

Infraestrutura

Para as questões a seguir, você deverá executar códigos em um notebook Jupyter, rodando em ambiente local, certifique-se que:

- Você está rodando em Python 3.9+
- Você está usando um ambiente virtual: Virtualenv ou Anaconda
- Todas as bibliotecas usadas nesse exercícios estão instaladas em um ambiente virtual específico
- Gere um arquivo de requerimentos (requirements.txt) com os pacotes necessários. É necessário se certificar que a versão do pacote está disponibilizada.
- Tire um printscreen do ambiente que será usado rodando em sua máquina.
- Disponibilize os códigos gerados, assim como os artefatos acessórios (requirements.txt) e instruções em um repositório GIT público. (se isso não for feito, o diretório com esses arquivos deverá ser enviado compactado no moodle).

```
In [3]: # python version  
!python --version
```

Python 3.11.6

```
In [4]: # virtualenv  
!which python
```

/home/bakudas/code/INFNET_POS/validacao-modelos-clusterizacao/.venv/bin/python

```
In [5]: # install libs  
!pip install scikit-learn scikit-learn-extra pandas matplotlib seaborn scipy num
```

```
In [6]: # generate requirements file  
!pip freeze > requirements.txt
```

```
In [7]: # check requirements file  
!cat requirements.txt
```

```
asttokens==3.0.0
comm==0.2.2
contourpy==1.3.1
cycler==0.12.1
debugpy==1.8.11
decorator==5.1.1
executing==2.1.0
fonttools==4.55.3
ipykernel==6.29.5
ipython==8.31.0
jedi==0.19.2
joblib==1.4.2
jupyter_client==8.6.3
jupyter_core==5.7.2
kiwisolver==1.4.8
matplotlib==3.10.0
matplotlib-inline==0.1.7
nest-asyncio==1.6.0
numpy==2.2.1
packaging==24.2
pandas==2.2.3
parso==0.8.4
pexpect==4.9.0
pillow==11.0.0
platformdirs==4.3.6
prompt_toolkit==3.0.48
psutil==6.1.1
ptyprocess==0.7.0
pure_eval==0.2.3
Pygments==2.18.0
pyparsing==3.2.0
python-dateutil==2.9.0.post0
pytz==2024.2
pyzmq==26.2.0
scikit-learn==1.6.0
scikit-learn-extra==0.3.0
scipy==1.14.1
seaborn==0.13.2
six==1.17.0
stack-data==0.6.3
threadpoolctl==3.5.0
tornado==6.4.2
traitlets==5.14.3
typing_extensions==4.12.2
tzdata==2024.2
wcwidth==0.2.13
```

Escolha de base de dados

Para as questões a seguir, usaremos uma base de dados e faremos a análise exploratória dos dados, antes da clusterização.

- Escolha uma base de dados para realizar o trabalho. Essa base será usada em um problema de clusterização.
- Escreva a justificativa para a escolha de dados, dando sua motivação e objetivos.

- Mostre através de gráficos a faixa dinâmica das variáveis que serão usadas nas tarefas de clusterização. Analise os resultados mostrados. O que deve ser feito com os dados antes da etapa de clusterização?
- Realize o pré-processamento adequado dos dados. Descreva os passos necessários.

```
In [21]: import pandas as pd
import numpy as np
import json
from datetime import datetime
```

```
In [121...]: # carregando os dados do json
with open("data/disparo_arma_fogo_pmerj.json", "r", encoding="utf-8") as f:
    data = json.load(f) # 'data' agora é um dicionário Python

# data["features"] contém uma lista de "features"; cada feature tem "attributes"
features = data["features"]

# Lista de dicionários (um por Linha/registro)
rows = []
for feat in features:
    # Pegamos os atributos
    attr = feat["attributes"]

    # Adicionamos também as coordenadas ao dicionário de atributos
    #   x = longitude, y = latitude
    #   Se a geometry for nula, tratamos como None
    geom = feat.get("geometry", {})
    attr["geometry_x"] = geom.get("x", None)
    attr["geometry_y"] = geom.get("y", None)

    rows.append(attr)

# Cria o DataFrame
df = pd.DataFrame(rows)

# print da relação
df.head(5)
```

Out[121...]

	objectid	id_ocorrencia	presen_agen_segur_ocorrencia	local_ocorrencia	data_ocorrencia
0	1	0		Não	Favela do Terreirão - Recreio dos Bandeirantes...
1	2	0		Sim	Av. Guilherme de Almeida - Recreio dos Bandeir...
2	3	0		Sim	Avenida Monsenhor Ascâneo - Barra da Tijuca, R...
3	4	0		Sim	Estrada do Itanhangá - Itanhangá, Rio de Janei...
4	5	0		Sim	Vidigal, Rio de Janeiro - RJ, Brasil

5 rows × 86 columns

In [122...]

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1074 entries, 0 to 1073
Data columns (total 86 columns):
 #   Column           Non-Null Count Dtype
 ---  -----
 0   objectid        1074 non-null  int64
 1   id_ocorrencia  1074 non-null  int64
 2   presen_agen_segur_ocorrencia 1074 non-null  object
 3   local_ocorrencia 1074 non-null  object
 4   data_ocorrencia  1074 non-null  int64
 5   hora_ocorrencia 1074 non-null  object
 6   timestamp_ocorrencia 1074 non-null  int64
 7   dia_semana_ocorrencia 1074 non-null  object
 8   dia_semana_ocorrencia_numero 1074 non-null  int64
 9   dia_ocorrencia    1074 non-null  int64
 10  mes_ocorrencia   1067 non-null  object
 11  mes_ocorrencia_numero 1074 non-null  int64
 12  ano_ocorrencia   1074 non-null  int64
 13  qtd_morto_civil_ocorrencia 1074 non-null  int64
 14  qtd_morto_agen_segur_ocorrencia 1074 non-null  int64
 15  homem_qtd_mortos_oc 1074 non-null  int64
 16  mulher_qtd_mortos_oc 1074 non-null  int64
 17  qtd_ferido_civil_ocorrencia 1074 non-null  int64
 18  qtd_ferido_agen_segur_ocorrencia 1074 non-null  int64
 19  homem_qtd_feridos_oc 1074 non-null  int64
 20  mulher_qtd_feridos_oc 1074 non-null  int64
 21  nome_cidade      1074 non-null  object
 22  gentilico_cidade 1074 non-null  object
 23  populacao_cidade 1074 non-null  int64
 24  area_cidade      1074 non-null  int64
 25  densidade_demo_cidade 1074 non-null  float64
 26  uf_estado        1074 non-null  object
 27  chacina_oc        1074 non-null  object
 28  chacina_qtd_mortos_oc 1074 non-null  int64
 29  chacina_unidades_policiais_oc 1074 non-null  object
 30  ag_seguranca_vitima_oc 1074 non-null  object
 31  ag_seguranca_mortos_status_oc 22 non-null  object
 32  ag_seguranca_feridos_status_oc 46 non-null  object
 33  bala_perdida_oc 1074 non-null  object
 34  bala_perdida_qtd_mortos_oc 1074 non-null  int64
 35  bala_perdida_qtd_feridos_oc 1074 non-null  int64
 36  interior_residencia_oc 1074 non-null  object
 37  interior_residencia_qtd_mortos_ 1074 non-null  int64
 38  interior_residencia_qtd_feridos 1074 non-null  int64
 39  imediacao_ensino_oc 1074 non-null  object
 40  imediacao_ensino_qtd_mortos_oc 1074 non-null  int64
 41  imediacao_ensino_qtd_feridos_oc 1074 non-null  int64
 42  vitima_crianca_oc 1074 non-null  object
 43  vitima_crianca_qtd_mortos_oc 1074 non-null  int64
 44  info_adicional_crianca_morta_oc 7 non-null  object
 45  vitima_crianca_qtd_feridos_oc 1074 non-null  int64
 46  info_adicional_crianca_ferida_o 5 non-null  object
 47  vitima_adolescente_oc 1074 non-null  object
 48  vitima_adolescente_qtd_mortos_o 1074 non-null  int64
 49  info_adicional_adolescente_mort 29 non-null  object
 50  vitima_adolescente_qtd_feridos_ 1074 non-null  int64
 51  info_adicional_adolescente_feri 7 non-null  object
 52  vitima_idoso_oc 1074 non-null  object
 53  vitima_idoso_qtd_mortos_oc 1074 non-null  int64
 54  info_adicional_idoso_morto_oc 4 non-null  object
```

```

55 vitima_idoso_qtd_feridos_oc      1074 non-null    int64
56 info_adicional_idoso_ferido_oc   7 non-null     object
57 informacao_via_oc                0 non-null     object
58 descricao_via_interrompida_oc    0 non-null     object
59 data_interrupcao_via_oc         0 non-null     object
60 data_liberacao_via_oc          0 non-null     object
61 outros_recortes                 90 non-null    object
62 motivo_principal                1074 non-null    object
63 motivo_complementar             100 non-null    object
64 longitude_ocorrencia            1074 non-null    int64
65 latitude_ocorrencia              1074 non-null    int64
66 aisp                            1074 non-null    int64
67 cisp                            1074 non-null    int64
68 delegacia                       1074 non-null    object
69 localidade                      1074 non-null    object
70 categoria                        1074 non-null    object
71 cod_craai                        1074 non-null    int64
72 craai                           1074 non-null    object
73 pip_dk_novo                     1027 non-null    object
74 pip_dscr_novo                   1027 non-null    object
75 pip_dk_novo_esp                 991 non-null    object
76 pip_dscr_novo_esp               991 non-null    object
77 pip_dk_novo_vd                  991 non-null    object
78 pip_dscr_novo_vd                991 non-null    object
79 globalid                         1074 non-null    object
80 created_user                     0 non-null     object
81 created_date                     0 non-null     object
82 last_edited_user                74 non-null     object
83 last_edited_date                74 non-null     float64
84 geometry_x                       1074 non-null    float64
85 geometry_y                       1074 non-null    float64
dtypes: float64(4), int64(36), object(46)
memory usage: 721.7+ KB

```

In [123...]

df.describe()

Out[123...]

	objectid	id_ocorrencia	data_ocorrencia	timestamp_ocorrencia	dia_semana_o
count	1074.000000	1074.000000	1.074000e+03	1.074000e+03	
mean	539.971136	3378.921788	1.645030e+12	1.645069e+12	
std	311.819080	8328.985030	5.574775e+10	5.575122e+10	
min	1.000000	0.000000	1.484795e+12	1.484795e+12	
25%	269.250000	0.000000	1.621955e+12	1.622010e+12	
50%	541.500000	0.000000	1.672844e+12	1.672873e+12	
75%	809.750000	0.000000	1.682478e+12	1.682512e+12	
max	1078.000000	32685.000000	1.691982e+12	1.692007e+12	

8 rows × 40 columns

In [124...]

```

# Converter campos de data (que estão em epoch milissegundos) para datas Legíveis
date_fields = [
    "data_ocorrencia",
    "timestamp_ocorrencia",
]

```

```

    "data_interrupcao_via_oc",
    "data_liberacao_via_oc",
    "created_date",
    "last_edited_date"
]

for col in date_fields:
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], unit='ms', errors='coerce')

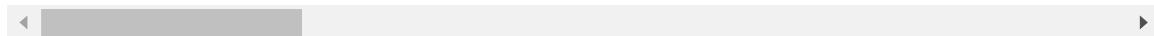
# print da relação
df.head(5)

```

Out[124...]

	objectid	id_ocorrencia	presen_agen_segur_ocorrencia	local_ocorrencia	data_ocorrencia
0	1	0	Não	Favela do Terreirão - Recreio dos Bandeirantes...	2021-06-03:00:
1	2	0	Sim	Av. Guilherme de Almeida - Recreio dos Bandeir...	2022-01-03:00:
2	3	0	Sim	Avenida Monsenhor Ascâneo - Barra da Tijuca, R...	2022-07-03:00:
3	4	0	Sim	Estrada do Itanhangá - Itanhangá, Rio de Janei...	2023-08-03:00:
4	5	0	Sim	Vidigal, Rio de Janeiro - RJ, Brasil	2020-01-03:00:

5 rows × 86 columns



Dados e Estrutura

Conforme a amostra, temos colunas como:

- Chaves de identificação: objectid, id_ocorrencia
- Atributos categóricos: presen_agen_segur_ocorrencia (Sim/Não), dia_semana_ocorrencia, uf_estado, etc.
- Datas e horas**: data_ocorrencia, hora_ocorrencia, timestamp_ocorrencia (convertido corretamente para datetime64)
- Características numéricas: qtd_morto_civil_ocorrencia, qtd_ferido_civil_ocorrencia, etc.

- Localização geográfica: geometry_x (longitude), geometry_y (latitude) – já em graus decimais .
- Também vemos colunas com strings como pip_dk_novo_vd e pip_dscr_novo_vd contendo valores concatenados ("29934401|29934732"). É possível que isso represente chaves de outras variáveis ou categorias.

```
In [96]: !pip install geopandas &> /dev/null
```

```
In [97]: import geopandas as gpd
from shapely.geometry import Point

# Converter pandas para GeoPandas
geometry = [Point(xy) for xy in zip(df["geometry_x"], df["geometry_y"])]
gdf = gpd.GeoDataFrame(df, geometry=geometry, crs="EPSG:4326")
```

```
In [125...]:
# -----
# 1. Remover colunas irrelevantes
# -----
# Vamos descartar sumariamente algumas colunas que de cara
# não fazem muito sentido para a análise:

cols_to_drop = ["dia_semana_ocorrencia", "globalid", "created_user", "outros_rec"
                 "last_edited_user", "last_edited_date"]

# Verifica se elas existem no DF antes de dropar
cols_to_drop = [c for c in cols_to_drop if c in df.columns]
df.drop(columns=cols_to_drop, inplace=True)

# -----
# 2. Análise de dados faltantes
# -----
# Vamos ver a soma de valores nulos por coluna
print("Valores nulos por coluna:")
print(df.isna().sum())

# Remover colunas que têm mais de 70% de valores nulos
threshold = 0.7 # 70%
n = len(df)
cols_remove = [col for col in df.columns
               if df[col].isna().sum() / n > threshold]

print("\nColunas com mais de 70% de valores nulos:", cols_remove)
df.drop(columns=cols_remove, inplace=True)

# -----
# 3. Tratar dados faltantes nas colunas que restaram
# -----
# (a) Se for numérica, vamos imputar a média ou mediana:
cols_numericas = df.select_dtypes(include=[np.number]).columns

for col in cols_numericas:
    media_col = df[col].mean()
    # Substituir os NaN pela média
    df[col].fillna(media_col, inplace=True)

# (b) Se for categórica (string), vamos criar categoria "Desconhecido":
cols_categoricas = df.select_dtypes(include=[object]).columns
```

```

for col in cols_categoricas:
    df[col].fillna("Desconhecido", inplace=True)

# -----
# 4. Conferir resultado
# -----
print("\nDepois do tratamento:")
print(df.isna().sum())

```

Valores nulos por coluna:

objectid	0
id_ocorrencia	0
presen_agen_segur_ocorrencia	0
local_ocorrencia	0
data_ocorrencia	0
	..
pip_dscr_novo_esp	83
pip_dk_novo_vd	83
pip_dscr_novo_vd	83
geometry_x	0
geometry_y	0

Length: 79, dtype: int64

Colunas com mais de 70% de valores nulos: ['ag_seguranca_mortos_status_oc', 'ag_seguranca_feridos_status_oc', 'info_adicional_crianca_morta_oc', 'info_adicional_crianca_ferida_o', 'info_adicional_adolescente_mort', 'info_adicional_adolescente_feri', 'info_adicional_idoso_morto_oc', 'info_adicional_idoso_ferido_oc', 'informacao_via_oc', 'descricao_via_interrompida_oc', 'data_interrupcao_via_oc', 'data_liberacao_via_oc', 'motivo_complementar']

Depois do tratamento:

objectid	0
id_ocorrencia	0
presen_agen_segur_ocorrencia	0
local_ocorrencia	0
data_ocorrencia	0
	..
pip_dscr_novo_esp	0
pip_dk_novo_vd	0
pip_dscr_novo_vd	0
geometry_x	0
geometry_y	0

Length: 66, dtype: int64

/tmp/ipykernel_5492/16581103.py:39: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df[col].fillna(media_col, inplace=True)
```

/tmp/ipykernel_5492/16581103.py:45: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df[col].fillna("Desconhecido", inplace=True)
```

In [126...]

```
# Separando as colunas numericas
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()

print("Colunas numéricas iniciais:")
print(numeric_cols)

# Suponha que essas colunas sejam IDs e não precisamos delas para análises estatísticas
id_cols = ["objectid", "id_ocorrencia"]

# Filtra, removendo as colunas de ID da lista:
numeric_cols = [c for c in numeric_cols if c not in id_cols]

print("\nColunas numéricas (excluindo IDs):")
print(numeric_cols)

df.drop(columns=id_cols, inplace=True, errors="ignore")

# exporta uma amostra dos dados
df.head(20).to_csv("data/export_data_processed.csv")
```

Colunas numéricas iniciais:

```
['objectid', 'id_ocorrencia', 'dia_semana_ocorrencia_numero', 'dia_ocorrencia',
'mes_ocorrencia_numero', 'ano_ocorrencia', 'qtd_morto_civil_ocorrencia', 'qtd_mor-
to_agen_segur_ocorrencia', 'homem_qtd_mortos_oc', 'mulher_qtd_mortos_oc', 'qtd_fe-
rido_civil_ocorrencia', 'qtd_ferido_agen_segur_ocorrencia', 'homem_qtd_feridos_o-
c', 'mulher_qtd_feridos_oc', 'populacao_cidade', 'area_cidade', 'densidade_demo_c-
idade', 'chacina_qtd_mortos_oc', 'bala_perdida_qtd_mortos_oc', 'bala_perdida_qtd_
feridos_oc', 'interior_residencia_qtd_mortos_', 'interior_residencia_qtd_ferido-
s', 'imediacao_ensino_qtd_mortos_oc', 'imediacao_ensino_qtd_feridos_oc', 'vitima_
crianca_qtd_mortos_oc', 'vitima_crianca_qtd_feridos_oc', 'vitima_adolescente_qtd_
mortos_o', 'vitima_adolescente_qtd_feridos_', 'vitima_idoso_qtd_mortos_oc', 'viti-
ma_idoso_qtd_feridos_oc', 'longitude_ocorrencia', 'latitude_ocorrencia', 'aisp',
'cisp', 'cod_craai', 'geometry_x', 'geometry_y']
```

Colunas numéricas (excluindo IDs):

```
['dia_semana_ocorrencia_numero', 'dia_ocorrencia', 'mes_ocorrencia_numero', 'ano_
ocorrencia', 'qtd_morto_civil_ocorrencia', 'qtd_morto_agente_segur_ocorrencia', 'ho-
mem_qtd_mortos_oc', 'mulher_qtd_mortos_oc', 'qtd_ferido_civil_ocorrencia', 'qtd_f-
erido_agente_segur_ocorrencia', 'homem_qtd_feridos_oc', 'mulher_qtd_feridos_oc', 'po-
pulacao_cidade', 'area_cidade', 'densidade_demo_cidade', 'chacina_qtd_mortos_oc',
'bala_perdida_qtd_mortos_oc', 'bala_perdida_qtd_feridos_oc', 'interior_residencia_
qtd_mortos_', 'interior_residencia_qtd_feridos', 'imediacao_ensino_qtd_mortos_o-
c', 'imediacao_ensino_qtd_feridos_oc', 'vitima_crianca_qtd_mortos_oc', 'vitima_cr-
ianca_qtd_feridos_oc', 'vitima_adolescente_qtd_mortos_o', 'vitima_adolescente_qtd_
_feridos_', 'vitima_idoso_qtd_mortos_oc', 'vitima_idoso_qtd_feridos_oc', 'longitu-
de_ocorrencia', 'latitude_ocorrencia', 'aisp', 'cisp', 'cod_craai', 'geometry_x',
'geometry_y']
```

In [127...]

```
# -----
# 3. Binarização de "Sim"/"Não" (Exemplo)
# -----
bin_cols = ["presen_agente_segur_ocorrencia",
            "chacina_oc",
            "ag_seguranca_vitima_oc",
            "bala_perdida_oc",
            "interior_residencia_oc",
            "imediacao_ensino_oc",
            "vitima_crianca_oc",
            "vitima_adolescente_oc",
            "vitima_idoso_oc"]

# Binariza as colunas binárias com valores "Sim", "Não" e "Desconhecido" em 1 e
for col in bin_cols:
    if col in df.columns:
        df[col] = df[col].map({"Sim": 1, "Não": 0, "Desconhecido": 0}).astype(int)

# -----
# 3.2. Conversões de Datas
# -----
if "data_ocorrencia" in df.columns:
    df["ano"] = df["data_ocorrencia"].dt.year
    df["mes"] = df["data_ocorrencia"].dt.month
    df["dia"] = df["data_ocorrencia"].dt.day
    df["hora"] = df["data_ocorrencia"].dt.hour

# Criação de faixa horária
def categoriza_hora(h):
    if 5 <= h < 12:
        return 0 # "Manha"
    elif 12 <= h < 18:
        return 1 # "Tarde"
    else:
        return 2 # "Noite"
```

```

        return 1 # "Tarde"
elif 18 <= h < 24:
    return 2 # "Noite"
else:
    return 3 # "Madrugada"

df["faixa_horaria"] = df["hora"].apply(categoriza_hora)

```

In [128...]: df.head(5)

Out[128...]:

		presen_agen_segur_ocorrencia	local_ocorrencia	data_ocorrencia	hora_ocorrencia	tim
0	0		Favela do Terreirão - Recreio dos Bandeirantes...	2021-06-29 03:00:00	13:55:00	:
1	1		Av. Guilherme de Almeida - Recreio dos Bandeir...	2022-01-20 03:00:00	04:00:00	:
2	1		Avenida Monsenhor Ascâneo - Barra da Tijuca, R...	2022-07-29 03:00:00	06:33:00	:
3	1		Estrada do Itanhangá - Itanhangá, Rio de Janei...	2023-08-02 03:00:00	06:00:00	:
4	1		Vidigal, Rio de Janeiro - RJ, Brasil	2020-01-16 03:00:00	05:45:00	:

5 rows × 69 columns

In [129...]:

```

# Separando as colunas de interesse para análise
df_cluster = df[[
    # Categorias binárias
    "presen_agen_segur_ocorrencia",
    "chacina_oc",
    "ag_seguranca_vitima_oc",
    "bala_perdida_oc",
    "interior_residencia_oc",
    "imediacao_ensino_oc",
    "vitima_crianca_oc",
    "vitima_adolescente_oc",
    "vitima_idoso_oc",

    # Numéricas
    "dia_semana_ocorrencia_numero",
    "dia_ocorrencia",
    "mes_ocorrencia_numero",
    "ano_ocorrencia",
    "qtd_morto_civil_ocorrencia",

```

```

    "qtd_ferido_civil_ocorrencia",
    "qtd_morto_agente_segur_ocorrencia",
    "qtd_ferido_agente_segur_ocorrenci",
    "faixa_horaria",

    # Coordenadas
    "longitude_ocorrencia",
    "latitude_ocorrencia",

]].copy()

# Juntar 'homem_qtd_mortos_oc' + 'mulher_qtd_mortos_oc' = 'mortos_totais'
df_cluster["mortos_totais"] = df["homem_qtd_mortos_oc"] + df["mulher_qtd_mortos_"]

# 'motivo_principal' é categórico (Operação, Ação policial, etc.):
# Vamos aplicar One-Hot encoding:
df_motivo = pd.get_dummies(df["motivo_principal"], prefix="motivo")
df_cluster = pd.concat([df_cluster, df_motivo], axis=1)

```

In [130...]

```
# exporta os dados limpos prontos para seguir com a análise
df_cluster.to_csv("data/data_cleaned.csv")
```

In [131...]

```
df_cluster
```

Out[131...]

	presen_agente_segur_ocorrencia	chacina_oc	ag_seguranca_vitima_oc	bala_perdida_oc
0	0	1	0	·
1	1	1	1	·
2	1	0	0	·
3	1	0	0	·
4	1	1	0	·
...	·
1069	0	1	0	·
1070	1	0	0	·
1071	1	0	0	·
1072	1	0	0	·
1073	0	0	0	·

1074 rows × 34 columns



Justificativa da Escolha da Base de Dados

Motivações e Objetivos

A base de dados escolhida foi a de Registro de Disparo de Arma de Fogo na PMERJ do Instituto Fogo Cruzado

(<https://geo.mprj.mp.br/portal/apps/sites/#/gestaodoterritorio/datasets/7100faf722a549ff98>) contém informações relevantes para o tipo de problema que queremos resolver com clusterização. Por exemplo, como ela traz registros de ocorrências/incidentes com várias variáveis (número de vítimas, localização, presença de determinada condição, etc.), podemos:

- Identificar perfis/padrões de eventos semelhantes.
- Agrupar regiões ou ocorrências por características semelhantes (ex.: tipo de crime, presença de agente de segurança, etc.).
- Melhorar a tomada de decisão de alocação de recursos ou análise de hotspots (se for um caso geoespacial).

A escolha se justifica pois:

- Variedade de atributos (categóricos e numéricos) que permitem explorar diferentes dimensões (sazonalidade, localização, etc.).
- Dimensionalidade moderada: nem muito grande a ponto de inviabilizar a análise manual, nem muito pequena que impeça uma clusterização significativa.
- Potencial de insights: as variáveis apresentam correlações que podem ser exploradas, e há interesse em segmentar os registros em grupos semelhantes.

In [132...]

```
# Visualizar tamanho e colunas
print("Tamanho do dataset:", df_cluster.shape)
print("Colunas:", df_cluster.columns.tolist())

# Estatísticas descritivas das colunas numéricas
print(df_cluster.describe())
```

Tamanho do dataset: (1074, 34)

Colunas: ['presen_agen_segur_ocorrencia', 'chacina_oc', 'ag_seguranca_vitima_oc', 'bala_perdida_oc', 'interior_residencia_oc', 'imediacao_ensino_oc', 'vitima_crianca_oc', 'vitima_adolescente_oc', 'vitima_idoso_oc', 'dia_semana_ocorrencia_numero', 'dia_ocorrencia', 'mes_ocorrencia_numero', 'ano_ocorrencia', 'qtd_morto_civil_ocorrencia', 'qtd_ferido_civil_ocorrencia', 'qtd_morto_agr_segur_ocorrencia', 'qtd_ferido_agr_segur_ocorrencia', 'faixa_horaria', 'longitude_ocorrencia', 'latitude_ocorrencia', 'mortos_totais', 'motivo_Arrastao', 'motivo_Ataque_a_civis', 'motivo_Acao_policial', 'motivo_Briga', 'motivo_Disputa', 'motivo_Execucao', 'motivo_Homicidio/Tentativa', 'motivo_Nao_identificado', 'motivo_Operacao_policial', 'motivo_Sequestro', 'motivo_Tentativa/Roubo', 'motivo_Tentativa/Roubo_a_banco', 'motivo_Tentativa/Roubo_de_cargas']

	presen_agen_segur_ocorrencia	chacina_oc	ag_seguranca_vitima_oc	\
count	1074.000000	1074.000000	1074.000000	
mean	0.905028	0.330540	0.057728	
std	0.293313	0.470627	0.233337	
min	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	
75%	1.000000	1.000000	0.000000	
max	1.000000	1.000000	1.000000	

	bala_perdida_oc	interior_residencia_oc	imediacao_ensino_oc	\
count	1074.000000	1074.000000	1074.000000	
mean	0.040968	0.028864	0.001862	
std	0.198309	0.167502	0.043133	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	

	vitima_crianca_oc	vitima_adolescente_oc	vitima_idoso_oc	\
count	1074.000000	1074.000000	1074.000000	
mean	0.013035	0.036313	0.010242	
std	0.113479	0.187155	0.100730	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	

	dia_semana_ocorrencia_numero	...	mes_ocorrencia_numero	\
count	1074.000000	...	1074.000000	
mean	3.037244	...	5.741155	
std	1.762325	...	3.179331	
min	0.000000	...	1.000000	
25%	2.000000	...	3.000000	
50%	3.000000	...	6.000000	
75%	4.000000	...	8.000000	
max	6.000000	...	12.000000	

	ano_ocorrencia	qtd_morto_civil_ocorrencia	\
count	1074.000000	1074.000000	
mean	2021.695531	1.507449	
std	1.800238	2.181093	
min	2017.000000	0.000000	
25%	2021.000000	0.000000	
50%	2023.000000	1.000000	
75%	2023.000000	3.000000	

```
max          2023.000000           27.000000
            qtd_ferido_civil_ocorrencia  qtd_morto_agen_segur_ocorrencia \
count          1074.000000           1074.000000
mean          0.647114             0.023277
std           1.103775             0.168370
min           0.000000             0.000000
25%          0.000000             0.000000
50%          0.000000             0.000000
75%          1.000000             0.000000
max          13.000000            2.000000

            qtd_ferido_agen_segur_ocorrenci  faixa_horaria  longitude_ocorrencia \
count          1074.000000           1074.0           1.074000e+03
mean          0.048417             3.0           -2.447492e+15
std           0.243235             0.0           8.774596e+15
min           0.000000             3.0           -4.361184e+16
25%          0.000000             3.0           -4.344736e+11
50%          0.000000             3.0           -4.347818e+09
75%          0.000000             3.0           -4.333554e+08
max          3.000000             3.0           -4.337500e+04

            latitude_ocorrencia  mortos_totais
count          1.074000e+03       1074.000000
mean          -2.411600e+15      1.499069
std           6.846497e+15       2.186653
min           -2.295745e+16      0.000000
25%          -2.287062e+11      0.000000
50%          -2.286906e+09       1.000000
75%          -2.286267e+08       3.000000
max          -2.288130e+05      28.000000
```

[8 rows x 21 columns]

In [137]: df_cluster.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1074 entries, 0 to 1073
Data columns (total 34 columns):
 #   Column           Non-Null Count Dtype
 ---  -----
 0   presen_agen_segur_ocorrencia    1074 non-null   int64
 1   chacina_oc                     1074 non-null   int64
 2   ag_seguranca_vitima_oc        1074 non-null   int64
 3   bala_perdida_oc              1074 non-null   int64
 4   interior_residencia_oc      1074 non-null   int64
 5   imediacao_ensino_oc         1074 non-null   int64
 6   vitima_crianca_oc            1074 non-null   int64
 7   vitima_adolescente_oc       1074 non-null   int64
 8   vitima_idoso_oc              1074 non-null   int64
 9   dia_semana_ocorrencia_numero 1074 non-null   int64
 10  dia_ocorrencia               1074 non-null   int64
 11  mes_ocorrencia_numero       1074 non-null   int64
 12  ano_ocorrencia               1074 non-null   int64
 13  qtd_morto_civil_ocorrencia  1074 non-null   int64
 14  qtd_ferido_civil_ocorrencia 1074 non-null   int64
 15  qtd_morto_agen_segur_ocorrencia 1074 non-null   int64
 16  qtd_ferido_agen_segur_ocorrencia 1074 non-null   int64
 17  faixa_horaria                1074 non-null   int64
 18  longitude_ocorrencia         1074 non-null   int64
 19  latitude_ocorrencia          1074 non-null   int64
 20  mortos_totais                1074 non-null   int64
 21  motivo_Arrastão              1074 non-null   bool
 22  motivo_Ataque a civis       1074 non-null   bool
 23  motivo_Ação policial        1074 non-null   bool
 24  motivo_Briga                 1074 non-null   bool
 25  motivo_Disputa               1074 non-null   bool
 26  motivo_Execução              1074 non-null   bool
 27  motivo_Homicidio/Tentativa  1074 non-null   bool
 28  motivo_Não identificado     1074 non-null   bool
 29  motivo_Operação policial    1074 non-null   bool
 30  motivo_Sequestro             1074 non-null   bool
 31  motivo_Tentativa/Roubo       1074 non-null   bool
 32  motivo_Tentativa/Roubo a banco 1074 non-null   bool
 33  motivo_Tentativa/Roubo de cargas 1074 non-null   bool
dtypes: bool(13), int64(21)
memory usage: 190.0 KB
```

In [147...]

```
import matplotlib.pyplot as plt
import seaborn as sns

# Separando as colunas numericas
numeric_cols = df_cluster.select_dtypes(include=[np.number]).columns.tolist()

print("Colunas numéricas iniciais:")
print(numeric_cols)

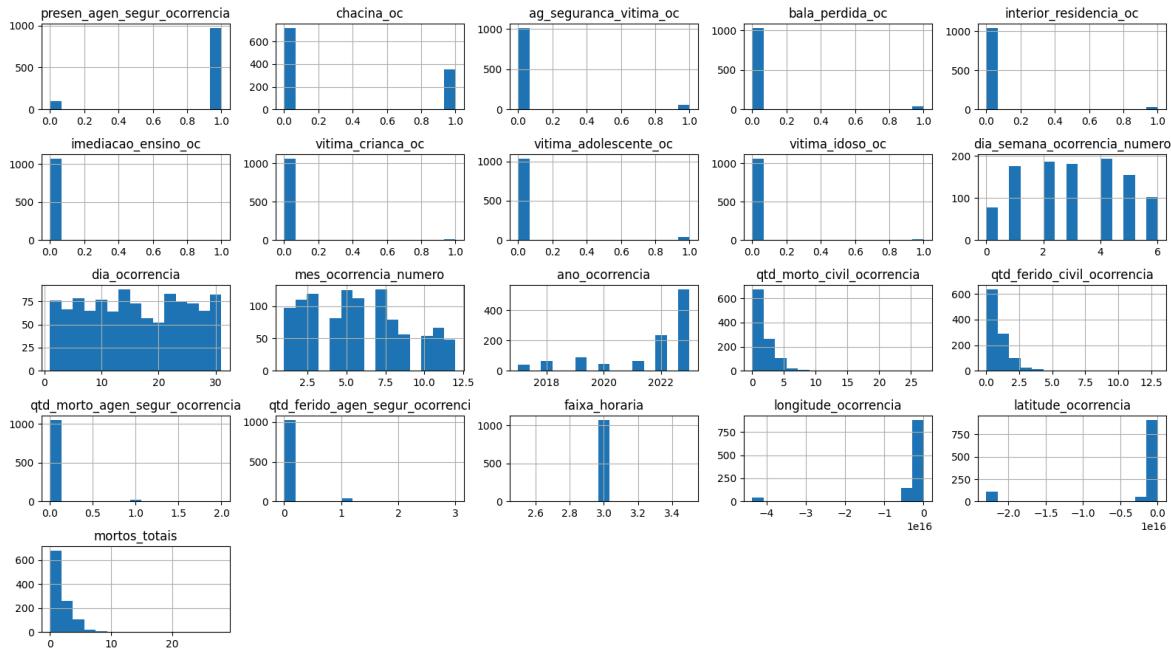
# Histogramas
df_cluster.hist(figsize=(16, 9), bins=15)
plt.tight_layout()
plt.show()

# Boxplots
for col in numeric_cols:
    plt.figure(figsize=(16, 9))
    sns.boxplot(y=df_cluster[col])
```

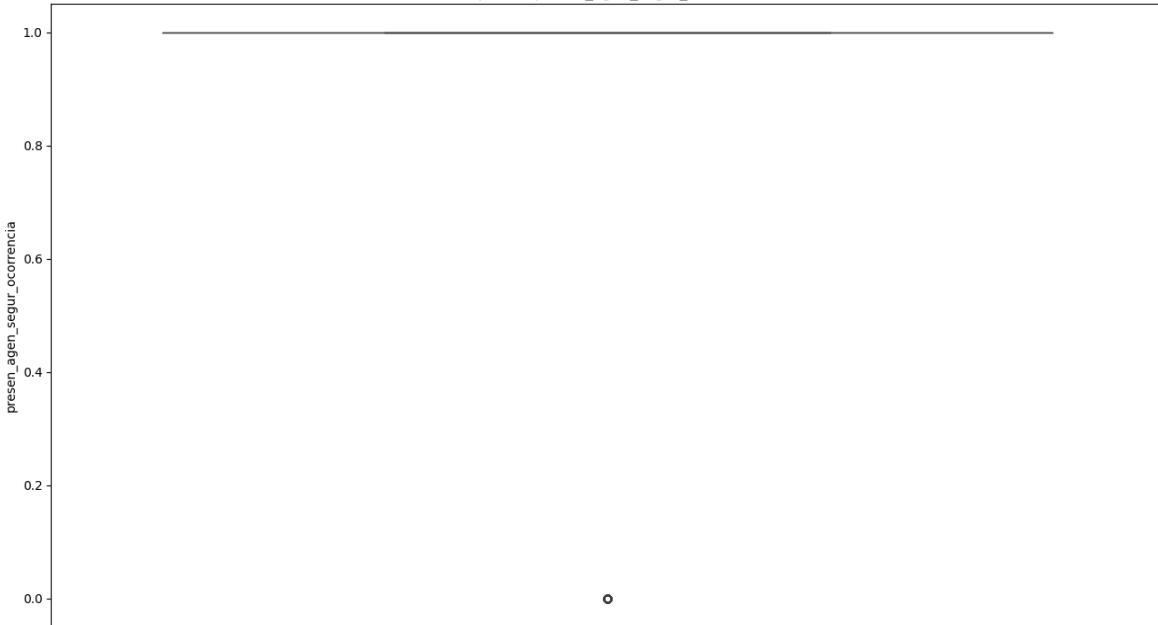
```
plt.title(f"Boxplot de {col}")
plt.show()
```

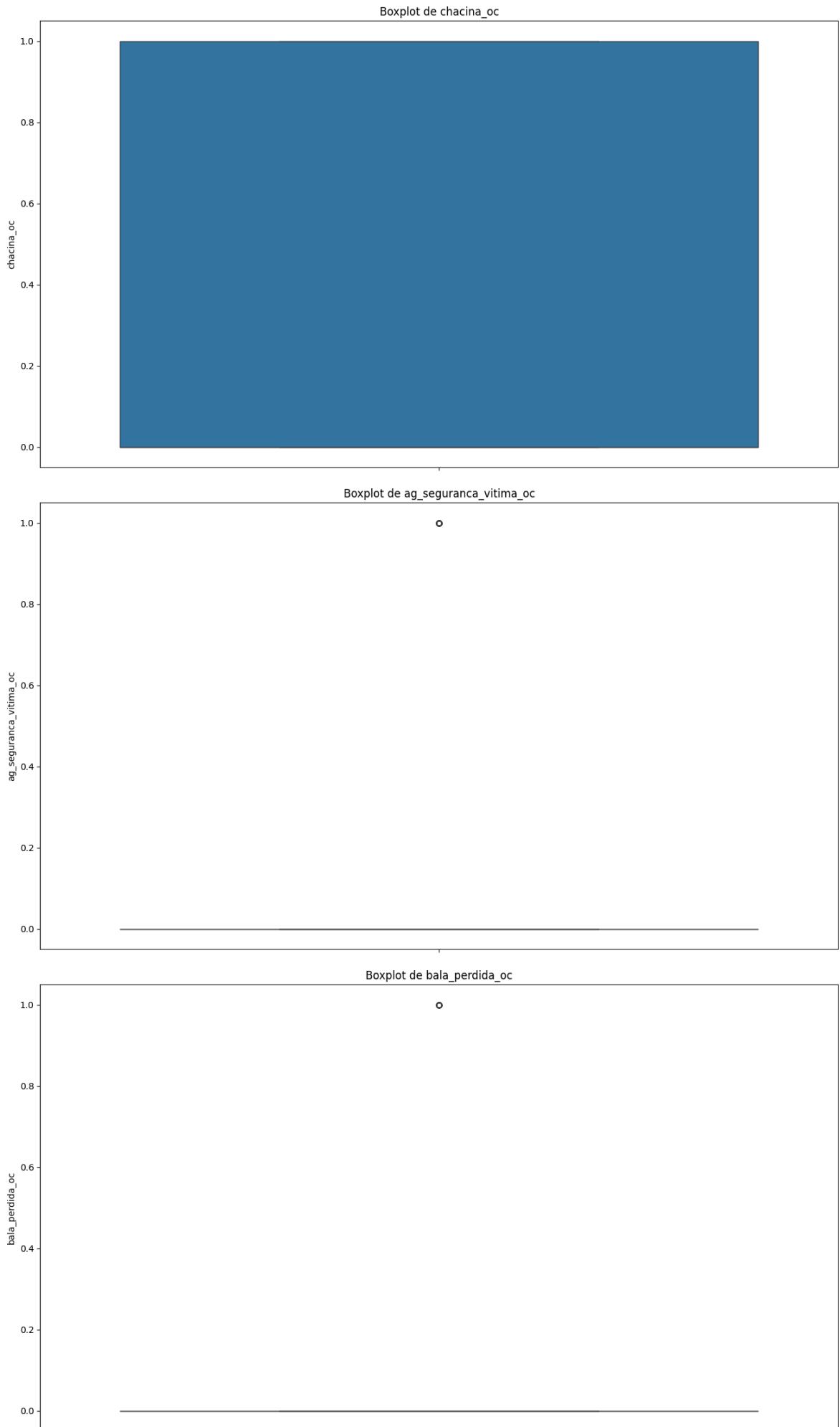
Colunas numéricas iniciais:

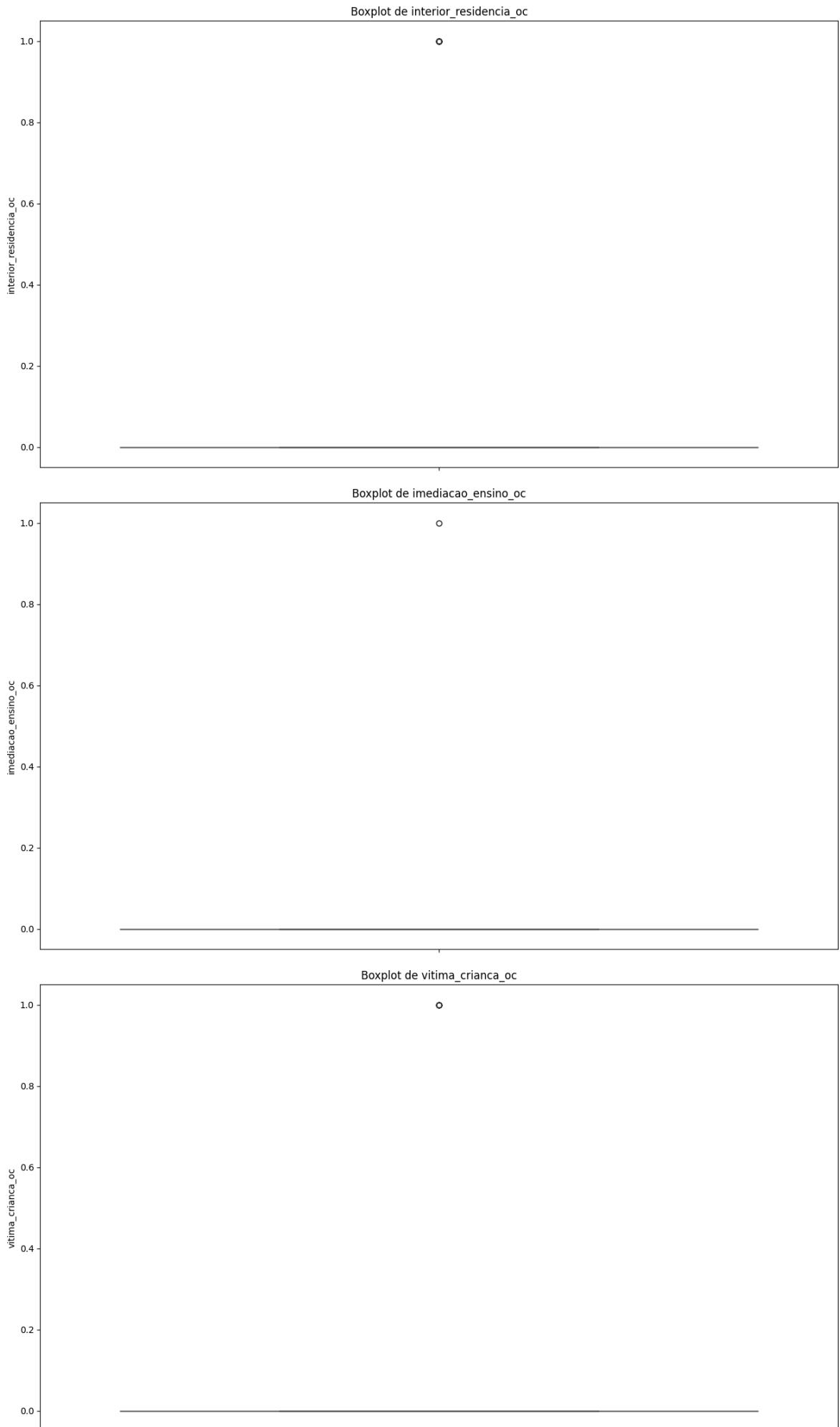
```
['presen_agen_segur_ocorrencia', 'chacina_oc', 'ag_seguranca_vitima_oc', 'bala_perdida_oc', 'interior_residencia_oc', 'imediacao_ensino_oc', 'vitima_crianca_oc', 'vitima_adolescente_oc', 'vitima_idoso_oc', 'dia_semana_ocorrencia_numero', 'dia_ocorrencia', 'mes_ocorrencia_numero', 'ano_ocorrencia', 'qtd_morto_civil_ocorrencia', 'qtd_ferido_civil_ocorrencia', 'qtd_morto_agr_segur_ocorrencia', 'qtd_ferido_agr_segur_ocorrencia', 'faixa_horaria', 'longitude_ocorrencia', 'latitude_ocorrencia', 'mortos_totais']
```



Boxplot de presen_agen_segur_ocorrencia



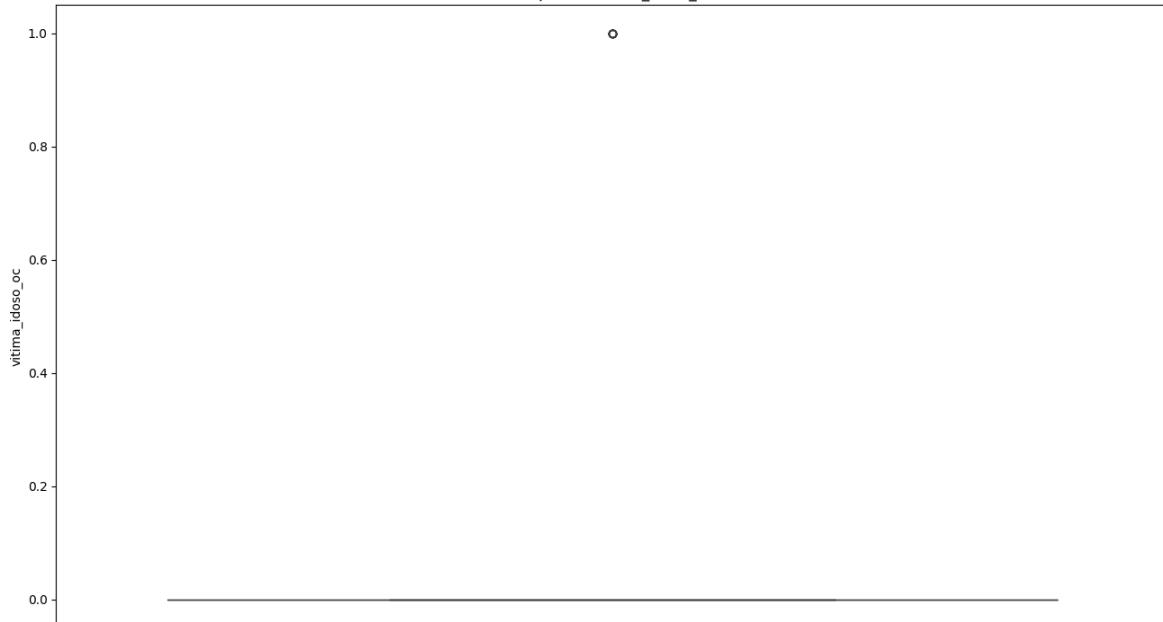




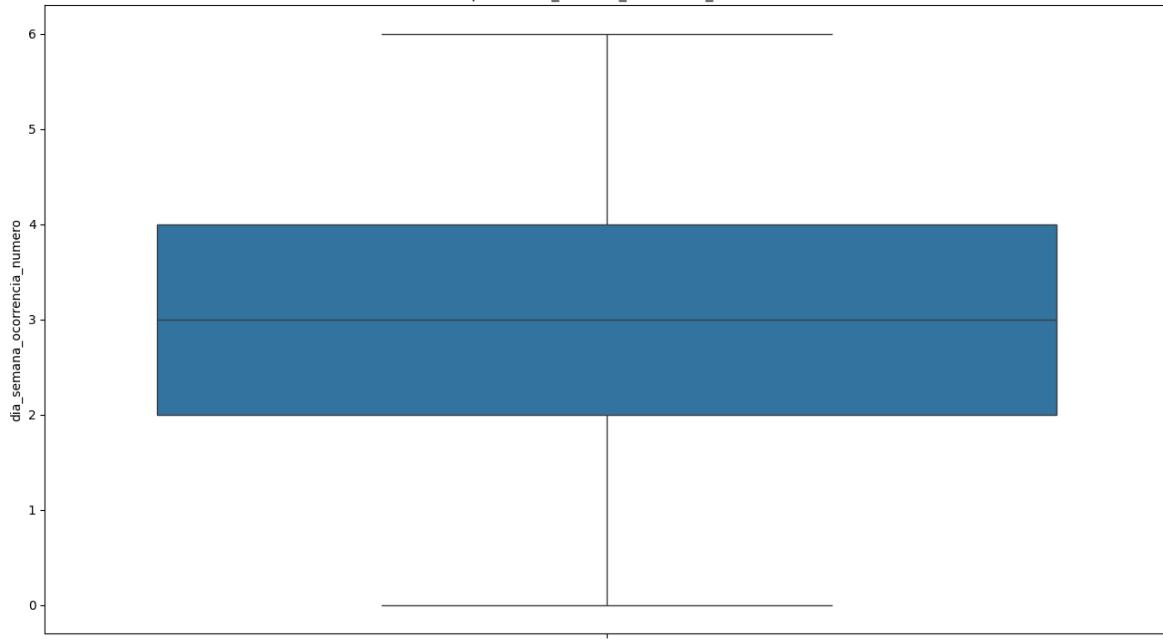
Boxplot de vitima_adolescente_oc

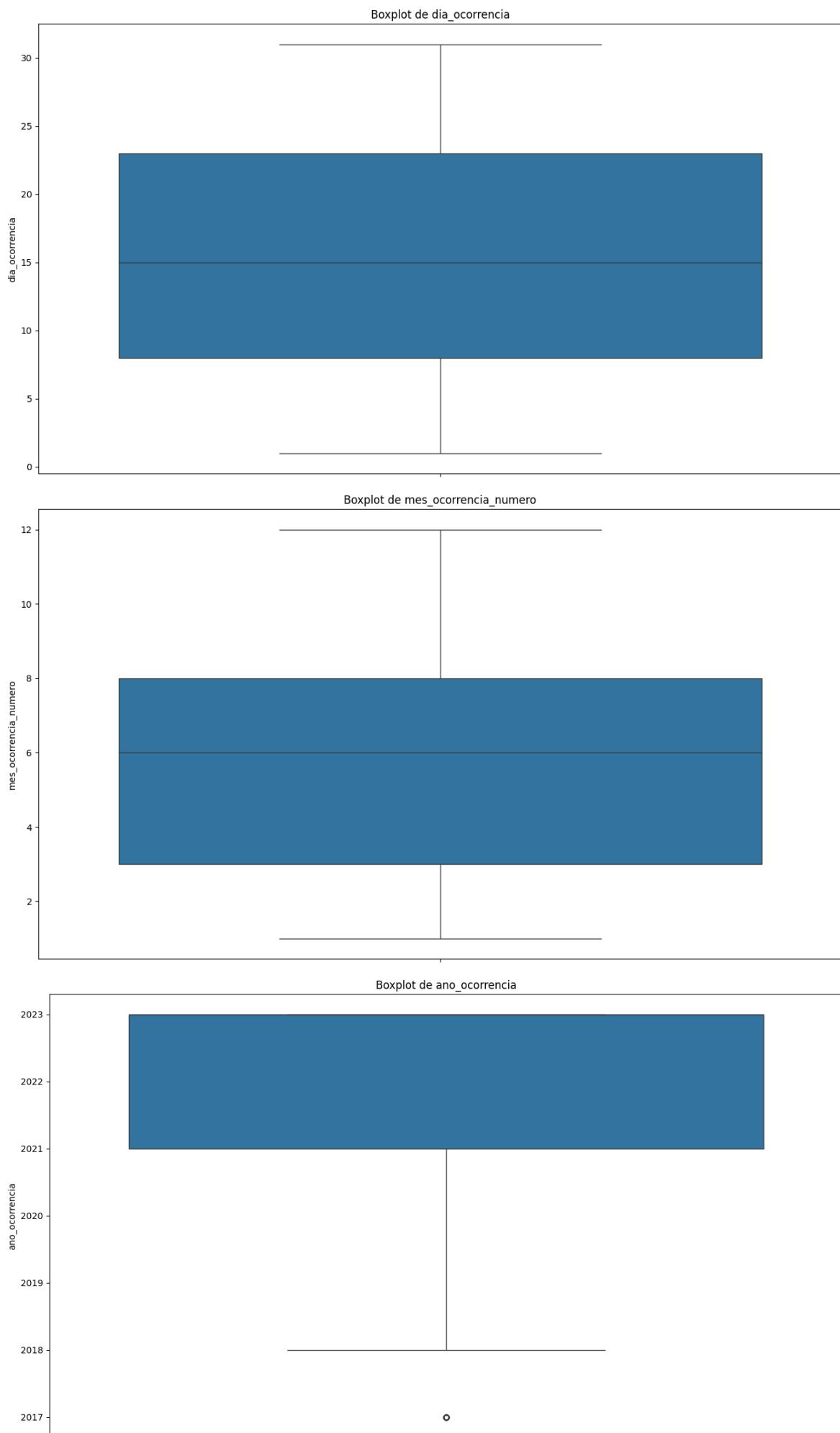


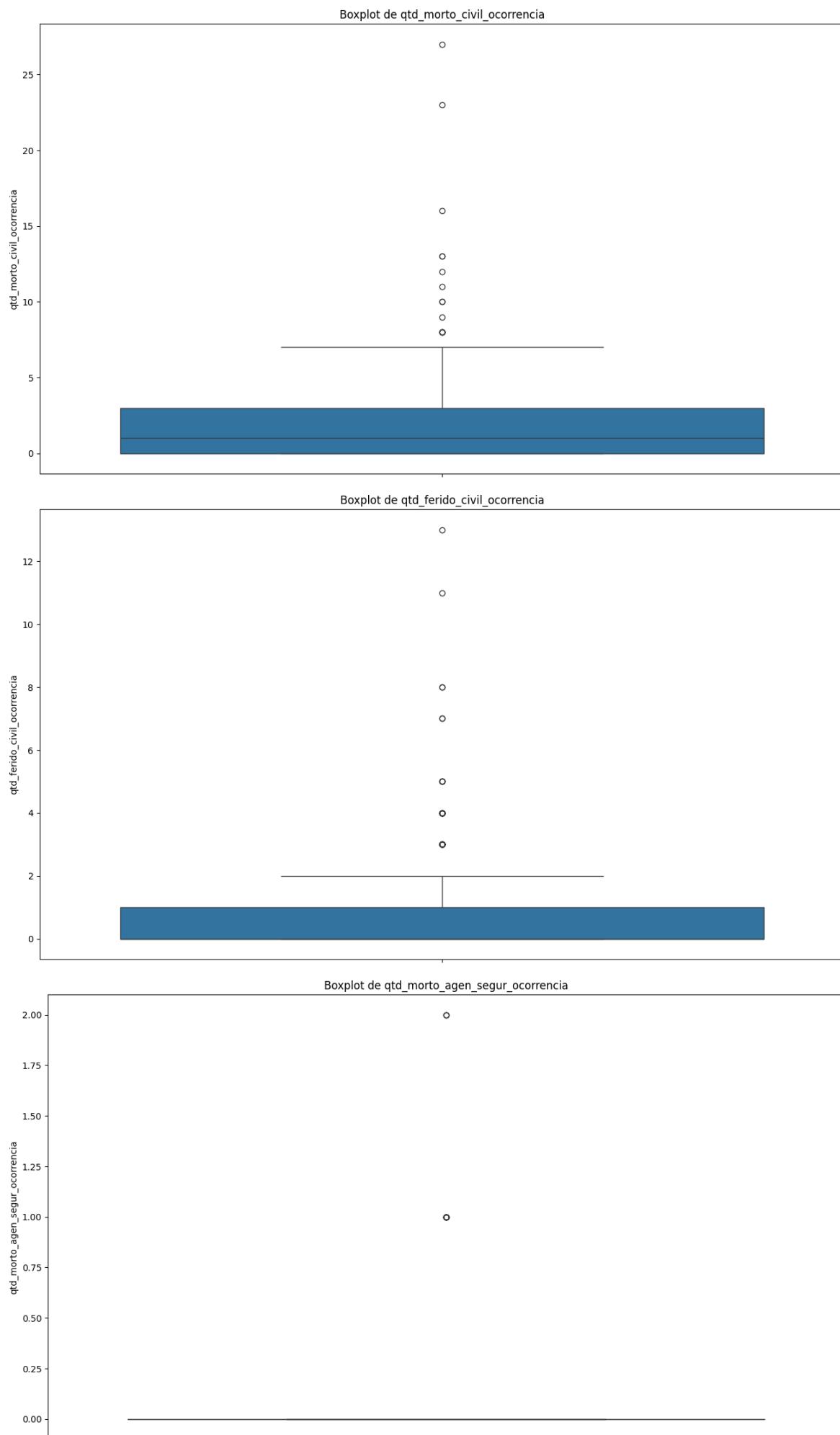
Boxplot de vitima_idoso_oc



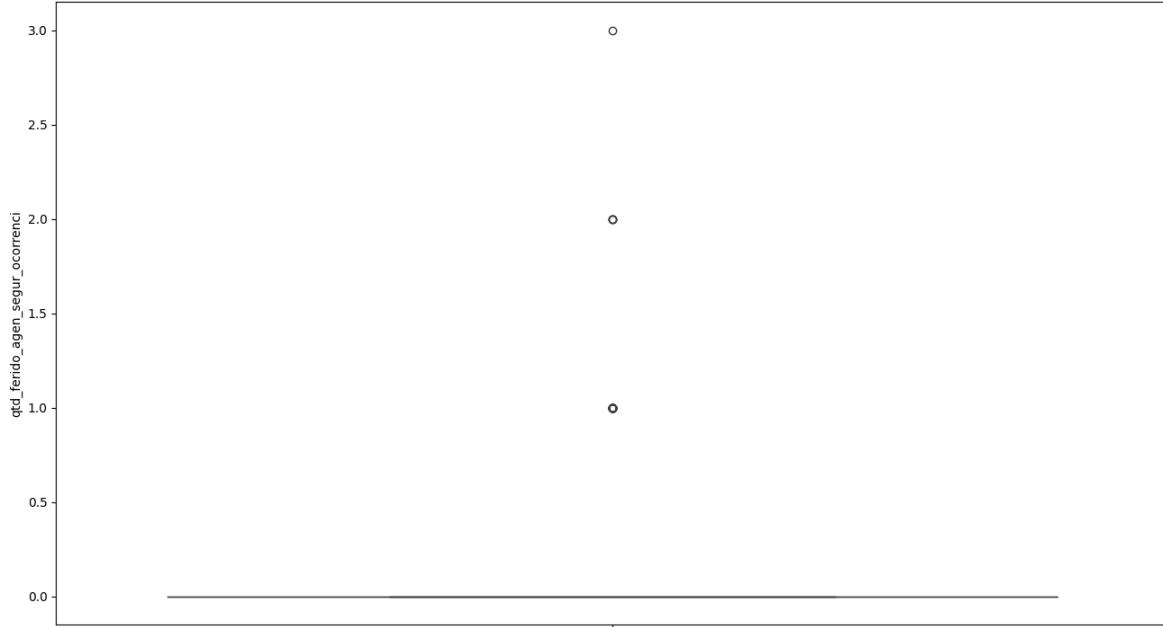
Boxplot de dia_semana_ocorrencia_numero







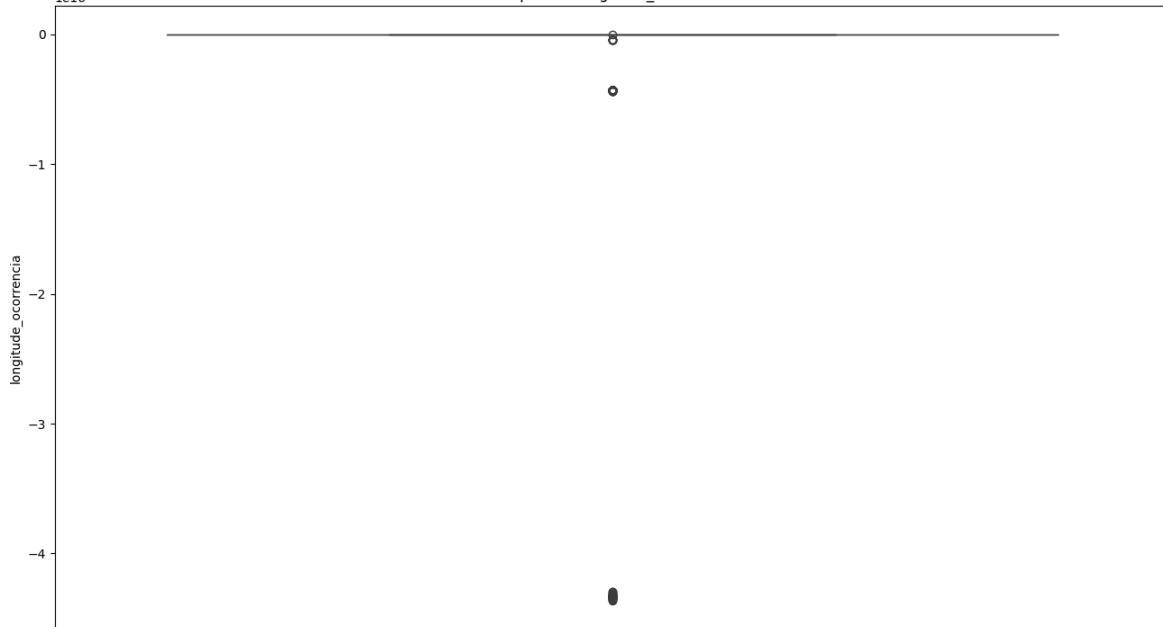
Boxplot de qtd_ferido_agen_segur_ocorrencia

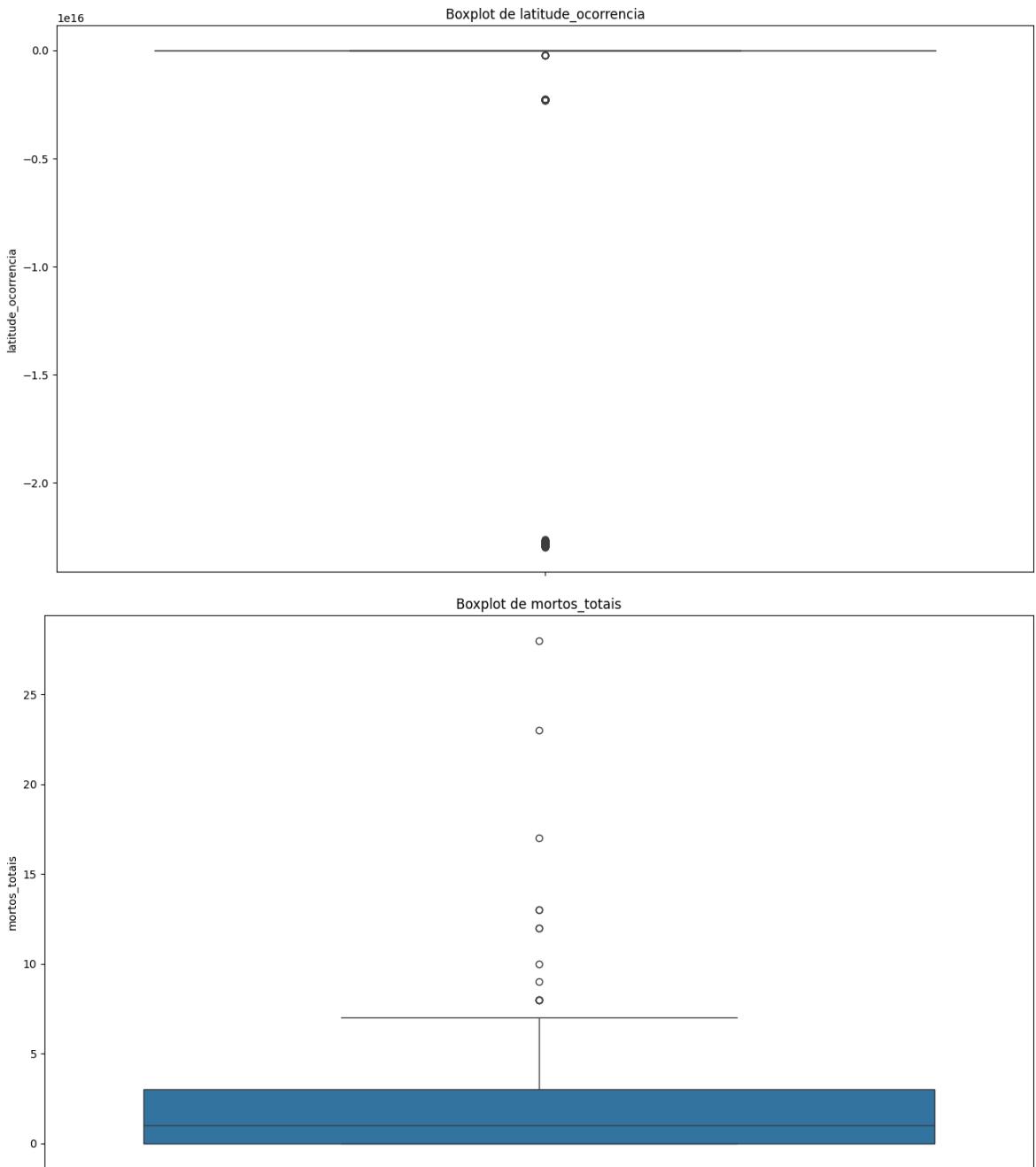


Boxplot de faixa_horaria



Boxplot de longitude_ocorrencia





In [208...]

```

from sklearn.preprocessing import QuantileTransformer

# Extraímos só essas colunas numa matriz
X = df_cluster[numeric_cols].values

# Criamos o transformer
# n_quantiles pode ser menor que o número de Linhas; 1000 é comum.
# 'output_distribution' pode ser 'uniform' ou 'normal'
qt = QuantileTransformer(n_quantiles=100, output_distribution='normal', random_state=42)

# Fit e transform
X_trans = qt.fit_transform(X)

# Substituímos no DataFrame
df_transformed = df_cluster.copy()
df_transformed[numeric_cols] = X_trans

# Agora, df_transformed tem as colunas transformadas

```

```
# Se vc quiser conferir a nova distribuição:  
print(df_transformed[numERIC_COLS].describe())
```

	presen_agen_segur_ocorrencia	chacina_oc	ag_seguranca_vitima_oc	\
count	1074.000000	1074.000000	1074.000000	1074.000000
mean	4.211754	-1.762159		-4.599042
std	3.050066	4.893895		2.426398
min	-5.199338	-5.199338		-5.199338
25%	5.199338	-5.199338		-5.199338
50%	5.199338	-5.199338		-5.199338
75%	5.199338	5.199338		-5.199338
max	5.199338	5.199338		5.199338
	bala_perdida_oc	interior_residencia_oc	immediacao_ensino_oc	\
count	1074.000000	1074.000000	1074.000000	1074.000000
mean	-4.773321	-4.899190		-5.179973
std	2.062153	1.741803		0.448527
min	-5.199338	-5.199338		-5.199338
25%	-5.199338	-5.199338		-5.199338
50%	-5.199338	-5.199338		-5.199338
75%	-5.199338	-5.199338		-5.199338
max	5.199338	5.199338		5.199338
	vitima_criancა_oc	vitima_adolescente_oc	vitima_idoso_oc	\
count	1074.000000	1074.000000	1074.000000	1074.000000
mean	-5.063787	-4.821732		-5.092833
std	1.180030	1.946160		1.047464
min	-5.199338	-5.199338		-5.199338
25%	-5.199338	-5.199338		-5.199338
50%	-5.199338	-5.199338		-5.199338
75%	-5.199338	-5.199338		-5.199338
max	5.199338	5.199338		5.199338
	dia_semana_ocorrencia_numero	...	mes_ocorrencia_numero	\
count	1074.000000	...	1074.000000	
mean	0.093675	...	-0.167400	
std	2.219424	...	2.025284	
min	-5.199338	...	-5.199338	
25%	-0.458679	...	-0.682458	
50%	-0.012660	...	0.126937	
75%	0.444660	...	0.698526	
max	5.199338	...	5.199338	
	ano_ocorrencia	qtd_morto_civil_ocorrencia	\	
count	1074.000000	1074.000000		
mean	2.108247	-2.161489		
std	3.232427	3.025576		
min	-5.199338	-5.199338		
25%	-0.682458	-5.199338		
50%	5.199338	0.152506		
75%	5.199338	0.747859		
max	5.199338	5.199338		
	qtd_ferido_civil_ocorrencia	qtd_morto_agen_segur_ocorrencia	\	
count	1074.000000	1074.000000		
mean	-2.695322		-5.037222	
std	3.038425		1.130438	
min	-5.199338		-5.199338	
25%	-5.199338		-5.199338	
50%	-5.199338		-5.199338	
75%	0.604585		-5.199338	
max	5.199338		5.199338	

```

      qtd_ferido_agen_segur_ocorrenci  faixa_horaria  longitude_ocorrencia \
count              1074.000000    1074.000000    1074.000000
mean             -4.887568     -5.199338     0.006557
std               1.478107     0.000000     1.031938
min             -5.199338     -5.199338    -5.199338
25%            -5.199338     -5.199338    -0.673977
50%            -5.199338     -5.199338    -0.005194
75%            -5.199338     -5.199338     0.673799
max              5.199338     -5.199338     5.199338

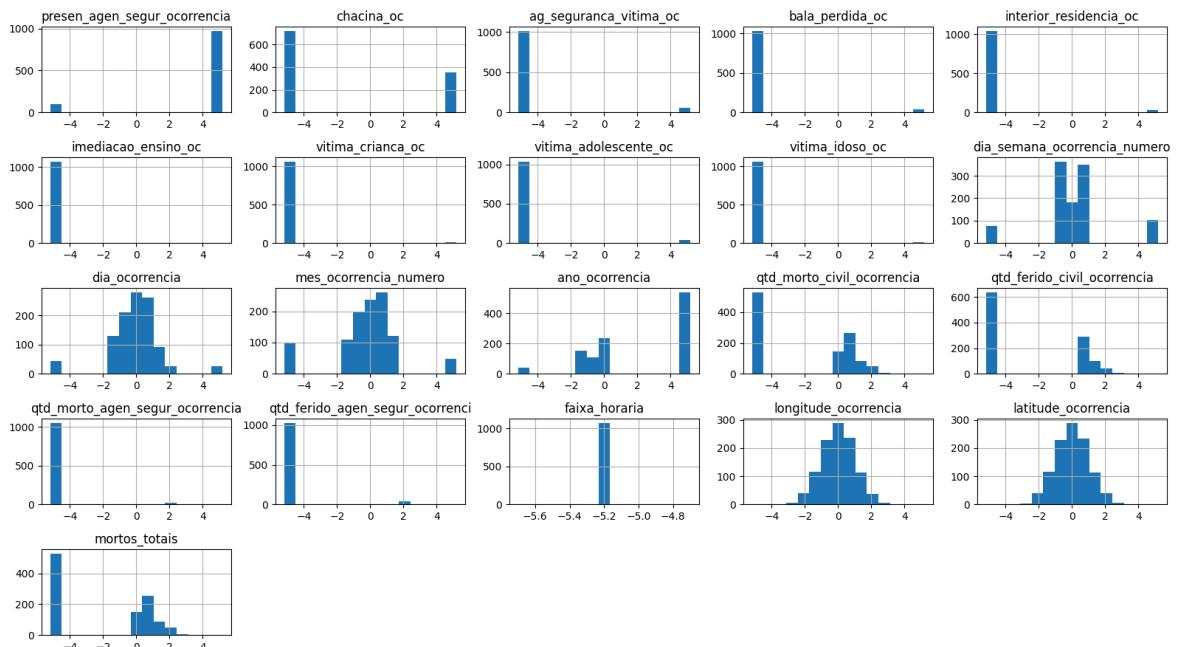
      latitude_ocorrencia  mortos_totais
count              1074.000000    1074.000000
mean              0.006373    -2.158984
std               1.032580     3.022421
min             -5.199338     -5.199338
25%            -0.673613     -5.199338
50%            -0.006184     0.152506
75%              0.668001     0.747859
max              5.199338     5.199338

```

[8 rows x 21 columns]

In [204...]

```
# Histogramas
df_transformed.hist(figsize=(16, 9), bins=15)
plt.tight_layout()
plt.show()
```



In [209...]

```
df_transformed
```

Out[209...]

	presen_agen_segur_ocorrencia	chacina_oc	ag_seguranca_vitima_oc	bala_perdida_o
0	-5.199338	5.199338	-5.199338	-5.19933
1	5.199338	5.199338	5.199338	-5.19933
2	5.199338	-5.199338	-5.199338	-5.19933
3	5.199338	-5.199338	-5.199338	-5.19933
4	5.199338	5.199338	-5.199338	-5.19933
...
1069	-5.199338	5.199338	-5.199338	-5.19933
1070	5.199338	-5.199338	-5.199338	-5.19933
1071	5.199338	-5.199338	-5.199338	-5.19933
1072	5.199338	-5.199338	-5.199338	-5.19933
1073	-5.199338	-5.199338	-5.199338	-5.19933

1074 rows × 34 columns

In [213...]

```

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# (1) Junte colunas numéricas transformadas com colunas categóricas encodadas
#df_final = pd.concat([df_transformed, df_binary, df_onehot], axis=1)

# (2) (Opcional) rodar StandardScaler se achar necessário
sc = StandardScaler()
df_final_scaled = sc.fit_transform(df_transformed[numeric_cols])

# (3) K-Means

X = df_transformed[numeric_cols].values # ou df_final_scaled

scores = []
K_range = range(2, 10)

for k in K_range:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(df_final_scaled)
    labels = km.labels_
    sil = silhouette_score(df_final_scaled, labels)
    scores.append(sil)

best_k = K_range[scores.index(max(scores))]
print("Melhor k (pelo índice de silhueta):", best_k)

```

Melhor k (pelo índice de silhueta): 2

In [214...]

```

import matplotlib.cm as cm
from sklearn.metrics import silhouette_samples

```

```

def plot_silhouette_analysis(X, range_n_clusters):
    """
        X           -> Dados (matriz n_amostras x n_features)
        range_n_clusters -> lista com valores de k para testar no KMeans
    """
    for n_clusters in range_n_clusters:
        # Cria subplots
        fig, ax1 = plt.subplots(1, 1)
        fig.set_size_inches(7, 5)

        # Instancia KMeans e obtém os rótulos (labels)
        clusterer = KMeans(n_clusters=n_clusters, random_state=42)
        cluster_labels = clusterer.fit_predict(X)

        # Cálculo do silhouette médio e do coeficiente individual de cada amostra
        silhouette_avg = silhouette_score(X, cluster_labels)
        sample_silhouette_values = silhouette_samples(X, cluster_labels)

        # Também podemos obter desvio-padrão e outras métricas de magnitude
        sil_std = np.std(sample_silhouette_values)
        sil_min = sample_silhouette_values.min()
        sil_max = sample_silhouette_values.max()

        print(f"\nPara n_clusters = {n_clusters}:")
        print(f" - Silhouette médio = {silhouette_avg:.4f}")
        print(f" - Silhouette std = {sil_std:.4f}")
        print(f" - Silhouette min/max = {sil_min:.4f}/{sil_max:.4f}")

        # Plot de silhueta
        y_lower = 10
        for i in range(n_clusters):
            # Coleta os valores de silhueta do cluster i
            ith_cluster_sil_values = sample_silhouette_values[cluster_labels == i]
            ith_cluster_sil_values.sort()

            size_cluster_i = len(ith_cluster_sil_values)
            y_upper = y_lower + size_cluster_i

            # Escolhe cor para esse cluster
            color = cm.nipy_spectral(float(i) / n_clusters)
            ax1.fill_betweenx(
                np.arange(y_lower, y_upper),
                0,
                ith_cluster_sil_values,
                facecolor=color,
                edgecolor=color,
                alpha=0.7
            )

            # Label no gráfico: cluster i
            ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

            # Atualiza y_lower para o próximo cluster
            y_lower = y_upper + 10

        ax1.set_title(f"Silhouette plot para k={n_clusters}. Média = {silhouette_avg:.4f}")
        ax1.set_xlabel("Coeficiente de Silhueta")
        ax1.set_ylabel("Índice de amostra")

        ax1.set_xlim([-0.1, 1]) # Faixa do silhouette

```

```

    ax1.set_yticks([])
    # Linha vertical indicando a média
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

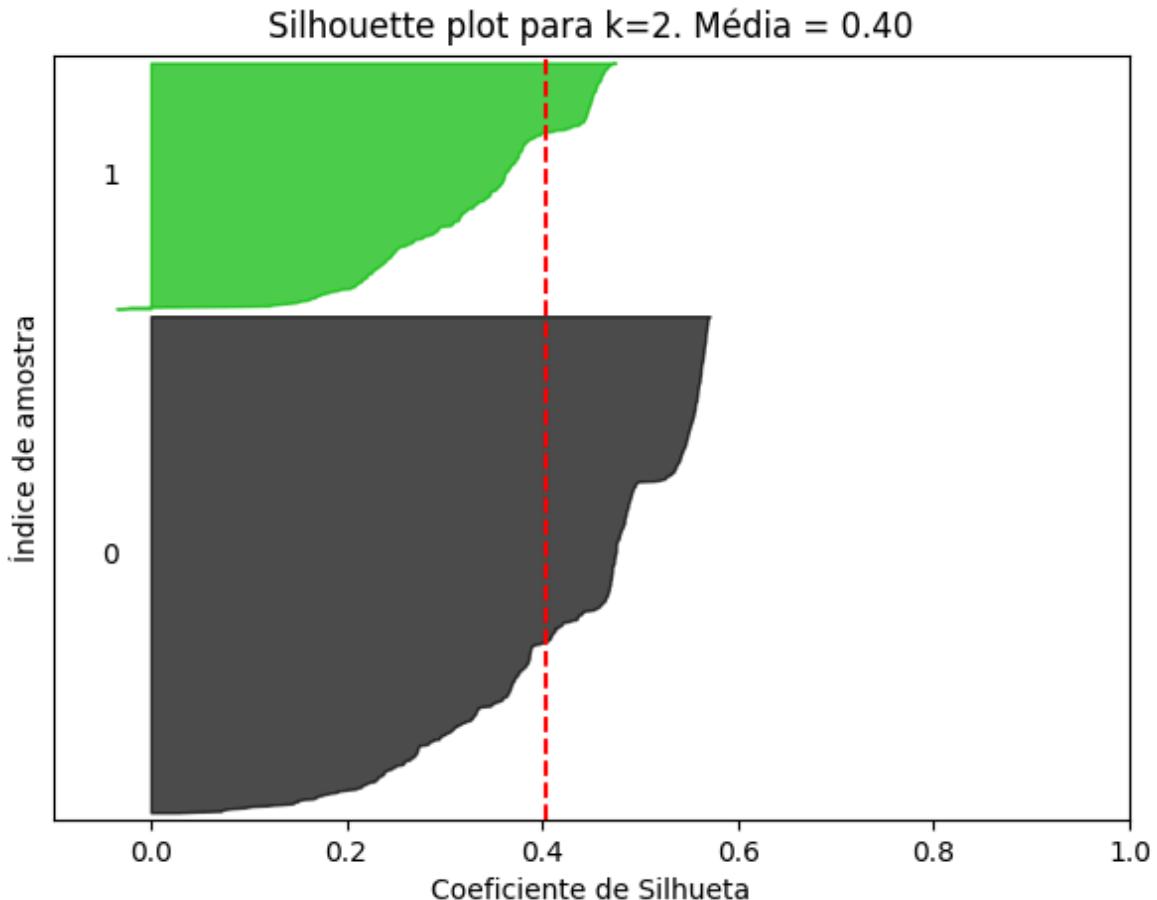
plt.show()

# chamada da função de silhueta:
plot_silhouette_analysis(X, K_range)

```

Para n_clusters = 2:

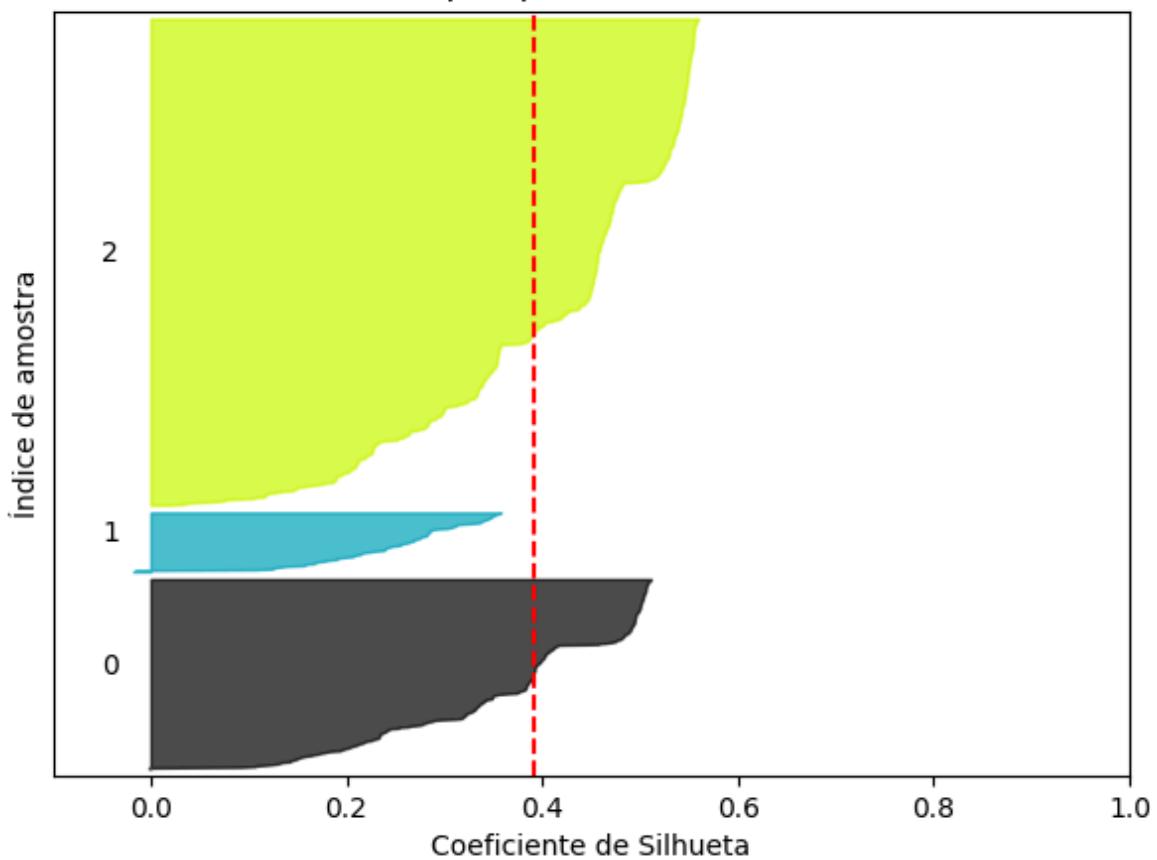
- Silhouette médio = 0.4036
- Silhouette std = 0.1230
- Silhouette min/max = -0.0354/0.5710



Para n_clusters = 3:

- Silhouette médio = 0.3924
- Silhouette std = 0.1306
- Silhouette min/max = -0.0176/0.5595

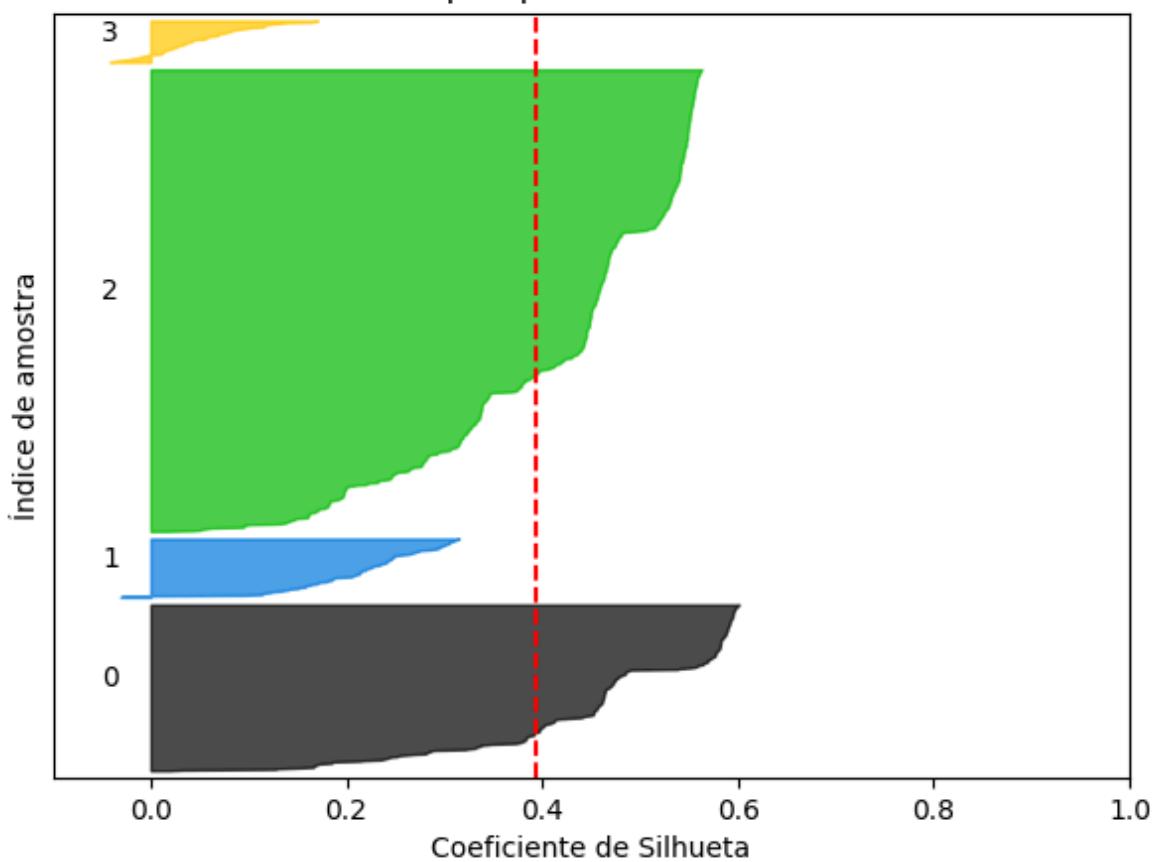
Silhouette plot para k=3. Média = 0.39



Para n_clusters = 4:

- Silhouette médio = 0.3930
- Silhouette std = 0.1590
- Silhouette min/max = -0.0417/0.6010

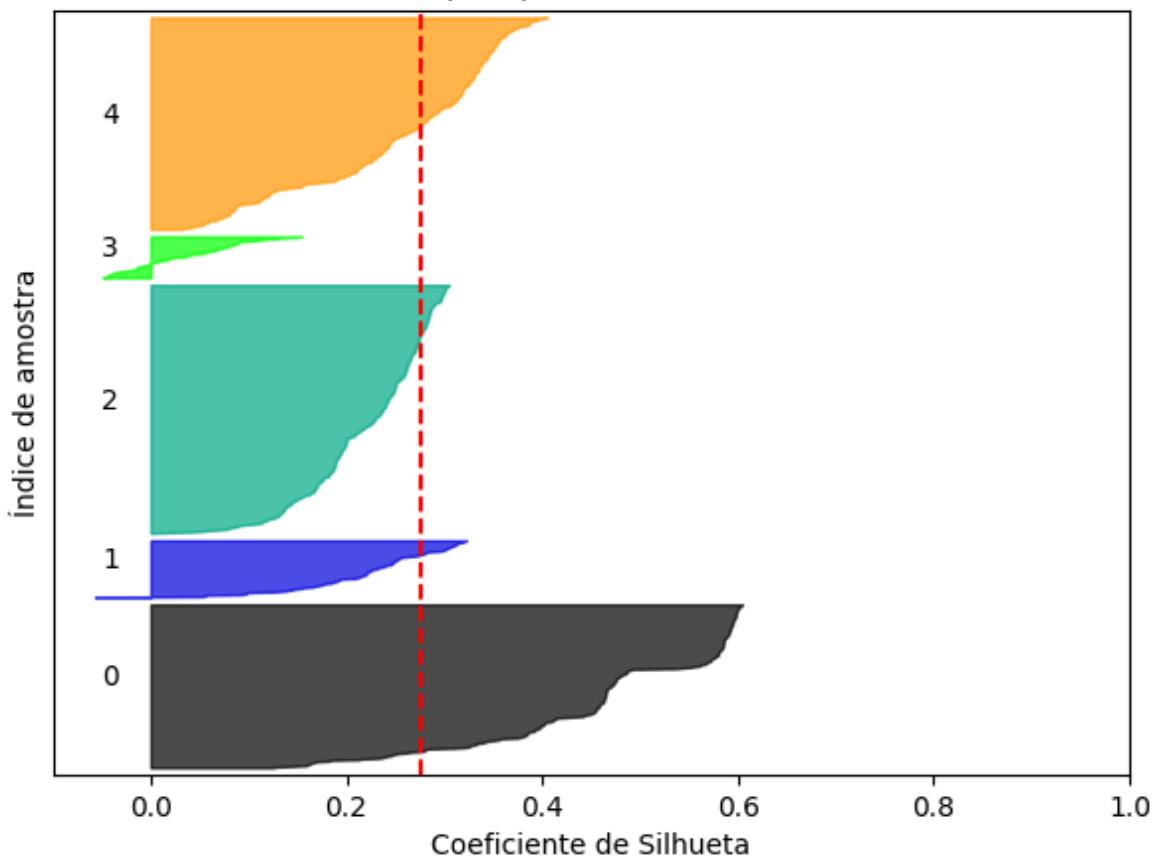
Silhouette plot para k=4. Média = 0.39



Para n_clusters = 5:

- Silhouette médio = 0.2756
- Silhouette std = 0.1454
- Silhouette min/max = -0.0569/0.6043

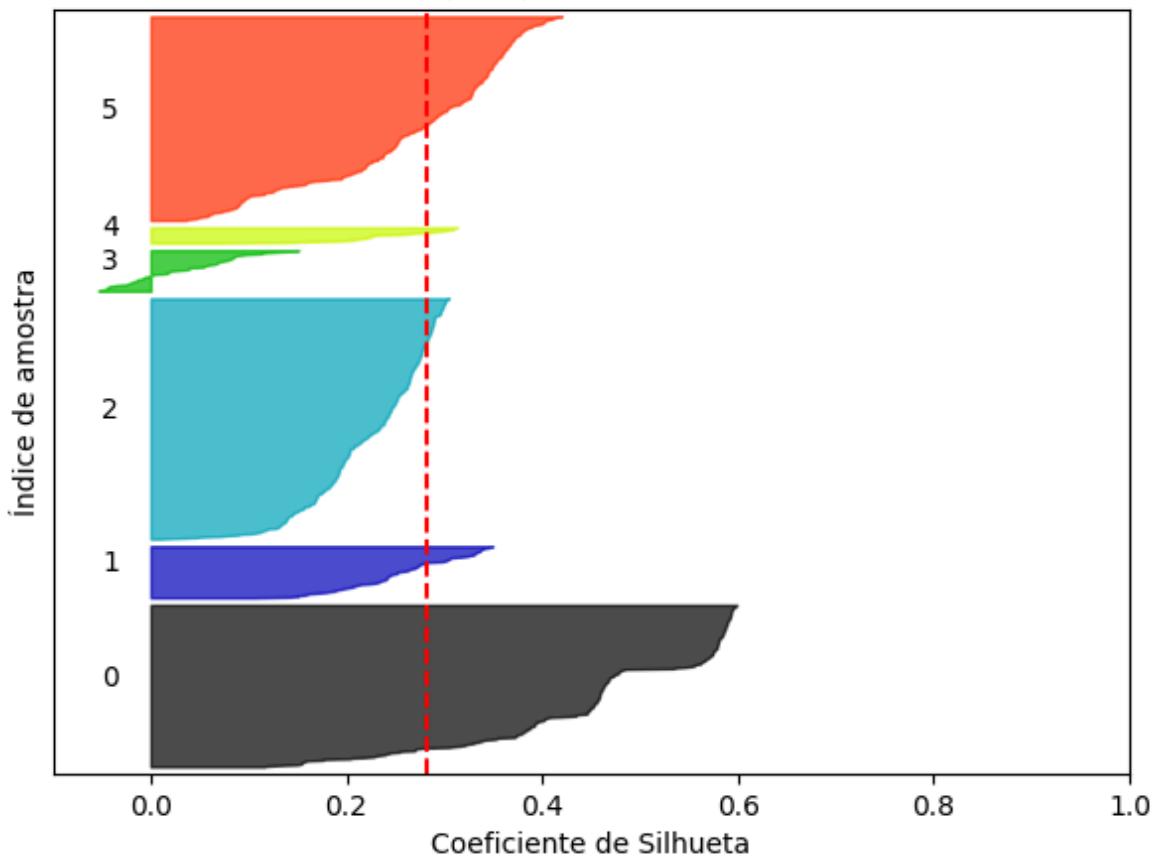
Silhouette plot para k=5. Média = 0.28



Para n_clusters = 6:

- Silhouette médio = 0.2820
- Silhouette std = 0.1413
- Silhouette min/max = -0.0533/0.5987

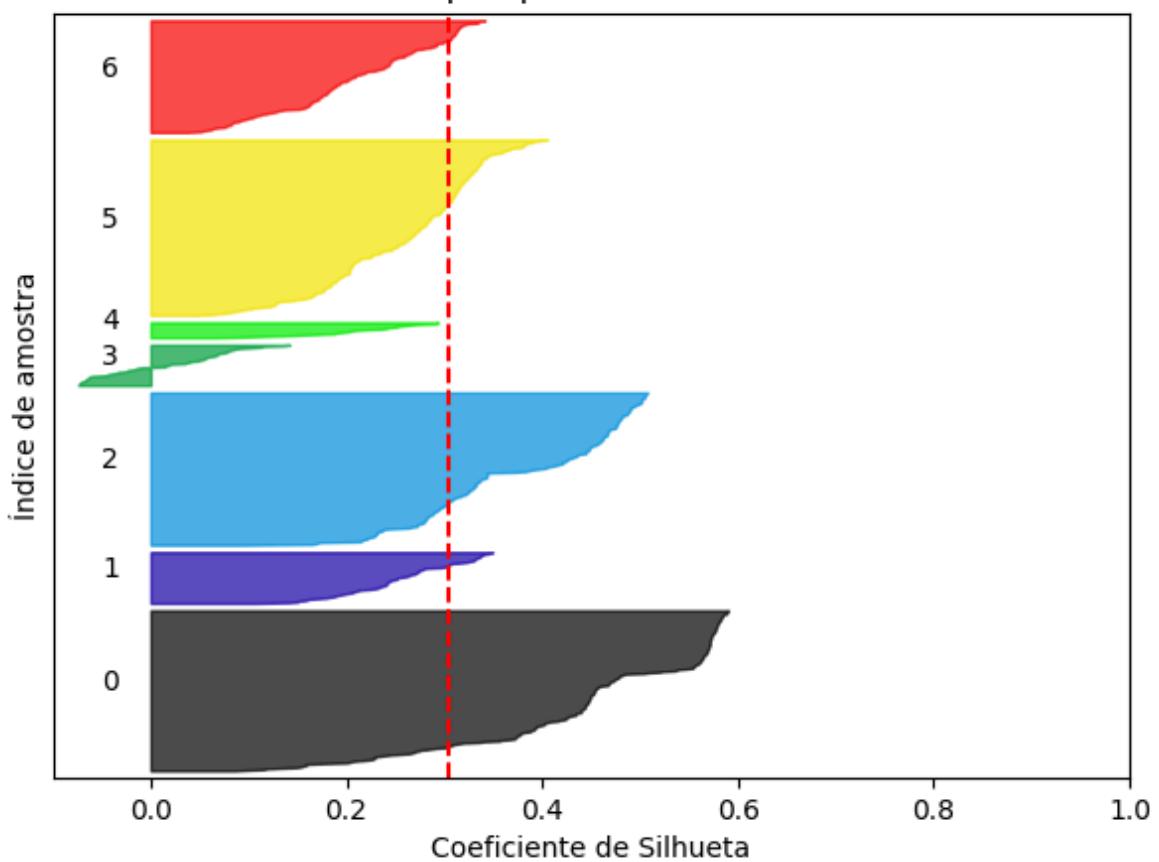
Silhouette plot para k=6. Média = 0.28



Para n_clusters = 7:

- Silhouette médio = 0.3049
- Silhouette std = 0.1473
- Silhouette min/max = -0.0739/0.5902

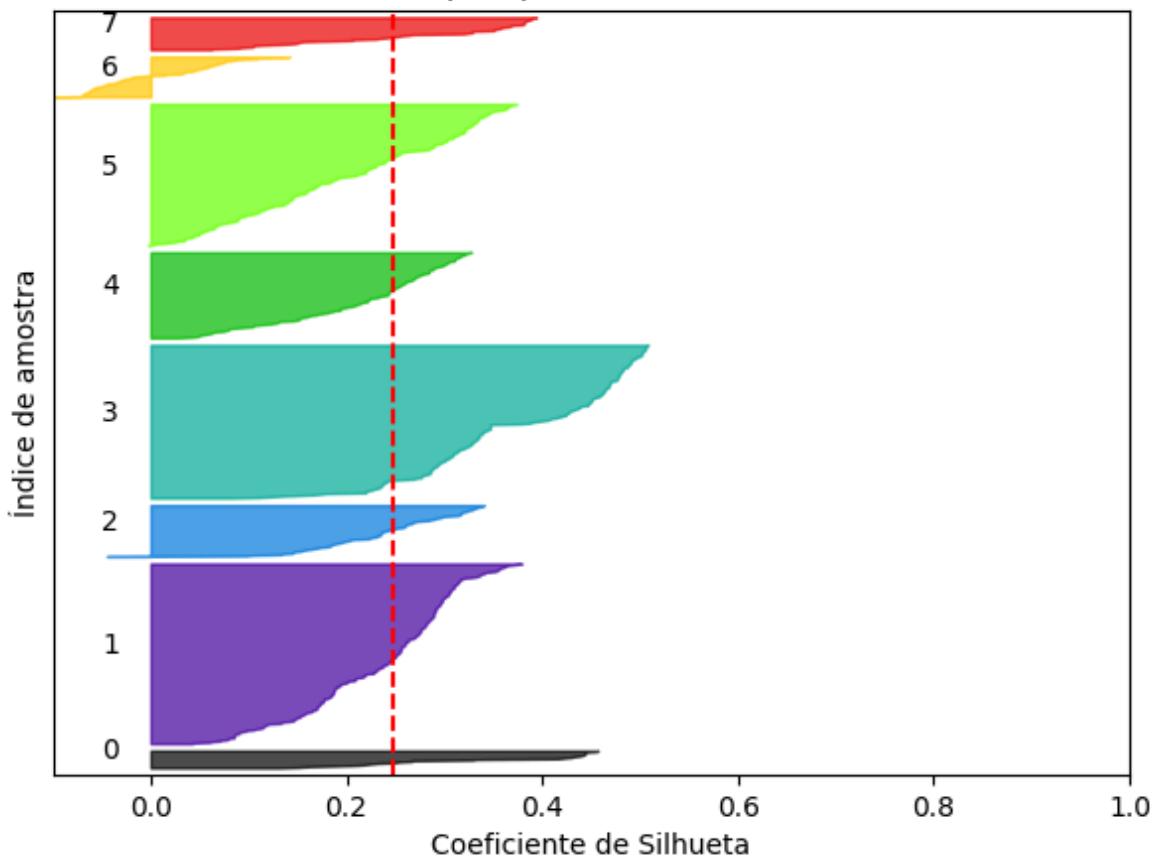
Silhouette plot para k=7. Média = 0.30



Para n_clusters = 8:

- Silhouette médio = 0.2483
- Silhouette std = 0.1266
- Silhouette min/max = -0.1033/0.5080

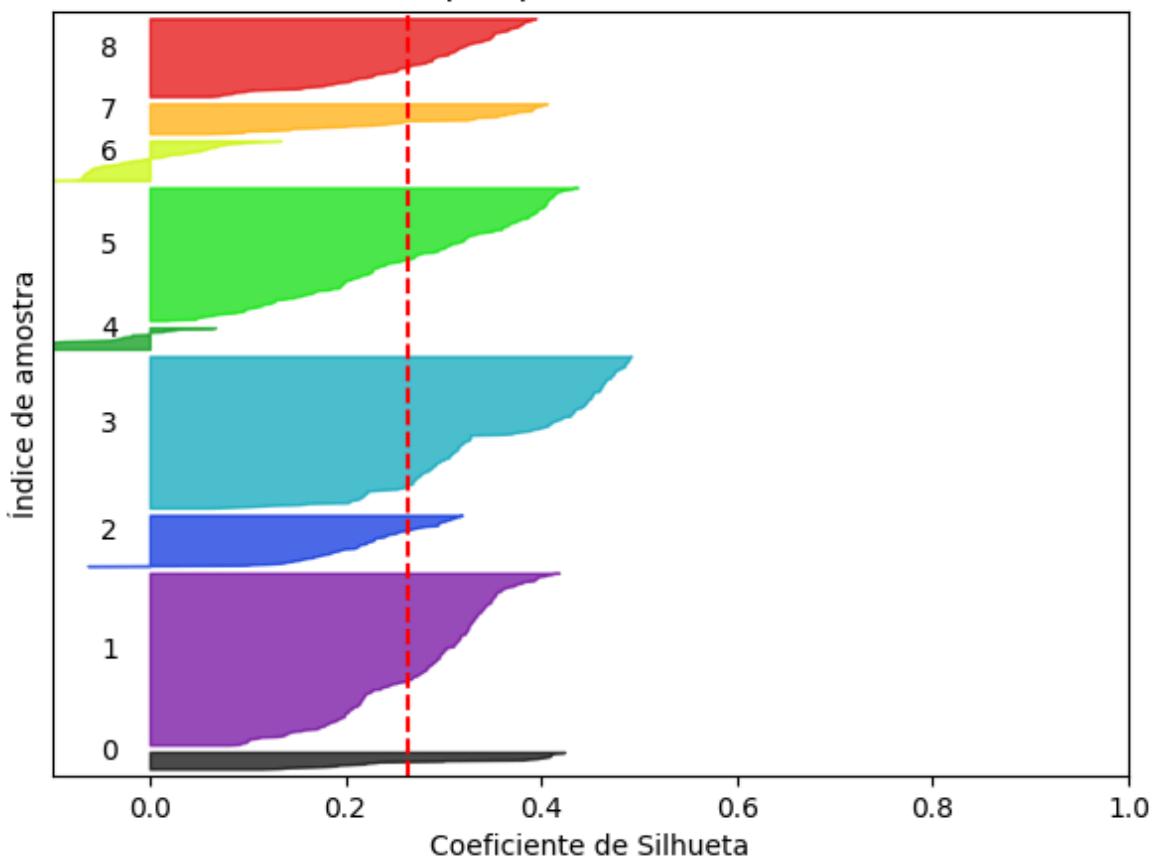
Silhouette plot para k=8. Média = 0.25



Para n_clusters = 9:

- Silhouette médio = 0.2634
- Silhouette std = 0.1340
- Silhouette min/max = -0.1523/0.4919

Silhouette plot para k=9. Média = 0.26



```
In [ ]: from sklearn.decomposition import PCA

# X é um array de forma
k = 2
kmeans = KMeans(n_clusters=k, random_state=42)
labels = kmeans.fit_predict(X)

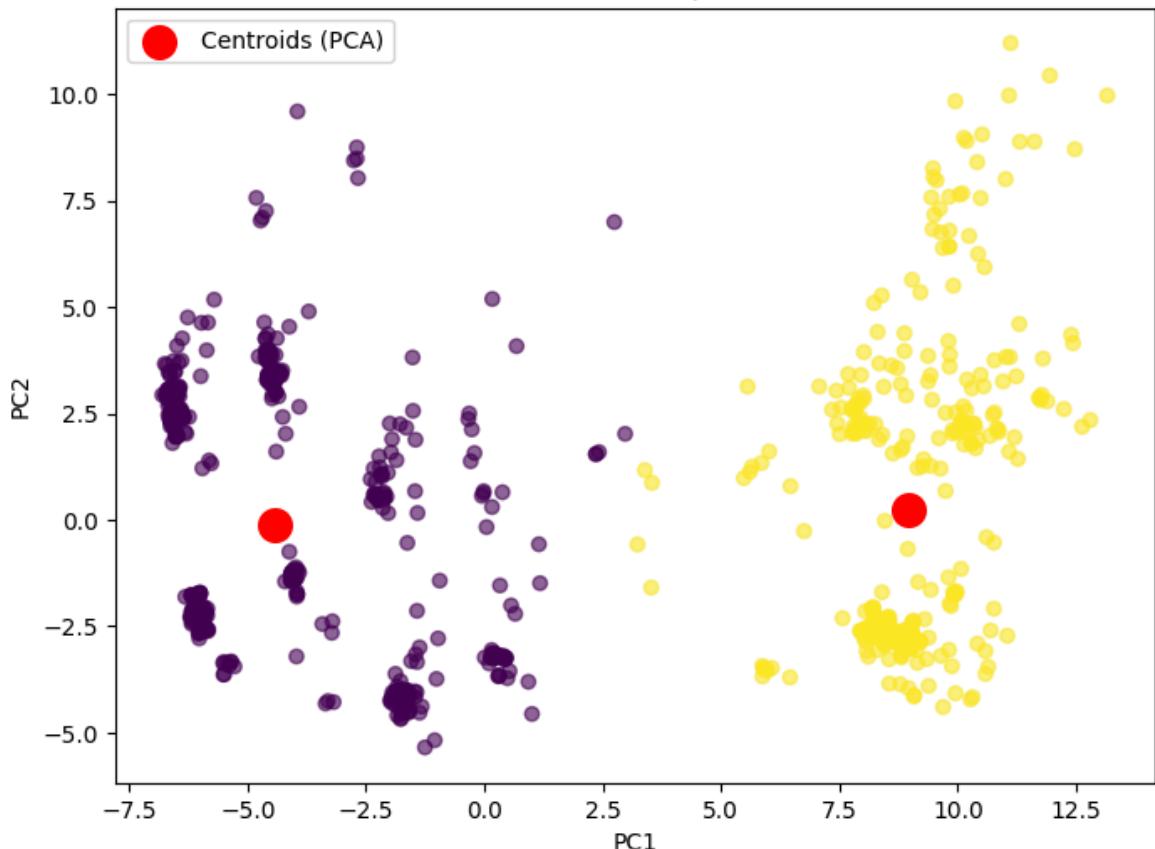
# Reduz para 2 componentes principais
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X) # shape (n_amostras, 2)

# Plotando em 2D
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', alpha=0.6)

# Plotar centróides "aproximados" em PCA (não exatamente os centróides no espaço
centers = kmeans.cluster_centers_ # centróides no espaço original
centers_pca = pca.transform(centers) # projeta centróides no espaço PCA
plt.scatter(centers_pca[:, 0], centers_pca[:, 1],
           c='red', marker='o', s=200,
           label='Centroids (PCA)')

plt.title(f"Clusters (k={k}) ap\u00f3s PCA")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend()
plt.show()
```

Clusters (k=2) após PCA



In [224...]

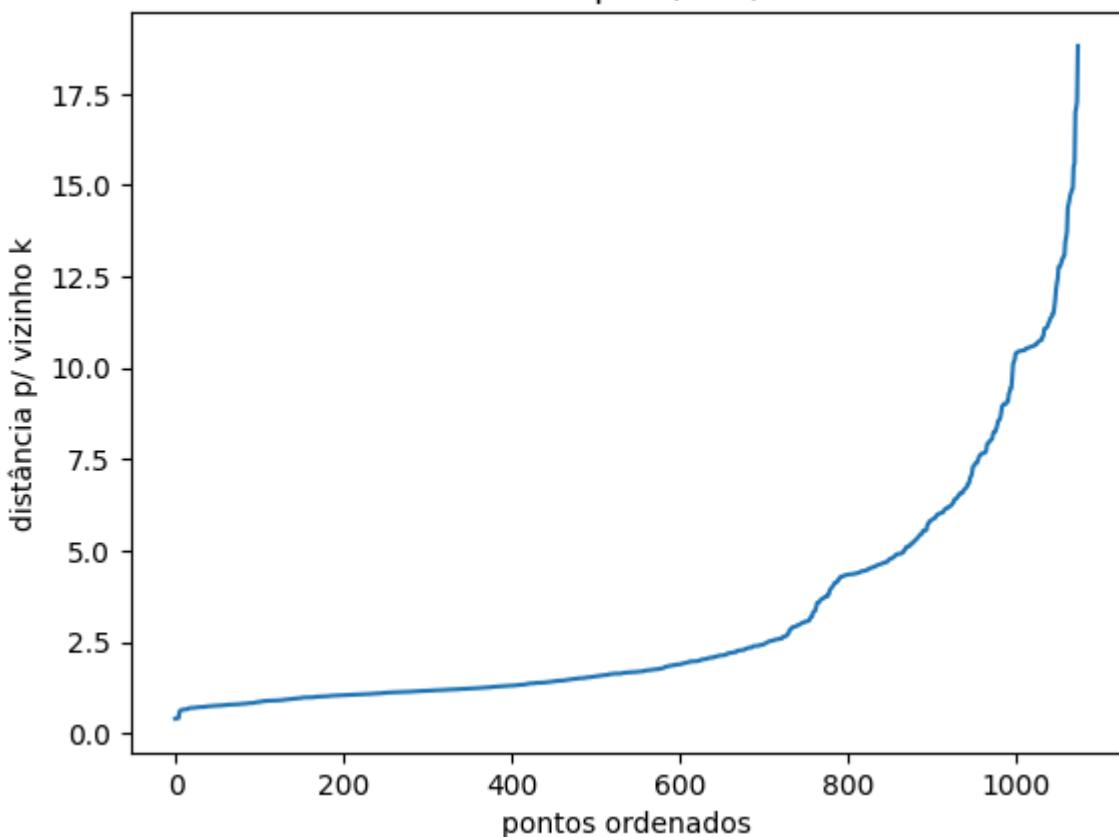
```
from sklearn.neighbors import NearestNeighbors

min_samples = 5

neighbors = NearestNeighbors(n_neighbors=min_samples)
neighbors_fit = neighbors.fit(X)
distances, indices = neighbors_fit.kneighbors(X)

# A distância para o k-ésimo vizinho mais próximo (no caso min_samples=5 -> k=5)
# mas a gente costuma pegar k = min_samples - 1 = 4, depende da convenção.
distances_k = np.sort(distances[:, min_samples - 1])
plt.plot(distances_k)
plt.title(f"k-dist plot (k={min_samples - 1})")
plt.ylabel("distância p/ vizinho k")
plt.xlabel("pontos ordenados")
plt.show()
```

k-dist plot (k=4)



In [225...]

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

candidate_eps = [0.2, 0.3, 0.5, 0.7, 1.0]
candidate_min_samples = [3, 5, 10]

best_score = -1
best_params = None

for eps_ in candidate_eps:
    for ms in candidate_min_samples:
        db = DBSCAN(eps=eps_, min_samples=ms)
        labels = db.fit_predict(X)

        # Se tudo cair como ruído (-1) ou se houver 1 cluster, silhouette não faz sentido
        if len(set(labels)) > 1 and len(set(labels)) != 1:
            score = silhouette_score(X, labels)
            if score > best_score:
                best_score = score
                best_params = (eps_, ms)

print("Melhor silhueta:", best_score)
print("Melhores params (eps, min_samples):", best_params)
```

Melhor silhueta: -0.17864452635297728

Melhores params (eps, min_samples): (0.5, 5)

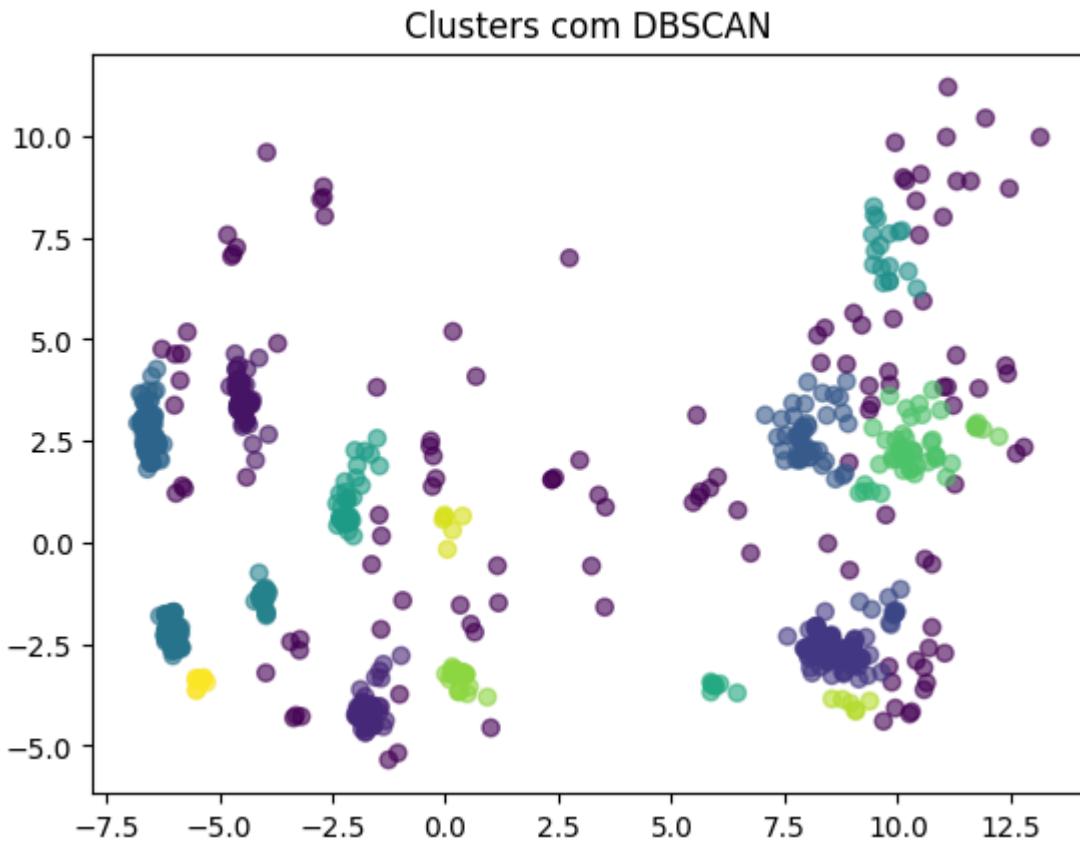
In [228...]

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.5, min_samples=5)
labels_db = db.fit_predict(X_pca)
```

```
# Se X tiver mais que 2 variáveis -> X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels_db, cmap='viridis', alpha=0.6)
plt.title("Clusters com DBSCAN")
plt.show()
```



Análise

Não existe fórmula exata para saber o número ideal de clusters (k) tão pouco para definir hyperparametros como eps e min_samples. Mas os algortimos aplicados dão uma boa noção do caminho para obter melhores resultados.

Justificar o Número de Clusters (K-Means): O gráfico (silhueta médio vs. k) mostrou que k=2 gerou a silhueta mais alta (~0.40). Portanto, justificamos a escolha de k=2 para esse dataset.

Comparar com DBSCAN: DBSCAN obteve silhueta média negativa (-0.1786), indicando alocação ruim de muitos pontos ou excesso de pontos como ruído. Assim, K-Means teve resultado claramente superior do ponto de vista de silhueta.

Silhueta em DBSCAN: Não é a melhor forma de “escolher número de clusters” no DBSCAN, pois este não define a partição de modo fixo, e o ruído distorce os coeficientes. É preferível usar heurísticas de densidade (gráfico k-dist) e/ou analisar a quantidade de clusters resultantes e a porção de ruído de acordo com eps e min_samples.

Medidas de similaridade

1. Um determinado problema, apresenta 10 séries temporais distintas. Gostaríamos de agrupá-las em 3 grupos, de acordo com um critério de similaridade, baseado no valor máximo de correlação cruzada entre elas. Descreva em tópicos todos os passos necessários.
2. Para o problema da questão anterior, indique qual algoritmo de clusterização você usaria. Justifique.
3. Indique um caso de uso para essa solução projetada.
4. Sugira outra estratégia para medir a similaridade entre séries temporais. Descreva em tópicos os passos necessários.

1. Passos para agrupar 10 séries temporais em 3 grupos com base no valor máximo de correlação cruzada

1. Coletar as 10 séries temporais
2. Preprocessar os dados para garantir que tenham o mesmo intervalo de tempo
3. Calcular a correlação cruzada entre as séries temporais
4. Construir uma matriz de similaridade
5. Executar o algoritmo de clusterização a partir desta matriz e definindo o número de clusters = 3
6. Avaliar as métricas de clusterização
7. iterar o modelo proposto com base nos resultados

2. Algoritmo de clusterização e justificativa

Acredito que uma clusterização hierárquica permita utilizar os critérios de similaridade escolhidos e traz uma visualização mais facilitada para uma interpretação e análise manual. É possível cortar o dendrograma em 3 grupos para ver como as séries vão se agrupando.

3. Caso de uso

Pensando nos dados de tiroteitos que trouxe para a análise anterior, podemos identificar, por exemplo, que regiões cujo padrão temporal de tiroteios segue uma dinâmica semelhante (mesmo que deslocada no tempo) é fundamental para:

- Planejamento de segurança
- Monitoramento compartilhado de regiões distintas
- Melhor compreensão do fenômeno dos conflitos

4. Passos para calcular a similaridade utilizando Dynamic Time Warping (DTW)

Dynamic time warping (DTW) é um algoritmo para comparar e alinhar duas séries temporais. A DTW é utilizada para encontrar o alinhamento não-linear ótimo entre duas sequências de valores numéricos. Dessa maneira, é possível encontrar padrões entre medições de eventos com diferentes ritmos.

1. Definir a Distância DTW
2. Calcular a Matriz DTW
3. Obter a Distância DTW para cada par
4. Aplicar um algoritmo de clusterização a partir desta matriz
5. Avaliar as métricas de clusterização
6. iterar o modelo proposto com base nos resultados