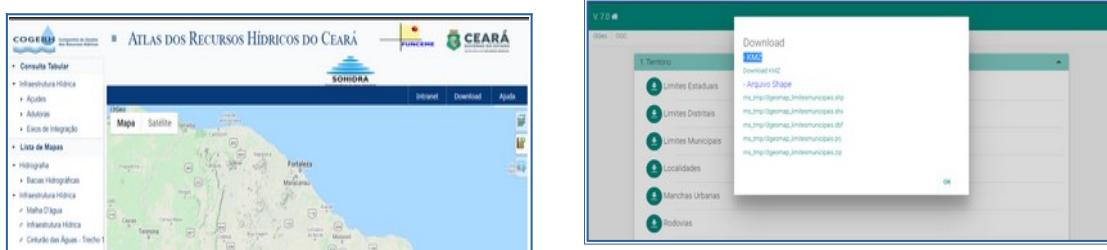
Iniciando alguns preparativos para visualização.

Inserindo planilha Excel com dados geográficos coordenadas decimais dos municípios do Brasil com na ferramenta PowerBI.

Alterando os tipos de dados que a ferramenta não reconheceu.

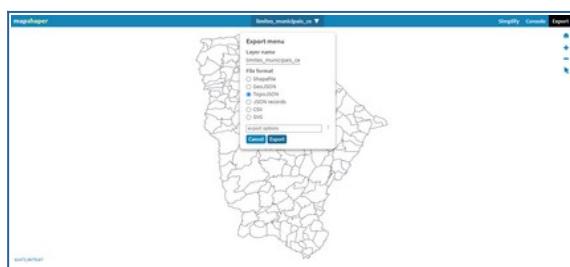
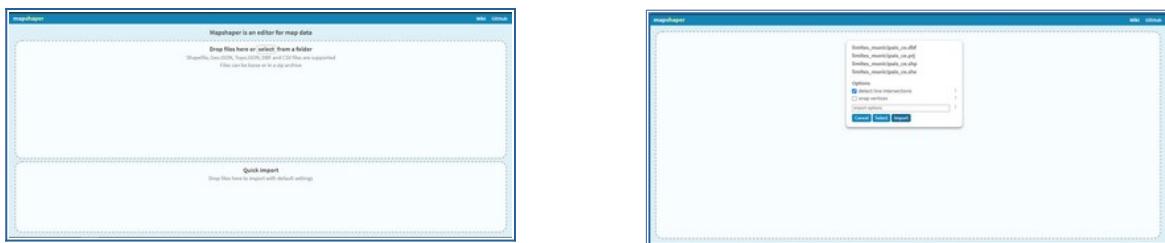
Campo	Formato	Propriedades
Nome Municipio	Formato: Texto	Não resumir
cod_latitude	Formato: Número decimal	Soma
cod_longitude	Formato: Número decimal	Soma

O arquivo com o mapa com a delimitação mais recente dos municípios do Ceará definida pelo IPECE em 2019 pode ser obtido no site da COGERH, em 02 formatos.

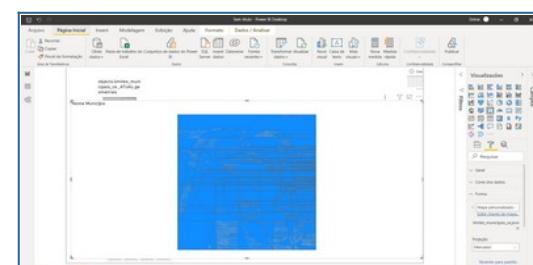
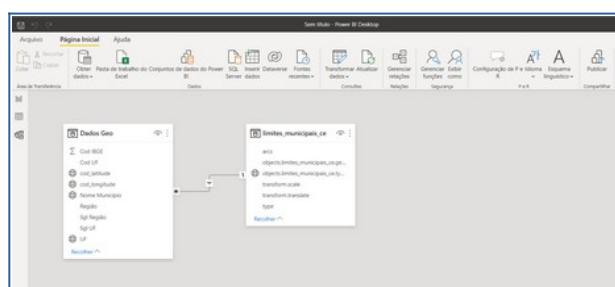
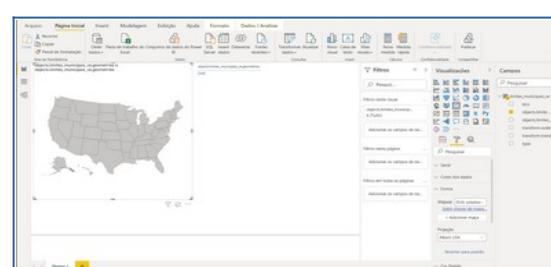
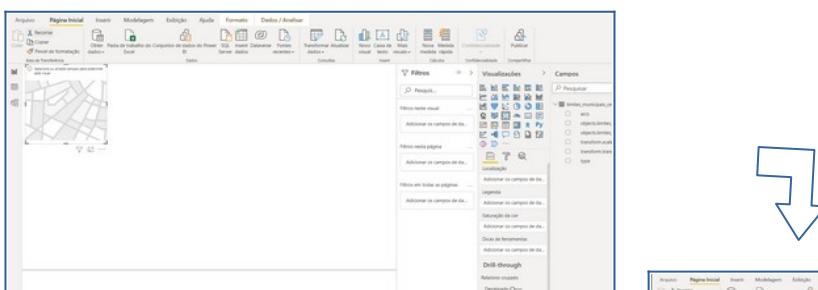
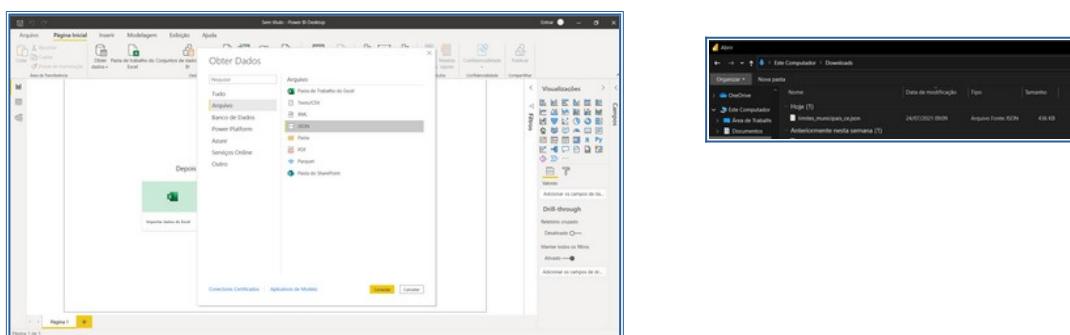
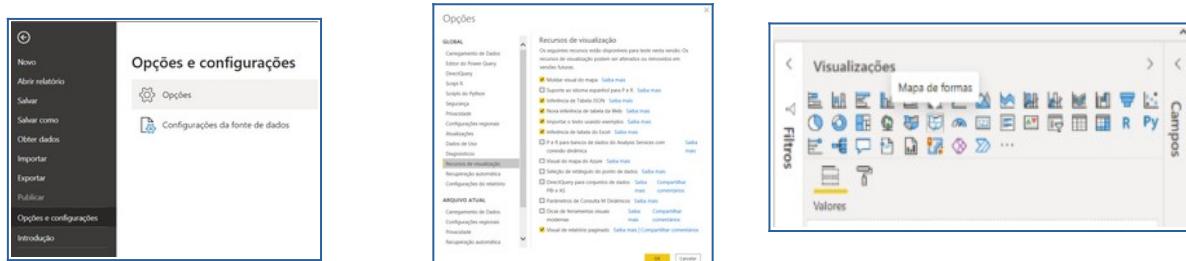


A conversão de .kmz para .topojson se mostrou mais rápida ao ser realizada online do arquivo com o mapa dos municípios para .topojson, onde são exigidos os arquivos .dbf, .pri, .shp e .skx no conversor mapshaper.org. Também exigiu menos ajustes ao inserir no PowerBI.

Convertendo através do aplicativo Mapshaper:



Na ferramenta PowerBI com o arquivo



Convertendo .kmz em '.t.json' no aplicativo MyGeodata Convert, lembrando que é gratuito até 5MB mensais.

The diagram illustrates a step-by-step process for converting a KMZ file to TopoJSON using the MyGeodata Converter online service. It consists of several screenshots and arrows indicating the flow:

- Step 1: Upload KMZ File**
A screenshot of the "Convert KMZ to TopoJSON Online" page. It shows a "Drag & Drop files here..." input field and a "Or Browse files to convert!" button. A note at the bottom states: "Please note that your data will not be shared to anybody without your intervention."
- Step 2: Select File**
A screenshot of a Windows file explorer window showing a folder named "limites_municipais_ce_ATUAL.kmz". An arrow points from this screen to the "Select From MyGeodata Drive" button on the converter's upload interface.
- Step 3: Input Data Selection**
A screenshot of the "MyGeodata Converter" interface under "1. Input Data". It shows the selected layer "limites_municipais_ce_ATUAL" and its details: 1 dataset, 1.8 MB volume, and coordinate system WGS 84 (EPSG:4326). The "Output Format" is set to "TopoJSON".
- Step 4: Conversion Preview**
A screenshot of the "MyGeodata Converter" interface under "3. Conversion". It displays a map of the state of Ceará, Brazil, with a red box highlighting the study area. Buttons for "Show in Map" and "Convert now!" are visible.
- Step 5: Conversion Result**
A screenshot of the "Conversion Result" page. It lists the converted data: "Data format: TopoJSON", "Output coordinate system: WGS 84", "File size: 456.640 bytes", and "Total size: 456.640 bytes". A "Download" button is present.
- Step 6: ZIP Archive**
A screenshot of a Windows File Explorer window showing a ZIP archive named "mygeodata (1).zip (evaluation copy)". The contents are listed as follows:

Name	Size	Packed	Type	Modified	CRC32
limites_municipais_ce_A...	456.640	157.901	Arquivo TOPOJSON	24/07/2021 15...	AA835762
topojson.log	0	0	Documento de Tex...	24/07/2021 15...	00000000

No PowerBI com arquivo '.topojoson'

Screenshot of Power BI Desktop showing the 'Obter Dados' (Get Data) screen. A map of Brazil is displayed on the left, and the 'Arquivo' (File) section is selected in the center. A file named 'Limites_municipais.topojson' is highlighted in the list.

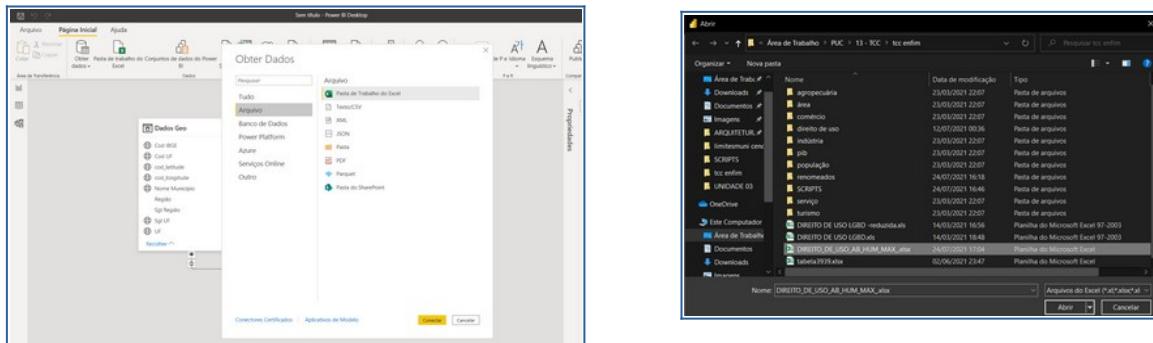
Screenshot of the Windows File Explorer showing the contents of the 'Área de Trabalho' (Workplace). It lists several files including 'Limites_municipais.topojson' and other shapefile extensions like 'shp', 'dbf', and 'prj'. The file 'Limites_municipais.topojson' is selected.

Screenshot of Power BI Desktop showing the 'Visualizações' (Visualizations) screen. A map of the United States is displayed, and the 'Campos' (Fields) pane on the right shows fields such as 'área', 'name', and 'type'. A tooltip for 'área' indicates it is a 'Geometrica' field.

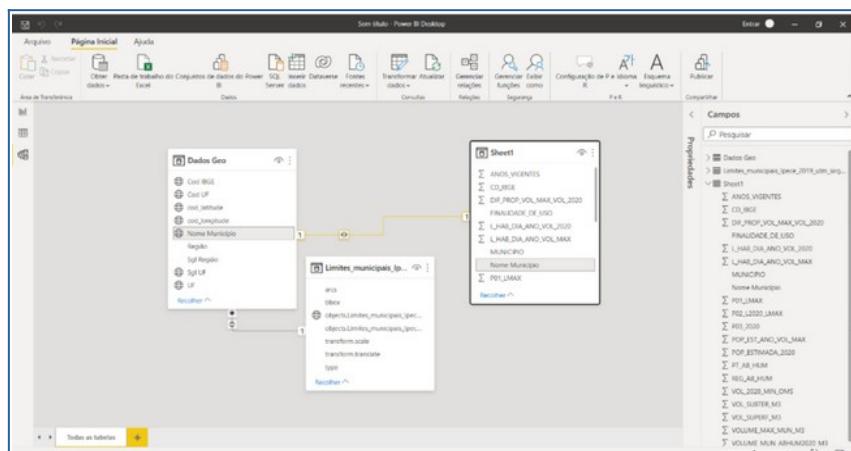
Screenshot of Power BI Desktop showing the 'Dados / Analisar' (Data / Analyse) screen. A map of Brazil is displayed, and the 'Campos' (Fields) pane on the right shows fields such as 'área', 'name', and 'type'. A tooltip for 'área' indicates it is a 'Geometrica' field.

Screenshot of Power BI Desktop showing the 'Visualizações' (Visualizations) screen. A map of Brazil is displayed, and the 'Campos' (Fields) pane on the right shows fields such as 'área', 'name', and 'type'. A tooltip for 'área' indicates it is a 'Geometrica' field.

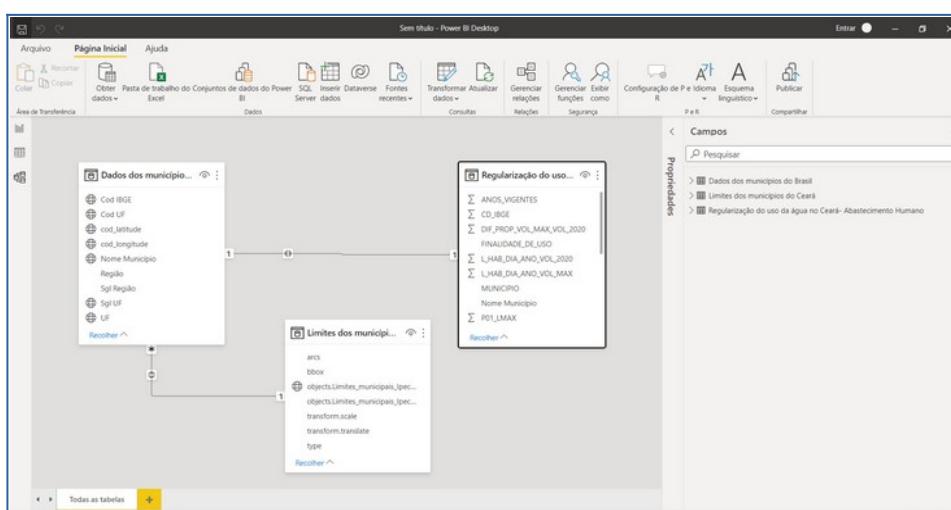
Inserindo o arquivo com os dados sobre a regularização do uso d'água na finalidade abastecimento humano com as notas obtidas na ferramenta PowerBI.



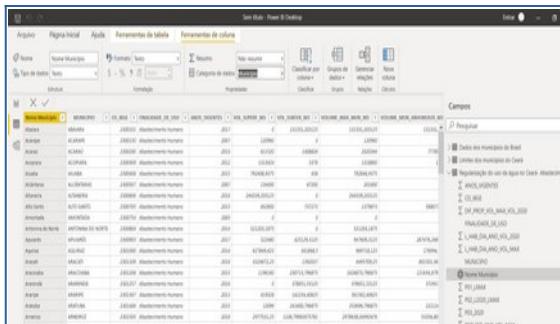
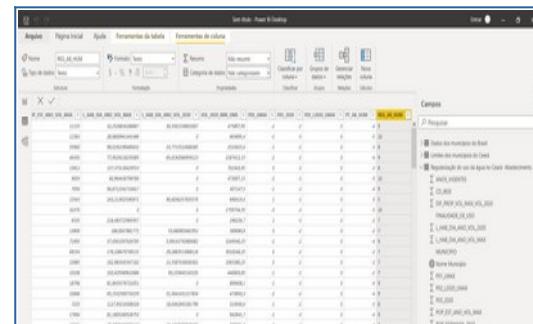
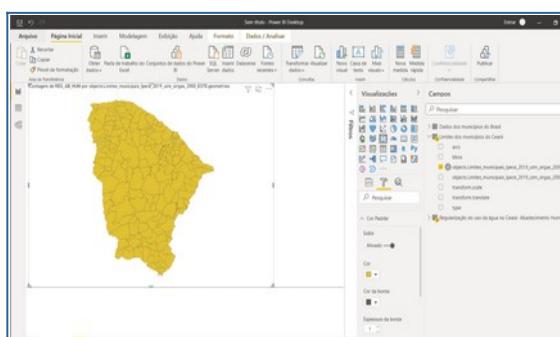
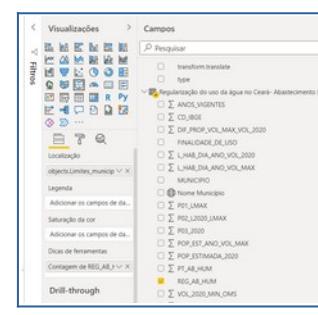
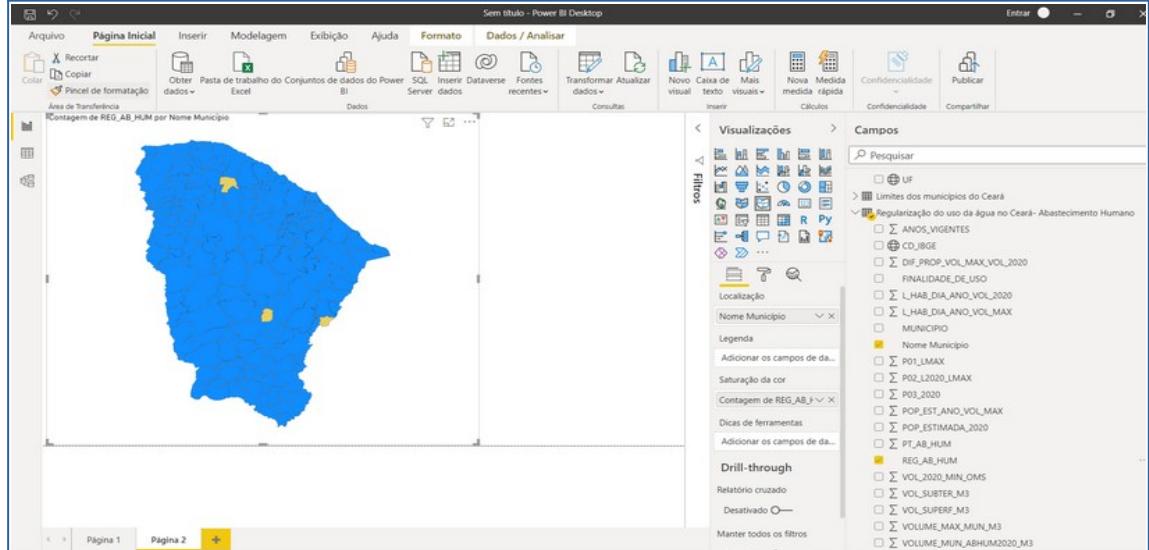
A coluna ‘Nome Município’ reinserida no dataframe foi reconhecida em um relacionamento com o arquivo de dados dos municípios do Brasil .



Renomeando as tabelas



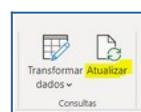
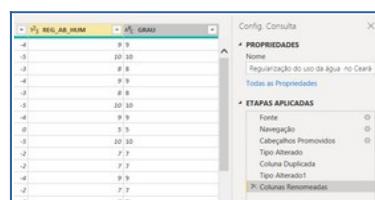
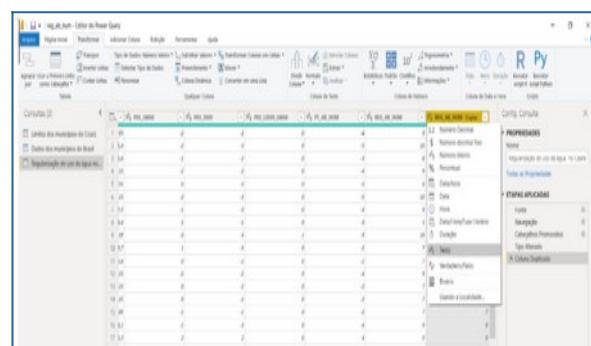
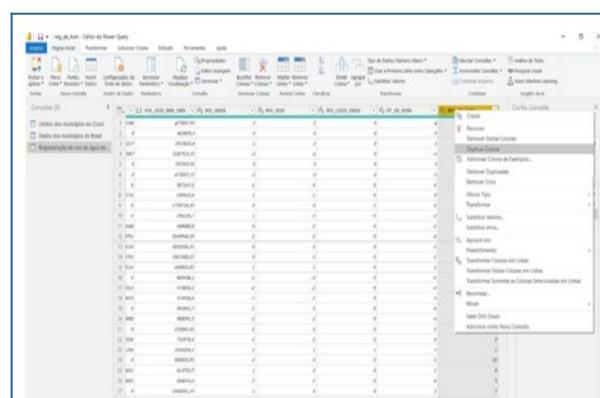
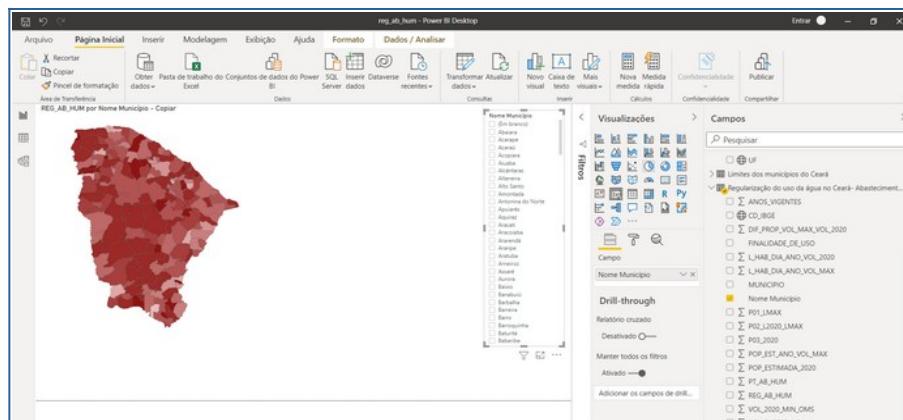
Alterando os tipos de dados

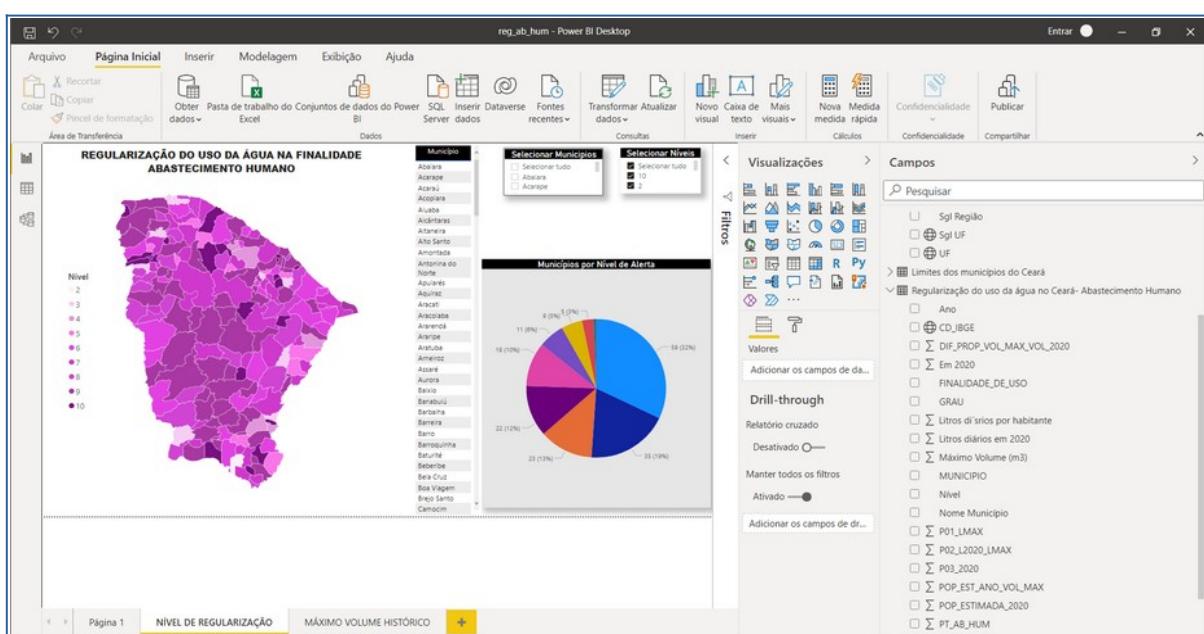
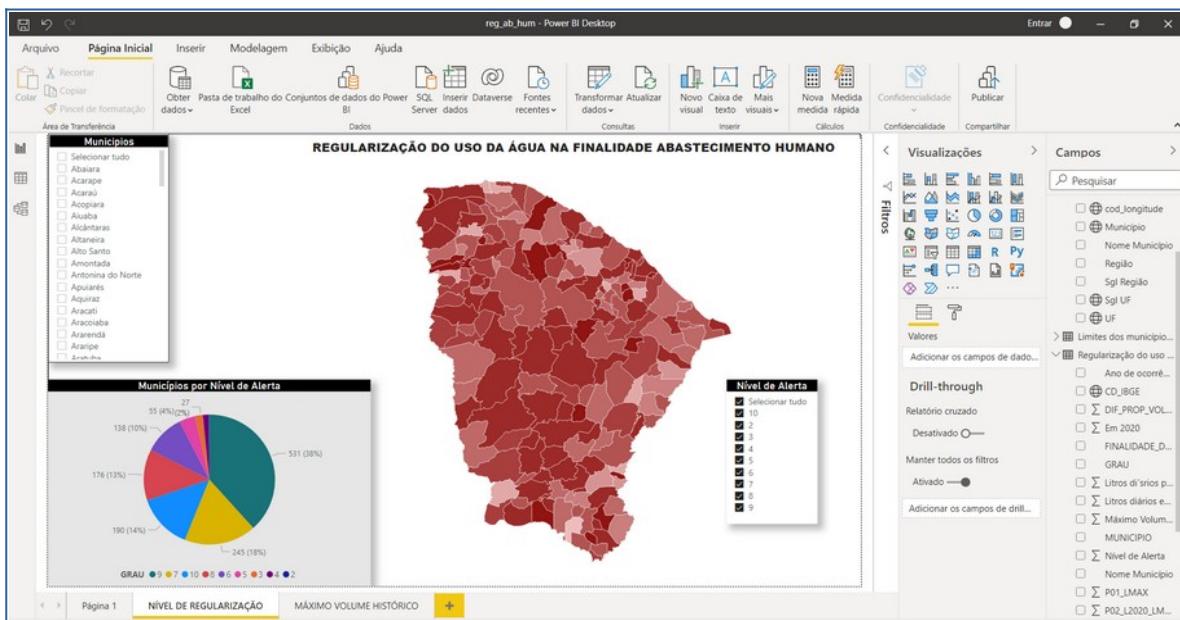






Alguns municípios não tiveram seus dados vinculados ao mapa

Considerando os relacionamentos e a quantidade de municípios a ajustar a tabela 'Dados dos municípios' foi eleita a melhor para absorver esse ajuste privilegiando a tabela 'Limites dos municípios'

Configurando a visualização.





Ajustando a escala de níveis e as cores e outras formatações na página “NÍVEL DAS REGULARIZAÇÕES”.

COLUNAS A REMOVER IX	
INT_QT_VAZAO MAXIMA	INT_QT_VAZAO MEDIA

A página ‘NÍVEL DE REGULARIZAÇÃO’ traz os dados referentes aos níveis de carência de regularização em cada município do Ceará, conforme critérios desenvolvidos em *insights* adotados durante a exploração de dados e contato com a área de negócios:



Mapa com o nível em que se encontra cada município em 2020



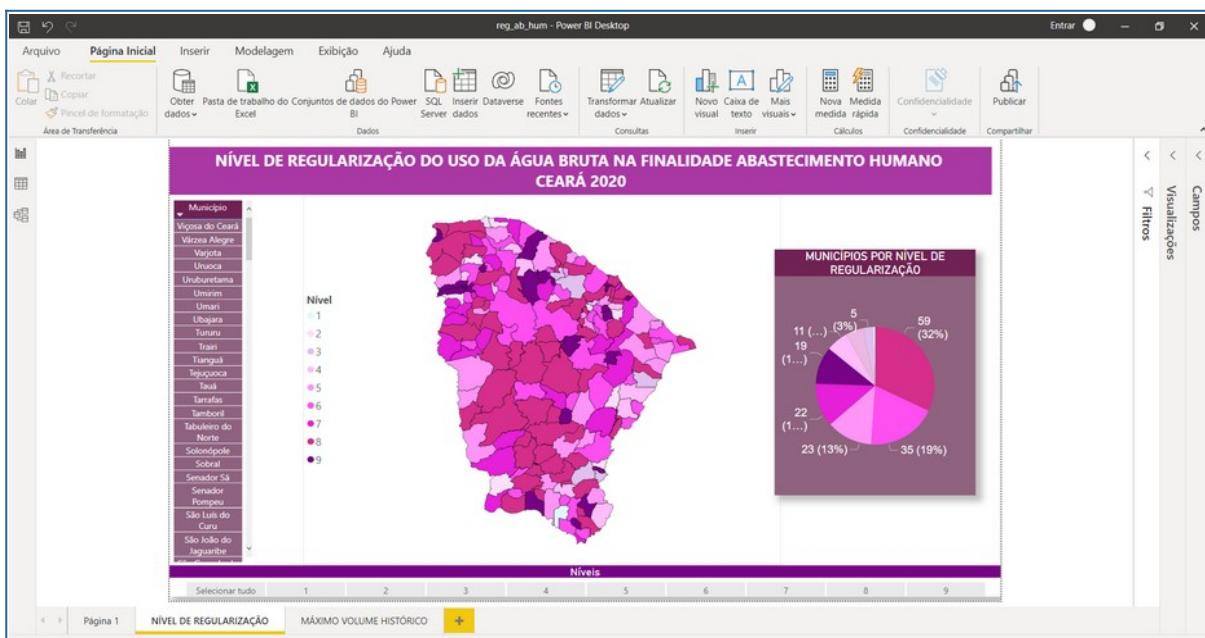
Gráfico de pizza com o número de municípios em cada nível e a distribuição percentual.



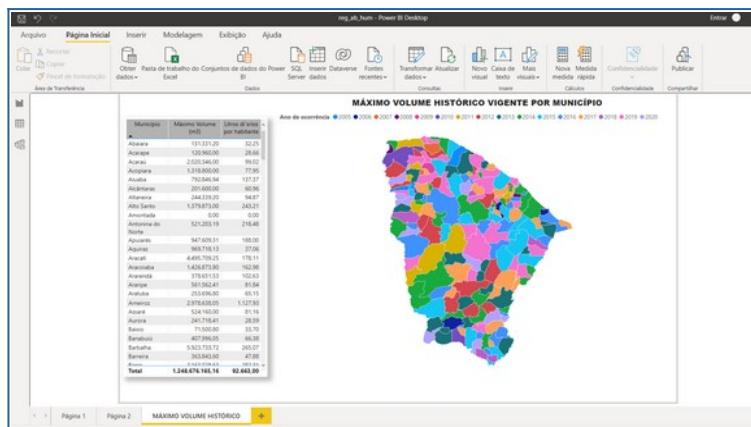
Tabela com a lista de municípios conforme o nível selecionado.



Segmentador de dados para seleção dos níveis.

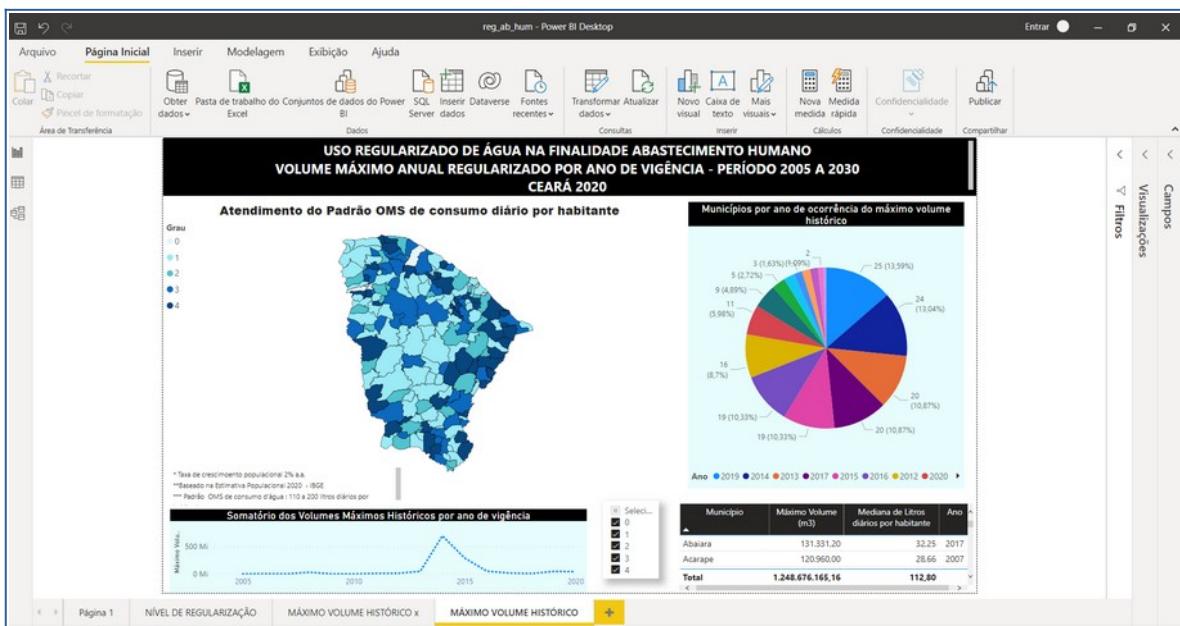


Preparando a visualização dos dados referentes aos volumes máximos vigentes para consultas.



A página ‘MÁXIMO VOLUME HISTÓRICO’ traz os dados referentes aos volumes máximos históricos vigentes em cada município:

- Mapa com grau de atendimento do consumo padrão de água estipulado pela OMS no ano de ocorrência do volume máximo histórico vigente em cada município do Ceará.
- Gráfico de pizza com o número de municípios com ocorrência de volumes máximos vigentes em cada ano e a distribuição percentual.
- Tabela com o valor do volume máximo, a quantidade de litros por habitante dia no referido ano e ano de ocorrência para cada município.
- Gráfico de linha com somatório de volumes máximos ocorridos em cada ano.
- Segmentador de dados para seleção dos níveis.



Esse foi um pontapé inicial na visualização dos dados. Esse *dashboard* teste foi para ter uma imagem mental de como poderão ser apresentados os resultados, além de manipular um pouco a ferramenta.

Observados os dados referentes a finalidade de uso no abastecimento humano, volta-se ao dataframe sobre dados de uso onde constam todas as finalidades de uso para seguir na observação dos dados das demais finalidades de uso e decidir sobre seu uso ou não nessa análise.

Esse dataframe foi obtido após estender os usos de cada ponto de captação pelos anos de sua vigência, em seguida totalizar ano e finalidade de uso para cada município, isso resultou em 38272 linhas e 6 colunas. Esses registros englobam os anos de vigência do uso regular da água de 2005 a 2030 nos 184 municípios do estado por cada finalidade de uso.

É possível que um determinado município não apresente uso regularizado no ano e finalidade de interesse, isso não significa necessariamente deficit na regularização do uso de água, embora mereça uma posterior verificação. Essa situação é mais flexível nas demais finalidades de uso do que no uso para o abastecimento humano dada a natureza da atividade essencial à vida humana.

DIREITO_DE_USO						
	MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3
1	ABAIRARA	Abastecimento Humano	2005	0.0	0.0	0.0
3	ABAIRARA	Abastecimento Humano	2006	0.0	0.0	0.0
5	ABAIRARA	Abastecimento Humano	2007	0.0	0.0	0.0
7	ABAIRARA	Abastecimento Humano	2008	0.0	0.0	0.0
9	ABAIRARA	Abastecimento Humano	2009	0.0	0.0	0.0
...
76535	VICOSA DO CEARA	Serviço e Comércio	2026	0.0	0.0	0.0
76537	VICOSA DO CEARA	Serviço e Comércio	2027	0.0	0.0	0.0
76539	VICOSA DO CEARA	Serviço e Comércio	2028	0.0	0.0	0.0
76541	VICOSA DO CEARA	Serviço e Comércio	2029	0.0	0.0	0.0
76543	VICOSA DO CEARA	Serviço e Comércio	2030	0.0	0.0	0.0

38272 rows × 6 columns

```
#OBSERVANDO AS FINALIDADES DE USO PRESENTES NO DF 'DIREITO_DE_USO'
FINS = set(DIREITO_DE_USO['FINALIDADE_DE_USO'])
FINS
```

```
{'Abastecimento Humano',
'Aquicultura em Tanque Escavado',
'Criação Animal',
'Indústria',
'Irrigação',
'Mineração',
'Outros usos',
'Serviço e Comércio'}
```

Na finalidade de uso ‘Mineração’, a área de negócio informou que, na prática, a classificação dessa atividade já não prevalece na categoria ‘uso’, tendo apenas alguns poucos registros antigos.

Separando os registros por finalidade de uso, obteve-se 6 tabelas com 4784 linhas e 6 colunas cada.

# OBTENDO O DATAFRAME DE CADA FINALIDADE DE USO DE INTERESSE NA ANÁLISE, EXCETO MINERAÇÃO						
DIREITO_DE_USO_IRRIC						
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3	
209	ABAIARA	Irrigação	2005	0.0	0.0	0.0
211	ABAIARA	Irrigação	2006	0.0	0.0	0.0
213	ABAIARA	Irrigação	2007	0.0	0.0	0.0
215	ABAIARA	Irrigação	2008	0.0	0.0	0.0
217	ABAIARA	Irrigação	2009	0.0	0.0	0.0
...
76379	VICOSA DO CEARA	Irrigação	2026	0.0	0.0	0.0
76381	VICOSA DO CEARA	Irrigação	2027	0.0	0.0	0.0
76383	VICOSA DO CEARA	Irrigação	2028	0.0	0.0	0.0
76385	VICOSA DO CEARA	Irrigação	2029	0.0	0.0	0.0
76387	VICOSA DO CEARA	Irrigação	2030	0.0	0.0	0.0

4784 rows x 6 columns

Na exploração dos dados das demais finalidades de uso foi aplicada a mesma metodologia para obtenção dos volumes máximos vigentes e em 2020 para posterior classificação semelhante a realizada para a finalidade abastecimento humano. Os parâmetros de referência, entretanto, serão diferentes.

A finalidade ‘serviço e comércio’ é recente e surgiu de uma reclassificação desse tipo de uso, conforme informado pela área de negócio, ela absorveu um grupo que vinha se destacando dentro da categoria genérica ‘outros usos’ além de extinta categoria ‘turismo e lazer’. Em resumo, para efeito dessa análise, foram unificadas ambas categorias.



Um município pode ter diversos anos com o mesmo volume máximo, então, em caso de empate no valor de máximo volume vigente, o ano retornado como o de ocorrência do volume máximo é o mais antigo. Então se, por exemplo, o volume máximo for zero, o ano de ocorrência representa desde o qual não há registro nessa finalidade. Exemplo: Abaiara – 2005.

DIREITO_DE_USO_OUTROS.head(50)						
MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3	
313	ABAIIARA	Outros usos	2005	0.0	0.000000	0.000000
315	ABAIIARA	Outros usos	2006	0.0	0.000000	0.000000
317	ABAIIARA	Outros usos	2007	0.0	0.000000	0.000000
319	ABAIIARA	Outros usos	2008	0.0	0.000000	0.000000
321	ABAIIARA	Outros usos	2009	0.0	0.000000	0.000000
323	ABAIIARA	Outros usos	2010	0.0	0.000000	0.000000
325	ABAIIARA	Outros usos	2011	0.0	0.000000	0.000000
327	ABAIIARA	Outros usos	2012	0.0	0.000000	0.000000
329	ABAIIARA	Outros usos	2013	0.0	0.000000	0.000000
331	ABAIIARA	Outros usos	2014	0.0	0.000000	0.000000
333	ABAIIARA	Outros usos	2015	0.0	0.000000	0.000000
335	ABAIIARA	Outros usos	2016	0.0	0.000000	0.000000
337	ABAIIARA	Outros usos	2017	0.0	0.000000	0.000000
339	ABAIIARA	Outros usos	2018	0.0	0.000000	0.000000
341	ABAIIARA	Outros usos	2019	0.0	0.000000	0.000000
343	ABAIIARA	Outros usos	2020	0.0	0.000000	0.000000
345	ABAIIARA	Outros usos	2021	0.0	0.000000	0.000000
347	ABAIIARA	Outros usos	2022	0.0	0.000000	0.000000
349	ABAIIARA	Outros usos	2023	0.0	0.000000	0.000000
351	ABAIIARA	Outros usos	2024	0.0	0.000000	0.000000
353	ABAIIARA	Outros usos	2025	0.0	0.000000	0.000000
355	ABAIIARA	Outros usos	2026	0.0	0.000000	0.000000
357	ABAIIARA	Outros usos	2027	0.0	0.000000	0.000000
359	ABAIIARA	Outros usos	2028	0.0	0.000000	0.000000
361	ABAIIARA	Outros usos	2029	0.0	0.000000	0.000000
363	ABAIIARA	Outros usos	2030	0.0	0.000000	0.000000
729	ACARAPE	Outros usos	2005	0.0	0.000000	0.000000

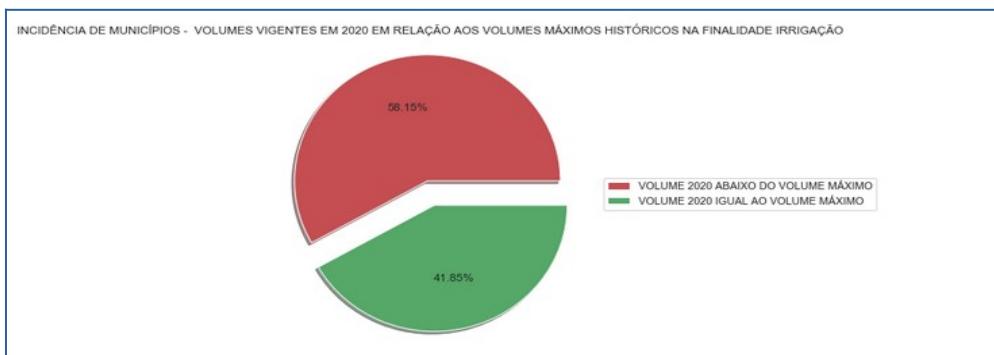
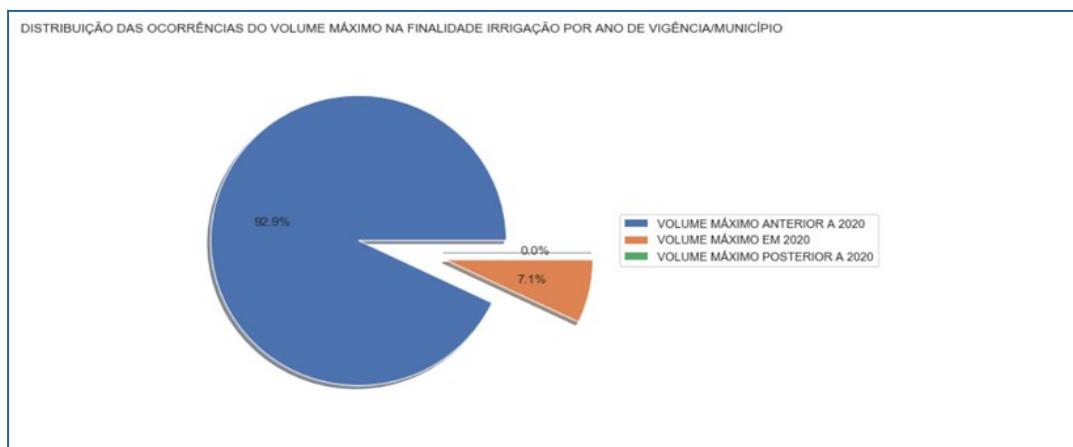
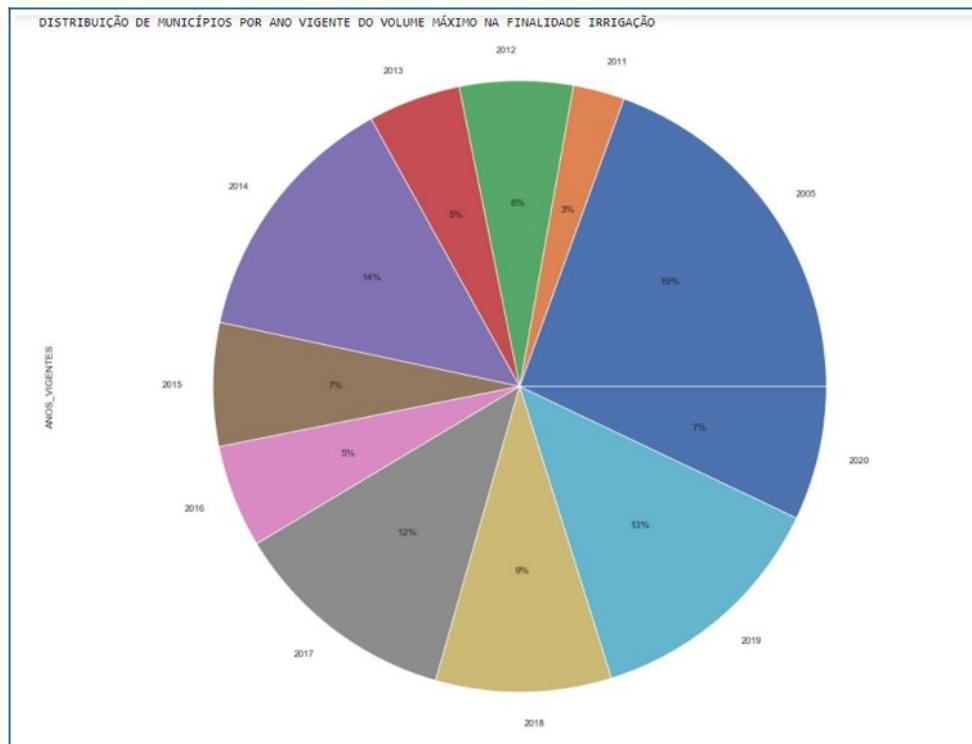
DIREITO_DE_USO_OUTROS_MAX						
MUNICIPIO	ANO_VOL_MAX_VIGENTE_OUTROS	VOL_SUPERF_OUTROS_M3	VOL_SUBTER_OUTROS_M3	VOLUME_MAX_MUN_OUTROS_M3	VOLUME_	
313	ABAIIARA	2005	0.0	0.000000	0.000000	
755	ACARAPE	2018	0.0	8271.360352	8271.360352	
1175	ACARAU	2020	0.0	19056.380859	19056.380859	
1587	ACOPIARA	2018	0.0	668.159973	668.159973	
1995	AIUABA	2014	0.0	584.000000	584.000000	
...	
74777	URUBURETAMA	2005	0.0	0.000000	0.000000	
75213	URUOCA	2015	0.0	12137.280273	12137.280273	
75637	VARJOTA	2019	0.0	16128.000000	16128.000000	
76051	VARZEA ALEGRE	2018	0.0	30897.000000	30897.000000	
76441	VICOSA DO CEARA	2005	0.0	0.000000	0.000000	

184 rows x 7 columns

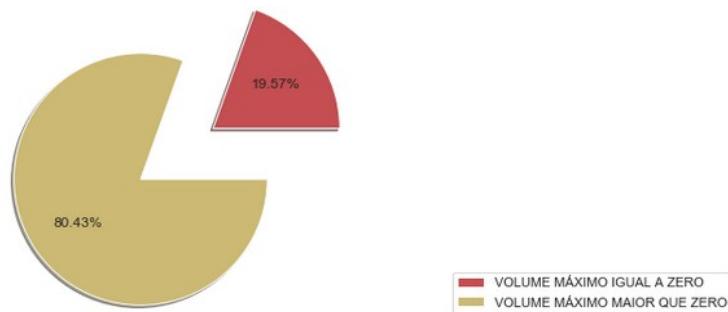
As finalidades de uso atuais foram sendo incorporadas ao longo dos anos, conforme informações da área de negócio, então algumas ocorrências de valor zero se devem ao fato de ainda não existirem, porém zero em todos os anos pode ser indício de ausência de regularização, ou apenas que em determinado município a atividade não é exercida. Daí o destaque dado ao uso no abastecimento humano, por ser essencial. Há ainda a possibilidade da captação não ocorrer no município de uso da água.

Os gráficos gerados estão disponíveis nas páginas a seguir. Não foi dada ênfase a estéticas, nessa fase de reconhecimento.

Exploração de dados da finalidade de uso irrigação

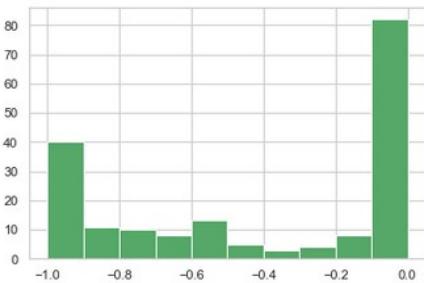


INCIDÊNCIA DE MUNICÍPIOS COM VOLUMES MÁXIMOS 'ZERO' NA FINALIDADE IRRIGAÇÃO

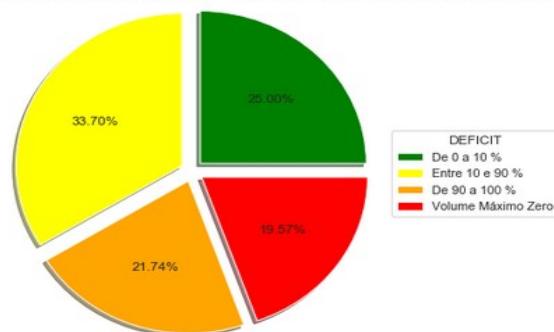


DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR FAIXA PROPORCIONAL DA DIFERENÇA ENTRE O VOLUME MÁXIMO E O VOLUME 2020 EM RELAÇÃO AO VOLUME MÁXIMO HISTÓRICO NA FINALIDADE IRRIGAÇÃO

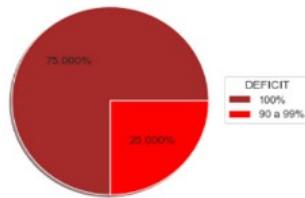
```
(array([40., 11., 10., 8., 13., 5., 3., 4., 8., 82.]),
 array([-1., -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0. ]),
 <a list of 10 Patch objects>)
```



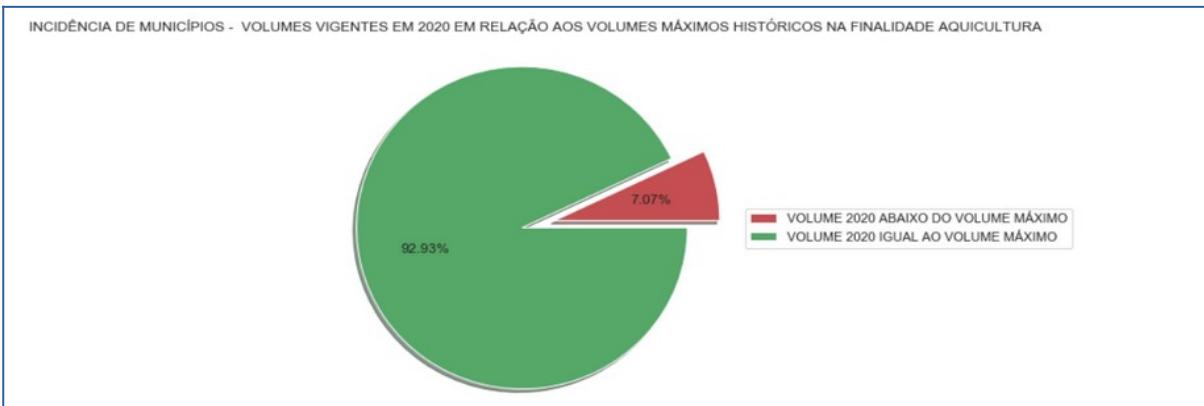
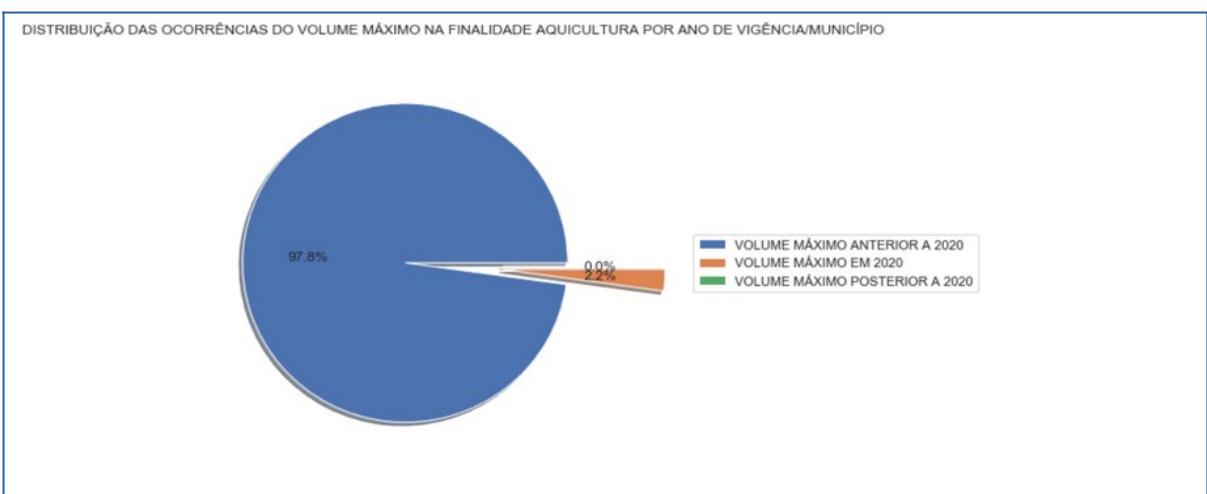
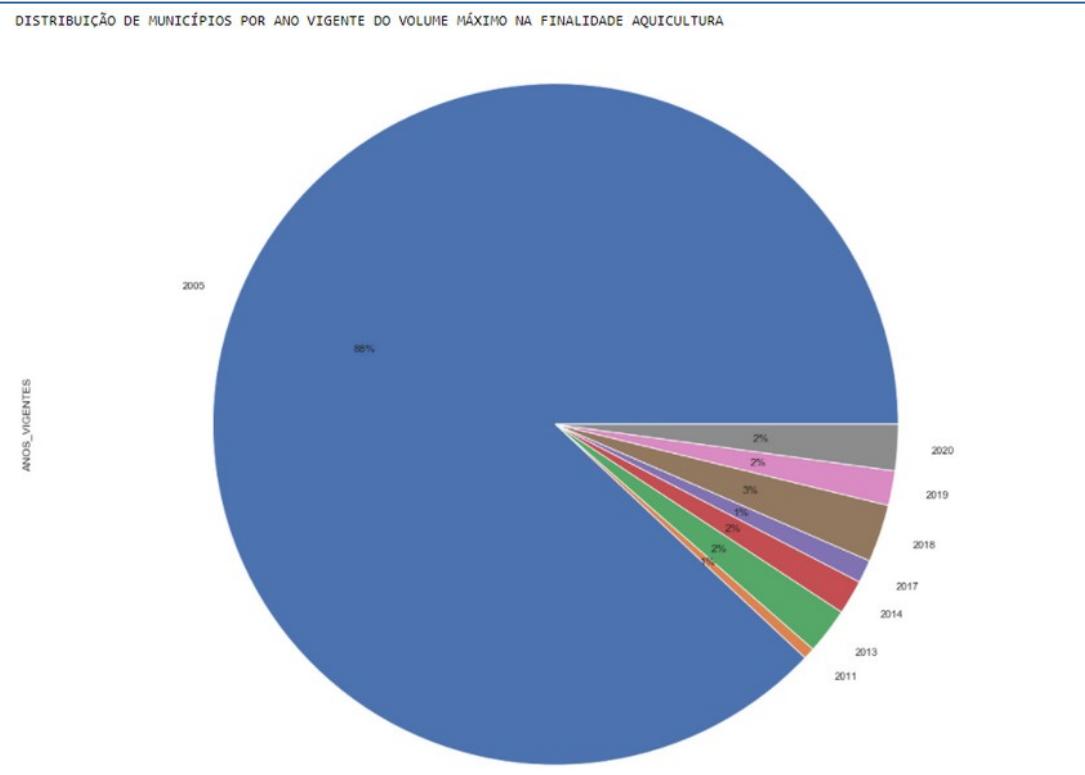
MUNICÍPIOS POR FAIXA DE DEFICIT EM VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS JÁ VIGENTES NA FINALIDADE IRRIGAÇÃO



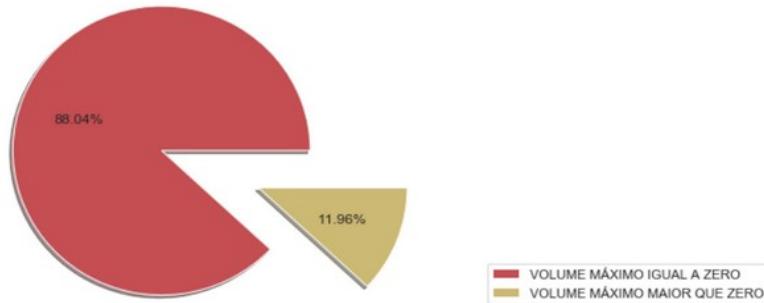
DISTRIBUIÇÃO DOS MUNICÍPIOS DENTRO DA FATIA DE 90 A 100% DE DEFICIT DE VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS HISTÓRICOS NA FINALIDADE IRRIGAÇÃO



Exploração de dados da finalidade de uso aquicultura

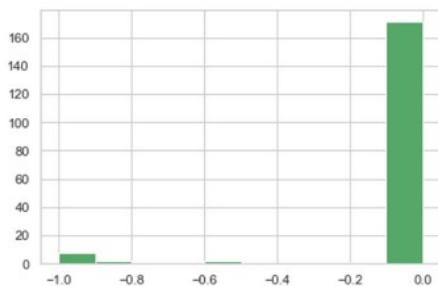


INCIDÊNCIA DE MUNICÍPIOS COM VOLUMES MÁXIMOS 'ZERO' NA FINALIDADE AQUICULTURA

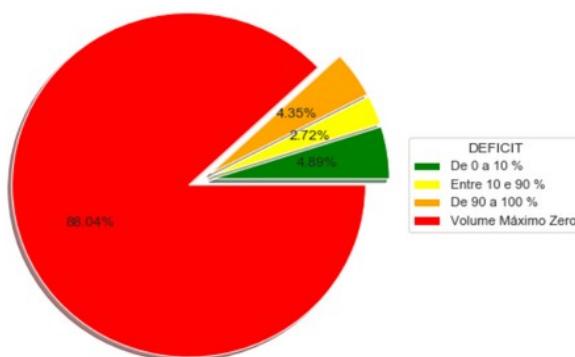


DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR FAIXA PROPORCIONAL DA DIFERENÇA ENTRE O VOLUME MÁXIMO E O VOLUME 2020 EM RELAÇÃO AO VOLUME MÁXIMO HISTÓRICO NA FINALIDADE AQUICULTURA

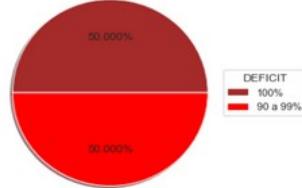
```
(array([ 8.,  2.,  0.,  0.,  2.,  0.,  0.,   1., 171.]),
array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ]),
<a list of 10 Patch objects>)
```



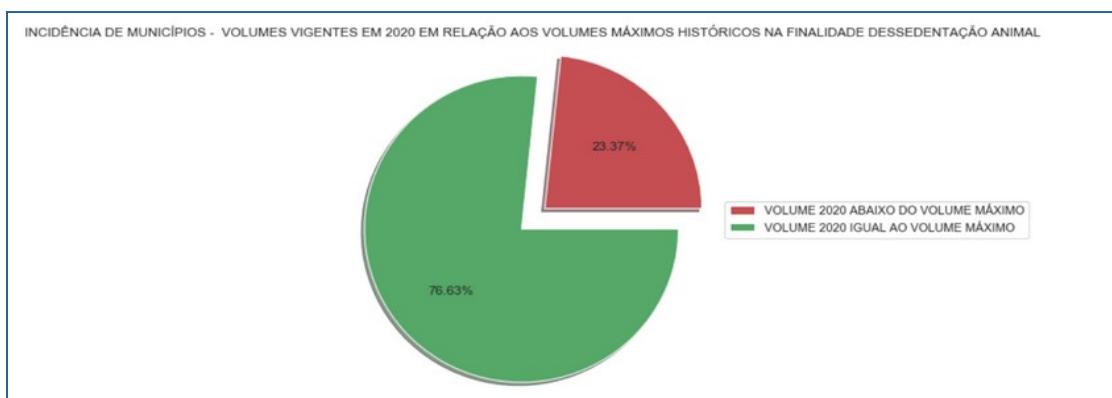
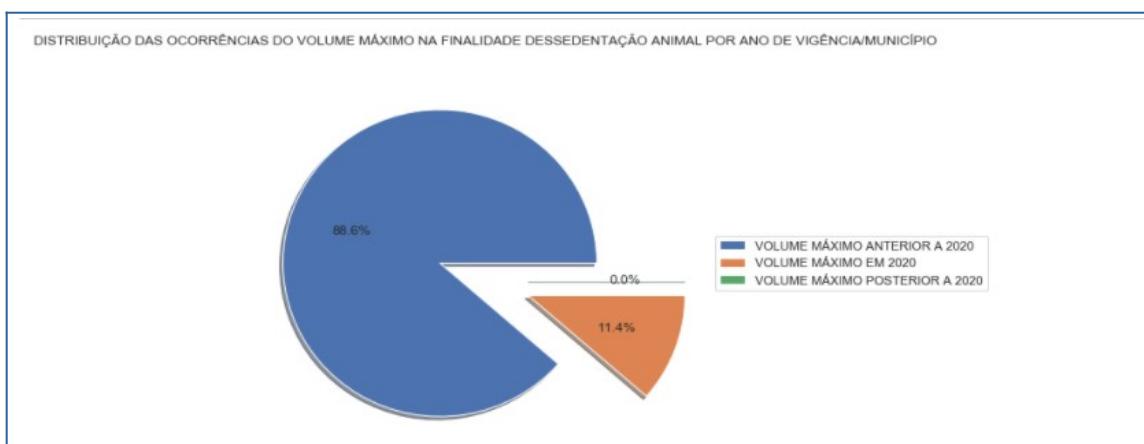
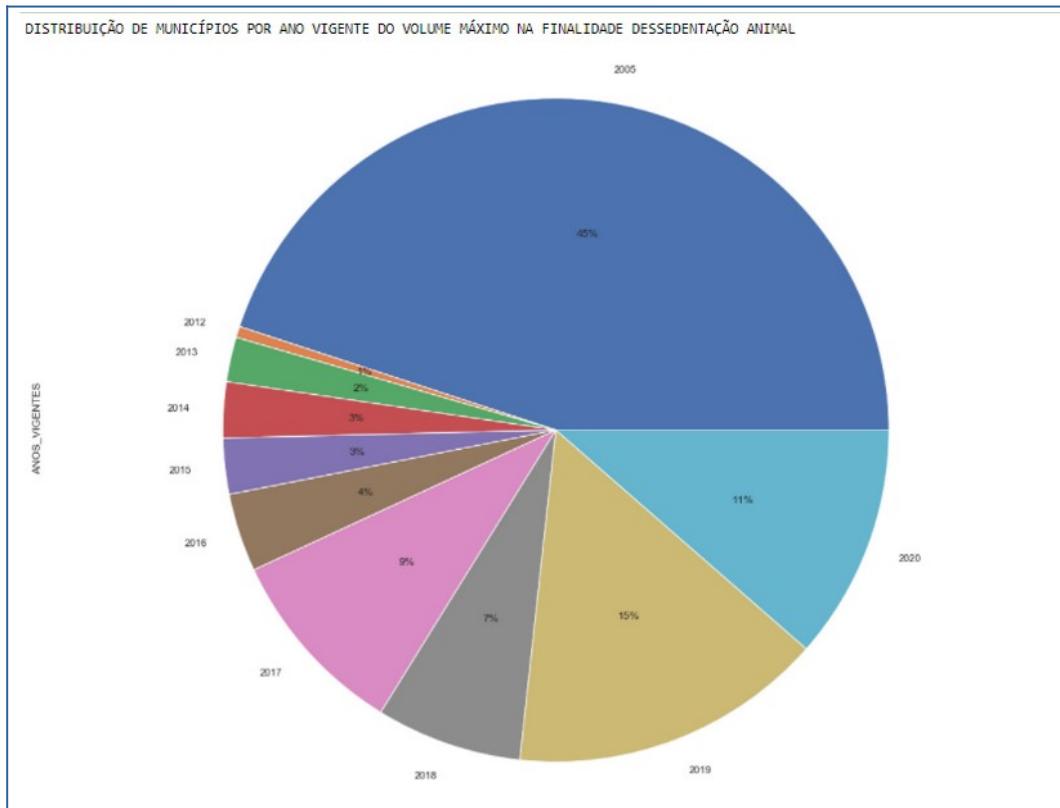
MUNICÍPIOS POR FAIXA DE DEFICIT EM VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS JÁ VIGENTES NA FINALIDADE AQUICULTURA



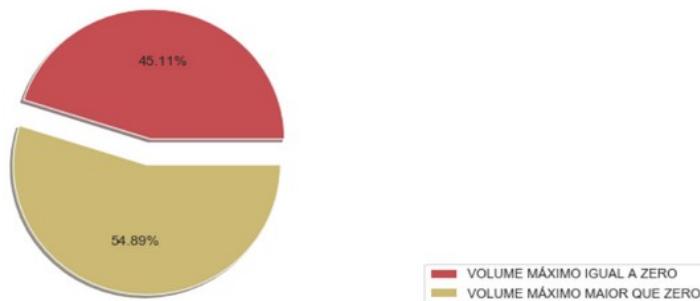
DISTRIBUIÇÃO DOS MUNICÍPIOS DENTRO DA FATIA DE 90 A 100% DE DEFICIT DE VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS HISTÓRICOS NA FINALIDADE AQUICULTURA



Exploração de dados da finalidade de uso dessedentação animal

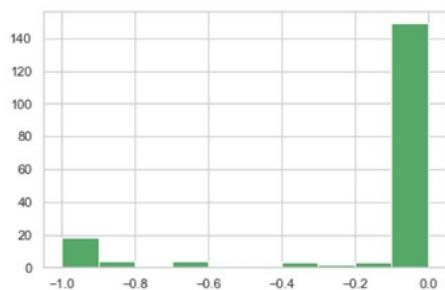


INCIDÊNCIA DE MUNICÍPIOS COM VOLUMES MÁXIMOS 'ZERO' NA FINALIDADE DESSEDENTAÇÃO ANIMAL

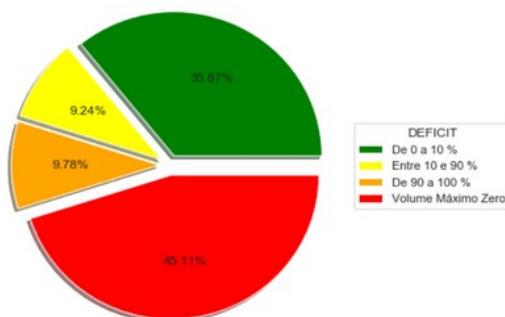


DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR FAIXA PROPORCIONAL DA DIFERENÇA ENTRE O VOLUME MÁXIMO E O VOLUME 2020 EM RELAÇÃO AO VOLUME MÁXIMO HISTÓRICO NA FINALIDADE DESSEDENTAÇÃO ANIMAL

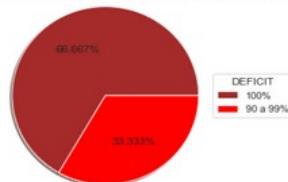
```
(array([ 18.,   4.,   0.,   4.,   1.,   0.,   3.,   2.,   3., 149.]),
 array([-1., -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ]),
 <a list of 10 Patch objects>)
```



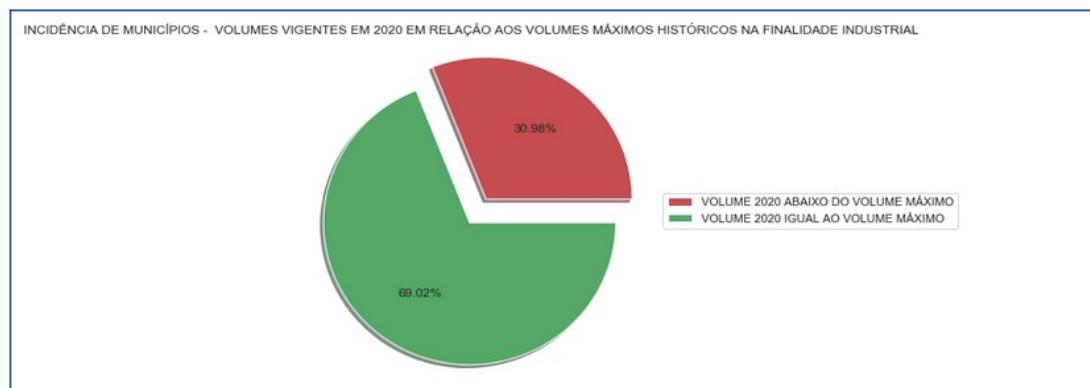
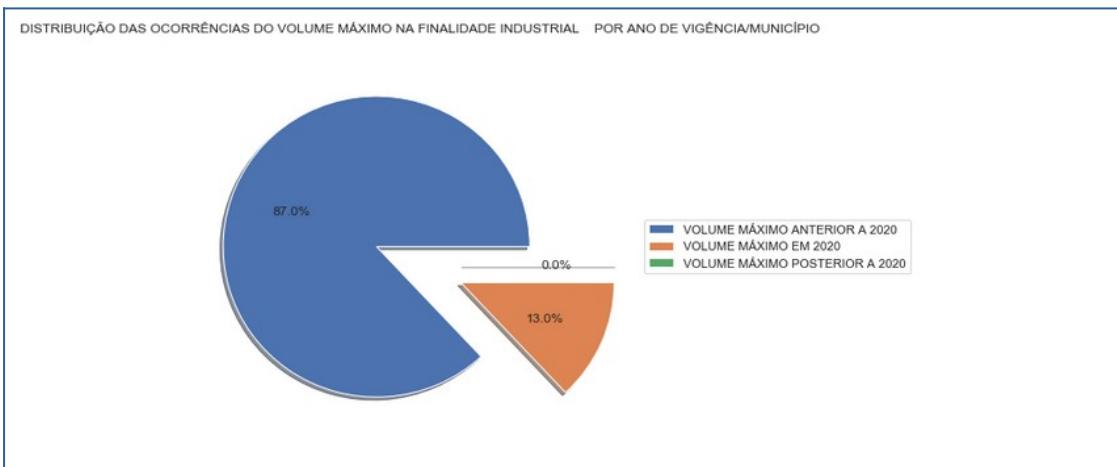
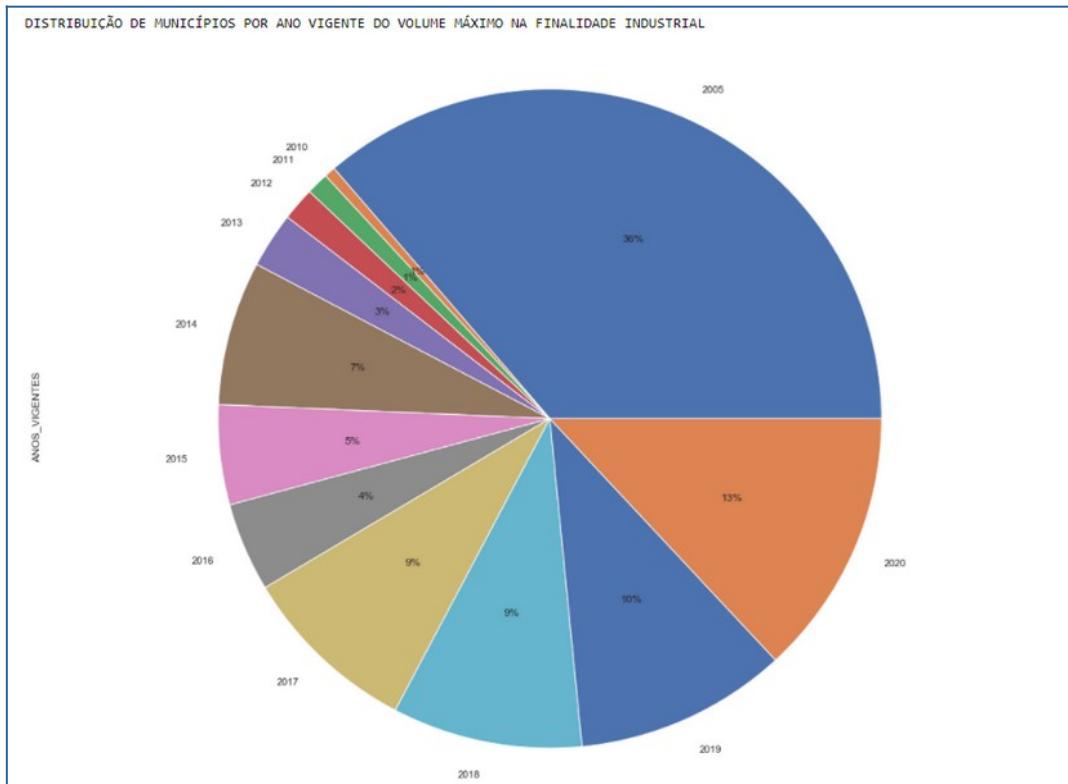
MUNICÍPIOS POR FAIXA DE DEFICIT EM VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS JÁ VIGENTES NA FINALIDADE DESSEDENTAÇÃO ANIMAL

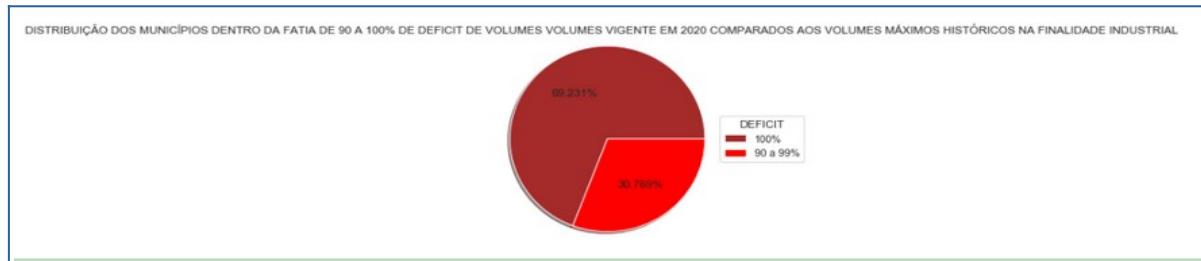
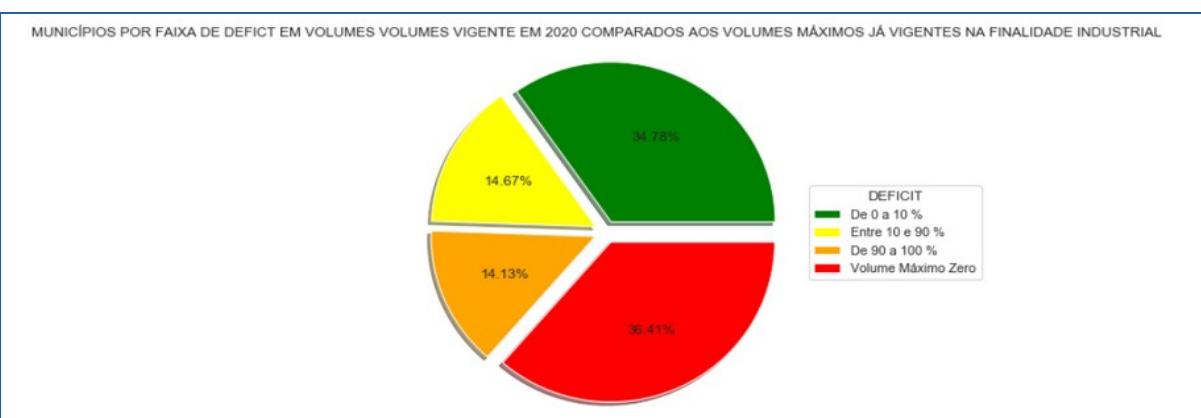
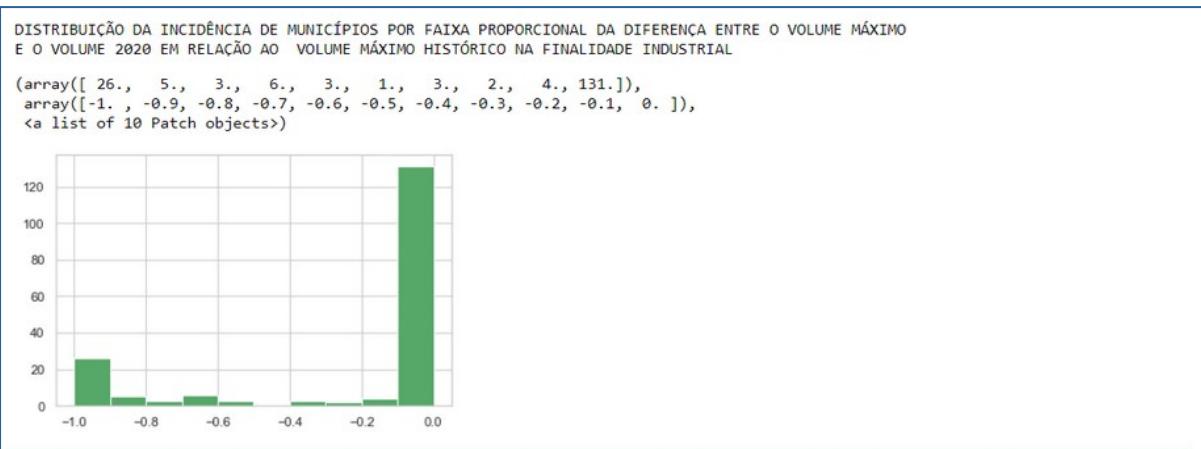
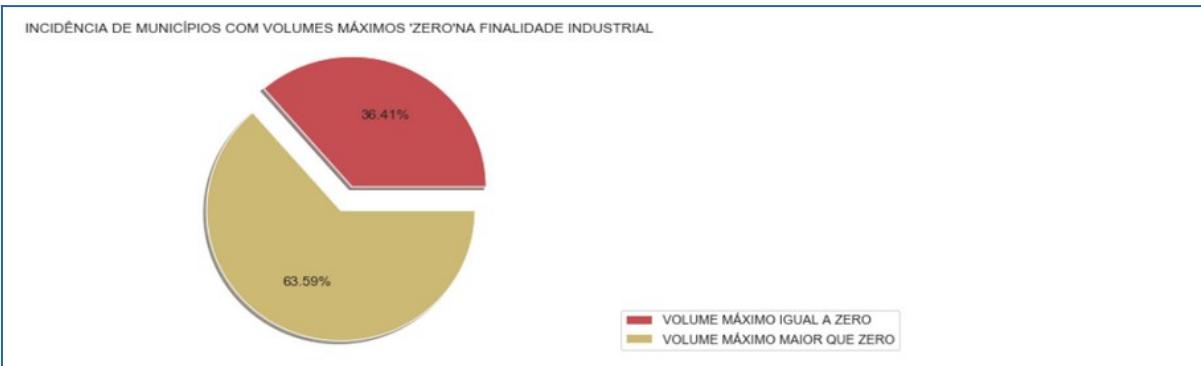


DISTRIBUIÇÃO DOS MUNICÍPIOS DENTRO DA FATIA DE 90 A 100% DE DEFICIT DE VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS HISTÓRICOS NA FINALIDADE DESSEDENTAÇÃO ANIMAL



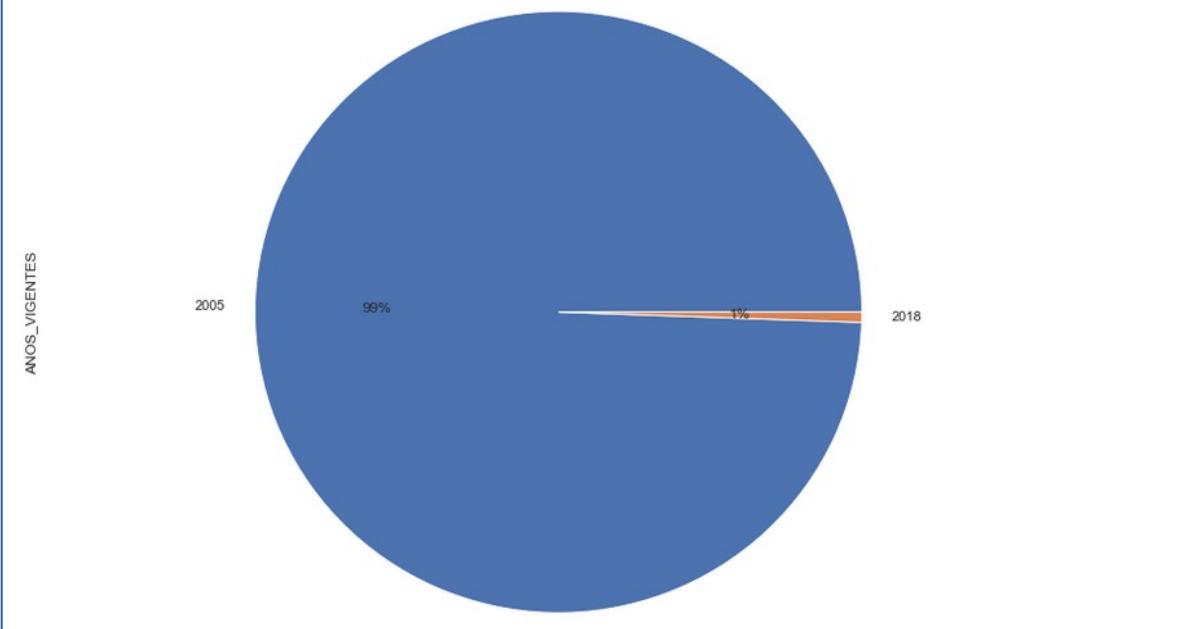
Exploração de dados da finalidade de uso industrial



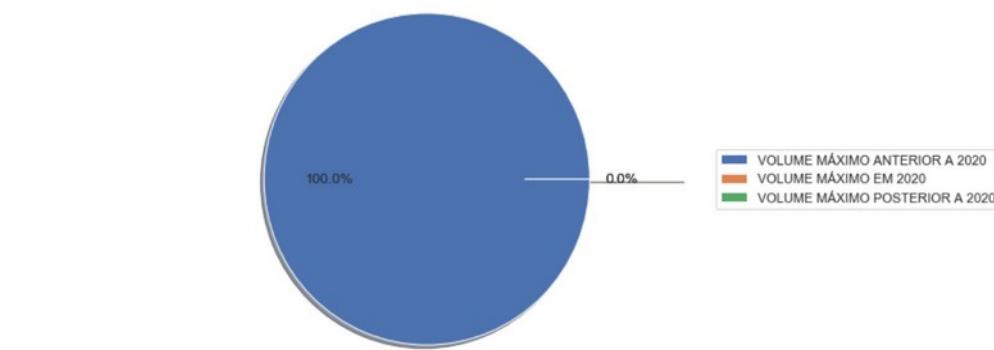


Exploração de dados da finalidade de uso serviço e comércio,

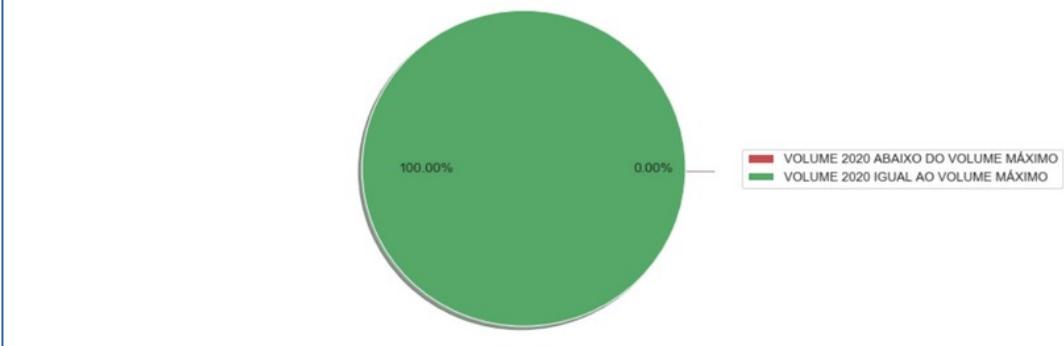
DISTRIBUIÇÃO DE MUNICÍPIOS POR ANO VIGENTE DO VOLUME MÁXIMO NA FINALIDADE SERVIÇO E COMÉRCIO

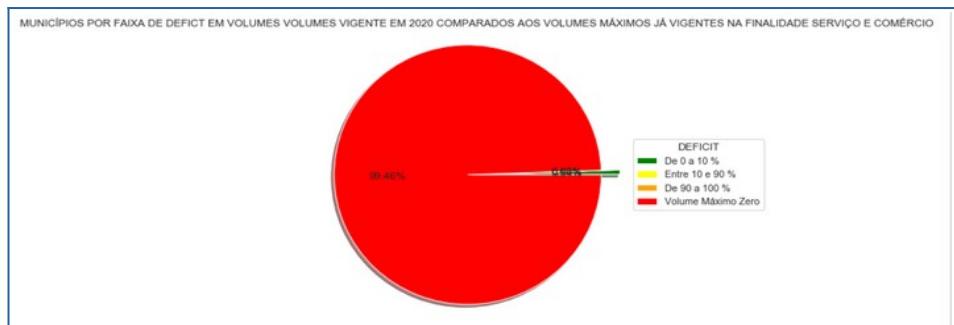
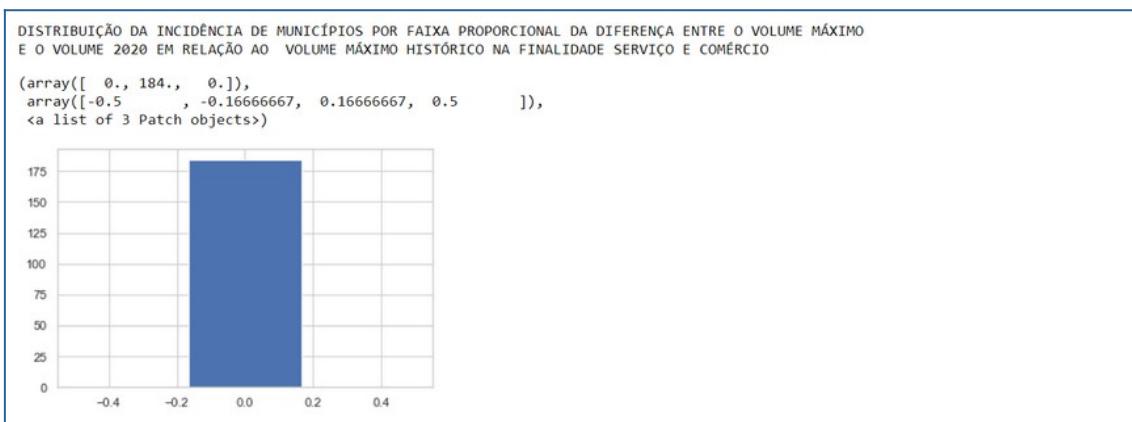
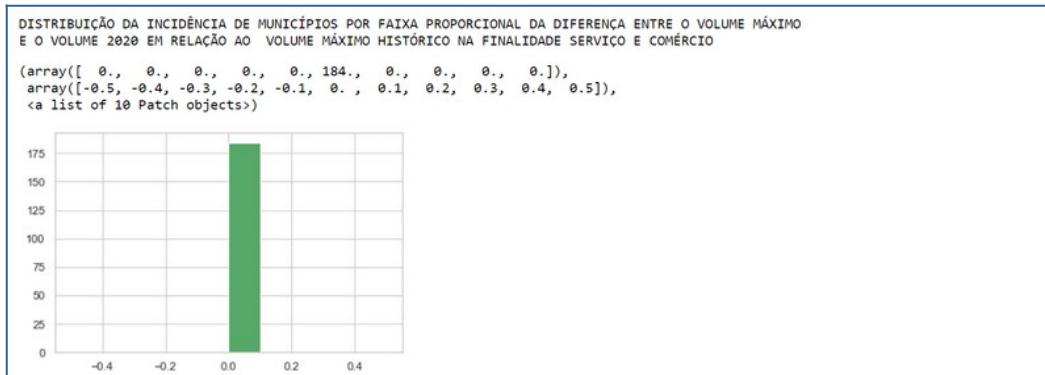
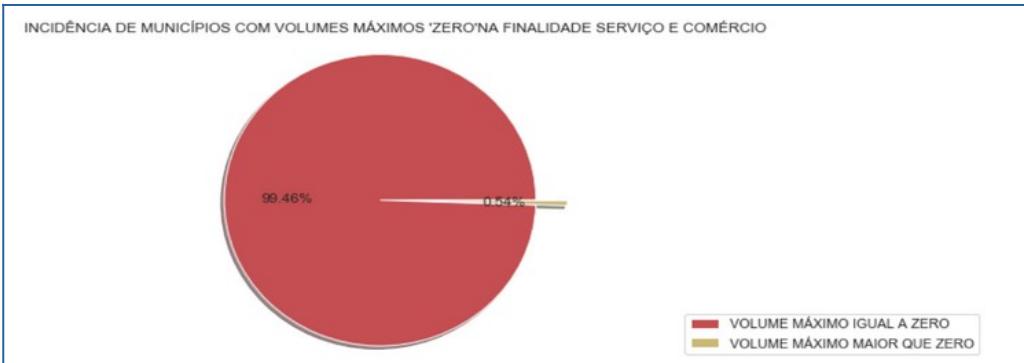


DISTRIBUIÇÃO DAS OCORRÊNCIAS DO VOLUME MÁXIMO NA FINALIDADE SERVIÇO E COMÉRCIO POR ANO DE VIGÊNCIA/MUNICÍPIO

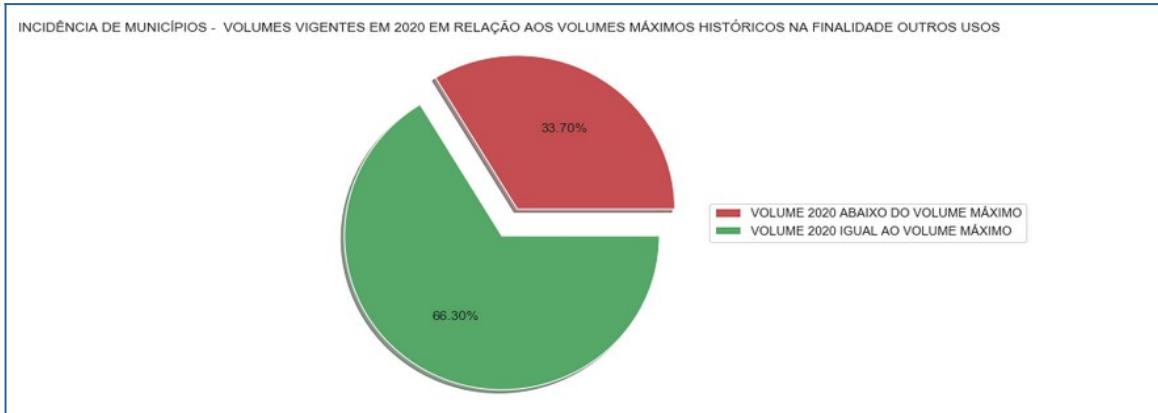
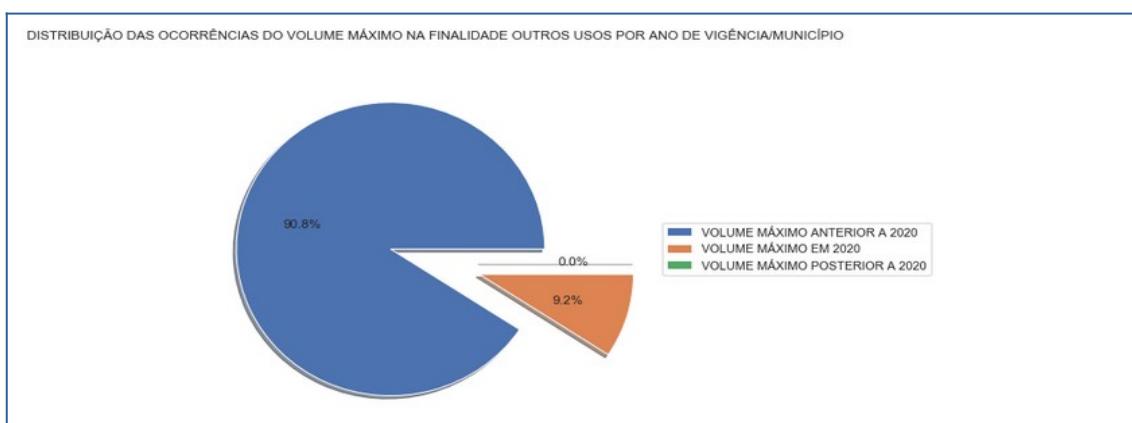
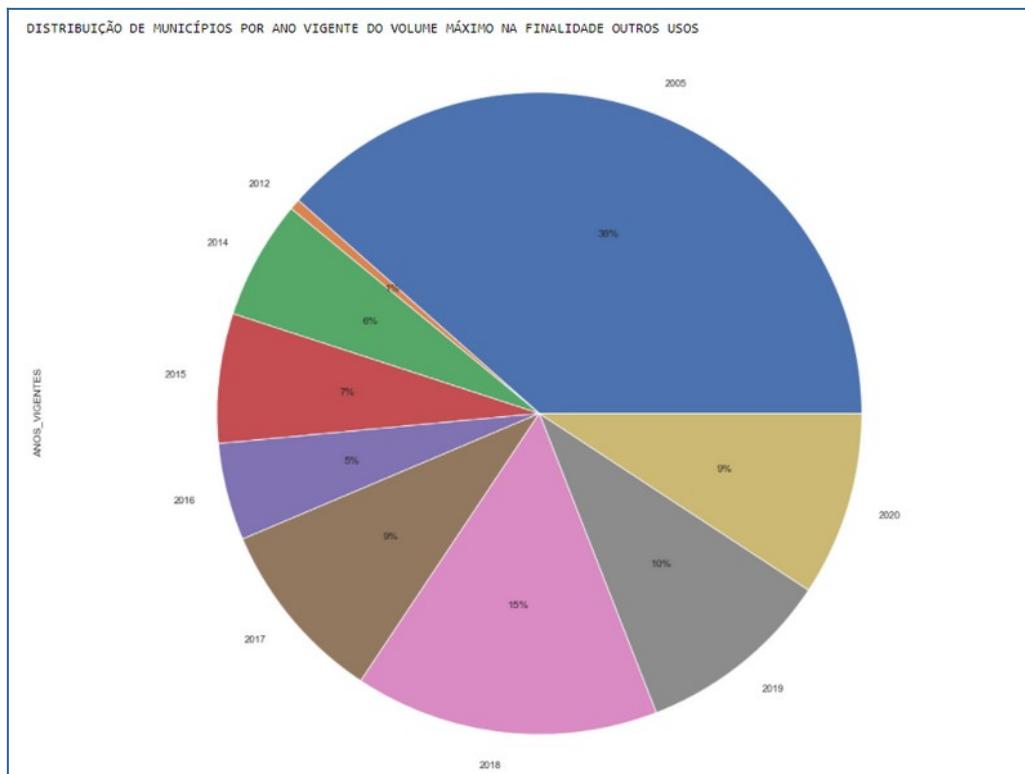


INCIDÊNCIA DE MUNICÍPIOS - VOLUMES VIGENTES EM 2020 EM RELAÇÃO AOS VOLUMES MÁXIMOS HISTÓRICOS NA FINALIDADE SERVIÇO E COMÉRCIO

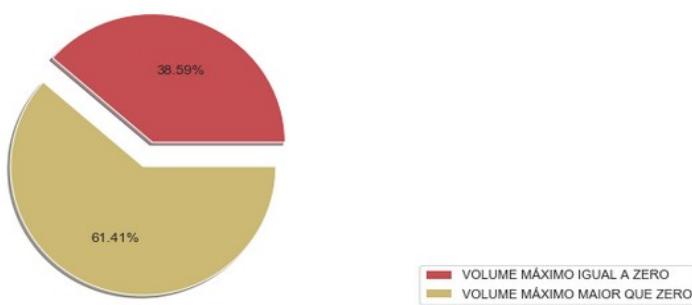




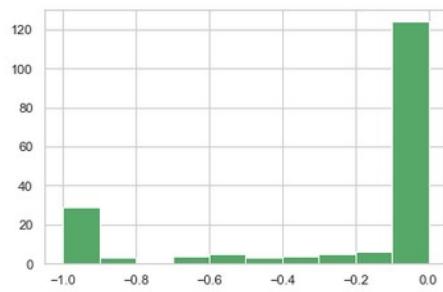
Exploração de dados da finalidade de uso outros usos.



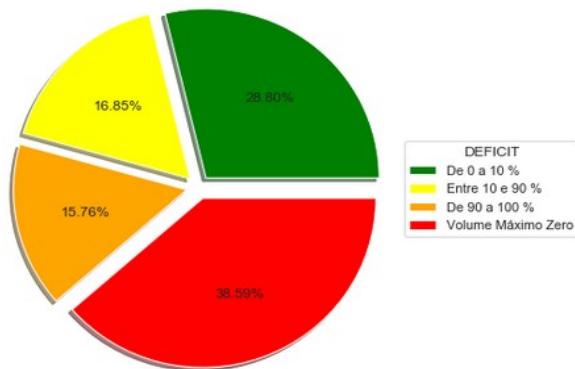
INCIDÊNCIA DE MUNICÍPIOS COM VOLUMES MÁXIMOS 'ZERO' NA FINALIDADE OUTROS USOS



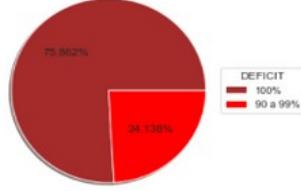
```
(array([ 29.,  3.,  1.,  4.,  5.,  3.,  4.,  5.,  6., 124.]),
array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ]),
<a list of 10 Patch objects>)
```



MUNICÍPIOS POR FAIXA DE DEFICIT EM VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS JÁ VIGENTES NA FINALIDADE OUTROS USOS



DISTRIBUIÇÃO DOS MUNICÍPIOS DENTRO DA FATIA DE 90 A 100% DE DEFICIT DE VOLUMES VIGENTE EM 2020 COMPARADOS AOS VOLUMES MÁXIMOS HISTÓRICOS NA FINALIDADE OUTROS USOS



Com relação ao uso da água na finalidade de irrigação, um município pode apresentar os maiores valores absolutos em volumes, e não ser o mais denso em relação a sua área territorial ou a área de seus imóveis rurais. Como ocorre esse comportamento? O que se pode extrair dele para apoiar nas tarefas de regularização de uso da água? Fatores climáticos também podem exercer influência, contudo não serão avaliados nesse estudo.

Inserindo a área dos imóveis rurais e territoriais dos municípios.

```
#INSERINDO A ÁREA DOS IMÓVEIS RURAIS (2005) DOS MUNICÍPIOS
lista_AREA_HA_IMOVEIS_2005 = list(IBGE_IPECE['AREA_HA_IMOVEIS_2005'])#Gerando uma lista a partir de uma coluna do df 'IBGE_IPECE'
DIREITO_DE_USO_IRRIG_MAX.insert(7,'AREA_HA_IMOVEIS_2005',lista_AREA_HA_IMOVEIS_2005,True)#Inserindo uma lista no df 'DIREITO_DE_U
#INSERINDO A ÁREA TERRITORIAL DOS MUNICÍPIOS
lista_AREA_KM2_2020 = list(IBGE_IPECE['AREA_KM2_2020'])#Gerando uma lista a partir de uma coluna do df 'IBGE_IPECE'
DIREITO_DE_USO_IRRIG_MAX.insert(7,'AREA_KM2_2020',lista_AREA_KM2_2020,True)#Inserindo uma lista no df 'DIREITO_DE_USO_IRRIG_MAX'
```

Obtendo as densidades, em cada município, dos volumes máximo e 2020, por áreas de imóveis rurais em m³/ha, e por áreas territoriais em m³/km².

```
#OBSERVANDO OS VOLUMES MÁXIMO E 2020 EM REALÇÃO ÀS ÁREAS TERRITORIAIS E DOS IMÓVEIS RURAIS
for iir in DIREITO_DE_USO_IRRIG_MAX.index:
    vol_irrig2020_area_terr = DIREITO_DE_USO_IRRIG_MAX['VOLUME_MUN_IRRIG2020_M3']*DIREITO_DE_USO_IRRIG_MAX['AREA_KM2_2020'] #Volume
    vol_irrigmax_area_terr = DIREITO_DE_USO_IRRIG_MAX['VOLUME_MAX_IRRIG_MUN_M3']*DIREITO_DE_USO_IRRIG_MAX['AREA_KM2_2020'] #Volume
    vol_irrig2020_area_imov_rur = DIREITO_DE_USO_IRRIG_MAX['VOLUME_MUN_IRRIG2020_M3']*DIREITO_DE_USO_IRRIG_MAX['AREA_HA_IMOVEIS'];
    vol_irrigmax_area_imov_rur = DIREITO_DE_USO_IRRIG_MAX['VOLUME_MAX_IRRIG_MUN_M3']*DIREITO_DE_USO_IRRIG_MAX['AREA_HA_IMOVEIS'];
DIREITO_DE_USO_IRRIG_MAX['DENS_VOL2020_TERR'] = vol_irrig2020_area_terr #Inserindo a coluna no df
DIREITO_DE_USO_IRRIG_MAX['DENS_VOLMAX_TERR'] = vol_irrigmax_area_terr #Inserindo a coluna no df
DIREITO_DE_USO_IRRIG_MAX['DENS_VOL2020_IMOVRUR'] = vol_irrig2020_area_imov_rur #Inserindo a coluna no df
DIREITO_DE_USO_IRRIG_MAX['DENS_VOLMAX_IMOVRUR'] = vol_irrigmax_area_imov_rur #Inserindo a coluna no df
```

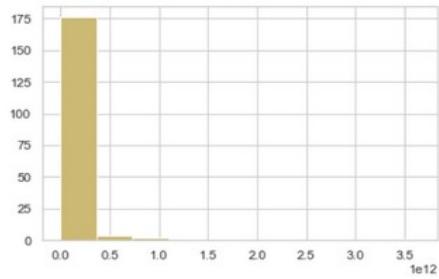
DIREITO_DE_USO_IRRIG_MAX						
VOL_2020_IRRIG	AREA_KM2_2020	AREA_HA_IMOVEIS_2005	DENS_VOL2020_TERR	DENS_VOLMAX_TERR	DENS_VOL2020_IMOVRUR	DENS_VOLMAX_IMOVRUR
-0.755261	180.833	10810.2000	7.940525e+07	3.244491e+08	4.746847e+09	1.939557e+10
-1.000000	130.002	8230.0000	0.000000e+00	9.235342e+05	0.000000e+00	5.846592e+07
-0.586937	842.471	44914.1000	6.908948e+08	1.672614e+09	3.683322e+10	8.917096e+10
-0.920886	2254.279	133636.9500	2.904774e+07	3.671635e+08	1.721992e+09	2.176599e+10
-0.462236	2438.563	131668.1000	4.674911e+07	8.693243e+07	2.524178e+09	4.693841e+09
...
0.000000	99.400	12368.0000	7.361119e+07	7.361119e+07	9.159187e+09	9.159187e+09
0.000000	697.683	38063.6000	3.891257e+06	3.891257e+06	2.122959e+08	2.122959e+08
-0.561187	179.239	6042.1000	1.927867e+07	4.393363e+07	6.498788e+08	1.480991e+09
0.000000	829.976	64269.7425	2.334756e+07	2.334756e+07	1.807934e+09	1.807934e+09
0.000000	1310.910	89404.1800	2.676361e+08	2.676361e+08	1.825281e+10	1.825281e+10

Observando os resultados

```
print('DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR DENSIDADE DE VOLUME 2020 EM RELAÇÃO A ÁREA DDE IMÓVEIS RURAIS')
plt.hist(DIREITO_DE_USO_IRRIG_MAX['DENS_VOL2020_IMOVRUR'], bins=10,color='y')
```

DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR DENSIDADE DE VOLUME 2020 EM RELAÇÃO A ÁREA DDE IMÓVEIS RURAIS

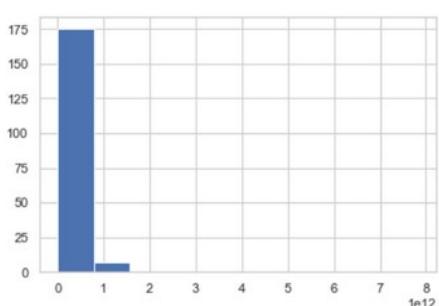
```
(array([176.,  4.,  2.,  1.,  0.,  0.,  0.,  0.,  1.]),
 array([0.0000000e+00, 3.65121843e+11, 7.30243686e+11, 1.09536553e+12,
 1.46048737e+12, 1.82560921e+12, 2.19073106e+12, 2.55585290e+12,
 2.92097474e+12, 3.28609659e+12, 3.65121843e+12]),
 <a list of 10 Patch objects>)
```



```
print('DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR DENSIDADE DE VOLUME MÁXIMO EM RELAÇÃO A ÁREA DDE IMÓVEIS RURAIS')
plt.hist(DIREITO_DE_USO_IRRIG_MAX['DENS_VOLMAX_IMOVRUR'], bins=10,color='b')
```

DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR DENSIDADE DE VOLUME MÁXIMO EM RELAÇÃO A ÁREA DDE IMÓVEIS RURAIS

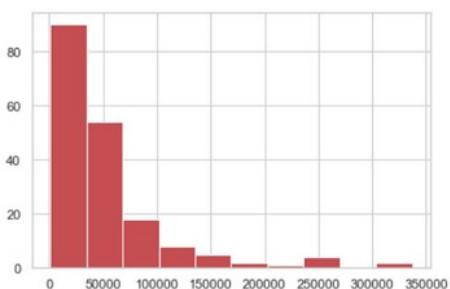
```
(array([176.,  7.,  0.,  1.,  0.,  0.,  0.,  0.,  1.]),
 array([0.0000000e+00, 7.82669316e+11, 1.56533863e+12, 2.34800795e+12,
 3.13067726e+12, 3.91334658e+12, 4.69601590e+12, 5.47868521e+12,
 6.26135453e+12, 7.04402385e+12, 7.82669316e+12]),
 <a list of 10 Patch objects>)
```



```
print('DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR ÁREA DE IMÓVEIS RURAIS')
plt.hist(DIREITO_DE_USO_IRRIG_MAX['AREA_HA_IMOVEIS_2005'], bins=10,color='r')
```

DISTRIBUIÇÃO DA INCIDÊNCIA DE MUNICÍPIOS POR ÁREA DE IMÓVEIS RURAIS

```
(array([90., 54., 18., 8., 5., 2., 1., 4., 0., 2.]),
 array([ 1017.8 , 34624.51, 68231.22, 101837.93, 135444.64, 169051.35,
 202658.06, 236264.77, 269871.48, 303478.19, 337084.9 ]),
 <a list of 10 Patch objects>)
```



Com relação ao uso da água na finalidade de aquicultura, um município pode apresentar os maiores valores absolutos em volumes, e não ser o mais denso em relação a sua área territorial ou a área de seus imóveis rurais. Como ocorre esse comportamento? O que se pode extrair dele para apoiar nas tarefas de regularização de uso da água? Fatores climáticos também podem exercer influência, contudo não serão avaliados nesse estudo.

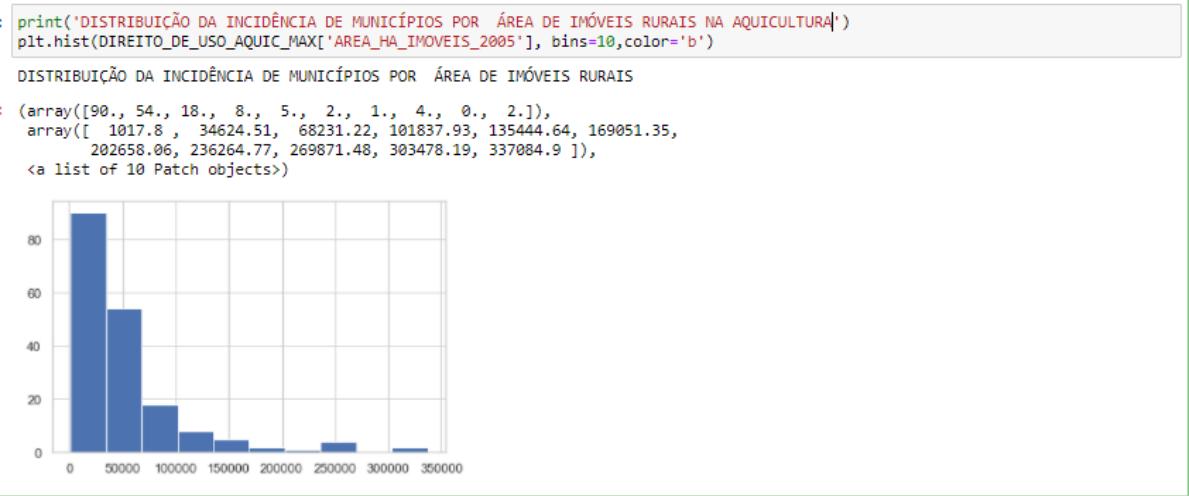
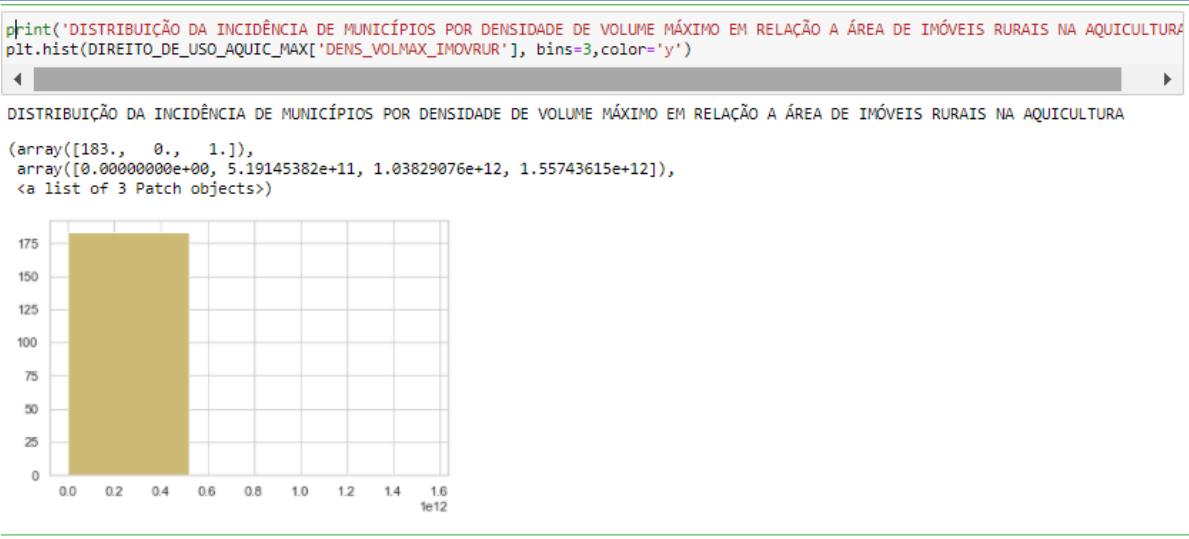
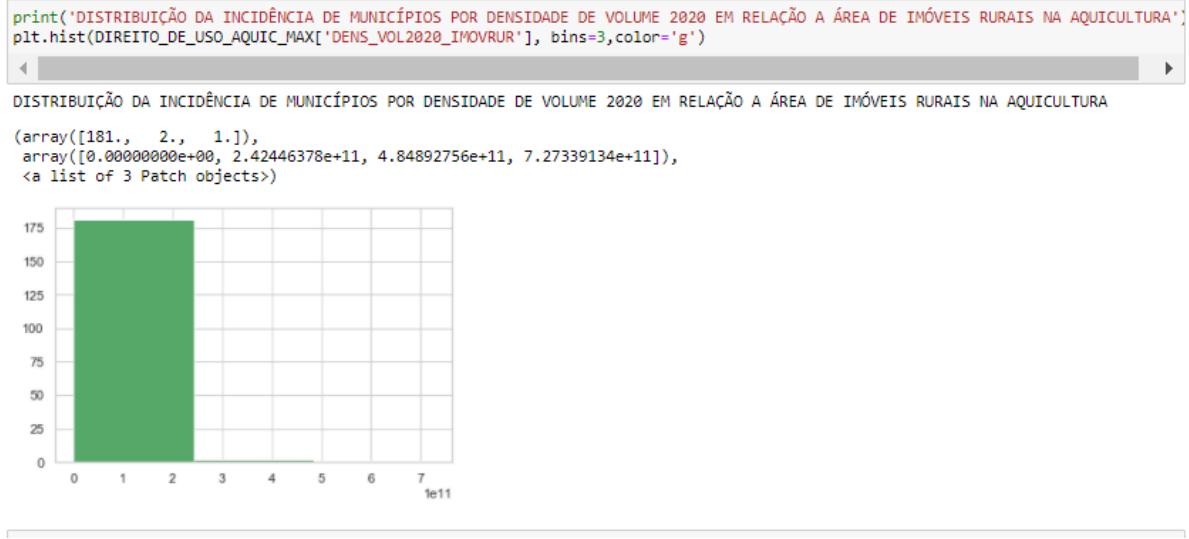
Inserindo a área dos imóveis rurais e territoriais dos municípios e obtendo as densidades em cada município dos volumes máximo e 2020, por áreas de imóveis rurais em m³/ha, e territoriais em m³/km².

```
#INSERINDO A ÁREA DOS IMÓVEIS RURAIS (2005) DOS MUNICÍPIOS
lista_AREA_HA_IMOVEIS_2005 = list(IBGE_IPECE['AREA_HA_IMOVEIS_2005'])#Gerando uma Lista a partir de uma coluna do df 'IBGE_IPECE'
DIREITO_DE_USO_AQUIC_MAX.insert(7,'AREA_HA_IMOVEIS_2005',lista_AREA_HA_IMOVEIS_2005,True)#Inserindo uma Lista no df 'DIREITO_DE_U
#INSERINDO A ÁREA DOS MUNICÍPIOS
lista_AREA_KM2_2020 = list(IBGE_IPECE['AREA_KM2_2020'])#Gerando uma Lista a partir de uma coluna do df 'IBGE_IPECE'
DIREITO_DE_USO_AQUIC_MAX.insert(7,'AREA_KM2_2020',lista_AREA_KM2_2020,True)#Inserindo uma Lista no df 'DIREITO_DE_USO_AQUIC_MAX'

#OBSERVANDO OS VOLUMES MÁXIMO E 2020 EM RELAÇÃO ÀS ÁREAS TERRITORIAIS E DOS IMÓVEIS RURAIS
for iaq in DIREITO_DE_USO_AQUIC_MAX.index:
    vol_aquic2020_area_terr = DIREITO_DE_USO_AQUIC_MAX['VOLUME_MUN_AQUIC2020_M3']*DIREITO_DE_USO_AQUIC_MAX['AREA_KM2_2020'] #Vol2020
    vol_aquicmax_area_terr = DIREITO_DE_USO_AQUIC_MAX['VOLUME_MAX_AQUIC_MUN_M3']*DIREITO_DE_USO_AQUIC_MAX['AREA_KM2_2020'] #Volmax
    vol_aquic2020_area_imov_rur = DIREITO_DE_USO_AQUIC_MAX['VOLUME_MUN_AQUIC2020_M3']*DIREITO_DE_USO_AQUIC_MAX['AREA_HA_IMOVEIS_2005']
    vol_aquicmax_area_imov_rur = DIREITO_DE_USO_AQUIC_MAX['VOLUME_MAX_AQUIC_MUN_M3']*DIREITO_DE_USO_AQUIC_MAX['AREA_HA_IMOVEIS_2005']
    DIREITO_DE_USO_AQUIC_MAX['DENS_VOL2020_TERR'] = vol_aquic2020_area_terr #Inserindo a coluna no df
    DIREITO_DE_USO_AQUIC_MAX['DENS_VOLMAX_TERR'] = vol_aquicmax_area_terr #Inserindo a coluna no df
    DIREITO_DE_USO_AQUIC_MAX['DENS_VOL2020_IMOVRUR'] = vol_aquic2020_area_imov_rur #Inserindo a coluna no df
    DIREITO_DE_USO_AQUIC_MAX['DENS_VOLMAX_IMOVRUR'] = vol_aquicmax_area_imov_rur #Inserindo a coluna no df
```

DIREITO_DE_USO_AQUIC_MAX						
/OL_2020_AQUIC	AREA_KM2_2020	AREA_HA_IMOVEIS_2005	DENS_VOL2020_TERR	DENS_VOLMAX_TERR	DENS_VOL2020_IMOVRUR	DENS_VOLMAX_IMOVRUR
0.0	180.833	10810.2000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	130.002	8230.0000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	842.471	44914.1000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	2254.279	133638.9500	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	2438.563	131668.1000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
...
0.0	99.400	12368.0000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	697.683	38063.6000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	179.239	6042.1000	4.464915e+06	4.464915e+06	1.505111e+08	1.505111e+08
0.0	829.976	64269.7425	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.0	1310.910	89404.1800	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

Observando os resultados



Preparando para observação da evolução do uso da água na finalidade irrigação ao longo do tempo. Será gerado um *dataframe* com os volumes em anos versus municípios para uso posterior.

```
#OBSERVANDO A EVOLUÇÃO DO USO DA ÁGUA NA FINALIDADE IRRIGAÇÃO AO LONGO DO TEMPO
DIREITO_DE_USO_IRRIG_ord = DIREITO_DE_USO_IRRIG # Definindo um novo df
DIREITO_DE_USO_IRRIG_ord = DIREITO_DE_USO_IRRIG_ord.sort_values(by=['ANOS_VIGENTES','MUNICIPIO'])# Ordenação encadeada por ano v

DIREITO_DE_USO_IRRIG_ord|
```

MUNICIPIO	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MUN_M3
209	ABAIARA	Irrigação	2005	0.0	0.0
625	ACARAPE	Irrigação	2005	0.0	0.0
1041	ACARAU	Irrigação	2005	0.0	0.0
1457	ACOPIARA	Irrigação	2005	0.0	0.0
1873	AIUABA	Irrigação	2005	0.0	0.0
...
74723	URUBURETAMA	Irrigação	2030	0.0	0.0
75139	URUOCA	Irrigação	2030	0.0	0.0
75555	VARJOTA	Irrigação	2030	0.0	0.0
75971	VARZEA ALEGRE	Irrigação	2030	0.0	0.0
76387	VICOSA DO CEARA	Irrigação	2030	0.0	0.0

4784 rows × 6 columns

Os intervalos de linhas referentes a um mesmo ano serão convertidos em colunas.

```
#GERANDO UMA LISTA COM TODOS OS VALORES DOS VOLUMES ANUAIS DE CADA MUNICÍPIO - IRRIGAÇÃO
volumes_irrig = DIREITO_DE_USO_IRRIG_ord['VOLUME_MUN_M3']
vol_irrig = list(volumes_irrig) #Convertendo em lista

#GERANDO UM DATAFRAME DE UMA COLUNA COM OS VALORES DO VOLUMES ANUAIS
dados_irrig= pd.DataFrame(vol_irrig) # Novo dataframe
nlinhas_irrig,ncols_irrig=dados_irrig.shape# Dimensões do dataframe

#O NÚMERO DE COLUNAS É DEFINIDO PELO NÚMERO DE MUNICÍPIOS
conj_munic_irrig = set(DIREITO_DE_USO_IRRIG_ord['MUNICIPIO'])# Agrupando por municípios
ncolunas_irrig = len(conj_munic_irrig)#Obtendo o número de municípios sem repetições
ncolunas_irrig = int(ncolunas_irrig)#Convertendo para número inteiro

ncolunas_irrig
184

#CRIANDO UMA LISTA DE ÍNDICES INICIAIS PARA O NOVO DATAFRAME
lista_irrig = list(range(0, nlinhas_irrig, ncolunas_irrig))

print(lista_irrig)
[0, 184, 368, 552, 736, 920, 1104, 1288, 1472, 1656, 1840, 2024, 2208, 2392, 2576, 2760, 2944, 3128, 3312, 3496, 3680, 3864, 4048, 4232, 4416, 4600]
```

```
DADOS DOS VOLUMES LINHA A LINHA
range(0, ncolunas_irrig)] #Gerando os nomes das colunas para que os dados encaixem corretamente durante o uso do append
me(columns=colnomes_irrig)#Gerando um Dataframe vazio para receber os dados dos volumes com as colunas nomeadas conforme a Lista
dados_transpostos_irrig|
```

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	...	C174	C175	C176	C177	C178	C179	C180	C181	C182	C183
----	----	----	----	----	----	----	----	----	----	-----	------	------	------	------	------	------	------	------	------	------

0 rows × 184 columns

```
#SUBSTITUINDO OS NOMES DAS COLUNAS PELO NOMES DOS MUNICÍPIOS
old_names_irrig = dados_transpostos_irrig.columns.tolist()#Obtendo a Lista dos nomes a serem substituídos
print(old_names_irrig)
```

```
['C0', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9', 'C10', 'C11', 'C12', 'C13', 'C14', 'C15', 'C16', 'C17', 'C18', 'C19', 'C20', 'C21', 'C22', 'C23', 'C24', 'C25', 'C26', 'C27', 'C28', 'C29', 'C30', 'C31', 'C32', 'C33', 'C34', 'C35', 'C36', 'C37', 'C38', 'C39', 'C40', 'C41', 'C42', 'C43', 'C44', 'C45', 'C46', 'C47', 'C48', 'C49', 'C50', 'C51', 'C52', 'C53', 'C54', 'C55', 'C56', 'C57', 'C58', 'C59', 'C60', 'C61', 'C62', 'C63', 'C64', 'C65', 'C66', 'C67', 'C68', 'C69', 'C70', 'C71', 'C72', 'C73', 'C74', 'C75', 'C76', 'C77', 'C78', 'C79', 'C80', 'C81', 'C82', 'C83', 'C84', 'C85', 'C86', 'C87', 'C88', 'C89', 'C90', 'C91', 'C92', 'C93', 'C94', 'C95', 'C96', 'C97', 'C98', 'C99', 'C100', 'C101', 'C102', 'C103', 'C104', 'C105', 'C106', 'C107', 'C108', 'C109', 'C110', 'C111', 'C112', 'C113', 'C114', 'C115', 'C116', 'C117', 'C118', 'C119', 'C120', 'C121', 'C122', 'C123', 'C124', 'C125', 'C126', 'C127', 'C128', 'C129', 'C130', 'C131', 'C132', 'C133', 'C134', 'C135', 'C136', 'C137', 'C138', 'C139', 'C140', 'C141', 'C142', 'C143', 'C144', 'C145', 'C146', 'C147', 'C148', 'C149', 'C150', 'C151', 'C152', 'C153', 'C154', 'C155', 'C156', 'C157', 'C158', 'C159', 'C160', 'C161', 'C162', 'C163', 'C164', 'C165', 'C166', 'C167', 'C168', 'C169', 'C170', 'C171', 'C172', 'C173', 'C174', 'C175', 'C176', 'C177', 'C178', 'C179', 'C180', 'C181', 'C182', 'C183']
```

```
new_names_irrig = list(conj_munic_irrig)#Obtendo a Lista dos nomes dos municípios
new_names_irrig.sort()#Colocando em ordem alfabética
print(new_names_irrig)
```

```
['ABAIARA', 'ACARAPE', 'ACARAU', 'ACOPIARA', 'AIUABA', 'ALCANTARAS', 'ALTANEIRA', 'ALTO SANTO', 'AMONTADA', 'ANTONINA DO NORTE', 'APUIARES', 'AQUIRAZ', 'ARACATI', 'ARACOIABA', 'ARARENDA', 'ARARIPE', 'ARATUBA', 'ARNEIROZ', 'ASSARE', 'AURORA', 'BAIXIO', 'BANABUIU', 'BARBALHA', 'BARREIRA', 'BARRO', 'BARROQUINHA', 'BATURITE', 'BEBERIBE', 'BELA CRUZ', 'BOA VIAGEM', 'BREJO SANTO', 'CAMOCIM', 'CAMPOS SALES', 'CANINDE', 'CAPISTRANO', 'CARIDADE', 'CARIRE', 'CARIRIACU', 'CARIUS', 'CARNAUBAL', 'CASCABEL', 'CATARINA', 'CATUNDU', 'CAUCAIA', 'CEDRO', 'CHAVAL', 'CHORO', 'CHOROZINHO', 'COREAU', 'CRATEUS', 'CRATO', 'CROATA', 'CRUZ', 'DEPUTADO IRAPUAN PINHEIRO', 'ERERE', 'EUSEBIO', 'FARIAS BRITO', 'FORQUELHA', 'FORTALEZA', 'FORTIM', 'FRECHEIRINHA', 'GENERAL SAMPAIO', 'GRACA', 'GRANJA', 'GRANJEIRO', 'GROAIRAS', 'GUAIUBA', 'GUARACIABA DO NORTE', 'GUARAMIRANGA', 'HIDROLANDIA', 'HORIZONTE', 'IBARETAMA', 'IBIAPINA', 'IBICUITINGA', 'ICAPIUI', 'ICO', 'IGUATU', 'INDEPENDENCIA', 'IPAPORANGA', 'IPAUMIRIM', 'IPU', 'IPUEIRAS', 'IRACEMA', 'IRACUBA', 'ITAICABA', 'ITAITINGA', 'ITAPAGE', 'ITAPIPOCA', 'ITAPIUNA', 'ITAREMA', 'ITATIRA', 'JAGUARETAMA', 'JAGUARI BARA', 'JAGUARIBE', 'JAGUARUANA', 'JARDIM', 'JATI', 'JIJOCA DE JERICOCOARA', 'JUAZEIRO DO NORTE', 'JUCAS', 'LAVRAS DA MANGABEIRA', 'LIMOEO DO NORTE', 'MADALENA', 'MARACANAU', 'MARANGUAPE', 'MARCO', 'MARTINOPOL', 'MASSAPE', 'MAURITI', 'MERUOCA', 'MILA GRES', 'MILHA', 'MIRAIMA', 'MISSAO VELHA', 'MOMBACA', 'MONSENHOR TABOSA', 'MORADA NOVA', 'MORAUJO', 'MORRINHOS', 'MUCAMBO', 'MULUNGU', 'NOVA OLINDA', 'NOVA RUSSAS', 'NOVO ORIENTE', 'OCARA', 'ODOS', 'PACAJUS', 'PACOTI', 'PACUJA', 'PALHANO', 'PALMACIA', 'PARACURU', 'PARAIPABA', 'PARAMBU', 'PARAMOTI', 'PEDRA BRANCA', 'PENAFORTE', 'PENTECOSTE', 'PEREIRO', 'PINDORETAMA', 'PIQUET CARNEIRO', 'PIRES FERREIRA', 'PORANGA', 'PORTEIRAS', 'POTENGI', 'POTIRETAMA', 'QUITERIANOPOLIS', 'QUIXADA', 'QUIXELO', 'QUIXERAMOBIM', 'QUIXERE', 'REDENCAO', 'RERIUTABA', 'RUSSAS', 'SABOEIRO', 'SALITRE', 'SANTA QUITERIA', 'SANTANA DO ACARAU', 'SATANA DO CARIRI', 'SAO BENEDITO', 'SAO GONCALO DO AMARANTE', 'SAO JOAO DO JAGUARIBE', 'SAO LUIS DO CURU', 'SEADOR POMPEU', 'SENADOR SA', 'SOBRAL', 'SOLONOPOLE', 'TABULEIRO DO NORTE', 'TAMBORIL', 'TARRAFAS', 'TAUA', 'TEJUCUOCA', 'TIANGUA', 'TRAIRI', 'TURURU', 'UBAJARA', 'UMARI', 'UMIRIM', 'URUBURETAMA', 'URUOCA', 'VARJOTA', 'VARZEA ALEGRE', 'VICOSA DO CEARA']
```

```
dados_transpostos_irrig.rename(columns=dict(zip(old_names_irrig, new_names_irrig)), inplace=True)#renomeando as colunas
```

```
dados_transpostos_irrig
```

	ABAIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	ANTONINA DO NORTE	...	TRA
0	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-
1	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-
2	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-
3	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-
4	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-
5	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-
6	1.216208e+06	0.0	3.788960e+05	0.000000	0.000000	0.0	0.000000	1.120000e+04	0.000000	0.0	...	0.000000e-
7	1.216208e+06	0.0	3.788960e+05	0.000000	0.000000	0.0	0.000000	1.569588e+07	0.000000	0.0	...	9.131120e-
8	1.216208e+06	7104.0	3.788960e+05	0.000000	0.000000	0.0	0.000000	1.590204e+07	0.000000	0.0	...	9.131120e-
9	1.794192e+06	7104.0	6.319300e+05	4353.120117	0.000000	0.0	0.000000	1.590204e+07	0.000000	0.0	...	3.015376e-
10	1.613502e+06	0.0	2.530340e+05	135894.890625	0.000000	0.0	17001.599609	1.589084e+07	0.000000	0.0	...	3.015376e-
11	1.613502e+06	0.0	1.365798e+06	162874.009766	16478.280273	0.0	17001.599609	6.711469e+05	0.000000	0.0	...	2.445044e-
12	1.689890e+06	0.0	1.985367e+06	162874.009766	35649.040039	0.0	17001.599609	4.251038e+06	56275.199219	0.0	...	3.073465e-
13	1.111906e+06	0.0	1.742511e+06	158520.884766	35649.040039	0.0	17001.599609	5.865088e+06	70704.599609	0.0	...	3.195146e-
14	7.638848e+04	0.0	1.742511e+06	39864.718750	35649.040039	0.0	0.000000	5.865088e+06	165284.281250	0.0	...	3.389930e-
15	4.391082e+05	0.0	8.200814e+05	12885.599609	19170.759766	0.0	0.000000	5.653898e+06	165284.281250	0.0	...	1.687436e-
16	3.627197e+05	0.0	2.005121e+05	12885.599609	0.000000	0.0	0.000000	2.338308e+06	109009.085938	0.0	...	1.059016e-
17	3.627197e+05	0.0	1.903338e+05	12885.599609	0.000000	0.0	0.000000	2.623732e+05	94579.683594	0.0	...	1.947840e-
18	3.627197e+05	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.623732e+05	0.000000	0.0	...	0.000000e-
19	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	0.0	...	0.000000e-
20	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	0.0	...	0.000000e-
21	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	0.0	...	0.000000e-
22	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	0.0	...	0.000000e-
23	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	0.0	...	0.000000e-
24	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	0.0	...	0.000000e-
25	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	0.0	...	0.000000e-

26 rows × 184 columns

```
DIREITO_DE_USOIRRIG_ord_ANOS_MUN = dados_transpostos_irrig #Redefinindo o df
conj_anos_irrig = set(DIREITO_DE_USOIRRIG_ord['ANOS_VIGENTES'])#Obtendo os anos vigentes
anos_irrig = list(conj_anos_irrig)# Listando os anos vigentes sem repetições
DIREITO_DE_USOIRRIG_ord_ANOS_MUN.insert(0,'ANOS_VIGENTES',anos_irrig,True) #Reinserindo os anos vigentes
```

DIREITO_DE_USOIRRIG_ord_ANOS_MUN

	ANOS_VIGENTES	ABAIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADIA	...	
0	2005	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00
1	2006	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00
2	2007	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00
3	2008	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00
4	2009	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00
5	2010	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00
6	2011	1.216208e+06	0.0	3.788980e+05	0.000000	0.000000	0.0	0.000000	1.120000e+04	0.000000	...	0.00
7	2012	1.216208e+06	0.0	3.788980e+05	0.000000	0.000000	0.0	0.000000	1.589588e+07	0.000000	...	9.11
8	2013	1.216208e+06	7104.0	3.788980e+05	0.000000	0.000000	0.0	0.000000	1.590204e+07	0.000000	...	9.11
9	2014	1.794192e+06	7104.0	6.319300e+05	4353.120117	0.000000	0.0	0.000000	1.590204e+07	0.000000	...	3.01
10	2015	1.613502e+06	0.0	2.530340e+05	135694.890625	0.000000	0.0	17001.599609	1.589084e+07	0.000000	...	3.01
11	2016	1.613502e+06	0.0	1.365798e+06	162874.009766	16478.280273	0.0	17001.599609	6.711469e+05	0.000000	...	2.44
12	2017	1.686989e+06	0.0	1.985367e+06	162874.009766	35649.040039	0.0	17001.599609	4.251038e+06	56275.199219	...	3.01
13	2018	1.111906e+06	0.0	1.742511e+06	158520.884766	35649.040039	0.0	17001.599609	5.8665088e+06	70704.599609	...	3.16
14	2019	7.638848e+04	0.0	1.742511e+06	39864.718750	35649.040039	0.0	0.000000	5.8665088e+06	165284.281250	...	3.38
15	2020	4.391082e+05	0.0	8.200814e+05	12885.599609	19170.759766	0.0	0.000000	5.653898e+06	165284.281250	...	1.68
16	2021	3.627197e+05	0.0	2.005121e+05	12885.599609	0.000000	0.0	0.000000	2.338308e+06	109009.085938	...	1.05
17	2022	3.627197e+05	0.0	1.903338e+05	12885.599609	0.000000	0.0	0.000000	2.623732e+05	94579.683594	...	1.94
18	2023	3.627197e+05	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.623732e+05	0.000000	...	0.00
19	2024	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	...	0.00
20	2025	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	...	0.00
21	2026	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	...	0.00
22	2027	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	...	0.00
23	2028	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	...	0.00
24	2029	0.000000e+00	0.0	1.903338e+05	0.000000	0.000000	0.0	0.000000	2.156482e+05	0.000000	...	0.00
25	2030	0.000000e+00	0.0	0.000000e+00	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	...	0.00

26 rows × 185 columns

O mesmo procedimento foi realizado nas demais finalidades de uso de água bruta para observação da evolução dos volumes de cada município ao longo do tempo.

DIREITO_DE_USO_AQUIC_ord_ANOS_MUN.head()														
	ANOS_VIGENTES	ABAIIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	...	TRAIRI	TURURU	UBAJARA
0	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
1	2006	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
2	2007	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
3	2008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
4	2009	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

5 rows × 185 columns

DIREITO_DE_USO_AB_HUM_ord_ANOS_MUN.head()														
	ANOS_VIGENTES	ABAIIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	...	TRAIRI	TURURU	UBAJARA
0	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
1	2006	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	107520.0	0.0
2	2007	0.0	120960.0	0.0	0.0	0.0	201600.0	0.0	0.0	0.0	...	0.0	107520.0	0.0
3	2008	0.0	120960.0	0.0	0.0	0.0	201600.0	0.0	0.0	0.0	...	0.0	107520.0	0.0
4	2009	0.0	120960.0	0.0	0.0	8064.0	201600.0	0.0	0.0	0.0	...	0.0	107520.0	0.0

5 rows × 185 columns

DIREITO_DE_USO_DESS_ord_ANOS_MUN.head()														
	ANOS_VIGENTES	ABAIIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	...	TRAIRI	TURURU	UBAJARA
0	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
1	2006	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
2	2007	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
3	2008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
4	2009	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

5 rows × 185 columns

DIREITO_DE_USO_IND_ord_ANOS_MUN.head()														
	ANOS_VIGENTES	ABAIIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	...	TRAIRI	TURURU	UBAJAF
DIREITO_DE_USO_SERV_COM_ord_ANOS_MUN.head()														
	ANOS_VIGENTES	ABAIIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	...	TRAIRI	TURURU	UBAJARA
0	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
1	2006	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
2	2007	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
3	2008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
4	2009	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

5 rows × 185 columns

Dando seguimento, nas finalidades irrigação e aquicultura, foi obtida a evolução da densidade de volume por área de imóveis rurais.

```
#OBSERVANDO A EVOLUÇÃO DA DENSIDADE DE VOLUME EM RELAÇÃO AS ÁREAS DE IMÓVEIS RURAIS AO LONGO DO TEMPO.
DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens = DIREITO_DE_USOIRRIG_ord_ANOS_MUN # Definindo um novo df

#MOVIMENTAR COLUNAS QUE NÃO FARÃO PARTE DOS CÁLCULOS
REITO_DE_USOIRRIG_ord_ANOS_MUN_dens = DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens.drop(columns='ANOS_VIGENTES')# Removendo as colunas

DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens_T = DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens.T #Gerando um df transposto

DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens_T
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19
ABAIARA	0.0	0.0	0.0	0.0	0.0	0.0	1216208.0	1216208.0	1216208.0	1.794192e+06	...	362719.687500	362719.687500	362719.6875	0.0000
ACARAPE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7104.0	7.104000e+03	...	0.000000	0.000000	0.0000	0.0000
ACARAU	0.0	0.0	0.0	0.0	0.0	0.0	378896.0	378896.0	378896.0	6.319300e+05	...	200512.109375	190333.812500	190333.8125	190333.8125
ACOPIARA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.353120e+03	...	12885.599609	12885.599609	0.0000	0.0000
AIUABA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000e+00	...	0.000000	0.000000	0.0000	0.0000
...
URUBURETAMA	0.0	0.0	0.0	0.0	0.0	0.0	108960.0	108960.0	123744.0	1.483680e+05	...	294010.375000	294010.375000	0.0000	0.0000
URUOCA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000e+00	...	0.000000	0.000000	0.0000	0.0000
VARJOTA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	60000.0	2.451120e+05	...	107558.437500	107558.437500	0.0000	0.0000
VARZEA ALEGRE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000e+00	...	13919.040039	13919.040039	0.0000	0.0000
VICOSA DO CEARA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000e+00	...	77545.609375	0.000000	0.0000	0.0000

184 rows × 26 columns

```
#INSERINDO A ÁREA DOS IMÓVEIS RURAIS (2005) DOS MUNÍCIPIOS NA IRRIGAÇÃO
lista_AREA_HA_IMOVEIS_2005 = list(IBGE_IPECE['AREA_HA_IMOVEIS_2005'])#Gerando uma lista a partir de uma coluna do df 'IBGE_IPECE'

DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens_T['AREA_HA_IMOVEIS_2005'] = lista_AREA_HA_IMOVEIS_2005#Inserindo uma lista no df transposto

DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens_T
```

8	9	...	17	18	19	20	21	22	23	24	25	AREA_HA_IMOVEIS_2005
18.0	1.794192e+06	...	362719.687500	362719.6875	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	10810.2000
14.0	7.104000e+03	...	0.000000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	8230.0000
16.0	6.319300e+05	...	190333.812500	190333.8125	190333.8125	190333.8125	190333.8125	190333.8125	190333.8125	190333.8125	0.0	44914.1000
0.0	4.353120e+03	...	12885.599609	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	133636.9500
0.0	0.000000e+00	...	0.000000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	131668.1000
...
14.0	1.483680e+05	...	294010.375000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	12368.0000
0.0	0.000000e+00	...	0.000000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	38063.6000
10.0	2.451120e+05	...	107558.437500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	6042.1000
0.0	0.000000e+00	...	13919.040039	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	64269.7425
0.0	0.000000e+00	...	0.000000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	89404.1800

```
colum_names_irrig = DIREITO_DE_USOIRRIG_ord_ANOS_MUN_dens_T.columns.tolist() #Obtendo a Lista com os nomes das colunas
colum_names_irrig.remove('AREA_HA_IMOVEIS_2005') #Removendo a coluna que não participará dos cálculos

print(colum_names_irrig)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
```

Calculando a densidade de cada volume anual pela área dos imóveis rurais de seus respectivos município em m³/ha

#CALCULANDO A DENSIDADE DE CADA VOLUME ANUAL PELA ÁREA DOS IMÓVEIS RURAIS DE SEUS RESPECTIVOS MUNICÍPIO EM M ³ /ha																									
DF_DENS_IRRIG_T =DIREITO_DE_USO_IRRIG_ord_ANOS_MUN_dens_T[colum_names_irrig].div(DIREITO_DE_USO_IRRIG_ord_ANOS_MUN_dens_T['AREA_H'])																									
DF_DENS_IRRIG_T																									
0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21		
ABAIARA	0.0	0.0	0.0	0.0	0.0	112.505597	112.505597	112.505597	165.972137	...	33.553467	33.553467	33.553467	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000		
ACARAPE	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.863183	0.863183	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000		
ACARAU	0.0	0.0	0.0	0.0	0.0	8.436015	8.436015	8.436015	14.069746	...	4.464347	4.237730	4.237730	4.23773	4.23773	4.23773	4.23773	4.23773	4.23773	4.23773	4.23773	4.23773	4.23773		
ACOPIARA	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.032574	...	0.096422	0.096422	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
AIUABA	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
...	
URUBURETAMA	0.0	0.0	0.0	0.0	0.0	0.0	8.809832	8.809832	10.005175	11.996119	...	23.771861	23.771861	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
URUOCA	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
VARJOTA	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	9.930322	40.567352	...	17.801499	17.801499	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
VARZEA ALEGRE	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.216572	0.216572	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VICOSA DO CEARA	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.867360	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

184 rows × 26 columns

DF_DEN_IRRIG =DF_DENS_IRRIG_T.T[[transpondo de volta]																								
DF_DEN_IRRIG																								
0	1	2	3	4	5	6	7	8	9	...	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ABAIARA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ACARAPE	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ACARAU	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ACOPIARA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AIUABA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ALCANTARA\$	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ALTANEIRA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ALTO SANTO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AMONTADA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ANTONINA DO NORTE	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
... TRAIRI	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
TURURU	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
UBAJ	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0	1	2	3	4	5	6	7	8	9	...	10	11	12	13	14	15	16	17	18	19	20	21	22	23

26 rows × 184 columns

```
conj_anos_irrig = set(DIREITO_DE_USO_IRRIG_ord['ANOS_VIGENTES'])
anos_irrig = list(conj_anos_irrig)
DF_DEN_IRRIG.insert(0,'ANOS_VIGENTES',anos_irrig,True) # Reinserindo a coluna com os anos
```

```
DENS_IRRIG_IMOV_RUR_ANOS = DF_DEN_IRRIG # Gerando um df
```

```
DENS_IRRIG_IMOV_RUR_ANOS
```

	ANOS_VIGENTES	ABAIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	...	TRAIRI	TURURU
0	2005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000
1	2006	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000
2	2007	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000
3	2008	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000
4	2009	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000
5	2010	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000
6	2011	112.505597	0.000000	8.436015	0.000000	0.000000	0.0	0.00000	0.158800	0.000000	...	0.000000	5.31022
7	2012	112.505597	0.000000	8.436015	0.000000	0.000000	0.0	0.00000	222.545669	0.000000	...	15.801952	40.612308
8	2013	112.505597	0.863183	8.436015	0.000000	0.000000	0.0	0.00000	225.468801	0.000000	...	15.801952	40.612308
9	2014	165.972137	0.863183	14.069746	0.032574	0.000000	0.0	0.00000	225.468801	0.000000	...	52.182898	40.612308
10	2015	149.257322	0.000000	5.633732	1.016898	0.000000	0.0	5.02545	225.310001	0.000000	...	52.182898	35.302286
11	2016	149.257322	0.000000	30.409103	1.218780	0.125150	0.0	5.02545	9.515926	0.000000	...	42.312952	0.000000
12	2017	158.323657	0.000000	44.203641	1.218780	0.270749	0.0	5.02545	60.273791	0.677537	...	53.188151	0.000000
13	2018	102.857116	0.000000	38.796525	1.186205	0.270749	0.0	5.02545	83.158773	0.851262	...	55.293927	0.000000
14	2019	7.0668333	0.000000	38.796525	0.298306	0.270749	0.0	0.00000	83.158773	1.989973	...	58.664781	0.000000
15	2020	40.619602	0.000000	18.258885	0.096422	0.145599	0.0	0.00000	80.164380	1.989973	...	29.202101	0.000000
16	2021	33.553467	0.000000	4.464347	0.096422	0.000000	0.0	0.00000	33.153939	1.312437	...	18.326901	0.000000
17	2022	33.553467	0.000000	4.237730	0.096422	0.000000	0.0	0.00000	3.720085	1.138711	...	3.370854	0.000000
18	2023	33.553467	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.720085	0.000000	...	0.000000	0.000000
19	2024	0.000000	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.057591	0.000000	...	0.000000	0.000000
20	2025	0.000000	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.057591	0.000000	...	0.000000	0.000000
21	2026	0.000000	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.057591	0.000000	...	0.000000	0.000000
22	2027	0.000000	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.057591	0.000000	...	0.000000	0.000000
23	2028	0.000000	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.057591	0.000000	...	0.000000	0.000000
24	2029	0.000000	0.000000	4.237730	0.000000	0.000000	0.0	0.00000	3.057591	0.000000	...	0.000000	0.000000
25	2030	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	...	0.000000	0.000000

26 rows × 185 columns

Na finalidade de uso aquicultura a densidade do volume pela área dos imóveis rurais foi obtida com a mesma trajetória utilizada para irrigação.

DENS_AQUIC_IMOV_RUR_ANOS														
ANOS_VIGENTES	ABAIARA	ACARAPE	ACARAU	ACOPIARA	AIUABA	ALCANTARAS	ALTANEIRA	ALTO SANTO	AMONTADA	... TRAIRI	TURURU	UBAJAF	C	
0	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
1	2006	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
2	2007	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
3	2008	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
4	2009	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
5	2010	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
6	2011	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
7	2012	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
8	2013	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
9	2014	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
10	2015	0.0	0.0	0.0	0.0	0.0	0.0	0.0 15.053071	0.0 ...	0.0	0.0	0.0	C	
11	2016	0.0	0.0	0.0	0.0	0.0	0.0	0.0 15.053071	0.0 ...	0.0	0.0	0.0	C	
12	2017	0.0	0.0	0.0	0.0	0.0	0.0	0.0 15.053071	0.0 ...	0.0	0.0	0.0	C	
13	2018	0.0	0.0	0.0	0.0	0.0	0.0	0.0 16.836480	0.0 ...	0.0	0.0	0.0	C	
14	2019	0.0	0.0	0.0	0.0	0.0	0.0	0.0 1.783410	0.0 ...	0.0	0.0	0.0	C	
15	2020	0.0	0.0	0.0	0.0	0.0	0.0	0.0 1.783410	0.0 ...	0.0	0.0	0.0	C	
16	2021	0.0	0.0	0.0	0.0	0.0	0.0	0.0 1.783410	0.0 ...	0.0	0.0	0.0	C	
17	2022	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
18	2023	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
19	2024	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
20	2025	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
21	2026	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
22	2027	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
23	2028	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
24	2029	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	
25	2030	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0 ...	0.0	0.0	0.0	C	

26 rows × 185 columns

Após a exploração das outras finalidades de uso, como o intuito inicial do projeto é ter uma forma de ‘ver’ como está a situação do uso regulamentado de água bruta nos municípios, uma espécie de autoavaliação, pois é sabido que muito foi evoluído em termos ampliar a regulamentação do uso flexibilizando exigências e custos, mas o quanto e onde ainda é necessário avançar?

Colocado este questionamento, volta-se o foco para o uso essencial da água, ou seja, na finalidade abastecimento humano, embora as demais finalidades possam apresentar correlações, nessa análise vamos nos dedicar a essa finalidade sem considerar as demais e ver os resultados.

5. Criação de Modelos de Machine Learning

A ideia inicial é observar como está a abrangência da regularização de uso da água e para tal buscou-se outros dados sobre os municípios. Ao explorar os dados percebeu-se que alguns deles têm evolução cronológica

O objetivo principal era identificar os municípios com maior carência na regularização do uso, então a opção por regressão linear foi se mostrando aos poucos menos atrativa, pois uma classificação da situação atual seria um ponto inicial mais relevante do que uma predição em si. Com esse olhar mais alinhado com os objetivos do diagnóstico, foi percebido que uma árvore de decisão poderia revelar o retrato buscado.

Para tal, uma visão geral dos dados levantados corrobora com o vislumbre de quais deles poderiam ser os mais representativos.

Unificando os dataframes gerados das finalidades de uso pelo município, inclusive os dados comparativos já incluídos para a finalidade de uso abastecimento humano, foi obtido um dataframe com 55 colunas e 184 linhas

```
#UNIFICANDO OS DATAFRAMES GERADOS PARA CADA FINALIDADE DE USO PELO MUNICÍPIO
DIREITO_DE_USO_FINALIDADES_MAX_AQUIC_IRRIG = pd.merge(DIREITO_DE_USO_AQUIC_MAX, DIREITO_DE_USO_IRRIG_MAX, how = 'inner', on = 'MUNICIPIO')
DIREITO_DE_USO_FINALIDADES_MAX_DESS_AB_HUM = pd.merge(DIREITO_DE_USO_DESS_MAX, DIREITO_DE_USO_AB_HUM_MAX, how = 'inner', on = 'MUNICIPIO')
DIREITO_DE_USO_FINALIDADES_MAX_OUTROS_SERV_COM = pd.merge( DIREITO_DE_USO_OUTROS_MAX, DIREITO_DE_USO_SERV_COM_MAX, how = 'inner', on = 'MUNICIPIO')
DIREITO_DE_USO_FINALIDADES_MAX_IND_OUTROS_SERV_COM = pd.merge(DIREITO_DE_USO_FINALIDADES_MAX_OUTROS_SERV_COM, DIREITO_DE_USO_IND_OUTROS_MAX, how = 'inner', on = 'MUNICIPIO')
DIREITO_DE_USO_FINALIDADES_MAX_AQUIC_IRRIG_DESS_AB_HUM = pd.merge(DIREITO_DE_USO_FINALIDADES_MAX_AQUIC_IRRIG, DIREITO_DE_USO_FINALIDADES_MAX_DESS_AB_HUM, how = 'inner', on = 'MUNICIPIO')
DIREITO_DE_USO_FINALIDADES_MAX= pd.merge(DIREITO_DE_USO_FINALIDADES_MAX_IND_OUTROS_SERV_COM,DIREITO_DE_USO_FINALIDADES_MAX_AQUIC_IRRIG, how = 'inner', on = 'MUNICIPIO')
```

DIREITO_DE_USO_FINALIDADES_MAX.info()			
	# Column	Non-Null Count	Dtype
0	MUNICIPIO	184	non-null object
1	ANO_VOL_MAX_VIGENTE_OUTROS	184	non-null int64
2	VOL_SUPERF_OUTROS_M3	184	non-null float32
3	VOL_SUBTER_OUTROS_M3	184	non-null float64
4	VOLUME_MAX_OUTROS_MUN_M3	184	non-null float64
5	VOLUME_MUN_OUTROS2020_M3	184	non-null float64
6	DIF_PROP_VOL_MAX_2020_OUTROS	184	non-null float64
7	ANO_VOL_MAX_VIGENTE_SERV_COM	184	non-null int64
8	VOL_SUPERF_SERV_COM_M3	184	non-null float32
9	VOL_SUBTER_SERV_COM_M3	184	non-null float64
10	VOLUME_MAX_SERV_COM_MUN_M3	184	non-null float64
11	VOLUME_MUN_SERV_COM2020_M3	184	non-null float64
12	DIF_PROP_VOL_MAX_VOL_2020_SERV_COM	184	non-null float64
13	ANO_VOL_MAX_VIGENTE_IND	184	non-null int64
14	VOL_SUPERF_IND_M3	184	non-null float32
15	VOL_SUBTER_IND_M3	184	non-null float64
16	VOLUME_MAX_IND_MUN_M3	184	non-null float64
17	VOLUME_MUN_IND2020_M3	184	non-null float64
18	DIF_PROP_VOL_MAX_VOL_2020_IND	184	non-null float64
19	ANO_VOL_MAX_VIGENTE_AQUIC	184	non-null int64
20	VOL_SUPERF_AQUIC_M3	184	non-null float32
21	VOL_SUBTER_AQUIC_M3	184	non-null float64
22	VOLUME_MAX_AQUIC_MUN_M3	184	non-null float64
23	VOLUME_MUN_AQUIC2020_M3	184	non-null float64
24	DIF_PROP_VOL_MAX_VOL_2020_AQUIC	184	non-null float64
25	ANO_VOL_MAX_VIGENTE_IRRIG	184	non-null int64
26	VOL_SUPERF_IRRIG_M3	184	non-null float32
27	VOL_SUBTER_IRRIG_M3	184	non-null float64
28	VOLUME_MAX_IRRIG_MUN_M3	184	non-null float64
29	VOLUME_MUN_IRRIG2020_M3	184	non-null float64
30	DIF_PROP_VOL_MAX_VOL_2020_IRRIG	184	non-null float64
31	ANO_VOL_MAX_VIGENTE_DESS	184	non-null int64
32	VOL_SUPERF_DESS_M3	184	non-null float32
33	VOL_SUBTER_DESS_M3	184	non-null float64
34	VOLUME_MAX_DESS_MUN_M3	184	non-null float64
35	VOLUME_MUN_DESS2020_M3	184	non-null float64
36	DIF_PROP_VOL_MAX_VOL_2020_DESS	184	non-null float64
37	Name Municipio	184	non-null object
38	CD_IDEG	184	non-null int64
39	ANO_VOL_MAX_VIGENTE_AB_HUM	184	non-null int64
40	VOL_SUPERF_AB_HUM_M3	184	non-null float32
41	VOL_SUBTER_AB_HUM_M3	184	non-null float64
42	VOLUME_MAX_AB_HUM_MUN_M3	184	non-null float64
43	VOLUME_MUN_ABHUM2020_M3	184	non-null float64
44	DIF_PROP_VOL_MAX_VOL_2020_AB_HUM	184	non-null float64
45	POF_ESTIMADA_2020	184	non-null int64
46	POF_EST_ANO_VOL_MAX	184	non-null int32
47	L_HAB_DIA_ANO_VOL_MAX	184	non-null float64
48	L_HAB_DIA_ANO_VOL_2020	184	non-null float64
49	VOL_2020_MTN_QMS	184	non-null float64
50	P01_LMAX	184	non-null int64
51	P03_2020	184	non-null int64
52	P02_L2020_LMAX	184	non-null int64
53	PT_AB_HUM	184	non-null int64
54	REG_AB_HUM	184	non-null int64

dtypes: float32(7), float64(31), int32(1), int64(14), object(2)

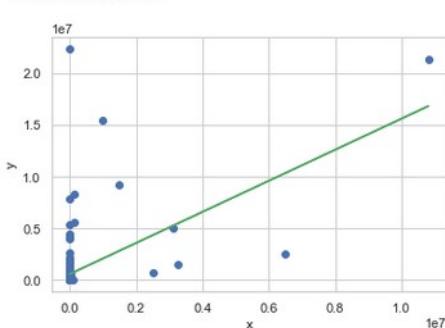
memory usage: 74.8+ KB

```
#BUSCANDO CORRELAÇÕES
x = DIREITO_DE_USO_FINALIDADES_MAX['VOLUME_MUN_AQUIC2020_M3']
y = DIREITO_DE_USO_FINALIDADES_MAX['VOLUME_MUN_IRRIG2020_M3']

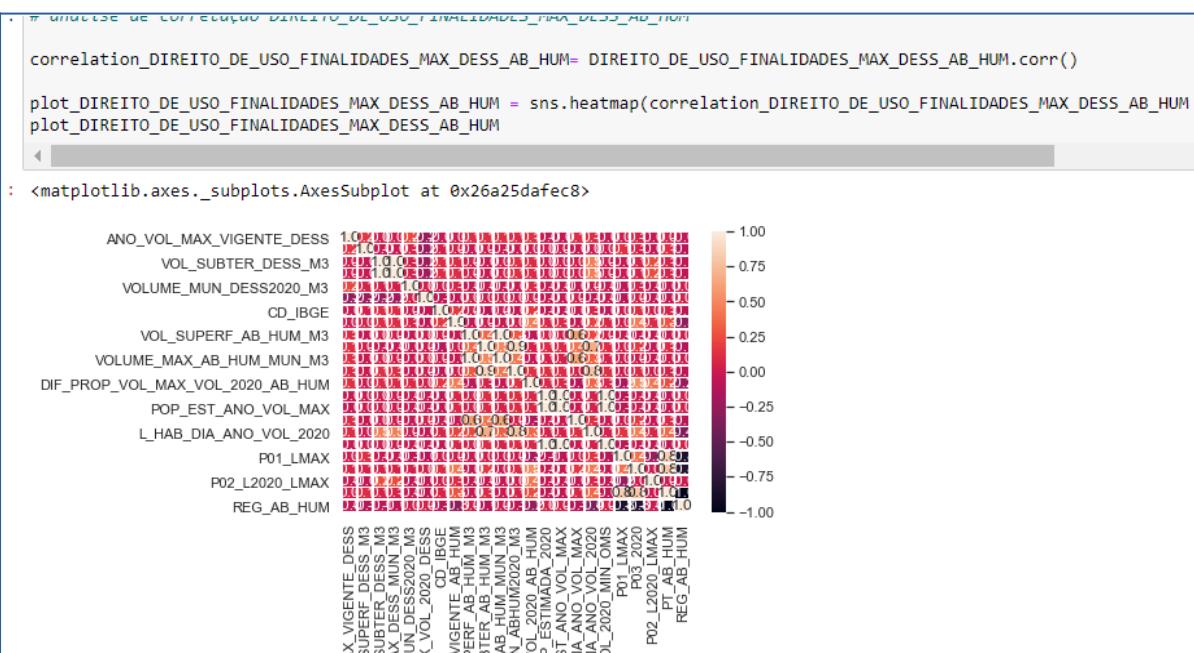
p1 = np.polyfit(x,y,1)

yfit = p1[0] *x +p1[1]
yresid = y-yfit
SQresid = sum(pow(yresid,2))
SQtotal = len(y) + np.var(y)
R2 = 1 - SQresid/SQtotal
print(p1)
print(R2)
plt.plot(x,y,'o')
plt.plot(x,np.polyval(p1,x), 'g')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

[1.50352332e+00 5.95302489e+05
-131.0950158724848



Especulando correlações entre as finalidades e/ou os dados externos.

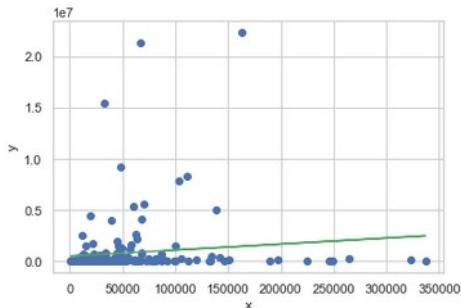


```
#BUSCANDO CORRELAÇÕES
Y = IBGE_IPECE['AREA_HA_IMOVEIS_2005']
X = DIREITO_DE_USO_FINALIDADES_MAX['VOLUME_MUN_IRRIG2020_M3']

p1 = np.polyfit(x,y,1)

yfit = p1[0] *x +p1[1]
yresid = y-yfit
SQresid = sum(pow(yresid,2))
SQtotal = len(y) + np.var(y)
R2 = 1 - SQresid/SQtotal
print(p1)
print(R2)
plt.plot(x,y,'o')
plt.plot(x,np.polyval(p1,x), 'g')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

[5.86058724e+00 5.21384382e+05]
-180.43183591603943
```



Retomando a observação dos dados já tratados, um parâmetro utilizado como comparativo foi o percentual de diferença do volume vigente em 2020 em relação ao máximo volume vigente. Os valores variam de -1 a 0, sendo que o zero indica que o volume vigente em 2020 coincide com o volume máximo. Para cada finalidade há esse comparativo, sendo que para o abastecimento humano, um elemento externo foi introduzido, o padrão OMS de uso de água diário ideal por habitante.

Mais alguns dados externos estão de ‘stand by’, mas por hora não serão utilizados.

A ideia é usar o scikit-learn, uma biblioteca para aprendizagem de máquina destinada a linguagem Python.

Gerando um dataframe com as ‘diferenças’ obtidas para as demais e o grau de regularização, lembrando que o grau varia de 1 a 10, mas no PowerBi foi adequado para 0 a 9, respectivamente. No abastecimento humano os valores mais baixos indicam situação mais confortável.

Nessa tentativa, foi utilizado o nível de regularização nas 10 classes definidas, e as diferenças entre os volume máximos e volumes vigentes em 2020 numa árvore de decisão para dados categóricos.

Convertendo variáveis categóricas em dummie.

#CONVERTENDO VARIÁVEIS CATEGÓRICAS EM DUMMIE										
DIREITO_DE_USO_DIF_getdummy=pd.get_dummies(data=DIREITO_DE_USO_DIF, columns=['REG_AB_HUM'])										
DIREITO_DE_USO_DIF_getdummy										
<hr/>										
REG_AB_HUM_2	REG_AB_HUM_3	REG_AB_HUM_4	REG_AB_HUM_5	REG_AB_HUM_6	REG_AB_HUM_7	REG_AB_HUM_8	REG_AB_HUM_9	REG_AB_HUM_10		
0	0	0	0	0	0	0	1	0		
0	0	0	0	0	0	0	0	1		
0	0	0	0	0	0	1	0	0		
0	0	0	0	0	0	0	1	0		
0	0	0	0	0	0	0	1	0		
...		
0	0	0	0	0	1	0	0	0		
0	0	0	0	0	0	0	1	0		
0	0	0	0	0	0	0	0	1		
0	0	0	0	0	0	0	1	0		
0	0	0	0	0	0	0	0	1		
0	0	0	0	0	0	0	0	1		

Separando o conjunto de treinamento

#SEPARANDO O CONJUNTO DE TREINAMENTO E O CONJUNTO DE TESTES
X = DIREITO_DE_USO_DIF_getdummy.drop('MUNICIPIO',axis=1)
y = DIREITO_DE_USO_DIF_getdummy['MUNICIPIO']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=101)
len(X_train)
128
len(X_test)
56

Ao usar 10 níveis de classificação nesse teste e sem usar parâmetros de controle ficou bem destacado o 'over-flow'

```
#ÁRVORE DE DECISÃO CATEGÓRICA
dtree = DecisionTreeClassifier(criterion='entropy',max_depth=6)
dtree.fit(X_train,y_train)
predictions = dtree.predict(X_test)

#EXPORTANDO A ÁRVORE EM FORMATO DE TEXTO
r = export_text(dtree, feature_names=X.columns.values.tolist ())
print(r)
```

--- DIF_PROP_VOL_MAX_VOL_2020_DESS <= -0.07
|--- class: ASSARE
|--- DIF_PROP_VOL_MAX_VOL_2020_DESS > -0.07
|--- class: ALTANEIRA
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG > -0.97
|--- DIF_PROP_VOL_MAX_VOL_2020_DESS <= -0.16
|--- class: LIMOEIRO DO NORTE
|--- DIF_PROP_VOL_MAX_VOL_2020_DESS > -0.16
|--- class: ACOPIARA
--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG > -0.84
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG <= -0.58
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG <= -0.69
|--- DIF_PROP_VOL_MAX_OUTROS_VOL_2020 <= -0.19
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG <= -0.77
|--- class: PARACURU
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG > -0.77
|--- class: CRATEUS
--- DIF_PROP_VOL_MAX_OUTROS_VOL_2020 > -0.19
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG <= -0.75
|--- class: ABAIARA

Foi realizada uma poda nos nós para comparação após a mesma

```
#SEPARANDO O CONJUNTO DE TREINAMENTO E O CONJUNTO DE TESTES
X_train, X_test, y_train, y_test = train_test_split(DIREITO_DE_USO_DIF.drop('MUNICIPIO',axis=1),DIREITO_DE_USO_DIF['MUNICIPIO'],t
|
```

len(X_train)
128

len(X_test)
56

```
#ÁRVORE DE DECISÃO CATEGÓRICA
dtree = DecisionTreeClassifier(criterion='entropy',max_depth=2)
dtree.fit(X_train,y_train)
predictions = dtree.predict(X_test)

#EXPORTANDO A ÁRVORE EM FORMATO DE TEXTO
r = export_text(dtree, feature_names=X.columns.values.tolist ())
print(r)
```

--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG <= -0.15
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG <= -0.84
|--- class: ACOPIARA
|--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG > -0.84
|--- class: ABAIARA
--- DIF_PROP_VOL_MAX_VOL_2020_IRRIG > -0.15
|--- DIF_PROP_VOL_MAX_OUTROS_VOL_2020 <= -0.03
|--- class: BARREIRA
|--- DIF_PROP_VOL_MAX_OUTROS_VOL_2020 > -0.03
|--- class: ALCANTARAS

Essa opção de árvore não foi convincente, então uma tentativa com a árvore de decisão com versão para dados contínuos.

Nessa etapa a quantidade de categorias de regularização no abastecimento humano definidas anteriormente foi agrupada em 03 faixas (0_a_3, 4_a_6, 7_a_9).

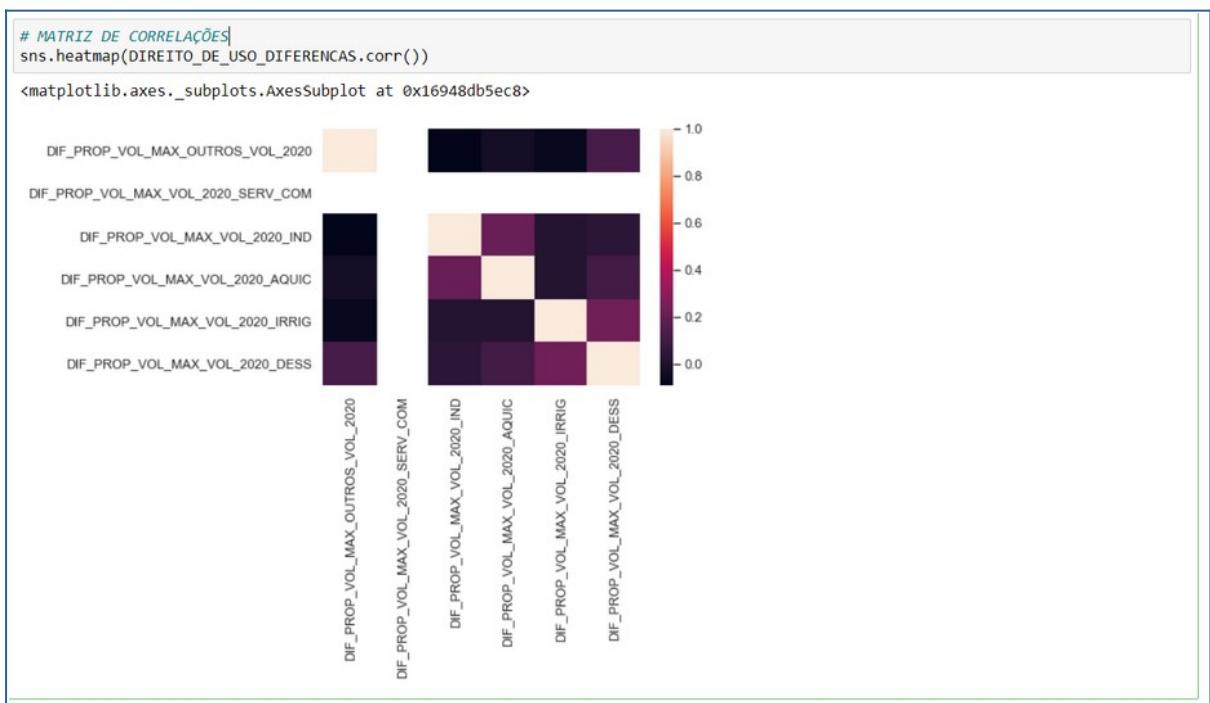
```
#CRIANDO O DATAFRAME PARA A CLASSIFICAÇÃO
DIREITO_DE_USO_DIFERENCIAS = DIREITO_DE_USO_FINALIDADES_MAX
#REMOVENDO COLUNAS QUE NÃO HÁ INTERESSE
conj_colunas_DIREITO_DE_USO_DIFERENCIAS=set(DIREITO_DE_USO_DIFERENCIAS.columns.values.tolist())#Criando uma lista com os nomes das
conj_colunas_DIREITO_DE_USO_DIFERENCIAS_mantidas = {'DIF_PROP_VOL_MAX_VOL_2020_DESS', 'DIF_PROP_VOL_MAX_VOL_2020_IRRIG', 'DIF_PROP_VOL_MAX_OUTROS_VOL_2020', 'DIF_PROP_VOL_MAX_VOL_2020_IND', 'DIF_PROP_VOL_MAX_VOL_2020_AQUIC', 'DIF_PROP_VOL_MAX_VOL_2020_SERV_COM', 'FAIXA_REG_AB_HUM'}
conj_colunas_remover = conj_colunas_DIREITO_DE_USO_DIFERENCIAS - conj_colunas_DIREITO_DE_USO_DIFERENCIAS_mantidas#Colunas a remover
#REMOVER COLUNAS SELECIONADAS
DIREITO_DE_USO_DIFERENCIAS = DIREITO_DE_USO_DIFERENCIAS.drop(columns=conj_colunas_remover)# Removendo as colunas
DIREITO_DE_USO_DIFERENCIAS .info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 0 to 183
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DIF_PROP_VOL_MAX_OUTROS_VOL_2020    184 non-null   float64
 1   DIF_PROP_VOL_MAX_VOL_2020_SERV_COM   184 non-null   float64
 2   DIF_PROP_VOL_MAX_VOL_2020_IND       184 non-null   float64
 3   DIF_PROP_VOL_MAX_VOL_2020_AQUIC    184 non-null   float64
 4   DIF_PROP_VOL_MAX_VOL_2020_IRRIG    184 non-null   float64
 5   DIF_PROP_VOL_MAX_VOL_2020_DESS    184 non-null   float64
 6   FAIXA_REG_AB_HUM                  184 non-null   object  
dtypes: float64(6), object(1)
memory usage: 15.6+ KB
```

Fazendo uma inspeção preliminar nas dimensões e presença de NAN.

```
#OBSERVANDO AS DIMENSÕES
DIREITO_DE_USO_DIFERENCIAS.shape
(184, 7)

#VERIFICANDO A PRESENÇA DE NAN
DIREITO_DE_USO_DIFERENCIAS.isnull().any()
DIF_PROP_VOL_MAX_OUTROS_VOL_2020      False
DIF_PROP_VOL_MAX_VOL_2020_SERV_COM    False
DIF_PROP_VOL_MAX_VOL_2020_IND        False
DIF_PROP_VOL_MAX_VOL_2020_AQUIC      False
DIF_PROP_VOL_MAX_VOL_2020_IRRIG      False
DIF_PROP_VOL_MAX_VOL_2020_DESS      False
FAIXA_REG_AB_HUM                     False
dtype: bool
```

```
#CHECANDO CORRELAÇÕES
sns.pairplot(data=DIREITO_DE_USO_DIFERENCIAS, hue = 'FAIXA_REG_AB_HUM')
C:\Users\55859\anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\55859\anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\55859\anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\55859\anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
```



```
#SEPARANDO VARIÁVEIS E FEATURES
target = DIREITO_DE_USO_DIFERENCIAS['FAIXA_REG_AB_HUM']
DIREITO_DE_USO_DIFERENCIAS = DIREITO_DE_USO_DIFERENCIAS.copy()
DIREITO_DE_USO_DIFERENCIAS1 = DIREITO_DE_USO_DIFERENCIAS1.drop('FAIXA_REG_AB_HUM', axis=1)
```

X = DIREITO_DE_USO_DIFERENCIAS1

```
# OBSERVANDO A VARIÁVEL  
target
```

```
0    De_7_a_9
1    De_7_a_9
2    De_7_a_9
3    De_7_a_9
4    De_7_a_9
      ...
179   De_7_a_9
180   De_7_a_9
181   De_7_a_9
182   De_7_a_9
183   De_7_a_9
Name: FAIXA_REG_AB_HUM, Length: 184, dtype: object
```

CODIFICAÇÃO A VÁRIAS

Dividindo o conjunto de dados em conjuntos de treinamento e teste e selecionando 20% de registros aleatoriamente para teste.

```
# FATIANDO O DATASET - 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(x , y, test_size = 0.2, random_state = 42)
print("Entrada de divisão de treinamento-", X_train.shape)
print("Testando entrada de divisão-", X_test.shape)

Entrada de divisão de treinamento- (147, 6)
Testando entrada de divisão- (37, 6)

# Defining the decision tree algorithm
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Classificador de árvore de decisão criado')

Decision Tree Classifier Created
```

Predizendo os valores dos dados de teste

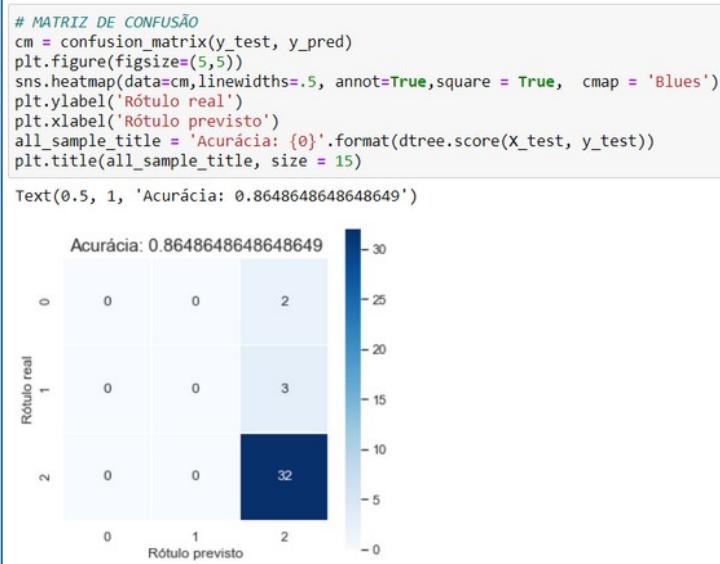
```
#PREDIZENDO OS VALORES DOS DADOS DE TESTE
y_pred = dtree.predict(X_test)
print("Relatório de classificação - \n", classification_report(y_test,y_pred))

Relatório de classificação - 
precision    recall   f1-score   support
          0       0.00     0.00      0.00       2
          1       0.00     0.00      0.00       3
          2       0.86     1.00      0.93      32

   accuracy                           0.86      37
  macro avg       0.29     0.33      0.31      37
weighted avg       0.75     0.86      0.80      37

C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Observando a matriz de confusão



Visualizando a árvore de decisão

```
#VISUALIZANDO A ÁRVORE DE DECISÃO
dec_tree = plot_tree(decision_tree=dtree, feature_names = DIREITO_DE_USO_DIFERENCIAS1.columns,
                     class_names =["De_0_a_3", "De_4_a_6", "De_7_a_9"] , filled = True , precision = 4, rounded = True)
```



Pôde-se ver aqui como a árvore é dividida, quais são os 'ginis' para os nós, os registros nesses nós e seus rótulos e que o modelo ainda não está como o esperado, a acurácia ficou em 86 %.

Requer nova tentativa:

```
DIREITO_DE_USO_DIFERENCIAS.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 0 to 183
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DIF_PROP_VOL_MAX_OUTROS_VOL_2020    184 non-null   float64
 1   DIF_PROP_VOL_MAX_VOL_2020_SERV_COM    184 non-null   float64
 2   DIF_PROP_VOL_MAX_VOL_2020_IND        184 non-null   float64
 3   DIF_PROP_VOL_MAX_VOL_2020_AQUIC      184 non-null   float64
 4   DIF_PROP_VOL_MAX_VOL_2020_IRRIG       184 non-null   float64
 5   DIF_PROP_VOL_MAX_VOL_2020_DESS       184 non-null   float64
 6   FAIXA_REG_AB_HUM                    184 non-null   object  
dtypes: float64(6), object(1)
memory usage: 15.6+ KB
```

Dividindo os dados para teste e treinamento

```
#DIVIDINDO OS DADOS PARA TESTE E TREINAMENTO
X_train, X_test, y_train, y_test = train_test_split(DIREITO_DE_USO_DIFERENCIAS.drop('FAIXA_REG_AB_HUM',axis=1),DIREITO_DE_USO_DIFERENCIAS['FAIXA_REG_AB_HUM'], test_size=0.2, random_state=42)

#VERIFICANDO AS FORMAS DOS DADOS:
X_train.shape,X_test.shape
((537, 8), (231, 8))

y_train.shape,y_test.shape
((537,), (231,))

((537,), (231,))
```

Nessa etapa a quantidade de categorias de regularização no abastecimento humano definidas anteriormente foi agrupada em 03 faixas (0_a_3, 4_a_6, 7_a_9).

```
DIREITO_DE_USO_AB_HUM_MAX.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 25 to 76147
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Nome Município    184 non-null    object  
 1   MUNICIPIO         184 non-null    object  
 2   CD_IBGE           184 non-null    int64  
 3   FINALIDADE_DE_USO 184 non-null    category
 4   ANOS_VIGENTES     184 non-null    int64  
 5   VOL_SUPERF_M3     184 non-null    float32 
 6   VOL_SUBTER_M3     184 non-null    float64 
 7   VOLUME_MAX_MUN_M3 184 non-null    float64 
 8   VOLUME_MUN_ABHUM2020_M3 184 non-null    float64 
 9   DIF_PROP_VOL_MAX_VOL_2020 184 non-null    float64 
 10  POP_ESTIMADA_2020 184 non-null    int64  
 11  POP_EST_ANO_VOL_MAX 184 non-null    int32  
 12  L_HAB_DIA_ANO_VOL_MAX 184 non-null    float64 
 13  L_HAB_DIA_ANO_VOL_2020 184 non-null    float64 
 14  VOL_2020_MIN_OMS 184 non-null    float64 
 15  P01_LMAX          184 non-null    int64  
 16  P03_2020          184 non-null    int64  
 17  P02_L2020_LMAX    184 non-null    int64  
 18  PT_AB_HUM         184 non-null    int64  
 19  REG_AB_HUM        184 non-null    int64  
dtypes: category(1), float32(1), float64(7), int32(1), int64(8), object(2)
memory usage: 27.8+ KB
```

```
#CRIANDO UM DATAFRAME PARA USO NA ÁRVORE COM OS DADOS LEVANTADOS NA FINALIDADE ABASTECIMENTO HUMANO
DIREITO_DE_USO_AB_HUM_MODEL_TREE = DIREITO_DE_USO_AB_HUM_MAX

DIREITO_DE_USO_AB_HUM_MODEL_TREE
```

AX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	P02_L2020_LMAX	PT_AB_HUM	REG_AB_HUM	NIVEL
155	32.252883	30.356156	475897.95	-2	-2	0	-4	9	2
563	28.660094	0.000000	603695.40	-2	-3	0	-5	10	2
300	99.019230	33.771552	2533625.60	-1	-2	0	-3	8	2
350	77.952922	65.619207	2187412.15	-2	-2	0	-4	9	2
312	137.373136	0.000000	702343.95	0	-3	0	-3	8	2
...
375	452.864807	0.000000	884906.00	1	-3	0	-2	7	2
336	61.325685	60.099171	558687.25	-2	-2	0	-4	9	2
337	665.891378	0.000000	741610.65	-1	-3	0	-4	9	2
798	78.625664	25.875177	1642255.45	-2	-2	0	-4	9	2
399	7.862035	2.327634	2465611.50	-2	-2	0	-4	9	2

```

: #REDUZINDO OS NÍVEIS DE REGULARIZAÇÃO DA FINALIDADE ABASTECIMENTO HUMANO A 03 FAIXAS
lista_faixa_reg_ab_hum = []
faixa_ab_hum = DIREITO_DE_USO_AB_HUM_MODEL_TREE['REG_AB_HUM']
for abhum in faixa_ab_hum:
    if abhum <= 4:
        regabhum = 0
        lista_faixa_reg_ab_hum.append(regabhum)
    if abhum >= 6:
        regabhum = 2
        lista_faixa_reg_ab_hum.append(regabhum)
    if abhum > 4 and abhum < 6 :
        regabhum = 1
        lista_faixa_reg_ab_hum.append(regabhum)
DIREITO_DE_USO_AB_HUM_MODEL_TREE['NIVEL'] = lista_faixa_reg_ab_hum

: DIREITO_DE_USO_AB_HUM_MODEL_TREE
: 
```

IAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	P02_L2020_LMAX	PT_AB_HUM	REG_AB_HUM	NIVEL
155	32.252883	30.356156	475897.95	-2	-2	0	-4	9	2
563	28.660094	0.000000	603695.40	-2	-3	0	-5	10	2
900	99.019230	33.771552	2533625.60	-1	-2	0	-3	8	2
350	77.952922	65.619207	2187412.15	-2	-2	0	-4	9	2
812	137.373136	0.000000	702343.95	0	-3	0	-3	8	2
...
375	452.864807	0.000000	884906.00	1	-3	0	-2	7	2
636	61.325685	60.099171	558687.25	-2	-2	0	-4	9	2
037	665.891378	0.000000	741610.65	-1	-3	0	-4	9	2
798	78.625664	25.875177	1642255.45	-2	-2	0	-4	9	2
399	7.862035	2.327634	2465611.50	-2	-2	0	-4	9	2

Removendo colunas que não há interesse, são colunas sem expressividade para a construção do modelo.

```

#REMOVENDO COLUNAS QUE NÃO HÁ INTERESSE
conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE=set(DIREITO_DE_USO_AB_HUM_MODEL_TREE.columns.values.tolist())#Criando uma Lista com todas as colunas do DataFrame
conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE_mantidas = ['DIF_PROP_VOL_MAX_VOL_2020','L_HAB_DIA_ANO_VOL_2020','L_HAB_DIA_ANO_VOL_MAX','VOL_SURFACE_M3','VOL_SUBTER_M3','VOLUME_MAX_MUN_M3','VOLUME_MUN_ABHUM2020_M3','POP_ESTIMADA_2020','POP_EST_ANO_VOL_MAX','NIVEL']#Colunas que serão mantidas
conj_colunas_remover = conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE - conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE_mantidas#Colunas que serão removidas

#REMOVER COLUNAS SELECIONADAS
DIREITO_DE_USO_AB_HUM_MODEL_TREE = DIREITO_DE_USO_AB_HUM_MODEL_TREE.drop(columns=conj_colunas_remover)# Removendo as colunas selecionadas

: DIREITO_DE_USO_AB_HUM_MODEL_TREE.info()
: 
```

#	Column	Non-Null Count	Dtype
0	VOL_SURFACE_M3	184	float32
1	VOL_SUBTER_M3	184	float64
2	VOLUME_MAX_MUN_M3	184	float64
3	VOLUME_MUN_ABHUM2020_M3	184	float64
4	DIF_PROP_VOL_MAX_VOL_2020	184	float64
5	POP_ESTIMADA_2020	184	int64
6	POP_EST_ANO_VOL_MAX	184	int32
7	L_HAB_DIA_ANO_VOL_MAX	184	float64
8	L_HAB_DIA_ANO_VOL_2020	184	float64
9	VOL_2020_MIN_OMS	184	float64
10	NIVEL	184	int64

dtypes: float32(1), float64(7), int32(1), int64(2)
memory usage: 19.9 KB

Renomeando colunas para melhorar a identidade no contexto.

```
REE = DIREITO_DE_USO_AB_HUM_MODEL_TREE.rename(columns={'VOL_SUPERF_M3': 'VOL_MAX_SUPERF_M3', 'VOL_SUBTER_M3': 'VOL_MAX_SUBTER_M3',})
```

Padronizado a formatação do tipo de coluna cujos valores que têm afinidade

```
: #PADRONIZADO A FORMATAÇÃO DO TIPO DE COLUNA CUJOS VALORES QUE TÊM AFINIDADE
DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOL_MAX_SUBTER_M3 = DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOL_MAX_SUBTER_M3.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOLUME_MAX_MUN_M3 = DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOLUME_MAX_MUN_M3.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOLUME_MUN_ABHUM2020_M3 = DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOLUME_MUN_ABHUM2020_M3.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.DIF_PROP_VOL_MAX_VOL_2020 = DIREITO_DE_USO_AB_HUM_MODEL_TREE.DIF_PROP_VOL_MAX_VOL_2020.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_MAX = DIREITO_DE_USO_AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_MAX.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_2020 = DIREITO_DE_USO_AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_2020.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.POP_ESTIMADA_2020 = DIREITO_DE_USO_AB_HUM_MODEL_TREE.POP_ESTIMADA_2020.astype('int32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOL_2020_MIN_OMS = DIREITO_DE_USO_AB_HUM_MODEL_TREE.VOL_2020_MIN_OMS.astype('float32')
DIREITO_DE_USO_AB_HUM_MODEL_TREE.NIVEL = DIREITO_DE_USO_AB_HUM_MODEL_TREE.NIVEL.astype('object')

: DIREITO_DE_USO_AB_HUM_MODEL_TREE.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 25 to 76147
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   VOL_MAX_SUPERF_M3    184 non-null   float32 
 1   VOL_MAX_SUBTER_M3    184 non-null   float32 
 2   VOLUME_MAX_MUN_M3    184 non-null   float32 
 3   VOLUME_MUN_ABHUM2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020    184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX  184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_OMS    184 non-null   float32 
 10  NIVEL              184 non-null   object  
dtypes: float32(8), int32(2), object(1)
memory usage: 14.1+ KB
```

Criando um dataframe

```
#CRIANDO UM DATAFRAME
AB_HUM_TREE_ = DIREITO_DE_USO_AB_HUM_MODEL_TREE
```

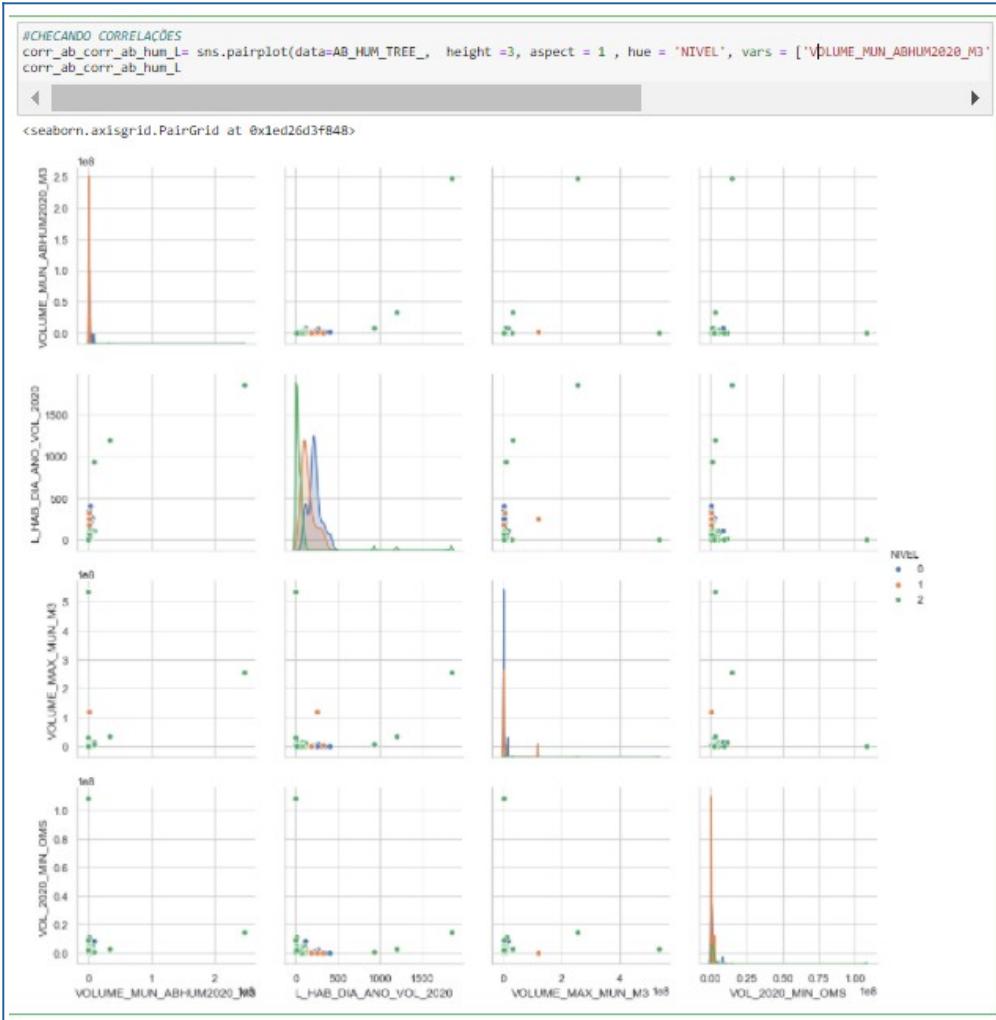
Fazendo uma inspeção preliminar nas dimensões e presença de NAN.

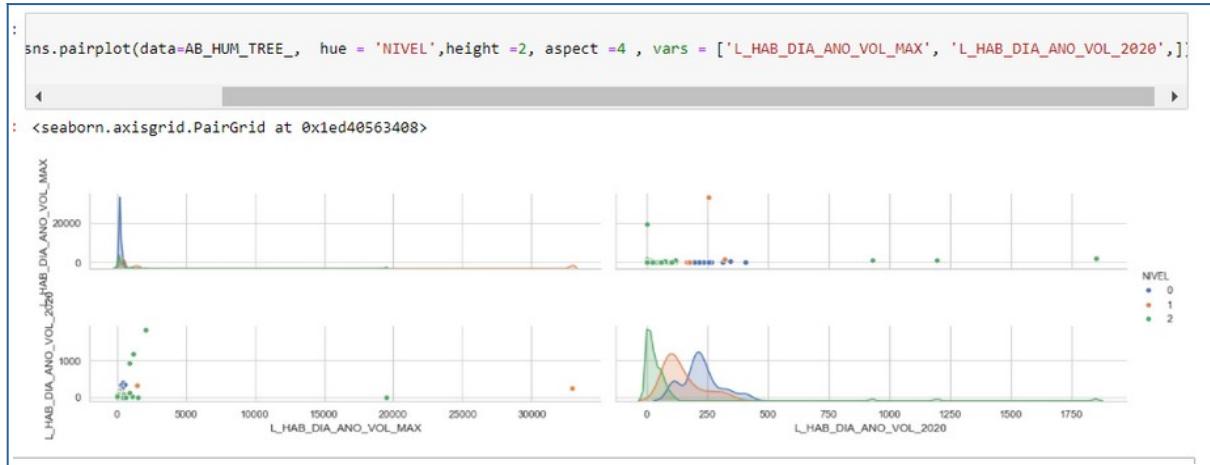
```
#OBSERVANDO AS DIMENSÕES
AB_HUM_TREE_.shape
(184, 11)

#VERIFICANDO A PRESENÇA DE NAN
AB_HUM_TREE_.isnull().any()

VOL_MAX_SUPERF_M3      False
VOL_MAX_SUBTER_M3       False
VOLUME_MAX_MUN_M3        False
VOLUME_MUN_ABHUM2020_M3  False
DIF_PROP_VOL_MAX_VOL_2020  False
POP_ESTIMADA_2020        False
POP_EST_ANO_VOL_MAX     False
L_HAB_DIA_ANO_VOL_MAX   False
L_HAB_DIA_ANO_VOL_2020   False
VOL_2020_MIN_OMS         False
NIVEL                      False
dtype: bool
```

Checando correlações





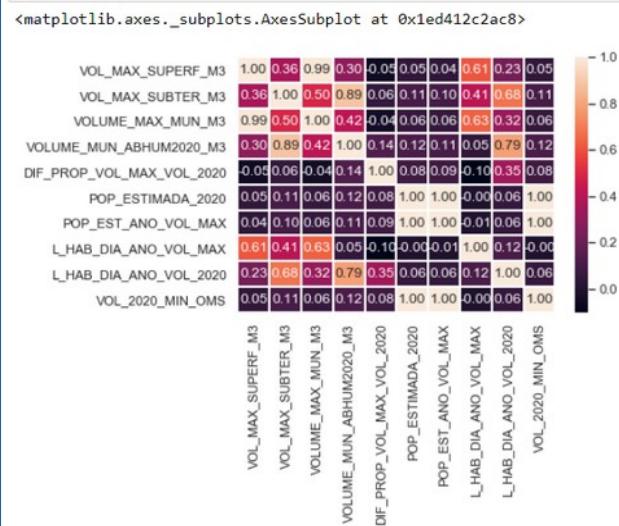
Obtendo da matriz de correlações

Faixas de deficit diferença percentual entre Volume máximo e Volume 2020			
0,70	-----	1,00	Forte
0,50	-----	0,70	Moderada
0,50	-----	0,25	Baixa
0,00		Inexistente	

```

# MATRIZ DE CORRELAÇÕES
correlation = AB_HUM_TREE_.corr() # análise de correlação
plot = sns.heatmap(correlation, annot = True, fmt=".2f", linewidths=1) # plot da matriz de correlação
plot

```



Separando as variáveis.

```
#SEPARANDO VARIÁVEIS
target = AB_HUM_TREE_['NIVEL']
AB_HUM_TREE_1 = AB_HUM_TREE_.copy()
AB_HUM_TREE_1 = AB_HUM_TREE_1.drop('NIVEL', axis =1)

X = AB_HUM_TREE_1 # Defining the attributes

# CODIFICANDO A VARIÁVEL
le = LabelEncoder()
target = le.fit_transform(target)
target

array([2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2,
       2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 1, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,
       2, 2, 1, 2, 2, 2, 2, 0, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 2, 2])

y = target
```

Dividindo o conjunto de dados em conjuntos de treinamento e teste sendo 20% de registros selecionados aleatoriamente para teste.

```
#DIVIDINDO O CONJUNTO DE DADOS EM CONJUNTOS DE TREINAMENTO E TESTE, SENDO 20% DE REGISTROS SELECIONADOS ALEATORIAMENTE PARA TESTE
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2, random_state = 42)
print("Entrada de divisão de treinamento: ", X_train.shape)
print("Testando entrada de divisão: ", X_test.shape)

Entrada de divisão de treinamento: (147, 10)
Testando entrada de divisão: (37, 10)
```

Definindo o algoritmo da árvore de decisão

```
#DEFININDO O ALGORITMO DA ÁRVORE DE DECISÃO
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Classificador de árvore de decisão criado')

Classificador de árvore de decisão criado
```

Treinando o modelo de árvore de decisão

```
# TREINANDO O MODELO DE ÁRVORE DE DECISÃO
clf = clf.fit(X_train,y_train)

array([0.          , 0.          , 0.          , 0.          , 0.          ,
       0.          , 0.          , 0.28867607, 0.64968871, 0.06163522])
```

Verificando as 'features' mais importantes para o modelo treinado

```
# VERIFICANDO AS FEATURES MAIS IMPORTANTES PARA O MODELO TREINADO, ONDE SERÁ RETORNADO UM ARRAY COM O VALOR DE CADA VARIÁVEL
clf.feature_importances_

#IMPORTÂNCIA DA COLUNA NA DEFINIÇÃO DO RESULTADO
for feature,importancia in zip(AB_HUM_TREE_.columns,clf.feature_importances_):
    print("{}:{}".format(feature, importancia))

VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.0
VOLUME_MAX_MUN_M3:0.0
VOLUME_MUN_ABHUM2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.0
POP_ESTIMADA_2020:0.0
POP_EST_ANO_VOL_MAX:0.0
L_HAB_DIA_ANO_VOL_MAX:0.28867606886474817
L_HAB_DIA_ANO_VOL_2020:0.6496887110094658
VOL_2020_MIN_OMS:0.061635220125786185
```

Obtendo o Relatório de Classificação

```
# MATRIZ DE CONFUSÃO
y_pred = dtree.predict(X_test)
print("Relatório de classificação - \n", classification_report(y_test,y_pred))

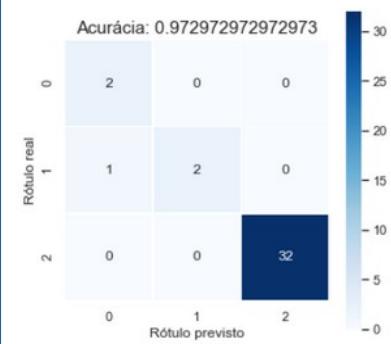
Relatório de classificação - \ n
precision      recall   f1-score   support
          0       0.67     1.00     0.80      2
          1       1.00     0.67     0.80      3
          2       1.00     1.00     1.00     32

accuracy                           0.97      37
macro avg       0.89     0.89     0.87      37
weighted avg    0.98     0.97     0.97      37
```

Gerando a Matriz de Confusão

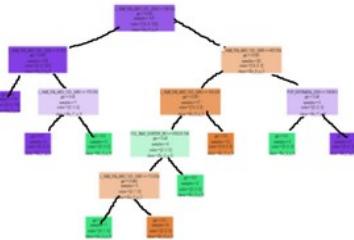
```
# MATRIZ DE CONFUSÃO
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True, square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

Text(0.5, 1, 'Acurácia: 0.972972972972973')



Visualizando a árvore de decisão

```
#VISUALIZANDO O DIAGRAMA DA ÁRVORE DE DECISÃO
dec_tree = plot_tree(decision_tree=dtree, feature_names = AB_HUM_TREE_1.columns,
                     class_names =["De_0_a_3", "De_4_a_6", "De_7_a_9"], filled = True , precision = 3, rounded = True)
```



```
#ÁRVORE DE DECISÃO EM FORMATO DE TEXTO
from sklearn.tree.export import export_text
r = export_text(dtree, feature_names=X.columns.values.tolist ())
print(r)
```

```
--- L_HAB_DIA_ANO_VOL_2020 <= 108.16
|--- L_HAB_DIA_ANO_VOL_2020 <= 81.60
|   |--- class: 2
|--- L_HAB_DIA_ANO_VOL_2020 > 81.60
|   |--- L_HAB_DIA_ANO_VOL_MAX <= 179.36
|       |--- class: 2
|       |--- L_HAB_DIA_ANO_VOL_MAX > 179.36
|           |--- class: 1
--- L_HAB_DIA_ANO_VOL_2020 > 108.16
|--- L_HAB_DIA_ANO_VOL_MAX <= 667.35
|   |--- L_HAB_DIA_ANO_VOL_MAX <= 198.20
|       |--- VOL_MAX_SUBTER_M3 <= 676009.19
|           |--- L_HAB_DIA_ANO_VOL_MAX <= 113.63
|               |--- class: 1
|               |--- L_HAB_DIA_ANO_VOL_MAX > 113.63
|                   |--- class: 0
|                   |--- VOL_MAX_SUBTER_M3 > 676009.19
|                       |--- class: 1
|--- L_HAB_DIA_ANO_VOL_MAX > 198.20
|   |--- class: 0
--- L_HAB_DIA_ANO_VOL_MAX > 667.35
|--- POP_ESTIMADA_2020 <= 19908.50
|   |--- class: 1
|--- POP_ESTIMADA_2020 > 19908.50
|   |--- class: 2
```

Fazendo a seguir uma poda com 04 e 03 nós para comparação

```
# REFAZENDO A ÁRVORE COM NOVOS PARÂMETROS ##

#DEFININDO O ALGORITMO DA ÁRVORE DE DECISÃO - MÁXIMO 04 NÓS
dtree1=DecisionTreeClassifier(max_depth=4)
dtree1.fit(X_train,y_train)
print('Classificador de árvore de decisão criado')

# INSTANCIANDO O CLASSIFICADOR:
clf1 = DecisionTreeClassifier()

# TREINANDO O MODELO DE ARVORE DE DECISÃO
clf1 = clf1.fit(X_train,y_train)

# VERIFICANDO AS FEATURES MAIS IMPORTANTES PARA O MODELO TREINADO, ONDE SERÁ RETORNADO UM ARRAY COM O VALOR DE CADA VARIÁVEL
clf1.feature_importances_

#IMPORTÂNCIA DA COLUNA NA DEFINIÇÃO DO RESULTADO
for feature,importancia in zip(AB_HUM_TREE_.columns,clf1.feature_importances_):
    print("{}:{}\n".format(feature, importancia))

# MATRIZ DE CONFUSÃO
y_pred = dtree1.predict(X_test)
print("Relatório de classificação - \n", classification_report(y_test,y_pred))

#VISUALIZANDO O DIAGRAMA DA ÁRVORE DE DECISÃO

dec_tree1 = plot_tree(decision_tree=dtree1, feature_names = AB_HUM_TREE_1.columns,
                      class_names =["De_0_a_3", "De_4_a_6", "De_7_a_9"] , filled = True , precision = 0, rounded = True)

#ÁRVORE DE DECISÃO EM FORMATO DE TEXTO
from sklearn.tree import export_text
r1 = export_text(dtree1, feature_names=X.columns.values.tolist ())
print(r1)

# MATRIZ DE CONFUSÃO
cm1 = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm1,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(dtree1.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

```

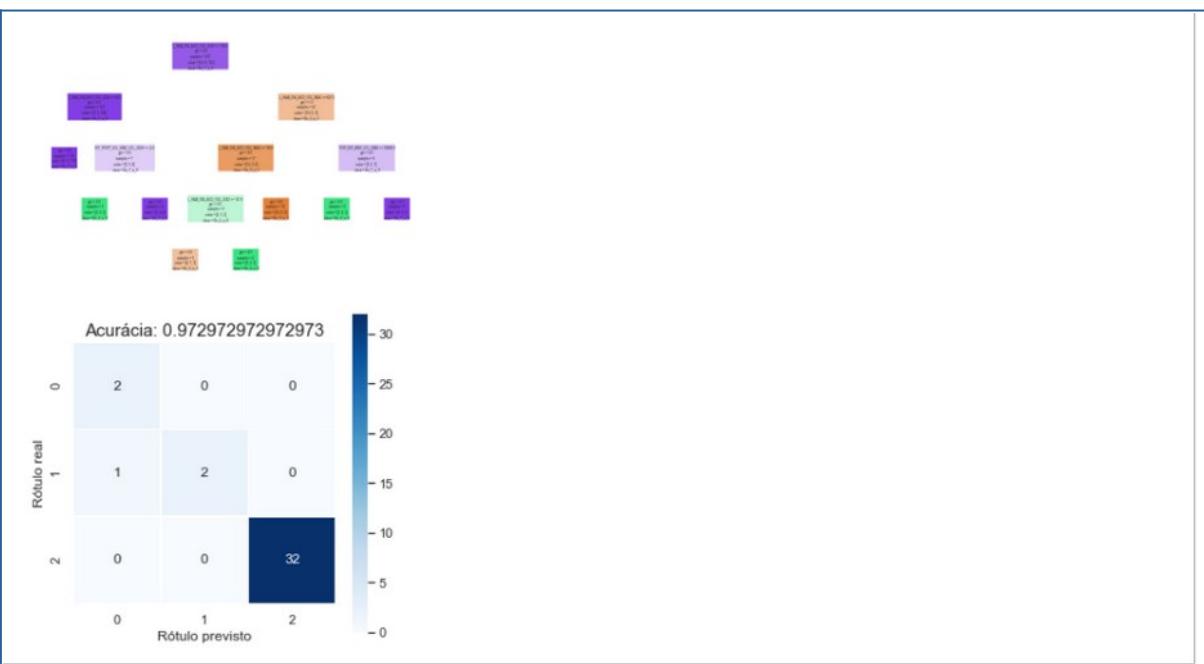
Classificador de árvore de decisão criado
VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.027393431167016077
VOLUME_MAX_MUN_M3:0.0
VOLUME_MUN_ABHUM2020_M3:0.06163522012578617
DIF_PROP_VOL_MAX_VOL_2020:0.08805031446540883
POP_ESTIMADA_2020:0.0
POP_EST_ANO_VOL_MAX:0.0
L_HAB_DIA_ANO_VOL_MAX:0.17323232323232318
L_HAB_DIA_ANO_VOL_2020:0.6496887110094657
VOL_2020_MIN_OHS:0.0
Relatório de classificação - \ n
                                         precision    recall   f1-score   support
                                         0       0.67     1.00     0.80      2
                                         1       1.00     0.67     0.80      3
                                         2       1.00     1.00     1.00     32

accuracy                           0.97      37
macro avg                           0.89      0.89     0.87      37
weighted avg                        0.98      0.97     0.97      37

|--- L_HAB_DIA_ANO_VOL_2020 <= 108.16
|   |--- L_HAB_DIA_ANO_VOL_2020 <= 81.60
|   |   |--- class: 2
|   |--- L_HAB_DIA_ANO_VOL_2020 >  81.60
|   |   |--- DIF_PROP_VOL_MAX_VOL_2020 <= -0.37
|   |   |   |--- class: 1
|   |   |   |--- DIF_PROP_VOL_MAX_VOL_2020 >  -0.37
|   |   |   |   |--- class: 2
|   |--- L_HAB_DIA_ANO_VOL_2020 >  108.16
|   |--- L_HAB_DIA_ANO_VOL_MAX <= 667.35
|   |   |--- L_HAB_DIA_ANO_VOL_MAX <= 198.20
|   |   |   |--- L_HAB_DIA_ANO_VOL_2020 <= 141.02
|   |   |   |   |--- class: 0
|   |   |   |   |--- L_HAB_DIA_ANO_VOL_2020 >  141.02
|   |   |   |   |   |--- class: 1
|   |   |   |--- L_HAB_DIA_ANO_VOL_MAX >  198.20
|   |   |   |   |--- class: 0
|   |--- L_HAB_DIA_ANO_VOL_MAX >  667.35
|   |   |--- POP_EST_ANO_VOL_MAX <= 18985.00
|   |   |   |--- class: 1
|   |   |   |--- POP_EST_ANO_VOL_MAX >  18985.00
|   |   |   |   |--- class: 2

```

Text(0.5, 1, 'Acurácia: 0.972972972972973')



```

## REFAZENDO A ÁRVORE COM NOVOS PARÂMETROS ##

#DEFININDO O ALGORITMO DA ÁRVORE DE DECISÃO - MÁXIMO 03 NÓS
dtree2=DecisionTreeClassifier(max_depth=3)
dtree2.fit(X_train,y_train)
print('Classificador de árvore de decisão criado')

# INSTANCIANDO O CLASSIFICADOR:
clf2 = DecisionTreeClassifier()

# TREINANDO O MODELO DE ARVORE DE DECISÃO
clf2 = clf2.fit(X_train,y_train)

# VERIFICANDO AS FEATURES MAIS IMPORTANTES PARA O MODELO TREINADO, ONDE SERÁ RETORNADO UM ARRAY COM O VALOR DE CADA VARIÁVEL
clf2.feature_importances_

#IMPORTÂNCIA DA COLUNA NA DEFINIÇÃO DO RESULTADO
for feature,importancia in zip(AB_HUM_TREE_.columns,clf2.feature_importances_):
    print("{}:{}".format(feature, importancia))

# MATRIZ DE CONFUSÃO
y_pred = dtree2.predict(X_test)
print("Relatório de classificação - \n", classification_report(y_test,y_pred))

#VISUALIZANDO O DIAGRAMA DA ÁRVORE DE DECISÃO

dec_tree = plot_tree(decision_tree=dtree2, feature_names = AB_HUM_TREE_1.columns,
                     class_names =["De_0_a_3", "De_4_a_6", "De_7_a_9"], filled = True , precision = 0, rounded = True)

#ÁRVORE DE DECISÃO EM FORMATO DE TEXTO
from sklearn.tree import export_text
r2 = export_text(dtree2, feature_names=X.columns.values.tolist ())
print(r2)

# MATRIZ DE CONFUSÃO
cm2 = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm2,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(dtree2.score(X_test, y_test))
plt.title(all_sample_title, size = 15)

```

```

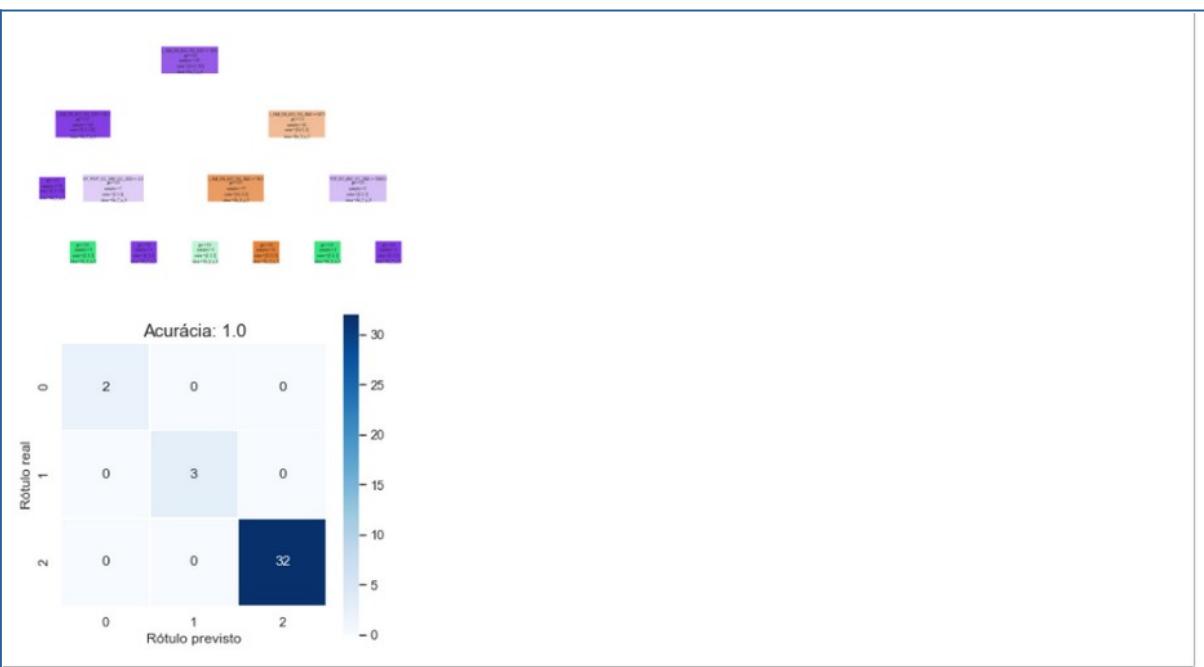
Classificador de árvore de decisão criado
VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.0
VOLUME_MAX_MUN_M3:0.0
VOLUME_MUN_ABHUM2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.08805031446540883
POP_ESTIMADA_2020:0.06163522012578617
POP_EST_ANO_VOL_MAX:0.0
L_HAB_DIA_ANO_VOL_MAX:0.2074741121910933
L_HAB_DIA_ANO_VOL_2020:0.6428403532177116
VOL_2020_MIN_OMS:0.0
Relatório de classificação - \
n                                               precision    recall   f1-score  support
                                                 0       1.00      1.00     1.00      2
                                                 1       1.00      1.00     1.00      3
                                                 2       1.00      1.00     1.00     32

accuracy                           1.00      37
macro avg                           1.00      1.00      1.00      37
weighted avg                        1.00      1.00     1.00      37

|--- L_HAB_DIA_ANO_VOL_2020 <= 108.16
|--- L_HAB_DIA_ANO_VOL_2020 <= 81.60
|   |--- class: 2
|--- L_HAB_DIA_ANO_VOL_2020 > 81.60
|   |--- DIF_PROP_VOL_MAX_VOL_2020 <= -0.37
|   |   |--- class: 1
|   |   |--- DIF_PROP_VOL_MAX_VOL_2020 > -0.37
|   |   |   |--- class: 2
|--- L_HAB_DIA_ANO_VOL_2020 > 108.16
|--- L_HAB_DIA_ANO_VOL_MAX <= 667.35
|   |--- L_HAB_DIA_ANO_VOL_MAX <= 198.20
|   |   |--- class: 1
|   |   |--- L_HAB_DIA_ANO_VOL_MAX > 198.20
|   |   |   |--- class: 0
|--- L_HAB_DIA_ANO_VOL_MAX > 667.35
|   |--- POP_EST_ANO_VOL_MAX <= 18985.00
|   |   |--- class: 1
|   |   |--- POP_EST_ANO_VOL_MAX > 18985.00
|   |   |   |--- class: 2

Text(0.5, 1, 'Acurácia: 1.0')

```



Refazendo com implantação de algumas alterações, pois foi percebido que o alvo “NIVEL”, estava como categórico , o que estava prejudicando a elaboração do modelo. Vamos ver se é mesmo isso.

```
#CRIANDO UM DATAFRAME PARA USO NA ÁRVORE COM OS DADOS LEVANTADOS NA FINALIDADE ABASTECIMENTO HUMANO
AB_HUM_MODEL_TREE = DIREITO_DE_USO_AB_HUM_MAX

#REDUZINDO OS NÍVEIS DE REGULARIZAÇÃO DA FINALIDADE ABASTECIMENTO HUMANO A 03 FAIXAS
lista_faixa_reg_ab_hum = []
faixa_ab_hum = AB_HUM_MODEL_TREE['REG_AB_HUM']
for abhum in faixa_ab_hum:
    if abhum <= 4:
        regabhum = 0
        lista_faixa_reg_ab_hum.append(regabhum)
    if abhum >= 6:
        regabhum = 2
        lista_faixa_reg_ab_hum.append(regabhum)
    if abhum > 4 and abhum < 6 :
        regabhum = 1
        lista_faixa_reg_ab_hum.append(regabhum)
AB_HUM_MODEL_TREE['NIVEL'] = lista_faixa_reg_ab_hum
```

```
AB_HUM_MODEL_TREE.head()
```

	Nome Município	MUNICIPIO	CD_IBGE	FINALIDADE_DE_USO	ANOS_VIGENTES	VOL_SUPERF_M3	VOL_SUBTER_M3	VOLUME_MAX_MUN_M3	VOLUME_MUN_J
25	Abaiara	ABAIARA	2300101	Abastecimento Humano	2017	0.000000e+00	1.313312e+05	1.313312e+05	
421	Acarape	ACARAPE	2300150	Abastecimento Humano	2007	1.209600e+05	0.000000e+00	1.209600e+05	
851	Acarau	ACARAU	2300200	Abastecimento Humano	2014	6.115200e+05	1.408826e+06	2.020346e+06	
1263	Acopiara	ACOPIARA	2300309	Abastecimento Humano	2012	1.313424e+06	5.376000e+03	1.318800e+06	
1685	Aiuaba	AIUABA	2300408	Abastecimento Humano	2015	7.924089e+05	4.380000e+02	7.928469e+05	

5 rows × 21 columns

```
#GERANDO NOVO ÍNDICE COM SEQUÊNCIA UNIFORME
AB_HUM_MODEL_TREE = AB_HUM_MODEL_TREE.reset_index()
```

```
AB_HUM_MODEL_TREE.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   index            184 non-null    int64  
 1   Nome Município  184 non-null    object  
 2   MUNICIPIO        184 non-null    object  
 3   CD_IBGE          184 non-null    int64  
 4   FINALIDADE_DE_USO 184 non-null    category
 5   ANOS_VIGENTES   184 non-null    int64  
 6   VOL_SUPERF_M3   184 non-null    float32 
 7   VOL_SUBTER_M3   184 non-null    float64 
 8   VOLUME_MAX_MUN_M3 184 non-null    float64 
 9   VOLUME_MUN_ABHUM2020_M3 184 non-null    float64 
 10  DIF_PROP_VOL_MAX_VOL_2020 184 non-null    float64 
 11  POP_ESTIMADA_2020 184 non-null    int64  
 12  POP_EST_ANO_VOL_MAX 184 non-null    int32  
 13  L_HAB_DIA_ANO_VOL_MAX 184 non-null    float64 
 14  L_HAB_DIA_ANO_VOL_2020 184 non-null    float64 
 15  VOL_2020_MIN_OMS 184 non-null    float64 
 16  P01_LMAX         184 non-null    int64  
 17  P03_2020          184 non-null    int64  
 18  P02_L2020_LMAX   184 non-null    int64  
 19  PT_AB_HUM         184 non-null    int64  
 20  REG_AB_HUM         184 non-null    int64  
 21  NIVEL             184 non-null    int64  
dtypes: category(1), float32(1), float64(7), int32(1), int64(10), object(2)
memory usage: 29.4+ KB
```

```
#SEPARANDO COLUNAS QUE NÃO HÁ INTERESSE

conj_colunas_AB_HUM_MODEL_TREE=set(AB_HUM_MODEL_TREE.columns.values.tolist())#Criando uma lista com os nomes das colunas
conj_colunas_AB_HUM_MODEL_TREE_mantidas = {'DIF_PROP_VOL_MAX_VOL_2020','L_HAB_DIA_ANO_VOL_2020','L_HAB_DIA_ANO_VOL_MAX','POP_EST'}
conj_colunas_remover = conj_colunas_AB_HUM_MODEL_TREE - conj_colunas_AB_HUM_MODEL_TREE_mantidas#Colunas a remover

#REMOVER COLUNAS SELECIONADAS
AB_HUM_MODEL_TREE = AB_HUM_MODEL_TREE.drop(columns=conj_colunas_remover)# Removendo as colunas

AB_HUM_MODEL_TREE.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   VOL_SUPERF_M3    184 non-null    float32 
 1   VOL_SUBTER_M3    184 non-null    float64 
 2   VOLUME_MAX_MUN_M3 184 non-null    float64 
 3   VOLUME_MUN_ABHUM2020_M3 184 non-null    float64 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null    float64 
 5   POP_ESTIMADA_2020 184 non-null    int64  
 6   POP_EST_ANO_VOL_MAX 184 non-null    int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null    float64 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null    float64 
 9   VOL_2020_MIN_OMS 184 non-null    float64 
 10  NIVEL            184 non-null    int64  
dtypes: float32(1), float64(7), int32(1), int64(2)
memory usage: 14.5 KB
```

```
#RENOMEAR COLUNAS
AB_HUM_MODEL_TREE = AB_HUM_MODEL_TREE.rename(columns={'VOL_SUPERF_M3':'VOL_MAX_SUPERF_M3','VOL_SUBTER_M3':'VOL_MAX_SUBTER_M3',})

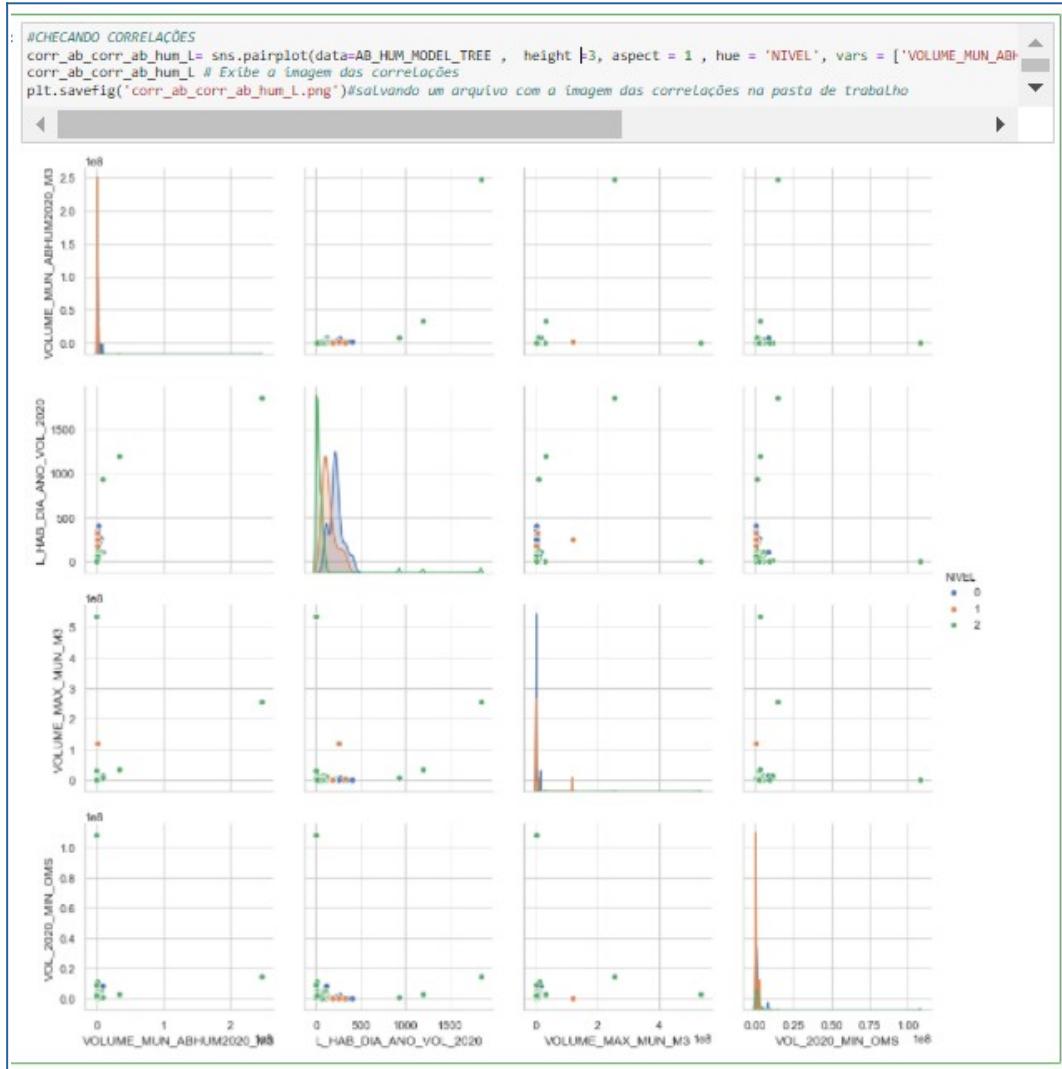
AB_HUM_MODEL_TREE.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   VOL_MAX_SUPERF_M3 184 non-null    float32 
 1   VOL_MAX_SUBTER_M3 184 non-null    float64 
 2   VOLUME_MAX_MUN_M3 184 non-null    float64 
 3   VOLUME_MUN_ABHUM2020_M3 184 non-null    float64 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null    float64 
 5   POP_ESTIMADA_2020 184 non-null    int64  
 6   POP_EST_ANO_VOL_MAX 184 non-null    int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null    float64 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null    float64 
 9   VOL_2020_MIN_OMS 184 non-null    float64 
 10  NIVEL            184 non-null    int64  
dtypes: float32(1), float64(7), int32(1), int64(2)
memory usage: 14.5 KB
```

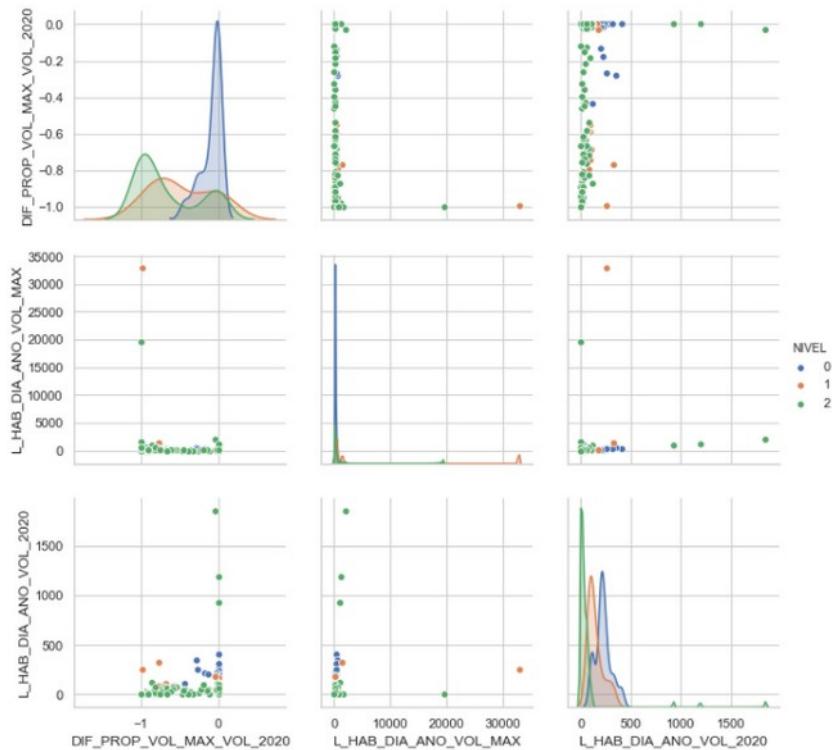
```
#ALTERANDO A FORMATAÇÃO DO TIPO DE COLUNA - REDUZIR A MEMÓRIA USADA
AB_HUM_MODEL_TREE.VOL_MAX_SUBTER_M3 = AB_HUM_MODEL_TREE.VOL_MAX_SUBTER_M3.astype('float32')
AB_HUM_MODEL_TREE.VOLUME_MAX_MUN_M3 = AB_HUM_MODEL_TREE.VOLUME_MAX_MUN_M3.astype('float32')
AB_HUM_MODEL_TREE.VOLUME_MUN_ABHUM2020_M3 = AB_HUM_MODEL_TREE.VOLUME_MUN_ABHUM2020_M3.astype('float32')
AB_HUM_MODEL_TREE.DIF_PROP_VOL_MAX_VOL_2020 = AB_HUM_MODEL_TREE.DIF_PROP_VOL_MAX_VOL_2020.astype('float32')
AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_MAX = AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_MAX.astype('float32')
AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_2020 = AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_2020.astype('float32')
AB_HUM_MODEL_TREE.POP_ESTIMADA_2020 = AB_HUM_MODEL_TREE.POP_ESTIMADA_2020.astype('int32')
AB_HUM_MODEL_TREE.VOL_2020_MIN_OMS = AB_HUM_MODEL_TREE.VOL_2020_MIN_OMS.astype('float32')
AB_HUM_MODEL_TREE.NIVEL = AB_HUM_MODEL_TREE.NIVEL.astype('int32')

AB_HUM_MODEL_TREE.info()

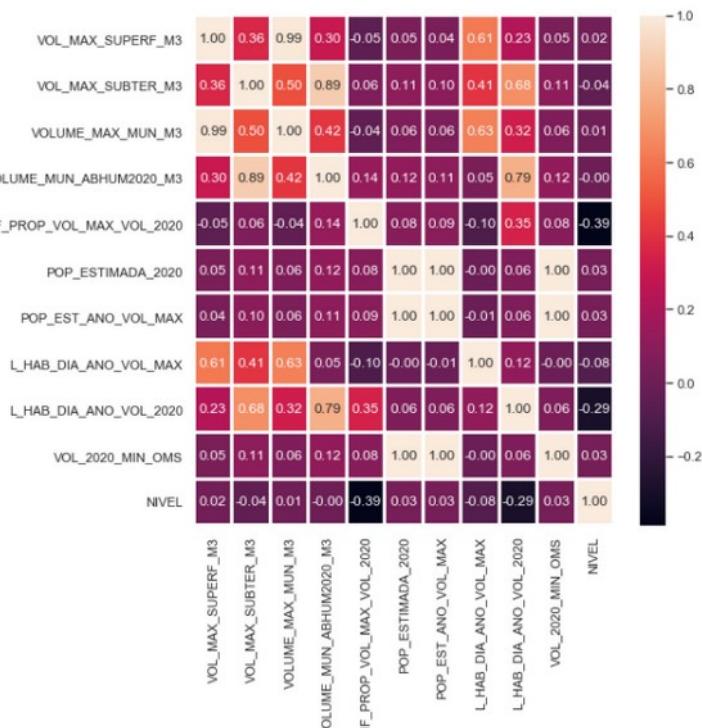
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   VOL_MAX_SUPERF_M3    184 non-null   float32 
 1   VOL_MAX_SUBTER_M3    184 non-null   float32 
 2   VOLUME_MAX_MUN_M3    184 non-null   float32 
 3   VOLUME_MUN_ABHUM2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020    184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX  184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_OMS    184 non-null   float32 
 10  NIVEL               184 non-null   int32  
dtypes: float32(8), int32(3)
memory usage: 8.0 KB
```



```
#CHECANDO CORRELAÇÕES
corr_ab_corr_ab_hum_L3= sns.pairplot(data=AB_HUM_MODEL_TREE , hue = 'NIVEL',height =3, aspect =1 , vars = ['DIF_PROP_VOL_MAX_VOL_2020','L_HAB_DIA_ANO_VOL_MAX'])
corr_ab_corr_ab_hum_L3 # Exibe a imagem das correlações
plt.savefig('corr_ab_corr_ab_hum_L3.png') # Salva um arquivo com a imagem das correlações na pasta de trabalho
```



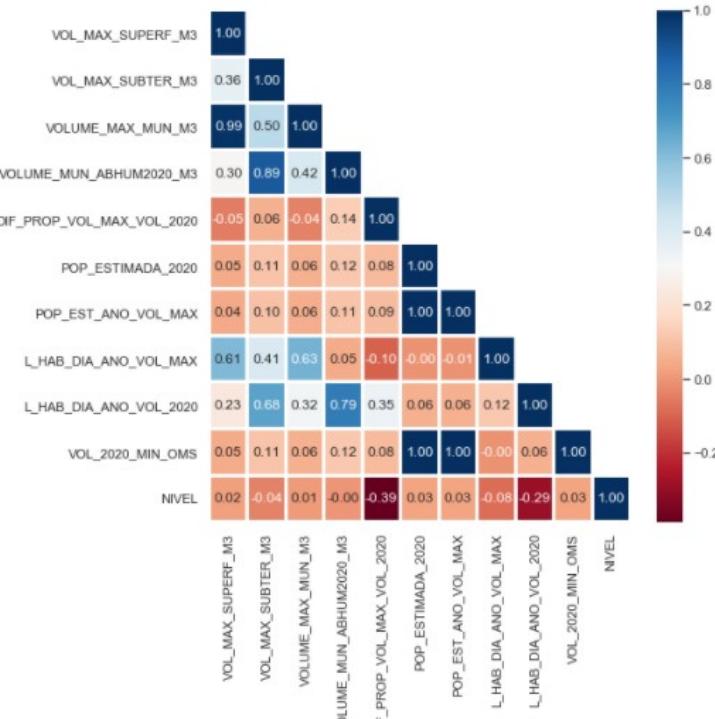
```
# MATRIZ DE CORRELAÇÕES
correlation = AB_HUM_MODEL_TREE .corr() # análise de correlação
plt.figure(figsize=(8, 8))
plot = sns.heatmap(correlation, annot = True, fmt=".2f", linewidths=2) # plot da matriz de correlação
plot
plt.savefig('matriz_correlation4.png')
```



```

def plot_corr(corr):
    # Cortaremos a metade de cima pois é o espelho da metade de baixo
    mask = np.zeros_like(corr, dtype=np.bool)
    mask[np.triu_indices_from(mask, 1)] = True
    plt.figure(figsize=(8, 8))
    sns.heatmap(corr, mask=mask, cmap='RdBu', annot = True, fmt=".2f", linewidths=2)
# Calculando a correlação
corr = AB_HUM_MODEL_TREE.corr()
plot_corr(corr)
plt.savefig('matriz_correlation6.png')

```



```

: #SEPARANDO VARIÁVEIS: ATRIBUTOS E ALVO
alvo = AB_HUM_MODEL_TREE['NIVEL'] # definindo o alvo
AB_HUM_TREE_COPIA = AB_HUM_MODEL_TREE.copy()
AB_HUM_TREE_COPIA = AB_HUM_TREE_COPIA.drop('NIVEL', axis =1)

```

```

: AB_HUM_MODEL_TREE.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
 --- 
 0   VOL_MAX_SUPERF_M3    184 non-null   float32 
 1   VOL_MAX_SUBTER_M3    184 non-null   float32 
 2   VOLUME_MAX_MUN_M3    184 non-null   float32 
 3   VOLUME_MUN_ABHUM2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020    184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX  184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_OMS     184 non-null   float32 
 10  NIVEL               184 non-null   int32  
dtypes: float32(8), int32(3)
memory usage: 8.0 KB

```

```
: AB_HUM_TREE_COPIA.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 10 columns):
 #   Column           Non-Null Count Dtype  
 ---  -- 
 0   VOL_MAX_SUPERF_M3    184 non-null   float32 
 1   VOL_MAX_SUBTER_M3    184 non-null   float32 
 2   VOLUME_MAX_MUN_M3    184 non-null   float32 
 3   VOLUME_MUN_ABHUM2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020     184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX   184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_OMS      184 non-null   float32 
dtypes: float32(8), int32(2)
memory usage: 7.3 KB

: alvo

: 0    2
1    2
2    2
3    2
4    2
..
179   2
180   2
181   2
182   2
183   2
Name: NIVEL, Length: 184, dtype: int32
```

```
X = AB_HUM_TREE_COPIA # Definindo os atributos

# CODIFICANDO A VARIÁVEL
LE = LabelEncoder()
alvo = LE.fit_transform(alvo)
alvo

array([2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2,
       2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 1, 2,
       2, 2, 1, 2, 2, 2, 2, 0, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 2, 2], dtype=int64)

#DEFININDO A VARIÁVEL 'Y' (alvo)
y = alvo

y

array([2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 1, 2,
       2, 2, 1, 2, 2, 2, 2, 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 0, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 2, 2], dtype=int64)

#DIVIDINDO O CONJUNTO DE DADOS EM CONJUNTOS DE TREINAMENTO E TESTE, SENDO 20% DE REGISTROS SELECIONADOS ALEATORIAMENTE COM SEMEJANTEZA

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
print("Entrada de divisão de treinamento: ", X_train.shape)
print("Testando entrada de divisão: ", X_test.shape)
< [REDACTED] >

Entrada de divisão de treinamento: (147, 10)
Testando entrada de divisão: (37, 10)
```

```

### ALGORITMO ÁRVORE DE DECISÃO

# INSTANCIANDO O CLASSIFICADOR:
dtree=DecisionTreeClassifier()

# TREINANDO O MODELO DE ÁRVORE DE DECISÃO:
dtree = dtree.fit(X_train,y_train)
print('Classificador de árvore de decisão criado')

Classificador de árvore de decisão criado

# VERIFICANDO OS RECURSOS MAIS IMPORTANTES PARA O MODELO TREINADO, ONDE SERÁ RETORNADO UM ARRAY COM O VALOR DE CADA VARIÁVEL
dtree.feature_importances_

array([0.          , 0.02739343, 0.          , 0.          , 0.          ,
       0.          , 0.06163522, 0.26128264, 0.64968871, 0.          ])

```

```

#IMPORTÂNCIA DA COLUNA NA DEFINIÇÃO DO RESULTADO DA CLASSIFICAÇÃO
print ('Importância do atributo na definição do resultado da classificação')
print('')
for feature,importancia in zip(AB_HUM_TREE_COPIA.columns,dtree.feature_importances_):
    print("{}:{}.".format(feature, importancia))

Importância do atributo na definição do resultado da classificação

VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.027393431167016077
VOLUME_MAX_MUN_M3:0.0
VOLUME_MUN_ABHUM2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.0
POP_ESTIMADA_2020:0.0
POP_EST_ANO_VOL_MAX:0.06163522012578617
L_HAB_DIA_ANO_VOL_MAX:0.261282637697732
L_HAB_DIA_ANO_VOL_2020:0.6496887110094657
VOL_2020_MIN_OMS:0.0

```

```

# RESULTADO DA CLASSIFICAÇÃO

y_pred = dtree.predict(X_test)
resultado = y_pred
print('')
print('Resultado da predição')
print(y_pred)
print('')
print("      Relatório de classificação ")
print('')
print(metrics.classification_report(y_test,resultado))

Resultado da predição
[2 2 2 2 2 2 2 0 2 0 2 2 2 2 2 2 0 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2]

Relatório de classificação

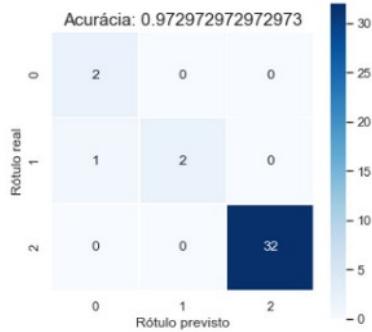
      precision    recall   f1-score   support

          0       0.67      1.00      0.80       2
          1       1.00      0.67      0.80       3
          2       1.00      1.00      1.00      32

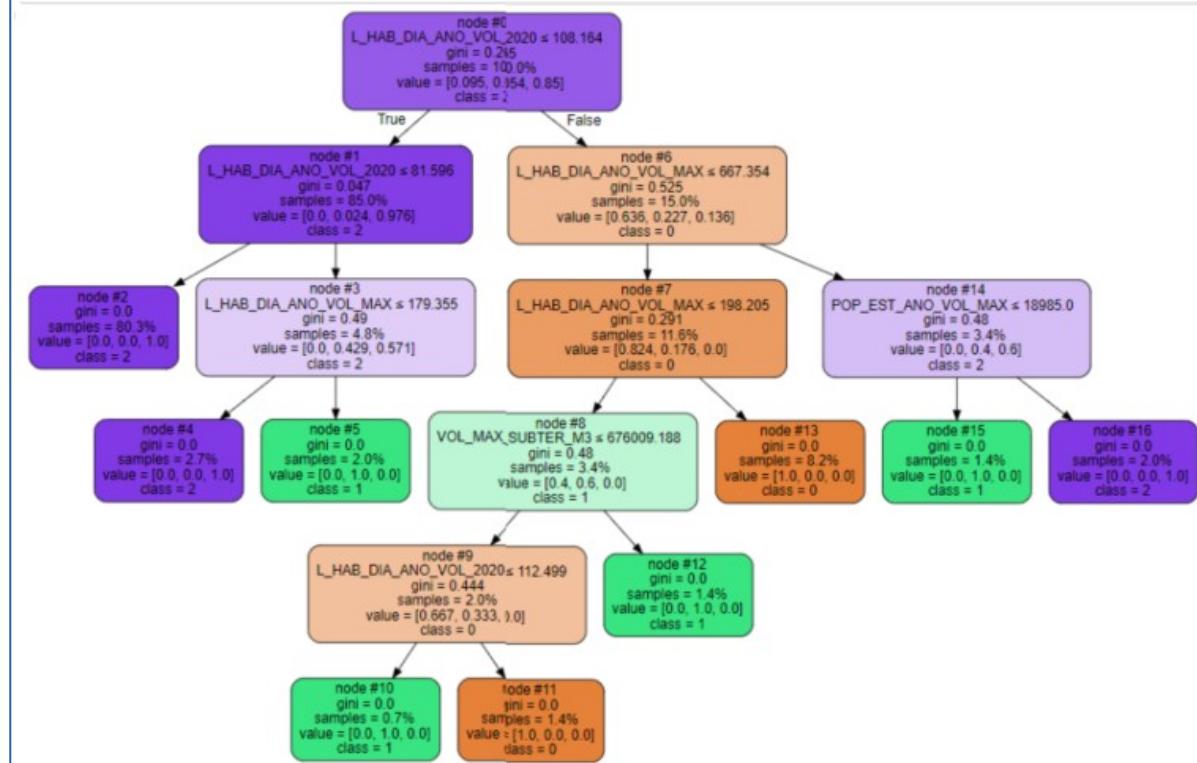
   accuracy                           0.97      37
  macro avg       0.89      0.89      0.87      37
weighted avg       0.98      0.97      0.97      37

```

```
: # MATRIZ DE CONFUSÃO
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
mconf = sns.heatmap(data=cm, linewidths=.5, annot=True, square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
mconf.figure.savefig('matriz_confusao.png')
```



```
: # RENDERIZANDO A ÁRVORE DE FORMA GRÁFICA - sem parâmetros
dot_data = export_graphviz(
    dtree,
    out_file=None,
    feature_names=AB_HUM_TREE_COPIA.columns,
    class_names=['0','1','2'],
    filled=True, rounded=True,
    proportion=True,
    node_ids=True,
    rotate=False,
    label='all',
    special_characters=True
)
graph = graphviz.Source(dot_data)
graph
```



```
#RENDERIZANDO A ÁRVORE DE FORMA INTERATIVA:

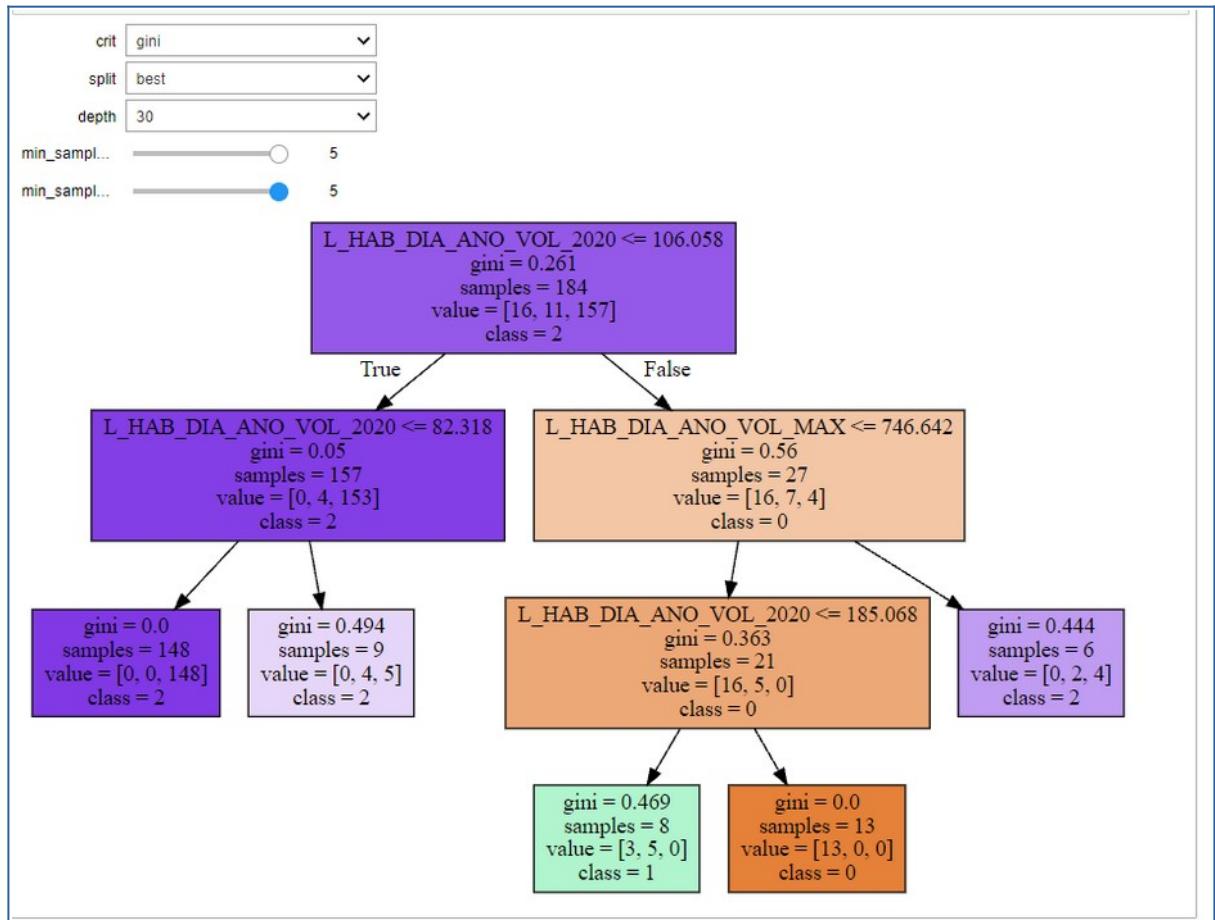
# feature matrix
X,y = AB_HUM_TREE_COPIA, AB_HUM_MODEL_TREE['NIVEL']

# feature labels
features_label = AB_HUM_TREE_COPIA.columns

# class label
class_label = ['0','1','2']
| def plot_tree(crit, split, depth, min_samples_split, min_samples_leaf=0.2):
    estimator = DecisionTreeClassifier(
        random_state = 0
        ,criterion = crit
        ,splitter = split
        ,max_depth = depth
        ,min_samples_split=min_samples_split
        ,min_samples_leaf=min_samples_leaf
    )
    estimator.fit(X, y)
    graph = Source(export_graphviz(estimator
        , out_file=None
        , feature_names=features_label
        , class_names=class_label
        , impurity=True
        , filled = True))
    display(SVG(graph.pipe(format='svg')))
    return estimator

inter=interactive(plot_tree
    , crit = ["gini", "entropy"]
    , split = ["best", "random"]
    , depth=[1,2,3,4,5,10,20,30]
    , min_samples_split=(1,5)
    , min_samples_leaf=(1,5))

display(inter)
```



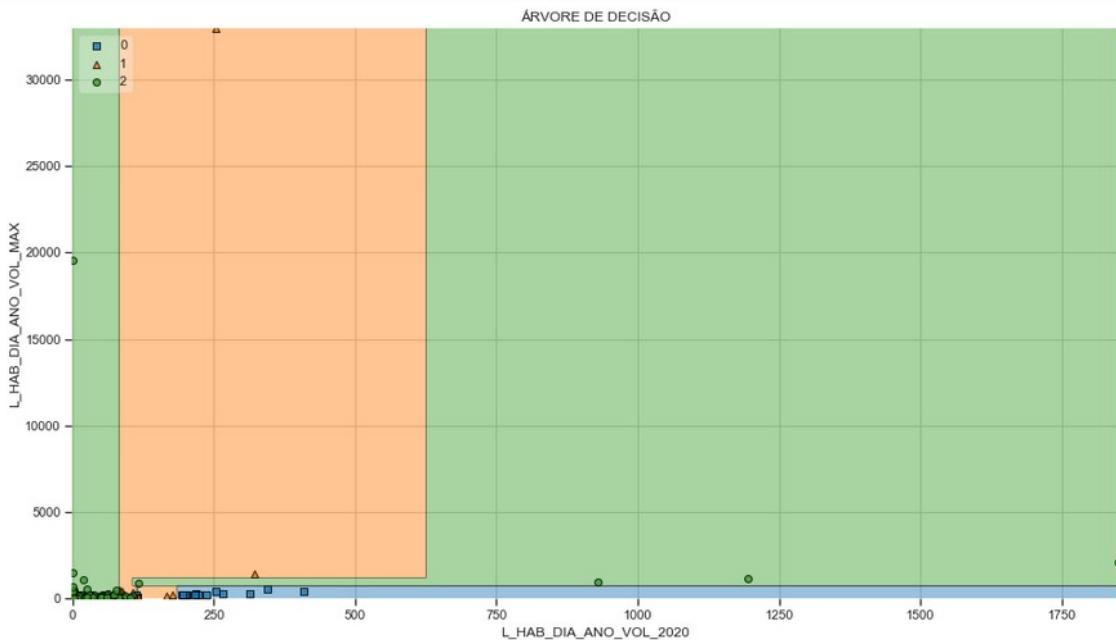
```
# VISUALIZANDO AS FRONTEIRAS CRIADAS PELA ÁRVORE CONSIDERANDO 02 ATRIBUTOS:

def visualize_fronteiras(msamples_split,max_depth):
    X = AB_HUM_MODEL_TREE[['L_HAB_DIA_ANO_VOL_2020','L_HAB_DIA_ANO_VOL_MAX']].values
    y = AB_HUM_MODEL_TREE.NIVEL.values
    clf = DecisionTreeClassifier(min_samples_split=msamples_split,max_depth=max_depth)
    tree = clf.fit(X, y)

    plt.figure(figsize=(16,9))
    plot_decision_regions(X, y, clf=tree, legend=2)

    plt.xlabel('L_HAB_DIA_ANO_VOL_2020')
    plt.ylabel('L_HAB_DIA_ANO_VOL_MAX')
    plt.title('ÁRVORE DE DECISÃO')
    plt.show()
```

```
# Chamando a função criada anteriormente:
visualize_fronteiras(msamples_split=2,max_depth=30)
```



```
#VISUALIZANDO O CAMINHO QUE A ÁRVORE TOMOU PARA CHEGAR EM UMA DETERMINADA DECISÃO
```

```
# NÚMERO DE NÓS DA ÁRVORE DE DECISÃO:
estimator = DecisionTreeClassifier(max_depth=3,min_samples_split=2,min_samples_leaf=2)
estimator.fit(X, y)
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=2, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort=False,
                      random_state=None, splitter='best')
n_nodes = estimator.tree_.node_count
children_left = estimator.tree_.children_left
children_right = estimator.tree_.children_right
feature = estimator.tree_.feature
threshold = estimator.tree_.threshold

print("Número de nós da árvore: {}".format(n_nodes))
```

Número de nós da árvore: 13

```
#DESCRICAÇÃO DA ESTRUTURA DA ÁRVORE

node_depth = np.zeros(shape=n_nodes, dtype=np.int64)
is_leaves = np.zeros(shape=n_nodes, dtype=bool)
stack = [(0, -1)]
while len(stack) > 0:
    node_id, parent_depth = stack.pop()
    node_depth[node_id] = parent_depth + 1

    if (children_left[node_id] != children_right[node_id]):
        stack.append((children_left[node_id], parent_depth + 1))
        stack.append((children_right[node_id], parent_depth + 1))
    else:
        is_leaves[node_id] = True

print("\nA arvore binária tem %s nós e a seguinte estrutura: \n"
      "%s" % n_nodes)
for i in range(n_nodes):
    if is_leaves[i]:
        print("%snó=%s (nó folha)." % (node_depth[i] * "\t", i))
    else:
        print("%snó=%s (nó teste): vai para o nó %s se o valor do atributo %s <= %s \n se não, vai para o "
              "nó %s."
              "%s (node_depth[i] * "\t",
              i,
              children_left[i],
              AB_HUM_MODEL_TREE.columns[feature[i]],
              threshold[i],
              children_right[i],
              ))
```

A arvore binária tem 13 nós e a seguinte estrutura:

```
nó=0 (nó teste): vai para o nó 1 se o valor do atributo L_HAB_DIA_ANO_VOL_2020 <= 106.05838394165039
se não, vai para o nó 6.
    nó=1 (nó teste): vai para o nó 2 se o valor do atributo L_HAB_DIA_ANO_VOL_2020 <= 82.31771087646484
se não, vai para o nó 3.
    nó=2 (nó folha).
    nó=3 (nó teste): vai para o nó 4 se o valor do atributo L_HAB_DIA_ANO_VOL_MAX <= 179.35504150390625
se não, vai para o nó 5.
    nó=4 (nó folha).
    nó=5 (nó folha).
    nó=6 (nó teste): vai para o nó 7 se o valor do atributo L_HAB_DIA_ANO_VOL_MAX <= 746.6421508789062
se não, vai para o nó 10.
    nó=7 (nó teste): vai para o nó 8 se o valor do atributo L_HAB_DIA_ANO_VOL_2020 <= 185.06758880615234
se não, vai para o nó 9.
    nó=8 (nó folha).
    nó=9 (nó folha).
    nó=10 (nó teste): vai para o nó 11 se o valor do atributo POP_EST_ANO_VOL_MAX <= 17408.5
se não, vai para o nó 12.
    nó=11 (nó folha).
    nó=12 (nó folha).
```

```
#EXTRAINDO REGRAS DA ARVORE GERADA A PARTIR DE UMA DETERMINADA AMOSTRA DOS DADOS

def extrai_regras2(sample_id):
    node_indicator = estimator.decision_path(X)

    leave_id = estimator.apply(X)

    #sample_id = sample
    node_index = node_indicator.indices[node_indicator.indptr[sample_id]:node_indicator.indptr[sample_id + 1]]

    print('\nFeatures usadas para predizer a amostra %s' % (sample_id))

    for f,v in zip(AB_HUM_MODEL_TREE.columns,X.iloc[sample_id].values):
        print('%s = %s'%(f,v))

    print('\n')
    for node_id in node_index:
        if leave_id[sample_id] == node_id:
            continue

        if (X.iloc[sample_id, feature[node_id]] <= threshold[node_id]):
            threshold_sign = "<="
        else:
            threshold_sign = ">"

        print("id do nó de decisão %s : (atributo %s com valor = %s %s %s)"%
              (node_id,
               AB_HUM_TREE_COPIA.columns[feature[node_id]],
               X.iloc[sample_id, feature[node_id]],
               threshold_sign,
               threshold[node_id]))

    pred =estimator.predict(X.iloc[sample_id].values.reshape(1, -1))
    print(pred)
    print("\tClasse => %s" %pred)
```

```
# CHAMANDO A FUNÇÃO CRIADA ANTERIORMENTE:
extrai_regras(2)
```

```
Features usadas para predizer a amostra 2
VOL_MAX_SUPERF_M3 = 611520.0
VOL_MAX_SUBTER_M3 = 1408826.0
VOLUME_MAX_MUN_M3 = 2028346.0
VOLUME_MUN_ABHUM2020_M3 = 777858.8125
DIF_PROP_VOL_MAX_VOL_2020 = -0.614987313747406
POP_ESTIMADA_2020 = 63184.0
POP_EST_ANO_VOL_MAX = 55900.0
L_HAB_DIA_ANO_VOL_MAX = 99.01923370361328
L_HAB_DIA_ANO_VOL_2020 = 33.77155303955078
VOL_2020_MIN_OMS = 2533625.5
```

```
id do nó de decisão 0 : (atributo L_HAB_DIA_ANO_VOL_2020 com valor = 33.771553 <= 106.05838394165039)
id do nó de decisão 1 : (atributo L_HAB_DIA_ANO_VOL_2020 com valor = 33.771553 <= 82.31771087646484)
[2]
    classe => [2]
```

```
AB_HUM_MODEL_TREE.head(5)
```

OP_VOL_MAX_VOL_2020	POP_ESTIMADA_2020	POP_EST_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	NIVEL
0.000000	11853	11155	32.252884	30.356155	4.758979e+05	2
-1.000000	15036	11563	28.660093	0.000000	6.036954e+05	2
-0.614987	63104	55900	99.019234	33.771553	2.533626e+06	2
-0.010559	54481	46350	77.952919	65.619209	2.187412e+06	2
-1.000000	17493	15812	137.373138	0.000000	7.023439e+05	2

Instalados e testados alguns algoritmos de treinamento e visualização, a fase de desenvolvimento do modelo agora é iniciada, agora observando as variáveis em conjunto com os roteiros abordados.

```
#CRIANDO UM DATAFRAME PARA USO NA ÁRVORE COM OS DADOS LEVANTADOS NA FINALIDADE ABASTECIMENTO HUMANO
AB_HUM_MODEL_TREE = DIREITO_DE_USO_AB_HUM_MAX

#REDUZINDO OS NÍVEIS DE REGULARAÇÃO DA FINALIDADE ABASTECIMENTO HUMANO A 03 FAIXAS
lista_faixa_reg_ab_hum = []
faixa_ab_hum = AB_HUM_MODEL_TREE['REG_AB_HUM']
for abhum in faixa_ab_hum:
    if abhum <= 4:
        regabhum = 0
        lista_faixa_reg_ab_hum.append(regabhum)
    if abhum >= 6:
        regabhum = 2
        lista_faixa_reg_ab_hum.append(regabhum)
    if abhum > 4 and abhum < 6 :
        regabhum = 1
        lista_faixa_reg_ab_hum.append(regabhum)
AB_HUM_MODEL_TREE['NIVEL'] = lista_faixa_reg_ab_hum

#GERANDO NOVO ÍNDICE COM SEQUÊNCIA UNIFORME
AB_HUM_MODEL_TREE = AB_HUM_MODEL_TREE.reset_index()

#SEPARANDO COLUNAS QUE NÃO HÁ INTERESSE
conj_colunas_AB_HUM_MODEL_TREE=set(AB_HUM_MODEL_TREE.columns.values.tolist())#Criando uma lista com os nomes das colunas
conj_colunas_AB_HUM_MODEL_TREE_mantidas = {'DIF_PROP_VOL_MAX_VOL_2020','L_HAB_DIA_ANO_VOL_2020','L_HAB_DIA_ANO_VOL_MAX','POP_ESTIMADA_2020','VOL_2020_MIN_0MS'}
conj_colunas_remover = conj_colunas_AB_HUM_MODEL_TREE - conj_colunas_AB_HUM_MODEL_TREE_mantidas#Colunas a remover

#REMOVER COLUNAS SELECIONADAS
AB_HUM_MODEL_TREE = AB_HUM_MODEL_TREE.drop(columns=conj_colunas_remover)# Removendo as colunas

#RENOMEAR COLUNAS
AB_HUM_MODEL_TREE = AB_HUM_MODEL_TREE.rename(columns={'VOL_SUPERF_M3':'VOL_MAX_SUPERF_M3','VOL_SUBTER_M3':'VOL_MAX_SUBTER_M3','VOL_2020_MIN_0MS':'VOL_2020_MIN_0MS','NIVEL':'NIVEL'})#Renomeando as colunas

#ALTERANDO A FORMATAÇÃO DO TIPO DE COLUNA - REDUZIR A MEMÓRIA USADA
AB_HUM_MODEL_TREE.VOL_MAX_SUBTER_M3 = AB_HUM_MODEL_TREE.VOL_MAX_SUBTER_M3.astype('float32')
AB_HUM_MODEL_TREE.VOLUME_MAX_MUN_M3 = AB_HUM_MODEL_TREE.VOLUME_MAX_MUN_M3.astype('float32')
AB_HUM_MODEL_TREE.VOLUME_MUN_ABHUM_2020_M3 = AB_HUM_MODEL_TREE.VOLUME_MUN_ABHUM_2020_M3.astype('float32')
AB_HUM_MODEL_TREE.DIF_PROP_VOL_MAX_VOL_2020 = AB_HUM_MODEL_TREE.DIF_PROP_VOL_MAX_VOL_2020.astype('float32')
AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_MAX = AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_MAX.astype('float32')
AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_2020 = AB_HUM_MODEL_TREE.L_HAB_DIA_ANO_VOL_2020.astype('float32')
AB_HUM_MODEL_TREE.POP_ESTIMADA_2020 = AB_HUM_MODEL_TREE.POP_ESTIMADA_2020.astype('int32')
AB_HUM_MODEL_TREE.VOL_2020_MIN_0MS = AB_HUM_MODEL_TREE.VOL_2020_MIN_0MS.astype('float32')
AB_HUM_MODEL_TREE.NIVEL = AB_HUM_MODEL_TREE.NIVEL.astype('int32')

#OBSERVANDO AS CONFIGURAÇÕES DO DATAFRAME
AB_HUM_MODEL_TREE.info()

#VERIFICANDO A PRESENÇA DE NAN
AB_HUM_MODEL_TREE.isnull().any()
```

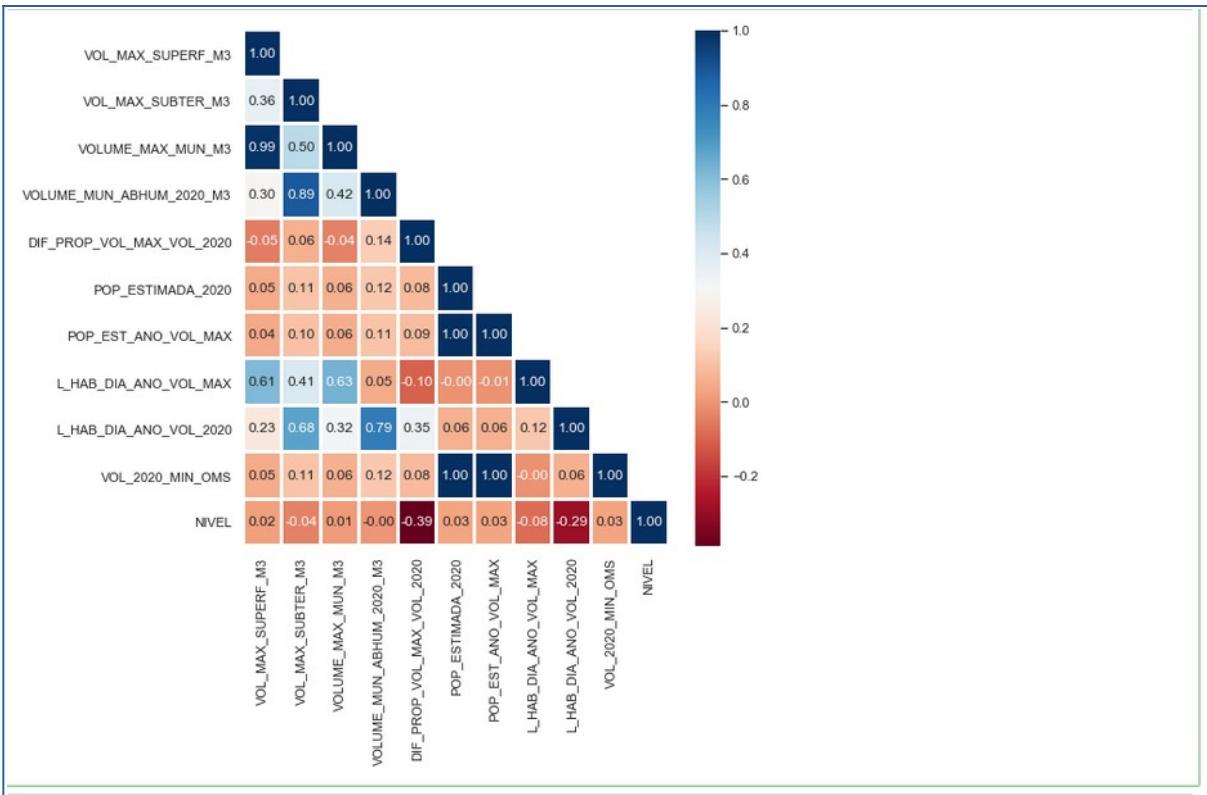
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   VOL_MAX_SUPERF_M3      184 non-null   float32 
 1   VOL_MAX_SUBTER_M3      184 non-null   float32 
 2   VOLUME_MAX_MUN_M3      184 non-null   float32 
 3   VOLUME_MUN_ABHUM_2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020      184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX    184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX  184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_0MS       184 non-null   float32 
 10  NIVEL                184 non-null   int32  
dtypes: float32(8), int32(3)
memory usage: 8.0 KB

VOL_MAX_SUPERF_M3      False
VOL_MAX_SUBTER_M3      False
VOLUME_MAX_MUN_M3      False
VOLUME_MUN_ABHUM_2020_M3 False
DIF_PROP_VOL_MAX_VOL_2020 False
POP_ESTIMADA_2020      False
POP_EST_ANO_VOL_MAX    False
L_HAB_DIA_ANO_VOL_MAX  False
L_HAB_DIA_ANO_VOL_2020  False
VOL_2020_MIN_0MS       False
NIVEL                  False
dtype: bool
```

```

: # MATRIZ DE CORRELAÇÕES
def plot_corr(corr):
    # Cortaremos a metade de cima pois é o espelho da metade de baixo
    mask = np.zeros_like(corr, dtype=np.bool)
    mask[np.triu_indices_from(mask,1)] = True
    plt.figure(figsize=(8, 8))
    sns.heatmap(corr, mask=mask, cmap='RdBu', annot = True, fmt=".2f", linewidths=2)
# Calculando a correlação
corr = AB_HUM_MODEL_TREE.corr()
plot_corr(corr)
plt.savefig('matriz_correlation.png')
print('MATRIZ DE CORRELAÇÕES')
print('Totalmente azul (+1) => correlação máxima')
print('Totalmente vermelho (-1) => correlação inversa')
print('Totalmente Branco ( 0) => sem qualquer correlação')

```



Quanto ao quadro de correlações observa-se:

- O ‘NIVEL’ de regularização apresentou maior correlação com o atributo ‘DIF_PROP_VOL_MAX_VOL_2020’, seguido do atributo ‘L_HAB_DIA_ANO_VOL_2020’, sendo essas inversas.

Lembrando que o ‘DIF_PROP_VOL_MAX_VOL_2020’ varia de (-1) a (0), sendo zero a situação mais confortável, ou seja, quanto menor for esse valor, maior é a necessidade de regularização., já que o ‘NIVEL’ varia de (0) a (2),, sendo o nível 2 o mais crítico.

Já o ‘L_HAB_DIA_ANO_VOL_2020’, quanto maior o valor, menor é a necessidade de regularização

- O ‘VOLUME_MAX_MUN_M3’ apresentou quase 100% de correlação com o ‘VOL_MAX_SUPERF_M3’ e 50% com o ‘VOL_MAX_SUBTER_M3’.

Os valores do ‘VOL_MAX_SUPERF_M3’ e do ‘VOL_MAX_SUBTER_M3’ no dataset são vinculados aos registrados nos anos de ocorrência do ‘VOLUME_MAX_MUN_M3’, não são as ocorrências máximas do volume superficial ou subterrâneo na janela temporal, ou seja, não representam o máximo volume superficial do ocorrido no município nem o máximo volume subterrâneo. A soma de seus valores definem o ‘VOLUME_MAX_MUN_M3’.

A participação das águas superficiais no abastecimento humano é muito mais elevada do que a das águas subterrâneas, a correlação apresentada confirma esse fato.

- O ‘VOLUME_MUN_ABHUM_2020_M3’ apresentou 89% de correlação com o ‘VOL_MAX_SUBTER_M3’.

Essa correlação pode indicar que quando a ocorrência do volume máximo se deu em 2020, a influência das águas subterrâneas foi bastante acentuada.

- O atributo ‘L_HAB_DIA_ANO_VOL_2020’ apresentou 79% de correlação com o atributo ‘VOLUME_MUN_ABHUM_2020_M3’.

Lembrando que ‘L_HAB_DIA_ANO_VOL_2020’ diz respeito ao consumo diário de água por habitante em 2020, poderia parecer estranho não constar 100%, entretanto, supondo um mesmo valor de volume em 02 municípios podemos ter valores de consumo diferentes, a população de um município pode ter maior ou menor conforto hídrico que a de outro.

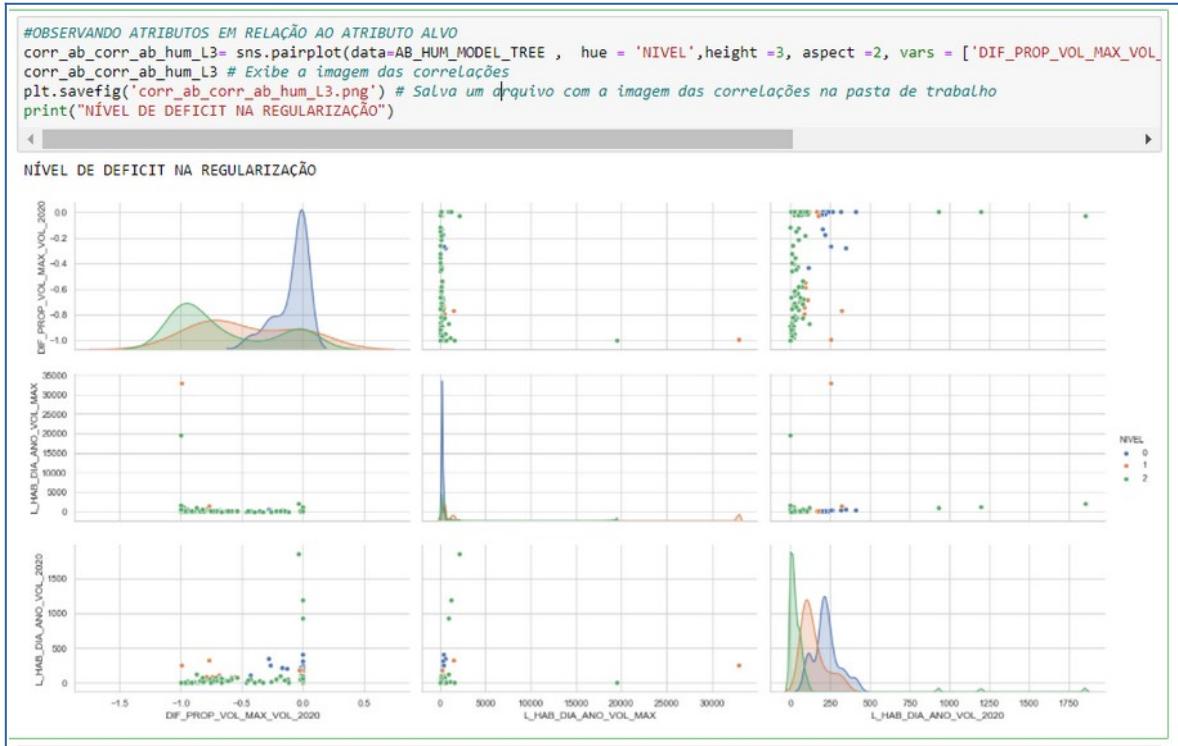
- O atributo ‘VOL_2020_MIN_OMS’ apresentou correção de 100% tanto com o atributo ‘POP_ESTIMADA_2020’ quanto com ‘POP_EST_ANO_VOL_MAX’.

Essas correlações, na verdade são relações, sendo a primeira direta e a segunda indireta.

O ‘VOL_2020_MIN_OMS’ foi calculado adotando o consumo mínimo dentro do padrão da OMS de consumo humano diário de água (valor invariável) e multiplicando pela população estimada para 2020 de cada município.

A ‘POP_EST_ANO_VOL_MAX’ foi determinada também com base na ‘POP_ESTIMADA_2020’ de cada município retrocedendo o número de anos até o ano volume máximo considerando a taxa de crescimento, nesse caso decrescimento, de 2% a.a.

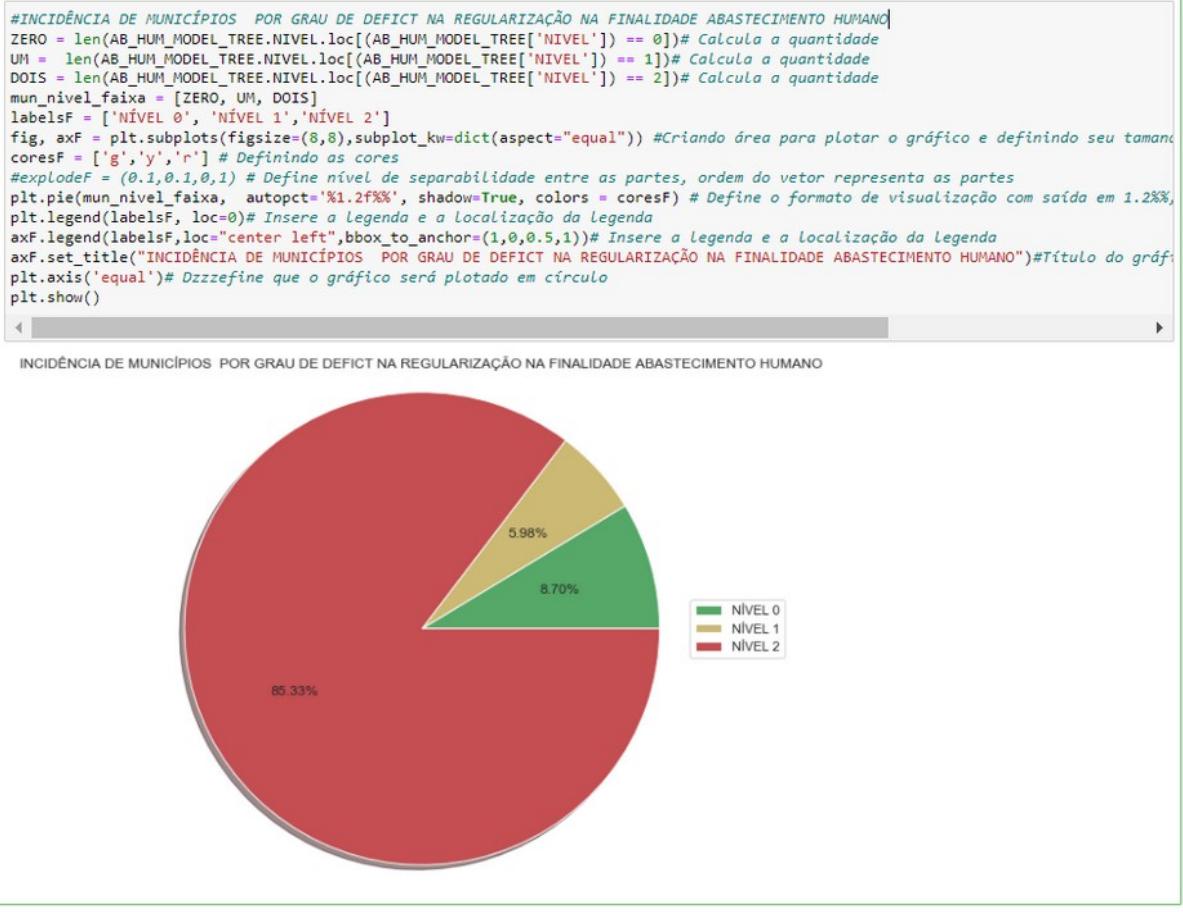
Observando as correlações do 'NIVEL' com os atributos utilizados para pontuação e definição dos 10 níveis de deficit de regularização, agrupadas em 03 classes de nível.



Considerações:

- As curvas de distribuição em relação ao 'DIF_PROP_VOL_MAX_VOL_2020', mostram que registros das classes de nível 1 e 2 encontram-se presentes na região cuja diferença, entre os volumes de 2020 e máximo, supera os 70 % (-1,0 a -0,7).
- Em relação a 'L_HAB_DIA_ANO_VOL_MAX', destaca-se que a classe de nível 0 chega a ultrapassar o patamar de 35.000 L diários por habitante, mas é algo bastante pontual, onde é possível que num município vizinho o consumo esteja bem abaixo, lembrando que o cálculo de estimativa desse consumo foi baseado no volume e na população do município de captação das águas.
- Quanto a 'L_HAB_DIA_ANO_VOL_2020' , no nível 2, ou seja, o mais crítico há curva supera os 1.500 L diários por habitante, isso por que essa situação exageradamente confortável ainda em 2020, considerando a conscientização quanto ao desperdício e as melhorias implementadas ao longo dos anos recebe avaliação, vamos dizer, menos positiva.

Observando os registros já rotulados.



A maior parte desses dados está numa mesma classe, então a divisão da base para o desenvolvimento do modelo não deve deixar de considerar essa distribuição não homogênea, ou seja, os dados estão desbalanceados.

Separando as variáveis: recursos e alvo e criando um dataframe sem a variável rotulada para desenvolvimento do modelo.

```
#SEPARANDO VARIÁVEIS: ATRIBUTOS E ALVO
alvo = AB_HUM_MODEL_TREE['NIVEL'] # definindo o alvo
AB_HUM_TREE_COPIA = AB_HUM_MODEL_TREE.copy()
AB_HUM_TREE_COPIA = AB_HUM_TREE_COPIA.drop('NIVEL', axis =1)

print('AB_HUM_MODEL_TREE')
AB_HUM_MODEL_TREE.info()

AB_HUM_MODEL_TREE
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
 ---  -- 
 0   VOL_MAX_SUPERF_M3    184 non-null   float32 
 1   VOL_MAX_SUBTER_M3    184 non-null   float32 
 2   VOLUME_MAX_MUN_M3    184 non-null   float32 
 3   VOLUME_MUN_ABHUM_2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020     184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX   184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_0MS      184 non-null   float32 
 10  NIVEL                184 non-null   int32  
dtypes: float32(8), int32(3)
memory usage: 8.0 KB

print('AB_HUM_TREE_COPIA')
AB_HUM_TREE_COPIA.info()

AB_HUM_TREE_COPIA
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
 ---  -- 
 0   VOL_MAX_SUPERF_M3    184 non-null   float32 
 1   VOL_MAX_SUBTER_M3    184 non-null   float32 
 2   VOLUME_MAX_MUN_M3    184 non-null   float32 
 3   VOLUME_MUN_ABHUM_2020_M3 184 non-null   float32 
 4   DIF_PROP_VOL_MAX_VOL_2020 184 non-null   float32 
 5   POP_ESTIMADA_2020     184 non-null   int32  
 6   POP_EST_ANO_VOL_MAX   184 non-null   int32  
 7   L_HAB_DIA_ANO_VOL_MAX 184 non-null   float32 
 8   L_HAB_DIA_ANO_VOL_2020 184 non-null   float32 
 9   VOL_2020_MIN_0MS      184 non-null   float32 
dtypes: float32(8), int32(2)
memory usage: 7.3 KB
```

A base para uso no modelo tem 10 colunas e 184 linhas é uma base pequena, então vamos ver como se comporta.

```
# DEFININDO A BASE SEM RÓTULO NO ALVO NA VARIÁVEL 'X'
X = AB_HUM_TREE_COPIA # Definindo os atributos

# CODIFICANDO A VARIÁVEL
LE = LabelEncoder()
alvo = LE.fit_transform(alvo)

#DEFININDO A VARIÁVEL 'Y' (alvo)
y = alvo
```

Dividindo o conjunto de dados em conjuntos de treinamento e teste, sendo 20% de registros selecionados aleatoriamente com semente 26 para teste

```
#DIVIDINDO O CONJUNTO DE DADOS EM CONJUNTOS DE TREINAMENTO E TESTE, SENDO 20% DE REGISTROS SELECIONADOS ALEATORIAMENTE COM SEMEDE 26

X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2, random_state = 26)
print("Dimensões dos conjuntos de treino e teste:")
print("Entrada de divisão de treinamento: ", X_train.shape)
print("Testando entrada de divisão: ", X_test.shape)

Dimensões dos conjuntos de treino e teste:
Entrada de divisão de treinamento: (147, 10)
Testando entrada de divisão: (37, 10)
```

Renderizando uma árvore de decisão de forma interativa.

```
### ALGORITMO ÁRVORE DE DECISÃO

#RENDERIZANDO A ÁRVORE DE FORMA INTERATIVA:

# MATRIZ DE RECURSOS
X,y = AB_HUM_TREE_COPIA, AB_HUM_MODEL_TREE['NIVEL']

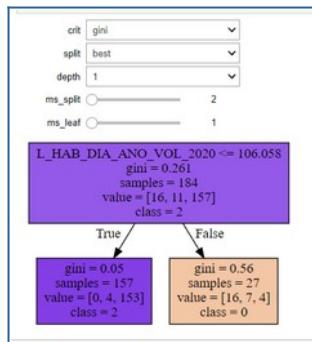
#RÓTULOS DE RECURSOS
features_label = AB_HUM_TREE_COPIA.columns

# ETIQUETA DE CLASSE
class_label = ['0','1','2']

def plot_tree(crit, split, depth, ms_split, ms_leaf=0.2):#Definindo uma função para receber os parâmetros de controle
    estimator = DecisionTreeClassifier(
        random_state = 26
        ,criterion = crit
        ,splitter = split
        ,max_depth = depth
        ,min_samples_split=ms_split
        ,min_samples_leaf=ms_leaf
    )
    estimator.fit(X, y)
    graph = Source(export_graphviz(estimator
        , out_file=None
        , feature_names=features_label
        , class_names=class_label
        , impurity=True
        , filled = True))
    display(SVG(graph.pipe(format='svg')))
    return estimator

inter=interactive(plot_tree
    , crit = ["gini", "entropy"] # Métrica adotada.
    , split = ["best", "random"] # Estratégia utilizada para dividir o nó de decisão.
    , depth=[1,2,3,4,5,10,20,30] # Profundidade máxima da árvore.
    , ms_split=(2,5) # Número de amostras mínimas para considerar um nó para divisão.
    , ms_leaf=(1,5)) # Número de amostras mínimas no nível folha.

display(inter)
```



A ideia é utilizar a árvore de decisão interativa para realizar simulações e fazer ajustes no modelo, quando necessário.

Abaixo uma tabela com algumas simulações realizadas, a tabela completa com as simulações foi apensada no arquivo ‘Tabelas.ods’.

DecisionTreeClassifie – Modelos		M1	M2	M3
Base de Dados	total registros	184	184	184
	qtd total atributos	9	9	9
	target	1	1	1
	classes	3	3	3
Partição dos Dados	test_size %	20	20	20
	X_train	147	147	147
	X_test	10	10	10
	y_train	37	37	37
	y_test	10	10	10
Argumentos	random_state	26	26	26
	criterion:	gini	gini	gini
	splitter	best	best	best
	max_depth	30	30	30
	min_samples_split	2	2	2
Resultados	min_samples_leaf	1	2	3
	nodes total (com folhas)	17	17	15
	node penúltima camada	# 3	# 3	# 8
	atributo	DIF_PROP_VOL_MAX_VOL_2020	DIF_PROP_VOL_MAX_VOL_2020	VOL_MAX_SUBTER_M3
	valor	<= -0.368	<= -0.368	<= 532423.688
	índice penúltima camada	0,494	0,494	0,469
	folha de maior índice	todos	todos	#9
	maior índice na folha	0,000	0,000	0,480

Dentre as simulações, foram testados os argumentos de M7, M9 e M12, com índice *gini* zero e divisão dos dados de treino e teste utilizando a técnica Holdout. Lembrando que a base é pequena, outra técnica de dividir e testar as amostras é recomendada pela literatura, trata-se do *cross validation*, onde a cada rodada de um rodízio uma nova amostra de treinamento realiza o treinamento e depois os testes com cada uma das demais parcelas. Para validar o modelo é obtida a média da métrica de validação adotada nas rodadas. Foram aplicados para 5 e 10 *folds*.

Observando 'M7'

```
# INSTANCIANDO O CLASSIFICADOR:
dtree=DecisionTreeClassifier(criterion = 'gini' , max_depth = 10, min_samples_split= 3, min_samples_leaf= 2)
```

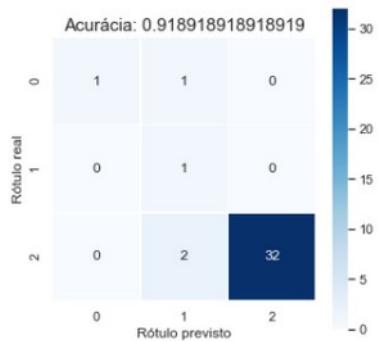
Classificador de árvore de decisão criado
Importância do atributo na definição do resultado da classificação

```
VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.0
VOLUME_MAX_MUN_M3:0.017826825127334477
VOLUME_MUN_ABHUM_2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.11205980612300785
POP_ESTIMADA_2020:0.0
POP_EST_ANO_VOL_MAX:0.0
L_HAB_DIA_ANO_VOL_MAX:0.10087011884550086
L_HAB_DIA_ANO_VOL_2020:0.688131195574785
VOL_2020_MIN_OMS:0.08111205432937184
```

Resultado da predição
[2 2 2 2 2 1 2 1 2]

Relatório de classificação

	precision	recall	f1-score	support
0	1.00	0.50	0.67	2
1	0.25	1.00	0.40	1
2	1.00	0.94	0.97	34
accuracy			0.92	37
macro avg	0.75	0.81	0.68	37
weighted avg	0.98	0.92	0.94	37



```
#CROSS VALIDATION
```

```
#INSERINDO OS PARÂMETROS NO MODELO
```

```
score= cross_val_score(dtree, X, y, cv=5, scoring='accuracy')
```

```
#IMPRIMINDO
scores
```

```
array([0.94594595, 0.94594595, 0.91891892, 0.89189189, 0.91666667])
```

```
#MÉDIA DOS RESULTADOS DOS SCORES
scores.mean()
```

```
0.9238738738738739
```

```
#INSERINDO NOVOS OS PARÂMETROS NO MODELO
```

```
scores = cross_val_score(dtree, X, y, cv=10, scoring='accuracy')
```

```
#IMPRIMINDO
scores
```

```
array([1.          , 0.94736842, 0.89473684, 0.94736842, 1.          ,
       0.94444444, 0.94444444, 0.88888889, 0.88888889, 1.          ])
```

```
#MÉDIA DOS RESULTADOS DOS SCORES
scores.mean()
```

```
0.9456140350877194
```

Observando 'M9'

```
# INSTANCIANDO O CLASSIFICADOR:
dtree=DecisionTreeClassifier(criterion = 'gini' , max_depth = 10, min_samples_split= 5, min_samples_leaf= 2)
```

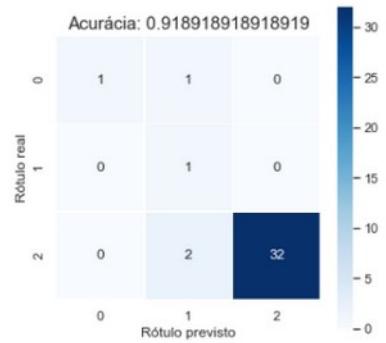
Classificador de árvore de decisão criado
Importância do atributo na definição do resultado da classificação

```
VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.0
VOLUME_MAX_MUN_M3:0.10018911716668102
VOLUME_MUN_ABHUM_2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.11347584596656343
POP_ESTIMADA_2020:0.0
POP_EST_ANO_VOL_MAX:0.0
L_HAB_DIA_ANO_VOL_MAX:0.22535029657010233
L_HAB_DIA_ANO_VOL_2020:0.5609847402966532
VOL_2020_MIN_0MS:0.0

Resultado da predição
[2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2]
```

Relatório de classificação

	precision	recall	f1-score	support
0	1.00	0.50	0.67	2
1	0.25	1.00	0.40	1
2	1.00	0.94	0.97	34
accuracy			0.92	37
macro avg	0.75	0.81	0.68	37
weighted avg	0.98	0.92	0.94	37



```
#CROSS VALIDATION
#INSERINDO OS PARÂMETROS NO MODELO
score= cross_val_score(dtree, X, y, cv=5, scoring='accuracy')
```

```
#IMPRIMINDO
scores
array([1.          , 0.94736842, 0.89473684, 0.94736842, 1.          ,
       0.94444444, 0.94444444, 0.88888889, 0.88888889, 1.          ],)
```

```
#MÉDIA DOS RESULTADOS DOS SCORES
scores.mean()
```

```
0.9456140350877194
```

```
#INSERINDO NOVOS OS PARÂMETROS NO MODELO
scores = cross_val_score(dtree, X, y, cv=10, scoring='accuracy')
```

```
#IMPRIMINDO
scores
array([1.          , 0.94736842, 0.89473684, 1.          , 1.          ,
       0.94444444, 0.94444444, 0.88888889, 0.88888889, 1.          ],)
```

```
#MÉDIA DOS RESULTADOS DOS SCORES
scores.mean()
```

```
0.9508771929824562
```

- Observando 'M12'

```
# INSTANCIANDO O CLASSIFICADOR:
dtree=DecisionTreeClassifier(criterion = 'gini' , max_depth = 5, min_samples_split= 5, min_samples_leaf= 2)
```

Classificador de árvore de decisão criado
 Importância do atributo na definição do resultado da classificação

```
VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.0
VOLUME_MAX_MUN_M3:0.018052093183185777
VOLUME_MUN_ABHUM_2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.11347584596656343
POP_ESTIMADA_2020:0.08213702398349525
POP_EST_ANO_VOL_MAX:0.0
L_HAB_DIA_ANO_VOL_MAX:0.22535029657010233
L_HAB_DIA_ANO_VOL_2020:0.5609847402966532
VOL_2020_MIN_OMS:0.0

Resultado da predição
[2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2]

Relatório de classificação
```

	precision	recall	f1-score	support
0	1.00	0.50	0.67	2
1	0.25	1.00	0.48	1
2	1.00	0.94	0.97	34
accuracy			0.92	37
macro avg	0.75	0.81	0.68	37
weighted avg	0.98	0.92	0.94	37

Acurácia: 0.918918918918919

Rótulo real	0	1	2
0	1	1	0
1	0	1	0
2	0	2	32

Rótulo previsto

```
#CROSS VALIDATION
#INSERINDO OS PARÂMETROS NO MODELO
scores= cross_val_score(dtree, X, y, cv=5, scoring='accuracy')
```

```
#IMPRIMINDO
scores
```

```
array([1.          , 0.94736842, 0.89473684, 1.          , 1.          ,
       1.          , 0.94444444, 0.88888889, 0.88888889, 1.          ],)
```

```
#MÉDIA DOS RESULTADOS DOS SCORES
scores.mean()
```

```
0.9564327485380117
```

```
#INSERINDO NOVOS OS PARÂMETROS NO MODELO
scores = cross_val_score(dtree, X, y, cv=10, scoring='accuracy')
```

```
#IMPRIMINDO
scores
```

```
array([1.          , 0.94736842, 0.89473684, 0.94736842, 1.          ,
       1.          , 0.94444444, 0.88888889, 0.88888889, 1.          ],)
```

```
#MÉDIA DOS RESULTADOS DOS SCORES
scores.mean()
```

```
0.9511695906432749
```

Observando os resultados obtidos até o momento.

Modelos			M7	M9	M12	
Controles	max_depth	10	10	5		
	min_samples_split	3	5	5		
	min_samples_leaf	2	2	2		
Partições			Métrica	Resultados		
HOLD-OUT			accuracy	0,9189	0,9189	
CROSS VALIDATION	Nº folds	5	score	0,9239	0,9456	
		10	Mean	0,9456	0,9509	
Média das avaliações				0,9295	0,9385	
					0,9368	

Com a técnica Hold-out, a acurácia apresentou o mesmo resultado nos modelos selecionados.

O *score mean*, quando aplicados com 5 *folds*, apresentou variabilidade com amplitude menor entre as suas execuções de testes , já quando aplicado com 10 *folds* os resultados tiveram amplitude maior. Os valores acima representam apenas uma dessas aplicações, permanecendo em torno de 0,94 nas 02 configurações de *folds*.

Com a aplicação *Cross Validation* uma pequena melhora em todos. As melhores performances média foram dos modelos M9 e M12.

Os resultados ainda não foram totalmente convincentes, então foi selecionado o M12 para aplicação de mais algumas métricas por ter apresentado um dos melhores resultados com profundidade máxima menor.

Observando novamente o balanceamento de classes da base rotulada.

```
#OBSERVANDO O BALANCEAMENTO DE CLASSES NOVAMENTE
print('Nível Quantidade')
print(AB_HUM_MODEL_TREE.NIVEL.value_counts())
print("\nNível 1 representa {:.4f}% do dataset.\n".format((AB_HUM_MODEL_TREE[AB_HUM_MODEL_TREE.NIVEL == 1].shape[1] / AB_HUM_MODEL_TREE.shape[1]) * 100))
print("\nNível 0 representa {:.4f}% do dataset.\n".format((AB_HUM_MODEL_TREE[AB_HUM_MODEL_TREE.NIVEL == 0].shape[0] / AB_HUM_MODEL_TREE.shape[0]) * 100))
#GRÁFICO DE BARRAS COM AS CLASSES
sns.countplot('NIVEL', data=AB_HUM_MODEL_TREE);
```

Nível	Quantidade
2	157
0	16
1	11

Name: NIVEL, dtype: int64

Nível 1 representa 5.9783% do dataset.

Nível 0 representa 8.6957% do dataset.

Observando novamente o balanceamento de classes de treino.

```
#BALANCEAMENTO DOS DADOS DAS AMOSTRAS DE TREINO
print('BALANCEAMENTO DOS DADOS DAS AMOSTRAS DE TREINO')

# VER O BALANCEAMENTO DAS CLASSES
print(pd.Series(y_train).value_counts())

# PLOTAR A DISTRIBUIÇÃO DE CLASSES
sns.countplot(y_train);
```

Nível	Quantidade
2	123
0	14
1	10

dtype: int64

O modelo parece promissor quanto a acurácia, e isso pode ser consequência do desbalanceamento dos dados que tende a classificar bem a classe majoritária. Mesmo sendo ela a mais crítica e de maior interesse, pois seria para onde os esforços de regularização se concentrariam, é importante, considerando esse vasto campo, não desperdiçar energia onde não for pertinente, e o modelo deve estar mais assertivo para facilitar esse planejamento classificando bem as classes minoritárias também.

Dito isso, recorremos a algumas técnicas para balanceamento de dados durante o treinamento a fim de comparar, através das respectivas métricas, o desempenho do modelo com a aplicação das mesmas.

A abordagem *sampling* tem sido bastante usada, trata-se de um pré-processamento para reduzir as diferenças quantitativas entre as classes através de reamostragem. As técnicas mais usadas para gerar essa nova amostragem são:

- *Over-sampling*, onde novas entradas são criadas a partir dos dados originais da classe minoritária, que podem ser aleatoriamente ou não.
- *Under-sample*, onde entradas da classe majoritária são aleatoriamente eliminadas.

As técnicas de balanceamento a seguir foram aplicadas na base de treino para preservar a de teste:

- *Randomundersampler* (RUS) – Duplica as amostras originais da classe minoritária
- *Synthetic Minority Supersampling* (SMOTE) - Gera amostras sem fazer distinção se são fáceis ou difíceis de serem classificadas
- *Adaptive Synthetic Sampling* (ADASYN) - Gera novas amostras parecidas com as originais

Aplicando a técnica *Under-sampling* (RUS)

```
#BALANCEAMENTO DOS DADOS COM Under-Sampling (RUS)
print('BALANCEAMENTO DOS DADOS COM Under-Sampling (RUS)')

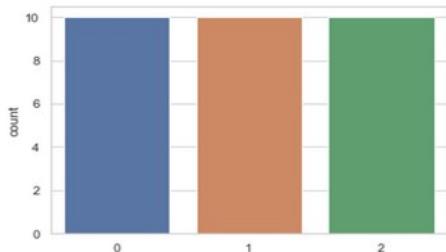
# BALANCEAMENTO DOS DADOS
rus = RandomUnderSampler()
X_rus, y_rus = rus.fit_resample(X_train, y_train)

# VER O BALANCEAMENTO DAS CLASSES
print(pd.Series(y_rus).value_counts())

# PLOTAR A NOVA DISTRIBUIÇÃO DE CLASSES
sns.countplot(y_rus);
```

BALANCEAMENTO DOS DADOS COM Under-Sampling (RUS)

```
0    10
1    10
2    10
dtype: int64
```



```
# INSTANCIANDO O CLASSIFICADOR:
model_tree_rus = DecisionTreeClassifier(criterion = 'gini', max_depth = 5, min_samples_split= 5, min_samples_leaf= 2)

# TREINANDO O MODELO DE ARVORE DE DECISÃO: COM DADOS RUS
model_tree_rus.fit(X_rus, y_rus)

# RESULTADO DA CLASSIFICAÇÃO
y_pred_tree_rus = model_tree_rus.predict(X_test)

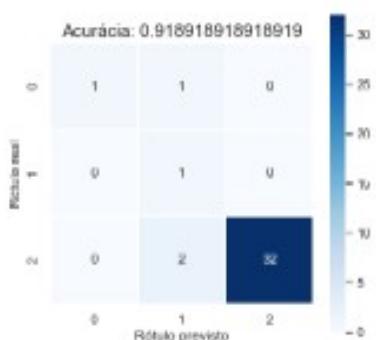
#AVALIAÇÃO DO MODELO
print("Relatório de classificação ")
print("")
print(metrics.classification_report(y_test,y_pred_tree_rus))

# MATRIZ DE CONFUSÃO
cmrus = confusion_matrix(y_test, y_pred_tree_rus)
plt.figure(figsize=(5,5))
mconf = sns.heatmap(data=cmrus,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.xlabel("Rótulo real")
plt.ylabel("Rótulo previsto")
all_sample_title = 'Acurácia: {:.2f}'.format(model_tree_rus.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

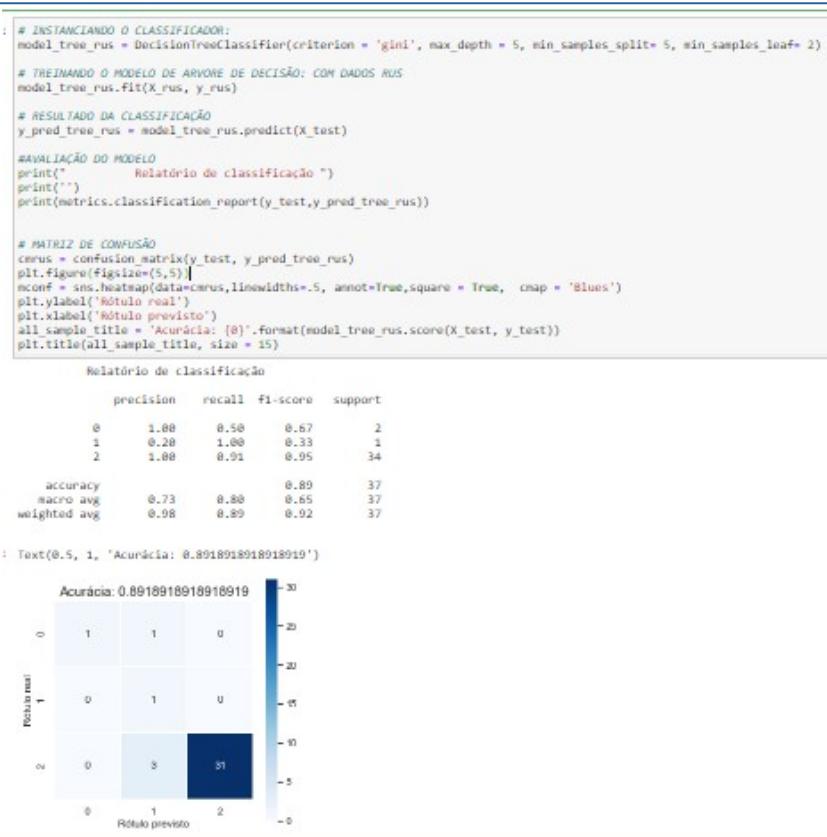
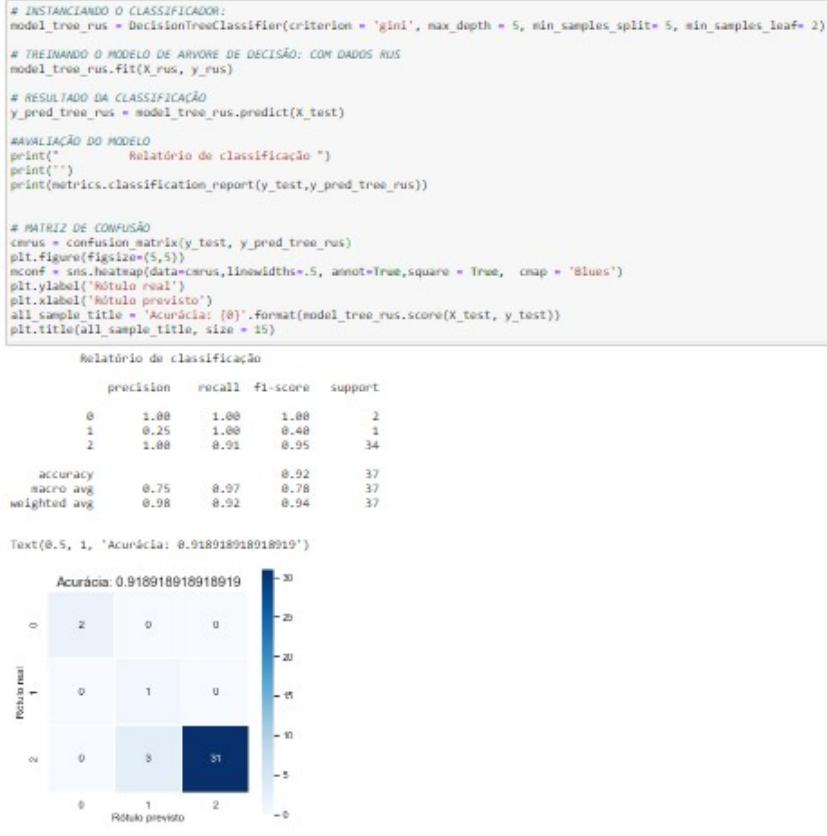
Relatório de classificação

	precision	recall	f1-score	support
0	1.00	0.50	0.67	2
1	0.25	1.00	0.48	1
2	1.00	0.94	0.97	34
accuracy			0.92	37
macro avg	0.75	0.81	0.68	37
weighted avg	0.98	0.92	0.94	37

Text(0.5, 1, 'Acurácia: 0.918918918918919')



Aplicando mais algumas vezes a técnica *Under-sampling* (RUS).



Aplicando técnica Over-Sampling (SMOTE)

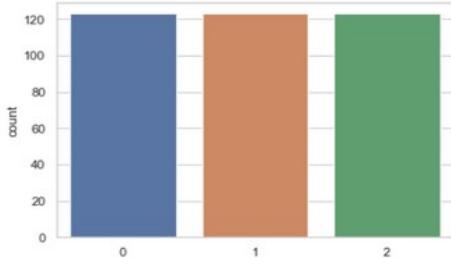
```
# BALANCEAMENTO DOS DADOS COM Over-Sampling (SMOTE)
print('BALANCEAMENTO DOS DADOS COM Over-Sampling (SMOTE)')

# BALANCEAMENTO DOS DADOS
smo = SMOTE()
X_smo, y_smo = smo.fit_resample(X_train, y_train)

# CHECAR O BALANCEAMENTO DAS CLASSES
print(pd.Series(y_smo).value_counts())

# PLOTAR A NOVA DISTRIBUIÇÃO DE CLASSES
sns.countplot(y_smo);
```

BALANCEAMENTO DOS DADOS COM Over-Sampling (SMOTE)
0 123
1 123
2 123
dtype: int64



```
# INSTANCIANDO O CLASSIFICADOR:
model_tree_smo = DecisionTreeClassifier(criterion = 'gini', max_depth = 5, min_samples_split= 5, min_samples_leaf= 2)

# TREINANDO O MODELO DE ARVORE DE DECISÃO: COM DADOS SMOTE
model_tree_smo.fit(X_smo, y_smo)

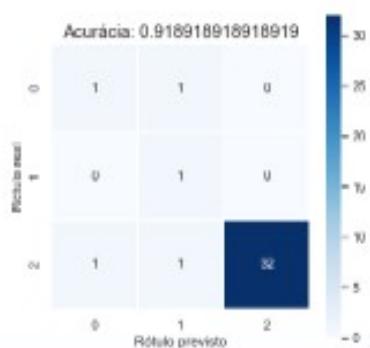
# RESULTADO DA CLASSIFICAÇÃO
y_pred_tree_smo = model_tree_smo.predict(X_test)
print("Relatório de classificação ")
print("")
print(metrics.classification_report(y_test,y_pred_tree_smo))

# MATRIZ DE CONFUSÃO
cmsmo = confusion_matrix(y_test, y_pred_tree_smo)
plt.figure(figsize=(5,5))
mconf = sns.heatmap(data=cmsmo, linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {0}'.format(model_tree_smo.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

Relatório de classificação

	precision	recall	f1-score	support
0	0.50	0.50	0.50	2
1	0.33	1.00	0.50	1
2	1.00	0.94	0.97	34
accuracy			0.92	37
macro avg	0.61	0.81	0.66	37
weighted avg	0.95	0.92	0.93	37

: Text(0.5, 1, 'Acurácia: 0.918918918918919')



Aplicando técnica Over-Sampling (ADASYN)

```
#BALANCEAMENTO DOS DADOS COM Over-Sampling (ADASYN)
print('BALANCEAMENTO DOS DADOS COM Over-Sampling (ADASYN)')

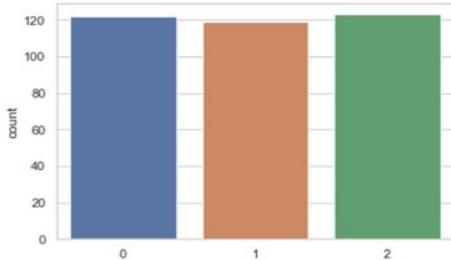
# BALANCEAMENTO DOS DADOS
from imblearn.over_sampling import ADASYN

# DEFININDO PARÂMETROS
ada = ADASYN()
X_ada, y_ada = ada.fit_resample(X_train, y_train)

# CHECAR O BALANCEAMENTO DAS CLASSES
print(pd.Series(y_ada).value_counts())

# PLOTAR A NOVA DISTRIBUIÇÃO DE CLASSES
sns.countplot(y_ada);
```

BALANCEAMENTO DOS DADOS COM Over-Sampling (ADASYN)
2 123
0 122
1 119
dtype: int64



```
# INSTANCIANDO O CLASSIFICADOR:
model_tree_ada = DecisionTreeClassifier(criterion = 'gini', max_depth = 5, min_samples_split= 5, min_samples_leaf= 2)

# TREINANDO O MODELO DE ARVORE DE DECISÃO: COM DADOS ADASYN
model_tree_ada.fit(X_ada, y_ada)

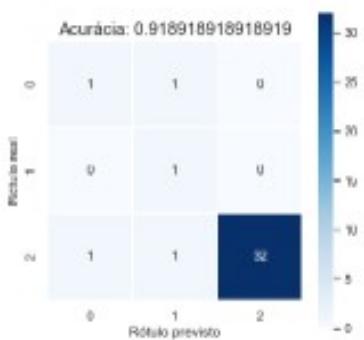
# RESULTADO DA CLASSIFICAÇÃO
y_pred_tree_ada = model_tree_ada.predict(X_test)
print("Relatório de classificação")
print("")
print(metrics.classification_report(y_test,y_pred_tree_ada))

# MATRIZ DE CONFUSÃO
cmada = confusion_matrix(y_test, y_pred_tree_ada)
plt.figure(figsize=(5,5))
mconf = sns.heatmap(data=cmada,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(model_tree_ada.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

Relatório de classificação

	precision	recall	f1-score	support
0	0.58	0.58	0.58	2
1	0.33	1.00	0.58	1
2	1.00	0.94	0.97	34
accuracy			0.92	37
macro avg	0.61	0.81	0.66	37
weighted avg	0.95	0.92	0.93	37

: Text(0.5, 1, 'Acurácia: 0.918918918918919')



Aplicando a técnica de validação *cross validation* com 5 e 10 folds nos 03 modelos treinados com dados balanceados :

- *Under-sampling (RUS)*

```
#CROSS VALIDATION (RUS)

#INSERINDO OS PARÂMETROS NO MODELO
scores_tree_rus5 = cross_val_score(model_tree_rus, X, y, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_tree_rus5

#MÉDIA DOS RESULTADOS DOS SCORES
scores_tree_rus5.mean()

0.9346846846846848

#INSERINDO OS PARÂMETROS NO MODELO
scores_tree_rus10 = cross_val_score(model_tree_rus, X, y, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_tree_rus10

#MÉDIA DOS RESULTADOS DOS SCORES
scores_tree_rus10.mean()

0.9564327485380117
```

- *Over-Sampling (SMOTE)*

```
#CROSS VALIDATION (SMOTE)

#INSERINDO OS PARÂMETROS NO MODELO
scores_tree_smo5 = cross_val_score(model_tree_smo, X, y, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_tree_smo5

#MÉDIA DOS RESULTADOS DOS SCORES
scores_tree_smo5.mean()

0.9238738738738739

#INSERINDO OS PARÂMETROS NO MODELO
scores_tree_smo10 = cross_val_score(model_tree_smo, X, y, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_tree_smo10

#MÉDIA DOS RESULTADOS DOS SCORES
scores_tree_smo10 .mean()

0.9511695906432749
```

Over-Sampling (ADASYN)

```
#CROSS VALIDATION (ADASYN)
#INSERINDO OS PARÂMETROS NO MODELO
scores_tree_ada5 = cross_val_score(model_tree_ada, X, y, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_tree_ada5

#MÉDIA DOS RESULTADOS DOS SCORES
scores_tree_ada5.mean()

0.9346846846846848
```

```
#INSERINDO OS PARÂMETROS NO MODELO
scores_tree_ada10 = cross_val_score(model_tree_ada, X, y, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_tree_ada10

#MÉDIA DOS RESULTADOS DOS SCORES
scores_tree_ada10.mean()

0.9564327485380117
```



Resultados obtidos nas métricas de avaliação dos modelos com dados desbalanceados e após a aplicação das técnicas de balanceamento das amostras.

DADOS DESBALANCEADOS				
CLASSES	0	1	2	
	PREVISTO			
0	R E	1	1	0
1	E A	0	1	0
2	L	0	2	32

DADOS BALANCEADOS - Under-Sampling (RUS)				
CLASSES	0	1	2	
	PREVISTO			
0	R E	1	1	0
1	E A	0	1	0
2	L	0	2	32

DADOS BALANCEADOS - Over-Sampling (SMOTE)				
CLASSES	0	1	2	
	PREVISTO			
0	R E	1	1	0
1	E A	0	1	0
2	L	1	1	32

DADOS BALANCEADOS - Over-Sampling (ADASYN)				
CLASSES	0	1	2	
	PREVISTO			
0	R E	1	1	0
1	E A	0	1	0
2	L	1	1	32

dados	Algoritmo		ÁRVORE DE DECISÃO						
	classes	amostra s treino	precision	recall	f1-score	specificity	precision pos	precision neg	support
D E S B A L C D A E O N A S	0	14	1,0000	0,5000	0,6667	1,0000	1,0000	0,9706	2
	1	10	0,2500	1,0000	0,4000	0,9706	0,2500	1,0000	1
	2	123	1,0000	0,9412	0,9697	1,0000	1,0000	0,5000	34
	macro avg		0,7500	0,8137	0,6788	0,9902	0,7500	0,8235	total
	weighted avg		0,9797	0,9189	0,9379	0,9991	0,9779	0,5285	37
	Under-Sampling (RUS)								
B A L A N C E A D O S	0	10	1,0000	0,5000	0,6667	1,0000	1,0000	0,9706	2
	1	10	0,2500	1,0000	0,4000	0,9167	0,2500	1,0000	1
	2	10	1,0000	0,9412	0,9697	1,0000	1,0000	0,5000	34
	macro avg		0,7500	0,8137	0,6788	0,9722	0,7500	0,8235	total
	weighted avg		0,9797	0,9189	0,9379	0,9975	0,9779	0,5285	37
	Over-Sampling (SMOTE)								
N C E A D O S	0	123	0,5000	0,5000	0,5000	0,9706	0,5000	0,9706	2
	1	123	0,3333	1,0000	0,5000	0,9429	0,0000	1,0000	1
	2	123	1,0000	0,9412	0,9697	1,0000	0,0000	0,5000	34
	macro avg		0,6111	0,8137	0,6566	0,9711	0,1667	0,8235	total
	weighted avg		0,9550	0,9189	0,9316	0,9975	0,0147	0,5285	37
	Over-Sampling (ADASYN)								
D O S	0	123	0,5000	0,5000	0,5000	0,9706	0,5000	0,9706	2
	1	122	0,3333	1,0000	0,5000	0,9429	0,0000	1,0000	1
	2	119	1,0000	0,9412	0,9697	1,0000	0,0000	0,5000	34
	macro avg		0,6111	0,8137	0,6566	0,9711	0,1667	0,8235	total
	weighted avg		0,9550	0,9189	0,9316	0,9975	0,0147	0,5285	37

DADOS DA AMOSTRAGEM TREINO	VALIDAÇÃO			
	ÚNICA	cross validation		
		nº de folds		
		5	10	
BALANCEADOS	DESBALANCEADOS	0,9189	0,9347	0,9512
	Under-Sampling (RUS)	0,9189	0,9347	0,9564
	Over-Sampling (SMOTE)	0,9189	0,9239	0,9512
	Over-Sampling (ADASYN)	0,9189	0,9347	0,9564

Os controles utilizados com os dados desbalanceados foram mantidos na execução dos testes com dados balanceados.

Os testes aplicando RUS apresentaram resultados aleatórios, o que já seria esperado, pois as amostras para os treinos apesar do equilíbrio quanto a representatividade dos dados, tiveram um quantitativo baixo em relação as demais. Embora esse quantitativo amostras para treino esteja próximo das amostras de teste, a cada execução tem-se um novo grupo de dados de treino para ser testado no mesmo grupo de dados para teste, pois os dados para teste foram separados previamente. Na planinha acima constam os resultados apenas da última execução.

Classificando os dados com a técnica Naive Bayes

```
# CLASSIFICAÇÃO NAIVE BAYES

# CRIAR CÓPIA DO DATAFRAME ROTULADO
AB_HUM_MODEL_bayes = AB_HUM_MODEL_TREE.copy()

# SEPARAR OS DADOS ENTRE MATRIZ RECURSOS E ALVO
Xnb = AB_HUM_MODEL_bayes.drop('NIVEL', axis=1)
ynb = AB_HUM_MODEL_bayes['NIVEL']

#DIVIDIR OS DADOS EM TREINO E TESTE
Xnb_train, Xnb_test, ynb_train, ynb_test = train_test_split(Xnb, ynb, test_size=0.3, random_state=109)

#INSTANCIAR O CLASSIFICADOR
gnb = GaussianNB()

#TREINAR COM OS DADOS RESERVADOS PARA TREINO
gnb.fit(Xnb_train, ynb_train)

# RESULTADO DA CLASSIFICAÇÃO
ynb_pred_gnb = gnb.predict(Xnb_test)
#resultado = ynb_pred_gnb
print('')
print('Resultado da predição')
print(ynb_pred_gnb)
print('')
print("          Relatório de classificação ")
print('')
print(metrics.classification_report(ynb_test, ynb_pred_gnb))

# MATRIZ DE CONFUSÃO
cm_gnb = confusion_matrix(ynb_test, ynb_pred_gnb)
plt.figure(figsize=(5,5))
mconf_gnb = sns.heatmap(data=cm_gnb, linewidths=.5, annot=True, square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(gnb.score(Xnb_test, ynb_test))
plt.title(all_sample_title, size = 15)
```

Obtendo os resultados

```
Resultado da predição
[2 2 2 0 2 2 2 0 0 0 2 2 2 0 2 2 0 2 2 2 0 2 0 2 2 2 0 2 2 2 0 2 2
 2 2 2 2 0 0 0 0 0 2 2 2 2 0 2 0 0 0 2]

Relatório de classificação

      precision    recall  f1-score   support
0        0.05     1.00    0.09       1
1        0.00     0.00    0.00       4
2        1.00     0.69    0.81      51

   accuracy         0.64      56
  macro avg     0.35     0.56    0.30      56
weighted avg    0.91     0.64    0.74      56

C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

Alguns rótulos em `y_teste` não aparecem em `y_pred`, nunca são previstos, possivelmente uma ocorrência de Probabilidade Zero. Isso significa que não há pontuação a ser calculada para esse rótulo e, portanto, a pontuação para esse caso é considerada '0,0', no caso o rótulo '1'



```

#INSTANCIAR O CLASSIFICADOR
gnb_rus = GaussianNB()

#TREINAR COM OS DADOS RESERVADOS PARA TREINO
gnb_rus.fit(X_rus_nb, y_rus_nb)

# RESULTADO DA CLASSIFICAÇÃO
ynb_pred_gnb_rus = gnb_rus.predict(Xnb_test)

print('')
print('Resultado da predição')
print(ynb_pred_gnb_rus)
print('')
print("Relatório de classificação ")
print('')
print(metrics.classification_report(ynb_test, ynb_pred_gnb_rus))

# MATRIZ DE CONFUSÃO

cm_gnb_rus = confusion_matrix(ynb_test, ynb_pred_gnb_rus)
plt.figure(figsize=(5,5))
mconf_gnb_rus= sns.heatmap(data=cm_gnb_rus , linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(gnb_rus.score(Xnb_test, ynb_test))
plt.title(all_sample_title, size = 15)

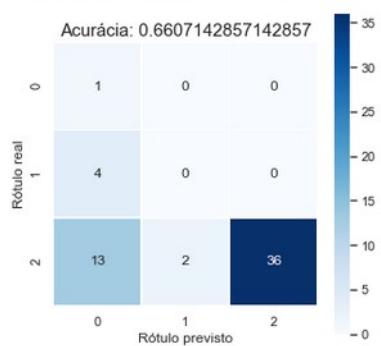
```

Resultado da predição
[2 2 2 0 1 2 2 0 0 2 2 2 2 0 2 2 2 2 2 0 2 1 2 0 2 0 2 0 0 2 2 2 2 0 2 2
2 2 2 2 0 2 2 0 0 0 2 2 2 0 2 0 0 2]

Relatório de classificação

	precision	recall	f1-score	support
0	0.06	1.00	0.11	1
1	0.00	0.00	0.00	4
2	1.00	0.71	0.83	51
accuracy			0.66	56
macro avg	0.35	0.57	0.31	56
weighted avg	0.91	0.66	0.76	56

Text(0.5, 1, 'Acurácia: 0.6607142857142857')



```

: # BALANÇAMENTO DOS DADOS COM Over-Sampling (SMOTE)
print('BALANÇAMENTO DOS DADOS COM Over-Sampling (SMOTE)')

# BALANÇAMENTO DOS DADOS
smo_nb = SMOTE()
X_smo_nb, y_smo_nb = smo_nb.fit_resample(X_train, y_train)

# CHECAR O BALANÇAMENTO DAS CLASSES
print(pd.Series(y_smo_nb).value_counts())

# PLOTAR A NOVA DISTRIBUIÇÃO DE CLASSES
sns.countplot(y_smo_nb);

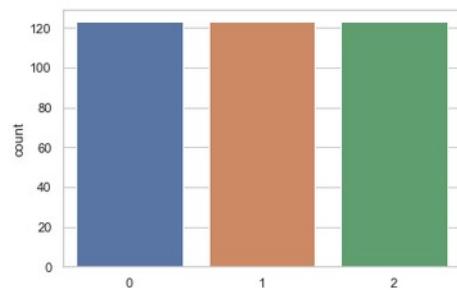
```

BALANÇAMENTO DOS DADOS COM Over-Sampling (SMOTE)

```

0    123
1    123
2    123
dtype: int64

```



```

#INSTANCIAR O CLASSIFICADOR
gnb_smo = GaussianNB()

#TREINAR COM OS DADOS RESERVADOS PARA TREINO
gnb_smo.fit(X_smo_nb, y_smo_nb)

# RESULTADO DA CLASSIFICAÇÃO
ynb_pred_gnb_smo = gnb_smo.predict(Xnb_test)

print('')
print('Resultado da predição')
print(ynb_pred_gnb_smo)
print('')
print("          Relatório de classificação ")
print('')
print(metrics.classification_report(ynb_test, ynb_pred_gnb_smo))

# MATRIZ DE CONFUSÃO
cm_gnb_smo = confusion_matrix(ynb_test, ynb_pred_gnb_smo)
plt.figure(figsize=(5,5))
mconf_gnb_smo= sns.heatmap(data=cm_gnb_smo ,linewdiths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(gnb_smo.score(Xnb_test, ynb_test))
plt.title(all_sample_title, size = 15)

```

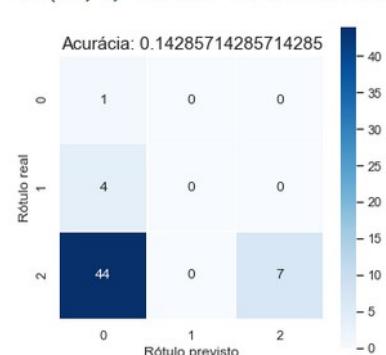
```
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

```
_warn_prf(average, modifier, msg_start, len(result))
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

```
_warn_prf(average, modifier, msg_start, len(result))
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

```
_warn_prf(average, modifier, msg_start, len(result))

Text(0.5, 1, 'Acurácia: 0.14285714285714285')
```



```
#BALANCEAMENTO DOS DADOS COM Over-Sampling (ADASYN)

print('BALANCEAMENTO DOS DADOS COM Over-Sampling (ADASYN)')

# BALANCEAMENTO DOS DADOS
#from imblearn.over_sampling import ADASYN

ada_nb = ADASYN()

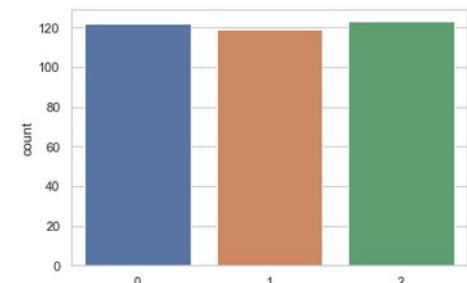
X_ada_nb, y_ada_nb = ada_nb.fit_resample(X_train, y_train)

# CHECAR O BALANCEAMENTO DAS CLASSES
print(pd.Series(y_ada_nb).value_counts())

# PLOTAR A NOVA DISTRIBUIÇÃO DE CLASSES
sns.countplot(y_ada_nb);
```

```
BALANCEAMENTO DOS DADOS COM Over-Sampling (ADASYN)
```

```
2    123
0    122
1    119
dtype: int64
```



```

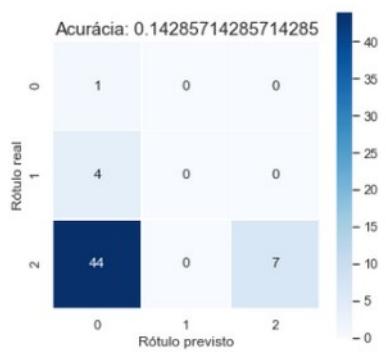
#INSTANCIAR O CLASSIFICADOR
gnb_ada = GaussianNB()

#TREINAR COM OS DADOS RESERVADOS PARA TREINO
gnb_ada.fit(X_ada_nb, y_ada_nb)

# RESULTADO DA CLASSIFICAÇÃO
ynb_pred_gnb_ada = gnb_ada.predict(Xnb_test)
#resultado = ynb_pred_gnb_ada
print('')
print('Resultado da predição')
print (ynb_pred_gnb_ada )
print('')
print("          Relatório de classificação ")
print('')
print(metrics.classification_report(ynb_test,ynb_pred_gnb_ada))

# MATRIZ DE CONFUSÃO
cm_gnb_ada = confusion_matrix(ynb_test, ynb_pred_gnb_ada)
plt.figure(figsize=(5,5))
mconf_gnb_ada = sns.heatmap(data=cm_gnb_ada ,linewdiths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(gnb_ada.score(Xnb_test, ynb_test))
plt.title(all_sample_title, size = 15)

```



```
#CROSS VALIDATION NB RUS

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb_rus5 = cross_val_score(gnb_rus, Xnb, ynb, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_gnb_rus5

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb_rus5.mean()

0.10330330330330331

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb_rus10 = cross_val_score(gnb_rus, Xnb, ynb, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_gnb_rus10

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb_rus10.mean()

0.11403508771929824
```

```
#CROSS VALIDATION NB SMOTE

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb_smo5 = cross_val_score(gnb_smo, Xnb, ynb, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_gnb_smo5

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb_smo5.mean()

0.10330330330330331

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb_smo10 = cross_val_score(gnb_smo, Xnb, ynb, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_gnb_smo10

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb_smo10.mean()

0.11403508771929824
```

```
#CROSS VALIDATION NB ADANYS

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb_ada_nb5 = cross_val_score(gnb_ada, Xnb, ynb, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_gnb_ada_nb5

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb_ada_nb5.mean()

0.10330330330330331

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb_ada_nb10 = cross_val_score(gnb_ada, Xnb, ynb, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_gnb_ada_nb10

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb_ada_nb10.mean()

0.11403508771929824
```

DADOS DESBALANCEADOS													
CLASSES		PREVISTO			CLASS '0'			CLASS '1'			CLASS '2'		
		0	1	2	1	0	0	1	0	0	1	0	0
0	R	1	0	0	1	0	0	1	0	0	1	0	0
1	E	4	0	0	4	0	0	4	0	0	4	0	0
2	L	16	0	35	16	0	35	16	0	35	16	0	35

DADOS BALANCEADOS - Under-Sampling (RUS)													
CLASSES		PREVISTO			CLASS '0'			CLASS '1'			CLASS '2'		
		0	1	2	1	0	0	1	0	0	1	0	0
0	R	1	0	0	1	0	0	1	0	0	1	0	0
1	E	4	0	0	4	0	0	4	0	0	4	0	0
2	L	13	2	36	13	2	36	13	2	36	13	2	36

DADOS BALANCEADOS - Over-Sampling (SMOTE)													
CLASSES		PREVISTO			CLASS '0'			CLASS '1'			CLASS '2'		
		0	1	2	1	0	0	1	0	0	1	0	0
0	R	1	0	0	1	0	0	1	0	0	1	0	0
1	E	4	0	0	4	0	0	4	0	0	4	0	0
2	L	44	0	7	44	0	7	44	0	7	44	0	7

DADOS BALANCEADOS - Over-Sampling (ADASYN)													
CLASSES		PREVISTO			CLASS '0'			CLASS '1'			CLASS '2'		
		0	1	2	1	0	0	1	0	0	1	0	0
0	R	1	0	0	1	0	0	1	0	0	1	0	0
1	E	4	0	0	4	0	0	4	0	0	4	0	0
2	L	44	0	7	44	0	7	44	0	7	44	0	7

Dados	Algoritmo		NAIVE BAYES								
	classes	amostras treino	precision	recall	f1-score	specificity	precision pos	precision neg	support	VP VN	
DESBALANÇADOS	0	14	0,0476	1,0000	0,0909	0,6364	0,0476	1,0000	1	1	
	1	10	#DIV/0!	0,0000	Erro:502	0,6923	#DIV/0!	0,9000	4	0	
	2	123	1,0000	0,6863	0,8140	1,0000	1,0000	0,0588	51	35	
	macro avg		#DIV/0!	0,5621	Erro:502	0,7762	#DIV/0!	0,6529	56	accuracy	
	weighted avg		#DIV/0!	0,6429	Erro:502	0,9899	#DIV/0!	0,0850	56	0,642857	
	Under-Sampling (RUS)										
BALANCEADOS	0	10	0,0556	1,0000	0,1053	0,6792	0,0556	1,0000	1	1	
	123	10	0,0000	0,0000	Erro:502	0,9487	0,0000	0,9024	4	0	
	122	10	1,0000	0,7059	0,8276	1,0000	1,0000	0,0625	51	36	
	macro avg		0,3519	0,5686	Erro:502	0,8760	0,3519	0,6550	56	accuracy	
	weighted avg		0,9117	0,6607	Erro:502	0,9913	0,9745	0,0878	56	0,660714	
Over-Sampling (SMOTE)											
BALANCEADOS	0	123	0,0204	1,0000	0,0400	0,1273	0,0204	1,0000	1	1	
	0	123	#DIV/0!	0,0000	#DIV/0!	1,0000	0,0000	0,6667	4	0	
	0	123	1,0000	0,1373	0,2414	1,0000	0,0000	0,0222	51	7	
	macro avg		#DIV/0!	0,3791	#DIV/0!	0,7091	0,0068	0,5630	56	accuracy	
	weighted avg		#DIV/0!	0,1429	#DIV/0!	0,8909	0,0026	0,1444	56	0,142857	
Over-Sampling (ADASYN)											
BALANCEADOS	0	123	0,0204	1,0000	0,0400	0,1273	0,0204	1,0000	1	1	
	1	122	#DIV/0!	0,0000	#DIV/0!	1,0000	0,0000	0,6667	4	0	
	2	119	1,0000	0,1373	0,2414	1,0000	0,0000	0,0222	51	7	
	macro avg		#DIV/0!	0,3791	#DIV/0!	0,7091	0,0068	0,5630	56	accuracy	
	weighted avg		#DIV/0!	0,1429	#DIV/0!	0,8909	0,0026	0,1444	56	0,142857	

DADOS DA AMOSTRAGEM TREINO	VALIDAÇÃO			
	ÚNICA	CRUZADA		
		nº de folds		
DESBALANCEADOS	0,0000	0,1033	0,1140	
BALANCEADOS	Under-Sampling (RUS)	0,0000	0,9347	0,9564
	Over-Sampling (SMOTE)	0,0000	0,9239	0,9512
	Over-Sampling (ADASYN)	0,0000	0,9347	0,9564

DESCONSIDERANDO A PREVISÃO ZERO – CLASSE '1'								
macro avg							accuracy	
precision	recall	f1-score	specificity	precision_pos	precision_neg	support	accuracy	
DESBALANCEADOS								
0,5238	0,8431	0,4524	0,8182	0,5238	0,5294	52	0,6923	
BALANCEADOS - Under-Sampling (RUS)								
0,5278	0,8529	0,4664	0,8396	0,5278	0,5313	52	0,0192	
BALANCEADOS - Over-Sampling (SMOTE)								
0,5102	0,5686	0,1407	0,5636	0,0102	0,5111	52	0,1538	
BALANCEADOS - Over-Sampling (ADASYN)								
0,5102	0,5686	0,1407	0,5636	0,0102	0,5111	52,0000	0,1538	

Alterando o tamanho da amostra de teste para observar

```
#ALTERANDO O TAMANHO DA AMOSTRAS DE TREINO PARARA OBSERVAR

#DIVIDIR OS DADOS EM TREINO E TESTE
Xnb2_train, Xnb2_test, ynb2_train, ynb2_test = train_test_split(Xnb, ynb, test_size=0.2,random_state=100)

#INSTANCIAR O CLASSIFICADOR
gnb2 = GaussianNB()

#TREINAR COM OS DADOS RESERVADOS PARA TREINO
gnb2.fit(Xnb2_train, ynb2_train)

# RESULTADO DA CLASSIFICAÇÃO
ynb2_pred_gnb2 = gnb2.predict(Xnb2_test)
#resultado = ynb2_pred_gnb2
print('')
print('Resultado da predição')
print (ynb2_pred_gnb2)
print('')
print("          Relatório de classificação ")
print('')
print(metrics.classification_report(ynb2_test,ynb2_pred_gnb2))

# MATRIZ DE CONFUSÃO
cm_gnb2 = confusion_matrix(ynb2_test, ynb2_pred_gnb2)
plt.figure(figsize=(5,5))
mconf_gnb2 = sns.heatmap(data=cm_gnb2 ,linewdiths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(gnb2.score(Xnb2_test, ynb2_test))
plt.title(all_sample_title, size = 15)
```

```

Resultado da predição
[0 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 2 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]

Relatório de classificação

      precision    recall   f1-score  support

          0       0.03     1.00     0.06      1
          1       0.00     0.00     0.00      1
          2       1.00     0.11     0.21     37

   accuracy                           0.14      37
macro avg       0.34     0.37     0.09      37
weighted avg    0.95     0.14     0.20      37

C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))

Text(0.5, 1, 'Acurácia: 0.13513513513513514')



```

```

#DIVIDIR OS DADOS EM TREINO E TESTE
Xnb5_train, Xnb5_test, ynb5_train, ynb5_test = train_test_split(Xnb, ynb, test_size=0.5,random_state=109)

#INSTANCIAR O CLASSIFICADOR
gnb5 = GaussianNB()

#TREINAR COM OS DADOS RESERVADOS PARA TREINO
gnb5.fit(Xnb5_train, ynb5_train)

# RESULTADO DA CLASSIFICAÇÃO
ynb5_pred_gnb5 = gnb5.predict(Xnb5_test)
#resultado = ynb5_pred_gnb5
print('')
print('Resultado da predição')
print(ynb5_pred_gnb5)
print('')
print("          Relatório de classificação ")
print('')
print(metrics.classification_report(ynb5_test,ynb5_pred_gnb5))

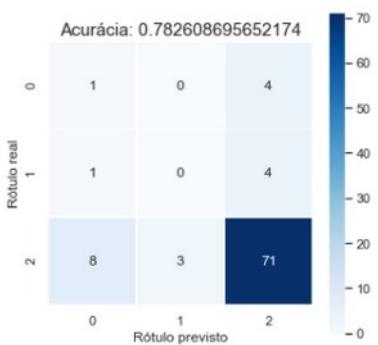
# MATRIZ DE CONFUSÃO
cm_gnb5 = confusion_matrix(ynb5_test, ynb5_pred_gnb5)
plt.figure(figsize=(5,5))
mconf_gnb5 = sns.heatmap(data=cm_gnb5 ,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(gnb5.score(Xnb5_test, ynb5_test))
plt.title(all_sample_title, size = 15)

```

Relatório de classificação

precision	recall	f1-score	support
0.10	0.20	0.13	5
0.00	0.00	0.00	5
0.90	0.87	0.88	82
		0.78	92
0.33	0.36	0.34	92
0.81	0.78	0.79	92

Text(0.5, 1, 'Acurácia: 0.782608695652174'))



O aviso não foi exibido, mas o comportamento em relação a classe '1' permanece.

```
#CROSS VALIDATION NB

#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb55 = cross_val_score(gnb5, Xnb, ynb, cv=5, scoring='accuracy')

#IMPRIMINDO
scores_gnb55

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb55.mean()

0.10330330330330331
```

```
#INSERINDO OS PARÂMETROS NO MODELO
scores_gnb510 = cross_val_score(gnb5, Xnb, ynb, cv=10, scoring='accuracy')

#IMPRIMINDO
scores_gnb510

#MÉDIA DOS RESULTADOS DOS SCORES
scores_gnb510.mean()

0.11403508771929824
```

Com a aplicação do cross validation, os resultados do score mean, para 70% ou 50% de amostragem de treino não apresentaram alterações significativas.

A performance da árvore de decisão se mostrou mais atrante em relação a bayesiana em praticamente todas as métricas utilizadas, , então mais uma abordagem tentativa será testada, dessa vez aumentando para 30% a base de treino da árvore de decisão, sendo mantidos os mesmos controles.

```
#DIVIDINDO O CONJUNTO DE DADOS EM CONJUNTOS DE TREINAMENTO E TESTE, SENDO 30% DE REGISTROS SELECIONADOS ALEATORIAMENTE COM SEMPREMOSA

X3 = X.copy()
y3 = y.copy()

X3_train, X3_test, y3_train, y3_test = train_test_split(X3 , y3, test_size = 0.3, random_state = 26)
print("Dimensões dos conjuntos de treino e teste:")
print("Entrada de divisão de treinamento: ", X3_train.shape)
print("Testando entrada de divisão: ", X3_test.shape)

### ALGORITMO ÁRVORE DE DECISÃO

# INSTANCIANDO O CLASSIFICADOR:
dtree3=DecisionTreeClassifier(criterion = 'gini' , max_depth = 5, min_samples_split= 5, min_samples_leaf= 2)

# TREINANDO O MODELO DE ÁRVORE DE DECISÃO:
dtree3 = dtree3.fit(X3_train,y3_train)
print('Classificador de árvore de decisão criado')

#IMPORTÂNCIA DA COLUNA NA DEFINIÇÃO DO RESULTADO DA CLASSIFICAÇÃO
print ('Importância do atributo na definição do resultado da classificação')
print('')
for feature,importancia in zip(AB_HUM_TREE_COPIA.columns,dtree3.feature_importances_):
    print("{}:{}".format(feature, importancia))
```

```
# RESULTADO DA CLASSIFICAÇÃO
y3_pred = dtree3.predict(X3_test)
resultado = y3_pred
print('')
print('Resultado da predição')
print(y3_pred)
print('')
print("          Relatório de classificação ")
print('')
print(metrics.classification_report(y3_test,resultado))

# MATRIZ DE CONFUSÃO
cm3 = confusion_matrix(y3_test, y3_pred)
plt.figure(figsize=(5,5))
mconf3 = sns.heatmap(data=cm3,linewidths=.5, annot=True,square = True, cmap = 'Blues')
plt.ylabel('Rótulo real')
plt.xlabel('Rótulo previsto')
all_sample_title = 'Acurácia: {}'.format(dtree3.score(X3_test, y3_test))
plt.title(all_sample_title, size = 15)
#mconf.figure.savefig('matriz_confusao.png')
```

```

Dimensões dos conjuntos de treino e teste:
Entrada de divisão de treinamento: (128, 10)
Testando entrada de divisão: (56, 10)
Classificador de árvore de decisão criado
Importância do atributo na definição do resultado da classificação

VOL_MAX_SUPERF_M3:0.0
VOL_MAX_SUBTER_M3:0.0
VOLUME_MAX_MUN_M3:0.0
VOLUME_MUN_ABHUM_2020_M3:0.0
DIF_PROP_VOL_MAX_VOL_2020:0.12206968113621262
POP_ESTIMADA_2020:0.0
POP_EST_ANO_VOL_MAX:0.07421365415277577
L_HAB_DIA_ANO_VOL_MAX:0.2067380365684467
L_HAB_DIA_ANO_VOL_2020:0.5969786281425649
VOL_2020_MIN_OMS:0.0

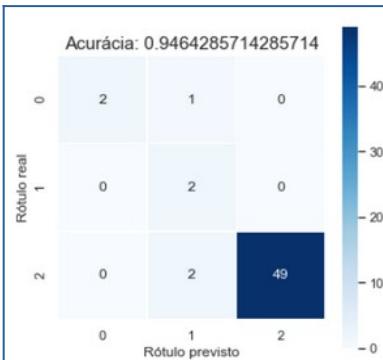
Resultado da predição
[2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0]

Relatório de classificação

      precision    recall   f1-score   support
0         1.00     0.67     0.80       3
1         0.40     1.00     0.57       2
2         1.00     0.96     0.98      51

accuracy                           0.95      56
macro avg       0.80     0.88     0.78      56
weighted avg    0.98     0.95     0.96      56

```



Esse modelo mostrou-se mais adequado a proposta, pois o recall de 100% na classe nível '1', evita que a canalização dos trabalhos ocorra onde é menos necessária a regularização do uso da água. Já a classe de nível '0', apresentou 67%, o que considerando os critérios que a definem, que não são exatamente ordenados em termos de valores, pode ser considerado um bom índice.

Uma outra tentativa foi testada, uma vez que a aplicação da metodologia fugiu a ideia inicial com 10 classes de nível de regularização. Então, para elocubrações, o procedimento foi refeito utilizando a árvore de decisão para elocubrações, com os 03 atributos que revelaram maior correlação com o ‘NIVEL’.

DIREITO_DE_USO_AB_HUM_MAX								
_VOL_MAX	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	VOL_2020_MIN_OMS	P01_LMAX	P03_2020	P02_L2020_LMAX	PT_AB_HUM	REG_AB_HUM
11155	32.252883	30.356156	475897.95	-2	-2	0	-4	9
11563	28.660094	0.000000	603695.40	-2	-3	0	-5	10
55900	99.019230	33.771552	2533625.60	-1	-2	0	-3	8
46350	77.952922	65.619207	2187412.15	-2	-2	0	-4	9
15812	137.373136	0.000000	702343.95	0	-3	0	-3	8
...
18375	452.864807	0.000000	884906.00	1	-3	0	-2	7
13636	61.325685	60.099171	558687.25	-2	-2	0	-4	9
17037	665.891378	0.000000	741610.65	-1	-3	0	-4	9
34798	78.625664	25.875177	1642255.45	-2	-2	0	-4	9
54399	7.862035	2.327634	2465611.50	-2	-2	0	-4	9

#CRIANDO UM DATAFRAME PARA USO NA ÁRVORE COM OS DADOS LEVANTADOS NA FINALIDADE ABASTECIMENTO HUMANO
DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10 = DIREITO_DE_USO_AB_HUM_MAX
DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 25 to 76147
Data columns (total 20 columns):
Column Non-Null Count Dtype
--- -----
0 Nome Município 184 non-null object
1 MUNICIPIO 184 non-null object
2 CD_IBGE 184 non-null int64
3 FINALIDADE_DE_USO 184 non-null category
4 ANOS_VIGENTES 184 non-null int64
5 VOL_SUPERF_M3 184 non-null float32
6 VOL_SUBTER_M3 184 non-null float64
7 VOLUME_MAX_MUN_M3 184 non-null float64
8 VOLUME_MUN_ABHUM2020_M3 184 non-null float64
9 DIF_PROP_VOL_MAX_VOL_2020 184 non-null float64
10 POP_ESTIMADA_2020 184 non-null int64
11 POP_EST_ANO_VOL_MAX 184 non-null int32
12 L_HAB_DIA_ANO_VOL_MAX 184 non-null float64
13 L_HAB_DIA_ANO_VOL_2020 184 non-null float64
14 VOL_2020_MIN_OMS 184 non-null float64
15 P01_LMAX 184 non-null int64
16 P03_2020 184 non-null int64
17 P02_L2020_LMAX 184 non-null int64
18 PT_AB_HUM 184 non-null int64
19 REG_AB_HUM 184 non-null int64
dtypes: category(1), float32(1), float64(7), int32(1), int64(8), object(2)
memory usage: 27.8+ KB

```
#RENOMEAR COLUNAS
DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10 = DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10.rename(columns={'REG_AB_HUM': 'NIVEL',})

#SEPARANDO AS COLUNAS QUE NÃO HÁ INTERESSE

conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10=set(DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10.columns.values.tolist())#Criando uma lista com todas as colunas
conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10_mantidas = {'DIF_PROP_VOL_MAX_VOL_2020','L_HAB_DIA_ANO_VOL_2020','L_HAB_DIA_ANO_VOL_MAX'}
conj_colunas_remover = conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10 - conj_colunas_DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10_mantidas

#REMOVER COLUNAS SELECIONADAS
DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10 = DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10.drop(columns=conj_colunas_remover)# Removendo as colunas que não interessam
```

```
DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 25 to 76147
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   DIF_PROP_VOL_MAX_VOL_2020  184 non-null   float64
 1   L_HAB_DIA_ANO_VOL_MAX    184 non-null   float64
 2   L_HAB_DIA_ANO_VOL_2020   184 non-null   float64
 3   NIVEL                  184 non-null   int64  
dtypes: float64(3), int64(1)
memory usage: 7.2 KB
```

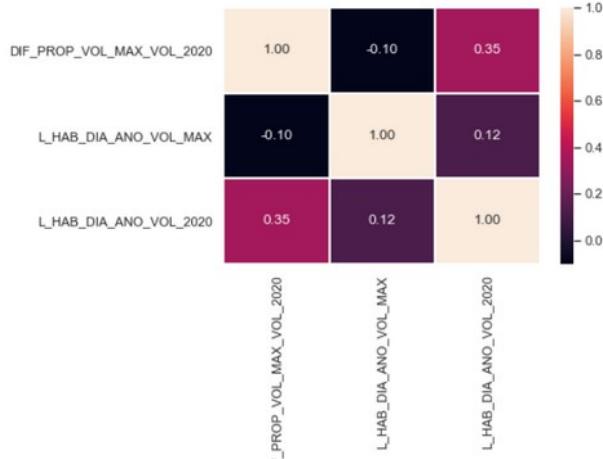
```
DIREITO_DE_USO_AB_HUM_MODEL_TREE_N10
```

		DIF_PROP_VOL_MAX_VOL_2020	L_HAB_DIA_ANO_VOL_MAX	L_HAB_DIA_ANO_VOL_2020	NIVEL
25		0.000000	32.252884	30.356155	9
421		-1.000000	28.660093	0.000000	10
851		-0.614987	99.019234	33.771553	8
1263		-0.010559	77.952919	65.619209	9
1685		-1.000000	137.373138	0.000000	8
...
74477		-1.000000	452.864807	0.000000	7
74909		0.000000	61.325684	60.099171	9
75319		-1.000000	665.891357	0.000000	9
75727		-0.613179	78.625664	25.875177	9
76147		-0.665787	7.862035	2.327634	9

184 rows × 4 columns

```
# MATRIZ DE CORRELAÇÕES
correlation = AB_HUM_TREE_N10.corr() # análise de correlação
plot = sns.heatmap(correlation, annot = True, fmt=".2f", linewidths=1) # plot da matriz de correlação
plot
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e0e029b808>
```



```
#SEPARANDO VARIÁVEIS
target = AB_HUM_TREE_N10['NIVEL']
AB_HUM_TREE_N101 = AB_HUM_TREE_N10.copy()
AB_HUM_TREE_N101 = AB_HUM_TREE_N101.drop('NIVEL', axis =1)
```

```
X = AB_HUM_TREE_N101 # Defining the attributes
```

```
# CODIFICANDO A VARIÁVEL
le = LabelEncoder()
target = le.fit_transform(target)
target
y = target
```

```
target
```

```
array([7, 8, 6, 7, 6, 8, 7, 3, 8, 5, 5, 7, 5, 5, 7, 7, 6, 7, 7, 8, 7,
       0, 8, 4, 7, 5, 4, 4, 7, 7, 1, 5, 4, 7, 3, 4, 7, 6, 6, 7, 5, 5,
       7, 8, 1, 8, 5, 4, 4, 6, 1, 8, 5, 6, 4, 5, 7, 4, 3, 3, 7, 8, 4, 3,
       4, 6, 7, 7, 7, 6, 8, 8, 7, 5, 4, 7, 6, 2, 7, 7, 3, 8, 7, 8, 8, 4,
       7, 5, 7, 5, 1, 6, 4, 5, 5, 1, 5, 7, 2, 7, 7, 8, 7, 5, 6, 6, 3, 5,
       8, 5, 3, 5, 7, 7, 6, 1, 3, 8, 4, 7, 6, 4, 7, 4, 3, 7, 1, 6, 7, 3,
       5, 4, 4, 7, 5, 6, 2, 7, 8, 7, 4, 6, 8, 5, 5, 7, 5, 7, 7, 4, 5, 7,
       4, 7, 6, 5, 5, 8, 6, 1, 5, 4, 5, 7, 2, 5, 2, 7, 7, 6, 4, 7, 7,
       8, 1, 2, 5, 7, 7, 7, 7])
```

```
#DIVIDINDO O CONJUNTO DE DADOS EM CONJUNTOS DE TREINAMENTO E TESTE.
#SELECIONAR 20% DE REGISTROS ALEATORIAMENTE PARA TESTE
```

```
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2, random_state = 42)
print("Entrada de divisão de treinamento: ", X_train.shape)
print("Testando entrada de divisão: ", X_test.shape)
```

```
Entrada de divisão de treinamento: (147, 3)
Testando entrada de divisão: (37, 3)
```

```
#DEFININDO O ALGORITMO DA ÁRVORE DE DECISÃO
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Classificador de árvore de decisão criado')

Classificador de árvore de decisão criado

# INSTANCIANDO O CLASSIFICADOR:
clf = DecisionTreeClassifier()

# TREINANDO O MODELO DE ARVORE DE DECISÃO
clf = clf.fit(X_train,y_train)

# VERIFICANDO AS FEATURES MAIS IMPORTANTES PARA O MODELO TREINADO, ONDE SERÁ RETORNADO UM ARRAY COM O VALOR DE CADA VARIÁVEL
clf.feature_importances_

array([0.20107315, 0.44026845, 0.3586584 ])
```

```
#IMPORTÂNCIA DA COLUNA NA DEFINIÇÃO DO RESULTADO
for feature,importancia in zip(AB_HUM_TREE_N10.columns,clf.feature_importances_):
    print("{}:{}{}".format(feature, importancia))

DIF_PROP_VOL_MAX_VOL_2020:0.12144951690406235
L_HAB_DIA_ANO_VOL_MAX:0.41774685595920336
L_HAB_DIA_ANO_VOL_2020:0.4608036271367343
```

```
# MATRIZ DE CONFUSÃO
y_pred = dtree.predict(X_test)
print("Relatório de classificação - \n", classification_report(y_test,y_pred))

Relatório de classificação - 
          precision    recall   f1-score   support
           1       1.00     1.00    1.00      2
           2       0.00     0.00    0.00      0
           3       1.00     0.67    0.80      3
           4       0.50     0.67    0.57      3
           5       0.89     0.80    0.84     10
           6       0.75     1.00    0.86      3
           7       1.00     0.79    0.88     14
           8       0.50     1.00    0.67      2

      accuracy                           0.81      37
     macro avg       0.70     0.74    0.70      37
weighted avg       0.88     0.81    0.83      37
```

C:\Users\55859\anaconda3\lib\site-packages\sklearn\metrics_classification.py:1272: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
`_warn_prf(average, modifier, msg_start, len(result))`

Este aviso está dizendo que a saída de `classification_report` é influenciada porque um dos rótulos nunca é previsto para seu modelo, no caso, o rótulo "2".

6. Apresentação dos Resultados

A água é um recurso limitado, sua universalização exige esforços no intuito de fazer com ela chegue onde o seu uso seja sendo necessário. Desde os tempos mais remotos, diversas tecnologias foram aplicadas para vencer esse desafio. No Brasil, não é diferente. Há um certo desequilíbrio hídrico entre as regiões, e nem sempre onde há uma aparente fartura de oferta hídrica a água tem condições de ser potabilizada através tratamentos convencionais. Em situações onde a água local não é suficiente, seja por fatores quantitativos ou qualitativos, ela será transportada de um local para outro, para que a população tenha segurança hídrica.

Gerenciar o uso da água é crucial para a saúde e para praticamente toda a cadeia produtiva. Para tal alguns instrumentos foram elaborados e, em conjunto, contribuem para a regulamentação da água pela sociedade.

Esses instrumentos precisam atingir o conjunto universo desses usuários de água, esse é o desafio, mas qual é esse universo? Já foi atingido?

Analizando os dados tentamos de obter algumas respostas.

Para tal, foi selecionado um estado brasileiro localizado no semiárido, região com características climáticas que se destacam pela escassez de oferta hídrica e densidade demográfica expressiva maior que regiões de maior abundância.

Um mergulho na base de dados de uso regulamentado foi o ponto de partida para essa viagem. Para olhar esses dados e enxergar o que eles dizem, a inspiração vem dos escultores, que extraem seres que vivem dentro das pedras de mármore. Eles os trazem à tona para que todos possam vê-los, como fizeram Giovanni Strazza com a Virgem Maria sob um véu e Michelangelo com Davi.



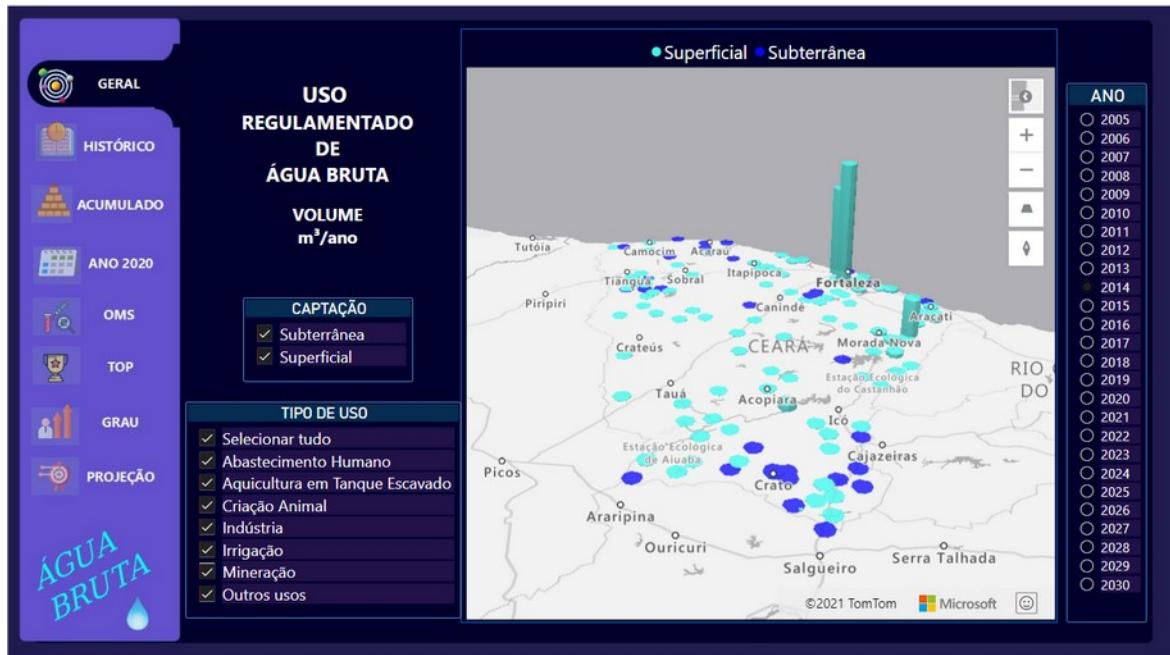
O dataset principal traz dados individuais de cada ponto de captação, é bem detalhado, como são sobre as outorgas de uso de água, têm coordenadas geográficas, volumes, período de vigência, dados de pessoas físicas e jurídicas, etc.

Um dashboard foi elaborado voltado para o uso da água no abastecimento humano, mas com uma página que abriga as demais finalidades de uso de forma universal.

Esses registros foram preparados para que possibilitasse realizar consultas com filtros por ano de vigência, finalidade de uso e forma de captação, sustentados por um anteprojeto geográfico, conforme combinação desejada. É possível, nesse visual observar o comportamento nos 184 municípios simultaneamente, selecionando um dos anos 25 anos.

A janela temporal engloba todos os anos onde já houve, onde há e onde haverá uso regulamentado de água bruta, lembrando que esses registros disponibilizados no *dataset* não contemplam as inclusões de 2021. O visual está apto a receber dados mais recentes.

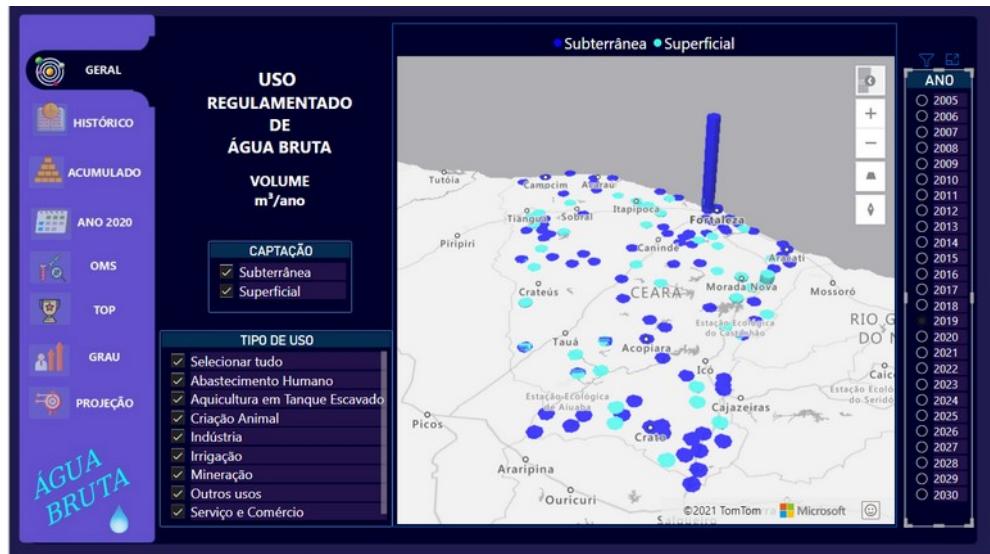
Os cilindros empilhados contêm os volumes, cujo uso de água, ocorreu, ocorre ou ocorrerá de forma regulamentada.



Imaginando que encontramos uma caixa de recordações, nessa ‘foto’ tirada em 2014 e aqui revelada, é notória a concentração de uso regulamentado na RMF e a prevalência da captação superficial de água.

Nesse ano, as notícias da época diziam que o nível dos açudes do estado estavam baixando, mas as esperanças de que a próxima quadra chuvosa reverteria a situação mantinha o otimismo em relação aos aportes hídricos que os açudes em breve receberiam.

Em 2015 foi decretado estado de escassez hídrica no estado, mas o otimismo ainda alimentava a inércia da população em geral, mas não dos especialistas.

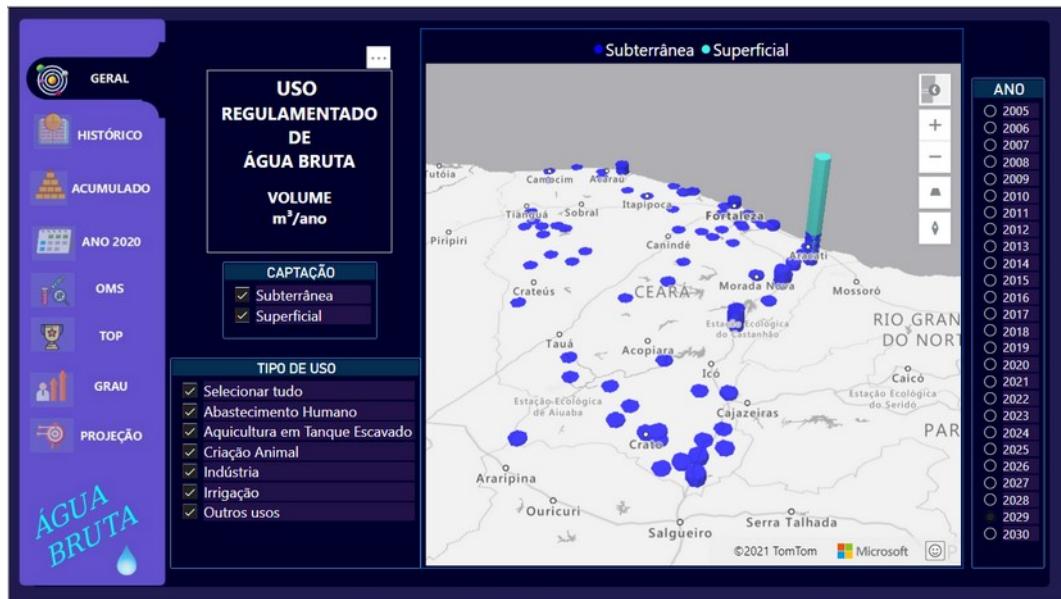


A foto' acima é de 2019, é perceptível que o maior volume de captações regulamentadas é de água subterrânea e também da RMF, mas nas demais regiões também mudou o cenário.

Os recortes de jornal disseram que nesse intervalos, foi facilitada a regularização das perfurações de poços, a regulamentação de uso de águas subterrâneas, dispensadas taxas, e até dentro de açudes secos foi explotada a água subterrânea, além disso, a grande maioria dos açudes monitorados ficou exclusivamente para captação com finalidade de abastecimento humano.

É na região sul do estado que o balanço hídrico de águas subterrâneas tem mais conflitos, pois os aquíferos também sofrem com a estiagem. Por conta disso, a procura por regulamentação de uso nessa região também cresceu. A imagem acima confirma essa notícia que correu de boca em boca.

O decreto de estado de escassez hídrica de 2015 ainda está vigente, ao todo já são quase 10 anos de estiagem prolongada, mas a população não precisou deixar seu município com ocorrência antigamente. O uso regulamentado favorece o planejamento e execução de medidas mitigatórias de segurança hídrica.



Essa ‘foto’ encontrada na caixa de fotos e mostrou que o uso regulamentado já chegou em 2019 e está lá esperando por seus usuários.

Ao passar o cursor no cilindro do município, é informado o volume anual disponibilizado subterrâneo e superficial., além das coordenadas básicas do município.



Uma questão sobre o uso da água estimulou a preparação dos dados para que um visual mostrasse por finalidade de uso, quanto já foi outorgado em cada município e com essa essa distribuição em cada município

O visual foi desenvolvido para trazer em suas fatias a proporção de cada uso em cada município, assim como os respectivos volumes anuais. Ao selecionar uma das finalidades, ela é destacada, mas as demais fatias continuam. Caso nenhuma seja selecionada todas aparecem no mesmo nível de destaque.

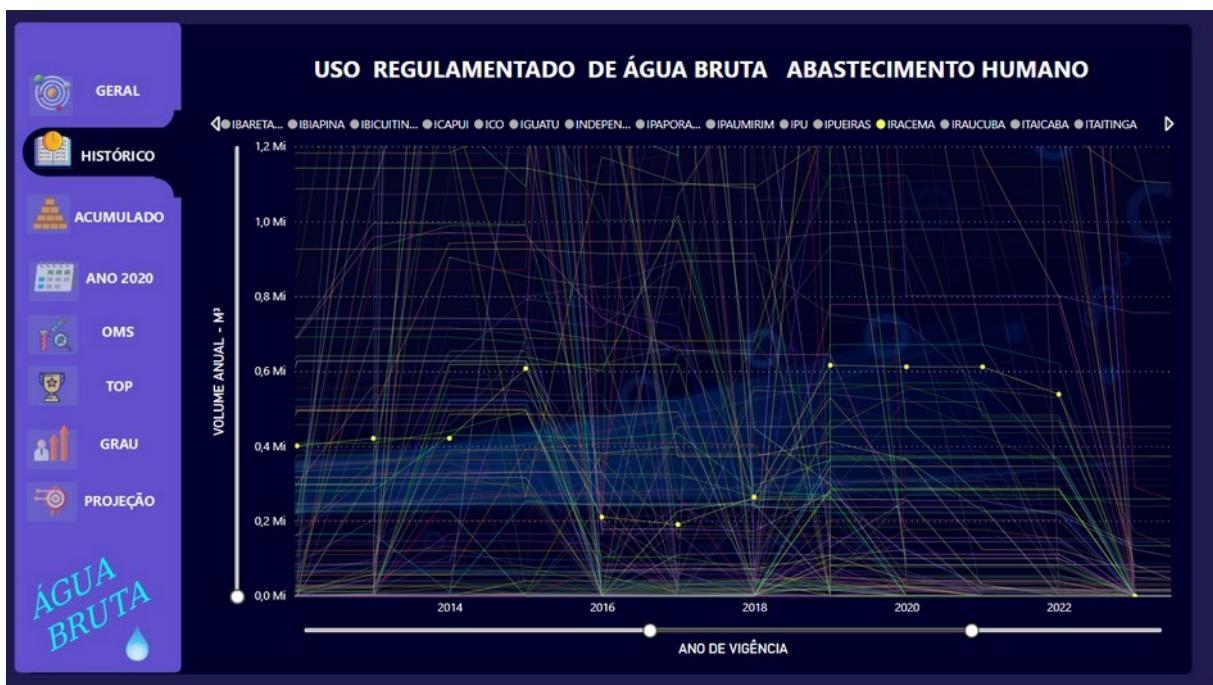
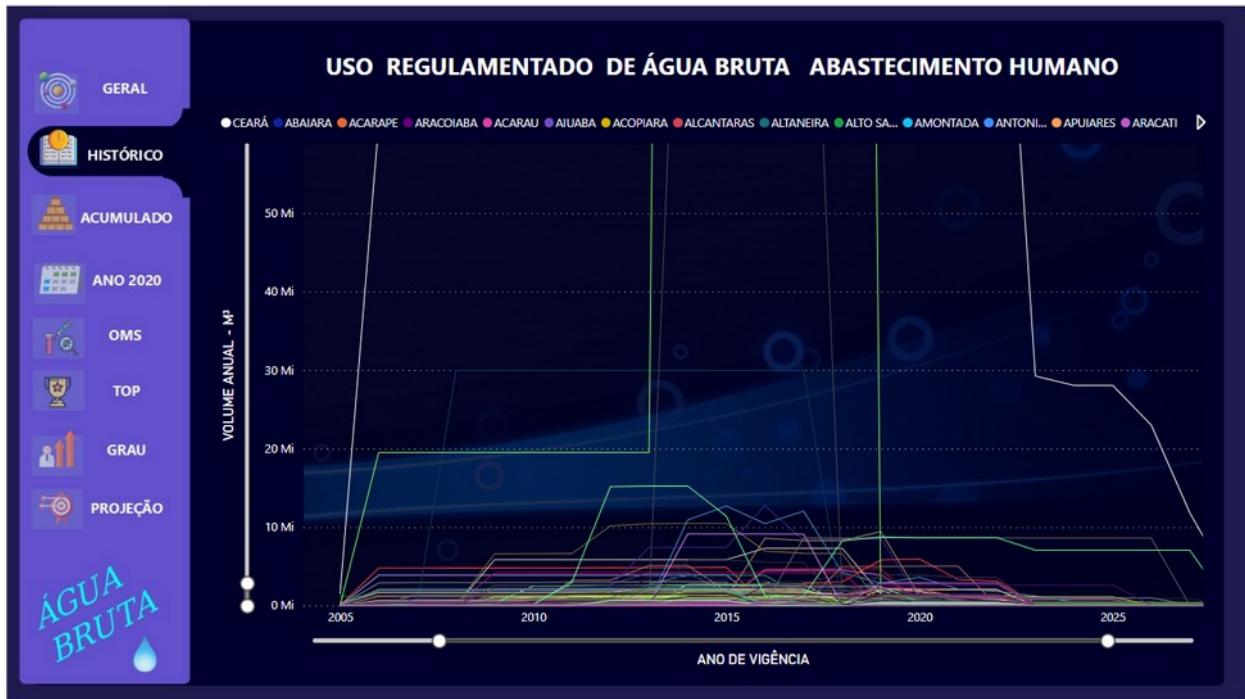


Nesse visual, sem destaque por filtro, a predominância da cor azul evidencia que o abastecimento humano foi o que mais esteve presente nos municípios e é que tem o maior volume regulamentado acumulado.

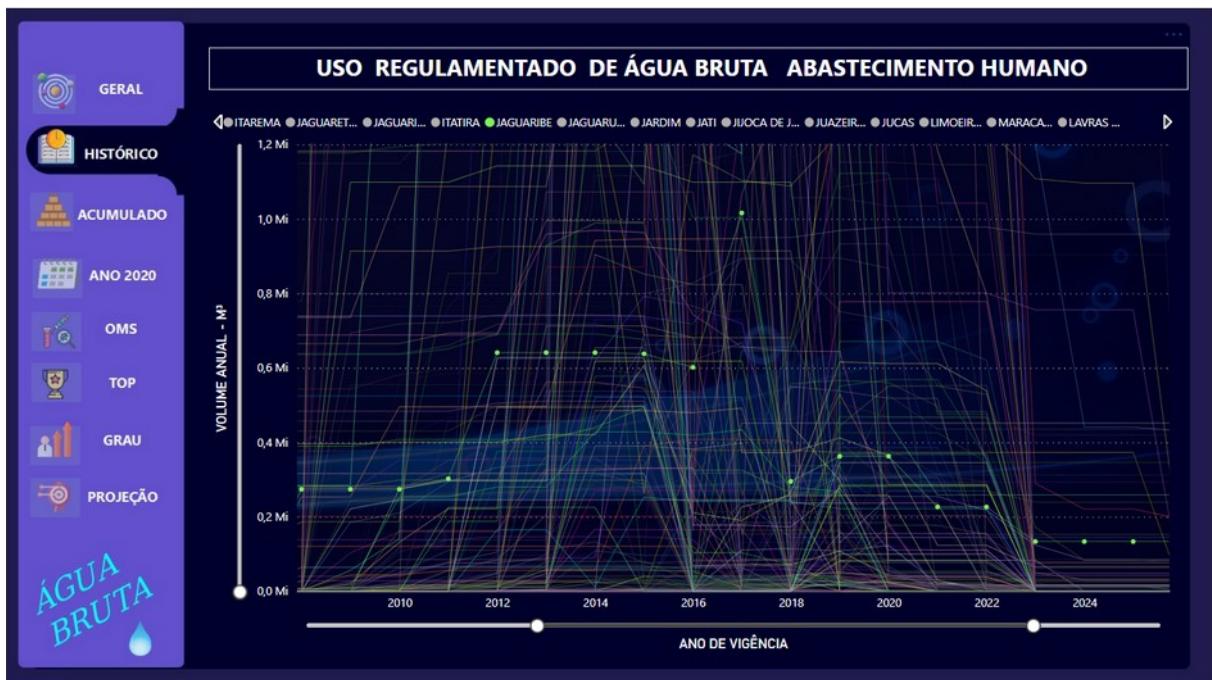
Na RMF o município de Caucaia apresenta um comportamento atípico em relação ao tendênciam dos municípios vizinhos. É provável que suas captações para abastecimento humano aconteçam em um desses municípios, pois é bastante populoso.

Essa 'linha' verde da irrigação coincide com o Rio Jaguaribe, num trecho a jusante do Açude Castanhão,

Os dados foram preparados para que um visual com gráfico de linha fosse elaborado, tendo em seus eixos os anos de 2005 a 2030 e o volume anual regulamentado. As linhas contêm a trajetória de cada município e também do estado. Ao selecionar qualquer um dos 184 municípios ou estado, as demais linhas permanecem no visual, mas com menos destaque. Também é possível controlar as faixas de valores de interesse em ambos os eixos. Esse visual é exclusivo de abastecimento humano.



Como exemplo, o município de Iracema apresentou, em seu uso regulamentado no período de 2015 a 2019, uma queda. Nessa longa estiagem, algumas captações tradicionalmente realizadas tiveram que ser realizadas em outros pontos e tipo de manancial. Como a curva diz respeito a abastecimento humano, não parece plausível que a atividade tenha sido suspensa, sendo pertinente supor que um outro local tenha sido usado, provavelmente através de uma AMR.



Observando o município de Jaguaribe, que além de próximo fica na margem do Rio Jaguaribe e a jusante do Açude Castanhão, o maior do estado, percebe-se um pico entre os anos 2016 e 2018. As outorgas concedidas para abastecimento público costumam ser mais duradouras, esse pico sugere algo mais temporário, como um AMR ou inversão de fluxo em adutoras convencionais ou mesmo em canais abertos, quando possível.

As curvas dizem que Iracema desceu de 0,6 para 0,2 (-0,4) e Jaguaribe subiu de 0,6 para 1,0 (+0,4).

Há que se aprofundar mais na investigação para obter confirmação, mas esse movimento das curvas sugere esse município, o que pode restringir essa busca.

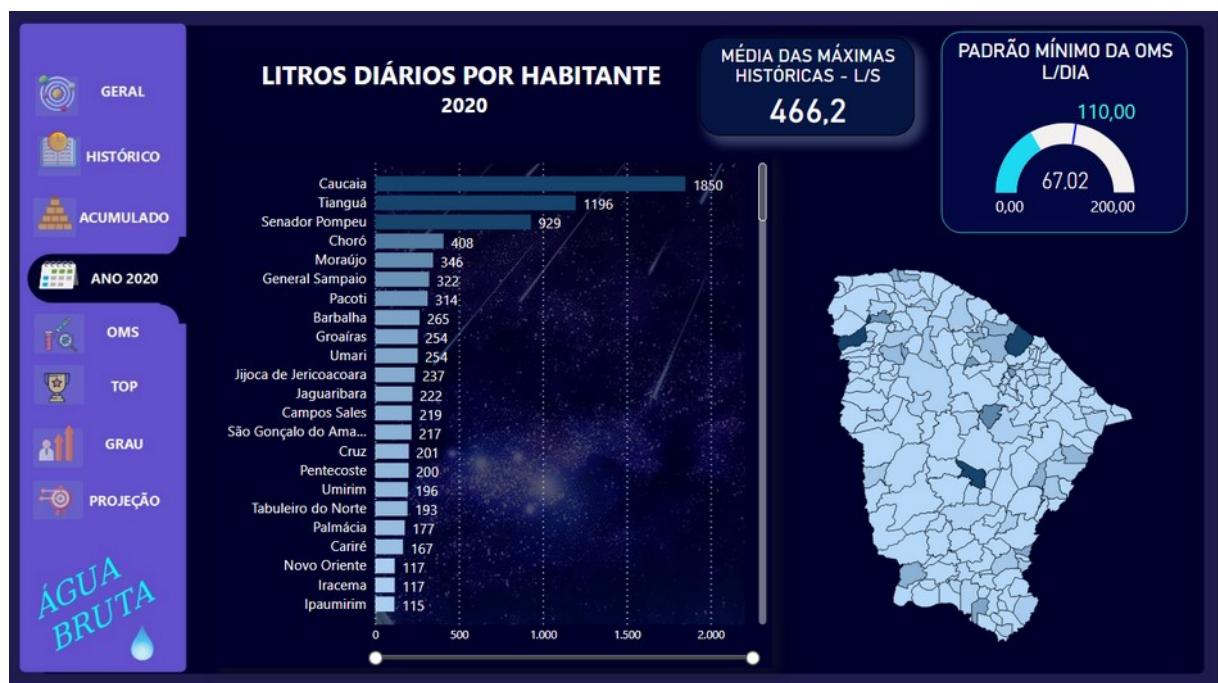
Buscando um indicador, durante a exploração de dados, uma das referências foi o consumo mínimo recomendado pela ONU através de sua agência da saúde a OMS. Segundo a OMS é o mínimo para 110 litros diários por habitante para seu consumo, higiene pessoal e de sua casa.

Os dados foram tratados para que nesse visual possa se ter uma ‘foto’ de 2020, do volume regulamentado vigente em 2020 de cada município em relação a esse padrão mínimo e, como está a distribuição espacial desse parâmetro, pois, as captações não necessariamente ocorrem no município de uso da água.

Para efeito de comparação também é exibida a média dos máximos históricos dos litros diários por habitante quando nenhum município é selecionado, caso contrário traz o máximo valor para o município selecionado.

Também é possível, como controle no eixo x do gráfico de barras, ter outras aproximações manipulando a escala.

O velocímetro tem como valor máximo os 200,00 l/hab.dia do padrão Brasil e como destino os 110,00 l/hab.dia da OMS. Também exibe a média do estado quando nenhum município é selecionado.



Anteriormente o município de Caucaia em 2020 apareceu com volumes vultuosos. Nessa ‘foto’ de 2020, a quantidade de litros diários por habitante é mesmo estratosférica sendo 16 vezes maior que a mínima recomendada pela OMS 9,25 vezes o padrão do Brasil. Já para Fortaleza, município vizinho e capital do estado, consta volume zero, apesar das populações serem semelhantes. Sem essa visão geográfica a compreensão da situação seria prejudicada.

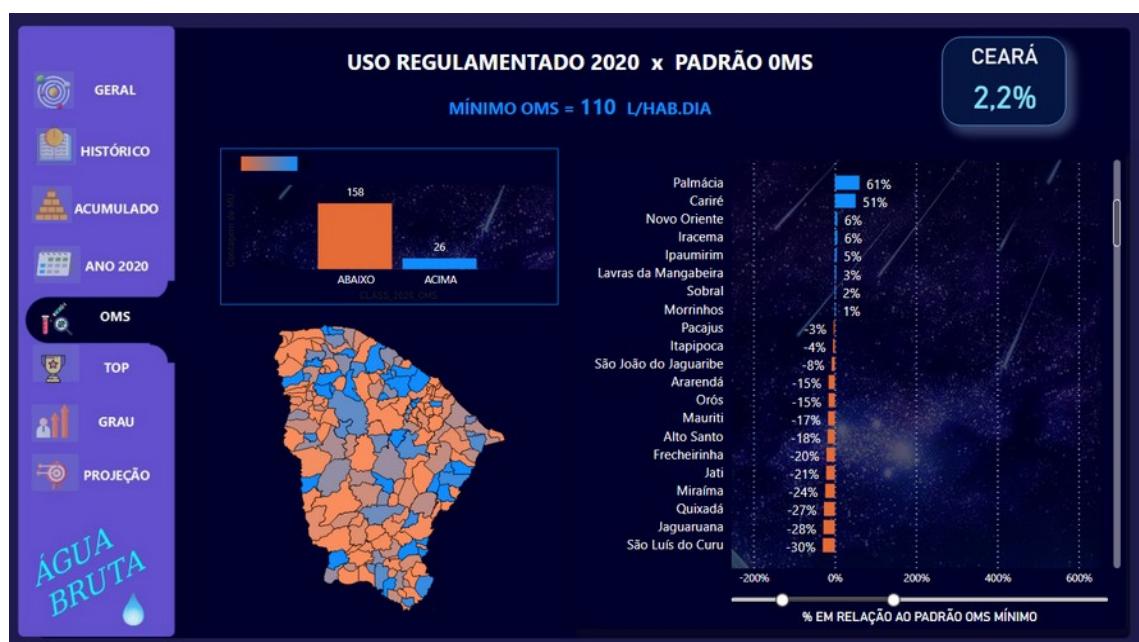


O município de Jijoca de Jericoacoara, quanto aos padrões, está acima de ambos, mas os valores estão calculados em cima da população estimada para 2020 (20.087 habitantes), e essa estimativa populacional é coerente, entretanto o município é relevante para o turismo, a população de visitantes supera a dos residentes locais com frequência, então esse conforto hídrico regulamentado pode não ser tão confortável. Por outro lado, o uso regulamentado na finalidade serviço e comércio, que abriga a atividade hoteleira apresentou volume nulo para 2020. Como o carro-chefe do turismo no município são as atividades típica de verão, os hotéis são equipados com piscinas, como pode ser contatado na página 'Geral' do *dashboard* onde o visual permite escolher uma imagem de satélite como pano de fundo, além de mais algumas outras opções.

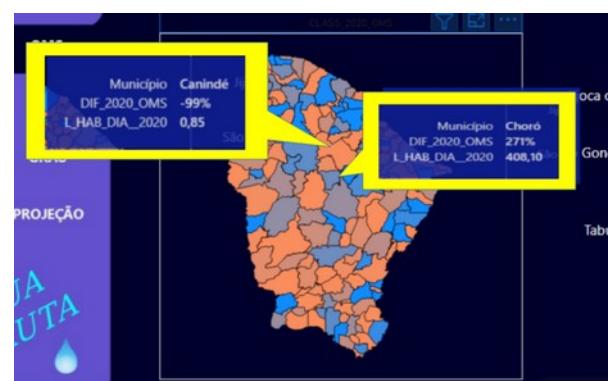


Na exploração de dados, foi desenvolvido um indicador, um desses critérios é observada a situação do uso regulamentado de água na finalidade abastecimento humano.

A página ‘OMS’ faz um comparativo e retorna com ajuda de uma escala de cores a distância que cada município está desse padrão mínimo. Já no quantitativo de municípios acima e abaixo e nos percentuais em relação ao mesmo do gráfico de barras, a cor vermelha está associada a valores abaixo do padrão mínimo e a azul acima desse padrão, sem nuances.. A escala no eixo digital do gráfico de barras pode ser manipulada através do controle no gráfico. Também traz a diferença do estado em relação a esse padrão.



A quantidade de municípios abaixo do padrão é esmagadora em comparação com os que estão acima. A distribuição espacial coloca os municípios mais afastados do litoral e simultaneamente das serras com as situações mais preocupantes, no entanto o estado está 2,2% acima do padrão mínimo.



Também percebe-se que municípios vizinhos apresentam diferenças extremas como, por exemplo, como Choró e Canindé

Reunindo alguns 03 parâmetros os municípios foram classificados conforme o somatório em cada um deles.

A ideia consistem em comparações da situação de 2020 com o Padrão OMS, o Padrão Brasil e o maior uso anual regularizado pelo município na finalidade abastecimento humano.

Se a quantidade em 2020 é menor que esse valor máximo já regularizado, houve um declínio, tendo em vista que as populações tendem a crescer a cada ano. Nesse sentido, uma taxa de 2% a.a. foi utilizada para estimar as populações dos anos onde foi observado máximo volume registrado, partindo da estimativa de 2020 e por conseguinte estimado os litros diários com uso regulamentado por habitante.

Vale ressaltar que trata-se do uso regulamentado de ativo em cada ano. Esse ‘ano’ de ocorrência de ‘máximo’ é no somatório das outorgas ativas em determinado ano e município. Seu seu teto [é em 2020, ou seja, não há ocorrência posterior. Isso foi confirmado na etapa de exploração de dados e é pertinente, pois em caso de empate, o volume máximo foi considerado o do ano mais antigo.

Além disso, na efetivação das regulamentações, o estado a partir de 2020 passou a conceder a regulamentação do uso com vigência mínima de 10 anos, as anteriores, salvo algumas exceções eram de 4 anos. Então, os valores dos volumes com início de vigência em 2020 se estendem até 2030, mas o ano de 2020 por ser o mais antigo desse grupo, é nomeado ano de ocorrência do volume máximo.

Feitas essas considerações, o uso diário por habitante no caso acima, como o volume seria, o mesmo ao final do período e por outro lado a população teria aumentado conforme a taxa de crescimento adotada, o ano de maior conforto hídrico seria o primeiro, daquele desempate sobre o ano de máximo uso por habitante dia.

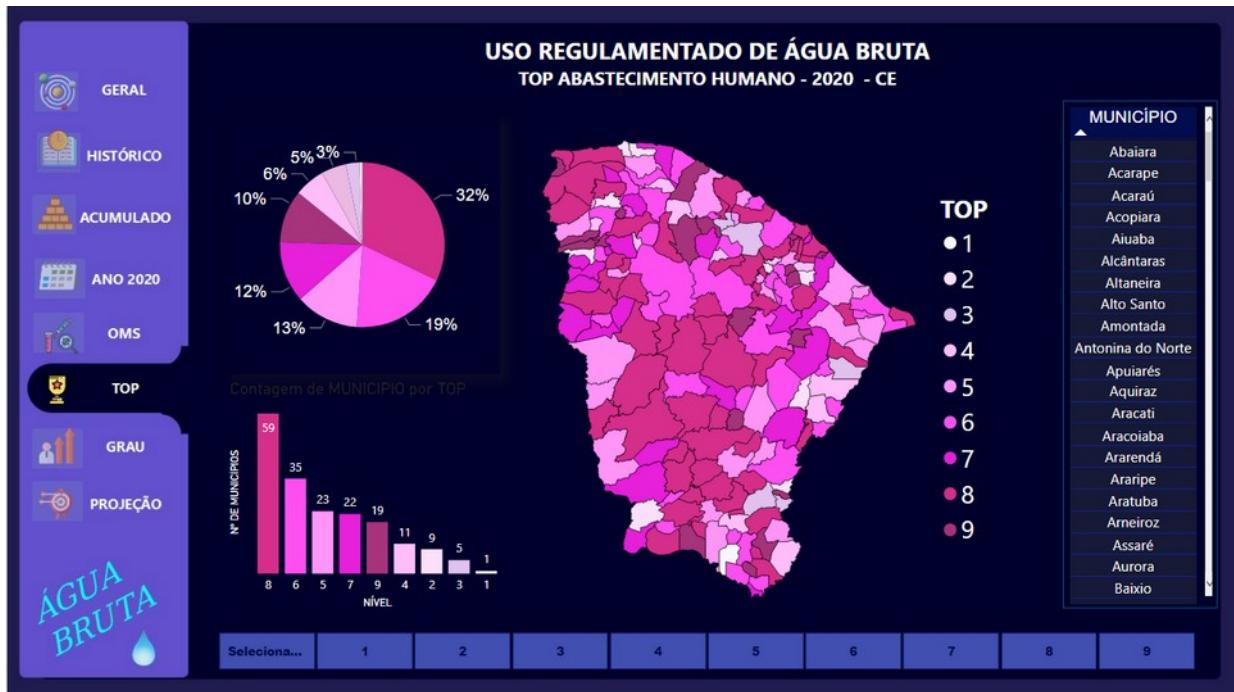
Se na comparação de L2020 com o Lmax, se L2020 for menor, perde ponto ponto, se for igual, ganha ponto, e não tem como ser menor, pois também compete na mesma categoria.

A próxima comparação, é em relação aos padrões OMS e Brasil, onde foram estabelecidas faixas abaixo 110, acima de 200, entre 100 e 200, entre 80 e 110, entre 200 e 600, abaixo de 80, acima de 600 e zero.

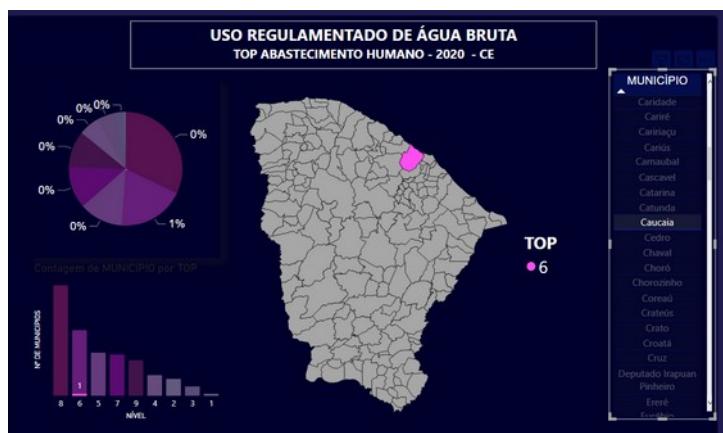
Foram comparados com esses limites, os valores Lmax e L2020, sendo que ao cair dentro dos mesmos limites, de 110 a 600, os L2020 recebeu, vamos dizer uma estrelinha extra, na verdade foi um ponto.

Uma das ideias era conhecer quais são os municípios TOP 1, TOP 2 e assim sucessivamente, dentro desse quesito.

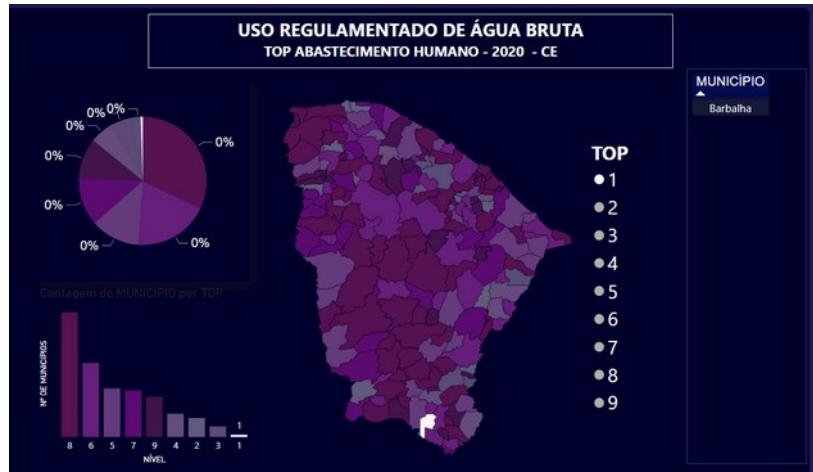
O visual a seguir foi elaborado baseado nessa ideia. Os mais bem colocados estão mais clarinhos e os que precisam de mais esforço, para compreender o motivo de não estarem entre os primeiros, estão mais escurinho.



Esse agrupamento categórico, por si só não traduz a situação, é preciso olhar a vizinhança e mesmo além dela, pois no estado água é conduzida por longas distâncias, algumas vezes atravessa outros municípios para atender um mais distante.



O município de Caucaia, apareceu no Top 06, mesmo tendo chamado a atenção nos mapas anteriores, certamente pelo fato de valores “L” acima causam perda de pontos, pois sugere desperdício, o que pode ser contestado se seus vizinhos não tiverem obtido pontuação igual ou maior, pois pode ser apenas um indício de que o município tem sido provedor de água para outros. Nesse mesmo grupo há 36 municípios.



E quais são os municípios do TOP 01 ? Apenas Barbalha, um município do sul do estado onde há muitas nascentes e captações de água subterrâneas (poços).

Ser TOP 01 significa autonomia, ou seja, não está tendo necessidade de ajuda de outros municípios, água do município fica no município, o que facilita também o gerenciamento. A população, de posse dessa informação pode tomar medidas para manter essa autonomia, preservando as nascentes e protegendo os aquíferos.

Durante o desenvolvimento do algoritmo de aprendizado de máquina, ficou claro que uma classificação em 09 categorias, com 03 critérios para apenas 184 registros, não era uma boa opção.

Uma reflexão a respeito, trouxe um novo entendimento sobre o uso regulamentado de água bruta. Um município que ficou entre os TOP's 01, 02 e 03, indica maior engajamento da população em buscar a regulamentação, seja individualmente ou através de seus gestores, além dos fatores descritos sobre autonomia. Isso nada mais representa do que a representação do grau de maturidade desse município nos aspectos elencados.

As categorias foram reduzidas a 03, conforme o grau de maturidade no uso regulamentado de água bruta, e os resultados preliminarmente desanimadores foram melhorando.

A essa altura, os dados já estavam rotulados, pois as amostras foram classificadas. O aprendizado seria supervisionado.

A princípio os testes foram executados alterando um parâmetro e mantendo os demais para observar o que ocorria. Nas primeiras tentativas ocorreram com algoritmo de classificação por árvore de decisão, com 20 % de amostras reservadas para teste os resultados foram melhores que os da tentativa com Naive Bayes.

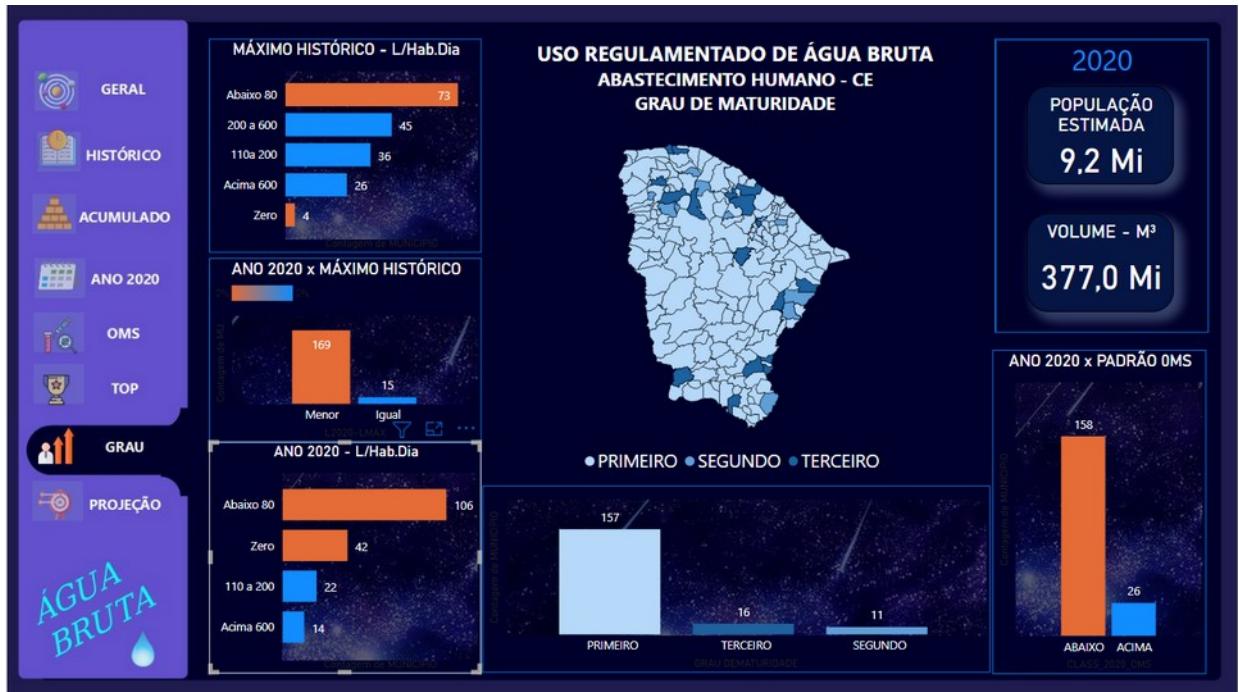
Com a redução para 03 categorias, a base se mostrou desbalanceada. Técnicas de desbalanceamento foram aplicadas para equilibrar a representatividade das amostras, mas na validação os resultados não foram muito exitosos.

A validação cruzada foi aplicada, e alguns resultados apresentaram comportamentos muito aleatórios, as tentativas foram executadas com 5 e 10 *folds*.

Uma tentativa com algoritmo de árvore de decisão foi realizada alterando as amostras de teste para 30%. Desta vez o índice *recall*, que nesse contexto informa onde a necessidade de regulamentação é menor, obteve valor compatível com o objetivo que era buscado, ou seja, evitar o direcionamento dos esforços onde a regulamentação está num grau de maturidade mais avançado.

Os dados foram tratados e dispostos para compor o visual abaixo. Os critérios de comparação adotados estão reunidos num mesmo visual permitindo observar a composição do grau de maturidade de cada município.

O visual também exibe a população estimada para 2020 e o volume anual com uso regulamentado para uso no abastecimento humano nesse mesmo ano.



Dos 184 municípios, 16 estão do terceiro grau, 11 no segundo e 157 no primeiro.

Os visuais anteriores exibem o que já foi realizado em termos de uso de água regulamentado, e quanto ao futuro? Que volume é necessário regulamentar anualmente, para que o estado atenda o padrão OMS?

Uma projeção de crescimento populacional ano a ano para os próximos 10 anos, foi realizada a uma taxa de 2% a.a. e aplicado o Padrão OMS. Chegou-se ao volume anual ideal desse foi subtraída a porção já regulamentada. O resultado obtido é volume que precisa ser regulamentado para atingir esse objetivo.



O gráfico acima indica que os valores volumétricos que precisam estar regulamentados na finalidade de uso abastecimento humano até o ano limite para que o estado não fique abaixo do padrão OMS.

Até o último dia de 2021, por exemplo, além do que foi regulamentado até 2020, são necessários 260,2 milhões de metros cúbicos.

Numa situação hipotética, se em 2021 o valor regulamentado for acima, esse excedente ajuda a reduzir os valores dos próximos 10 anos (devido a vigência mínima de 10 anos).

Numa situação oposta, se esse valor for abaixo, os valores dos anos subsequentes não são aumentados, pois desse volume exibido ano a ano já foi retirado o que foi regulamentado. O que aconteceria é que teria um ano abaixo do padrão.

O impacto de um ano com volumes abaixo do padrão repercutirá no nível de maturidade do estado, ou dos municípios onde essa baixa ocorreu.

5. Links

Link para o vídeo: https://youtu.be/Wnm-pl-2_0Y

Link para o repositório: <https://github.com/CARLAMORENCY/REPOSITORIO-AGUA-BRUTA>

APÊNDICE

Termologia temática	
Agua bruta	Água sem nenhum tratamento qualitativo.
Captação	Retirada de água do manancial
Direito de Uso	Outorga para fins de uso de água bruta
Espelho d'água	Área da superfície superior de um polígono de um corpo hídrico
Leito	Calhas dos rios, riachos, córregos (águas correntes)
Litros diários por habitante L/hab.dia	Consumo médio de um habitante por dia
Manancial	Rios, riachos, córregos, lagoas, açudes(represas),canais, adutoras, poços e fontes (nascentes)
Manejo	Forma de utilização da água quanto aos horários, tempo de captação e intervalos
Massa de água	Água acumulada
Métodos	Técnicas produtivas
Outorga	Autorização formal para uso de recursos hídricos ou Execução de obras/serviços de interferência hídrica
Uso consuntivo	Consumo com captação de água de água
Uso não consuntivo	Sem consumo expressivo de água, 'sem' captação de água (Ex.: peixe em gaiola)
Vazão (l/s)	Volume de um fluido por tempo, nesse caso volume de água em litros por segundo
Vazão máxima (l/s)	Máxima vazão instantânea de captação em litros por segundo
Vigência	Período de validade da outorga (anual)
Volume anual (m³/ano)	Volume de água bruta para uso anual em metro cúbicos
Volume máximo (m³/ano)	Máximo volume anual em determinado local (Ex.: município)

Sigla	Nome/Descrição
IBGE	Instituto Brasileiro de Geografia e Estatística
ANA	Agência Nacional de Águas
COGERH	Companhia de Gestão em Recursos Hídricos
CEP	Código de Endereçamento Postal
CORREIOS	Empresa Brasileira de Correios e Telégrafos (ECT)
REGLA	Sistema Federal de Regulação de Usos
SRH/CE	Secretaria de Recursos Hídricos do Ceará
CNARH	Cadastro Nacional de Recursos Hídricos da ANA
PROGESTÃO	Programa de Consolidação do Pacto Nacional pela Gestão das Águas
DDD	Discagem Direta à Distância
CNA	Cadastro Nacional de Atividades
CNAE	Classificação Nacional de Atividades Econômicas
CONAMA	Conselho Nacional do Meio Ambiente
OMS	Organização Mundial da Saúde
IPECE	Instituto de Pesquisa e Estratégia Econômica do Ceará
PIB	Produto Interno Bruto
IPECEDATA	Sistema de informações geossocioeconômicas do Ceará
CPF	Cadastro de Pessoa Física
CNPJ	Cadastro Nacional de Pessoa jurídica
ONU	Organização das Nações Unidas
RMF	Região metropolitana de Fortaleza
AMR	Adutora de montagem Rápida

Ferramenta	Versão
Windows 10	20H2
Anaconda-Jupyter Notebook	6.0.3
Libre Office – Calc	6.3.4.2 (x64)
Libre Office – Writer	6.3.4.2 (x64)
Power Bi Desktop	: 2.98.1004.0 64-bit (outubro de 2021)
Google Earth Pro	7.3.4.82248(64-bit)
Ferramenta de Captura	10.0.19042
Python	3.7.6