

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

Rafael Soares Cruz

ANÁLISE DO PERFIL OCUPACIONAL DAS FOLHAS DE PAGAMENTO
MUNICIPAIS COM CLUSTERS

Belo Horizonte
2021

Rafael Soares Cruz

**ANÁLISE DO PERFIL OCUPACIONAL DAS FOLHAS DE PAGAMENTO
MUNICIPAIS COM CLUSTERS**

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

1. Introdução.....	4
1.1. Contextualização.....	4
1.2. O problema proposto.....	4
2. Coleta de Dados	5
3. Processamento/Tratamento de Dados	12
4. Análise e Exploração dos Dados	18
5. Criação de Modelos de Machine Learning	19
6. Apresentação dos Resultados	33
7. Links	59
REFERÊNCIAS.....	60
APÊNDICE.....	61

1. Introdução

1.1. Contextualização

Busca-se com o presente estudo identificar similaridades e diferenças entre todos os Municípios brasileiros com base na quantidade e espécies de ocupações presentes em suas folhas de pagamentos. Para tanto, obteve de bases públicas (Ministério do Trabalho e Emprego) dados quantitativos dos CBOs (Classificação Brasileira de Ocupações) informados na RAIS (Relação Anual de Informações Sociais). Para que fosse possível analisar os resultados, a base de dados foi enriquecida com dados da localização dos Municípios (longitude e latitude) obtidos no site da Controladoria Geral da União (CGU).

Ademais, em tempos de orçamento público deficitário e de poucos recursos para manutenção da máquina estatal, faz-se necessário estudar e analisar como ocorrem os gastos públicos, principalmente o gasto com pessoal, que pode chegar a 60% (sessenta por cento) dos gastos da máquina estatal. Informações sobre as diferentes estruturas funcionais de cada Ente (quantidade e perfil de ocupações presentes em suas folhas de pagamento) podem compor uma combinação de índices que auxiliem na análise da estrutura e condições dos Municípios do País.

Um estudo completo só é possível com essa integração, na qual é imprescindível a presença de informações sobre a estrutura do funcionalismo. E estas não são simples análises quantitativas, ou de uma ocupação específica, já que precisam indicar, de alguma forma, Municípios com características semelhantes e/ou diferentes, considerando todas as ocupações existentes. Com o presente trabalho, busca-se verificar agrupamentos a partir de tais aspectos similares e divergentes identificados entre os Entes, através do uso de ferramentas de aprendizado não supervisionado.

1.2. O problema proposto

Objetiva-se com a presente análise a identificação de padrões ou perfis das folhas de pagamentos dos Entes municipais, tendo-se como referência os quantitativos e espécies de categorias ocupacionais (CBO) informados em sua

RAIS. Deseja-se identificar grupos de Municípios com folhas de pagamentos de alguma forma semelhantes utilizando-se como base de análise informações de todas as ocupações existentes na estrutura dos Municípios. Assim, estão sendo analisadas as informações mais recentes prestadas pelos Municípios e pelo Distrito Federal, da RAIS do ano-base de 2019.

A otimização dos gastos públicos é tarefa sempre necessária, mostrando-se fundamental em tempos de falta de recursos como o que vivemos. A maior parte do orçamento dos Municípios é gasto com pessoal, na manutenção das folhas de pagamentos, e uma análise desses gastos deve levar em consideração não apenas a quantidade de empregados públicos, mas também os diferentes perfis identificáveis na estrutura funcional dos Municípios, através da análise das ocupações.

Assim, analisou-se dados provenientes da Relação Anual de Informações Sociais (RAIS), informados pelas empresas e empregadores e disponibilizados publicamente pelo Ministério da Economia, através de sua Secretaria Especial de Previdência e Trabalho. A RAIS é importante instrumento de coleta de dados do mercado de trabalho, instituída pelo Decreto nº 76.900, de 23/12/75. Também são utilizados dados cartográficos (longitude e latitude das sedes municipais), a título de enriquecimento da base de dados e para análise visual dos resultados alcançados, obtidos por meio da CGU.

2. Coleta de Dados

A RAIS foi instituída pelo Decreto nº 76.900, de 23 de dezembro de 1975, devendo ser preenchida pelas empresas, contendo elementos destinados a suprir as necessidades de controle, estatística e informações das entidades governamentais da área social (art. 1º). A RAIS é enviada anualmente, sempre relativa ao ano-base anterior. Conforme a Apresentação constante do Manual de Orientação da Relação Anual de Informações Sociais: ano-base 2019:

A RAIS, com transcorrer do tempo, foi se tornando, no País, uma das fontes estatísticas mais confiáveis sobre o mercado de trabalho formal. Este registro administrativo constitui referência nacional e internacional sendo considerado um verdadeiro CENSO.

Suas informações têm por objetivo:

- a) o suprimento às necessidades de controle da atividade trabalhista no País;
- b) o provimento de dados para a elaboração de estatísticas do trabalho; e
- c) a disponibilização de informações do mercado de trabalho às entidades governamentais.

Os dados coletados pela RAIS constituem insumos para atendimento das necessidades:

- a) da legislação da nacionalização do trabalho;
- b) de controle dos registros do FGTS;
- c) dos Sistemas de Arrecadação e de Concessão e Benefícios Previdenciários;
- d) de estudos técnicos de natureza estatística e atuarial; e
- e) de identificação do trabalhador com direito ao abono salarial PIS/PASEP.

Para o ano-base de 2019, as informações do preenchimento constam da Portaria nº 6.136, de 3 de março de 2020, do Ministério da Economia/Secretaria Especial de Previdência e Trabalho. Quando do preenchimento da RAIS, o empregador, ou aquele legalmente responsável pela prestação das informações, deverá relacionar na RAIS de cada estabelecimento os vínculos laborais havidos ou em curso no ano-base, e não apenas os existentes em 31 de dezembro, abrangendo, entre outros, os servidores da administração pública direta ou indireta federal, estadual, do Distrito Federal ou municipal, bem como das fundações supervisionadas; e os servidores públicos não-efetivos, demissíveis *ad nutum* ou admitidos por meio de legislação especial, não regidos pela CLT (art. 3º, incisos IV e V). Por outro lado, não devem ser declarados ocupantes de cargos eletivos (governadores, deputados, prefeitos, vereadores, Conselheiro Tutelar etc.), a partir da data da posse, desde que não tenham feito opção pelos vencimentos do órgão de origem.

Devem ser prestadas na RAIS todas as informações de cada empregado/servidor, tais como dados pessoais, informações da admissão (data, salário/vencimento básico, quantidade de horas de trabalho semanais, código CBO), tipo de vínculo empregatício, informações de afastamentos/licenças, informações de desligamento, contribuições sindicais do empregado, remunerações mensais, verbas pagas na rescisão, entre outras.

Utilizou-se nessa análise os dados ocupacionais dos empregados/servidores dos Municípios, informados na RAIS através do uso de códigos CBO. A Classificação Brasileira de Ocupações, conforme informações do site do Ministério do Trabalho:

...é um documento que retrata a realidade das profissões do mercado de trabalho brasileiro. Foi instituída com base legal na Portaria nº 397, de 10.10.2002.

Acompanhando o dinamismo das ocupações, a CBO tem por filosofia sua atualização constante de forma a expor, com a maior fidelidade possível, as diversas atividades profissionais existentes em todo o país, sem diferenciação entre as profissões regulamentadas e as de livre exercício profissional.

A CBO tem o reconhecimento no sentido classificatório da existência de determinada ocupação e não da sua regulamentação. A regulamentação da profissão diferentemente da CBO, é realizada por Lei cuja apreciação é feita pelo Congresso Nacional, por meio de seus Deputados e Senadores e submetida à sanção do Presidente da República. A CBO não tem poder de Regular Profissões.

Seus dados alimentam as bases estatísticas de trabalho e servem de subsídio para a formulação de políticas públicas de emprego.

Os trabalhadores sentem-se amparados e valorizados ao terem acesso a um documento, elaborado pelo governo, que identifica e reconhece seu ofício. As inclusões das ocupações na CBO têm gerado, tanto para categorias profissionais quanto para os trabalhadores, uma maior visibilidade, um sentimento de valorização e de inclusão social. A atualização da CBO ocorre em geral, anualmente e tem como foco revisões de descrições com incorporação de ocupações e famílias ocupacionais que englobem todos os setores da atividade econômica e segmentos do mercado de trabalho, e não somente canalizados para algum setor específico.

Conforme o Ministério do Trabalho e Emprego, a Classificação Brasileira de Ocupações - CBO é o documento normalizador do reconhecimento (para fins classificatórios, sem função de regulamentação profissional), da nomeação e da codificação dos títulos e conteúdos das ocupações do mercado de trabalho brasileiro. É ao mesmo tempo uma classificação enumerativa e uma classificação descritiva. Classificação enumerativa: codifica empregos e outras situações de trabalho para fins estatísticos de registros administrativos, censos populacionais e outras pesquisas domiciliares. Inclui códigos e títulos ocupacionais e a descrição sumária. Ela também é conhecida pelos nomes de nomenclatura ocupacional e estrutura ocupacional. Classificação descritiva: inventaria detalhadamente as atividades realizadas no trabalho, os requisitos de formação e experiência profissionais e as condições de trabalho.

Código	Título	Total de Empregos
1421	Gerentes administrativos e financeiros	124.165

Figura 1. Exemplo de classificação de ocupação

A nomenclatura ou estrutura da CBO é o conjunto de códigos e títulos que é utilizada na sua função enumerativa. É uma estrutura hierárquico-piramidal composta de:

- 10 grandes grupos (GG)
- 47 sete subgrupos principais (SGP)
- 192 subgrupos (SG)
- 596 grupos de base ou famílias ocupacionais (FO), onde se agrupam 2.422 ocupações e cerca de 7.258 títulos sinônimos.

Grande grupo é a categoria de classificação mais agregada. Reúne amplas áreas de emprego, mais do que tipos específicos de trabalho. Por força de sua amplitude, nem sempre se estabelecem inter-relações dos conjuntos aí reunidos. Representado pelo 1º número do código da família (1 mais à esquerda, na figura 1).

Subgrupo Principal trata de agrupamento mais restrito que o grande grupo, e configura, principalmente, as grandes linhas do mercado de trabalho. Representado pelos 2 primeiros números do código da família (14 na figura 1).

Subgrupo, também denominado grupo primário, grupo unitário e família ocupacional, reúne ocupações que apresentam estreito parentesco tanto em relação à natureza de trabalho quanto aos níveis de qualificação exigidos. Representado pelos 3 primeiros números do código da família (142 na figura 1).

Família, a unidade do sistema de classificação. Para efeitos práticos, define-se a ocupação como o conjunto de postos de trabalho substancialmente iguais quanto a sua natureza e as qualificações exigidas (o posto de trabalho corresponde a cada unidade de trabalho disponível ou satisfeita). Constitui-se de tarefas, obrigações e responsabilidades atribuídas a cada trabalhador. Pode-se ainda conceituar a ocupação como o conjunto articulado de funções, tarefas e operações destinadas à obtenção de produtos ou serviços. Representado pelo código total de 4 números (1421 na figura 1).

Para esta análise buscou-se dados mais específicos possíveis, utilizando-se também os códigos que diferenciam as ocupações dentro de cada Família, pela escolha da opção “CBO Ocupação 2002” quando da estruturação da consulta à base da RAIS.

Analista de suporte computacional	2124-20
<i>Analista de suporte de banco de dados</i>	2124-20
<i>Analista de suporte de sistema</i>	2124-20
<i>Analista de suporte técnico</i>	2124-20
<i>Analista de telecomunicação</i>	2124-10
<i>Analista de testes automatizados de ti</i>	2124-30
Analista de testes de tecnologia da informação	2124-30

Figura 2. Nível mais descritivo de uma ocupação

Os dados da RAIS encontram-se disponíveis em plataforma pública na internet, acessíveis através do link <http://bi.mte.gov.br/bgcaged/login.php> (usuário: *básico*, senha: 12345678). Após acessar a plataforma, escolhe-se no menu do lado esquerdo a opção “RAIS”, depois a opção “RAIS VÍNCULOS [+]” e por fim o ano/período que deseja analisar, podendo obter dados estatísticos a partir de 1985. Tendo-se optado por escolher a análise do período mais recente disponível, 2019, deve-se escolher a opção “Ano corrente a 2002”.

Após o procedimento de entrada, chega-se à tela de definições da pesquisa. A primeira definição a ser configurada são as chamadas “Seleções aceleradoras”, onde se escolhe o ano do vínculo e qual tipo de vínculo, se ativo ou não em 31/12. Escolheu-se o ano de 2019 no primeiro e a opção “Todos” na segunda com vistas a ter a visão mais ampla possível sobre os dados ocupacionais, inclusive daquelas funções/cargos sazonais.

Seleção	Condição	Valor
Ano	=	2019

Figura 3. Configuração da pesquisa

A plataforma fornece como resposta uma planilha e, portanto, deve-se configurar quais informações devem estar presentes nas linhas e nas colunas (atentando-se à limitação de no máximo 1.000 colunas). Essa configuração é feita na opção “Estrutura” do menu à esquerda. Deve-se escolher para as linhas a opção “CBO Ocupação 2002” e para a coluna a opção “Município”. Quando é feita essa última escolha, surge novo elemento no formulário de configuração da tabela, com a possibilidade de escolher o Estado a que se refere a consulta. Em razão da limitação da quantidade de colunas possíveis em uma consulta, optou-se por consultar os dados dos Municípios de cada Estado por vez. Demais opções referentes à estrutura da tabela não devem ser alteradas. Concluída a configuração, executa-se a consulta pressionando o botão em formato de raio, na barra de ferramentas superior. Nova janela abre com o status do processo de consulta, cujo resultado é exibido na mesma janela em que foi realizada a configuração. Para obtenção de todos os dados necessários à análise, a consulta deve ser repetida para cada um dos Estados e para o Distrito Federal.

Informações: RAIS Vínculo Id

Definição da tabela

- ☐ Seleções aceleradoras
- ☐ Estrutura
- ☐ Documentação
- ☐ Ordem
- ☒ Seleções por assunto

Linha	CBO Ocupação 2002	
Coluna	Município	Maranhão
Subcoluna	-----Não-----	
Quadro	-----Não-----	
Sublinha	-----Não-----	
Conteúdo	-> Frequência -Individual Idade -Vínculo Qtd Hora Contr Tempo Emprego VI Remun Dezembro (SM)	

Opções

☐ Exibe linha zerada ☐ Exibe coluna zerada

Esconder total


☐ Linha ☐ Coluna ☐ Subcoluna ☐ Quadro ☐ Sublinha

Seleção	Condição	Valor
Ano	=	2019

Figura 4. Escolha das linhas e colunas a serem retornadas pela pesquisa

Dardo - Google Chrome

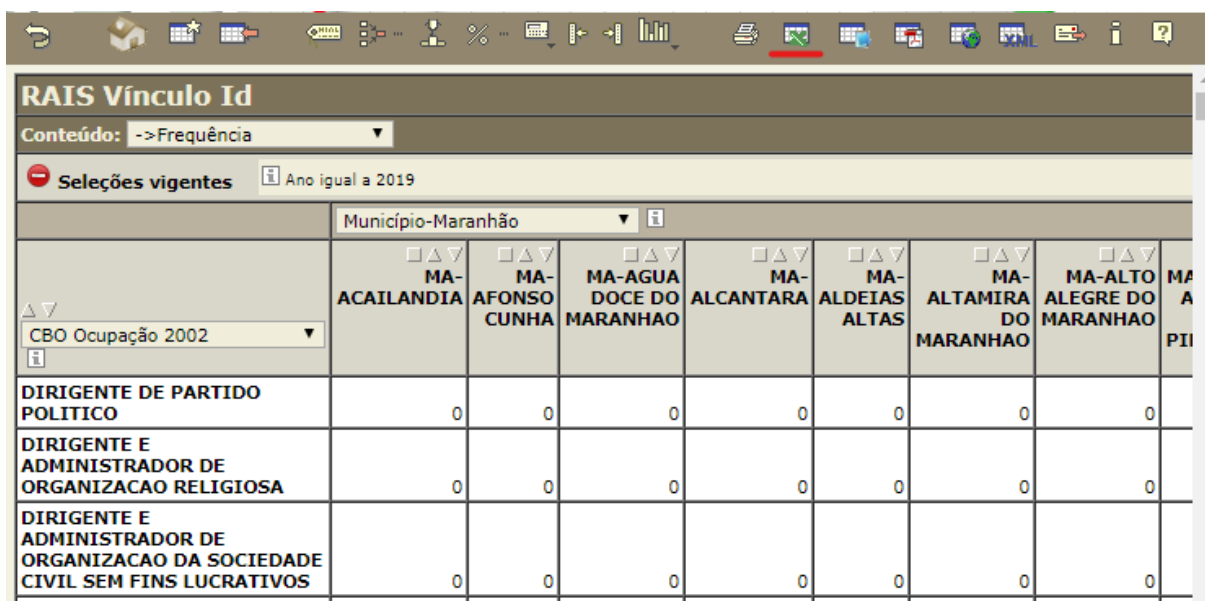
Não seguro | bi.mte.gov.br/temp/consulta62177189.h...

Acompanhamento da execução		
Registros a ler	66.667.417	100%
Registros lidos	4.608.072	6,9%
Registros selecionados	938.793	1,4%
Velocidade	12.201.426 reg/min	
		

Se esta janela não for recarregada, tecle F5!

Figura 5. Processamento da pesquisa

Encerrada a execução e carregada a tabela com os dados requisitados, pode-se exportá-las em diversos formatos, tendo-se optado pelo “.csv”.



The screenshot shows the RAIS Vínculo Id application interface. At the top, there's a toolbar with various icons. Below it, the title bar reads "RAIS Vínculo Id". A dropdown menu shows "Conteúdo: ->Frequência". Below that, a section labeled "Seleções vigentes" includes a filter for "Ano igual a 2019". The main table is titled "Município-Maranhão" and has columns for different municipalities: MA-ACAILANDIA, MA-AFONSO CUNHA, MA-AGUA DOCE DO MARANHÃO, MA-ALCANTARA, MA-ALDEIAS ALTAS, MA-ALTAMIRA DO MARANHÃO, MA-ALTO ALEGRE DO MARANHÃO, and MA-PII. The rows represent different occupations: "DIRIGENTE DE PARTIDO POLITICO", "DIRIGENTE E ADMINISTRADOR DE ORGANIZACAO RELIGIOSA", and "DIRIGENTE E ADMINISTRADOR DE ORGANIZACAO DA SOCIEDADE CIVIL SEM FINS LUCRATIVOS". Each cell in the table contains the number "0".

	MA-ACAILANDIA	MA-AFONSO CUNHA	MA-AGUA DOCE DO MARANHÃO	MA-ALCANTARA	MA-ALDEIAS ALTAS	MA-ALTAMIRA DO MARANHÃO	MA-ALTO ALEGRE DO MARANHÃO	MA-PII
DIRIGENTE DE PARTIDO POLITICO	0	0	0	0	0	0	0	
DIRIGENTE E ADMINISTRADOR DE ORGANIZACAO RELIGIOSA	0	0	0	0	0	0	0	
DIRIGENTE E ADMINISTRADOR DE ORGANIZACAO DA SOCIEDADE CIVIL SEM FINS LUCRATIVOS	0	0	0	0	0	0	0	

Figura 6. Resultado apresentado

Pesquisados todos os Estados e o Distrito Federal, obteve-se, em 15/12/2020, 27 planilhas no formato “.csv”, em que as linhas indicam a descrição das ocupações para as quais houve informação de empregado/servidor e as colunas indicam cada um dos Municípios que compõem o Estado. No cruzamento dos Municípios com a descrição das ocupações são informados os quantitativos informados nas RAIS enviadas pelos Entes.

3. Processamento/Tratamento de Dados

Foram realizados alguns ajustes nas bases obtidas para que fosse possível o uso dos modelos de Machine Learning escolhidos.

Os dados da RAIS são fornecidos em arquivos com extensão “.csv”, um arquivo por Estado, com um Município por coluna e todas as ocupações para as quais houve informação de ao menos um funcionário, para algum Município daquele Estado, nas linhas.

Acompanham as informações requisitadas a indicação do tipo de dado fornecido nas linhas. Nos dataset em análise há a informação “CBO Ocupação 2002” que indica a versão mais recente da Classificação Brasileira de Ocupações.

	A	B	C	D
1	Coluna	Município-Maranhão		
2	CBO Ocupação 2002	MA-ACAILANDIA	MA-AFONSO CUNHA	MA-AGUA DOCE DO
3	DIRIGENTE DE PARTIDO POLITICO	0	0	
4	DIRIGENTE E ADMINISTRADOR DE ORGANIZACAO RELIGI	0	0	
5	DIRIGENTE E ADMINISTRADOR DE ORGANIZACAO DA SO	0	0	
6	DIRETOR DE PLANEJAMENTO ESTRATEGICO	4	0	
7	DIRETOR GERAL DE EMPRESA E ORGANIZACOES (EXCETO	6	0	
8	DIRETOR DE PRODUCAO E OPERACOES EM EMPRESA AGF	0	0	
9	DIRETOR DE PRODUCAO E OPERACOES EM EMPRESA AQUI	0	0	
10	DIRETOR DE PRODUCAO E OPERACOES EM EMPRESA FLO	0	0	
11	DIRETOR DE PRODUCAO E OPERACOES EM EMPRESA PES	0	0	
12	DIRETOR DE PRODUCAO E OPERACOES DA INDUSTRIA DE	0	0	

Figura 7. Formato dos dados nas primeiras linhas

No fim da planilha, após a indicação de todas as ocupações, encontra-se a indicação do total de funcionários por Município, o quantitativo de funcionários com ocupações não classificadas e as regras de seleção utilizadas na obtenção do dataset.

1922	PEDREIRO DE CONSERVACAO DE VIAS PERMANENTES (E	0	0
1923	AUXILIAR GERAL DE CONSERVACAO DE VIAS PERMANEN	9	0
1924	{# class}	258	3
1925	Total	24056	317
1926			
1927	Seleções vigentes		
1928	Variável	Critério	Valor
1929	Ano	igual a	2019

Figura 8. Formato dos dados nas últimas linhas

É indicado também, nas últimas colunas, a informação do total de funcionários por ocupação naquele Estado.

HH	HI	HJ	HK	
MA-VITOF	MA-VITORINO FREIRE	MA-ZE DOCA	Total	
0	0	0	1	
0	0	0	3	
0	0	0	10	
0	0	0	110	
0	0	0	259	
0	0	0	2	
0	0	0	1	
0	0	0	3	

Figura 9. Últimas colunas

Para a análise, elaborou-se *Jupyter Notebooks* para leitura e processamento de todos os 27 arquivos “.csv” obtidos, que foram lidos e carregados em um único DataFrame. Inicialmente importou-se as bibliotecas para processamento e análise do dataset.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Figura 10. Importação das bibliotecas de processamento dos dados

Em seguida criou-se um array com o nome de todos os arquivos (salvos na pasta *Bases 2019*).

```
#Obter lista de arquivos baixados

from os import listdir
from os.path import isfile, join
arquivos_csv_estados = [f for f in listdir('Bases 2019') if
                        isfile(join('Bases 2019', f))]
```

Figura 11. Array com nomes dos arquivos

São então lidos cada um dos arquivos como *dataframes* e salvos em um dicionário em que a chave é o nome do Estado. Deve-se indicar o separador “;”, a codificação ‘latin-1’ para identificação dos acentos e cedilhas e ignorar a primeira linha, que não possui dados úteis à análise. Indica-se também a primeira coluna como índice.

```
# Ler o arquivo de cada estado e salvar em um dicionário de dataframes
# em que a chave é o nome do Estado

df_estados = {}

for arquivo in arquivos_csv_estados:
    df_estados[arquivo[:-4]] = pd.read_csv('Bases 2019\\'
                                           + arquivo, sep=';',
                                           encoding='latin-1',
                                           skiprows=[0],
                                           index_col="CBO Ocupação 2002")
```

Figura 12. Código para leitura de cada arquivo “.csv”

Efetua-se, então, a junção dos dataframes de todos os Estados.

```
# Join de todos os dataframes dos Estados

juncao = df_estados['Acre']

for k,v in df_estados.items():
    if (k != 'Acre'):
        juncao = juncao.join(v,how='outer')
```

Figura 13. Criação do dataframe com informações de todos os Estados

Com a junção, surgem diversas células com a informação “NaN”, que são ocupações que não foram indicadas naquele determinado Município. Preencheu-se tais células com ‘0’. Por fim, é feita a transposição do dataframe para que fique um Município por linha e estes sejam os índices.

```
# Eliminando a linha NaN que surgiu
dataset = dataset.drop(dataset.index[0])
```

```
# Fazendo a transposição e preenchendo as células que possuem NaN
dataset = dataset.transpose()
dataset = dataset.fillna(0)
```

Figura 14. Substituição dos valores nulos por 0 e transposição do dataframe

CBO Ocupação 2002	ABATEDOR	ACABADOR DE CALÇADOS	ACABADOR DE EMBALAGENS (FLEXÍVEIS E CARTOTÉCNICAS)	ACABADOR DE SUPERFÍCIES DE CONCRETO	ACOUGUEIRO	ACR
AC-ACRELÂNDIA	0	0	0	0	18	
AC-ASSIS BRASIL	3	0	0	0	5	
AC-BRASILÉIA	3	0	0	0	23	
AC-BUJARI	0	0	0	0	9	
AC-CAPIXABA	0	0	0	0	6	
...
TO-TOCANTINÓPOLIS	8	0	0	0	24	
TO-TUPIRAMA	0	0	0	0	1	
TO-TUPIRATINS	0	0	0	0	1	
TO-WANDERLÂNDIA	0	0	0	0	1	
TO-XAMBIOA	0	0	0	0	2	

5570 rows x 2567 columns



Figura 15. *Dataframe* completo com os 5.570 Municípios e 2.567 ocupações identificadas

É sobre essa base que foram feitos os experimentos com clusterização. São 5.570 linhas, uma para cada Município, e 2.567 colunas, uma para cada ocupação informada pelos Entes. As células indicam as quantidades de funcionários informados para cada ocupação por cada Município, com valores que variam de 0 a 'n', sendo que não há, teoricamente, limite para 'n', pois não há limite para a quantidade de funcionários por ocupação.

Não há registros duplicados, bem como não há informações ausentes, pois os 'NaN' que surgiram após a junção dos dados de todos os Municípios significam apenas ocupações para as quais aquele Ente não contratou, podendo ser substituídos pelo número 0.

A outra base de dados utilizada contém a informação da latitude e longitude das sedes de cada Município. A informação é relacionada ao código do Município, sendo fornecido um segundo arquivo, no formato ".txt", com a relação de códigos e respectivos municípios e estados.

	A	B	C	D
1	CODIGO MUNICIPIO	LATITUDE	LONGITUDE	
2	520005	-16.75	-49.43	
3	310010	-18.48	-47.4	
4	520010	-16.2	-48.7	
5	310020	-19.15	-45.44	
6	150010	-1.71	-48.88	
7	230010	-7.35	-39.04	
8	290010	-13.25	-41.66	
9	290020	-8.72	-39.11	
10	410010	-23.3	-50.31	
11	420005	-27.61	-51.02	
12	150013	-4.95	-48.39	
13	420010	-26.56	-52.32	

Figura 16. Localização das sedes

```

Codigo_Nome_Municipios.txt - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
CODIGO MUNICIPIO;UF;NOME MUNICIPIO
"110001";"RO";"Alta Floresta d'Oeste"
"110002";"RO";"Ariquemes"
"110003";"RO";"Cabixi"
"110004";"RO";"Cacoal"
"110005";"RO";"Cerejeiras"
"110006";"RO";"Colorado do Oeste"
"110007";"RO";"Corumbiara"
"110008";"RO";"Costa Marques"
"110009";"RO";"Espigão d'Oeste"
"110010";"RO";"Guajará-Mirim"
"110011";"RO";"Jaru"
"110012";"RO";"Ji-Paraná"

```

Figura 17. Relação de códigos municipais

Após importação das bases com dados de localização dos Municípios, os dados foram concatenados em um único *dataframe*. Um pequeno ajuste ainda precisou ser feito: precisou-se retirar as acentuações e cedilhas dos nomes dos Municípios uma vez que a base obtida na RAIS não os possuía, o que atrapalha junção das bases.

```

import pandas as pd

coordenadas = pd.read_csv('Sedes_Coordenadas_Municipios.csv',
                          sep = ';', index_col=0)

```

Figura 18. Importação da localização das sedes

```
nome = pd.read_csv('Codigo_Nome_Municipios.txt', encoding='latin1',
                  sep=';', index_col=0)
```

Figura 19. Importação dos nomes e códigos dos Municípios

```
import unicodedata

def converte_utf(com_acento):
    sem_acento = unicodedata.normalize("NFD", com_acento)
    sem_acento = sem_acento.encode("ascii", "ignore")
    sem_acento = sem_acento.decode("utf-8")
    return sem_acento
```

Figura 20. Função para retirar acentos

```
df2['UF-MUNICÍPIO'] = df2['UF-MUNICÍPIO'].apply(converte_utf)
```

```
df2.set_index(['UF-MUNICÍPIO'], inplace=True)
```

Figura 21. Ajuste dos nomes e configuração do index

Após carregada e ajustada a base de localizações, efetuou-se a sua junção com a base de dados da RAIS (já processada e com diversas novas colunas, uma para cada processamento feito durante a análise).

```
dataset_analises = dataset_analises.join(df2, how='outer')
```

Figura 22. Junção definitiva das bases

4. Análise e Exploração dos Dados

A análise dos dados se deu principalmente olhando-se a descrição das ocupações e quantidades informadas. Pôde-se identificar diversas peculiaridades nos dados informados no RAIS, tais como ocupações que aparentemente não deveriam estar presentes entre empregados dos Municípios (salva-vidas e

vendedores e demonstradores em lojas ou mercados), e certas ocupações que concentram boa parte dos empregados municipais (as relacionadas à educação básica, como professores de nível médio no ensino fundamental). Assim, espera-se que esses Municípios com informações “peculiares”, aparentemente incomuns, sejam agrupados em um mesmo cluster.

Busca-se com a presente análise a obtenção de agrupamentos de Municípios com estruturas de ocupações parecidas e que para isso sejam levadas em consideração todas as ocupações possíveis, até as mais incomuns e raras. Tal análise, de 2.567 ocupações, é inviável sem uma ferramenta que possa lidar com grande quantidade de informações não previamente classificadas. Assim optou-se por ferramentas de Machine Learning de aprendizado não supervisionado. *Insights* sobre Municípios com informações extraordinárias podem servir de ponto de partida para análise mais aprofundadas, confrontando-se os dados obtidos com dados de outras fontes, ou integrando-os a diversas outras informações para uma visão socioeconômica completa dos Municípios.

Portanto, não se busca certezas ou verdades absolutas nas análises descritas a seguir, mas uma classificação/organização que possa indicar pontos de destaque, previsíveis, ou inesperados, através de agrupamentos de Municípios com alguma afinidade em sua estrutura de pessoal. Acredita-se que tais informações são interessantes fontes de *insights* sobre a base de dados em análise.

5. Criação de Modelos de Machine Learning

Os experimentos em aprendizado não supervisionado utilizando-se clusters foram feitos em Python, através da elaboração de *Jupyter Notebook*.

A ideia do trabalho é identificar agrupamentos de Municípios pela análise de suas folhas de pagamentos. Tal análise é dificultada pela quantidade de ocupações (dimensões/*features*) que os Municípios podem ter (são 2.567 nesta base de dados). Algumas dessas ocupações, apesar de pouco representativas na quantidade (em relação ao total da folha do Ente), podem indicar alguma característica peculiar, sendo importante na análise comparativa que se propõe. Assim, foram mantidas na análise todas as ocupações.

Portanto, deseja-se produzir um importante indicador de comparação entre as despesas municipais uma vez que gastos com pessoal costumam ser o maior dentro

do orçamento de um Município. Com a organização em cluster, através da consideração de todas as ocupações informadas – e não apenas a quantidade total ou a quantidade de um grupo pequeno de ocupações – obtêm-se importantes informações e *insights* que podem auxiliar ou integrar, junto com outros dados (socioeconômicos, por exemplo), uma avaliação dos gastos públicos municipais.

Para alcançar o objetivo proposto, utilizou-se a abordagem de clusterização com K-Means e HDBSCAN. Diversos testes e ajustes foram feitos durante a execução dos algoritmos em busca da identificação de agrupamentos que pudessem representar e destacar grupos de Municípios com estruturas ocupacionais similares.

Inicialmente aplicou-se o K-Means aos dados originais, com suas 2.567 colunas. Para tanto, partiu-se da apuração da quantidade ideal de clusters usando o Método do Cotovelo, que indicou 3 clusters.

```
# Escolha do número de clusters

#Cotovelo
sse = []
for k in range(1, 15):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(dataset)
    sse.append(kmeans.inertia_)

plt.style.use("fivethirtyeight")
plt.plot(range(1, 15), sse)
plt.xticks(range(1, 15))
plt.xlabel("Número de Clusters")
plt.ylabel("SSE")
plt.show()
```

Figura 23. Código para apuração da quantidade ideal de clusters pelo Método do Cotovelo

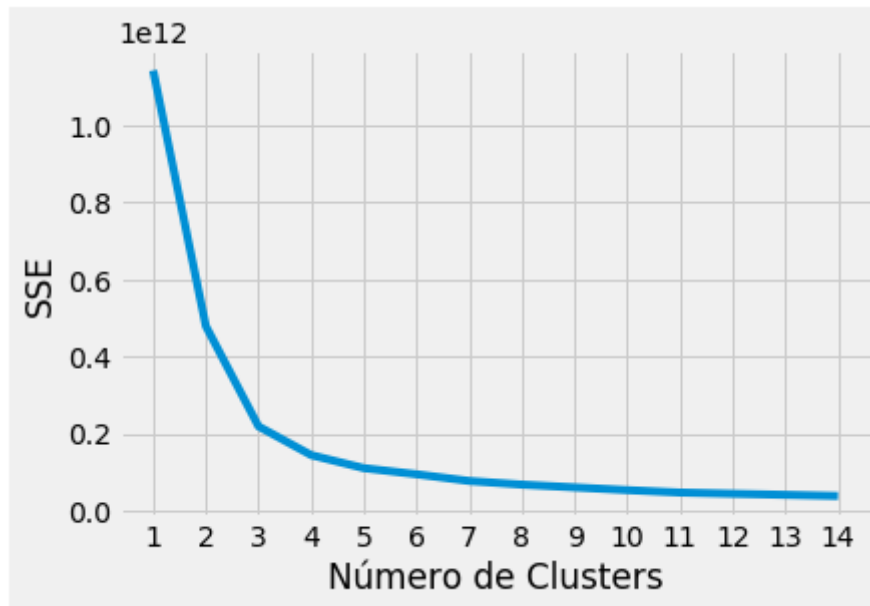


Figura 24. Apuração da quantidade ideal de clusters

Apurou-se, ainda, utilizando os coeficientes *Silhouette*, que também indicou 3 clusters.

```
# Escolha do número de clusters

#Silhouette

from sklearn.metrics import silhouette_score

# Lista que armazena os coeficientes silhouette para cada k
silhouette_coefficients = []

# Pesquisa inicia em 2
for k in range(2, 15):
    kmeans = KMeans(n_clusters=k, random_state = 101)
    kmeans.fit(dataset)
    score = silhouette_score(dataset, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 15), silhouette_coefficients)
plt.xticks(range(2, 15))
plt.xlabel("Número de Clusters")
plt.ylabel("Coeficiente Silhouette")
plt.show()
```

Figura 25. Código para apuração da quantidade ideal de clusters pelo método dos coeficientes *Silhouette*

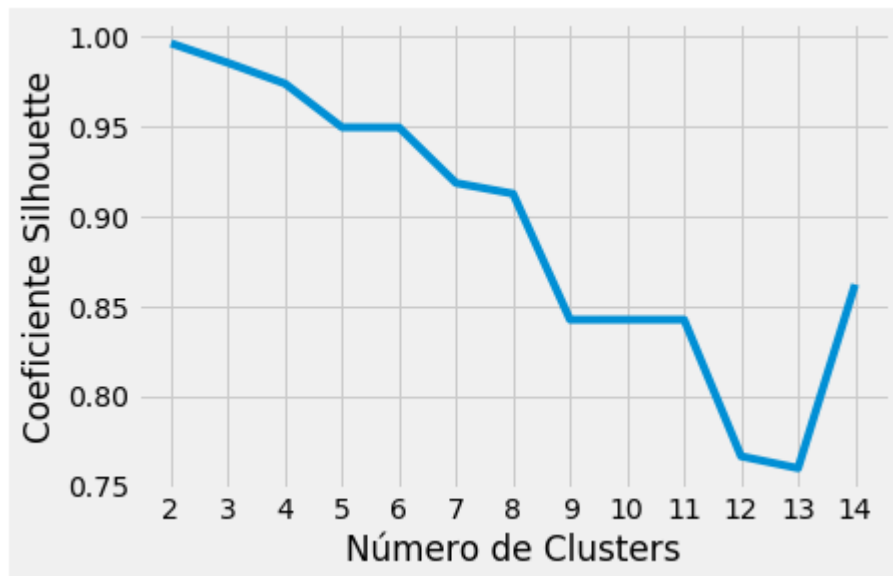


Figura 26. Quantidade de clusters pelo Coeficiente Silhouette

Aplicou-se, então, o K-Means com número de clusters igual a 3 sobre a base original. Nessa abordagem, os clusters obtidos aparentemente apenas levaram em consideração o tamanho das folhas de pagamento, pois colocou em um cluster o município de São Paulo (que tem a maior quantidade de funcionários), em outro as maiores capitais depois de São Paulo: Salvador, Fortaleza, Brasília, Goiânia, Belo Horizonte, Curitiba, Recife, Rio de Janeiro e Porto Alegre. E, no terceiro cluster, todos os demais 5.560 Municípios. Assim, pouca informação pôde ser extraída dessa abordagem, uma vez que aparentemente apenas distribuiu os Municípios pelo tamanho de suas folhas.

No experimento seguinte, os dados foram normalizados utilizando a função `StandardScaler` do `SKLearn` e então aplicado novamente o algoritmo K-Means.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_rescaled = scaler.fit_transform(dataset)

dataset_normalizado = pd.DataFrame(data_rescaled)
```

Figura 27. Código de normalização da base

Aplicou-se o Método do Cotovelo para apuração do número ideal de clusters, chegando-se ao número 4.

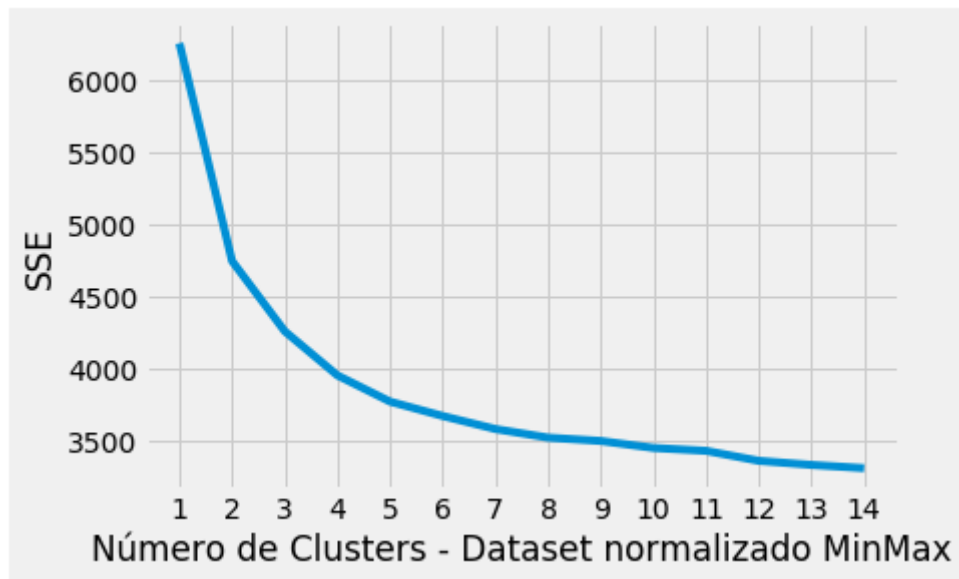


Figura 28. Quantidade de clusters da base normalizada pelo Método do Cotovelo

Também foram analisados os coeficientes *Silhouette*, que confirmaram 4 como a quantidade ideal de clusters.

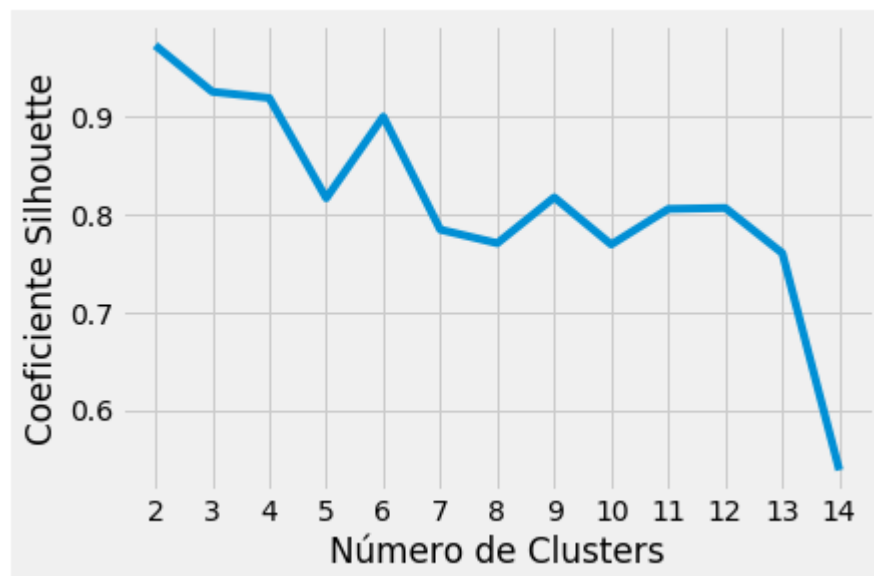


Figura 29. Quantidade de clusters da base normalizada pelo Coeficiente Silhouette

Pouca diferença foi identificada em relação à análise sem normalização dos dados. Mais uma vez alguns clusters ficaram com poucos Municípios; um apenas com São Paulo; outro apenas com o Rio de Janeiro; um terceiro com 9 capitais (Manaus, Salvador, Fortaleza, Brasília, Goiânia, Belo Horizonte, Curitiba, Recife e Porto Alegre); e o quarto com os demais 5.559 Municípios.

Novo experimento foi feito, agora com redução das dimensões utilizando o Principal Component Analysis (PCA). Inicialmente, analisou-se o número ideal de componentes. Para tanto, foi avaliado o percentual de variância em função da quantidade de *features* presentes nos dados. Conforme gráfico, cerca de 95% da variância é causado por apenas duas *features*.

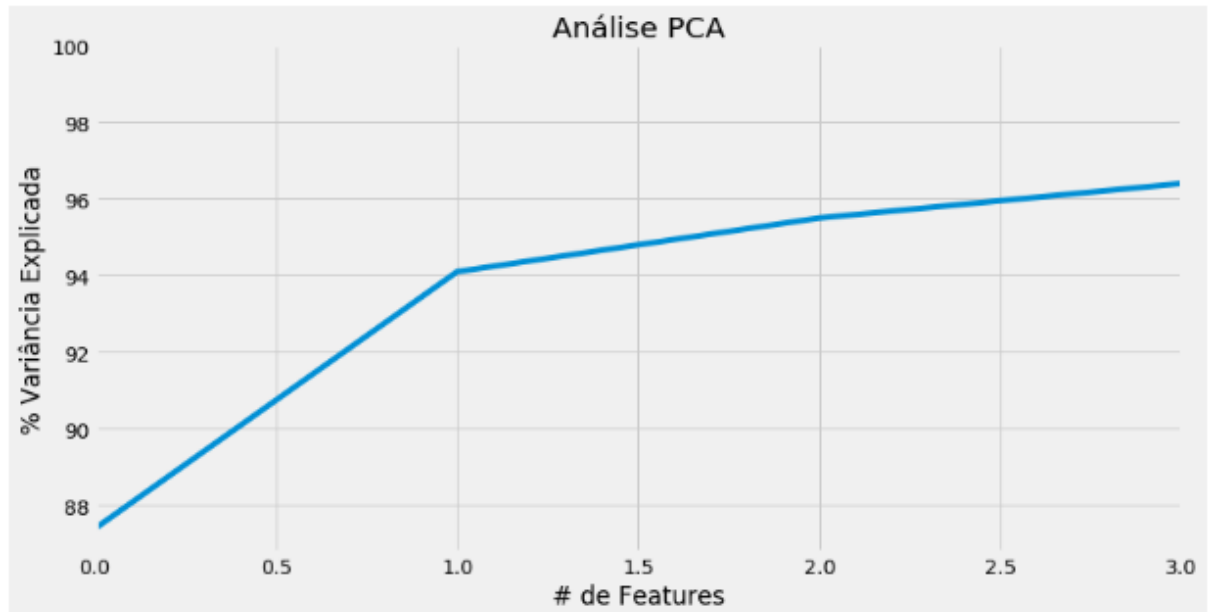


Figura 30. Percentual da variância pela quantidade de features

Foi então aplicado o PCA para reduzir a base de dados para duas dimensões.

```
pca = PCA(n_components=2)
pca_dataset = pca.fit_transform(dataset)
pca_df_dataset = pd.DataFrame(pca_dataset, columns=['pc1', 'pc2'])
```

Figura 31. Redimensionamento dos dados

Apurou-se o número ideal de clusters pelo Método do Cotovelo. Conforme análise do gráfico, o valor ideal é de 3 clusters.

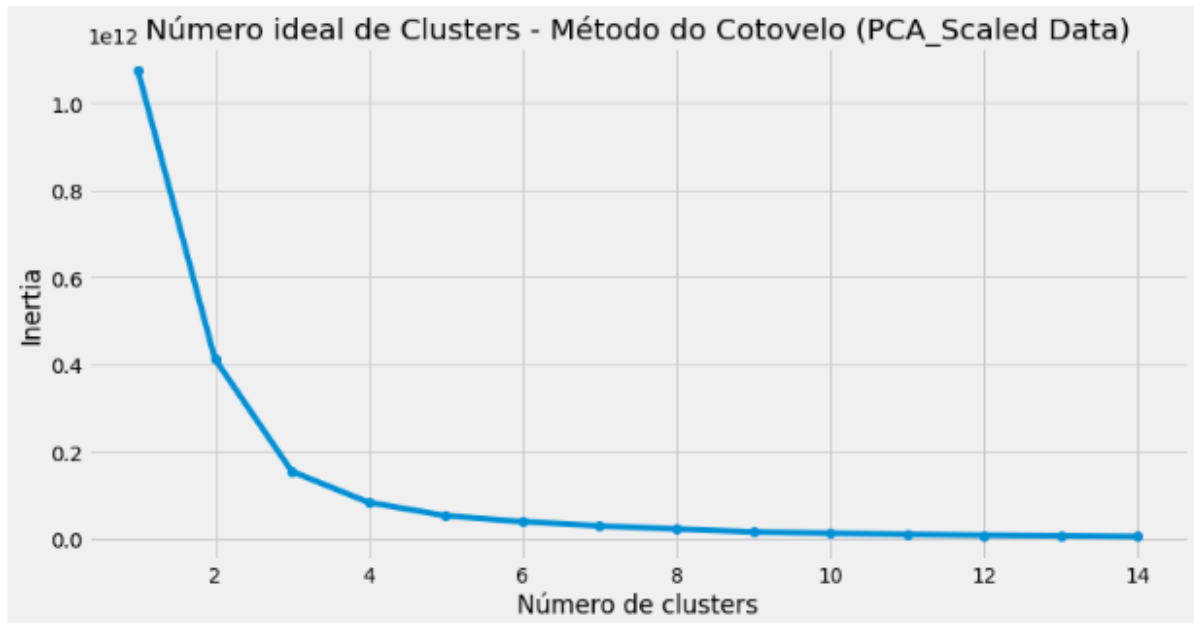


Figura 32. Quantidade ideal de clusters

Aplicou-se o K-Means nos dados redimensionados usando o PCA.

```
kmeans = KMeans(n_clusters=3, random_state=101)
kmeans_pca_dataset = kmeans.fit(pca_df_dataset)
labels_pca_dataset = kmeans_pca_dataset.labels_
clusters_pca_dataset = pd.concat([pca_df_dataset,
                                   pd.DataFrame({'pca_clusters': labels_pca_dataset})],
                                   axis=1)
```

Figura 33. Redução para duas dimensões

Que produziu o gráfico da figura 34:

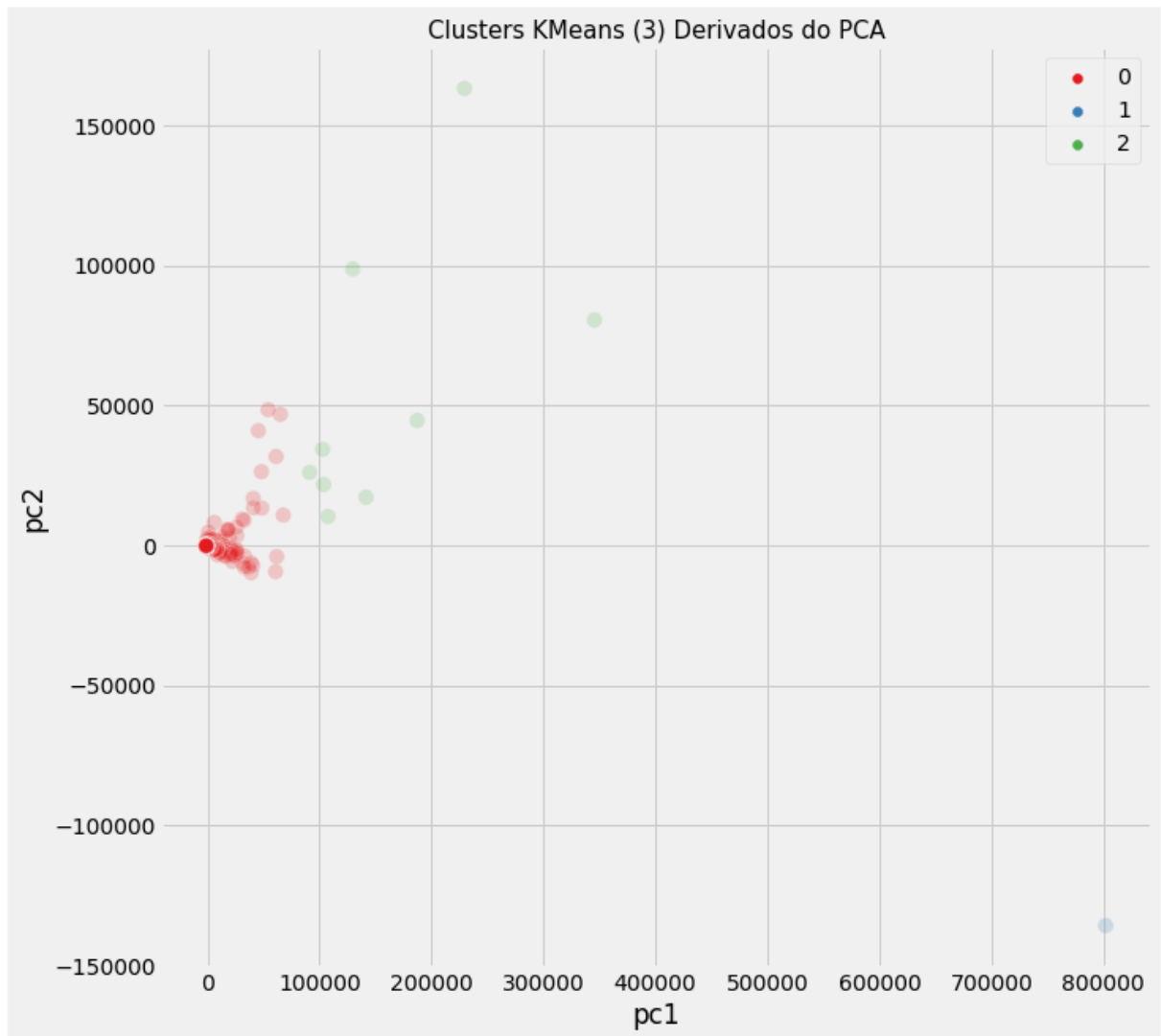


Figura 34. Resultado do K-Means sobre a base reduzida com PCA

A aplicação do K-Means sobre a base redimensionada com PCA obteve a mesma distribuição encontrada na aplicação do K-Means nos dados originais sem redução: um cluster com apenas um Município, São Paulo; outro cluster com 9 Municípios: Salvador, Fortaleza, Brasília, Goiânia, Belo Horizonte, Curitiba, Recife, Rio de Janeiro e Porto Alegre; e o terceiro cluster com os demais 5.560 Municípios. Também não trouxe muita informação.

Aplicou-se então outra técnica de redução de dimensionalidade, a t-Distributed Stochastic Neighbour Embedding (t-SNE) sobre cópia dos dados originais.

```

from sklearn.manifold import TSNE
tSNE=TSNE(n_components=2)
tSNE_result=tSNE.fit_transform(dataset3)

x=tSNE_result[:,0]
y=tSNE_result[:,1]

```

Figura 35. Código de redução com t-SNE

O resultado da aplicação do t-SNE sobre os dados originais foi salvo em um *dataframe* com os nomes dos Municípios e as duas coordenadas, 'x' e 'y'.

	x	y
AC-ACRELANDIA	-0.581011	1.603488
AC-ASSIS BRASIL	-37.403313	-12.700782
AC-BRASILEIA	17.426058	7.299423
AC-BUJARI	20.867836	-27.259459
AC-CAPIXABA	-2.318874	-39.240349
...
TO-TOCANTINOPOLIS	7.305816	4.582033
TO-TUPIRAMA	-29.422674	-24.588524
TO-TUPIRATINS	-21.855385	-14.000484
TO-WANDERLANDIA	8.203442	-11.677487
TO-XAMBIOA	15.864646	-29.397064

5570 rows x 2 columns

Figura 36. Dados após aplicação do t-SNE

Sendo gerado o gráfico da figura 37.

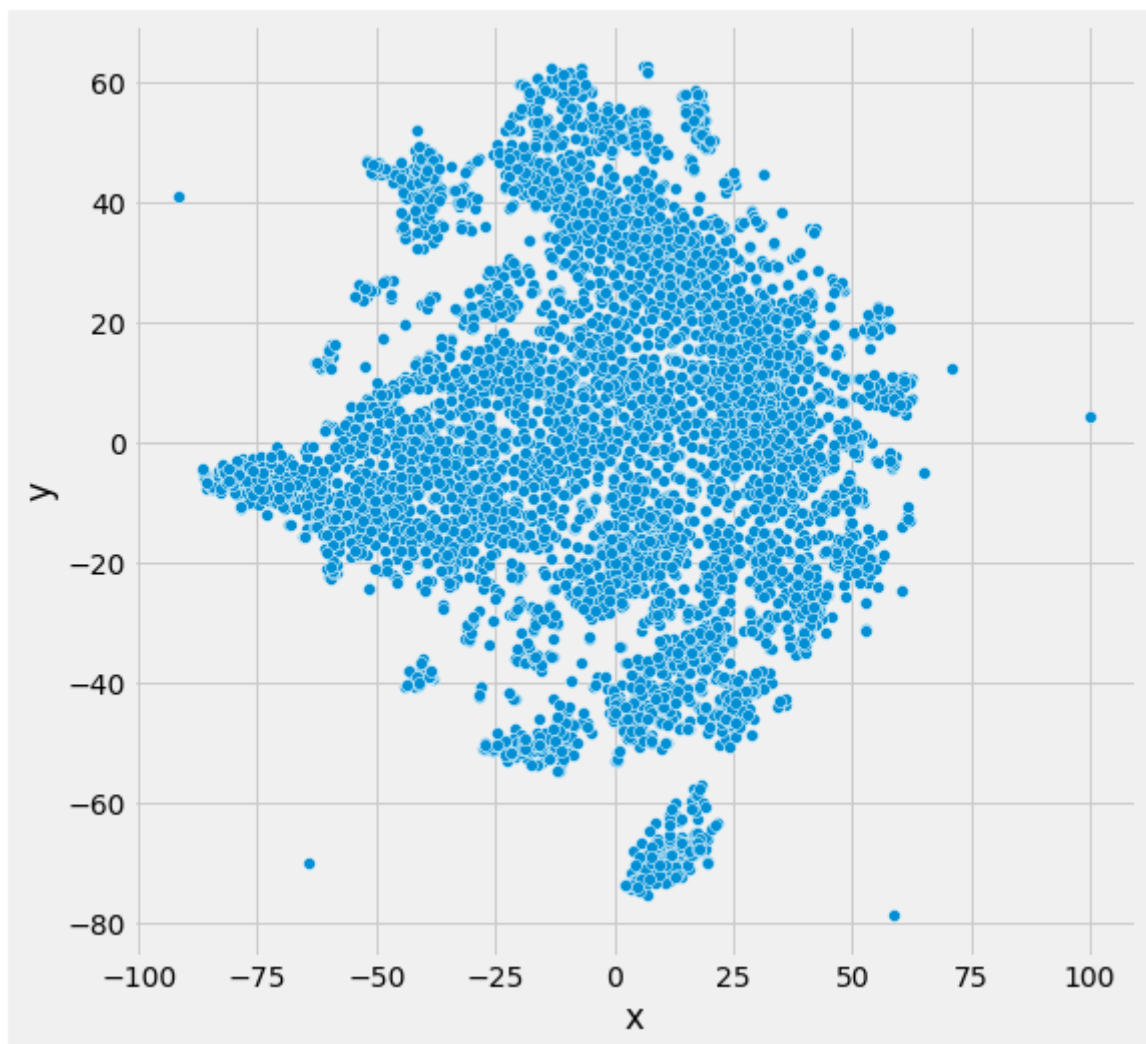


Figura 37. Distribuição gráfica dos dados após a redução com t-SNE

Para se determinar a quantidade de clusters, aplicou-se o Método do Cotovelo, chegando ao número de 4 clusters.

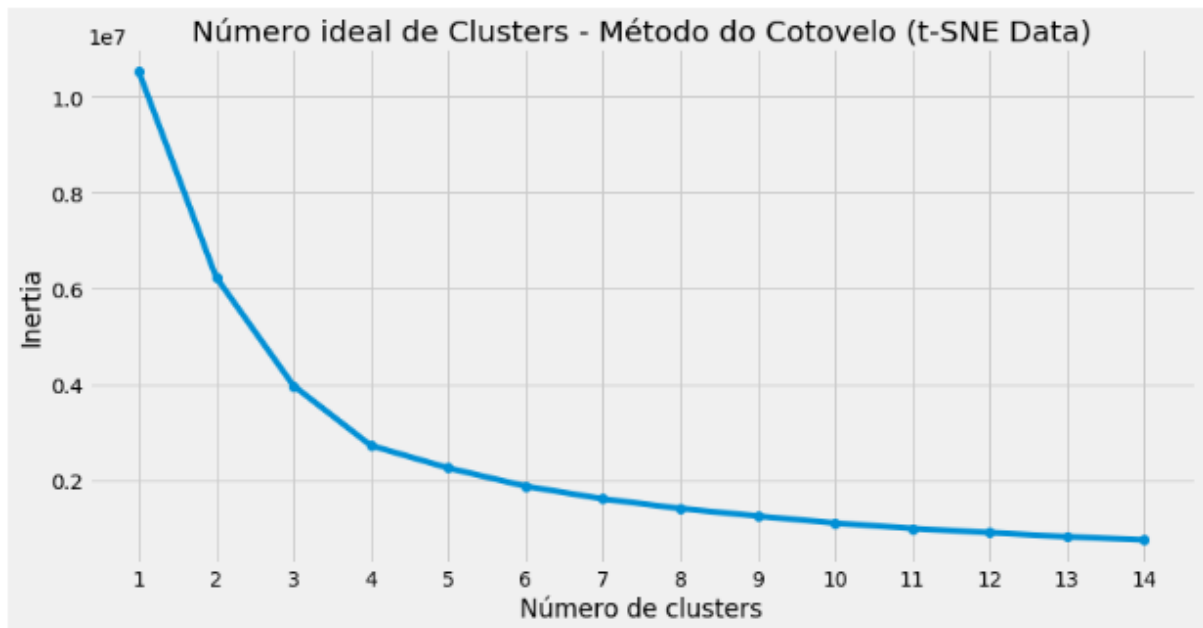


Figura 38. Quantidade ideal de cluster sobre a base reduzida com t-SNE

Apurou-se a quantidade de clusters também pelo método dos coeficientes *Silhouette*. Este também indicou 4 como a quantidade ideal.

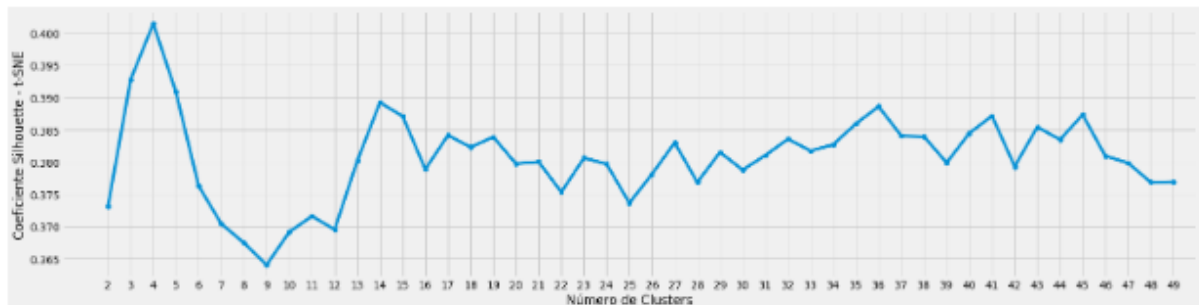


Figura 39. Quantidade de clusters sobre a base reduzida com t-SNE - Coeficiente Silhouette

Aplicar o K-Means com 4 clusters sobre a base redimensionada com o t-SNE resultou em uma distribuição equilibrada de Municípios, ficando o primeiro cluster com 1.317, o segundo com 1.490, o terceiro com 1.516 e o quarto com 1.247.

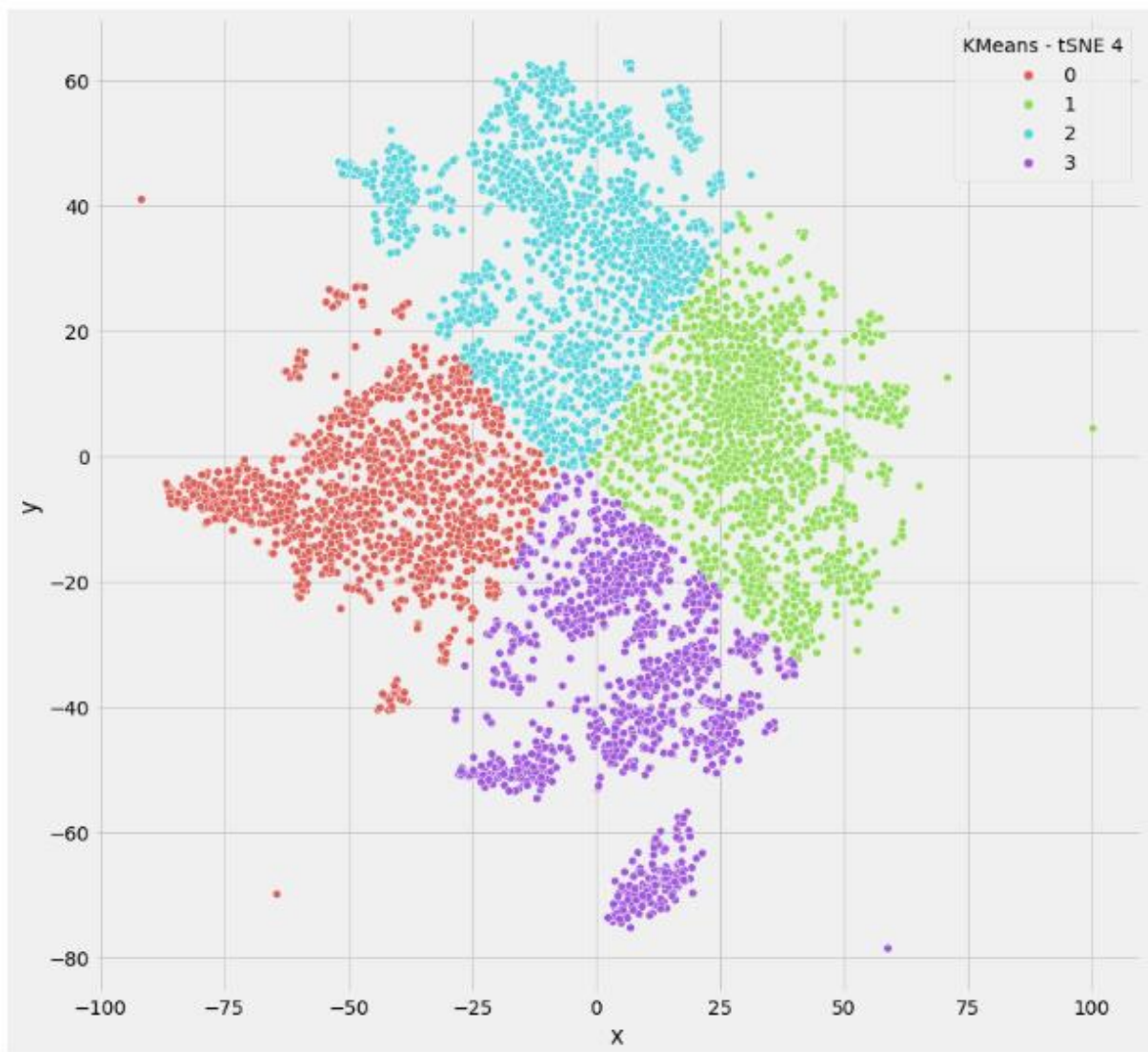


Figura 40. K-Means para 4 clusters sobre a base t-SNE

Uma das dificuldades encontradas na clusterização dos dados foi a determinação do número ideal de clusters, que no K-Means deve ser informado previamente. Assim, para efeitos comparativos, buscou-se uma alternativa de clusterização que fosse capaz de obter por conta própria a quantidade ideal de clusters. Conforme o gráfico anterior (figura 40), gerado após a aplicação de K-Means sobre a base redimensionada com o t-SNE, pôde-se confirmar a sobreposição dos elementos, já apontada nas análises anteriores (K-Means sobre os dados originais, K-Means sobre dados normalizados e K-Means sobre dados redimensionados com PCA), mas também diversos elementos dispersos (*outliers*) ou pequenos grupos mais coesos.

Para obter uma nova abordagem ou nova visão sobre os dados, foi aplicado o algoritmo hierárquico HDBSCAN (Hierarchical Density-Based Spatial Clustering of

Applications with Noise) sobre a base redimensionada com o algoritmo t-SNE. Uma de suas vantagens é a capacidade de rodar com pouco ou nenhum ajuste de parâmetro (este experimento foi conduzido apenas com a avaliação de diversos valores de tamanho de cluster, *min_cluster_size*) sem a necessidade de indicar previamente a quantidade de clusters, bastando informar o seu tamanho ideal. Outra vantagem do algoritmo é sua capacidade de indicar *outliers*.

Utilizou-se a base redimensionada pelo algoritmo t-SNE pois esse, conforme visualização do gráfico da figura 40, conseguiu identificar diferenças entre os dados dos diferentes Municípios, indicando aqueles mais próximos, mas também diversos Municípios mais afastados e, aparentemente, mais agrupados.

Aplicou-se então o HDBSCAN com diversos valores de *min_cluster_size* com o propósito de avaliar os tamanhos dos clusters e a quantidade de outliers resultantes.

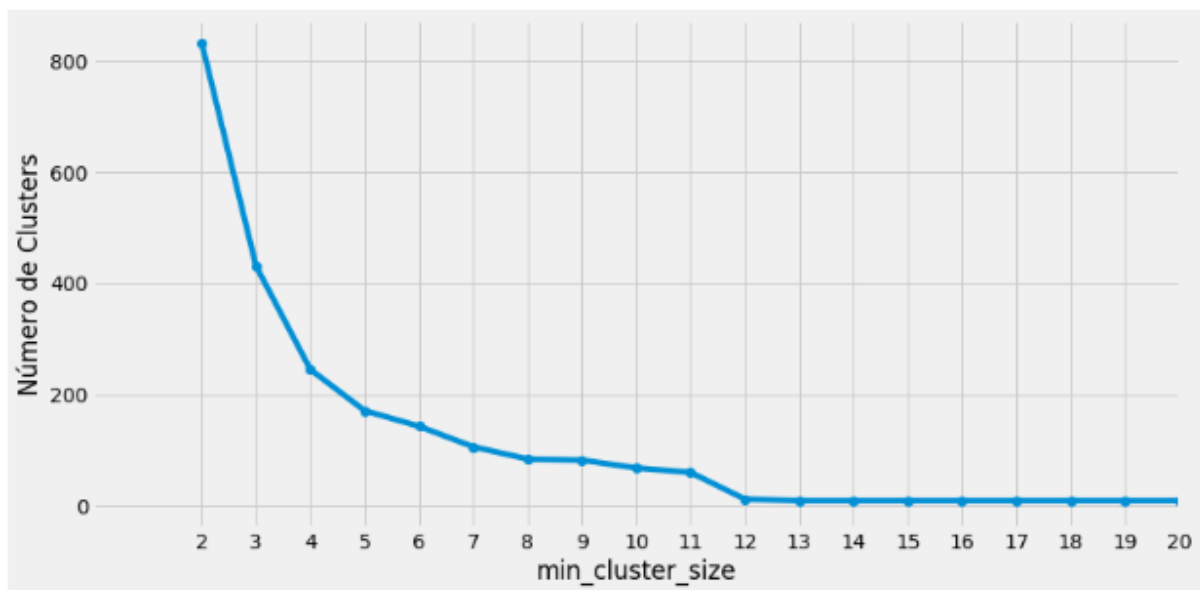


Figura 41. Número de clusters para diferentes valores de *min_cluster_size*

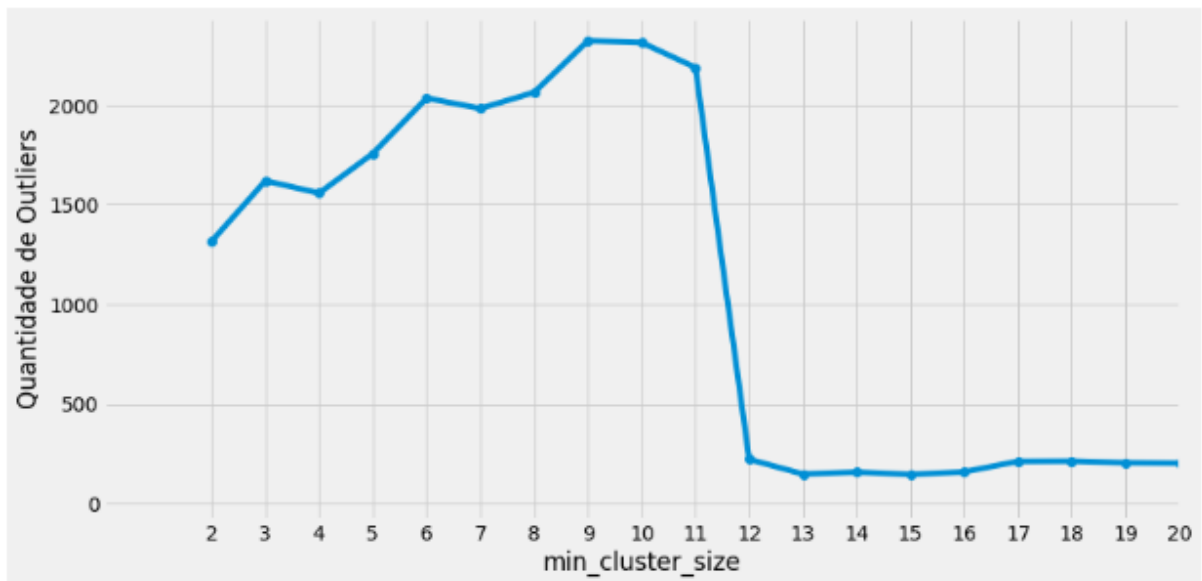


Figura 42. Quantidade de outliers para diferentes valores de *min_cluster_size*

Optou-se então por aplicar o algoritmo com o *min_cluster_size* igual a 13, que indicava menos outliers e uma quantidade estável de clusters. Nessa configuração, o HDBSCAN encontrou 8 clusters, conforme a distribuição da figura 43, e os outliers destacados na figura 44.

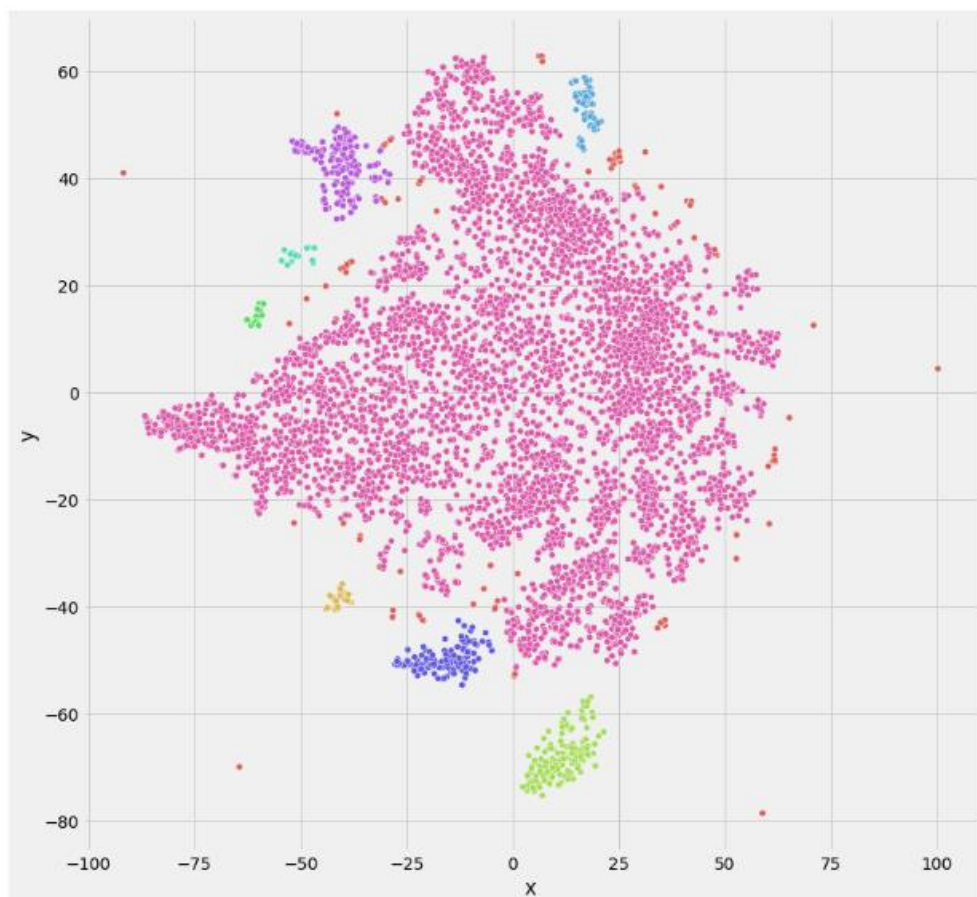


Figura 43. HDBSCAN sobre a base t- SNE

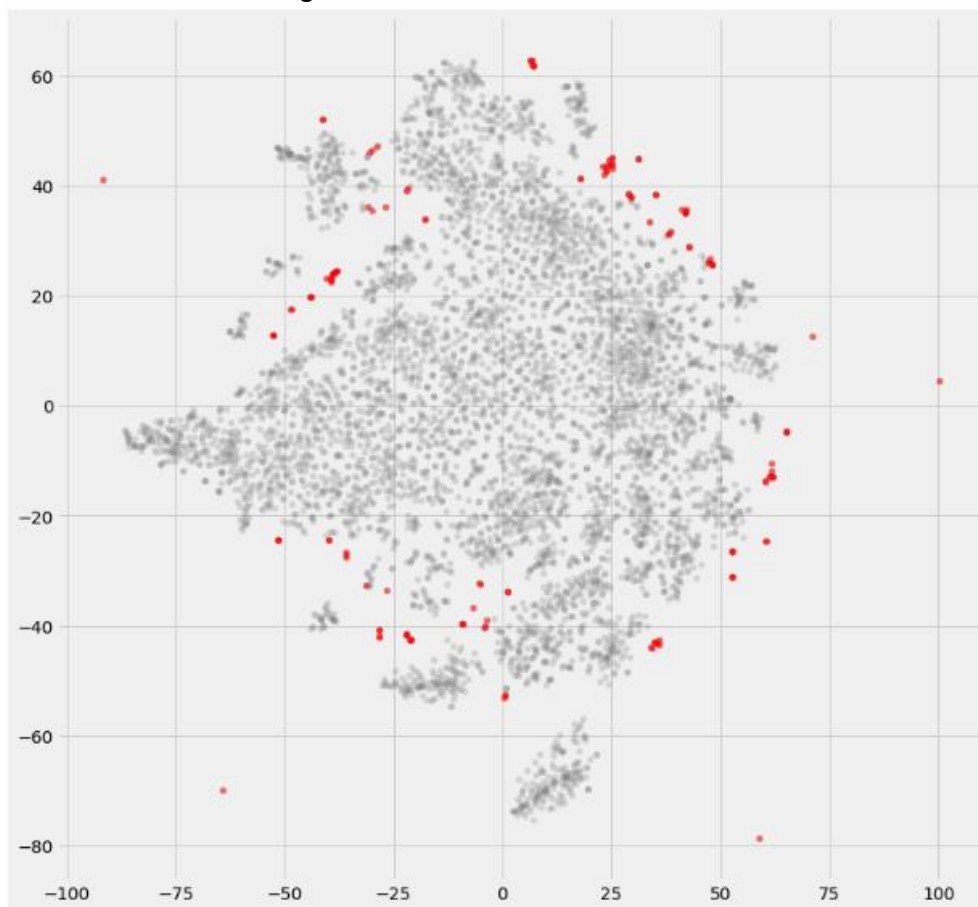


Figura 44. Outliers identificados

6. Apresentação dos Resultados

Uma característica importante das folhas de pagamento são os tipos de ocupações possíveis para os empregados municipais, que na base analisada chegaram a 2.567, incluídos os funcionários sem classificação. Ademais, devem ser consideradas cada uma das ocupações, principalmente porque a presença ou não de determinado profissional entre os funcionários indicam características peculiares daquele Município. Como exemplo, a cidade de Imperatriz/MA possui em sua folha de pagamentos a profissão de Salva-Vidas (CBO 5-89.50), cuja contratação é temporária por cerca de três meses ao ano, para o período em que no Rio Tocantins, que margeia a cidade, surgem bancos de areia que são transformados em praias de água doce. Esses profissionais são contratados para evitar afogamentos nesse período. Tendo essas características peculiares em mente, buscou-se analisar os dados das folhas dos Municípios com base na existência e na quantidade de ocupações

indicadas, na busca de um perfil ou de perfis, sem desconsiderar nenhuma das ocupações, ou seja, considerando todas as 2.567 dimensões do conjunto de dados.

Os serviços prestados pelos Municípios tendem a ser os mesmos, independentemente da sua localização ou situação econômico-social. Assim, em uma análise prévia, não deveriam existir muitas ou grandes distinções entre as folhas de pagamento dos Entes municipais, apenas diferenças pontuais, como aquela indicada no parágrafo anterior. O que se buscou com o presente trabalho foi identificar e agrupar os Municípios pelas características de suas folhas, assumindo-se a complexidade e impossibilidade de tal tarefa pela simples análise direta das folhas. Buscou-se auxílio de ferramentas de aprendizado não-supervisionado (de clusterização) para se obter indícios de perfis e características peculiares da estrutura funcional dos municípios brasileiros.

Inicialmente, adotou-se a abordagem mais básica, de se tentar agrupar os Municípios pela aplicação do algoritmo K-Means à base original, sem normalização ou redução de dimensionalidade.

O Método do Cotovelo e a análise de coeficientes *Silhouette* indicam 3 como a quantidade ideal de clusters. Aplicado o algoritmo, obteve-se poucos dados relevantes, uma vez que um cluster ficou apenas com o Município de São Paulo, o de maior folha de pagamentos, outro ficou com 9 capitais: Salvador, Fortaleza, Brasília, Goiânia, Belo Horizonte, Curitiba, Recife, Rio de Janeiro e Porto Alegre, e o terceiro com os 5.560 demais Municípios.

Tentou-se então a normalização dos dados utilizando o StandardScaler do SKLearn para tal tarefa. Uma vez normalizado, aplicou-se o Método do Cotovelo para se obter a quantidade ideal de clusters, que foi indicado como 4, enquanto a análise dos coeficientes *Silhouette* indicou valor 3. Aplicou-se o K-Means para 4 clusters, obtendo-se resultado similar ao teste anterior: um cluster com 5.558 municípios, um segundo apenas com o São Paulo, o terceiro com apenas o Rio de Janeiro e o quarto com 10 grandes cidades: Manaus, Salvador, Fortaleza, Brasília, Goiânia, Belo Horizonte, Curitiba, Recife, Porto Alegre e Campinas. Pouca informação pôde ser extraída dessa análise, porque, mais uma vez, o algoritmo foi guiado pela quantidade total de empregados municipais.

Tentou-se então reduzir a dimensão dos dados. Para isso aplicou-se o algoritmo t-SNE sobre a base original, não normalizada. Esse redimensionamento da

base produziu resultado interessante, sendo capaz de indicar agrupamentos de Municípios mais coesos, bem como alguns mais distantes.

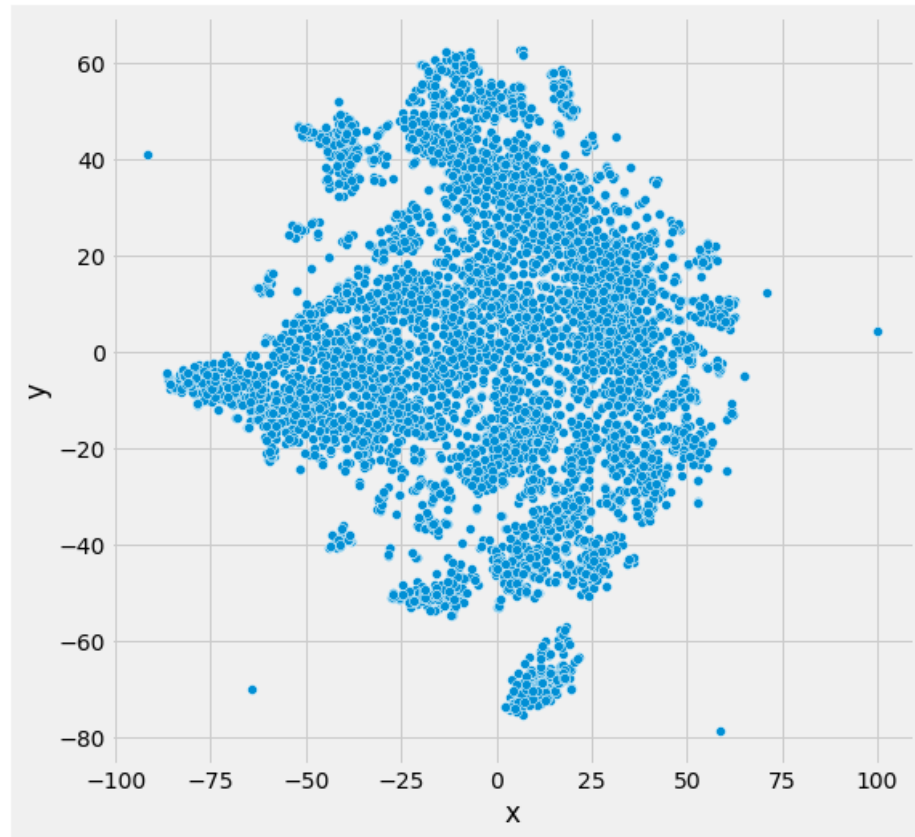


Figura 45. Base de dados redimensionada com o t-SNE

Aplicando o Método do cotovelo, chegou-se a 4 como o número ideal de clusters. Com essa quantidade, a distribuição de Municípios ficou mais homogênea, sem cluster com apenas um Município ou cluster com milhares, como pode ser visto no gráfico da figura 46.

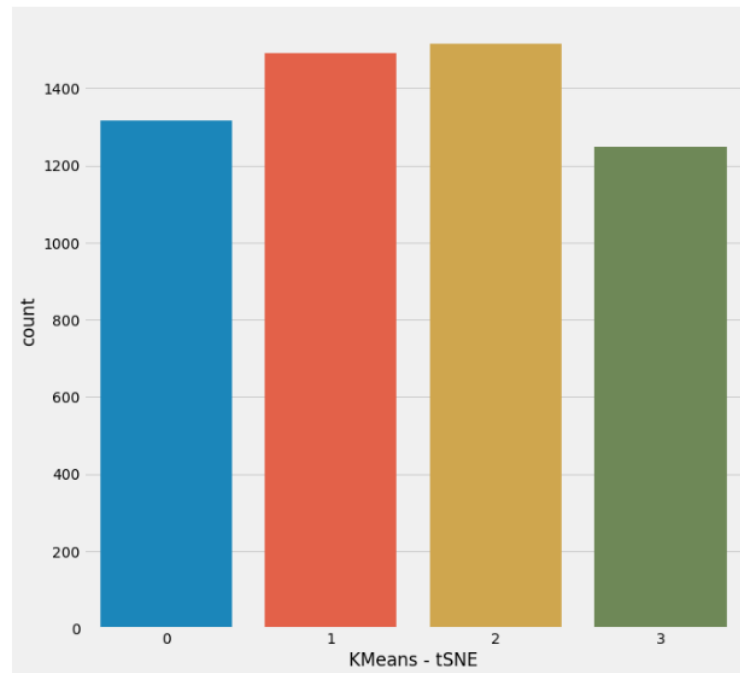


Figura 46. Distribuição de Municípios entre os clusters - base t-SNE

Entretanto, pela inspeção visual do gráfico, ainda se identifica clusters pouco coesos, apesar de o algoritmo ter conseguido delimitá-los satisfatoriamente.

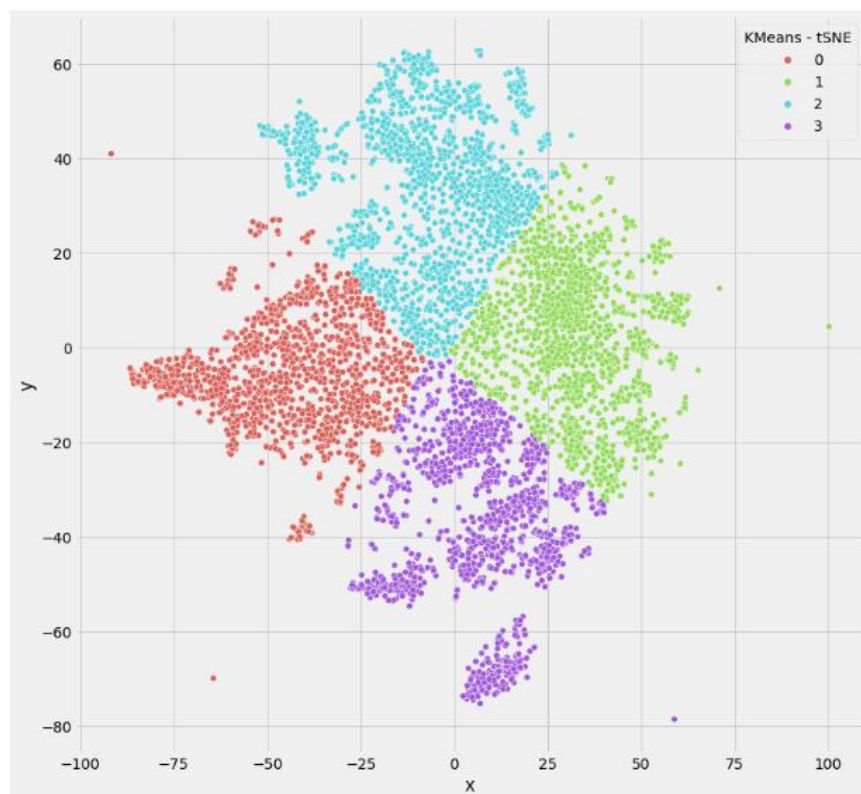


Figura 47. Distribuição dos elementos em cluster após aplicação do K-Means sobre a base redimensionada

Constata-se pela análise do gráfico que existem diversos pequenos agrupamentos, mais coesos que os agrupamentos maiores. Testou-se então rodar o K-Means com quantidades maiores de clusters. Para estimar as quantidades a serem testadas, usou-se o coeficiente *Silhouette*, que indicou também 14 e 35 clusters. Ainda que com coeficientes ruins, próximos de 0 (o que indica proximidade a elementos de outros clusters), espera-se destacar os pequenos agrupamentos que se distanciam do agrupamento central na tentativa de identificar características peculiares.

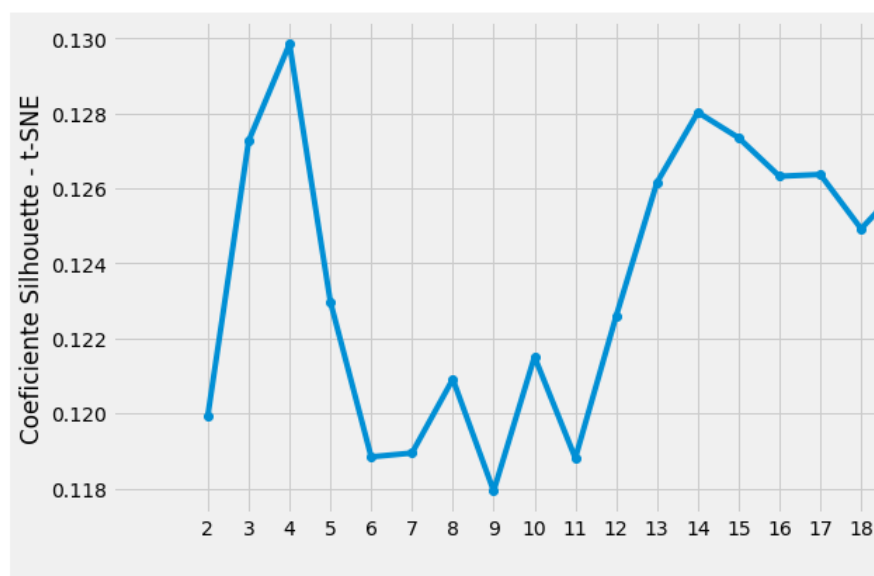


Figura 48. Coeficiente Silhouette indicando 14 clusters

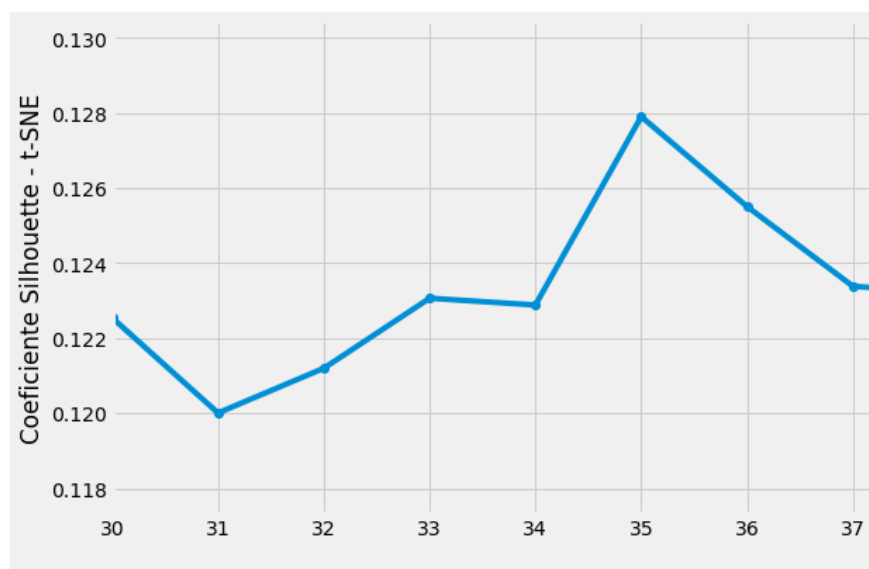


Figura 49. Coeficiente Silhouette indicando 35 clusters

Com 14 clusters, foram destacados e melhor delimitados agrupamentos visualmente mais coesos.

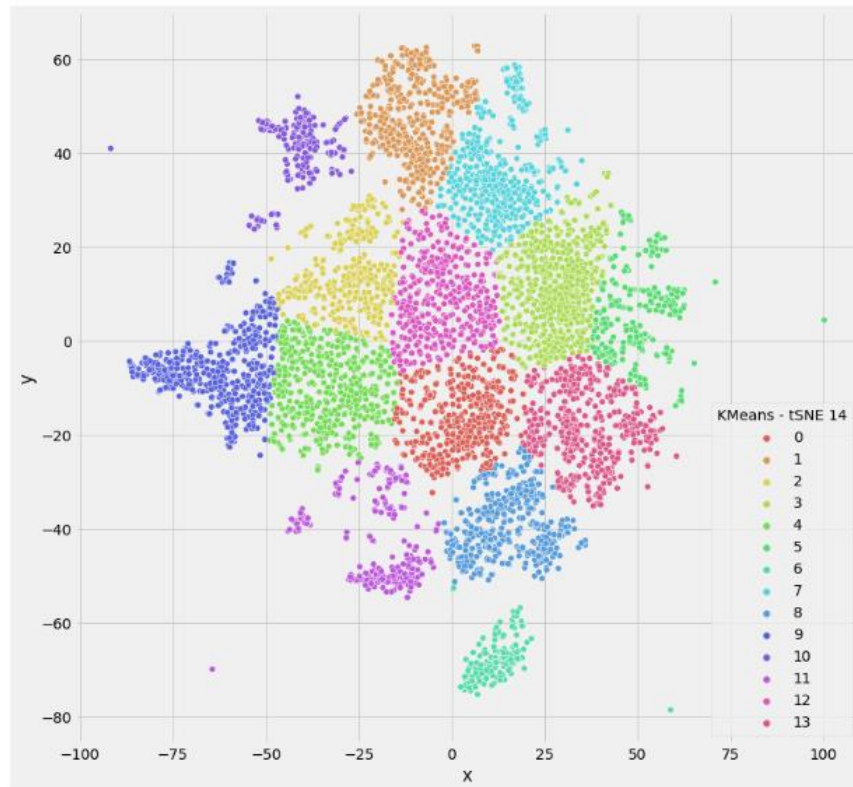


Figura 50. Base redimensionada com K-Means de 14 clusters

Com 35 clusters o algoritmo identificou agrupamentos mais marginais bem como grupos ainda mais coesos (figura 51).

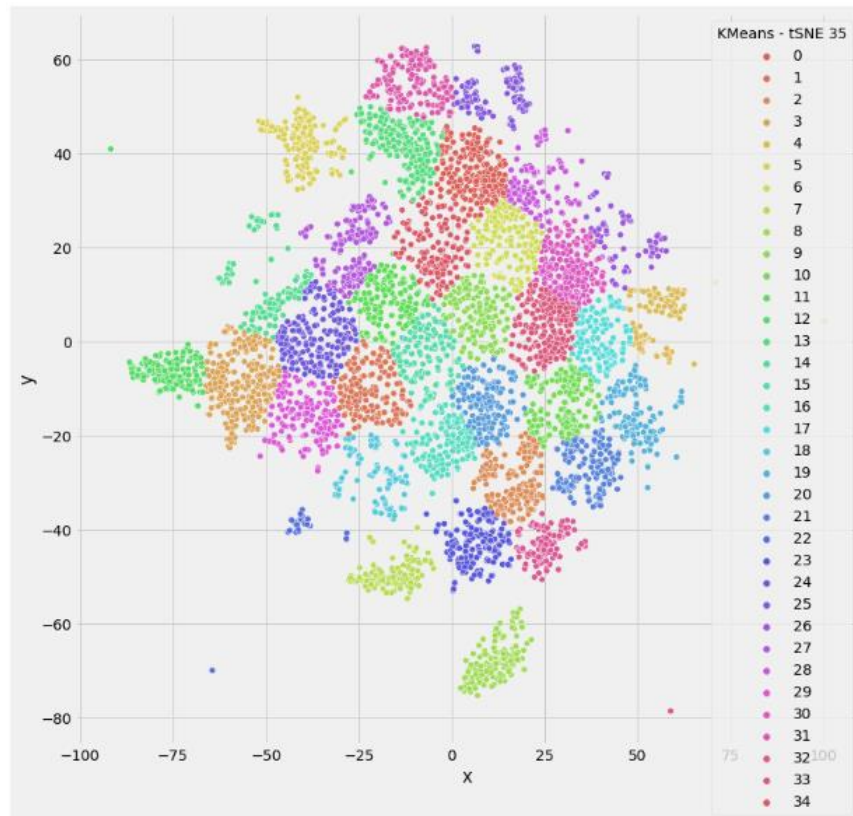


Figura 51. Base redimensionada com K-Means de 35 clusters

Por fim, buscou-se um algoritmo de clusterização alternativo ao K-Means, em que não fosse necessário indicar a quantidade de clusters previamente e que também indicasse outliers. Escolheu-se para tal tarefa o HDBSCAN, que foi aplicado sobre os dados reduzidos com algoritmo t-SNE.

O único parâmetro informado foi o tamanho mínimo do cluster (*min_cluster_size*). Estimou-se então as quantidades possíveis de tamanho de cluster e de quantidade de outliers, com valores estabilizando a partir do *min_cluster_size* = 13. Com esse valor, foram gerados 8 clusters e identificados 147 outliers (cluster -1). Um dos clusters reuniu a maior parte dos Municípios, 4.810, e os demais foram formados pelos pequenos agrupamentos mais marginais.

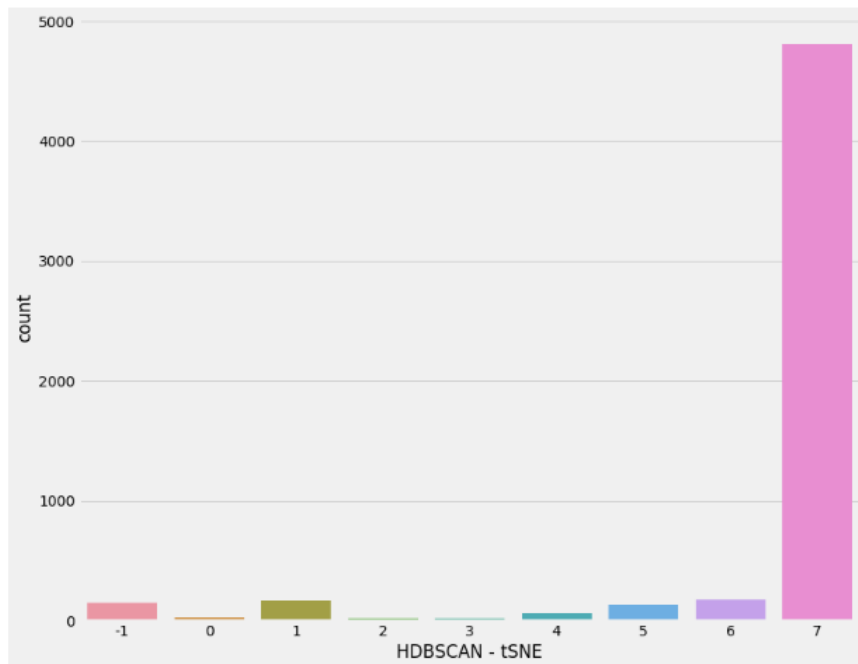


Figura 52. Resultado do HDBSCAN com *min_cluster_size* igual a 13

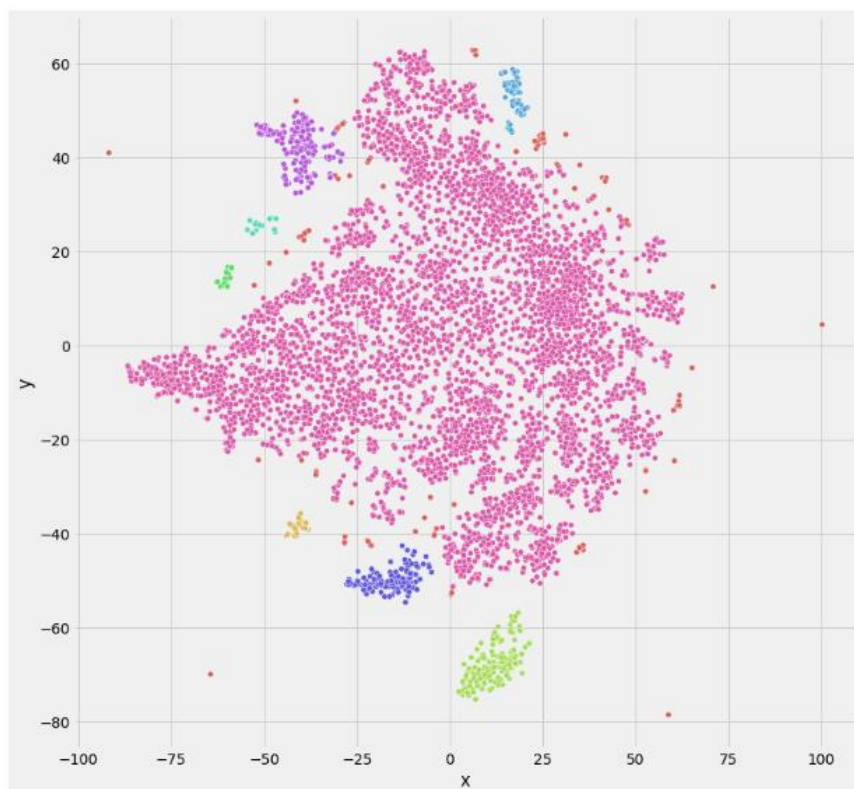


Figura 53. Gráfico com o HDBSCAN com *min_cluster_size* igual a 13

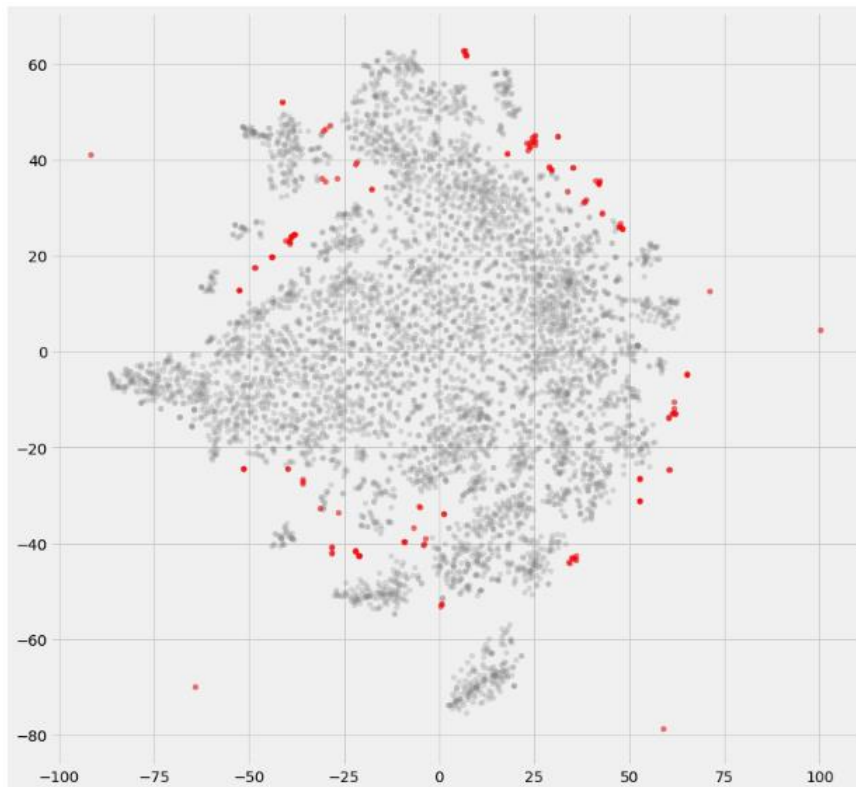


Figura 54. Outliers identificados pelo HDBSCAN

Concluídos os experimentos, selecionou-se os modelos que, visualmente, melhor separaram os grupos de Municípios e, teoricamente, foram capazes de identificar semelhanças e diferenças, e melhor conseguiriam indicar tendências em uma análise puramente visual: K-Means nos dados com redução de dimensão usando o t-SNE e com 4 clusters e HDSCAN nos mesmos dados redimensionados e escolha do *min_cluster_size* de 13, que indicou 8 agrupamentos e 147 outliers. Essa abordagem foi uma forma de contornar a dificuldade de se analisar resultados obtidos em aprendizados não supervisionados de uma base com muitas dimensões, quando não é claro quais e como cada uma das *features*/ocupações influenciaram no resultado alcançado.

Buscou-se com tal abordagem confirmar se os algoritmos aplicados, t-SNE para o redimensionamento, e o K-Means e HDBSCAN para clusterização, foram capazes de identificar alguma tendência ou similaridades significantes. Assim, optou-se pela análise gráfica dos resultados, primeiro na determinação das quantidades de clusters, quando se buscou destacar os pequenos agrupamentos visualizáveis após o redimensionamento usando o algoritmo t-SNE e, por fim, na análise dos resultados

produzidos, através da plotagem dos pontos que representam os Municípios no mapa, e assim identificar características regionais.

Nessa análise procedeu-se à distribuição espacial dos agrupamentos identificados. Buscou-se dessa forma identificar regiões do país em que os Municípios pertençam aos mesmos agrupamentos obtidos.

Para tal tarefa, a base de dados foi enriquecida com dados de longitude e latitude de cada Município, obtidas junto à Controladoria Geral da União. Os pontos foram plotados sobre a imagem do Brasil, baixada do OpenStreetsMap.

Analizou-se o resultado obtido pela aplicação do algoritmo K-Means sobre a base redimensionada com o algoritmo t-SNE e com indicação de 4 clusters, conforme sugerido pelo método do cotovelo.

Com 4 clusters, verificou-se que os agrupamentos apresentam concentrações regionais, com cada um dominando algumas regiões específicas do País.

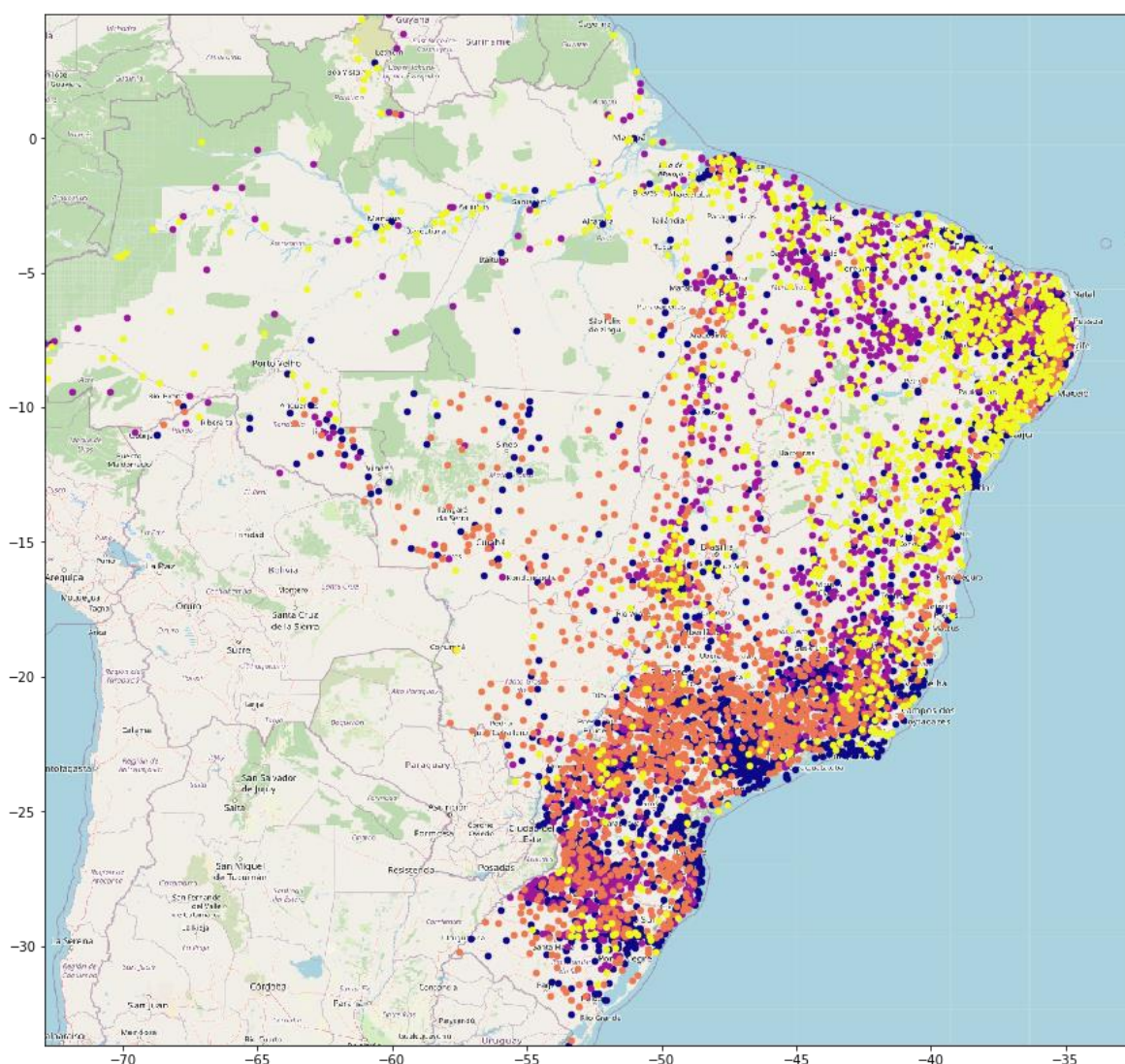


Figura 55. Distribuição espacial dos 4 clusters identificados pelo K-Means sobre a base redimensionada

O cluster 0 apresenta componentes espalhados por todas as regiões do país, mas possui concentração maior nas regiões Sudeste e Sul, nessa última, com vários Municípios da região litoral.

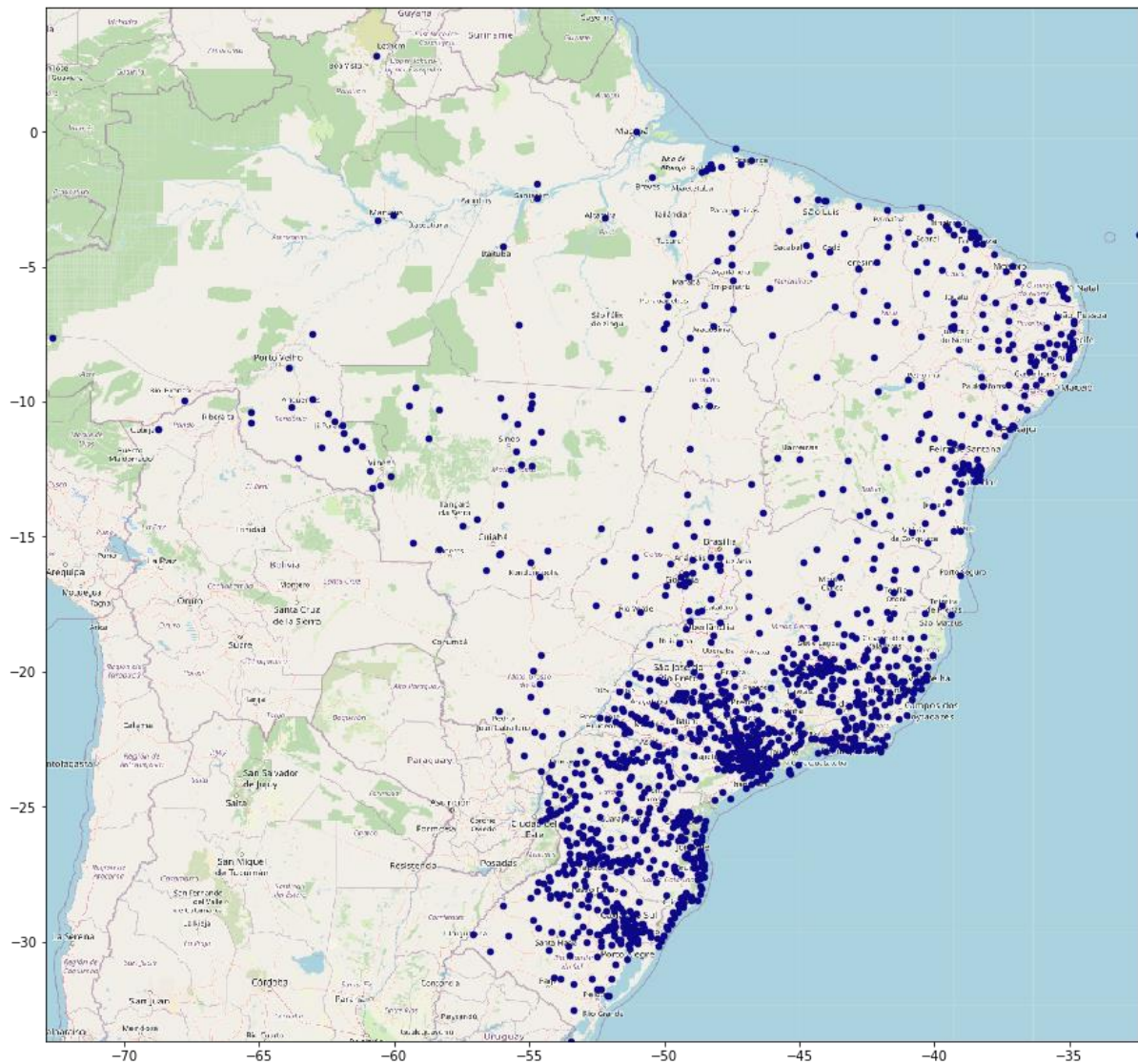


Figura 56. Cluster 0 - K-Means sobre base redimensionada com t-SNE

O cluster 1 também possui componentes em todas as regiões do país, entretanto também apresenta agrupamentos mais densos no oeste da região Sul e no leste da região Nordeste, sem incluir muitos Municípios mais litorâneos dessa última.

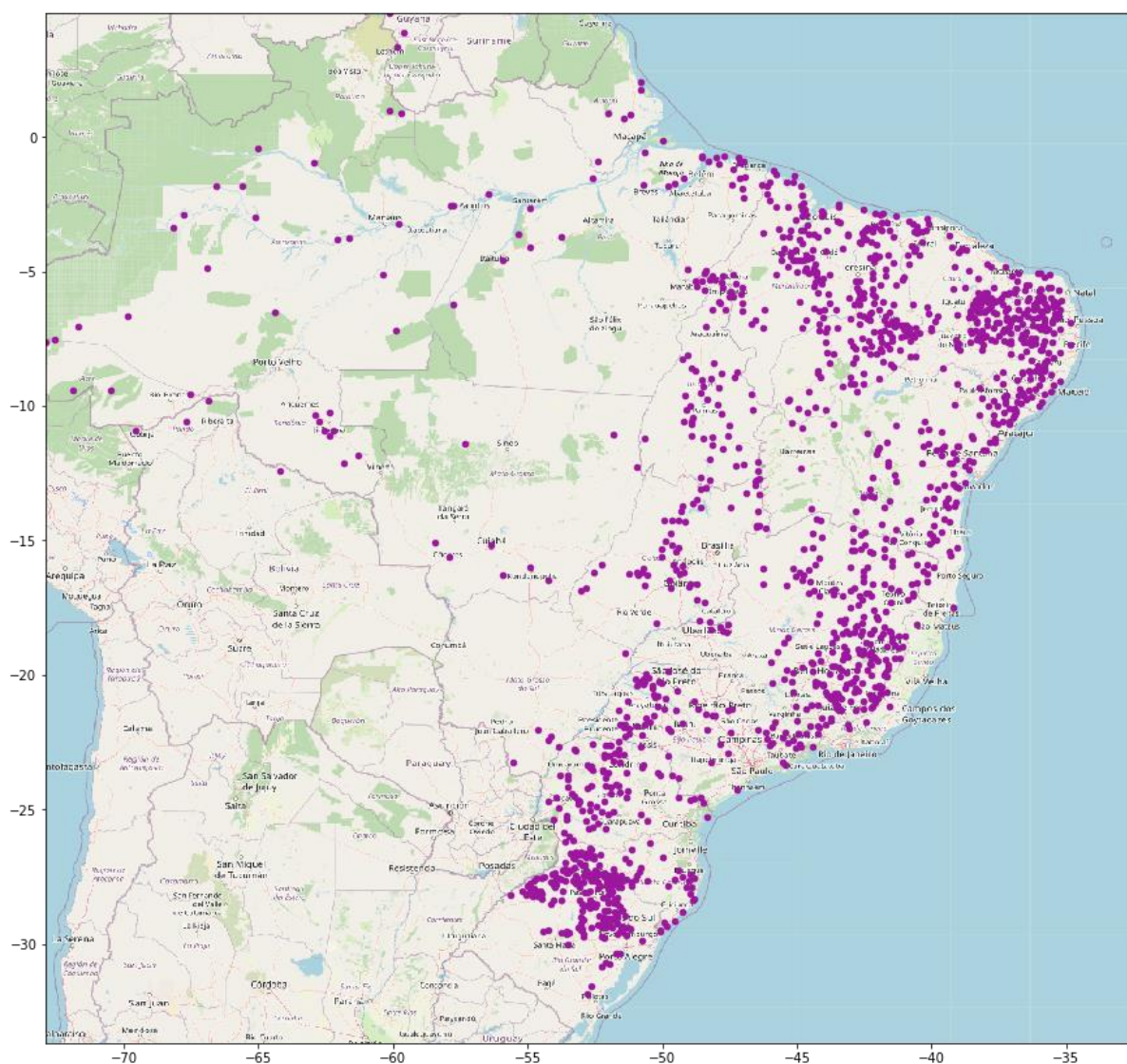


Figura 57. Cluster 1 - K-Means sobre base redimensionada com t-SNE

Os Municípios mais próximos do litoral ao leste do Nordeste aparecem agrupados no cluster 2. Este possui maior concentração de componentes na região Sudeste e Sul e possui poucos componentes da região Norte.

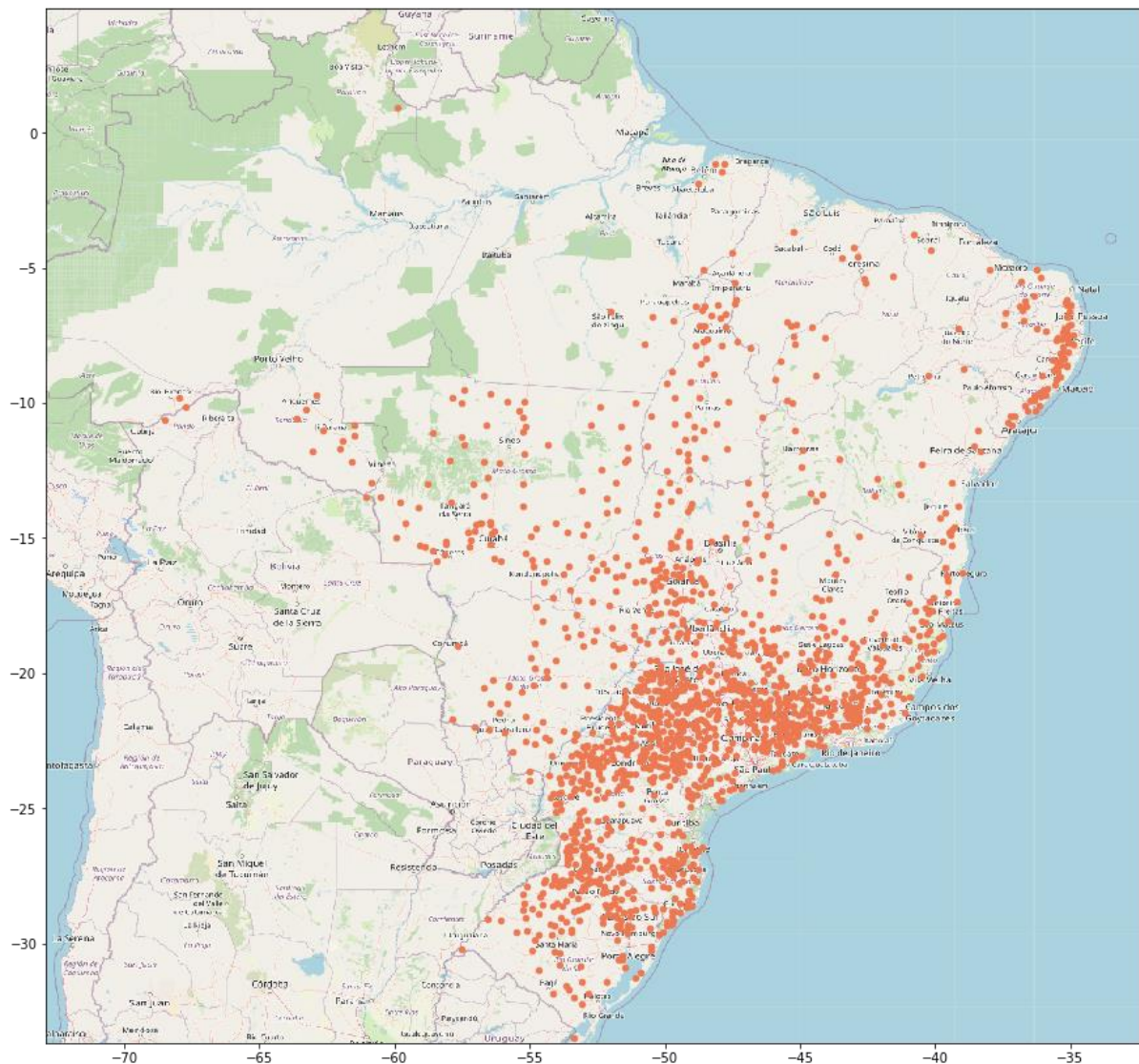


Figura 58. Cluster 2 - K-Means sobre base redimensionada com t-SNE

Por último, o cluster 3 apresenta mais componentes no leste da região Nordeste, bem como pequenas concentrações na região Norte, no litoral do Pará e em diversas cidades próximas ao Rio Amazonas.

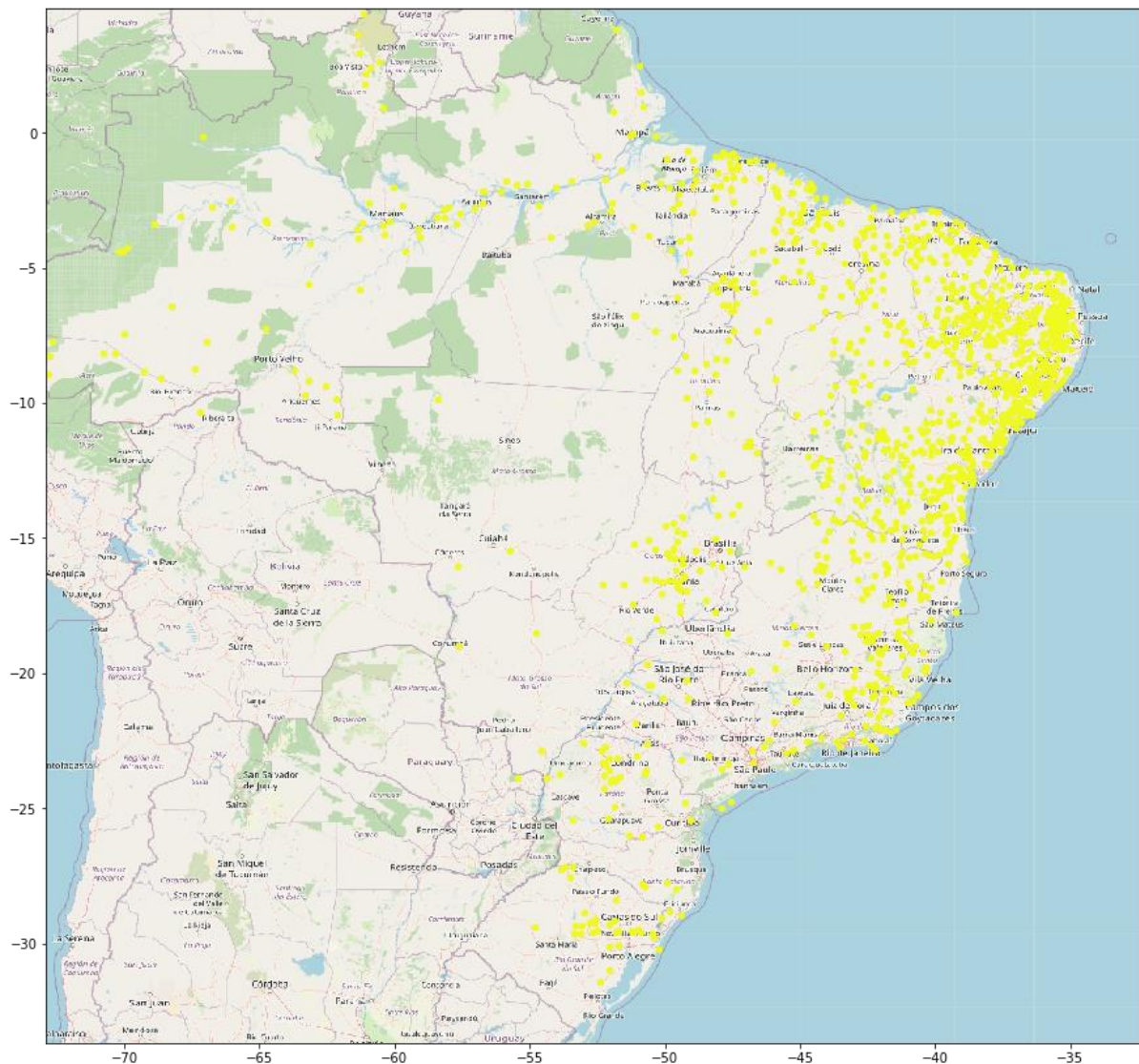


Figura 59. Cluster 3 - K-Means sobre base redimensionada com t-SNE

Concentrando-se a análise no Estado de Minas Gerais, também são identificadas, ao se comparar a distribuição espacial de cada cluster, concentrações regionais, áreas do Estado que possuem mais componentes de determinado cluster. Observa-se também que o Estado possui diversos representantes em cada um dos clusters.

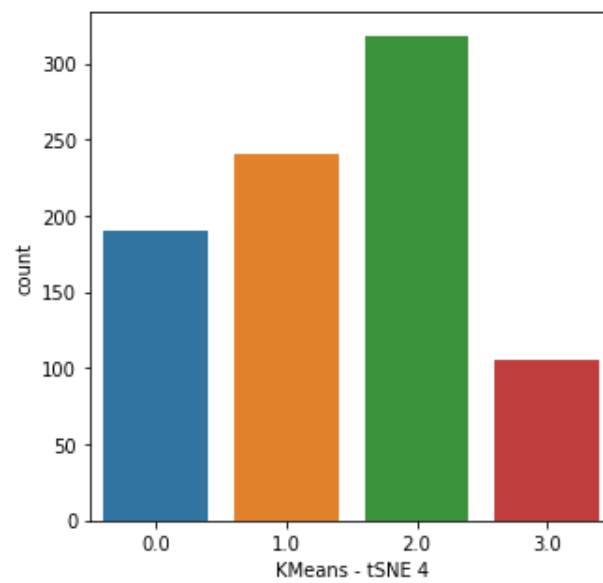


Figura 60. Contagem de componentes de Minas Gerais por cluster

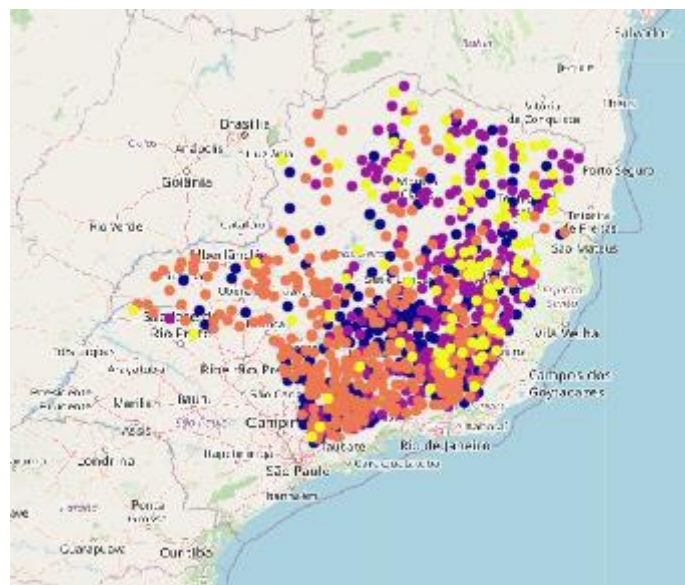


Figura 61. Distribuição espacial de municípios mineiros

Para Minas Gerais, o cluster 0 apresenta mais Município na região central e sudeste do Estado.



Figura 62. Cluster 0 - K-Means sobre base redimensionada com t-SNE - MG

Já o cluster 1 possui distribuição mais dispersa, mais ao leste do Estado.

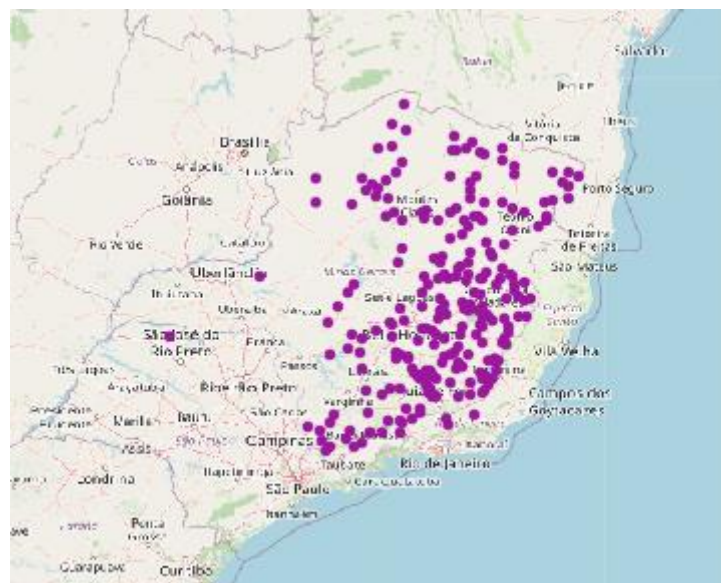


Figura 63. Cluster 1 - K-Means sobre base redimensionada com t-SNE - MG

O cluster 2, o que possui mais Municípios de Minas Gerais, possui maior concentração de Municípios na região sul do Estado.

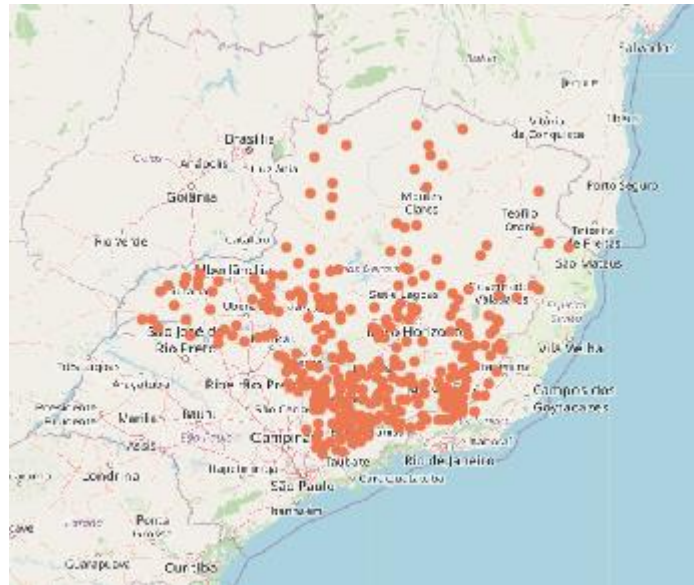


Figura 64. Cluster 2 - K-Means sobre base redimensionada com t-SNE - MG

Por fim, o menor cluster, de número 3, apresenta pontos mais dispersos, sem grandes concentrações.



Figura 65. Cluster 3 - K-Means sobre base redimensionada com t-SNE - MG

A análise foi concluída com a plotagem do resultado obtido com o HDBSCAN com *min_cluster_size* de 13, que identificou 8 cluster e 147 outliers.

Como já comentado, um cluster, o de número 7 concentrou a maior parte os Municípios, com 4.810 componentes. Exibindo-os no mapa, não se identifica pontos de destaque ou agrupamentos mais densos.

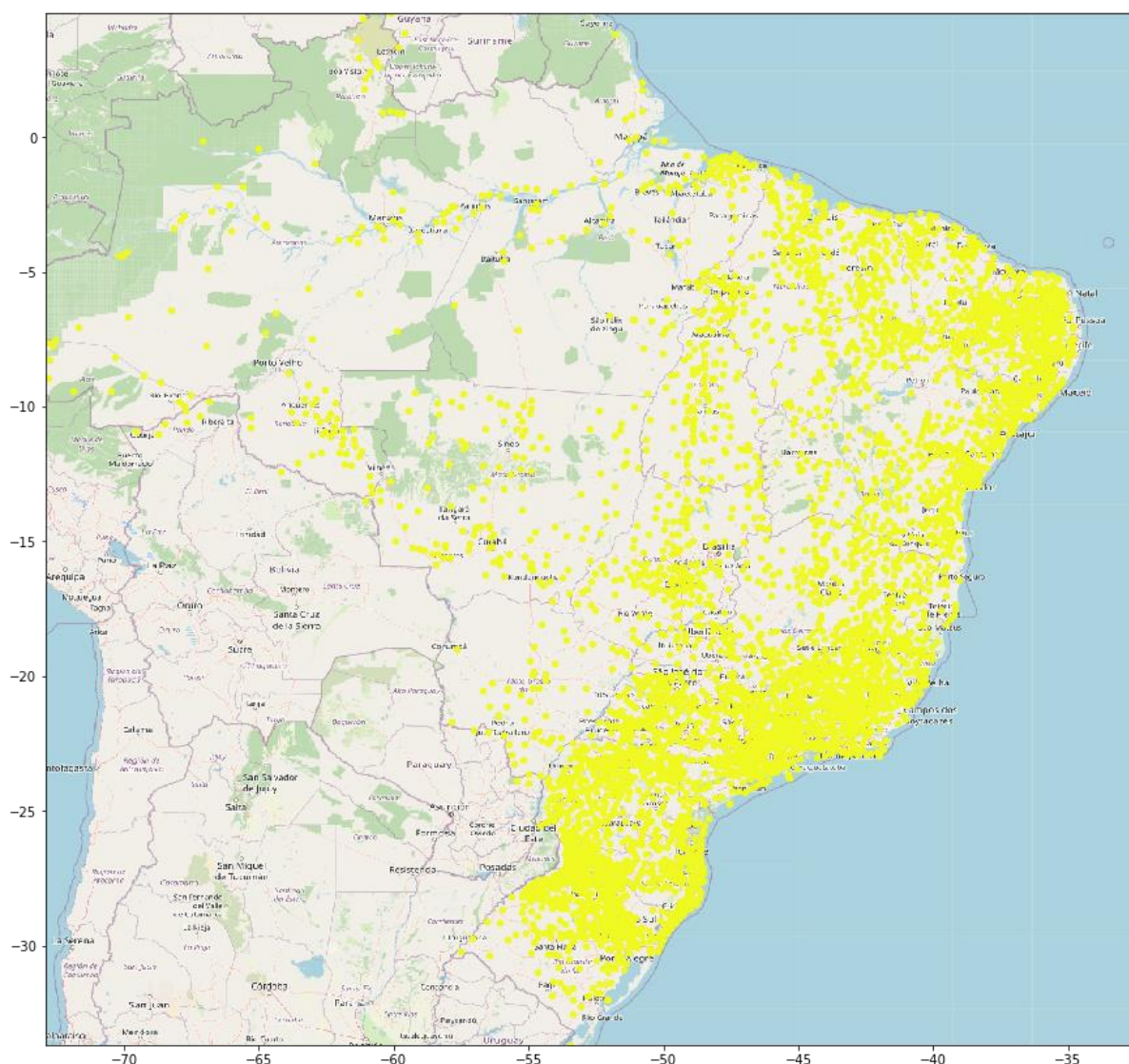


Figura 66. Cluster 7 - HDBSCAN sobre base redimensionada com t-SNE

Entretanto, os demais clusters apresentaram concentrações de Municípios. O cluster 6, com 176 componentes, a exemplo do que foi identificado no cluster 2 pelo algoritmo K-Means (figura 56) apresentou duas concentrações regionais de Municípios. Uma no litoral leste do Nordeste e outra no oeste de São Paulo.

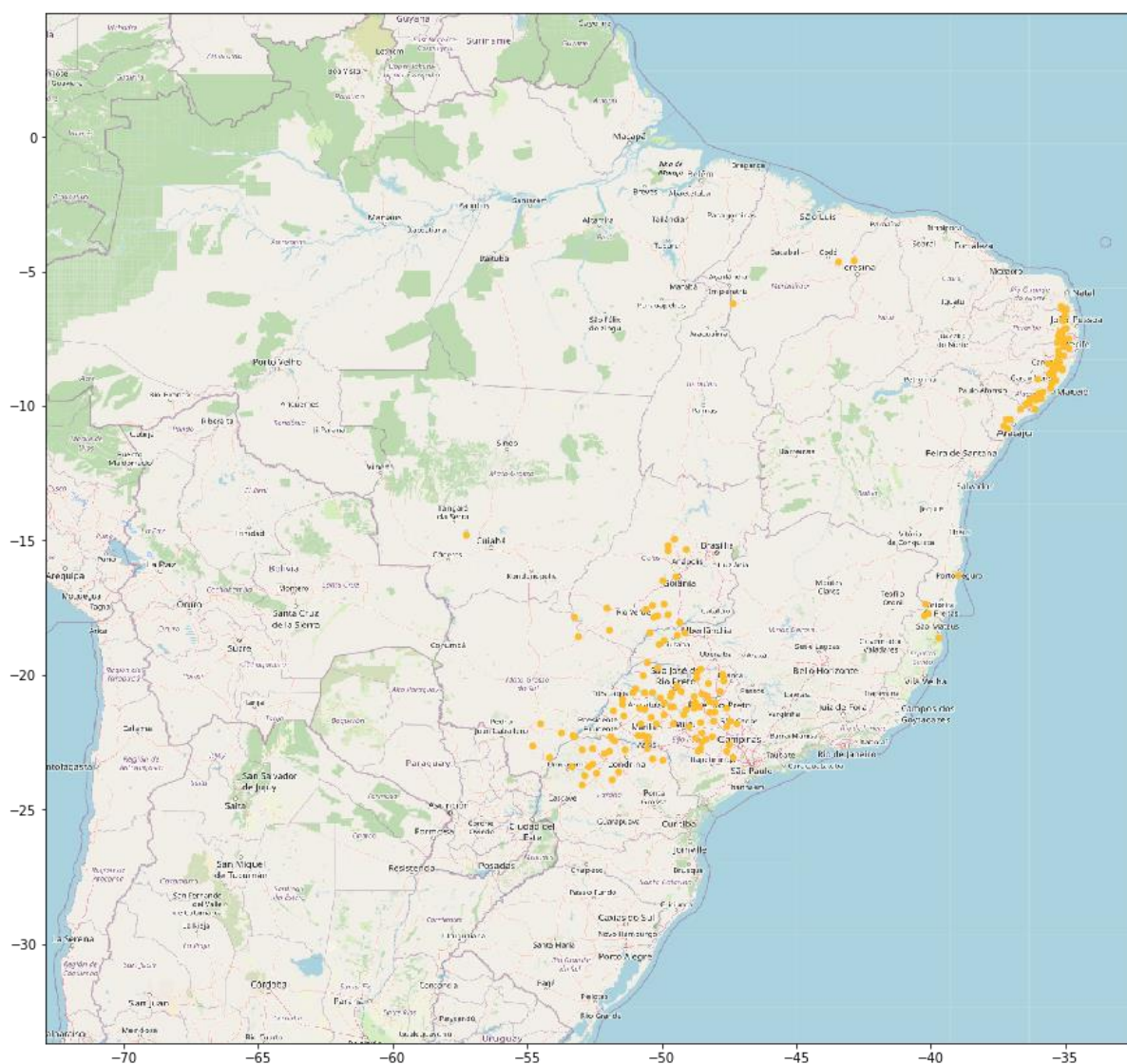


Figura 67. Cluster 6 - HDBSCAN sobre base redimensionada com t-SNE

O cluster 5, com 134 componentes, também concentra boa parte de seus componentes no leste da região Nordeste, mas com diversos pontos espalhados pelo país.

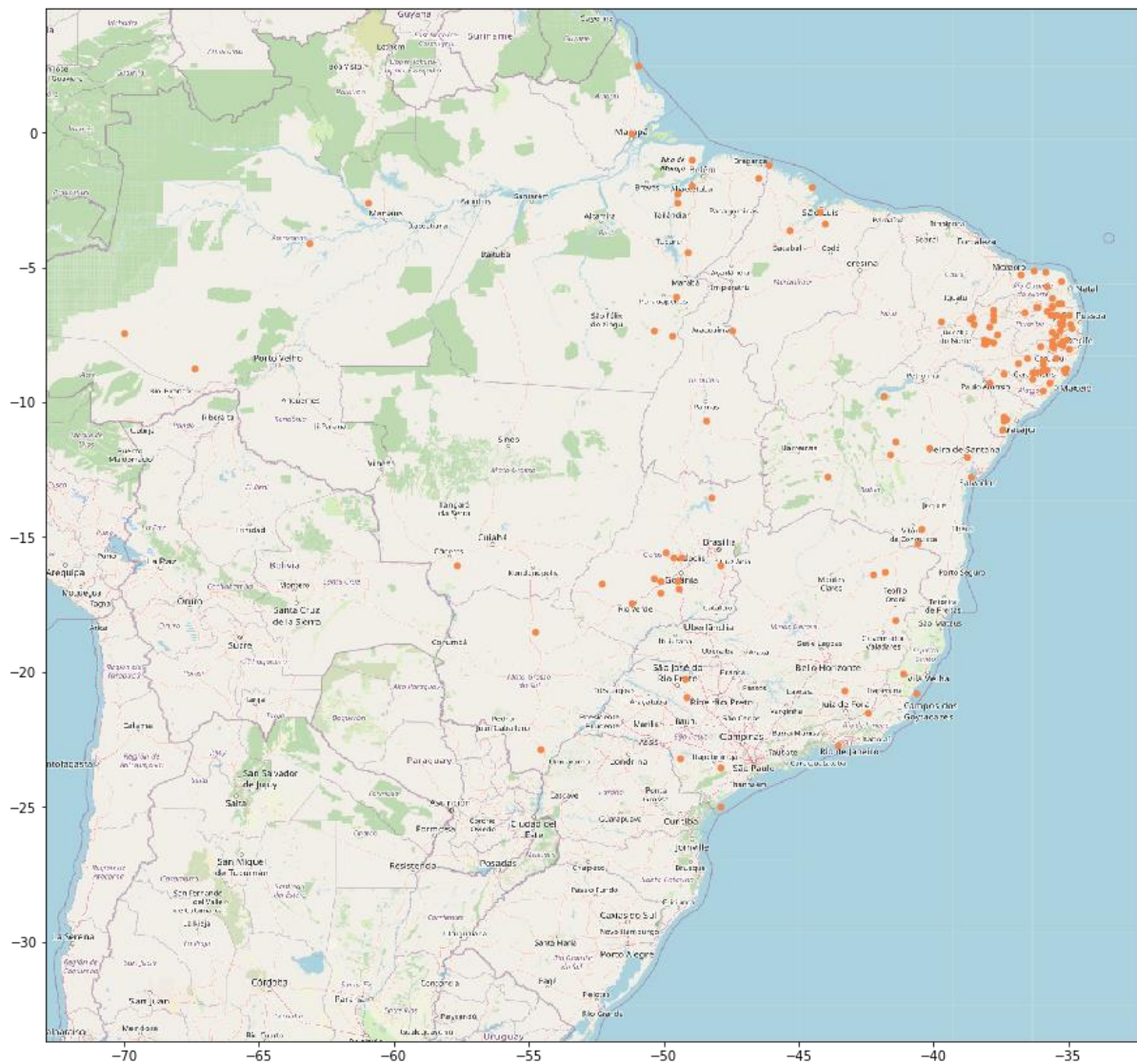


Figura 68. Cluster 5 - HDBSCAN sobre base redimensionada com t-SNE

O cluster 4 por sua vez concentra a maioria de seus 60 componentes na região sudeste, em especial no Estado de São Paulo.

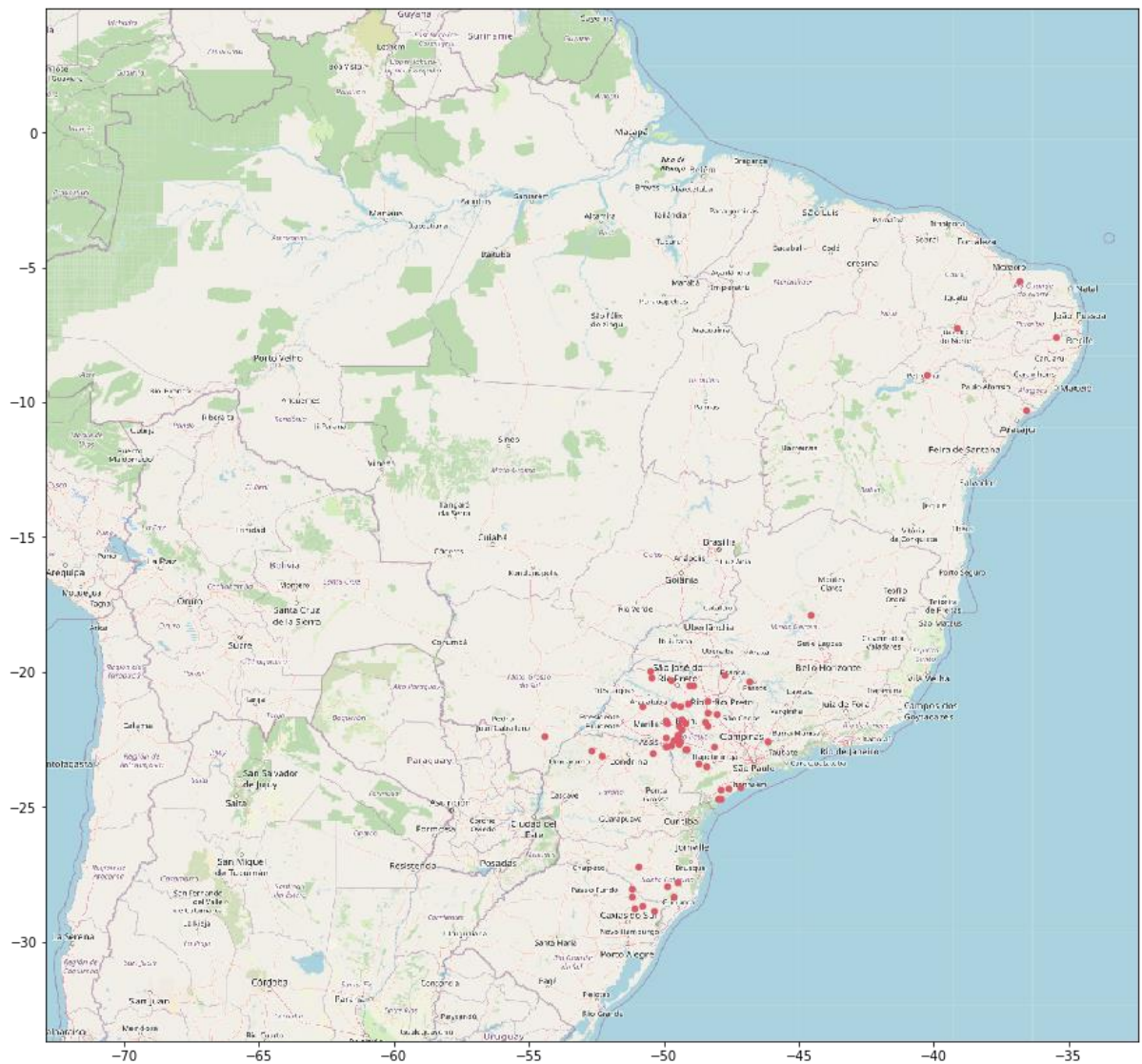


Figura 69. Cluster 4 - HDBSCAN sobre base redimensionada com t-SNE

O cluster 3, com 23 componentes, segue padrão parecido com o de número 5, com concentração de componentes no Estado de São Paulo.

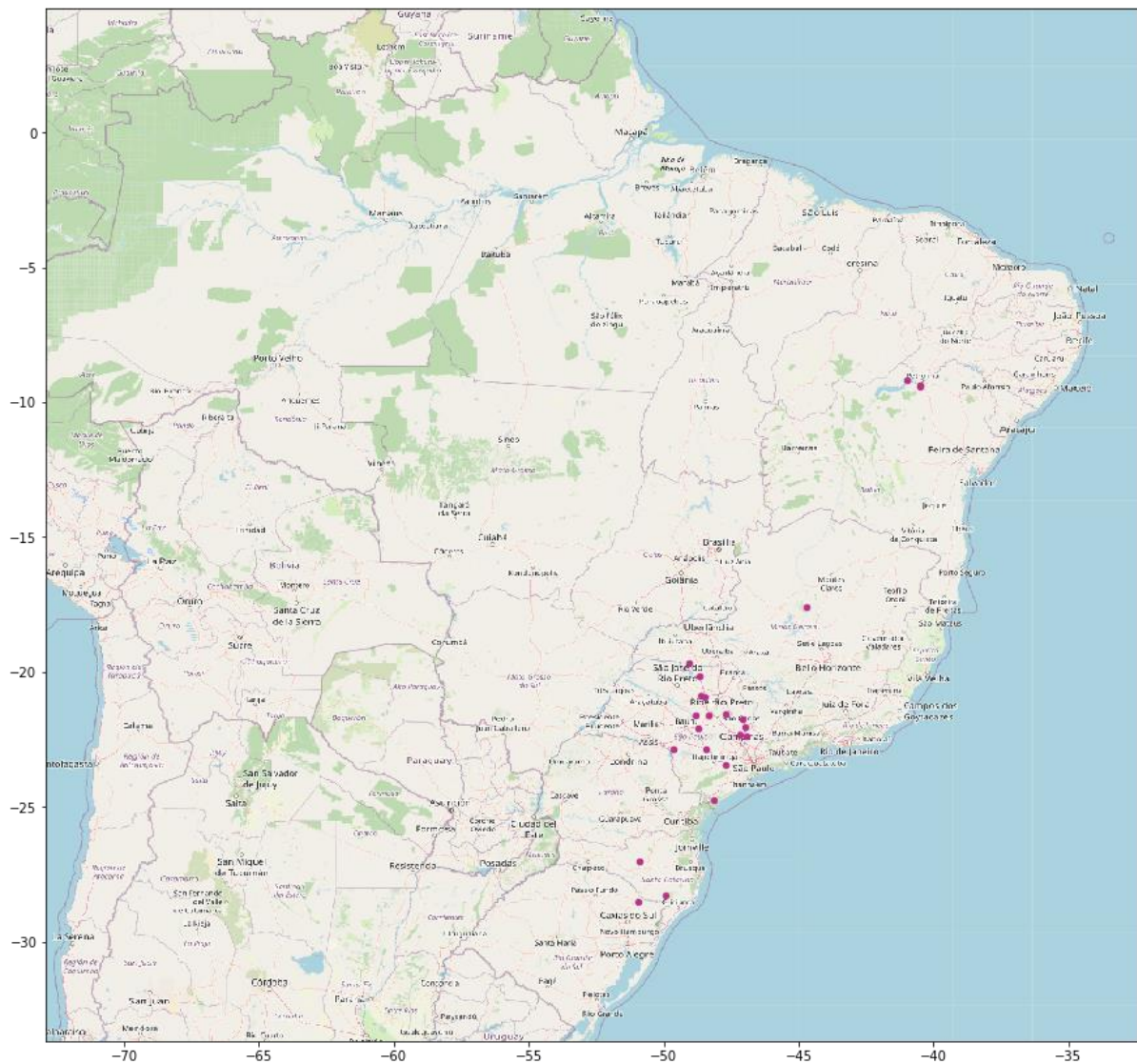


Figura 70. Cluster 3 - HDBSCAN sobre base redimensionada com t-SNE

O cluster 2 possui 23 componentes, a grande maioria concentrada na parte oeste da região Sul.

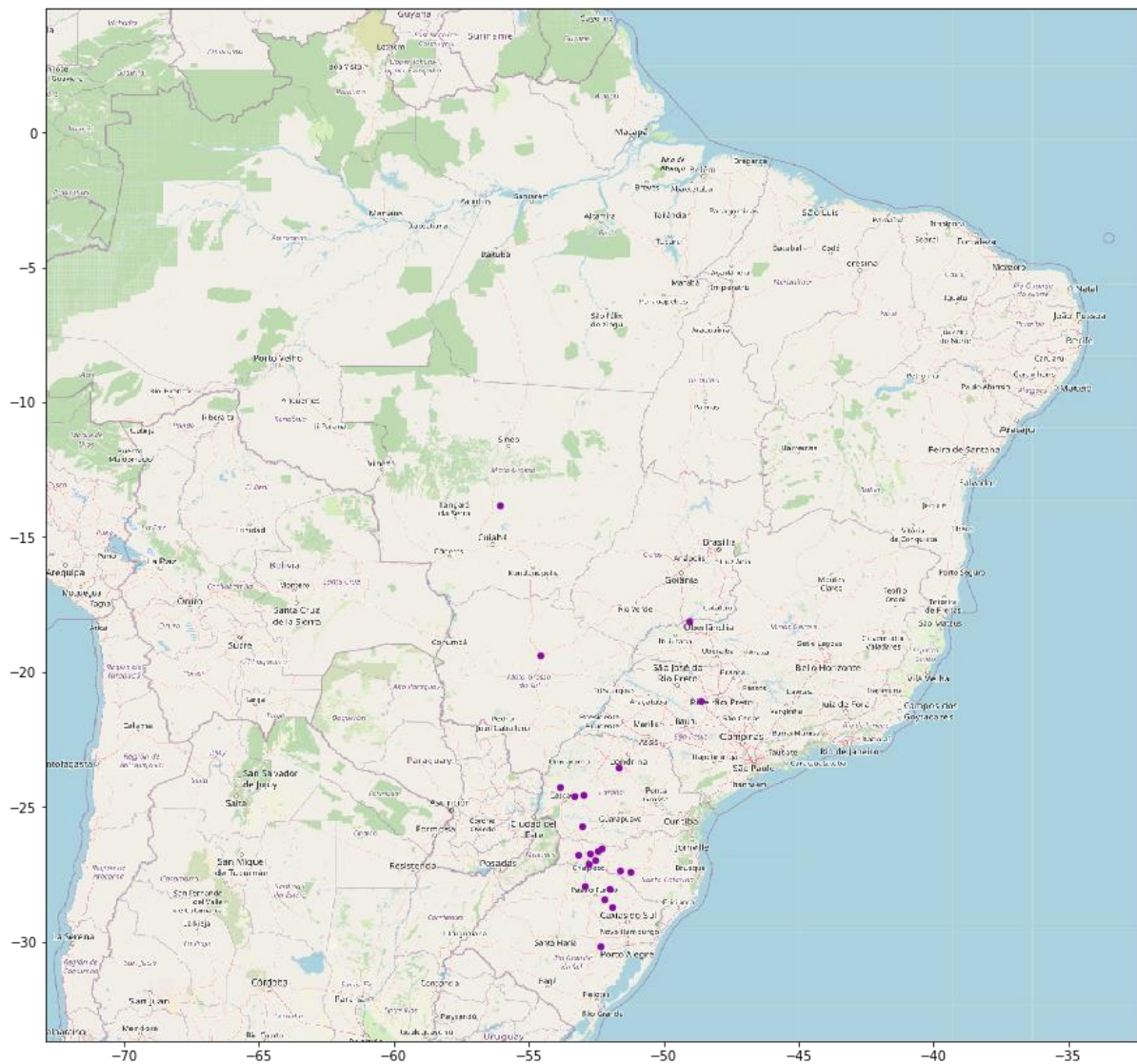


Figura 71. Cluster 2 - HDBSCAN sobre base redimensionada com t-SNE

O cluster 1 apresenta maior quantidade de seus 166 componentes dispersos, com algumas concentrações, sendo uma no Rio de Janeiro e outra em Sergipe.

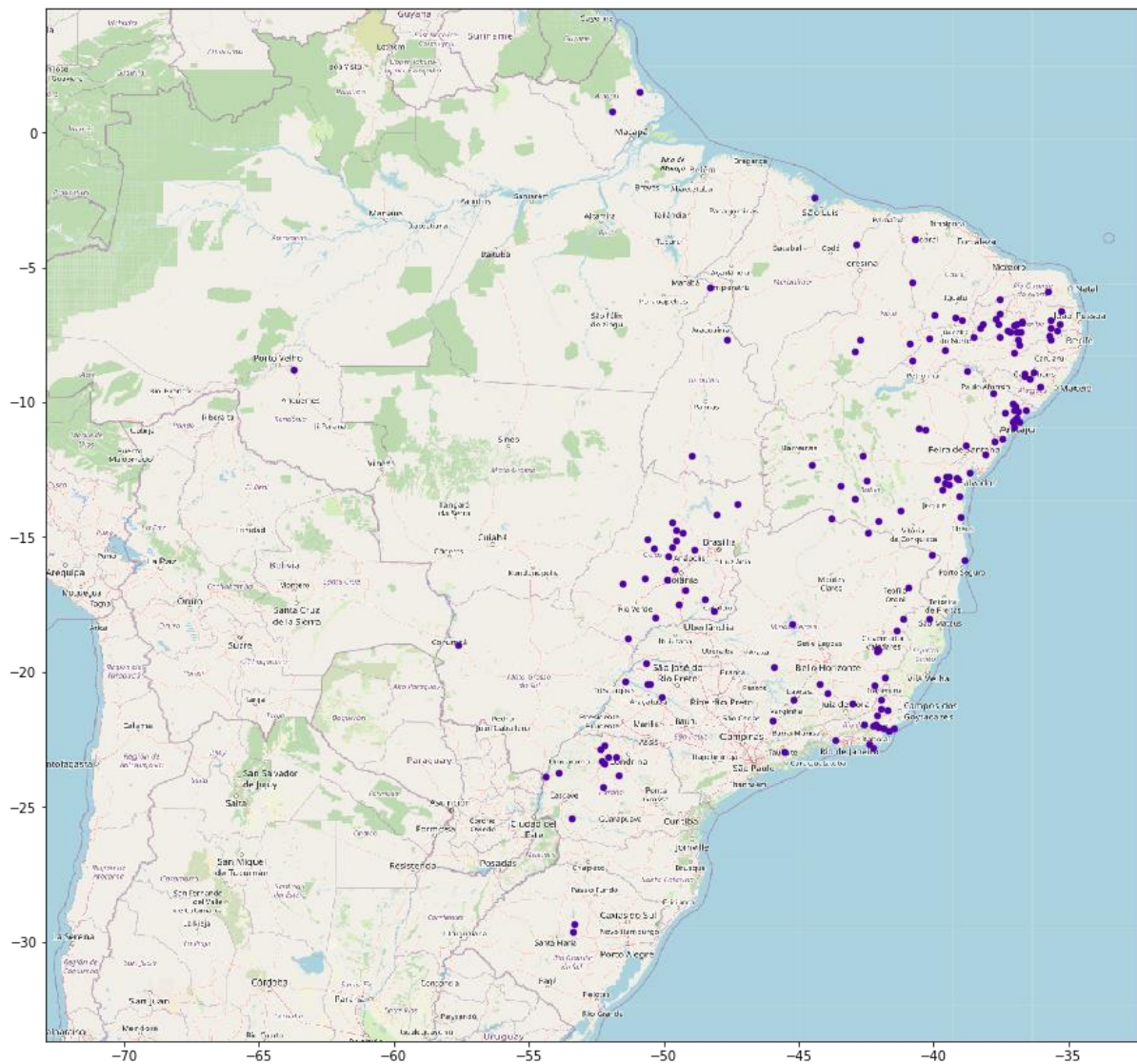


Figura 72. Cluster 1 - HDBSCAN sobre base redimensionada com t-SNE

O cluster 0, com 31 elementos, mas ainda com algumas concentrações, uma no Estado do Ceará e outra no Rio Grande do Sul.

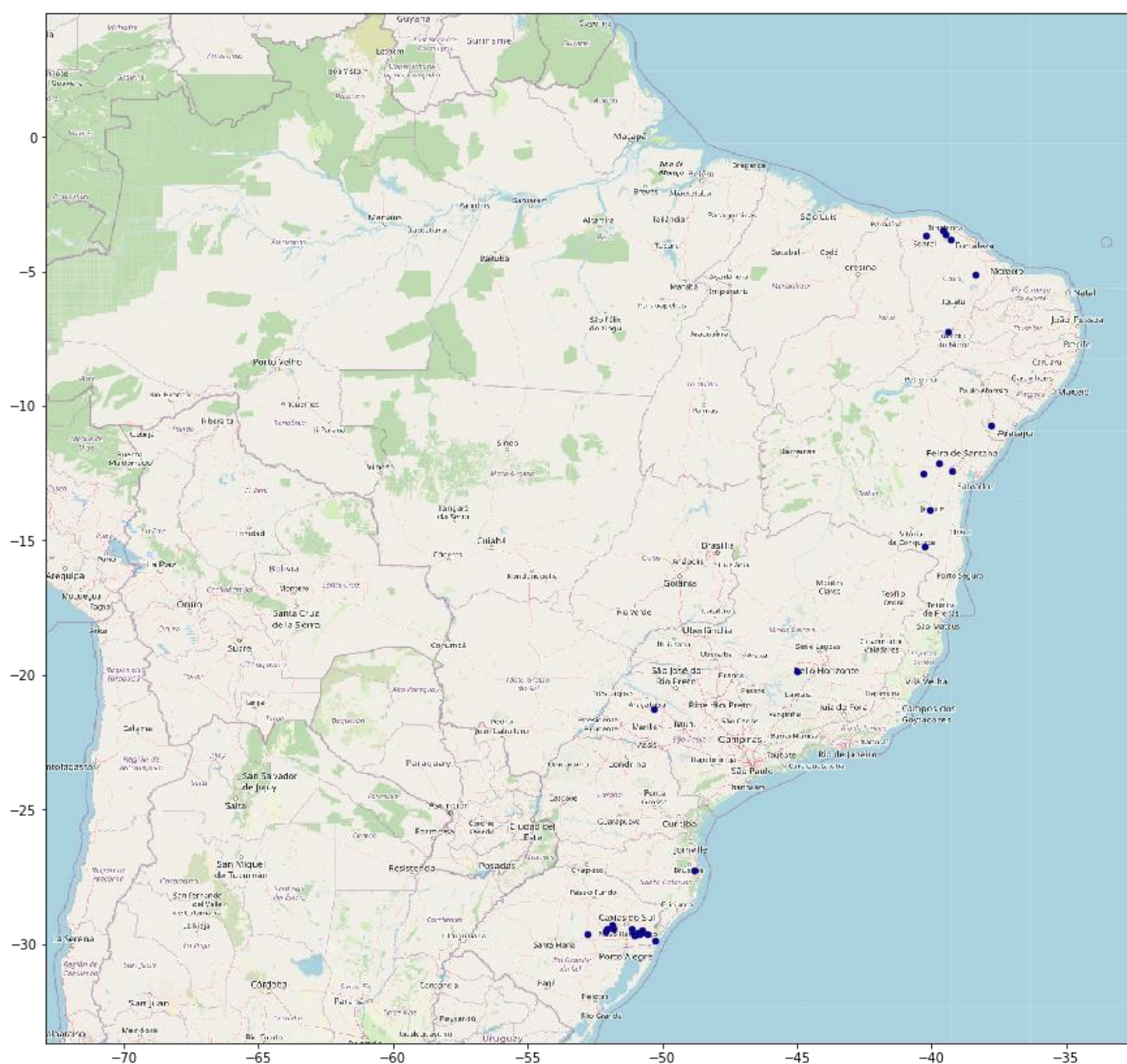


Figura 73. Cluster 0 - HDBSCAN sobre base redimensionada com t-SNE

Por fim, no mapa com a indicação dos 147 outliers identificados, nota-se grande distribuição de casos pelos País, com apenas uma concentração significativa no sul do Estado do Espírito Santo.

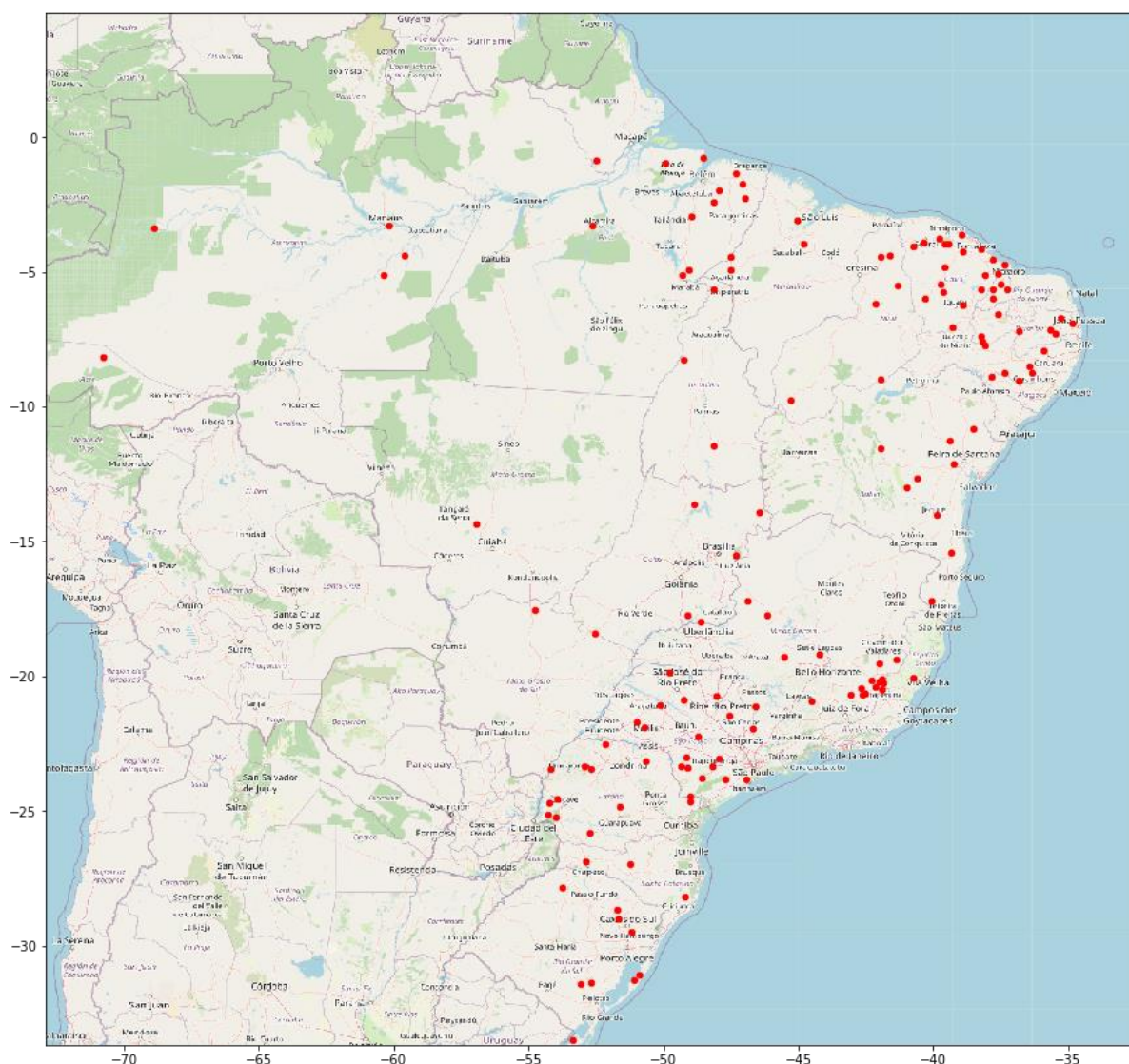


Figura 74. Outliers - HDBSCAN sobre base redimensionada com t-SNE

Diversos indícios foram obtidos dos testes e experimentos feitos com a clusterização dos quantitativos de ocupações informados pelos Municípios nas suas RAIS. Em uma análise superficial, pôde-se identificar diversas regiões do Brasil com Municípios com folhas de pagamento com perfis semelhantes. Não obstante os dois algoritmos utilizados na análise dos agrupamentos terem produzido distribuições distintas sobre a mesma base redimensionada, foi possível identificar diversos agrupamentos regionais. Análises mais aprofundadas desses subagrupamentos podem buscar explicações, causas ou consequências para essas concentrações utilizando outras bases de dados, tais como fatores socioeconômicos comuns.

Análises mais aprofundadas na base original são desafiadoras em razão da quantidade de dimensões que precisam ser consideradas. Em que pese uma única

ocupação representar cerca de 94% da distribuição, conforme estimou-se quando da redução de dimensão usando o algoritmo PCA (figura 27), demais ocupações com poucos funcionários podem indicar características únicas de determinado Município ou região do País, não podendo ser excluídos da base em análise. Por isso, escolheu-se a redução de dimensões produzida pelo algoritmo t-SNE, que, visualmente, melhor distribuiu os Municípios graficamente, e que apesar de ainda ter concentrado a maioria deles, foi capaz de indicar agrupamentos mais coesos e, também, outliers, destacados com o uso do algoritmo HDBSCAN.

7. Links

Link para o vídeo: <https://youtu.be/nBBqG3Oh9Jo>

Link para o repositório:

<https://drive.google.com/drive/folders/1BnvRY51PDQrcR5X4nuk7PMuIBHD050T5?usp=sharing>

REFERÊNCIAS

JAMES, Gareth, WITTEN, Daniela, HASTIE, Trevor, TIBSHIRANI, Robert. **An Introduction to Statistical Learning with Applications in R**. Springer, 2013.

Classificação Brasileira de Ocupações.:

<<http://www.mtecbo.gov.br/cbosite/pages/home.jsf>>. Acesso em 02/02/2021.

Bases Estatísticas RAIS e CAGED: <<http://bi.mte.gov.br/bgcaged/login.php>>.

Usuário: basico, senha: 12345678. Acesso em 15/12/2020.

OSKOLKOV, Nilkolay. How to cluster in High Dimensions. **Towards Data Science**, 23 de jul. de 2019. Disponível em: <<https://towardsdatascience.com/how-to-cluster-in-high-dimensions-4ef693bacc6>>. Acesso em 17/12/2020.

Dados de Longitude e Latitude de cada Município Brasileiro. **Acesso à Informação – GOVERNO FEDERAL**. Disponível em:

<<http://www.consultaesic.cgu.gov.br/busca/dados/Lists/Pedido/Item/displayifs.aspx?List=0c839f31-47d7-4485-ab65-ab0cee9cf8fe&ID=1012693&Web=88cc5f44-8cfe-4964-8ff4-376b5ebb3bef>>. Acesso em 02/01/2021.

SILVA, Jesué Graciliano da. **RAIS / CAGED**. Estatística para Geografia. Disponível em: <https://estatisticaparageografia.wordpress.com/banco-de-dados-rais/>. Acesso em 05/12/2020.

APÊNDICE

Programação/Scripts

Notebook Jupyter: **Ocupações nos Municípios - RAIS 2019 - Análise com Clusters**

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

get_ipython().run_line_magic('matplotlib', 'inline')

# In[ ]:

### MONTANDO O DATASET ###

#Obter lista de arquivos baixados

from os import listdir
from os.path import isfile, join
arquivos_csv_estados = [f for f in listdir('Bases 2019') if isfile(join('Bases 2019', f))]

# Ler o arquivo de cada estado e salvar em um dicionário de dataframes em que a chave é o nome do Estado

df_estados = {}

for arquivo in arquivos_csv_estados:
    df_estados[arquivo[:-4]] = pd.read_csv('Bases 2019\\' + arquivo, sep=';', encoding='latin-1',
skiprows=[0], index_col="CBO Ocupação 2002")

# Ajusta dataframes excluindo linhas e coluna que não interessam à análise

for k in df_estados.keys():
    df_estados[k].drop(columns='Total', inplace=True)
    df_estados[k].drop(['Total', 'Seleções vigentes', 'Variável', 'Ano'], inplace=True)

# In[ ]:
```

```
# fazendo o join de todos os dataframes dos Estados

dataset = df_estados['Acre']

for k,v in df_estados.items():
    if (k != 'Acre'):
        dataset = dataset.join(v,how='outer')

# In[ ]:

dataset

# In[ ]:

# Eliminando a linha NaN que surgiu
dataset = dataset.drop(dataset.index[0])

# In[ ]:

# Fazendo a transposição e preenchendo as células que possuem NaN
dataset = dataset.transpose()
dataset = dataset.fillna(0)

# In[ ]:

dataset

# In[ ]:

# alterando o tipo de dado para integer

for column in dataset.columns:
    dataset[column] = pd.to_numeric(dataset[column], downcast='integer')

# In[ ]:

dataset.dtypes

# Aplicando KMeans no dataset original

# In[ ]:
```

```

from sklearn.cluster import KMeans

# In[ ]:

# Escolha do número de clusters

#Cotovelo
sse = []
for k in range(1, 15):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(dataset)
    sse.append(kmeans.inertia_)

plt.style.use("fivethirtyeight")
plt.plot(range(1, 15), sse)
plt.xticks(range(1, 15))
plt.xlabel("Número de Clusters")
plt.ylabel("SSE")
plt.show()

# In[ ]:

# Escolha do número de clusters

#Silhouette

from sklearn.metrics import silhouette_score

# Lista que armazena os coeficientes silhouette para cada k
silhouette_coefficients = []

# Pesquisa inicia em 2
for k in range(2, 15):
    kmeans = KMeans(n_clusters=k, random_state = 101)
    kmeans.fit(dataset)
    score = silhouette_score(dataset, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 15), silhouette_coefficients)
plt.xticks(range(2, 15))
plt.xlabel("Número de Clusters")
plt.ylabel("Coeficiente Silhouette")
plt.show()

# In[ ]:

kmeans = KMeans(n_clusters=3, random_state = 101)
kmeans.fit(dataset)
kmeans.labels_

```

```
# In[ ]:

dataset_analises = dataset.copy()

# In[ ]:

dataset_analises['KMeans - original'] = kmeans.labels_

# In[ ]:

dataset_analises[dataset_analises['KMeans - original'] == 2]

# In[ ]:

plt.figure(figsize = (10,10))

sns.countplot(x="KMeans - original", data=dataset_analises)

# KMeans sobre base normalizada

# In[ ]:

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_rescaled = scaler.fit_transform(dataset)

# In[ ]:

dataset_normalizado = pd.DataFrame(data_rescaled)

# In[ ]:

dataset_normalizado

# In[ ]:

# Escolha do número de clusters

#Cotovelo
sse = []
```



```

for k in range(1, 15):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(dataset_normalizado)
    sse.append(kmeans.inertia_)

plt.style.use("fivethirtyeight")
plt.plot(range(1, 15), sse, marker = 'o')
plt.xticks(range(1, 15))
plt.xlabel("Número de Clusters - Dataset normalizado StandardScaler")
plt.ylabel("SSE")
plt.show()

# In[ ]:

# Escolha do número de clusters

#Silhouette

from sklearn.metrics import silhouette_score

# Lista que armazena os coeficientes silhouette para cada k
silhouette_coefficients = []

# Pesquisa inicia em 2
for k in range(2, 15):
    kmeans = KMeans(n_clusters=k, random_state = 101)
    kmeans.fit(dataset_normalizado)
    score = silhouette_score(dataset_normalizado, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 15), silhouette_coefficients, marker='o')
plt.xticks(range(2, 15))
plt.xlabel("Número de Clusters")
plt.ylabel("Coeficiente Silhouette")
plt.show()

# In[ ]:

kmeans = KMeans(n_clusters=4, random_state = 101)
kmeans.fit(dataset_normalizado)
kmeans.labels_

# In[ ]:

dataset_analises['KMeans - normalizada'] = kmeans.labels_

# In[ ]:

```

```
dataset_analises
```

```
# In[ ]:
```

```
dataset_analises[dataset_analises['KMeans - normalizada'] == 3]
```

```
# In[ ]:
```

```
plt.figure(figsize = (10,10))
```

```
sns.countplot(x="KMeans - normalizada", data=dataset_analises)
```

```
# Aplicando K-Means no dataset reduzido com PCA
```

```
# In[ ]:
```

```
dataset
```

```
# In[ ]:
```

```
#n_components= 2567 porque há 2567 dimensões no dataset
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2567)
pca.fit(dataset)
variance = pca.explained_variance_ratio_
var = np.cumsum(np.round(variance, 3)*100)
plt.figure(figsize=(12,6))
plt.ylabel('% Variância Explicada')
plt.xlabel('# de Features')
plt.title('Análise PCA')
#plt.ylim(0,100.5)
plt.xlim(0,3)
plt.plot(var, marker = 'o')
```

```
# In[ ]:
```

```
pca = PCA(n_components=2)
pca_dataset = pca.fit_transform(dataset)
pca_df_dataset = pd.DataFrame(pca_dataset, columns=['pc1','pc2'])
```

```
# In[ ]:
```

```
# Número ideal de clusters pelo método do cotovelo
```

```

sse = []
k_list = range(1, 15)
for k in k_list:
    km = KMeans(n_clusters=k)
    km.fit(pca_df_dataset)
    sse.append([k, km.inertia_])

pca_results_dataset = pd.DataFrame({'Cluster': range(1,15), 'SSE': sse})
plt.figure(figsize=(12,6))
plt.plot(pd.DataFrame(sse)[0], pd.DataFrame(sse)[1], marker='o')
plt.title('Número ideal de Clusters - Método do Cotovelo (PCA_Scaled Data)')
plt.xlabel('Número de clusters')
plt.ylabel('Inertia')

# In[ ]:

kmeans = KMeans(n_clusters=3, random_state=101)
kmeans_pca_dataset = kmeans.fit(pca_df_dataset)
labels_pca_dataset = kmeans_pca_dataset.labels_
clusters_pca_dataset = pd.concat([pca_df_dataset,
                                  pd.DataFrame({'pca_clusters':labels_pca_dataset})],
                                  axis=1)

# In[ ]:

dataset_analises['KMeans - PCA'] = labels_pca_dataset

# In[ ]:

dataset_analises

# In[ ]:

dataset_analises[dataset_analises['KMeans - PCA'] == 2]

# In[ ]:

plt.figure(figsize = (10,10))
sns.scatterplot(x = clusters_pca_dataset.iloc[:,0], y = clusters_pca_dataset.iloc[:,1],
                hue=labels_pca_dataset, palette='Set1',
                s=100, alpha=0.2).set_title('Clusters KMeans (3) Derivados do PCA', fontsize=15)
plt.legend()
plt.show()

# Aplicando o KMeans após redução de dimensão com t-SNE

```

```
# In[ ]:

dataset3 = dataset.copy()

# In[ ]:

from sklearn.manifold import TSNE
tSNE=TSNE(n_components=2)
tSNE_result=tSNE.fit_transform(dataset3)

x=tSNE_result[:,0]
y=tSNE_result[:,1]

# In[ ]:

dataset3['x'] = x
dataset3['y'] = y

# In[ ]:

dataset_tSNE = dataset3['x']

# In[ ]:

dataset_tSNE = pd.DataFrame(dataset_tSNE)

# In[ ]:

dataset_tSNE['y'] = dataset3['y']

# In[ ]:

dataset_tSNE

# In[ ]:

plt.figure(figsize=(8,8))
sns.scatterplot(x='x',y='y',data=dataset_tSNE,legend="full")
plt.show()
```

```
# In[ ]:

# Escolha do número de clusters

# Cotovelo
sse = []
for k in range(1, 15):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(dataset_tSNE)
    sse.append(kmeans.inertia_)

plt.style.use("fivethirtyeight")
plt.figure(figsize=(12,6))
plt.plot(range(1, 15), sse, marker='o')
plt.xticks(range(1, 15))
plt.title('Número ideal de Clusters - Método do Cotovelo (t-SNE Data)')
plt.xlabel('Número de clusters')
plt.ylabel('Inertia')
plt.show()
```

```
# In[ ]:

# Escolha do número de clusters

#Silhouette

from sklearn.metrics import silhouette_score

# Lista que armazena os coeficientes silhouette para cada k
silhouette_coefficients = []

# Pesquisa inicia em 2
for k in range(2, 50):
    kmeans = KMeans(n_clusters=k, random_state = 101)
    kmeans.fit(dataset_tSNE)
    score = silhouette_score(df, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.style.use("fivethirtyeight")
plt.figure(figsize=(24,6))
plt.plot(range(2, 50), silhouette_coefficients, marker='o')
plt.xticks(range(2, 50))
plt.xlabel("Número de Clusters")
plt.ylabel("Coeficiente Silhouette - t-SNE")
plt.xlim(0,50)
plt.show()
```

```
# In[ ]:

kmeans = KMeans(n_clusters=4, random_state = 101)
kmeans.fit(dataset_tSNE)
```

```
# In[ ]:

kmeans.cluster_centers_

# In[ ]:

kmeans.labels_

# In[ ]:

n_labels = len(set(kmeans.labels_))

# In[ ]:

dataset_analises['KMeans - tSNE 4'] = kmeans.labels_
dataset_tSNE['KMeans - tSNE 4'] = kmeans.labels_

# In[ ]:

dataset_analises[dataset_analises['KMeans - tSNE 4'] == 3]

# In[ ]:

dataset_tSNE

# In[ ]:

plt.figure(figsize=(12,12))
sns.scatterplot(x='x',y='y',hue='KMeans - tSNE
4',palette=sns.color_palette("hls",len(set(kmeans.labels_))),data=dataset_tSNE,
               legend="full")
plt.show()

# In[ ]:

plt.figure(figsize = (10,10))

sns.countplot(x="KMeans - tSNE 4", data=dataset_tSNE)
```

```

# In[ ]:

dataset_tSNE.drop('KMeans - tSNE 4', axis=1, inplace=True)

# In[ ]:

kmeans = KMeans(n_clusters=14, random_state = 101)
kmeans.fit(dataset_tSNE)
n_labels = len(set(kmeans.labels_))

# In[ ]:

dataset_analises['KMeans - tSNE 14'] = kmeans.labels_
dataset_tSNE['KMeans - tSNE 14'] = kmeans.labels_

# In[ ]:

plt.figure(figsize=(12,12))
sns.scatterplot(x='x',y='y',hue='KMeans - tSNE
14',palette=sns.color_palette("hls",n_labels),data=dataset_tSNE,
               legend="full")
plt.show()

# In[ ]:

dataset_tSNE

# In[ ]:

dataset_tSNE.drop('KMeans - tSNE 14', axis=1, inplace=True)

# In[ ]:

kmeans = KMeans(n_clusters=35, random_state = 101)
kmeans.fit(dataset_tSNE)
n_labels = len(set(kmeans.labels_))

# In[ ]:

dataset_tSNE['KMeans - tSNE 35'] = kmeans.labels_
dataset_analises['KMeans - tSNE 35'] = kmeans.labels_

```

```

# In[ ]:

plt.figure(figsize=(12,12))
sns.scatterplot(x='x',y='y',hue='KMeans - tSNE
35',palette=sns.color_palette("hls",n_labels),data=dataset_tSNE,
               legend="full")
plt.show()

# HDBSCAN

# In[ ]:

dataset_tSNE.drop('KMeans - tSNE 35', axis=1, inplace=True)

# In[ ]:

clusters_sizes = []
num_outliers = []

for k in range (2,50):
    clusterer = hdbscan.HDBSCAN(min_cluster_size=k)
    clusterer.fit(dataset_tSNE)
    clusters_sizes.append(len(set(clusterer.labels_)))
    num_outliers.append(sum(clusterer.labels_ == -1))

# In[ ]:

clusters_sizes

# In[ ]:

num_outliers

# In[ ]:

plt.style.use("fivethirtyeight")
plt.figure(figsize=(12,6))
plt.plot(range(2, 50), clusters_sizes, marker='o')
plt.xticks(range(2, 50))
plt.xlabel("min_cluster_size")
plt.ylabel("Número de Clusters")
plt.xlim(0,20)
plt.show()

```



```
# In[ ]:

plt.style.use("fivethirtyeight")
plt.figure(figsize=(12,6))
plt.plot(range(2, 50), num_outliers, marker='o')
plt.xticks(range(2, 50))
plt.xlabel("min_cluster_size")
plt.ylabel("Quantidade de Outliers")
plt.xlim(0,20)
plt.show()

# In[ ]:

dataset_tSNE

# In[ ]:

clusterer = hdbscan.HDBSCAN(min_cluster_size=13)
clusterer.fit(dataset_tSNE)

# In[ ]:

len(set(clusterer.labels_))

# In[ ]:

dataset_tSNE['HDBSCAN - tSNE'] = clusterer.labels_
dataset_analises['HDBSCAN - tSNE'] = clusterer.labels_

# In[ ]:

dataset_tSNE[dataset_tSNE['HDBSCAN - tSNE'] == -1]

# In[ ]:

plt.figure(figsize = (12,10))

sns.countplot(x="HDBSCAN - tSNE", data=dataset_tSNE)

# In[ ]:

plt.figure(figsize=(12,12))
```

```

sns.scatterplot(x='x',y='y',hue='HDBSCAN -
tSNE',palette=sns.color_palette("hls",len(set(clusterer.labels_))),data=dataset_tSNE,
               legend=False,s=30)

plt.show()

# In[ ]:

dataset_tSNE[dataset_tSNE['HDBSCAN - tSNE'] == -1]

# In[ ]:

outliers = dataset_tSNE[dataset_tSNE['HDBSCAN - tSNE'] == -1]

# In[ ]:

plt.figure(figsize=(12,12))

plt.scatter(x = dataset_tSNE['x'],y=dataset_tSNE['y'], s=30, c='gray', linewidth=0,alpha=0.25)
plt.scatter(x = outliers['x'],y=outliers['y'], s=30, c='red',linewidth=0,alpha=0.5)

# In[ ]:

dataset_analises.to_csv('Dataset_analises.csv', sep=';', encoding='latin-1')

# In[ ]:

dataset_tSNE.drop('HDBSCAN - tSNE',axis=1,inplace=True)
dataset_tSNE.to_csv('Dataset tSNE.csv', sep=';', encoding='latin-1')

# EXIBINDO RESULTADOS NO MAPA

# In[ ]:

coordenadas = pd.read_csv('Coordenadas_Sedes_Municipios\Sedes_Coordenadas_Municipios.csv',
                          sep = ';', index_col=0)

nome = pd.read_csv('Coordenadas_Sedes_Municipios\Codigo_Nome_Municipios.txt', encoding='latin-1',
                  sep=';', index_col=0)

# In[ ]:

frames = [nome, coordenadas]

```

```
municipios = pd.concat(frames, axis = 1)

# In[ ]:

municipios

# In[ ]:

municipios['NOME MUNICIPIO'] = municipios['NOME MUNICIPIO'].str.upper()

# In[ ]:

municipios

# In[ ]:

municipios['UF-MUNICÍPIO'] = municipios['UF']+'-'+municipios['NOME MUNICIPIO']

# In[ ]:

municipios = municipios[['UF-MUNICÍPIO','LATITUDE','LONGITUDE']]

# In[ ]:

municipios

# In[ ]:

import unicodedata

def converte_utf(com_acento):
    sem_acento = unicodedata.normalize("NFD", com_acento)
    sem_acento = sem_acento.encode("ascii", "ignore")
    sem_acento = sem_acento.decode("utf-8")
    return sem_acento

# In[ ]:

municipios['UF-MUNICÍPIO'] = municipios['UF-MUNICÍPIO'].apply(converte_utf)
```

```

# In[ ]:

municipios.set_index(['UF-MUNICÍPIO'], inplace=True)

# In[ ]:

municipios = municipios.sort_index()

# In[ ]:

BBox = ((municipios.LONGITUDE.min(), municipios.LONGITUDE.max(),
          municipios.LATITUDE.min(), municipios.LATITUDE.max()))

# In[ ]:

brasil = plt.imread('Coordenadas_Sedes_Municipios\map_modificado.png')

# In[ ]:

dataset_analises = dataset_analises.join(municipios, how='outer')

# In[ ]:

import matplotlib.cm as cm
fig, ax = plt.subplots(figsize = (15,15))
ax.set_xlim(BBox[0], BBox[1])
ax.set_ylim(BBox[2], BBox[3])
ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')

x = 4 # quantidade de clusters

color = cm.plasma(np.linspace(0, 1, x)) # cm.hot(np.linspace(0, 1, x))

for i in range(0,x):

    municipios_cluster = dataset_analises[dataset_analises['KMeans - tSNE 4'] == i]
    ax.scatter(municipios_cluster.LONGITUDE, municipios_cluster.LATITUDE,
               c=color[i], s=20, alpha=1, zorder=1)

# In[ ]:

x = 4 # quantidade de clusters

```

```

color = cm.plasma(np.linspace(0, 1, x))#cm.hot(np.linspace(0, 1, x))

for i in range(0,x):
    fig, ax = plt.subplots(figsize = (15,15))
    ax.set_xlim(BBox[0],BBox[1])
    ax.set_ylim(BBox[2],BBox[3])
    ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')
    municipios_cluster = dataset_analises[dataset_analises['KMeans - tSNE 4'] == i]
    ax.scatter(municipios_cluster.LONGITUDE, municipios_cluster.LATITUDE,
               c=color[i], s=20, alpha=1,zorder=1)

# MINAS GERAIS

# In[ ]:

MG = pd.read_csv('Coordenadas_Sedes_Municipios\Dataset_analises.csv', sep = ';', encoding = 'latin-1')#,
index_col=0)
MG.rename(columns={"Unnamed: 0": "UF-MUNICÍPIO"},inplace=True)

# In[ ]:

MG[['ESTADO','MUNICÍPIO']] = MG['UF-MUNICÍPIO'].str.split('-',1,expand = True)
MG.set_index(['UF-MUNICÍPIO'],inplace=True)

# In[ ]:

MG = MG.join(municipios,how='outer')

# In[ ]:

MG = pd.DataFrame(MG[MG['ESTADO'] == 'MG'])

# In[ ]:

# Distribuição de municípios de Minas Gerais entre os clusters

plt.figure(figsize = (5,5))

sns.countplot(x="KMeans - tSNE 4", data=MG)

# In[ ]:

# Minas Gerais

```

```

fig, ax = plt.subplots(figsize = (15,15))
ax.set_xlim(BBox[0],BBox[1])
ax.set_ylim(BBox[2],BBox[3])
ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')

x = 4 # quantidade de clusters

color = cm.plasma(np.linspace(0, 1, x))#cm.hot(np.linspace(0, 1, x))

for i in range(0,x):

    municipios_cluster = MG[MG['KMeans - tSNE 4'] == i]
    ax.scatter(municipios_cluster.LONGITUDE, municipios_cluster.LATITUDE,
               c=color[i], s=20, alpha=1,zorder=1)

# In[ ]:

x = 4 # quantidade de clusters

color = cm.plasma(np.linspace(0, 1, x))#cm.hot(np.linspace(0, 1, x))
#color = ['r','g','b']

for i in range(0,x):
    fig, ax = plt.subplots(figsize = (15,15))
    ax.set_xlim(BBox[0],BBox[1])
    ax.set_ylim(BBox[2],BBox[3])
    ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')
    municipios_cluster = MG[MG['KMeans - tSNE 4'] == i]
    ax.scatter(municipios_cluster.LONGITUDE, municipios_cluster.LATITUDE,
               c=color[i], s=20, alpha=1,zorder=1)

# HDBSCAN NO MAPA

# In[ ]:

fig, ax = plt.subplots(figsize = (15,15))
ax.set_xlim(BBox[0],BBox[1])
ax.set_ylim(BBox[2],BBox[3])
ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')

x = 8 # quantidade de clusters

color = cm.plasma(np.linspace(0, 1, x))#cm.hot(np.linspace(0, 1, x))

for i in range(x-1,-1,-1):

    municipios_cluster = dataset_analises[dataset_analises['HDBSCAN - tSNE'] == i]
    ax.scatter(municipios_cluster.LONGITUDE, municipios_cluster.LATITUDE,
               c=color[i], s=20, alpha=1,zorder=1)

```

```

# In[ ]:

x = 8 # quantidade de clusters

color = cm.plasma(np.linspace(0, 1, x))#cm.hot(np.linspace(0, 1, x))
#color = ['r','g','b']

for i in range(x-1,-1,-1):
    fig, ax = plt.subplots(figsize = (15,15))
    ax.set_xlim(BBox[0],BBox[1])
    ax.set_ylim(BBox[2],BBox[3])
    ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')
    municipios_cluster = dataset_analises[dataset_analises['HDBSCAN - tSNE'] == i]
    ax.scatter(municipios_cluster.LONGITUDE, municipios_cluster.LATITUDE,
               c=color[i], s=20, alpha=1,zorder=1)

# In[ ]:

# Outliers

fig, ax = plt.subplots(figsize = (15,15))
ax.set_xlim(BBox[0],BBox[1])
ax.set_ylim(BBox[2],BBox[3])
ax.imshow(brasil, zorder=0, extent = BBox, aspect= 'equal')

outliers = dataset_analises[dataset_analises['HDBSCAN - tSNE'] == -1]

ax.scatter(outliers.LONGITUDE, outliers.LATITUDE, c='r', s=20, alpha=1,zorder=1)

```