PUCPR
GRUPO MARISTA

ATIVIDADE

Projeto Oficina Maker

Felipe Ferro Ramires, Michael da Silva e Verônica Scheifer

Sumário

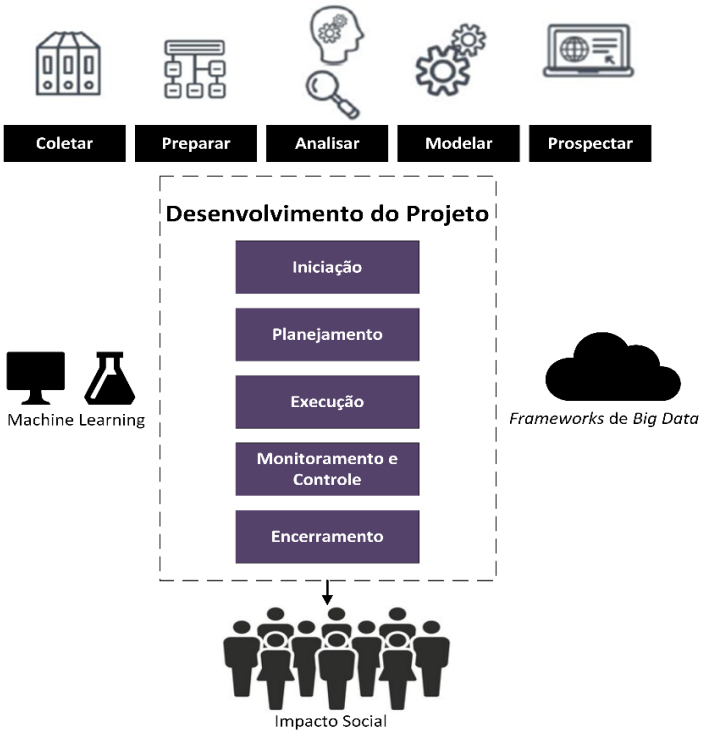
1. Contextualização (Cenário do Projeto)

Após o incidente de segurança que expôs diversos dados sigilosos de clientes que o banco de Tóquio sofreu, foram designados vários inquéritos. O banco foi julgado e condenado a pagar uma multa milionária, onde o juiz compreendeu que não aconteceu um ato de má fé, mas que demonstrou negligência e portanto, condenou o banco a ressarcir os clientes por danos morais e também reverteu parte da multa para fins sociais. Para isso, o banco solicitou a diversos analistas e diretores o auxílio na tomada de decisões.

A nossa equipe é uma das responsáveis por realizar um estudo de Data Science, que foi definido após diversas reuniões com os diretores do banco para contribuir na tomada de decisões para investimento em uma atuação com fins sociais.

2. Arquitetura do Projeto

Com base no cenário apresentado pelo banco de Tóquio, a arquitetura apresentada pela disciplina que será seguida no projeto está representada na figura a seguir, a qual demonstra as principais atividades que serão realizadas durante todo o desenvolvimento do projeto.



Fonte: Template para material de apoio da disciplina Oficina Maker.

3. Planejamento

A primeira etapa do projeto é o planejamento, onde o grupo busca entender o contexto apresentado pelo problema a ser solucionado durante o projeto e verificar quais serão os principais caminhos a serem seguidos nas próximas semanas. Além de coletar todos os requisitos necessários para a definição do escopo e fazer a seleção do dataset e a construção do cronograma detalhado.

- **Cronograma Detalhado:**

ID	FASE	DESCRIÇÃO DA ATIVIDADE	INÍCIO	TÉRMINO
1	Planejamento	Planejamento de todas as semanas do projeto	03/10/22	09/10/22
1.2	Planejamento	Entendimento do problema	03/10/22	09/10/22
1.3	Planejamento	Coleta de requisitos e definição do escopo	03/10/22	09/10/22
1.4	Planejamento	Seleção do Dataset	03/10/22	09/10/22
2	Coleta dos Dados	Análise prévia dos dados	10/10/22	16/10/22
2.1	Coleta dos Dados	Análise das variáveis presentes no dataset	10/10/22	16/10/22
2.2	Coleta dos Dados	Análise dos tipos de dados	10/10/22	16/10/22
2.3	Coleta dos Dados	Análise prévia para entender os dados presentes	10/10/22	16/10/22
2.4	Coleta dos Dados	Escolha do framework de Big Data a ser utilizado para preparação e análise	10/10/22	16/10/22
2.5	Coleta dos Dados	Construção do cronograma detalhado do projeto	10/10/22	16/10/22
3	Preparação e Análise dos Dados	Realização da análise exploratória dos dados (métricas de estatística descritiva, medidas de posição e dispersão, remoção de outliers)	17/10/22	30/10/22
3.1	Preparação e Análise dos Dados	Escolha e descrição dos principais experimentos de Big Data a serem realizados	17/10/22	30/10/22
3.2	Preparação e Análise dos Dados	Realização dos experimentos de Big Data utilizando 3 técnicas diferentes e resultados das primeiras perguntas selecionadas	17/10/22	30/10/22
3.3	Preparação e Análise dos Dados	Entrega do relatório e projeto parcial (Somativa 1)	17/10/22	30/10/22
4	Modelagem e Treinamento	Seleção do modelo que será utilizado	31/10/22	13/11/22
4.1	Modelagem e Treinamento	Construção e aplicação do modelo nas amostras selecionadas	31/10/22	13/11/22
4.2	Modelagem e Treinamento	Avaliação das principais estatísticas do modelo	31/10/22	13/11/22
5	Otimização do Modelo	Realização de novos testes e otimização do modelo	31/10/22	13/11/22
5.1	Otimização do Modelo	Reavaliação das principais estatísticas do modelo	31/10/22	13/11/22
6	Integração	Avaliação dos resultados e benefícios do projeto	14/11/22	27/11/22
6.1	Integração	Construção da conclusão	14/11/22	27/11/22
6.2	Integração	Últimos ajustes no relatório final	14/11/22	27/11/22
6.3	Integração	Criação do vídeo com os principais aspectos do projeto, relatório e resultados obtidos	14/11/22	27/11/22
6.4	Integração	Entrega do projeto (Somativa 2)	14/11/22	27/11/22

- **Descrição do dataset:**

    - *fonte/origem;*

O dataset selecionado foi o SiGesGuarda, disponibilizado pela Prefeitura Municipal de Curitiba no Estado do Paraná – Brasil.

    - *apresentação do cenário (dataset);*

O dataset é composto por dados das ocorrências que são recebidas e atendidas pela Guarda Municipal de Curitiba. O órgão responsável é a Defesa Social e Trânsito. O principal responsável é Sergio Roberto da Silva Cruz. O dataset possui uma frequência de atualização mensal, sendo um espectro temporal de 2009 até o momento da extração.

- *tipo do dataset (semi ou não estruturado);*

O dataset é semi estruturado, pois possui uma estrutura minimamente definida em arquivos csv ou xls, mas é necessário fazer um trabalho de tratamento, transformação e limpeza para que os dados estejam prontos para serem explorados e analisados.

- *descrição do dataset (número de instâncias, variáveis, tipos de dado etc.);*

O csv selecionado para análise que é referente ao mês de outubro de 2022 possui ao todo 35 colunas x 419072 linhas, com 8657547 dados. O dataset todo é composto por dados do tipo object, mas ao interpretar podemos ver alguns dados que podem ser transformados para tipos como datetime e boolean, por exemplo.

Alguns tipos de variáveis também podem ser percebidas: quantitativa contínua, como data, qualitativa nominal, como região e a qualitativa ordinal, ordinal como tipo de ocorrência.

```
RangeIndex: 419072 entries, 0 to 419071
Data columns (total 35 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   ATENDIMENTO_ANO           418983 non-null  object
 1   ATENDIMENTO_BAIRRO_NOME   418927 non-null  object
 2   EQUIPAMENTO_URBANO_NOME   150029 non-null  object
 3   FLAG_EQUIPAMENTO_URBANO   419072 non-null  object
 4   FLAG_FLAGRANTE            419072 non-null  object
 5   LOGRADOURO_NOME           419072 non-null  object
 6   NATUREZA1_DEFESA_CIVIL    419072 non-null  object
```

```
 7   NATUREZA1_DESCRICAO        419072 non-null  object
 8   NATUREZA2_DEFESA_CIVIL     21842 non-null   object
 9   NATUREZA2_DESCRICAO        21842 non-null   object
 10  NATUREZA3_DEFESA_CIVIL     1646 non-null    object
 11  NATUREZA3_DESCRICAO        1646 non-null    object
 12  NATUREZA4_DEFESA_CIVIL     280 non-null     object
 13  NATUREZA4_DESCRICAO        280 non-null     object
 14  NATUREZA5_DEFESA_CIVIL     62 non-null      object
 15  NATUREZA5_DESCRICAO        62 non-null      object
 16  SUBCATEGORIA1_DESCRICAO    289866 non-null  object
 17  SUBCATEGORIA2_DESCRICAO    13447 non-null   object
 18  SUBCATEGORIA3_DESCRICAO    939 non-null     object
 19  SUBCATEGORIA4_DESCRICAO    135 non-null     object
 20  SUBCATEGORIA5_DESCRICAO    31 non-null      object
 21  OCORRENCIA_ANO             419072 non-null  object
 22  OCORRENCIA_CODIGO          419072 non-null  object
 23  OCORRENCIA_DATA            419072 non-null  object
 24  OCORRENCIA_DIA_SEMANA      419072 non-null  object
 25  OCORRENCIA_HORA            419072 non-null  object
 26  OCORRENCIA_MES             419072 non-null  object
 27  OPERACAO_DESCRICAO         306849 non-null  object
 28  ORIGEM_CHAMADO_DESCRICAO   419072 non-null  object
 29  REGIONAL_FATO_NOME         419061 non-null  object
 30  SECRETARIA_NOME            419072 non-null  object
 31  SECRETARIA_SIGLA           419072 non-null  object
 32  SERVICO_NOME               419072 non-null  object
 33  SITUACAO_EQUIPE_DESCRICAO  293395 non-null  object
 34  NUMERO_PROTOCOLO_156       12110 non-null   object
dtypes: object(35)
```

- *manipulação prévia do dataset;*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats as sp

df = pd.read_csv("/content/2022-10-01_sigesguarda_-_Base_de_Dados.csv",
sep=';', encoding='latin-1')
```

```
df.head()

df.describe()

df.info()
```

```
df.head()
```

	QUANTIDADE_OCORRENCIA	ATENDIMENTO_ANO	ATENDIMENTO_BAIRRO_NOME	EQUIPAMENTO_URBANO_NOME	FLAG_EQUIPAMENTO_URBANO	FLAG_FLAGRANTE	LOGRADOURO_NOME
0	1	2009.0	CIDADE INDUSTRIAL	NaN	NÃO	NÃO	DAVI XAVIER DA SILVA
1	1	2009.0	FAZENDINHA	BOSQUE DA FAZENDINHA	SIM	NÃO	CARLOS KLEMTZ
2	1	2009.0	UBERABA	NaN	NÃO	NÃO	DOUTOR JOÃO DE PAULA MOURA BRITO
3	1	2009.0	SÍTIO CERCADO	NaN	NÃO	NÃO	EDGARD CAVALCANTI DE ALBUQUERQUE
4	1	2009.0	TATUQUARA	CENTRO DE ESPORTE E LAZER SANTA RITA	SIM	NÃO	CARLOS MUNHOZ DA ROCHA

5 rows × 36 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 419072 entries, 0 to 419071
Data columns (total 35 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   ATENDIMENTO_ANO            418983 non-null  object
 1   ATENDIMENTO_BAIRRO_NOME    418927 non-null  object
 2   EQUIPAMENTO_URBANO_NOME    150029 non-null  object
 3   FLAG_EQUIPAMENTO_URBANO    419072 non-null  object
 4   FLAG_FLAGRANTE             419072 non-null  object
 5   LOGRADOURO_NOME            419072 non-null  object
 6   NATUREZA1_DEFESA_CIVIL     419072 non-null  object
 7   NATUREZA1_DESCRICAO        419072 non-null  object
 8   NATUREZA2_DEFESA_CIVIL     21842 non-null   object
 9   NATUREZA2_DESCRICAO        21842 non-null   object
 10  NATUREZA3_DEFESA_CIVIL     1646 non-null    object
 11  NATUREZA3_DESCRICAO        1646 non-null    object
 12  NATUREZA4_DEFESA_CIVIL     280 non-null     object
 13  NATUREZA4_DESCRICAO        280 non-null     object
 14  NATUREZA5_DEFESA_CIVIL     62 non-null      object
 15  NATUREZA5_DESCRICAO        62 non-null      object
 16  SUBCATEGORIA1_DESCRICAO    289866 non-null  object
 17  SUBCATEGORIA2_DESCRICAO    13447 non-null   object
```

```
df.describe()
```

	QUANTIDADE_OCORRENCIA	ATENDIMENTO_ANO	NATUREZA1_DEFESA_CIVIL	NATUREZA2_DEFESA_CIVIL	NATUREZA3_DEFESA_CIVIL	NATUREZA4_DEFESA_CIVIL	NATUREZ
count	419071.0	418982.00000	419071.000000	21841.000000	1645.000000	279.000000	
mean	1.0	2016.65204	0.025874	0.020008	0.013374	0.014337	
std	0.0	4.14695	0.158759	0.140031	0.114904	0.119089	
min	1.0	2009.00000	0.000000	0.000000	0.000000	0.000000	
25%	1.0	2013.00000	0.000000	0.000000	0.000000	0.000000	
50%	1.0	2018.00000	0.000000	0.000000	0.000000	0.000000	
75%	1.0	2020.00000	0.000000	0.000000	0.000000	0.000000	
max	1.0	2022.00000	1.000000	1.000000	1.000000	1.000000	

- *descrição das atividades realizadas no dataset;*

A primeira manipulação no dataset foi a exclusão da primeira linha vazia que havia e também o acréscimo da coluna QUANTIDADE_OCORRENCIA populada com o valor 1 para cada linha, já que cada linha representa o valor de 1 ocorrência, que irá auxiliar nas somas e agrupamentos realizadas no Python.

QUANTIDADE_OCORRENCIA	ATENDIMI	ATENDIMI	EQUIPAMI	FLAG_EQU	FLAG_FLA	LOGRADO	NATUREZA	NATUREZA	NATUREZA	NATUREZA	NATUREZA	NATUREZA	NATUREZA	NATUREZA	NATUREZA	NATUREZA	SUB
1	2009	CIDADE INDUSTRIAL		NÃO	NÃO	DAVI XAVI	0	Alarmes									Disp
1	2009	FAZENDIN	BOSQUE D	SIM	NÃO	CARLOS K	0	Roubo									Tran
1	2009	UBERABA		NÃO	NÃO	DOUTOR J	0	Animais									Cão
1	2009	SÍTIO CERCADO		NÃO	NÃO	EDGARD C	0	Animais									Cão
1	2009	TATUQUA	CENTRO D	SIM	NÃO	CARLOS M	0	Alarmes									Disp
1	2009	SÍTIO CERCADO		NÃO	NÃO	HUSSEIN I	0	Alarmes									Disp
1	2009	SANTA CÂ	CEMITÉRI	SIM	NÃO	NOVA DE	0	Trânsito									Acid
1	2009	BOQUEIRÃ	CENTRO D	SIM	NÃO	CARLOS E	0	Alarmes									Disp
1	2009	CENTRO		NÃO	SIM	PRESIDEN	0	Roubo									Tran
1	2009	BOA VISTA	CENTRO M	SIM	NÃO	SANTA ED	0	Invasão									Inva
1	2009	TABOÃO	PARQUE T	SIM	NÃO	OSWALDO	0	Apoio									Apoi
1	2009	CENTRO		NÃO	NÃO	TOBIAS DE	0	Fundada Suspeita (Abordagem)									
1	2009	CENTRO		NÃO	NÃO	TIRADENT	0	Substância Ilícita									Tráfi
1	2009	CENTRO		NÃO	NÃO	TOBIAS DE	0	Atitude Suspeita									
1	2009	CIDADE INDUSTRIAL		NÃO	NÃO	MARIA HC	0	Alarmes									Disp
1	2009	XAXIM		NÃO	NÃO	1 DE MAIC	0	Alarmes									Disp
1	2009	UBERABA	LICEU DE C	SIM	NÃO	OLINDO C	0	Alarmes									Disp
1	2009	PILARZINH	CASA TINC	SIM	NÃO	FREDOLIN	0	Alarmes									Disp
1	2009	REBOUÇAS		NÃO	NÃO	ROCKEFEL	0	Alarmes									Disp
1	2009	ÁGUA VER	CATI ÁGU	SIM	NÃO	ÁGUA VEI	0	Alarmes									Disp

Após essa etapa carregou-se o csv no Google Colab para iniciar a etapa de trabalho diretamente na linguagem de programação Python. Para ler o csv importou-se a biblioteca de pandas que permite a manipulação dos dados e utilizou-se a função read_csv() que lê o arquivo csv.

A primeira função utilizada para observar os dados foi a head() que apresenta por default os 5 primeiros registros de cada coluna do dataframe e assim pode-se

verificar se os dados foram carregados da forma correta e separados em colunas como em um arquivo excel.

A segunda função é a info() que mostra a quantidade de registros no arquivo, bem como a quantidade de colunas e dados não nulos em cada uma das colunas e também o tipo de dado presente em cada.

A terceira função para ter uma visão prévia dos dados é a describe() que apresenta algumas estatísticas de cada uma das colunas presentes no dataset.

4. Coleta de Dados

- **Descrição dos experimentos iniciais:**
    - *seleção das variáveis relevantes para o projeto;*

O dataset selecionado possui 35 colunas no total e destas colunas foram selecionadas 12 variáveis, sendo elas:

ATENDIMENTO_ANO,
OCORRENCIA_ANO,
ATENDIMENTO_BAIRRO_NOME,
FLAG_FLAGRANTE,
NATUREZA1_DESCRICAO,
OCORRENCIA_DATA,
OCORRENCIA_DIA_SEMANA,
REGIONAL_FATO_NOME,
QUANTIDADE_OCORRENCIA ,
OCORRENCIA_HORA,
OCORRENCIA_MES e
OCORRENCIA_DIA.

- *descrição dos experimentos iniciais sobre o dataset do projeto;*

- **Contagem de dados nulos**

```
Contagem de dados nulos
def contagemNulos(tabela):
  for col in tabela.columns:
   if tabela[col].isnull().sum():
    total_null=tabela[col].isnull().sum()
    print('Column: {} total null {}, i.e. {}
```

```
%'.format(col,total_null,round(total_null*100/len(df),2)))

dfContagemNulos = contagemNulos(df)
```

```
Column: ATENDIMENTO_ANO total null 89, i.e. 0.02 %
Column: ATENDIMENTO_BAIRRO_NOME total null 145, i.e. 0.03 %
Column: EQUIPAMENTO_URBANO_NOME total null 269043, i.e. 64.2 %
Column: NATUREZA2_DEFESA_CIVIL total null 397230, i.e. 94.79 %
Column: NATUREZA2_DESCRICAO total null 397230, i.e. 94.79 %
Column: NATUREZA3_DEFESA_CIVIL total null 417426, i.e. 99.61 %
Column: NATUREZA3_DESCRICAO total null 417426, i.e. 99.61 %
Column: NATUREZA4_DEFESA_CIVIL total null 418792, i.e. 99.93 %
Column: NATUREZA4_DESCRICAO total null 418792, i.e. 99.93 %
Column: NATUREZA5_DEFESA_CIVIL total null 419010, i.e. 99.99 %
Column: NATUREZA5_DESCRICAO total null 419010, i.e. 99.99 %
Column: SUBCATEGORIA1_DESCRICAO total null 129206, i.e. 30.83 %
Column: SUBCATEGORIA2_DESCRICAO total null 405625, i.e. 96.79 %
Column: SUBCATEGORIA3_DESCRICAO total null 418133, i.e. 99.78 %
Column: SUBCATEGORIA4_DESCRICAO total null 418937, i.e. 99.97 %
Column: SUBCATEGORIA5_DESCRICAO total null 419041, i.e. 99.99 %
Column: OPERACAO_DESCRICAO total null 112223, i.e. 26.78 %
Column: REGIONAL_FATO_NOME total null 11, i.e. 0.0 %
Column: SITUACAO_EQUIPE_DESCRICAO total null 125677, i.e. 29.99 %
Column: NUMERO_PROTOCOLO_156 total null 406962, i.e. 97.11 %
```

- **Convertendo colunas de horas**

```
df['OCORRENCIA_DATA'] = pd.to_datetime(df.OCORRENCIA_DATA,
format='%Y-%m-%d')

df['OCORRENCIA_DATA'] = df['OCORRENCIA_DATA'].dt.strftime('%Y-%m-%d')

df['OCORRENCIA_DATA']
```

```
0         2009-01-01
1         2009-01-01
2         2009-01-01
3         2009-01-01
4         2009-01-01
             ...
419066    2022-10-01
419067    2022-10-01
419068    2022-10-01
419069    2022-09-30
419070    2022-10-01
Name: OCORRENCIA_DATA, Length: 419071, dtype: object
```

- **Agrupando as colunas e criando o novo dataset**

```
df2 = df.groupby(['ATENDIMENTO_ANO','OCORRENCIA_ANO',
"ATENDIMENTO_BAIRRO_NOME", "FLAG_FLAGRANTE", "NATUREZA1_DESCRICAO",
"OCORRENCIA_DATA", "OCORRENCIA_DIA_SEMANA", "REGIONAL_FATO_NOME"
])['QUANTIDADE_OCORRENCIA'].sum().reset_index()
```

```
df2
```

	ATENDIMENTO_ANO	OCORRENCIA_ANO	ATENDIMENTO_BAIRRO_NOME	FLAG_FLAGRANTE	NATUREZA1_DESCRICAO	OCORRENCIA_DATA	OCO
0	2009.0	2009	ABRANCHES	NÃO	AIFU	2009-05-17	
1	2009.0	2009	ABRANCHES	NÃO	AIFU	2009-05-28	
2	2009.0	2009	ABRANCHES	NÃO	AIFU	2009-07-23	
3	2009.0	2009	ABRANCHES	NÃO	Alagamento	2009-11-16	
4	2009.0	2009	ABRANCHES	NÃO	Alarmes	2009-09-03	
...	...	...	...	...	...	...	
324458	2022.0	2022	ÁGUA VERDE	SIM	Trânsito	2022-05-07	
324459	2022.0	2022	ÁGUA VERDE	SIM	Veículo	2022-03-24	
324460	2022.0	2022	ÁGUA VERDE	SIM	Violação de Medida Protetiva Lei Maria da Penha	2022-03-26	
324461	2022.0	2022	ÁGUA VERDE	SIM	Violação de Medida Protetiva Lei Maria da Penha	2022-04-11	
324462	2022.0	2022	ÁGUA VERDE	SIM	ZELADORIA URBANA	2022-03-16	

324463 rows × 9 columns

- **Quantidade de ocorrências atendidas por ano (ATENDIMENTO_ANO)**

```
-- Quantidade de ocorrencias atendidas por ano (ATENDIMENTO_ANO)

ocorrenciasAtendidasPorAno =
df.groupby(['ATENDIMENTO_ANO'])['QUANTIDADE_OCORRENCIA'].sum().reset_index()

ocorrenciasAtendidasPorAno =
ocorrenciasAtendidasPorAno.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Atendidas_Por_Ano'}, axis=1)

ocorrenciasAtendidasPorAno
```

	ATENDIMENTO_ANO	Quantidade_Ocorrencias_Atendidas_Por_Ano
0	2009.0	24903
1	2010.0	22531
2	2011.0	21481
3	2012.0	19778
4	2013.0	24468
5	2014.0	25292
6	2015.0	25566
7	2016.0	21470
8	2017.0	22133
9	2018.0	24155
10	2019.0	33754
11	2020.0	57457
12	2021.0	58726
13	2022.0	37268

```
ocorrenciasAtendidasPorAno.describe()
```

	ATENDIMENTO_ANO	QUANTIDADE_OCORRENCIA
count	14.0000	14.000000
mean	2015.5000	29927.285714
std	4.1833	12848.177108
min	2009.0000	19778.000000
25%	2012.2500	22232.500000
50%	2015.5000	24685.500000
75%	2018.7500	31707.000000
max	2022.0000	58726.000000

- **Quantidade de ocorrências recebidas por ano (OCORRENCIA_ANO)**

```
-- Quantidade de ocorrências recebidas por ano (OCORRENCIA_ANO)

ocorrenciasRegistradasPorAno =
df.groupby(['OCORRENCIA_ANO'])['QUANTIDADE_OCORRENCIA'].sum().reset_index()

ocorrenciasRegistradasPorAno =
ocorrenciasRegistradasPorAno.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Registradas_Por_Ano'}, axis=1)

ocorrenciasRegistradasPorAno
```

	OCORRENCIA_ANO	Quantidade_Ocorrencias_Registradas_Por_Ano
0	2009	24903
1	2010	22531
2	2011	21484
3	2012	19776
4	2013	24470
5	2014	25303
6	2015	25566
7	2016	21469
8	2017	22137
9	2018	24153
10	2019	33762
11	2020	57453
12	2021	58731
13	2022	37333

```
ocorrenciasRegistradasPorAno.describe()
```

	OCORRENCIA_ANO	Quantidade_Ocorrencias_Registradas_Por_Ano
count	14.0000	14.000000
mean	2015.5000	29933.642857
std	4.1833	12850.963637
min	2009.0000	19776.000000
25%	2012.2500	22235.500000
50%	2015.5000	24686.500000
75%	2018.7500	31713.000000
max	2022.0000	58731.000000

```
-- Quantidade de ocorrencias por bairro (ATENDIMENTO_BAIRRO_NOME)

ocorrenciasPorBairro = df.groupby(['ATENDIMENTO_BAIRRO_NOME'])['QUANTIDADE_OCORRENCIA'].sum().reset_index()

ocorrenciasPorBairro = ocorrenciasPorBairro.rename({'QUANTIDADE_OCORRENCIA': 'Quantidade_Ocorrencias_Registradas_Por_Bairro'}, axis=1)

ocorrenciasPorBairro
```

	ATENDIMENTO_BAIRRO_NOME	Quantidade_Ocorrencias_Registradas_Por_Bairro
0	JARDIM OSASCO	1
1	JARDIM PEDRO DEMETE	1
2	ABRANCHES	2143
3	AFONSO PENA	1
4	AFONSO PENA	1
...	...	...
186	VISTA ALEGRE	1499
187	XAXIM	6768
188	fanny	1
189	ÁGUA VERDE	7634
190	ÁGUAS BELAS	3

191 rows × 2 columns

```
ocorrenciasPorBairro.describe()
```

	QUANTIDADE_OCORRENCIA
count	191.000000
mean	2193.329843
std	6505.865766
min	1.000000
25%	1.000000
50%	2.000000
75%	1996.500000
max	72247.000000

- **Quantidade de ocorrências com flagrante (FLAG_FLAGRANTE)**

```
-- Quantidade de ocorrencias com flagrante (FLAG_FLAGRANTE)

ocorrenciasFlagrante =
df.groupby(['FLAG_FLAGRANTE'])['QUANTIDADE_OCORRENCIA'].sum().reset_index()

ocorrenciasFlagrante =
ocorrenciasFlagrante.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Flagrante'}, axis=1)
```

```
ocorrenciasFlagrante
```

	FLAG_FLAGRANTE	Quantidade_Ocorrencias_Flagrante
0	NÃO	400210
1	SIM	18861

- **Quantidade de ocorrências por tipo (NATUREZA1_DESCRICAO)**

```
-- Quantidade de ocorrencias por tipo (NATUREZA1_DESCRICAO)

ocorrenciasPorTipo =
df.groupby(['NATUREZA1_DESCRICAO'])['QUANTIDADE_OCORRENCIA'].sum().reset_in
dex()

ocorrenciasPorTipo = ocorrenciasPorTipo.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Por_Tipo'}, axis=1)

ocorrenciasPorTipo
```

	NATUREZA1_DESCRICAO	Quantidade_Ocorrencias_Por_Tipo
0	AIFU	662
1	Abalo Sísmico	2
2	Abandono de função	4
3	Abandono de incapaz	157
4	Abuso de incapazes	51
...	...	...
182	Vistoria	3751
183	ZELADORIA URBANA	1398
184	Óbito	156
185	Óbito (Defesa Civil)	1
186	Órgãos acionados	77

187 rows × 2 columns

```
ocorrenciasPorTipo.describe()
```

	Quantidade_Ocorrencias_Por_Tipo
count	187.000000
mean	2241.021390
std	7660.677328
min	1.000000
25%	11.000000
50%	69.000000
75%	554.500000
max	67967.000000

- **Quantidade de ocorrências por dia (OCORRENCIA_DATA)**

```
-- Quantidade de ocorrencias por dia (OCORRENCIA_DATA)

ocorrenciasPorDia =
df.groupby(['OCORRENCIA_DATA'])['QUANTIDADE_OCORRENCIA'].sum().reset_index(
)

ocorrenciasPorDia = ocorrenciasPorDia.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Por_Dia'}, axis=1)

ocorrenciasPorDia
```

	OCORRENCIA_DATA	Quantidade_Ocorrencias_Por_Dia
0	2009-01-01	30
1	2009-01-02	81
2	2009-01-03	96
3	2009-01-04	93
4	2009-01-05	59
...	...	...
5017	2022-09-27	135
5018	2022-09-28	125
5019	2022-09-29	149
5020	2022-09-30	164
5021	2022-10-01	8

5022 rows × 2 columns

```
ocorrenciasPorDia.describe()
```

	QUANTIDADE_OCORRENCIA
count	5022.000000
mean	83.447033
std	44.620405
min	2.000000
25%	56.000000
50%	69.000000
75%	95.000000
max	606.000000

- **Quantidade de ocorrências por dia da semana (OCORRENCIA_DIA_SEMANA)**

```
-- Quantidade de ocorrencias por dia da semana (OCORRENCIA_DIA_SEMANA)

ocorrenciasPorDiaDaSemana =
df.groupby(['OCORRENCIA_DIA_SEMANA'])['QUANTIDADE_OCORRENCIA'].sum().reset_
index()

ocorrenciasPorDiaDaSemana =
```

```
ocorrenciasPorDiaDaSemana.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Por_Dia_Da_Semana'}, axis=1)

ocorrenciasPorDiaDaSemana
```

	OCORRENCIA_DIA_SEMANA	Quantidade_Ocorrencias_Por_Dia_Da_Semana
0	DOMINGO	64524
1	QUARTA	57177
2	QUINTA	59549
3	SEGUNDA	53472
4	SEXTA	61533
5	SÁBADO	67692
6	TERÇA	55124

```
ocorrenciasPorRegional =
df.groupby(['REGIONAL_FATO_NOME'])['QUANTIDADE_OCORRENCIA'].sum().reset_index()

ocorrenciasPorRegional =
ocorrenciasPorRegional.rename({'QUANTIDADE_OCORRENCIA':
'Quantidade_Ocorrencias_Por_Regional'}, axis=1)

ocorrenciasPorRegional
```

	REGIONAL_FATO_NOME	Quantidade_Ocorrencias_Por_Regional
0	BAIRRO NOVO	29831
1	BOA VISTA	43063
2	BOQUEIRÃO	38847
3	CAJURU	36306
4	CIC	31537
5	Cajuru	1
6	MATRIZ	131935
7	PINHEIRINHO	29299
8	PORTÃO	38078
9	Pinheirinho	1
10	Portão	1
11	REGIÃO METROPOLITANA	430
12	SANTA FELICIDADE	29980
13	TATUQUARA	9751

Para realizar a segunda etapa da análise exploratória que são algumas principais perguntas que o grupo construiu e também a etapa de modelagem, construiu-se um novo notebook e foi carregado novamente um dataframe do dataset e realizado algumas transformações como apresenta-se a seguir.

Nesta etapa utiliza-se novamente as funções info() e describe() para verificar as informações e métricas do dataset e também a função median() que retorna a mediana de um conjunto de dados e a var() para a variância, bem como a função value_counts() que faz a contagem dos valores presentes.

- **Importação das bibliotecas**

```
import numpy as np
import pandas as pd # importando o pandas para manipularmos o dataset
import seaborn as sns # importando o Seaborn para visualizar o comportamento
dos dados
import matplotlib.pyplot as plt # importando o Matplotlib para o elbow method
from pandas_profiling import ProfileReport # importando o pandas-profiling para
fazer o profile do dataset
from scipy import stats as sp
```

```
from sklearn.model_selection import train_test_split # utilizado para o split entre treinamento e teste
from sklearn.neighbors import KNeighborsRegressor # KNN para regressão
from sklearn.linear_model import LinearRegression # Regressão linear
from sklearn.svm import SVR # SVM para regressão
from sklearn.decomposition import PCA # PCA como aprendizagem não-supervisionada
from sklearn.preprocessing import RobustScaler # utilizado para que todas as entradas estejam na mesma escala numérica
from sklearn.preprocessing import StandardScaler
from pandas.core.frame import DataFrame
from matplotlib import pyplot as plt
%matplotlib inline


df = pd.read_csv('/content/teste.csv', sep=';', encoding='ISO-8859-1')
df
```

	ATENDIMENTO_BAIRRO_NOME	NATUREZA1_DESCRICAO	OCORRENCIA_ANO	OCORRENCIA_DIA_SEMANA	OCORRENCIA_HORA	OCORRENCIA_MES	OCORRENCIA_DIA
0	CIDADE INDUSTRIAL	Alarmes	2009	QUINTA	15:14:00	1.0	1.0
1	FAZENDINHA	Roubo	2009	QUINTA	15:22:00	1.0	1.0
2	UBERABA	Animais	2009	QUINTA	15:59:00	1.0	1.0
3	SÍTIO CERCADO	Animais	2009	QUINTA	16:13:00	1.0	1.0
4	TATUQUARA	Alarmes	2009	QUINTA	16:29:00	1.0	1.0
...	...	...	...	...	...	...	...
249550	SÃO FRANCISCO	Trânsito	2019	QUINTA	05:15:00	8.0	1.0
249551	CENTRO	Fundada Suspeita (Abordagem)	2019	QUINTA	07:15:00	8.0	1.0
249552	CIDADE INDUSTRIAL	Apoio	2019	QUINTA	07:15:00	8.0	1.0
249553	CENTRO	Fundada Suspeita (Abordagem)	2019	QUINTA	07:35:00	8.0	1.0
249554	CENTRO	Fundada Suspeita (Abordagem)	2019	QUINTA	07:40:00	NaN	NaN

249555 rows × 7 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 249555 entries, 0 to 249554
Data columns (total 7 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   ATENDIMENTO_BAIRRO_NOME  249410 non-null  object 
 1   NATUREZA1_DESCRICAO      249555 non-null  object 
 2   OCORRENCIA_ANO           249555 non-null  int64  
 3   OCORRENCIA_DIA_SEMANA    249555 non-null  object 
 4   OCORRENCIA_HORA          249555 non-null  object 
 5   OCORRENCIA_MES           249554 non-null  float64
 6   OCORRENCIA_DIA           249554 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.3+ MB
```

```
# **Informação do dataset**

df['ATENDIMENTO_BAIRRO_NOME'].value_counts()
```

```
CENTRO                  38843
CIDADE INDUSTRIAL       17166
SÍTIO CERCADO           15471
CAJURU                  12581
BOQUEIRÃO               11464
                        ...
CAMPO PEQUENO               1
VILA PERNETA                1
CAMPO DE SÃO BENEDIT        1
QUATRO BARRAS               1
PLANTA DEODORO              1
Name: ATENDIMENTO_BAIRRO_NOME, Length: 144, dtype: int64
```

```
df['NATUREZA1_DESCRICAO'].value_counts()
```

```
Apoio                                37220
Dano                                 32576
Perturbação do sossego               18804
Roubo                                15559
Substância Ilícita                   15559
                                     ...
Corrupção ativa                          1
Rompimento de Barragem                   1
Prostituição                             1
Acidente Viatura                         1
Falsificação de documento Publico        1
Name: NATUREZA1_DESCRICAO, Length: 165, dtype: int64
```

```
df['OCORRENCIA_ANO'].value_counts()
```

```
2015    25566
2014    25303
2009    24903
2013    24470
2018    24153
2010    22531
2017    22137
2011    21484
2016    21469
2012    19776
2019    17763
Name: OCORRENCIA_ANO, dtype: int64
```

```
df['OCORRENCIA_DIA_SEMANA'].value_counts()
```

```
SÁBADO     40386
DOMINGO    39905
SEXTA      36436
QUINTA     35361
QUARTA     33685
TERÇA      32323
SEGUNDA    31459
Name: OCORRENCIA_DIA_SEMANA, dtype: int64
```

```
df['OCORRENCIA_MES'].value_counts()
```

```
5.0     23271
4.0     22481
3.0     22311
1.0     22208
7.0     21477
6.0     20608
2.0     20262
10.0    20180
8.0     19553
11.0    19434
9.0     18922
12.0    18847
Name: OCORRENCIA_MES, dtype: int64
```

```
df['OCORRENCIA_HORA'].value_counts()
```

```
15:00:00    1690
16:00:00    1688
10:00:00    1455
14:00:00    1442
17:00:00    1425
            ...
04:31:00      26
06:56:00      26
06:54:00      26
06:57:00      23
06:53:00      20
Name: OCORRENCIA_HORA, Length: 1440, dtype: int64
```

```
df.describe()
```

	OCORRENCIA_ANO	OCORRENCIA_MES	OCORRENCIA_DIA
count	249555.000000	249554.000000	249554.000000
mean	2013.908754	6.324848	15.780681
std	3.113483	3.423156	8.789824
min	2009.000000	1.000000	1.000000
25%	2011.000000	3.000000	8.000000
50%	2014.000000	6.000000	16.000000
75%	2017.000000	9.000000	23.000000
max	2019.000000	12.000000	31.000000

```
#.median() Função Pandas retorna a mediana dos valores para o eixo
solicitado.
df.median()
```

```
OCORRENCIA_ANO    2014.0
OCORRENCIA_MES       6.0
OCORRENCIA_DIA      16.0
dtype: float64
```

```
#.var() calcula a variância no Pandas através da função
df.var()
```

```
OCORRENCIA_ANO     9.693779
OCORRENCIA_MES    11.717994
OCORRENCIA_DIA    77.261000
dtype: float64
```

A próxima parte é realizado novamente o tratamento de dados, onde realiza-se a limpeza de dados nulos, converte datas, colunas para inteiro e também algumas transformações e criações de label. Também utiliza-se a função std() para retornar o desvio padrão das colunas numéricas.

- **Limpando dados nulos**

```
# **Tratamento de dados**
```

```
## *Limpando dados nulos*
for col in df.columns:
 if df[col].isnull().sum():
  total_null=df[col].isnull().sum()
  print('Column: {} total null {}, i.e. {}
%'.format(col,total_null,round(total_null*100/len(df),2)))
```

```
Column: ATENDIMENTO_BAIRRO_NOME total null 145, i.e. 0.06 %
Column: OCORRENCIA_MES total null 1, i.e. 0.0 %
Column: OCORRENCIA_DIA total null 1, i.e. 0.0 %
```

```
#Limpando dados nulos
df.dropna(inplace = True)
df.isnull().sum()
```

```
ATENDIMENTO_BAIRRO_NOME    0
NATUREZA1_DESCRICAO        0
OCORRENCIA_ANO             0
OCORRENCIA_DIA_SEMANA      0
OCORRENCIA_HORA            0
OCORRENCIA_MES             0
OCORRENCIA_DIA             0
dtype: int64
```

- **Convertendo para int e datetime**

```
## *convertendo para int*

### *OCORRENCIA_HORA*

#converteu para datetime
df['OCORRENCIA_HORA'] = pd.to_datetime(df['OCORRENCIA_HORA'])

df['OCORRENCIA_HORA'].dt.time
```

```
0         15:14:00
1         15:22:00
2         15:59:00
3         16:13:00
4         16:29:00
            ...
249549    03:20:00
249550    05:15:00
249551    07:15:00
249552    07:15:00
249553    07:35:00
Name: OCORRENCIA_HORA, Length: 249409, dtype: object
```

```
df['OCORRENCIA_HORA'] = df['OCORRENCIA_HORA'].dt.strftime('%H')

df['OCORRENCIA_HORA'] = df['OCORRENCIA_HORA'].astype(str).astype(int)

df['OCORRENCIA_HORA']
```

```
0         15:14:00
1         15:22:00
2         15:59:00
3         16:13:00
4         16:29:00
            ...
249549    03:20:00
249550    05:15:00
249551    07:15:00
249552    07:15:00
249553    07:35:00
Name: OCORRENCIA_HORA, Length: 249409, dtype: object
```

```
 df['OCORRENCIA_HORA'] = df['OCORRENCIA_HORA'].dt.strftime('%H')

df['OCORRENCIA_HORA'] = df['OCORRENCIA_HORA'].astype(str).astype(int)

df['OCORRENCIA_HORA']
```

```
0         15
1         15
2         15
3         16
4         16
          ..
249549     3
249550     5
249551     7
249552     7
249553     7
Name: OCORRENCIA_HORA, Length: 249409, dtype: int64
```

- **Criação de labels e cálculo desvio padrão**

```
### ATENDIMENTO_BAIRRO_NOME

df['ATENDIMENTO_BAIRRO_NOME'] =
df['ATENDIMENTO_BAIRRO_NOME'].replace({'CIDADE INDUSTRIAL':1,
'FAZENDINHA':2, 'UBERABA':3, 'SÍTIO CERCADO':4, 'TATUQUARA':5, 'SANTA
CÂNDIDA':6, 'BOQUEIRÃO':7, 'CENTRO':8, 'BOA VISTA':9, 'TABOÃO':10,
'XAXIM':11, 'PILARZINHO':12, 'REBOUÇAS':13, 'ÁGUA VERDE':14, 'BATEL':15,
'NOVO MUNDO':16, 'ALTO BOQUEIRÃO':17, 'CAPÃO RASO':18, 'JARDIM
BOTÂNICO':19, 'PORTÃO':20, 'ORLEANS':21, 'SANTA FELICIDADE':23,
'CASCATINHA':24, 'CAPÃO DA IMBUIA':25, 'BARREIRINHA':26, 'SEMINÁRIO':27,
'CAMPO COMPRIDO':28, 'PRADO VELHO':29, 'PINHEIRINHO':30, 'BUTIATUVINHA':31,
'CAMPINA DO SIQUEIRA':32, 'CAJURU':33, 'SÃO FRANCISCO':34, 'CENTRO
CÍVICO':35, 'SÃO BRAZ':36, 'UMBARÁ':37, 'CAXIMBA':38, 'JARDIM SOCIAL':39,
'BACACHERI':40, 'CAMPO DE SANTANA':41, 'SANTO INÁCIO':42, 'JARDIM DAS
AMÉRICAS':43, 'LINDÓIA':44, 'GANCHINHO':45, 'PAROLIN':46, 'ABRANCHES':47,
'SÃO JOÃO':48, 'ATUBA':49, 'TARUMÃ':50, 'ALTO DA RUA XV':51,
'MOSSUNGUÊ':52, 'TINGUI':53, 'BIGORRILHO':54, 'BAIRRO ALTO':55, 'HAUER':56,
'VILA IZABEL':57, 'CABRAL':58, 'BOM RETIRO':59, 'GUAÍRA':60,
'CACHOEIRA':61, 'AUGUSTA':62, 'CRISTO REI':63, 'AHÚ':64, 'ALTO DA
GLÓRIA':65, 'GUABIROTUBA':66, 'MERCÊS':67, 'SANTA QUITÉRIA':68, 'SÃO
MIGUEL':69, 'SÃO LOURENÇO':70, 'FANNY':71, 'JUVEVÊ':72, 'VISTA ALEGRE':73,
'HUGO LANGE':74, 'RIVIERA':75, 'LAMENHA PEQUENA':76, 'INDICAÇÕES
CANCELADA':77, 'BAIRRO NAO INFORMADO':78, 'BAIRRO FICTÍCIO':79, 'fanny':80,
'TINGÜI':81, 'CIDADE JARDIM':82, 'VENEZA':83, 'PLANTA MEIRELES':84,
'TANGUA':85, 'MONTE REY':86, 'JD EUROPA':87, 'BORDA DO CAMPO':88, 'JARDIM
BOA VISTA':89, 'SÃO JOSE':90, 'JARDIM COLONIAL':91, 'MENINO DEUS':92, 'SÃO
JUDAS TADEU':93, 'VILA MARIA ANTONIETA':94, 'MARIA ANTONIETA':95, 'SANTO
ANTONIO':96, 'COLOMBO':97, 'CANGUIRI':98, 'NÃO ENCONTRADO':99,
'FERRARIA':100, 'SÃO CRISTOVÃO':101, 'JD SUISSA':102, 'VILA FORMOSA':103,
'FORMOSO':104, 'SÃO PEDRO':105, 'SAO JOSE DOS PINHAIS':106, 'CAMPO
```

```
PEQUENO':107, 'PINHAIS':108, 'VILA PERNETA ':109, 'SEM DADOS':110, 'CAMPO
DE SÃO BENEDIT':111, 'QUATRO BARRAS':112, 'LOT. MARINONI':113, 'SÃO
JORGE':114, 'BAIRRO NÃO LOCALIZAD':115, 'BRAGA':116, 'JARDIM LOANDA':117,
'NÃO INFORMADO ':118, 'SANTA TEREZINHA':119, 'SANTA TERESINHA':120, 'JARDIM
WEISSOPOLIS':121, 'SITIO DAS PALMEIRAS':122, 'CAMPO PEQUENO ':123,
'NI':124, 'THOMAS COELHO':125, 'NF':126, 'SÃO THOMAS':127, 'JARDIM
INDUSTRIAL':128, 'ROÇA NEGRA':129, 'SÃO THOMAZ':130, 'GRALHA AZUL':131,
'MARACANÃ':132, 'VILA BANCÁRIA':133, 'JARDIM BOM PASTOR':134, 'SAO
GERONIMO':135, 'RIO VERDE':136, 'JD IPE':137, 'IGUAÇÚ 1':138, 'AGUAS
BELAS':139, 'ÁGUAS BELAS':140, 'IGUAÇU 01':141, 'ESTADOS':142, 'CIC':143,
'JR TAISA':144, 'PLANTA DEODORO':145, 'MAUA':146, 'COLONIA FARIA':147,
'NAÇÕES':148, 'JARDIM SANTA MÔNICA':149, 'LOTEAMENTO SÃO GERÔN':150,
'TAMANDARE ':151, 'CAMPO LARGO':152, 'BOQUEIRÃO ':153, 'JARDIM BELA
VISTA':154, 'ESTANCIA PINHAIS ':155, 'COLONIA SAO VENANCIO':156, 'FRANCISCO
GORSKI':157, 'OSASCO':158, 'BARIGUI':159, 'GUATUPE ':160, 'PARQUE DAS
NASCENTES':161, 'CENTRO ':162, 'JD. ORESTES THÁ':163, 'PARQUE DAS
FONTES':164, 'PINEVILLE':165, 'BORDA DO CAMPO ':166, ' JARDIM OSASCO':167,
'JARDIM PRIMAVERA':168, 'JD DONA BELIZARIA':169, 'PIRAQUARA':170, 'JARDIM
RAFAELA':171, 'BARRO PRETO':172, 'BELAS AGUAS':173, 'EUCALIPTOS':174, 'VILA
GRAZIELA':175, 'CIDADE INDUSTRIAL DE':178, 'AFONSO PENA':179,
'PALMEIRINHA':180, 'IPE 2':181, 'SANTA MONICA':182, 'GUATUPE':183, 'AFONSO
PENA ':184, 'SAO SEBASTIAO':185, 'MAUÁ':186, 'SÃO GERONIMO':187, 'OURO
FINO':188, 'SANTO ANTÔNIO':189, 'CAMPINHA GRANDE DO S':190, ' JARDIM PEDRO
DEMETE':191, 'ROÇA GRANDE':192, 'TINDIQUERA':193, 'SÃO BENEDITO':194})

df['ATENDIMENTO_BAIRRO_NOME'] =
df['ATENDIMENTO_BAIRRO_NOME'].astype(str).astype(int)

### NATUREZA1_DESCRICAO

df['NATUREZA1_DESCRICAO'] = df['NATUREZA1_DESCRICAO'].replace({'Apoio':1,
'Alarmes':2, 'Invasão':3, 'Vistoria':4, 'Roubo':5, 'Perturbação do
sossego':6, 'Trânsito':7, 'Risco de acidente/à vida (Defesa Civil)':8,
'Violação de Medida Protetiva Lei Maria da Penha':9, 'Dano':10, 'Lesão
Corporal':11, 'Fundada Suspeita (Abordagem)':12, 'Substância Ilícita':13,
'Orientação':14, 'Alagamento':15, 'Animais':16, 'Furto':17,
'Desinteligência':18, 'Patrulha Maria da Penha':19, 'Atitude Suspeita':20,
'Atos obscenos/libidinosos':21, 'Vias de fato':22, 'Queima a céu
aberto':23, 'Ameaça':24, 'Averiguação':25, 'Encaminhamento':26,
'Estupro':27, 'Saturação':28, 'Agressão física/verbal':29, 'AIFU':30,
'Escolta':31, 'Incêndio':32, 'Risco de acidente / à vida':33,
'Desacato':34, 'Paciente/usuário alterado':35, 'Veículo':36, 'Pesca em
local proibido':37, 'Ronda':38, 'Destelhamento':39, 'Construção
```

```
Irregular':40, 'Crime ambiental':41, 'Risco de desabamento /
desmoronamento':42, 'Tentativa':43, 'Fornecimento de Lona':44,
'Suicídio':45, 'Obstrução de via':46, 'Substância Lícita':47, 'Depósito
irregular':48, 'Corte irregular de árvore':49, 'Achado':50, 'Queda de
árvore':51, 'Disparo de arma':52, 'Órgãos acionados':53, 'Averiguação
(Defesa Civil)':54, 'Antecedentes Criminais - Verificação':55,
'Injúria':56, 'Desaparecimento':57, 'Manifestação':58, 'Seqüestro e cárcere
privado':59, 'Arrastão':60, 'Deslizamenton de Terra':61, 'ZELADORIA
URBANA':62, 'Desabamento':63, 'Devolução de coisa achada':64, 'Conduta
inconveniente':65, 'Uso indevido do cartão transporte':66, 'Maus tratos à
pessoas':67, 'Extravio de Equipamento':68, 'Porte Ilegal':69, 'Rixa':70,
'Erosão':71, 'Importunação\xa0sexual':72, 'Situação de risco':73, 'Queda de
fios de energia':74, 'Estelionato':75, 'Desobediência':76, 'Racismo':77,
'Homicídio':78, 'Queda de galho':79, 'Homofobia':80, 'Descumprimento lei
15799/2021 COVID-19':81, 'Fuga de aluno/interno':82, 'Menores abordando
transeuntes':83, 'Abandono de incapaz':84, 'Risco de queda de árvore':85,
'Retirada de invasão':86, 'Banho em local impróprio':87, 'Abuso de
incapazes':88, 'Contrabando ou descaminho':89, 'Criança
perdida/desaparecida':90, 'Extravio, sonegação ou inutilização de livro ou
doc.':91, 'Resistência':92, 'Aliciamento de menor':93, 'Apropriação
indébita':94, 'Proteção ao patrimônio':95, 'Infiltração':96, 'Roubo, furto,
extravio, recuperação, apreensão de armas de fogo.':97, 'Receptação':98,
'Ataque de insetos':99, 'Fiscalizações e Orientações':100, 'Vazamento ou
derramamento de Produto Perigoso ou Infectante':101, 'Falsidade ideológica
(Falsa Identidade)':102, 'Câmera Off-Line':103, 'Poluição
visual/ambiental':104, 'Óbito':105, 'Avaria em Equipamento/Patrimônio (não
intencional)':106, 'Fuga de paciente':107, 'Moeda Falsa':108,
'Embriaguez':109, 'Queda de poste':110, 'Material abandonado':111,
'Calote':112, 'Quedas de objetos ou partes de construções':113, 'Acidente
Viatura':114, 'Risco de queda de poste':105, 'Constrangimento ilegal':106,
'Comércio ambulante':107, 'Usar de uniforme, ou distintivo de função
pública que não exerce':108, 'Envenenamento':109, 'Denúncia de bomba':110,
'Mendigar, por ociosidade ou cupidez':111, 'Extorsão':112, 'Atentado
violento ao pudor':113, 'Verificação':114, 'Pragas Animais':115,
'Inundação/Enchente':116, 'Importunação ofensiva ao pudor':117, 'Jogo de
Azar':118, 'Porte de artefato explosivo':119, 'Maus tratos a animais':120,
'Calúnia':121, 'Sedução':122, 'Violência arbitrária':123, 'Afogamento':124,
'Explosão':125, 'Câmeras de videomonitoramento':126, 'Bueiro aberto/sem
tampa':127, 'Menor gazeando aula':128, 'Fornecimento de bebida alcoólica à
menores':129, 'Vadiagem':130, 'Discriminação':131, 'Escrito ou objeto
obsceno (panfletos pornográficos)':132, 'Favorecimento da
prostituição':133, 'Peculato':134, 'Impedimento ou perturbação de cerimônia
funerária':135, 'Risco de queda de fios de energia':136, 'Ataque cão
```

```
feroz':137, 'Abandono de função':138, 'Uso indevido do telefone
público':139, 'Aterro irregular':140, 'Risco de explosão':141, 'Obstrução
da Atividade Policial':142, 'Bueiro entupido':143, 'Corrupção de
menores':144, 'Queda de aeronave':145, 'Incendio/Explosão em
edificação':146, 'Vilipêndio a cadáver':147, 'Risco de queda de galho':148,
'Prostituição':149, 'Violação de sepultura/túmulo':150, 'Fingir-se
funcionário público':151, 'Trote Telefonico':152, 'Apologia de crime ou
criminoso':153, 'Falsificação de documento Publico':154, 'Denuncia
Improcedente':155, 'Quadrilha ou bando':156, 'Desabamento de
Telhado/Cobertura':157, 'Exploração de menores':158, 'Queda de Muro':159,
'Abalo Sísmico':160, 'Omissão de socorro':161, 'Rompimento de
Barragem':162, 'Liberação de pessoa presa/apreendida por recusa no
recebimento pela DP':163, 'Venda proibida de produtos específicos à
menores':164, 'Concussão':165, 'Charlatanismo':166, 'Difamação':167,
'RECUSAR SE IDENTIFICAR AO POLICIAL':168, 'Perseguição (stalking)':169,
'Enxurrada':170, 'Rufianismo':171, 'Incitação ao crime':172, 'Averiguação
(COSEDI)':173, 'Queda de Revestimento de Fachadas':174, 'Corrupção
ativa':175, 'Óbito (Defesa Civil)':176, 'Prevaricação':177})

df['NATUREZA1_DESCRICAO'] =
df['NATUREZA1_DESCRICAO'].astype(str).astype(int)

### OCORRENCIA_DIA_SEMANA

df['OCORRENCIA_DIA_SEMANA'] =
df['OCORRENCIA_DIA_SEMANA'].replace({'DOMINGO':1, 'SEGUNDA':2, 'TERÇA':3,
'QUARTA':4, 'QUINTA':5, 'SEXTA':6, 'SÁBADO':7})

df['OCORRENCIA_DIA_SEMANA'] =
df['OCORRENCIA_DIA_SEMANA'].astype(str).astype(int)

#.std() calcula o desvio padrão das colunas ou linhas numéricas
df.std()
```

```
ATENDIMENTO_BAIRRO_NOME    19.326657
NATUREZA1_DESCRICAO        16.781344
OCORRENCIA_ANO              3.113925
OCORRENCIA_DIA_SEMANA       2.062074
OCORRENCIA_HORA             6.611766
OCORRENCIA_MES              3.423113
OCORRENCIA_DIA              8.789952
dtype: float64
```

5. Preparação e Análise dos Dados

- **Análise exploratória do dataset do projeto:**
    - *seleção das ferramentas, técnicas e métricas de estatística descritiva; instalação, configuração e ajustes das ferramentas de análise estatística; resultado da estimativa estatística utilizando as medidas de posição e dispersão; descrição do protocolo para remoção de outliers; apresentação do processo de análise exploratória, medidas e métricas estatísticas utilizadas.*

Nessa próxima etapa realiza-se a análise descritiva e exploratória dos dados, para isso foram criadas e executadas algumas funções que calculam a média, mediana, variância, desvio padrão e quantis.

- **Média**

```
# Análise descritiva de dados

tabelas = [ocorrenciasAtendidasPorAno, ocorrenciasRegistradasPorAno,
ocorrenciasPorBairro, ocorrenciasFlagrante, ocorrenciasPorTipo,
ocorrenciasPorDia, ocorrenciasPorDiaDaSemana, ocorrenciasPorRegional]

### **- Medidas de posição**

#Média

def calculaMedia(tabela):

    media = tabela.mean()

    media = pd.DataFrame({'metricas':media.index, 'media':media.values})

    return media

medias = []
for i in tabelas:
  medias.append(calculaMedia(i))
```

```
dfMedia = pd.concat(medias)
dfMedia
```

	metricas	media
0	ATENDIMENTO_ANO	2015.500000
1	Quantidade_Ocorrencias_Atendidas_Por_Ano	29927.285714
0	OCORRENCIA_ANO	2015.500000
1	Quantidade_Ocorrencias_Registradas_Por_Ano	29933.642857
0	Quantidade_Ocorrencias_Registradas_Por_Bairro	2193.329843
0	Quantidade_Ocorrencias_Flagrante	209535.500000
0	Quantidade_Ocorrencias_Por_Tipo	2241.021390
0	Quantidade_Ocorrencias_Por_Dia	83.447033
0	Quantidade_Ocorrencias_Por_Dia_Da_Semana	59867.285714
0	Quantidade_Ocorrencias_Por_Regional	29932.857143

- **Mediana**

```
#Mediana
def calculaMediana(tabela):
    mediana = tabela.median()
    mediana = pd.DataFrame({'metricas':mediana.index,
'mediana':mediana.values})

    return mediana
medianas = []
for i in tabelas:
  medianas.append(calculaMediana(i))

dfMediana = pd.concat(medianas)
dfMediana
```

	metricas	mediana
0	ATENDIMENTO_ANO	2015.5
1	Quantidade_Ocorrencias_Atendidas_Por_Ano	24685.5
0	OCORRENCIA_ANO	2015.5
1	Quantidade_Ocorrencias_Registradas_Por_Ano	24686.5
0	Quantidade_Ocorrencias_Registradas_Por_Bairro	2.0
0	Quantidade_Ocorrencias_Flagrante	209535.5
0	Quantidade_Ocorrencias_Por_Tipo	69.0
0	Quantidade_Ocorrencias_Por_Dia	69.0
0	Quantidade_Ocorrencias_Por_Dia_Da_Semana	59549.0
0	Quantidade_Ocorrencias_Por_Regional	29905.5

- **Variância**

```
#Variância (valores ao quadrado)
#Distância dos termos com relação a média
#Quanto menor, melhor e mais próximo a média
def calculaVariancia(tabela):
    var = tabela.var()
    var = pd.DataFrame({'metricas':var.index, 'var':var.values})

    return var

variancias = []
for i in tabelas:
  variancias.append(calculaVariancia(i))

dfVariancia = pd.concat(variancias)
dfVariancia
```

	metricas	var
0	ATENDIMENTO_ANO	1.750000e+01
1	Quantidade_Ocorrencias_Atendidas_Por_Ano	1.650757e+08
0	OCORRENCIA_ANO	1.750000e+01
1	Quantidade_Ocorrencias_Registradas_Por_Ano	1.651473e+08
0	Quantidade_Ocorrencias_Registradas_Por_Bairro	4.232629e+07
0	Quantidade_Ocorrencias_Flagrante	7.271353e+10
0	Quantidade_Ocorrencias_Por_Tipo	5.868598e+07
0	Quantidade_Ocorrencias_Por_Dia	1.990981e+03
0	Quantidade_Ocorrencias_Por_Dia_Da_Semana	2.607052e+07
0	Quantidade_Ocorrencias_Por_Regional	1.133209e+09

- **Desvio Padrão**

```
#Desvio Padrão (√ Variância - mesma unidade media)
#Distância dos termos com relação à média
#Quanto menor, melhor e mais próximo a média

def calculateDesvioPadrao(tabela):

    desvioP = tabela.std()

    desvioP = pd.DataFrame({'metricas':desvioP.index,
'desvioP':desvioP.values})

    return desvioP

desvios = []
for i in tabelas:
  desvios.append(calculateDesvioPadrao(i))

dfDesvio = pd.concat(desvios)
dfDesvio
```

	metricas	desvioP
0	ATENDIMENTO_ANO	4.183300
1	Quantidade_Ocorrencias_Atendidas_Por_Ano	12848.177108
0	OCORRENCIA_ANO	4.183300
1	Quantidade_Ocorrencias_Registradas_Por_Ano	12850.963637
0	Quantidade_Ocorrencias_Registradas_Por_Bairro	6505.865766
0	Quantidade_Ocorrencias_Flagrante	269654.463899
0	Quantidade_Ocorrencias_Por_Tipo	7660.677328
0	Quantidade_Ocorrencias_Por_Dia	44.620405
0	Quantidade_Ocorrencias_Por_Dia_Da_Semana	5105.930007
0	Quantidade_Ocorrencias_Por_Regional	33663.175320

- **Quantis**

```
#Quantis
def calculaQuantil(tabela):

    q1 = tabela.quantile(0.25)
    q1Df = pd.DataFrame({'metricas':q1.index, 'q1':q1.values})

    q2 = tabela.quantile(0.50)
    q2Df = pd.DataFrame({'metricas':q2.index, 'q2':q1.values})

    q3 = tabela.quantile(0.75)
    q3Df = pd.DataFrame({'metricas':q3.index, 'q3':q3.values})

    quantile = pd.concat([q1Df['metricas'], q1Df['q1'], q2Df['q2'],
q3Df['q3']], axis=1, ignore_index=True)
    quantile.columns = ['metricas', 'quantil1(25%)', 'quantil2(50%)',
'quantil3(75%)']

    return quantile

quantis = []
for i in tabelas:
  quantis.append(calculaQuantil(i))

dfQuantis = pd.concat(quantis)
dfQuantis
```

	metricas	quantil1(25%)	quantil2(50%)	quantil3(75%)
0	ATENDIMENTO_ANO	2012.25	2012.25	2018.75
1	Quantidade_Ocorrencias_Atendidas_Por_Ano	22232.50	22232.50	31707.00
0	OCORRENCIA_ANO	2012.25	2012.25	2018.75
1	Quantidade_Ocorrencias_Registradas_Por_Ano	22235.50	22235.50	31713.00
0	Quantidade_Ocorrencias_Registradas_Por_Bairro	1.00	1.00	1996.50
0	Quantidade_Ocorrencias_Flagrante	114198.25	114198.25	304872.75
0	Quantidade_Ocorrencias_Por_Tipo	11.00	11.00	554.50
0	Quantidade_Ocorrencias_Por_Dia	56.00	56.00	95.00
0	Quantidade_Ocorrencias_Por_Dia_Da_Semana	56150.50	56150.50	63028.50
0	Quantidade_Ocorrencias_Por_Regional	2760.25	2760.25	37635.00

A etapa seguinte cria-se e executa-se funções para gráficos, para poder visualizar todas as variáveis que foram selecionadas e observadas anteriormente. Os gráficos são: gráfico de distribuição, histograma com bins, gráfico de pizza, gráfico de linha e boxplot.

```
def criaHistogramaComBins(title, table, bins, color, collumn = None):

    plt.figure(figsize=(10,5))

    plt.title(title, fontsize =13)

    if collumn is None:

        plt.hist((table), bins = np.array(bins), alpha = 0.8, color = color)

    else:

        plt.hist((table[collumn]), bins = np.array(bins), alpha = 0.8, color =
color)

    plt.show
def plotScatter(collumnX, collumnY, table, title, color):

    sns.set_style('white')

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

    plt.title(title, fontsize = 13)
```

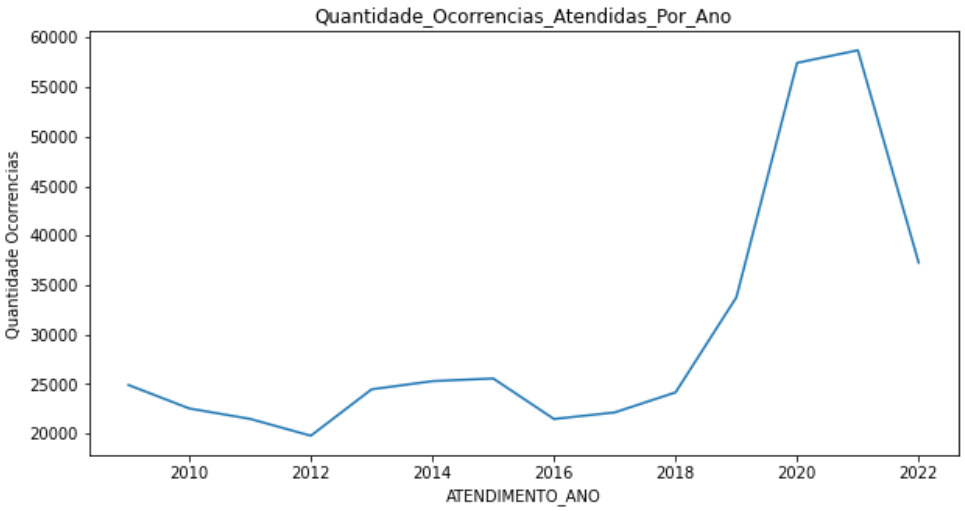
```
    sns.scatterplot(x=collumnX, y=collumnY, data= table, color = color)

    plt.show()

def criaGraficoPizza(table, y, label, title):
    mycolors = ['plum', 'bisque']

    plt.figure(figsize=(15,10), dpi=80)

    plt.pie(table[y], labels = table[label], autopct = '%1.1f%%', colors =
mycolors, frame = False)

    #plt.legend()

    plt.title(title)

    plt.rcParams['axes.facecolor'] = 'white'

    plt.show()

def criaGraficoLinha(table, x, y, title, xLabel, yLabel):

    plt.figure(figsize=(10,5))

    plt.plot(table[x], table[y])

    plt.title(title)

    plt.xlabel(xLabel)

    plt.ylabel(yLabel)

    plt.show()

def criaGraficoDistribuicao(table, coluna):

    plt.figure(figsize=(10,5))

    sns.distplot(table[coluna])

def criaGraficoBoxplot(table, x):

    plt.figure(figsize=(10,5))
```

```
    sns.boxplot(data=table, x=x)
```

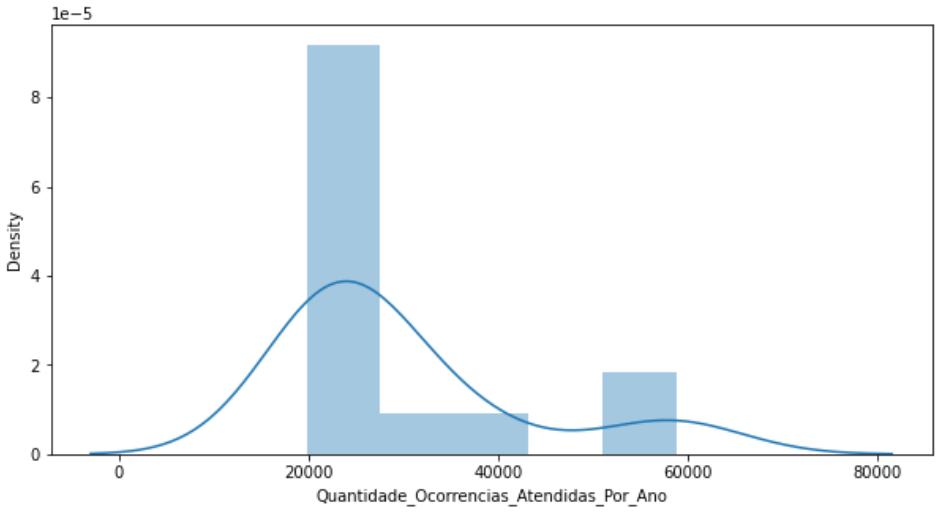
Visualizações dos dados

- **Ocorrências Atendidas Por Ano**

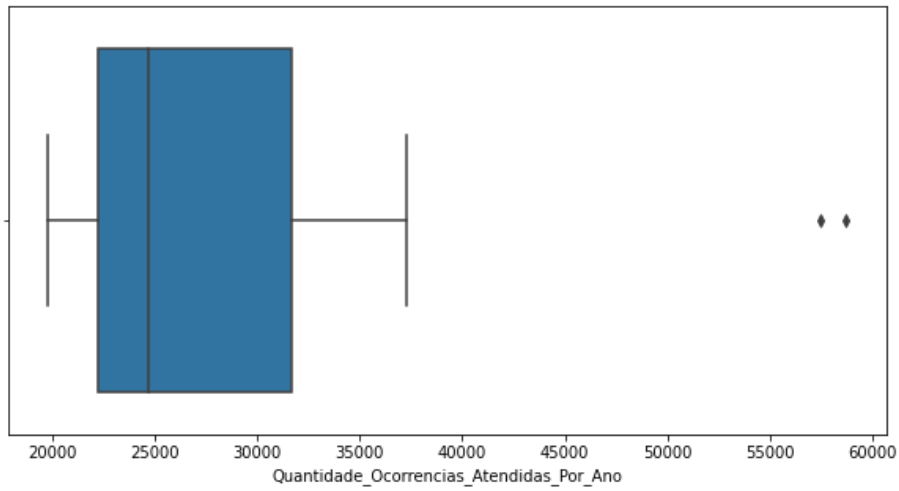
```
## Visualizações dos dados

#####**Ocorrencias Atendidas Por Ano**

criaGraficoLinha(ocorrenciasAtendidasPorAno, 'ATENDIMENTO_ANO',
'Quantidade_Ocorrencias_Atendidas_Por_Ano',
'Quantidade_Ocorrencias_Atendidas_Por_Ano', 'ATENDIMENTO_ANO', 'Quantidade
Ocorrencias')
```



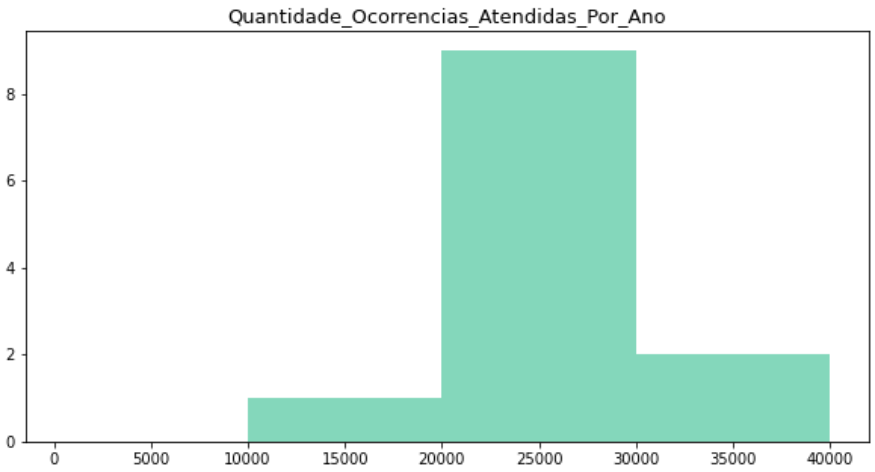

```
criaGraficoDistribuicao(ocorrenciasAtendidasPorAno,
'Quantidade_Ocorrencias_Atendidas_Por_Ano')
```



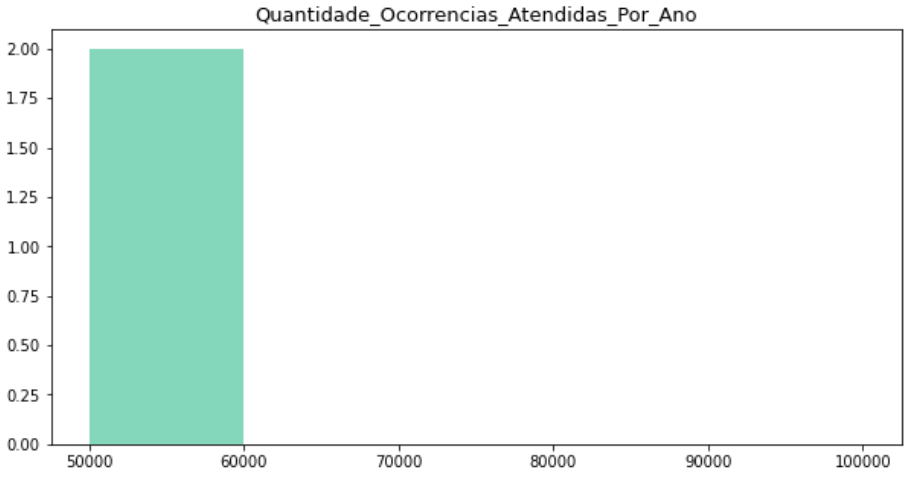

```
criaGraficoBoxplot(ocorrenciasAtendidasPorAno,
'Quantidade_Ocorrencias_Atendidas_Por_Ano')
```



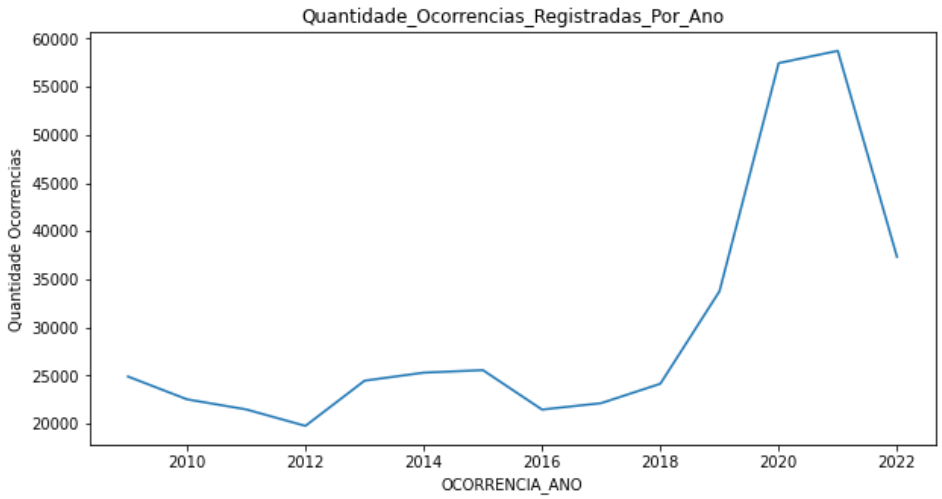

```
bins = [500, 10000, 20000, 30000, 40000]
criaHistogramaComBins('Quantidade_Ocorrencias_Atendidas_Por_Ano',
ocorrenciasAtendidasPorAno, bins, 'mediumaquamarine',
'Quantidade_Ocorrencias_Atendidas_Por_Ano')
```




```
bins = [50000, 60000, 70000, 80000, 100000]
criaHistogramaComBins('Quantidade_Ocorrencias_Atendidas_Por_Ano',
ocorrenciasAtendidasPorAno, bins, 'mediumaquamarine',
'Quantidade_Ocorrencias_Atendidas_Por_Ano')
```

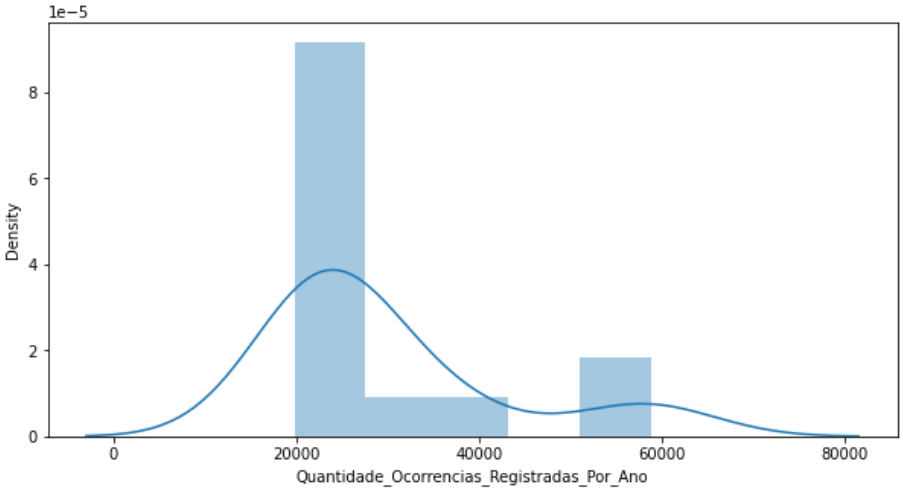



```
criaGraficoLinha(ocorrenciasRegistradasPorAno, 'OCORRENCIA_ANO',
'Quantidade_Ocorrencias_Registradas_Por_Ano',
'Quantidade_Ocorrencias_Registradas_Por_Ano', 'OCORRENCIA_ANO', 'Quantidade
Ocorrencias')
```

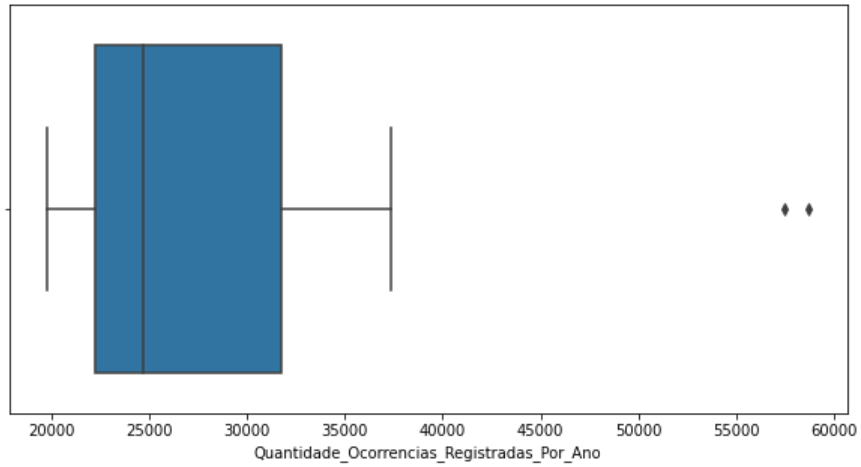


- **Ocorrências Registradas Por Ano**

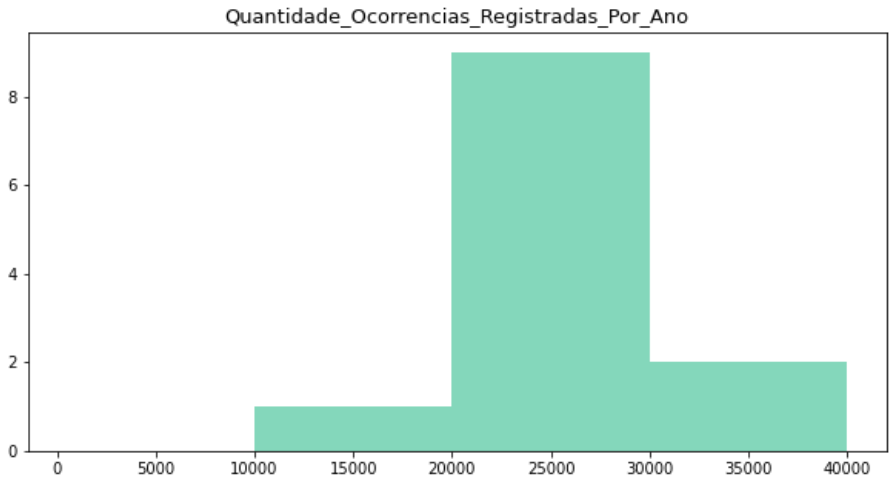
```
criaGraficoDistribuicao(ocorrenciasRegistradasPorAno,
'Quantidade_Ocorrencias_Registradas_Por_Ano')
```



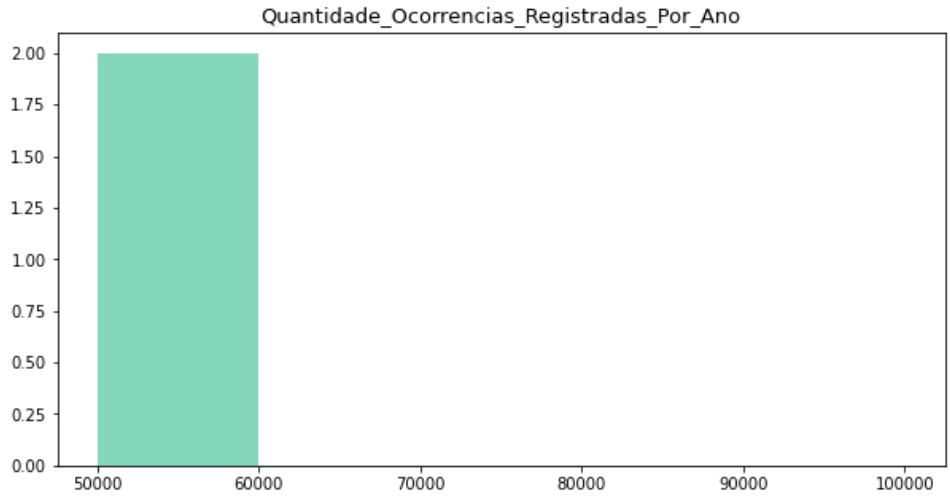

```
criaGraficoBoxplot(ocorrenciasRegistradasPorAno,
'Quantidade_Ocorrencias_Registradas_Por_Ano')
```




```
bins = [500, 10000, 20000, 30000, 40000]

criaHistogramaComBins('Quantidade_Ocorrencias_Registradas_Por_Ano',
ocorrenciasRegistradasPorAno, bins, 'mediumaquamarine',
'Quantidade_Ocorrencias_Registradas_Por_Ano')
```

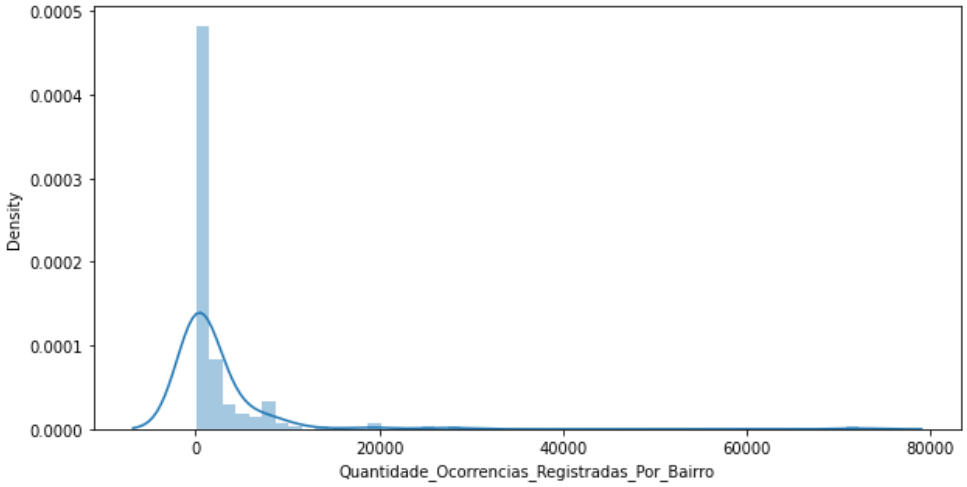



```
bins = [50000, 60000, 70000, 80000, 100000]
criaHistogramaComBins('Quantidade_Ocorrencias_Registradas_Por_Ano',
ocorrenciasRegistradasPorAno, bins, 'mediumaquamarine',
'Quantidade_Ocorrencias_Registradas_Por_Ano')
```

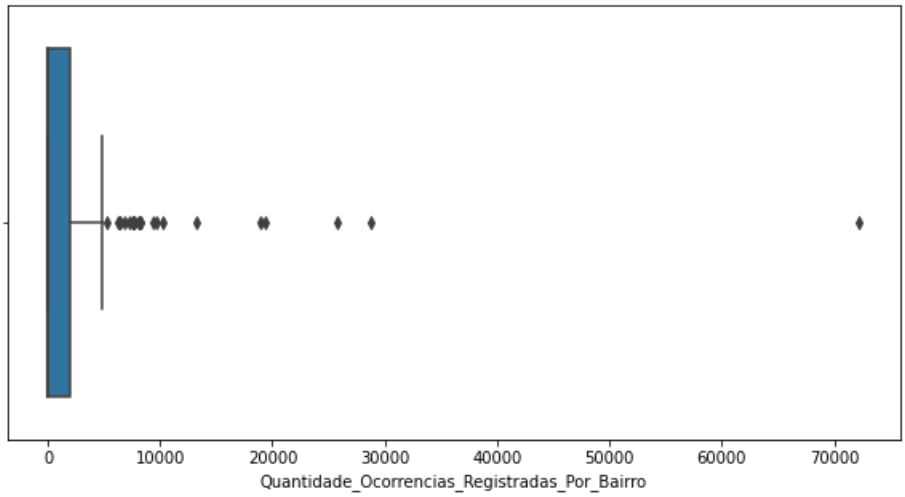


- **Ocorrências Por Bairro**

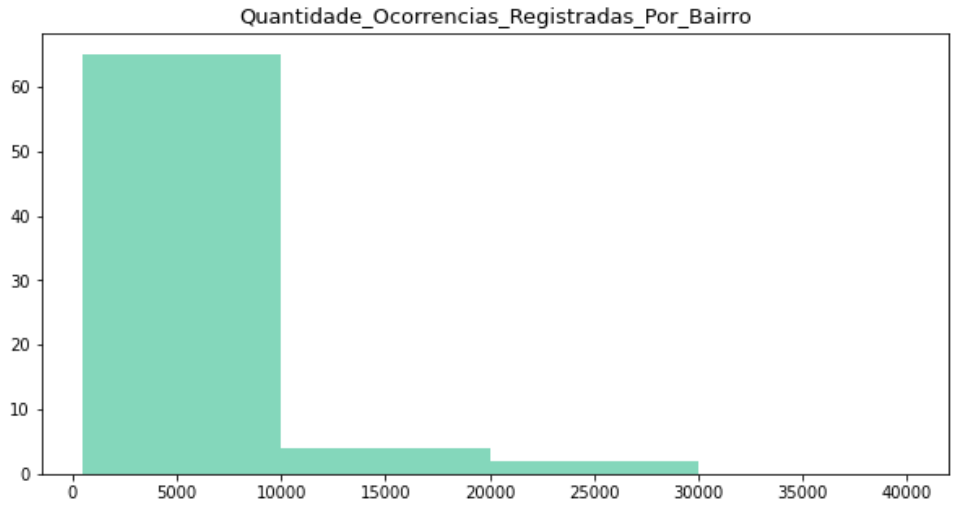
```
#####**Ocorrencias Por Bairro**

criaGraficoDistribuicao(ocorrenciasPorBairro,
'Quantidade_Ocorrencias_Registradas_Por_Bairro')
```




```
criaGraficoBoxplot(ocorrenciasPorBairro,
'Quantidade_Ocorrencias_Registradas_Por_Bairro')
```



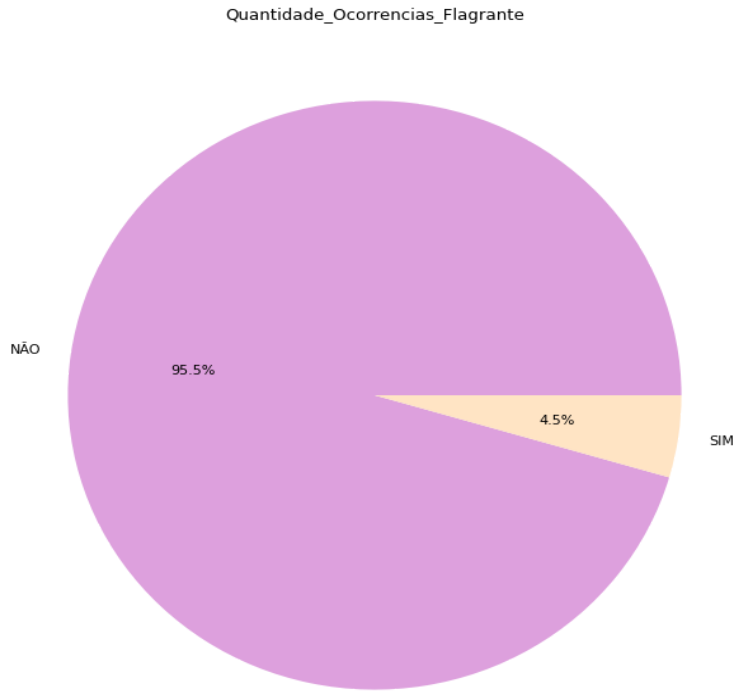

```
bins = [500, 10000, 20000, 30000, 40000]
criaHistogramaComBins('Quantidade_Ocorrencias_Registradas_Por_Bairro',
ocorrenciasPorBairro, bins, 'mediumaquamarine',
'Quantidade_Ocorrencias_Registradas_Por_Bairro')
```




```
bins = [50000, 60000, 70000, 80000, 100000]

criaHistogramaComBins('Quantidade_Ocorrencias_Registradas_Por_Bairro',
ocorrenciasPorBairro, bins, 'mediumaquamarine',
'Quantidade_Ocorrencias_Registradas_Por_Bairro')
```

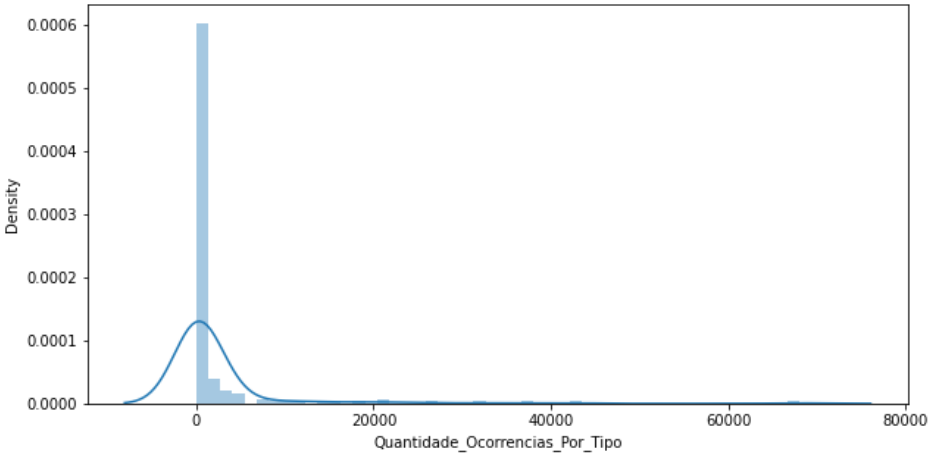
- **Ocorrências Flagrante**

```
####**Ocorrencias Flagrante**

quantidadeOcorrenciasFlagranteGraficoPizza =
criaGraficoPizza(ocorrenciasFlagrante, 'Quantidade_Ocorrencias_Flagrante',
'FLAG_FLAGRANTE', 'Quantidade_Ocorrencias_Flagrante')
```

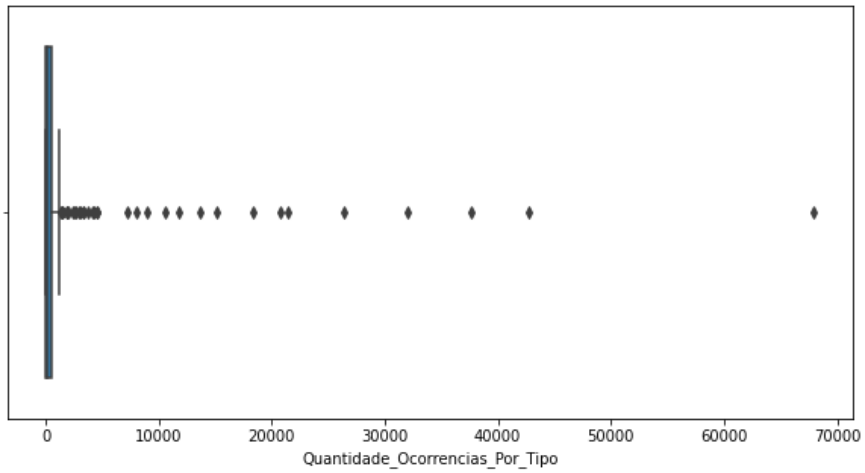


- **Ocorrências Atendidas Por Tipo**

```
#####**Ocorrencias Por Tipo**

criaGraficoDistribuicao(ocorrenciasPorTipo,
'Quantidade_Ocorrencias_Por_Tipo')
```

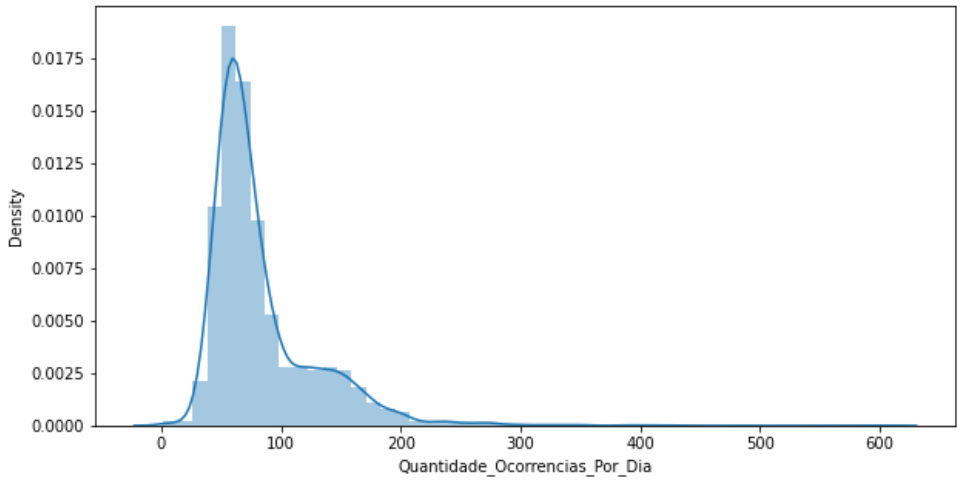



```
criaGraficoBoxplot(ocorrenciasPorTipo, 'Quantidade_Ocorrencias_Por_Tipo')
```

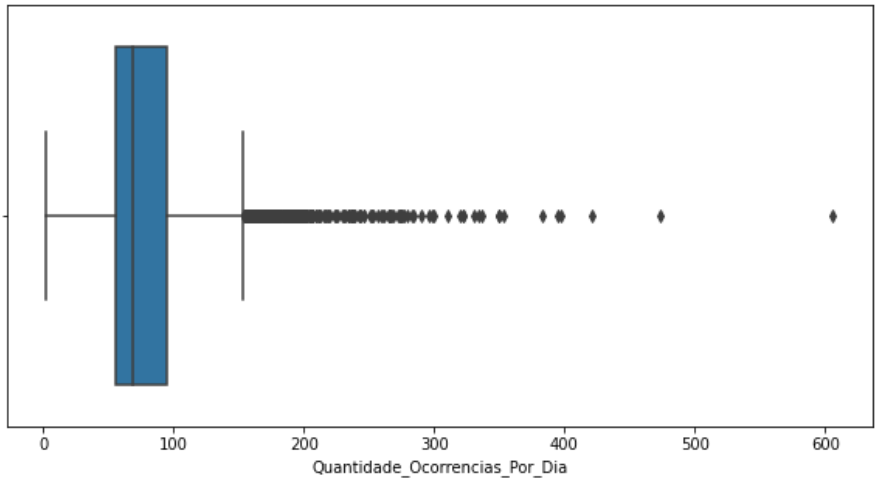


- **Ocorrências Atendidas Por Dia**

```
#####**Ocorrencias Por Dia**

criaGraficoDistribuicao(ocorrenciasPorDia,
'Quantidade_Ocorrencias_Por_Dia')
```

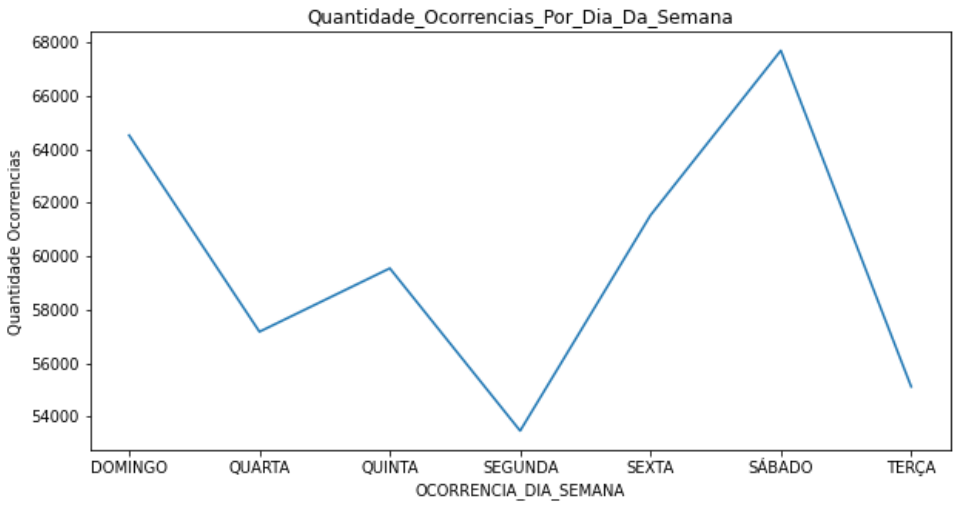



```
criaGraficoBoxplot(ocorrenciasPorDia, 'Quantidade_Ocorrencias_Por_Dia')
```

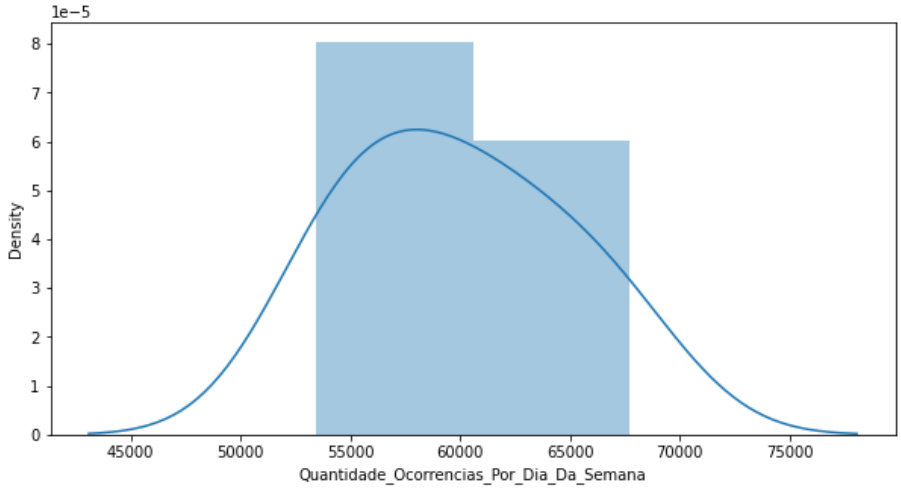


- **Ocorrências Atendidas Por Dia Da Semana**

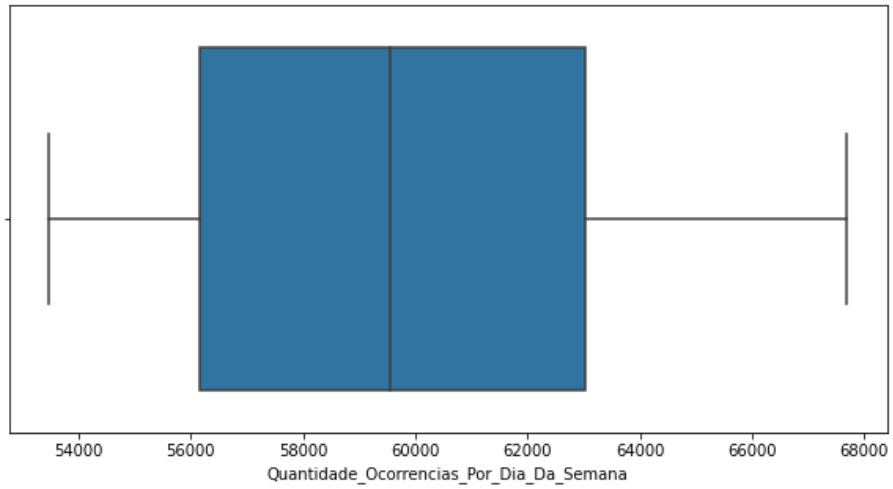
```
####**Ocorrencias Por Dia Da Semana**

criaGraficoLinha(ocorrenciasPorDiaDaSemana, 'OCORRENCIA_DIA_SEMANA',
'Quantidade_Ocorrencias_Por_Dia_Da_Semana',
'Quantidade_Ocorrencias_Por_Dia_Da_Semana', 'OCORRENCIA_DIA_SEMANA',
'Quantidade Ocorrencias')
```




```
criaGraficoDistribuicao(ocorrenciasPorDiaDaSemana,
'Quantidade_Ocorrencias_Por_Dia_Da_Semana')
```

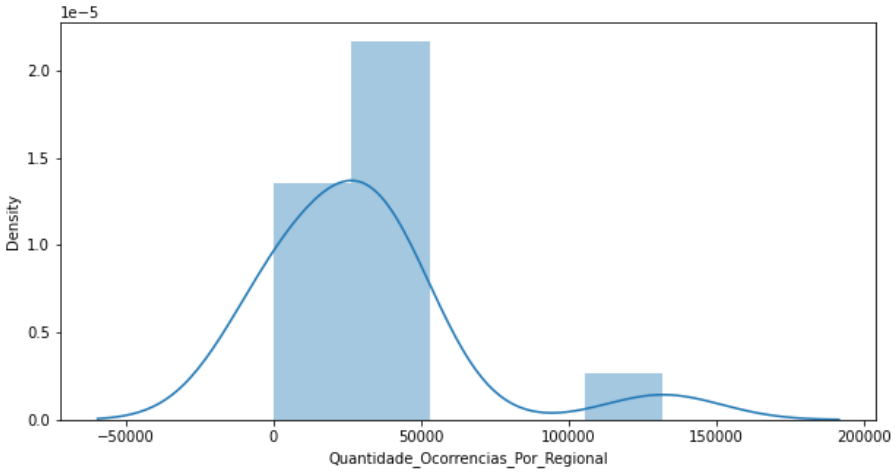



```
criaGraficoBoxplot(ocorrenciasPorDiaDaSemana,
'Quantidade_Ocorrencias_Por_Dia_Da_Semana')
```

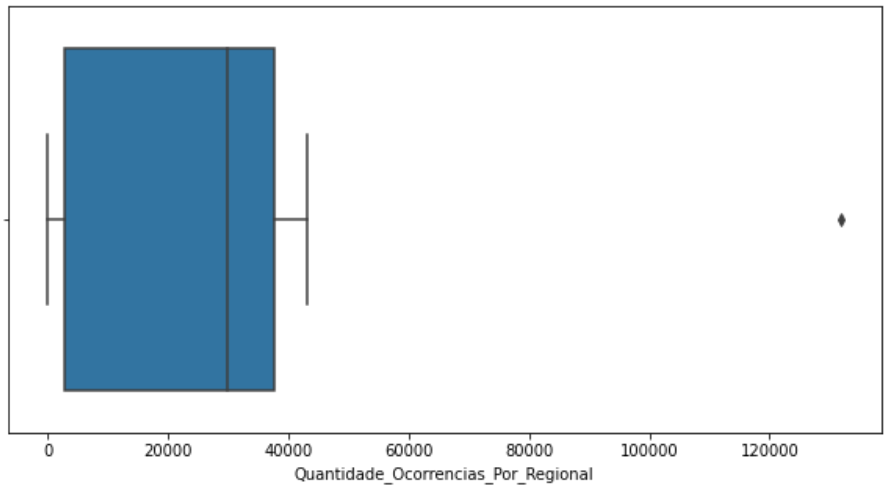


- **Ocorrências Atendidas Por Regional**

```
#####**Ocorrências Por Regional**

criaGraficoDistribuicao(ocorrenciasPorRegional,
'Quantidade_Ocorrencias_Por_Regional')
```



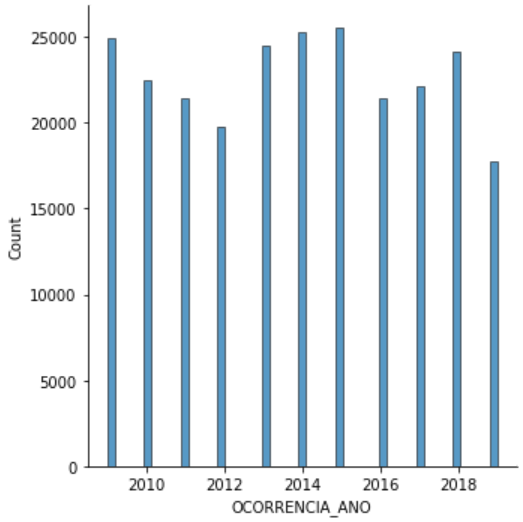

```
criaGraficoBoxplot(ocorrenciasPorRegional,
'Quantidade_Ocorrencias_Por_Regional')
```



Com base nas análises observadas anteriormente, foram criadas as perguntas a seguir para serem respondidas.

- **Qual ano possui mais atendimento?**

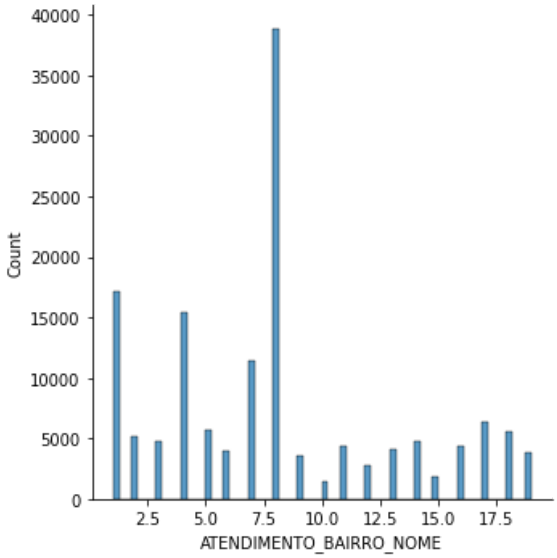
```
plt.figure(figsize=(20,20))
sns.displot(data=df,  x='OCORRENCIA_ANO')
```




```
Resposta: 2021, 2020 e 2022
```

- **Qual bairro possui mais atendimento?**

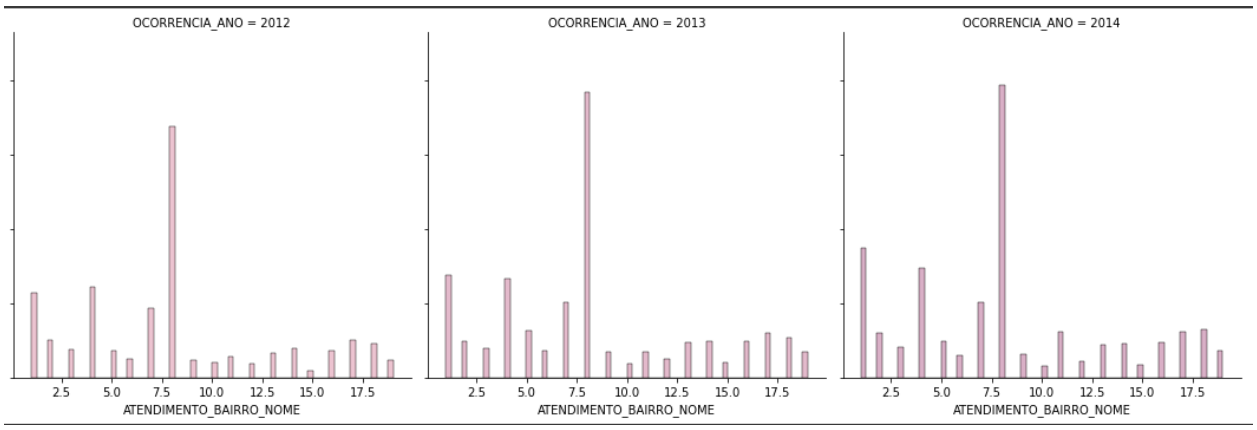
```
plt.figure(figsize=(10,10))
sns.displot(data=df[df['ATENDIMENTO_BAIRRO_NOME']<20],
x='ATENDIMENTO_BAIRRO_NOME')
```

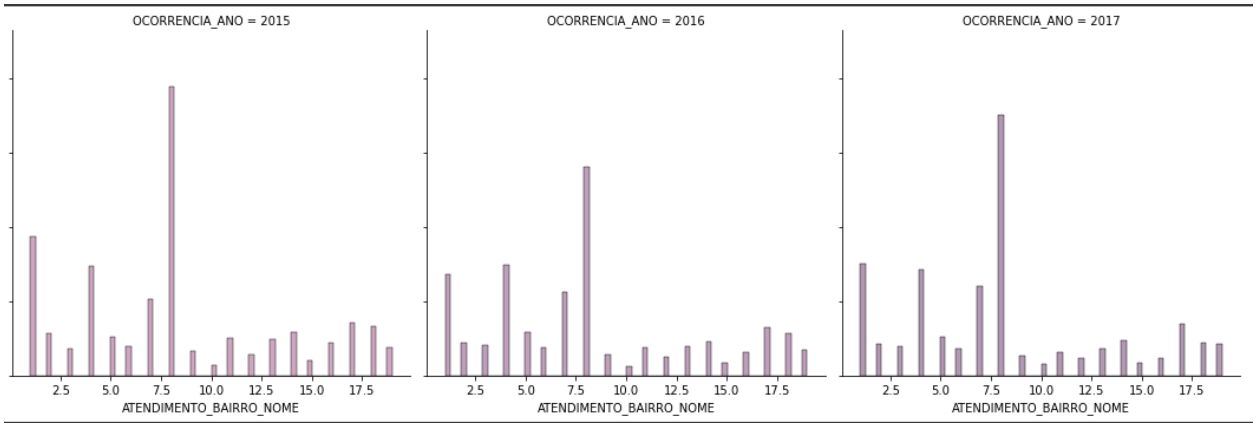


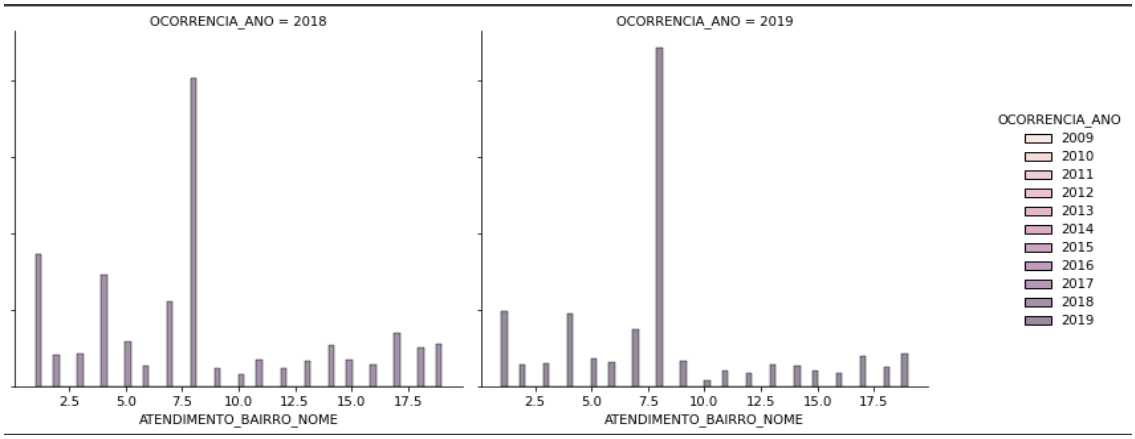

```
Resposta: 8: Centro, 1: CIC e 4: SÍTIO CERCADO
```

- **Qual ano por bairro possui mais atendimento?**

```
plt.figure(figsize=(100,80))
sns.displot(data=df[df['ATENDIMENTO_BAIRRO_NOME']<20],
x='ATENDIMENTO_BAIRRO_NOME', hue='OCORRENCIA_ANO', col='OCORRENCIA_ANO',
color='red')
```







Resposta:

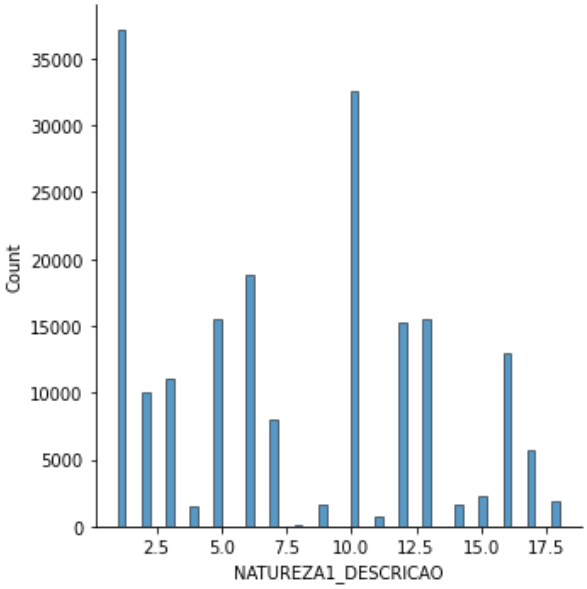
2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018 e 2019;
1ºlugar: 8, Centro.
2ºlugar: 1, CIC.
3ºlugar: 4: SÍTIO CERCADO.

2020:
1ºlugar: 8, Centro.
2ºlugar: 4: SÍTIO CERCADO.
3ºlugar: 1, CIC.

2021: 2022
1ºlugar: 8, Centro.
2ºlugar: 1, CIC.
3ºlugar: 4: SÍTIO CERCADO.

- **Qual o tipo de natureza da ocorrência?**

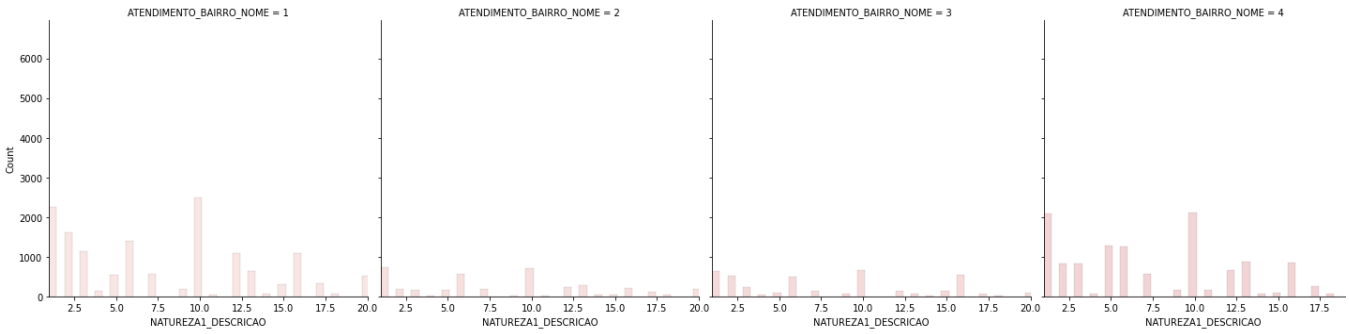
```
plt.figure(figsize=(20,20))
sns.displot(data=df[df['NATUREZA1_DESCRICAO']<20],  x='NATUREZA1_DESCRICAO')
```

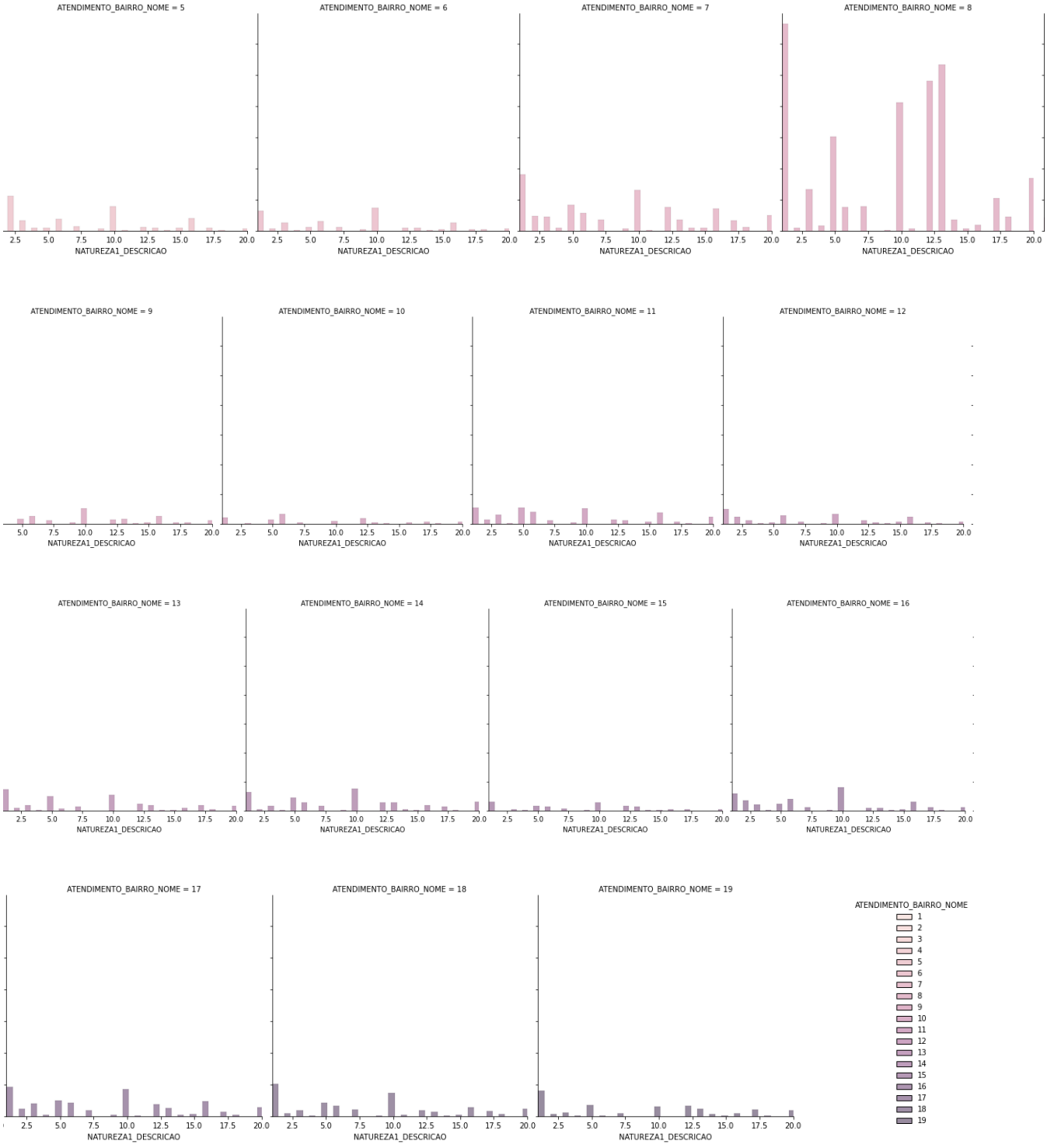



```
Resposta: 1: Apoio, 12: Fundada Suspeita (Abordagem) e 10: Dano
```

- **Qual o tipo de natureza da ocorrência em cada bairro?**

```
plt.figure(figsize=(20,20))
sns.displot(data=df[df['ATENDIMENTO_BAIRRO_NOME']<20],
x='NATUREZA1_DESCRICAO', hue='ATENDIMENTO_BAIRRO_NOME',
col='ATENDIMENTO_BAIRRO_NOME', color=['black'])
plt.xlim(1,20)
```

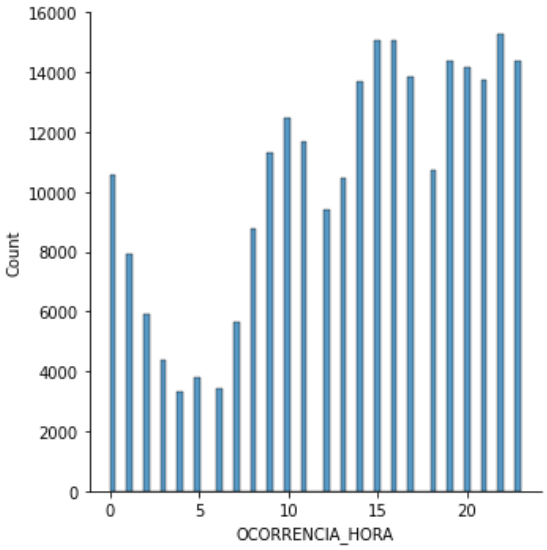




Resposta: foram utilizados só os 20 tipos de ATENDIMENTO_BAIRRO_NOME e ATENDIMENTO BAIRRO NOME, que se tem mais registro.

- **Qual horário da ocorrência mais frequente?**

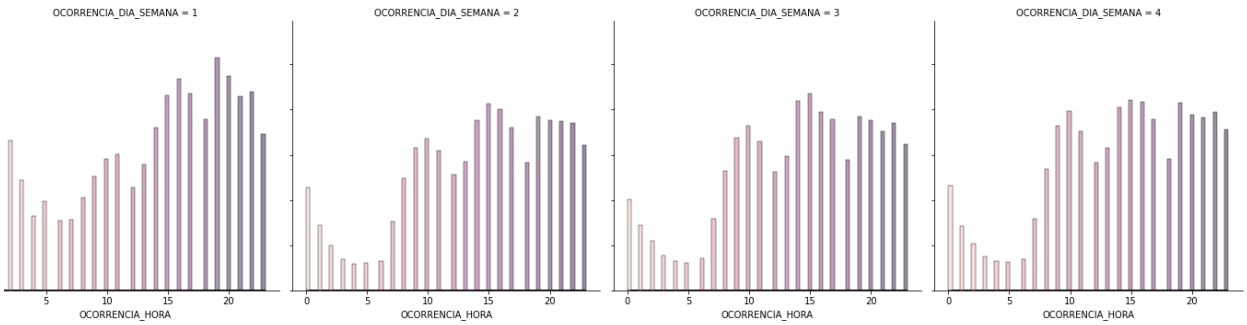
```
plt.figure(figsize=(20,20))
sns.displot(data=df,  x='OCORRENCIA_HORA')
```

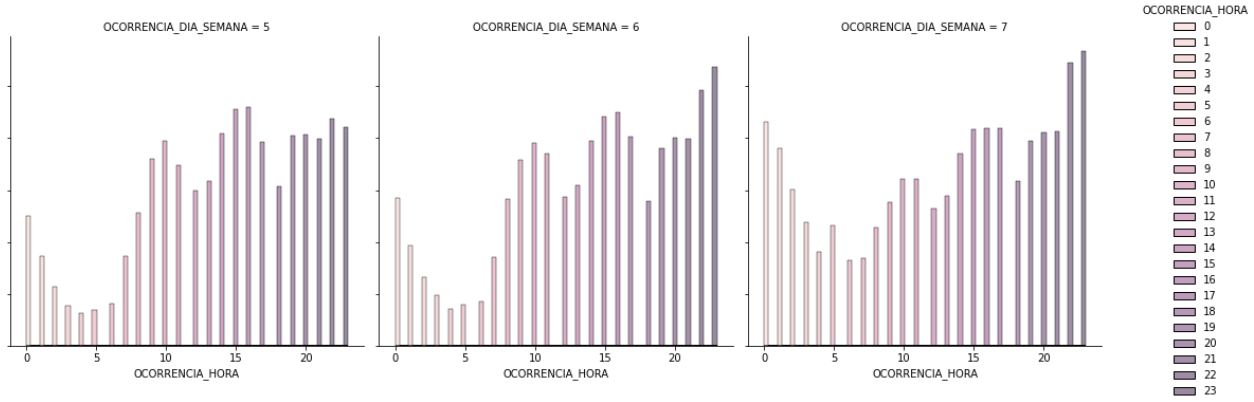



```
Resposta: 15, 14 e 16
```

- **Qual horário da semana e bairro mais frequentes?**

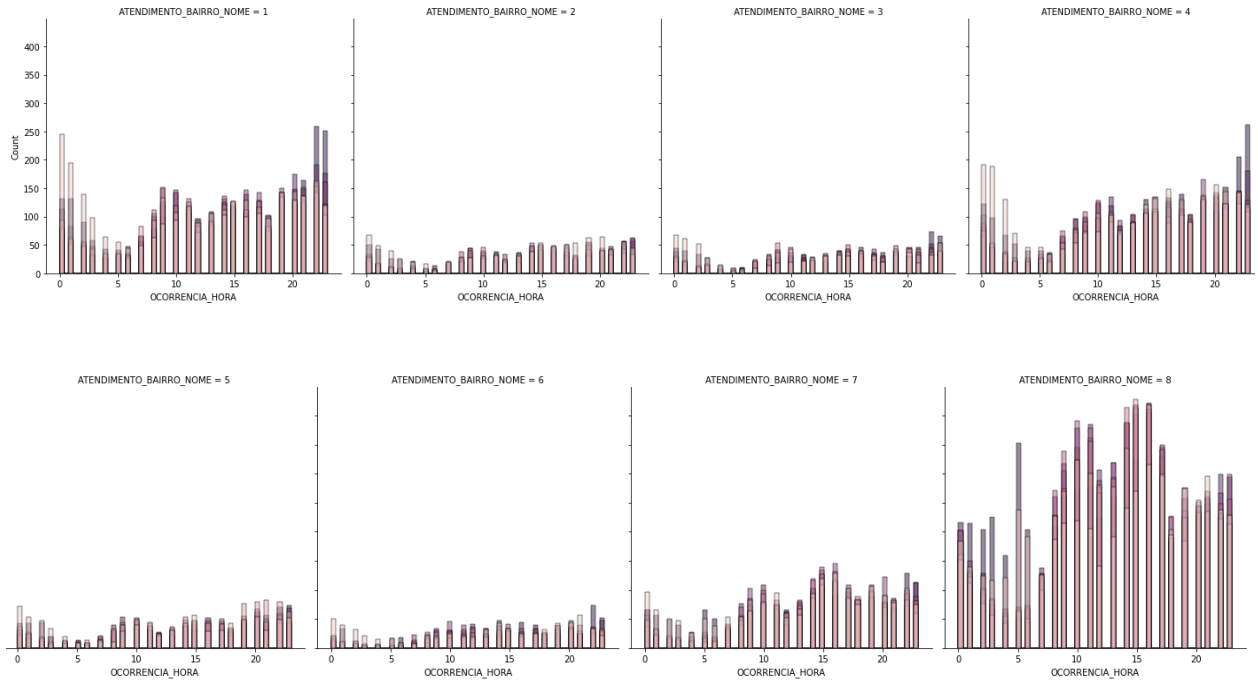
```
plt.figure(figsize=(20,20))
sns.displot(data=df,  x='OCORRENCIA_HORA', hue='OCORRENCIA_HORA',
col='OCORRENCIA_DIA_SEMANA')
```





Resposta: final de semana a mais ocorrência a tarde, 13h até umas 20h

```
plt.figure(figsize=(20,20))
sns.displot(data=df[df['ATENDIMENTO_BAIRRO_NOME']<20],
x='OCORRENCIA_HORA', hue='OCORRENCIA_DIA_SEMANA',
col='ATENDIMENTO_BAIRRO_NOME', color=['black'])
```




```
Resposta: Bairro 8 que é o Centro tem mais registro de manhã e à
tarde, 1 CIC e 4 SÍTIO CERCADO, se mantém igual com descanso só na
madrugada
```

- **Descrição dos experimentos com os frameworks de big data:**

    - *descrição das ferramentas utilizadas;*

A principal ferramenta utilizada para análise foi o PySpark, que é a API do Spark para Python. O Apache Spark é a estrutura de código aberto que proporciona o processamento em paralelo, ao mesmo tempo que demonstra a possibilidade do

suporte ao processamento na memória e auxilia no desempenho dos aplicativos que utilizam e analisam contextos de Big Data.

- *apresentação da motivação do uso do framework;*

A principal motivação para a utilização do framework de Big Data Apache Spark foi a relação com a velocidade de processamento e também da possibilidade processamento distribuído dos dados.

Durante o trabalho com o dataset um dos principais desafios foi o processamento dos dados. O dataset utilizado possui um total de 419072 linhas, o que torna o trabalho de processamento bastante custoso.

Sendo assim, optou-se pelo uso do Apache Spark, pois é uma estrutura rápida que possibilita o acesso a uma API que pode ser utilizada diretamente na linguagem Python, que é o PySpark.

- *apresentação das técnicas utilizadas; descrição do protocolo experimental; comparação entre as diferentes técnicas de big data (mínimo três técnicas diferentes); análise dos resultados obtidos.*

A primeira etapa para iniciar as análises e transformações é configurar o ambiente do Spark, para isso é necessário seguir todo o processo de instalação, importações e configurações a seguir.

Após todas as configurações necessárias, é preciso também criar uma sessão para inicializar todo o processo. Depois disso, basta criar o dataframe em Spark, passando o dataframe já utilizado anteriormente, utilizando spark.createDataFrame(). E então realiza-se o uso do método printSchema() para mostrar os tipos de dados e o show() para visualizar o dataframe carregado.

Sendo assim, é realizada também a utilização das funções já utilizadas anteriormente fora do Spark, como a describe() e head() para obter mais informações do dataset.

A primeira análise que é feita agora utilizando funções do Spark é a Contagem de colunas duplicadas e depois buscar colunas numéricas e categóricas e os valores nulos.

Análises com Spark

- **Configurando o Spark e Experimentos Iniciais com o Dataframe**

```
## Análises com Spark

Configurando o Spark

# install java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# install spark (change the version number if needed)
!wget -q
https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-had
oop3.2.tgz

# unzip the spark file to the current folder
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# set your spark folder to your system path environment.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"


# install findspark using pip
!pip install -q findspark

import findspark
findspark.init()

import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder\
        .master("local")\
        .appName("Colab")\
        .config('spark.ui.port', '4050')\
```

```
        .getOrCreate()

sparkDF=spark.createDataFrame(df2)
sparkDF.printSchema()
sparkDF.show()
```

```
root
 |-- ATENDIMENTO_ANO: double (nullable = true)
 |-- OCORRENCIA_ANO: long (nullable = true)
 |-- ATENDIMENTO_BAIRRO_NOME: string (nullable = true)
 |-- FLAG_FLAGRANTE: string (nullable = true)
 |-- NATUREZA1_DESCRICAO: string (nullable = true)
 |-- OCORRENCIA_DATA: string (nullable = true)
 |-- OCORRENCIA_DIA_SEMANA: string (nullable = true)
 |-- REGIONAL_FATO_NOME: string (nullable = true)
 |-- QUANTIDADE_OCORRENCIA: long (nullable = true)

+---------------+--------------+----------------------+--------------+-------------------+---------------+--------------
|ATENDIMENTO_ANO|OCORRENCIA_ANO|ATENDIMENTO_BAIRRO_NOME|FLAG_FLAGRANTE|NATUREZA1_DESCRICAO|OCORRENCIA_DATA|OCORRENCIA_DIA
+---------------+--------------+----------------------+--------------+-------------------+---------------+--------------
|         2009.0|          2009|              ABRANCHES|           NÃO|              AIFU|     2009-05-17|
|         2009.0|          2009|              ABRANCHES|           NÃO|              AIFU|     2009-05-28|
|         2009.0|          2009|              ABRANCHES|           NÃO|              AIFU|     2009-07-23|
|         2009.0|          2009|              ABRANCHES|           NÃO|         Alagamento|     2009-11-16|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Alarmes|     2009-09-03|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Alarmes|     2009-09-27|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Alarmes|     2009-11-07|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-01-04|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-01-07|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-02-06|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-02-16|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-04-07|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-04-23|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-05-11|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-06-27|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-07-03|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-07-22|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-08-04|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-08-27|
|         2009.0|          2009|              ABRANCHES|           NÃO|           Animais|     2009-09-20|
+---------------+--------------+----------------------+--------------+-------------------+---------------+--------------
only showing top 20 rows
```

```
sparkDF.head()
```

```
Row(ATENDIMENTO_ANO=2009.0, OCORRENCIA_ANO=2009, ATENDIMENTO_BAIRRO_NOME='ABRANCHES', FLAG_FLAGRANTE='NÃO',
NATUREZA1_DESCRICAO='AIFU', OCORRENCIA_DATA='2009-05-17', OCORRENCIA_DIA_SEMANA='DOMINGO', REGIONAL_FATO_NOME='BOA
VISTA', QUANTIDADE_OCORRENCIA=1)
```

```
sparkDF.describe().show()
```

```
+-------+------------------+------------------+----------------------+--------------+-------------------+---------------+
|summary|  ATENDIMENTO_ANO|    OCORRENCIA_ANO|ATENDIMENTO_BAIRRO_NOME|FLAG_FLAGRANTE|NATUREZA1_DESCRICAO|OCORRENCIA_DATA|
+-------+------------------+------------------+----------------------+--------------+-------------------+---------------+
|  count|            324463|            324463|                324463|        324463|             324463|         324463|
|   mean|2016.310155549323|2016.3100600068421|                  null|          null|               null|           null|
| stddev|4.125360146979848|  4.12533333626811|                  null|          null|               null|           null|
|    min|            2009.0|              2009|          JARDIM OSASCO|           NÃO|              AIFU|     2009-01-01|
|    max|            2022.0|              2022|            ÁGUAS BELAS|           SIM|   Órgãos acionados|     2022-10-01|
+-------+------------------+------------------+----------------------+--------------+-------------------+---------------+
```

```
sparkDF.printSchema()
```

```
root
 |-- ATENDIMENTO_ANO: double (nullable = true)
 |-- OCORRENCIA_ANO: long (nullable = true)
 |-- ATENDIMENTO_BAIRRO_NOME: string (nullable = true)
 |-- FLAG_FLAGRANTE: string (nullable = true)
 |-- NATUREZA1_DESCRICAO: string (nullable = true)
 |-- OCORRENCIA_DATA: string (nullable = true)
 |-- OCORRENCIA_DIA_SEMANA: string (nullable = true)
 |-- REGIONAL_FATO_NOME: string (nullable = true)
 |-- QUANTIDADE_OCORRENCIA: long (nullable = true)
```

- **Contagem de linhas duplicadas**

```
Contagem de linhas duplicadas

import pyspark.sql.functions as funcs
sparkDF.groupby(sparkDF.columns).count().where(funcs.col('count') >
1).select(funcs.sum('count')).show()
```

```
+----------+
|sum(count)|
+----------+
|      null|
+----------+
```

- **Buscando colunas numéricas e categóricas**

```
Achar colunas numéricas e categóricas

colunas_numericas = list()
colunas_categoricas = list()
for col_ in sparkDF.columns:
    if sparkDF.select(col_).dtypes[0][1] != "string":
        colunas_numericas.append(col_)
    else:
        colunas_categoricas.append(col_)
```

```
print("Colunas Numericas",colunas_numericas)
print("Colunas Categoricas",colunas_categoricas)
```

```
Colunas Numericas ['ATENDIMENTO_ANO', 'OCORRENCIA_ANO', 'QUANTIDADE_OCORRENCIA']
Colunas Categoricas ['ATENDIMENTO_BAIRRO_NOME', 'FLAG_FLAGRANTE', 'NATUREZA1_DESCRICAO', 'OCORRENCIA_DATA', 'OCORRENCIA_D
```

- **Contagem de valores nulos**

```
Contagem de valores nulos

from pyspark.sql.functions import *
print(sparkDF.select([count(when(isnan(c) | col(c).isNull(),
c)).alias(c) for c in sparkDF.columns]).show())
```

```
+-------------+-------------+--------------------+-------------+-----------------+--------------+--------------
|ATENDIMENTO_ANO|OCORRENCIA_ANO|ATENDIMENTO_BAIRRO_NOME|FLAG_FLAGRANTE|NATUREZA1_DESCRICAO|OCORRENCIA_DATA|OCORRENCIA_DIA
+-------------+-------------+--------------------+-------------+-----------------+--------------+--------------
|            0|            0|                   0|            0|                0|             0|
+-------------+-------------+--------------------+-------------+-----------------+--------------+--------------

None
```

- **Métricas**

```
sparkDF.summary().show()
```

```
+-------+------------------+------------------+--------------------+-------------+-----------------+--
|summary|   ATENDIMENTO_ANO|    OCORRENCIA_ANO|ATENDIMENTO_BAIRRO_NOME|FLAG_FLAGRANTE|NATUREZA1_DESCRICAO|OC
+-------+------------------+------------------+--------------------+-------------+-----------------+--
|  count|            323781|            323781|              323781|       323781|           323781|
|   mean|2016.2981706770934|2016.2980749333653|                null|         null|             null|
| stddev| 4.12142061320426| 4.121393441914671|                null|         null|             null|
|    min|            2009.0|              2009|        JARDIM OSASCO|          NÃO|             AIFU|
|    25%|            2013.0|              2013|                null|         null|             null|
|    50%|            2017.0|              2017|                null|         null|             null|
|    75%|            2020.0|              2020|                null|         null|             null|
|    max|            2022.0|              2022|          ÁGUAS BELAS|          SIM| Órgãos acionados|
+-------+------------------+------------------+--------------------+-------------+-----------------+--
```

6. Modelagem, Treinamento e Otimização

Para a etapa de modelagem foram testados diversos tipos de modelos, para verificar como os dados se comportam e qual deles tem o melhor desempenho na base de dados analisada.

A primeira etapa é a importação de todas as bibliotecas que serão utilizadas para análise e modelagem dos dados.

- **Importação das bibliotecas**

```
!pip install category_encoders pyenchant
!pip download unidecode
!pip install unidecode
!pip install python-docx
!pip install pyenchant && sudo apt-get install python-enchant
```

```
import enchant
import numpy as np
from unidecode import unidecode
import unidecode
import pandas as pd # importando o pandas para manipularmos o dataset
import seaborn as sns # importando o Seaborn para visualizar o
comportamento dos dados
import category_encoders as ce # Wellington
import matplotlib.pyplot as plt # importando o Matplotlib para o
elbow method

from category_encoders import *
from pandas_profiling import ProfileReport # importando o
pandas-profiling para fazer o profile do dataset
from scipy import stats as sp
from sklearn.model_selection import train_test_split # utilizado para
o split entre treinamento e teste
from sklearn.neighbors import KNeighborsRegressor # KNN para
regressão
from sklearn.linear_model import LinearRegression # Regressão linear
from sklearn.svm import SVR # SVM para regressão
from sklearn.decomposition import PCA # PCA como aprendizagem
não-supervisionada
from sklearn.preprocessing import RobustScaler # utilizado para que
todas as entradas estejam na mesma escala numérica
from sklearn.preprocessing import StandardScaler
from pandas.core.frame import DataFrame
```

```
from matplotlib import pyplot as plt
from sklearn import preprocessing
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import *
from sklearn.neighbors import KNeighborsClassifier

%matplotlib inline
```

A primeira etapa é a importação de todas as bibliotecas que serão utilizadas para análise e modelagem dos dados, que será realizada com a função read_csv, separando por ; e utilizando o encoding ISO-8859-1.

Após essa etapa é realizada a utilização da função info() para verificar as informações dos dados de cada colunas e posteriormente value_counts() para observar a quantidade dos valores de cada coluna. Essa etapa é realizada pois anteriormente foram refeitas novas transformações e limpezas nos dados e agora a coluna `ATENDIMENTO_BAIRRO_NOME` possui o total de 168 dados e não mais 191 como apresentados anteriormente na etapa da análise exploratória.

- **Importação dos dados**

```
df = pd.read_csv('/content/teste.csv', sep=';',
encoding='ISO-8859-1')
```

```
df
```

	ATENDIMENTO_BAIRRO_NOME	NATUREZA1_DESCRICAO	OCORRENCIA_ANO	OCORRENCIA_DIA_SEMANA	OCORRENCIA_HORA	OCORRENCIA_MES	OCORRENCIA_DIA
0	CIDADE INDUSTRIAL	Alarmes	2009	QUINTA	15:14:00	1	1
1	FAZENDINHA	Roubo	2009	QUINTA	15:22:00	1	1
2	UBERABA	Animais	2009	QUINTA	15:59:00	1	1
3	SÍTIO CERCADO	Animais	2009	QUINTA	16:13:00	1	1
4	TATUQUARA	Alarmes	2009	QUINTA	16:29:00	1	1
...	...	...	...	...	...	...	...
419066	REBOUÇAS	Apoio	2022	SÁBADO	01:15:00	10	1
419067	CAPÃO DA IMBUIA	Apoio	2022	SÁBADO	01:22:00	10	1
419068	CIDADE INDUSTRIAL	Fundada Suspeita (Abordagem)	2022	SÁBADO	00:38:00	10	1
419069	BATEL	Apoio	2022	SEXTA	22:30:00	9	30
419070	CAPÃO RASO	Apoio	2022	SÁBADO	01:42:00	10	1

419071 rows × 7 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 419071 entries, 0 to 419070
Data columns (total 7 columns):
 #   Column                  Non-Null Count   Dtype 
---  ------                  --------------   ----- 
 0   ATENDIMENTO_BAIRRO_NOME  418926 non-null  object
 1   NATUREZA1_DESCRICAO      419071 non-null  object
 2   OCORRENCIA_ANO           419071 non-null  int64 
 3   OCORRENCIA_DIA_SEMANA    419071 non-null  object
 4   OCORRENCIA_HORA          419071 non-null  object
 5   OCORRENCIA_MES           419071 non-null  int64 
 6   OCORRENCIA_DIA           419071 non-null  int64 
dtypes: int64(3), object(4)
memory usage: 22.4+ MB
```

```
df['ATENDIMENTO_BAIRRO_NOME'].value_counts()
```

```
CENTRO                72247
CIDADE INDUSTRIAL     28776
SÍTIO CERCADO         25791
CAJURU                19367
BOQUEIRÃO             18875
                      ...
JARDIM WEISSOPOLIS        1
SANTA TERESINHA           1
SÃO JORGE                 1
LOT. MARINONI             1
SÃO BENEDITO              1
Name: ATENDIMENTO_BAIRRO_NOME, Length: 191, dtype: int64
```

- **Aprendizado de Máquina**

- **WOEEncoder**

A primeira aplicação de aprendizado de máquina utilizada é o WOEEnconder, que é baseado no Teorema de Bayes e define o grau de crença em uma hipótese, após considerar evidências. A função basicamente é medir a força de todas as categorias presentes na feature, utilizando uma proporção entre classes negativas e positivas.

```
df['OCORRENCIA_FIM_SEMANA'] = df['OCORRENCIA_DIA_SEMANA'] <= 1
df['ATENDIMENTO_BAIRRO_NOME_FIXED'] =
ce.WOEEncoder().fit_transform(df['ATENDIMENTO_BAIRRO_NOME_FIXED'],
df['OCORRENCIA_FIM_SEMANA'])
```

```
df.drop('ATENDIMENTO_BAIRRO_NOME_FIXED', axis=1)
```

	ATENDIMENTO_BAIRRO_NOME	NATUREZA1_DESCRICAO	OCORRENCIA_ANO	OCORRENCIA_DIA_SEMANA	OCORRENCIA_HORA	OCORRENCIA_MES	OCORRENCIA_DIA	OCORRENCIA_FIM_SE
0	43	2	2009	5	15	1	1	
1	53	5	2009	5	15	1	1	
2	157	16	2009	5	15	1	1	
3	147	16	2009	5	16	1	1	
4	153	2	2009	5	16	1	1	
...	...	...	...	...	...	...	...	
419066	120	1	2022	0	1	10	1	
419067	37	1	2022	0	1	10	1	
419068	43	12	2022	0	0	10	1	
419069	17	1	2022	6	22	9	30	
419070	38	1	2022	0	1	10	1	

418926 rows × 8 columns

Depois dessa etapa, são realizados dois testes, um com o scaler e outro sem o scaler nos primeiros modelos testados. Sendo eles: KNeighborsRegressor, LinearRegression, SVM.

Sem o scaler

Divisão entre treino e teste

```
# split entre treinamento e teste
X_train, X_test, y_train, y_test =
```

```
train_test_split(df.drop('ATENDIMENTO_BAIRRO_NOME', axis=1), # aqui
informamos os atributos

df['ATENDIMENTO_BAIRRO_NOME'], # aqui informamos as labels e na mesma
ordem dos atributos

test_size=0.20, # informamos a porcentagem de divisão da base.
Geralmente é algo entre 20% (0.20) a 35% (0.35)

random_state=0) # aqui informamos um "seed". É um valor aleatório é
usado para que alguns algoritmos i
```

- **KNeighbors Regressor**

```
modelo_knn = KNeighborsRegressor().fit(X_train, y_train)
modelo_knn.score(X_test, y_test)
```

```
-0.12236316906201905
```

- **Linear Regression**

```
modelo_lr = LinearRegression().fit(X_train, y_train)
modelo_lr.score(X_test, y_test)
```

```
0.09663379738948774
```

- **SVM**

```
modelo_svm = SVR().fit(X_train, y_train)
modelo_svm.score(X_test, y_test)
```

Com o scaler

Divisão entre treino e teste

```
# split entre treinamento e teste
X_train, X_test, y_train, y_test =
```

```
train_test_split(RobustScaler().fit_transform(df.drop('ATENDIMENTO_BA
IRRO_NOME', axis=1)), # aqui informamos os atributos

df['ATENDIMENTO_BAIRRO_NOME'], # aqui informamos as labels e na mesma
ordem dos atributos

test_size=0.20, # informamos a porcentagem de divisão da base.
Geralmente é algo entre 20% (0.20) a 35% (0.35)

random_state=0) # aqui informamos um "seed". É um valor aleatório e
usado para que alguns algoritmos i
```

- **KNeighbors Regressor**

```
modelo_knn = KNeighborsRegressor().fit(X_train, y_train)
modelo_knn.score(X_test, y_test)
```

```
0.06822136943177604
```

- **Linear Regression**

```
modelo_lr = LinearRegression().fit(X_train, y_train)
modelo_lr.score(X_test, y_test)
```

```
0.09663379738948774
```

- **SVM**

```
modelo_svm = SVR().fit(X_train, y_train)
modelo_svm.score(X_test, y_test)
```

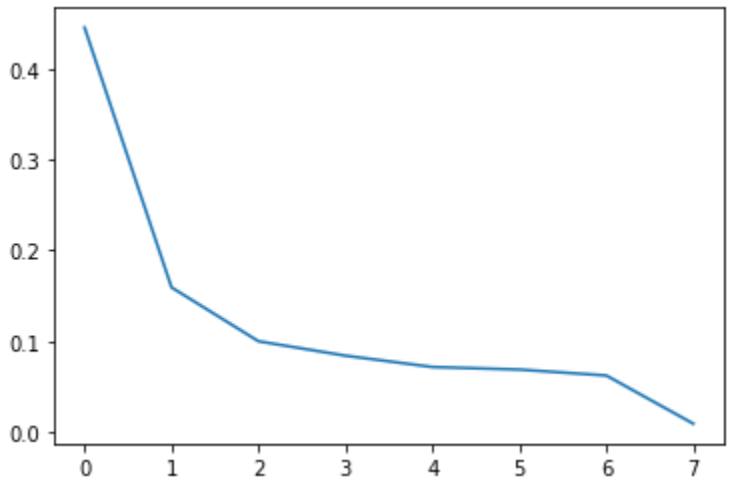
- **PCA**

Nessa etapa é realizada a tarefa de pré-processamento não supervisionada: PCA. A tarefa é realizada antes de aplicar o algoritmo e é baseada na "transformação linear ortogonal", uma técnica matemática que possibilita a projeção de atributos em um novo sistema de coordenadas.

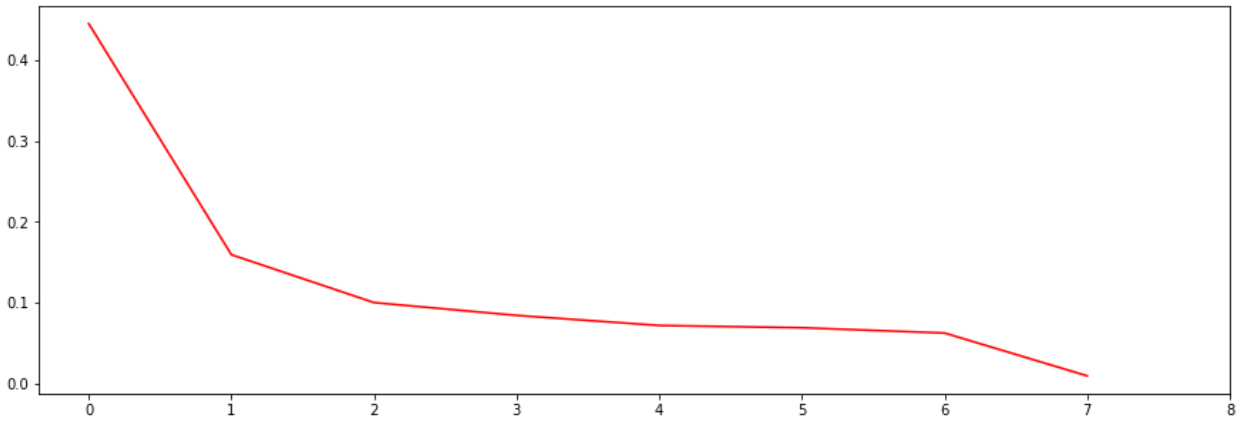
Basicamente o sistema de coordenadas coloca o primeiro atributo com maior variação como primeiro componente principal e fica localizado na primeira coordenada e todos os outros seguem o mesmo padrão.

Essa técnica possibilita a seleção dos atributos mais importantes de um conjunto de dados. O conjunto geralmente pode ser descrito por três componentes principais, que podem explicar até mais de 90% da variação.

```
pca =
PCA().fit(RobustScaler().fit_transform(df.drop('ATENDIMENTO_BAIRRO_NO
ME', axis=1)))
plt.plot(pca.explained_variance_ratio_)
```




```
plt.figure(figsize=(15, 5)) # criando um gráfico retangular para
facilitar a visualização
plt.plot(pca.explained_variance_ratio_, color='r') # colocando a
porcentagem de variância que cada componente nos trouxe
plt.xticks(np.arange(df.shape[1])) # mostrando todos os números no
eixo x
plt.show() # mostrando o gráfico final
```




```
# split entre treinamento e teste
X_train_win, X_test_win, y_train_win, y_test_win =
train_test_split(PCA(n_components=5).fit_transform(RobustScaler().fit_
transform(df.drop('ATENDIMENTO_BAIRRO_NOME', axis=1))), # aqui
informamos os atributos

df['ATENDIMENTO_BAIRRO_NOME'], # aqui informamos as labels e na mesma
ordem dos atributos

test_size=0.25, # informamos a porcentagem de divisão da base.
Geralmente é algo entre 20% (0.20) a 35% (0.35)

random_state=0) # aqui informamos um "seed". É usado um valor
aleatório
```

- **KNeighbors Regressor**

```
modelo_knn = KNeighborsRegressor().fit(X_train, y_train)
modelo_knn.score(X_test, y_test)
```

```
0.06822136943177604
```

- **Linear Regression**

```
modelo_lr = LinearRegression().fit(X_train, y_train)
```

```
modelo_lr.score(X_test, y_test)
```

```
0.09663379738948774
```

- **SVM**

```
modelo_svm = SVR().fit(X_train, y_train)
modelo_svm.score(X_test, y_test)
```

- **Mostrando as previsões**

```
df_test = pd.DataFrame(X_test)
df_test['Quality_Real'] = y_test.values
df_test['Quality_Predicao_KNN'] = modelo_knn.predict(X_test)
df_test['Quality_Predicao_Linear'] = modelo_lr.predict(X_test)
#df_test['Quality_Predicao_SVM'] = modelo_svm.predict(X_test)
df_test
```

	0	1	2	3	4	5	6	7	Quality_Real	Quality_Predicao_KNN	Quality_Predicao_Linear
0	0.000000	-0.857143	-0.50	-1.666667	-0.833333	0.400000	-1.184319	1.0	41	41.0	49.835526
1	2.545455	0.571429	0.75	-0.111111	0.000000	-0.400000	-1.184319	0.0	41	56.8	49.699750
2	2.363636	0.428571	-0.25	0.222222	-0.666667	0.400000	-1.184319	0.0	41	80.4	50.215436
3	0.181818	0.285714	0.00	-0.888889	-0.666667	-0.333333	-1.184319	0.0	41	41.0	50.742242
4	-0.818182	-0.285714	-0.50	0.777778	-0.333333	0.533333	0.000000	1.0	43	21.6	72.452248
...	...	...	...	...	...	...	...	...	...	...	...
83781	0.000000	-1.000000	0.25	-0.666667	0.666667	-0.266667	-0.389993	0.0	5	54.8	64.615779
83782	0.363636	0.285714	0.50	-0.666667	-0.833333	-0.466667	0.294471	0.0	17	51.2	77.001572
83783	-0.727273	-1.285714	-0.25	0.777778	0.833333	0.933333	-0.379551	0.0	32	102.6	65.653588
83784	0.454545	-1.000000	-0.75	0.555556	-0.333333	-0.933333	-0.412457	1.0	37	57.2	64.249554
83785	-0.818182	-0.428571	-0.50	-0.777778	0.333333	0.933333	-0.777419	1.0	48	55.6	58.070839

83786 rows × 11 columns

Na etapa seguinte são aplicados novos modelos, sendo eles: DecisionTreeRegressor, LGBMClassifier, RandomForestRegressor, XGBoost e suas respectivas métricas. E por fim, o modelo Prophet, - junto com as métricas de desempenho -, que é aplicado a base que estava sendo trabalhada com as transformações no Apache Spark.

- **DecisionTreeRegressor**

```
X_train, X_test, y_train, y_test =
train_test_split(df.drop('ATENDIMENTO_BAIRRO_NOME', axis=1),
df['ATENDIMENTO_BAIRRO_NOME'], test_size=0.2, random_state=0)
```

```
# criando o modelo
modelo_dt = DecisionTreeRegressor().fit(X_train, y_train)
modelo_dt.score(X_test, y_test)
```

```
0.9995795333375131
```

- **LGBMClassifier**

```
X_train, X_test, y_train, y_test =
train_test_split(df.drop('ATENDIMENTO_BAIRRO_NOME', axis=1),
df['ATENDIMENTO_BAIRRO_NOME'], test_size=0.2, random_state=0)
```

```
modelo_LGBM = LGBMClassifier(random_state=0).fit(X_train, y_train)
modelo_LGBM.score(X_test, y_test)
```

```
0.1960112667987492
```

```
modelo_LGBM.predict(X_test)
```

```
array([41,  5, 41, ..., 41, 28, 41])
```

- **RandomForestRegressor e XGBoost**

```
# split entre treinamento e teste
X_train, X_test, y_train, y_test =
```

```
train_test_split(df.drop('ATENDIMENTO_BAIRRO_NOME', axis=1), # aqui
informamos os atributos

df['ATENDIMENTO_BAIRRO_NOME'], # aqui informamos as labels e na mesma
ordem dos atributos

test_size=0.20, # informamos a porcentagem de divisão da base.
Geralmente é algo entre 20% (0.20) a 35% (0.35)

random_state=0) # aqui informamos um "seed". É um valor aleatório
usado para que alguns algoritmos iniciem de forma aleatória a sua
divisão.
```

- **RandomForestRegressor**

```
modelo_rfr = RandomForestRegressor(random_state=0).fit(X_train,
y_train)
predicoes_rfc = modelo_rfr.predict(X_test)
modelo_rfr.score(X_test, y_test)
```

```
0.9997674831763721
```

- **XGBoost**

```
modelo_xgb = XGBClassifier(random_state=0).fit(X_train, y_train)
predicoes_xgb = modelo_xgb.predict(X_test)
modelo_xgb.score(X_test, y_test)
```

```
0.999701620795837
```

```
resultados_classificacao = pd.DataFrame(y_test)
resultados_classificacao['Predicao_RandomForest'] = predicoes_rfc
resultados_classificacao['Predicao_XGBoost'] = predicoes_xgb

resultados_classificacao
```

	ATENDIMENTO_BAIRRO_NOME	Predicao_RandomForest	Predicao_XGBoost
69969	41	41.0	41
404424	41	41.0	41
331276	41	41.0	41
270468	41	41.0	41
171597	43	43.0	43
...	...	...	...
64754	5	5.0	5
266505	17	17.0	17
22812	32	32.0	32
53755	37	37.0	37
155643	48	48.0	48

83786 rows × 3 columns

- **ROC, AUC e F1 score com KNeighborsClassifier**

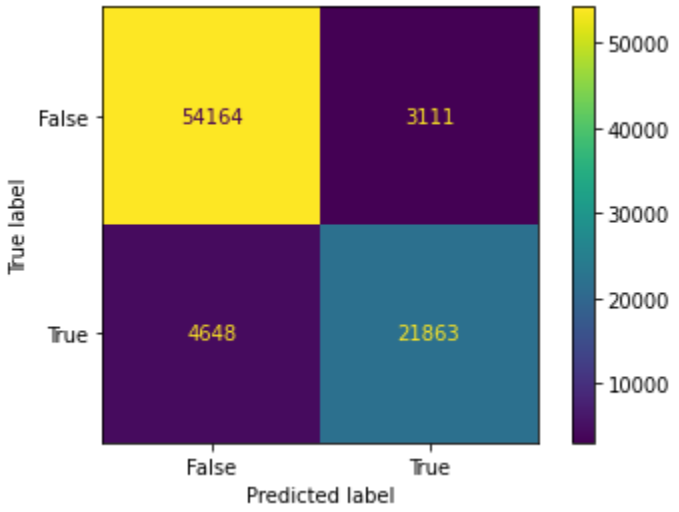
```
X_train, X_test, y_train, y_test =
train_test_split(df.drop('OCORRENCIA_FIM_SEMANA', axis=1),
df['OCORRENCIA_FIM_SEMANA'], test_size=0.2, random_state=0)
```

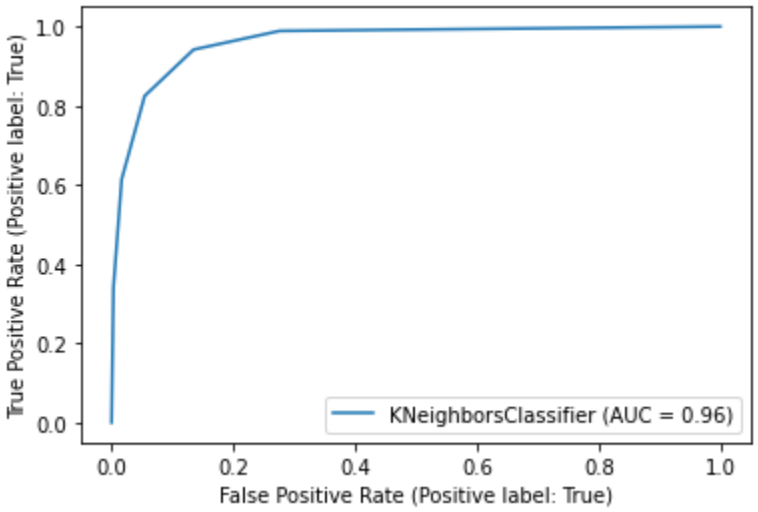
```
modelo = KNeighborsClassifier().fit(X_train, y_train)
```

```
# gerando as predições
y_pred = modelo.predict(X_test)
```

```
# matriz de confusão
plot_confusion_matrix(modelo, X_test, y_test)
plt.show()

# ROC
plot_roc_curve(modelo, X_test, y_test)
plt.show()
```

```
# AUC
display(f'AUC: {roc_auc_score(y_test, y_pred)}')

# F1 score
display(f'F1 Score: {f1_score(y_test, y_pred)}')
```





- **MAE, R2, MSE e RMSE DecisionTreeRegressor**

```
X_train, X_test, y_train, y_test =
train_test_split(df.drop('ATENDIMENTO_BAIRRO_NOME', axis=1),
df['ATENDIMENTO_BAIRRO_NOME'], test_size=0.2, random_state=0)
```

```
# criando o modelo
modelo_dt = DecisionTreeRegressor().fit(X_train, y_train)
modelo_dt.score(X_test, y_test)
```

```
0.9994747755883882
```

```
y_pred = modelo_dt.predict(X_test)
```

```
# MAE
display(f"MAE: {mean_absolute_error(y_test, y_pred)}")

# R2
display(f"R2: {r2_score(y_test, y_pred)}")

# MSE
display(f"MSE: {mean_squared_error(y_test, y_pred)}")

# RMSE
display(f"RMSE: {mean_squared_error(y_test, y_pred, squared=False)}")
```

```
'MAE: 0.016255699042799512'
'R2: 0.9994747755883882'
'MSE: 1.3666722364118111'
'RMSE: 1.1690475766245834'
```

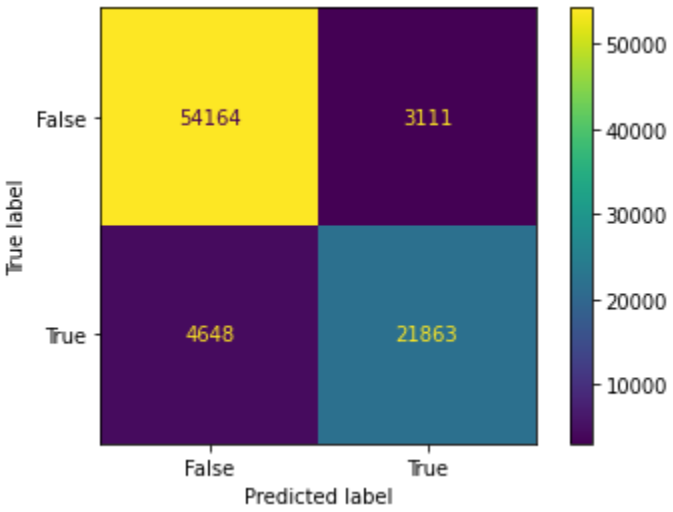
- **ROC, AUC e F1 score com LGBMClassifier**

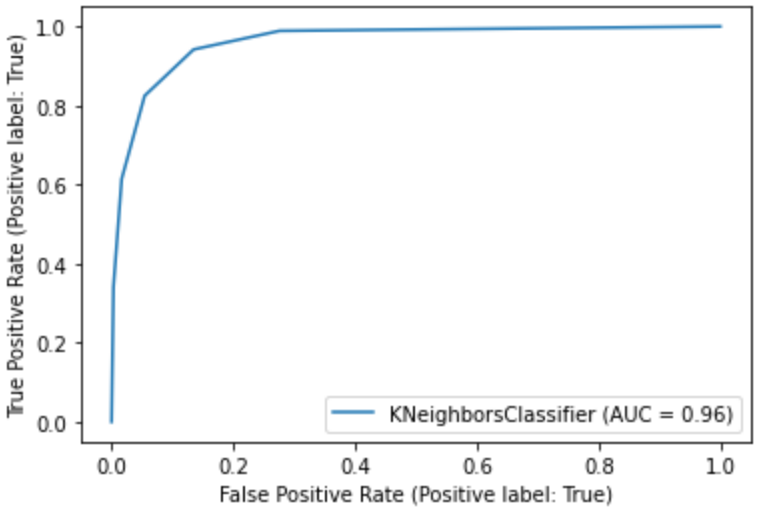
```
# split entre treinamento e teste
X_train, X_test, y_train, y_test =
train_test_split(df.drop('OCORRENCIA_FIM_SEMANA', axis=1),
df['OCORRENCIA_FIM_SEMANA'], test_size=0.2, random_state=0)
```

```
# criando o modelo
modelo_LGBMC = LGBMClassifier(random_state=0).fit(X_train, y_train)
```

```
# gerando as predições
y_pred = modelo.predict(X_test)
```

```
# matriz de confusão
plot_confusion_matrix(modelo, X_test, y_test)
plt.show()

# ROC
plot_roc_curve(modelo, X_test, y_test)
plt.show()

# AUC
display(f'AUC: {roc_auc_score(y_test, y_pred)}')

# F1 score
display(f'F1 Score: {f1_score(y_test, y_pred)}')
```





- **MSLE e MAPE**

```
# MSLE
display(f"RMSE: {mean_squared_log_error(df['OCORRENCIA_DIA'],
df['OCORRENCIA_MES'])}")

# MAPE
display(f"MAPE: {mean_absolute_percentage_error(df['OCORRENCIA_DIA'],
df['OCORRENCIA_MES'])}")
```

```
'RMSE: 1.3840747489753946'
'MAPE: 0.8562573221960479'
```

- **Prophet**

```
from pyspark.sql.functions import percent_rank
from pyspark.sql import Window

sparkDFModel = sparkDF.withColumn("rank",
percent_rank().over(Window.partitionBy().orderBy("OCORRENCIA_ANO")))
```

```
sparkDFModel
```

```
DataFrame[ATENDIMENTO_ANO: double, OCORRENCIA_ANO: bigint, ATENDIMENTO_BAIRRO_NOME: string, FLAG_FLAGRANTE: string,
NATUREZA1_DESCRICAO: string, OCORRENCIA_DATA: string, OCORRENCIA_DIA_SEMANA: string, REGIONAL_FATO_NOME: string,
QUANTIDADE_OCORRENCIA: bigint, rank: double]
```

```
train_df = sparkDFModel.where("rank <= .8").drop("rank")
train_df.show()
```

```
+---------------+-------------+---------------------+-------------+-----------------+--------------+-------------
|ATENDIMENTO_ANO|OCORRENCIA_ANO|ATENDIMENTO_BAIRRO_NOME|FLAG_FLAGRANTE|NATUREZA1_DESCRICAO|OCORRENCIA_DATA|OCORRENCIA_DIA
+---------------+-------------+---------------------+-------------+-----------------+--------------+-------------
|         2009.0|         2009|            ABRANCHES|          NÃO|             AIFU|    2009-05-17|
|         2009.0|         2009|            ABRANCHES|          NÃO|             AIFU|    2009-05-28|
|         2009.0|         2009|            ABRANCHES|          NÃO|             AIFU|    2009-07-23|
|         2009.0|         2009|            ABRANCHES|          NÃO|       Alagamento|    2009-11-16|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Alarmes|    2009-09-03|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Alarmes|    2009-09-27|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Alarmes|    2009-11-07|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-01-04|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-01-07|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-02-06|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-02-16|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-04-07|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-04-23|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-05-11|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-06-27|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-07-03|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-07-22|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-08-04|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-08-27|
|         2009.0|         2009|            ABRANCHES|          NÃO|          Animais|    2009-09-20|
+---------------+-------------+---------------------+-------------+-----------------+--------------+-------------
only showing top 20 rows
```

```
test_df = sparkDFModel.where("rank > .8").drop("rank")
test_df.show()
```

```
+---------------+-------------+---------------------+-------------+--------------------+--------------+-----------
|ATENDIMENTO_ANO|OCORRENCIA_ANO|ATENDIMENTO_BAIRRO_NOME|FLAG_FLAGRANTE| NATUREZA1_DESCRICAO|OCORRENCIA_DATA|OCORRENCIA_D
+---------------+-------------+---------------------+-------------+--------------------+--------------+-----------
|         2022.0|         2022|    JARDIM PEDRO DEMETE|          NÃO|               Apoio|    2022-07-01|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-02-10|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-03-05|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-04-09|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-05-27|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-07-15|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-08-17|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-08-27|
|         2022.0|         2022|            ABRANCHES|          NÃO|Agressão física/v...|    2022-09-06|
|         2022.0|         2022|            ABRANCHES|          NÃO|             Alarmes|    2022-02-09|
|         2022.0|         2022|            ABRANCHES|          NÃO|             Alarmes|    2022-03-02|
|         2022.0|         2022|            ABRANCHES|          NÃO|             Alarmes|    2022-04-16|
|         2022.0|         2022|            ABRANCHES|          NÃO|              Ameaça|    2022-03-12|
|         2022.0|         2022|            ABRANCHES|          NÃO|              Ameaça|    2022-06-18|
|         2022.0|         2022|            ABRANCHES|          NÃO|             Animais|    2022-03-30|
|         2022.0|         2022|            ABRANCHES|          NÃO|             Animais|    2022-05-04|
|         2022.0|         2022|            ABRANCHES|          NÃO|               Apoio|    2022-01-13|
|         2022.0|         2022|            ABRANCHES|          NÃO|               Apoio|    2022-01-15|
|         2022.0|         2022|            ABRANCHES|          NÃO|               Apoio|    2022-01-20|
|         2022.0|         2022|            ABRANCHES|          NÃO|               Apoio|    2022-01-21|
+---------------+-------------+---------------------+-------------+--------------------+--------------+-----------
only showing top 20 rows
```

```
train_df =
train_df.groupBy('OCORRENCIA_DATA').sum('QUANTIDADE_OCORRENCIA')
```

```
train_df = train_df.withColumnRenamed("OCORRENCIA_DATA","ds") \
   .withColumnRenamed("sum(QUANTIDADE_OCORRENCIA)","y")
```

```
train_df.show()
```

```
+----------+---+
|        ds|  y|
+----------+---+
|2009-05-17| 87|
|2009-05-28| 55|
|2009-07-23| 50|
|2009-11-16| 41|
|2009-09-03| 54|
|2009-09-27|107|
|2009-11-07| 96|
|2009-01-04| 93|
|2009-01-07| 70|
|2009-02-06| 74|
|2009-02-16| 53|
|2009-04-07| 54|
|2009-04-23| 60|
|2009-05-11| 57|
|2009-06-27| 86|
|2009-07-03| 64|
|2009-07-22| 49|
|2009-08-04| 63|
|2009-08-27| 49|
|2009-09-20|112|
+----------+---+
only showing top 20 rows
```

```
train_df = train_df.toPandas()
```

```
train_df
```

	ds	y
0	2009-05-17	87
1	2009-05-28	55
2	2009-07-23	50
3	2009-11-16	41
4	2009-09-03	54
...	...	...
4743	2021-12-04	148
4744	2021-12-07	145
4745	2021-12-03	170
4746	2021-12-09	152
4747	2021-12-02	129

4748 rows × 2 columns

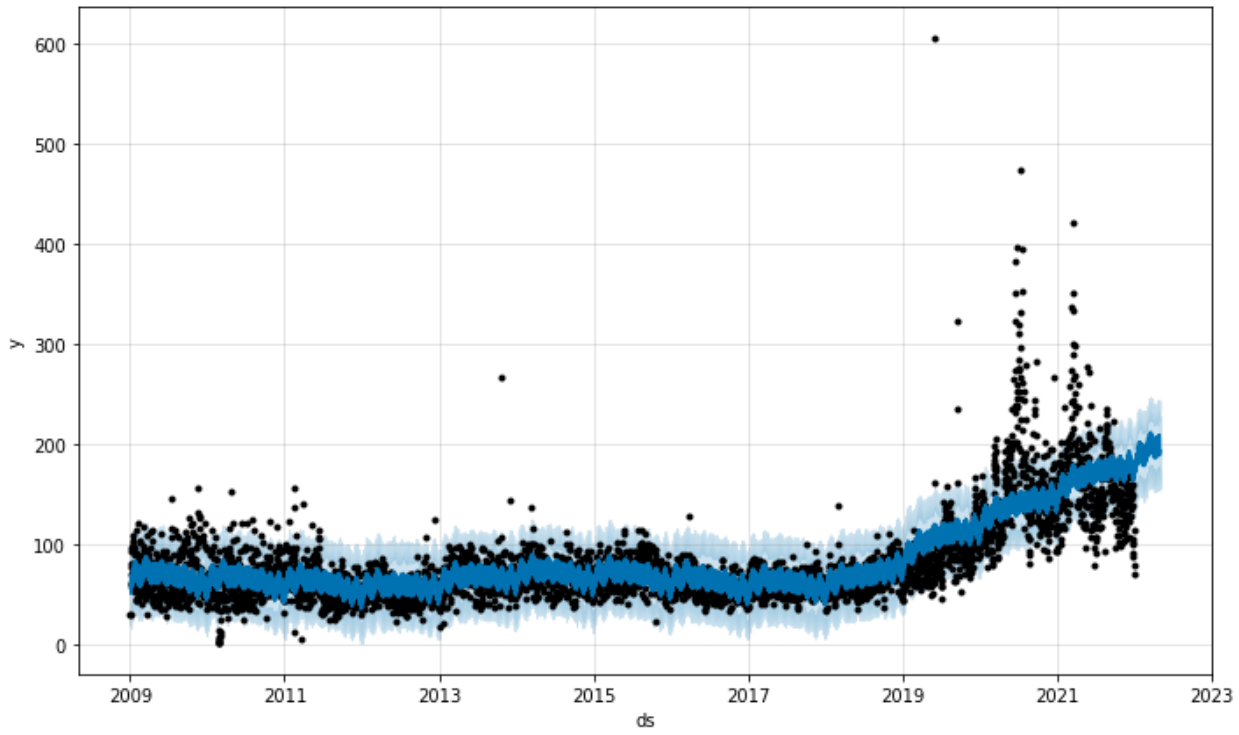
```
from prophet import Prophet
```

```
m = Prophet()
m = Prophet(yearly_seasonality = True, daily_seasonality=True)
m.fit(train_df)
m.yearly_seasonality
m.daily_seasonality
```

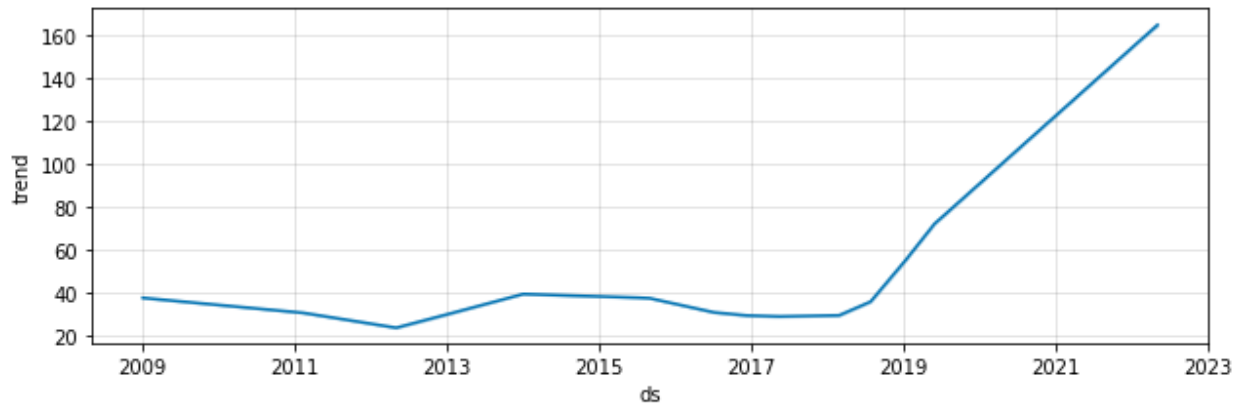
```
future = m.make_future_dataframe(periods=124) #Prevê os próximos 90 dias para comparar com a realidade

forecast = m.predict(future) #Prevê
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail() #Mostra as predições (yhat) e intervalos de confiança: máximo (yhat_upper) e mínimo (yhat_lower)
```

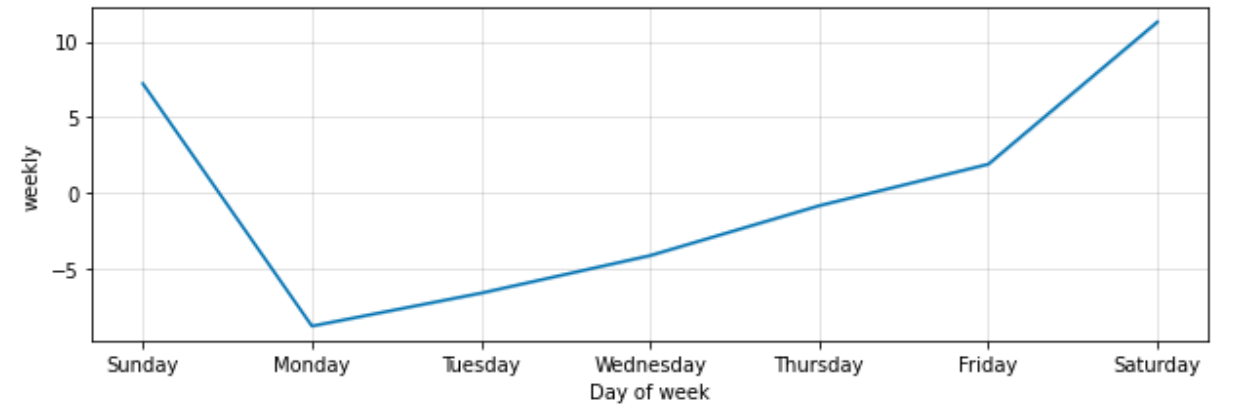
	ds	yhat	yhat_lower	yhat_upper
4867	2022-04-30	208.968225	173.412760	243.366437
4868	2022-05-01	204.863905	171.965964	239.563848
4869	2022-05-02	188.767131	155.346117	223.773896
4870	2022-05-03	190.888240	159.396108	224.613066
4871	2022-05-04	193.307337	159.020746	227.233597

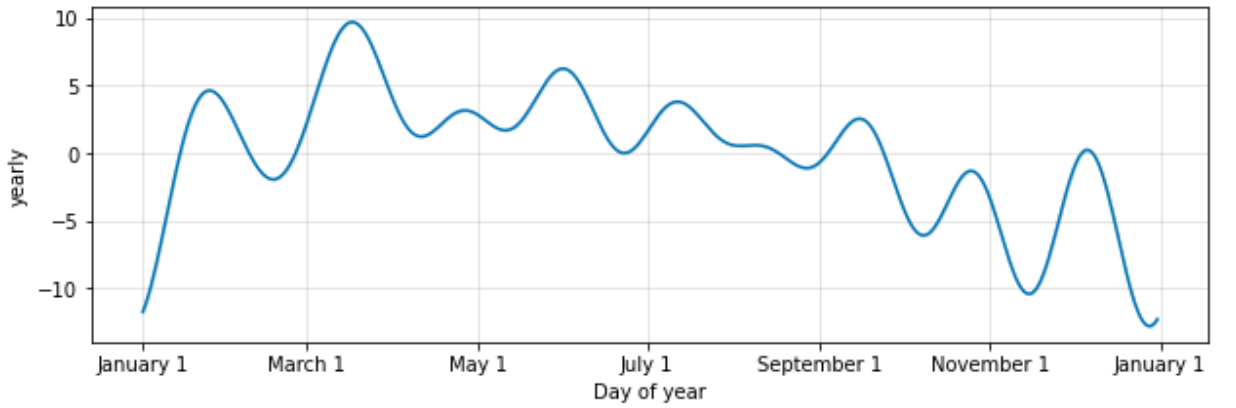
```
fig = m.plot(forecast)
```

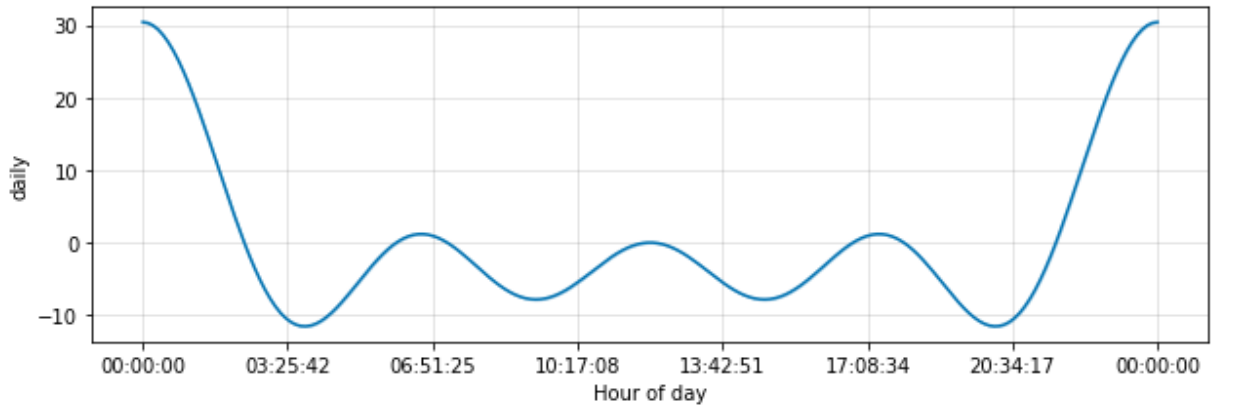



```
fig2 = m.plot_components(forecast)
```


trend
160
140
120
100
80
60
40
20
2009
2011
2013
2015
2017
2019
2021
2023
ds

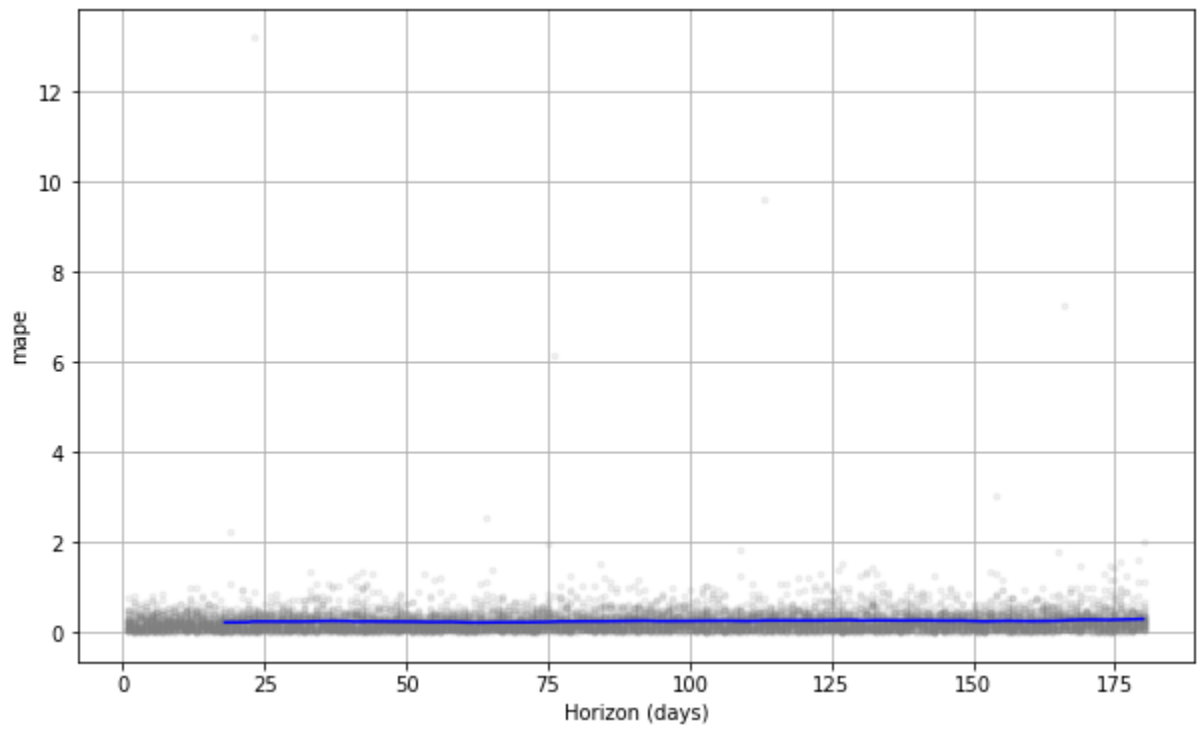

weekly
10
5
0
−5
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
10
5
0
−5
−10
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
30
20
10
0
−10
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day

```
from prophet.diagnostics import cross_validation

#criamos o dataframe de validação
dfCrossValidation = cross_validation(m, horizon='180 days')
```

```
from prophet.plot import plot_cross_validation_metric
from prophet.diagnostics import performance_metrics

# extrai métricas de desempenho
dfPerformance = performance_metrics(dfCrossValidation)

# exibe métricas de desempenho em um gráfico
fig = plot_cross_validation_metric(dfCrossValidation, metric='mape')
```




```
dfPerformance
```

	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	18 days	809.770754	28.456471	17.361695	0.208333	0.173176	0.204159	0.745679
1	19 days	780.258640	27.933110	17.217533	0.211068	0.173176	0.205127	0.741975
2	20 days	651.117638	25.517007	16.853655	0.209461	0.171963	0.203969	0.746914
3	21 days	619.291761	24.885573	16.814322	0.212357	0.176917	0.205509	0.741975
4	22 days	625.482016	25.009638	16.933972	0.213101	0.176917	0.206282	0.734568
...	...	...	...	...	...	...	...	...
158	176 days	1557.714262	39.467889	23.984745	0.273284	0.214027	0.264168	0.612346
159	177 days	1570.984395	39.635646	24.301819	0.276842	0.217701	0.267955	0.606173
160	178 days	1533.543369	39.160482	24.216031	0.277067	0.220457	0.267751	0.604938
161	179 days	1475.142364	38.407582	24.309637	0.281272	0.223604	0.271480	0.600000
162	180 days	1410.120298	37.551569	24.210060	0.283598	0.225037	0.272226	0.597531

163 rows × 8 columns

7. Conclusão

A ideia inicial do trabalho era contribuir na tomada de decisões para investimento em uma atuação com fins sociais. Para isso, a escolha do dataset norteou-se pela contribuição social e portanto, foi escolhido dataset SiGesGuarda da Prefeitura Municipal de Curitiba no Estado do Paraná no Brasil, em busca de encontrar possibilidades e melhorias na atuação social, também entender e responder perguntas relacionadas a segurança e prever e classificar possíveis novos casos de ocorrências.

Sendo assim, o primeiro objetivo foi entrevistar, pesquisar, explorar e analisar o dataset, buscando responder perguntas, entender e auxiliar a área de segurança, por meio de técnicas de Análise de Dados, Estatística e Frameworks de Big Data, como o Apache Spark (PySpark).

Buscando desenvolver e responder ao primeiro objetivo, o projeto respondeu diversas perguntas que buscam auxiliar a atuação social, como: Quantidade de ocorrências atendidas por ano, quantidade de ocorrências recebidas por ano, quantidade de ocorrências com flagrante, quantidade de ocorrências por tipo, quantidade de ocorrências por dia e também a quantidade de ocorrências por dia da semana.

E outras como: ano que possui mais atendimento, bairro que possui mais atendimento, ano por bairro que possui mais atendimento, o tipo de natureza da ocorrência, o tipo de natureza da ocorrência em cada bairro, o horário da ocorrência mais frequente, horário da semana e bairro mais frequentes.

Outro objetivo era buscar por modelos de machine learning para tentar prever e classificar as ocorrências e para isso foram utilizados os modelos: KNeighborsRegressor, LinearRegression, SVM, DecisionTreeRegressor, LGBMClassifier, RandomForestRegressor, XGBoost e Prophet. E para otimização WOEEnconder e PCA.

Neste objetivo, analisando os primeiros testes com e sem scaler com os modelos KNeighborsRegressor, LinearRegression, SVM, todos apresentaram um melhor desempenho utilizando o scaler e o que se mostrou melhor foi o LinearRegression. Ao utilizar o PCA, os modelos não demonstraram uma grande

diferença no desempenho, apresentando quase os mesmos resultados em todos os casos.

Com relação ao DecisionTreeRegressor, LGBMClassifier, RandomForestRegressor, XGBoost e Prophet, o LGBMClassifier e Prophet foram os que tiveram o menor score.

Quando avaliamos outras métricas, como ROC, AUC e F1 para o KNeighborsClassifier e para o LGBMClassifier, ambos obtiveram o mesmo score.

Ao observar MAE, R2, MSE, RMSE e MAPE para o DecisionTreeRegressor, todas tiveram um valor baixo e próximo a 1, demonstrando um bom desempenho do modelo.

8. Referências

AHMED, Asif. **PySpark in Google Colab.** Medium: 2019. Disponível em: <https://towardsdatascience.com/pyspark-in-google-colab-6821c2faf41c#:~:text=To%20run%20spark%20in%20Colab,Jupyter%20Notebook%20of%20the%20Colab.&text=Our%20Colab%20is%20ready%20to%20run%20PySpark>. Acesso em: 20/10/2022.

BARAD, Vishal. **Data Preprocessing Using Pyspark (Part:1).** Medium: 2021. Disponível em: <https://medium.com/vedity/python-data-preprocessing-using-pyspark-cc3f709c3c23>. Acesso em: 25/10/2022.

CETAX. **Tutorial Pyspark e MLlib.** Brasil: 2022. Disponível em: <https://cetax.com.br/tutorial-pyspark-e-mllib/>. Acesso em: 28/10/2022.

Data Science. **Tutorial PySpark.** Sem data. Disponível em: <https://datascience.eu/pt/programacao/tutorial-pyspark/>. Acesso em: 20/10/2022.

LOPES, Alexandre. **Funcionamento do PySpark.** Medium: 2020. Disponível em <https://medium.com/data-hackers/entendo-funcionamento-do-pyspark-2b5ab4321ab9>. Acesso em: 25/10/2022.

Microsoft. **O que é o Apache Spark?** Learn: 2022. Disponível em <https://learn.microsoft.com/pt-br/dotnet/spark/what-is-spark>. Acesso em: 15/10/2022.

Prefeitura Municipal de Curitiba. **Dados Abertos.** Sem data. Disponível em: <https://www.curitiba.pr.gov.br/dadosabertos/busca/?termo=sigesguarda>. Acesso em: 05/10/2022.

RAONIAR, RAHUL. **Introduction to PySpark.** Medium: 2021. Disponível em: <https://medium.com/the-researchers-guide/introduction-to-pyspark-a61f7217398e>. Acesso em: 15/10/2022.

ROSEN, Josh. **PySpark Internals.** Wiki: 2016. Disponível em: <https://cwiki.apache.org/confluence/display/SPARK/PySpark+Internals#:~:text=PySpark%20is%20built%20on%20top,JVM%20and%20create%20a%20JavaSparkContext.&text=RDD%20transformations%20in%20Python%20are,on%20PythonRDD%20objects%20in%20Java>. Acesso em: 15/10/2022.

SHAFIQUE, Ayesha. **Exploratory Data Analysis using Pyspark Dataframe in Python.** Medium: 2019. Disponível em: <https://medium.com/@aieeshashafique/exploratory-data-analysis-using-pyspark-dataframe-in-python-bd55c02a2852>. Acesso em: 18/10/2022.

Spark By Examples. **How to Convert Pandas to PySpark DataFrame.** 2022. Disponível em:

<https://sparkbyexamples.com/pyspark/convert-pandas-to-pyspark-dataframe/>. Acesso em: 10/10/2022.

Spark By Examples. **PySpark Union and UnionAll Explained.** 2022. Disponível em: <https://sparkbyexamples.com/pyspark/pyspark-union-and-unionall/>. Acesso em: 10/10/2022.

SÔTO, Rubens. **Encapsulando Transformações em DataFrames com Pyspark.** Medium: 2020. Disponível em: <https://medium.com/data-hackers/encapsulando-transformações-em-dataframes-com-pyspark-9543e0cc69f1>. Acesso em: 23/10/2022.