

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

**Arthur de Moura Mota**

**ANÁLISE DO IMPACTO DE OUTLIERS EM MODELO DE REGRESSÃO PARA  
PREVISÃO DO VALOR MÁXIMO DE DESPESAS DE CAMPANHA DE  
CANDIDATOS ELEITOS NAS ELEIÇÕES MUNICIPAIS BRASILEIRAS DE 2020  
COM BASE NO PIB E PORTE POPULACIONAL**

Belo Horizonte

2023

**Arthur de Moura Mota**

**ANÁLISE DO IMPACTO DE OUTLIERS EM MODELO DE REGRESSÃO PARA  
PREVISÃO DO VALOR MÁXIMO DE DESPESAS DE CAMPANHA DE  
CANDIDATOS ELEITOS NAS ELEIÇÕES MUNICIPAIS BRASILEIRAS DE 2020  
COM BASE NO PIB E PORTE POPULACIONAL**

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

Belo Horizonte  
2023

## Sumário

Sumário .....	3
1 Introdução .....	5
1.1 Contextualização .....	5
1.2 O problema proposto e Objetivos .....	6
2 Coleta de Dados .....	7
2.1 Candidatos - 2020 .....	7
2.2 Produto Interno Bruto dos Municípios - Base 2010-2020 .....	10
2.3 Estimativas da População 2020 .....	12
2.4 Municípios Brasileiros TSE .....	13
3 Processamento/Tratamento de Dados .....	14
3.1 Tecnologias e Bibliotecas .....	14
3.2 Leitura dos dados de Municípios Brasileiros TSE .....	14
3.3 Leitura dos dados Candidatos – 2020 .....	15
3.3.1 Filtragem de dados Candidatos – 2020 .....	17
3.4 Leitura dos dados Estimativas da População 2020 .....	21
3.4.1 Tratamento dos Dados de Estimativas da População 2020 .....	23
3.5 Leitura dos Dados Produto Interno Bruto dos Municípios - Base 2010-2020 .....	27
3.5.1 Filtragem de Dados Produto Interno Bruto dos Municípios - Base 2010-2020 ..	30
3.6 Junção de dados de População e PIB .....	35
3.7 Junção de Candidatos – 2020 (Eleitos) e Municípios Brasileiros TSE .....	41
3.8 Junção de Dados dos Eleitos com Dados Demográficos .....	44
3.8.1 Conversão de Tipos de Dados e Reordenação de Colunas .....	44
4 Análise e Exploração dos Dados .....	49
4.1 Separação dos Dados em DataFrames Diferentes .....	49
4.2 Exibição do Tamanho e Informações dos DataFrames .....	49
4.3 Análise de Variáveis Categóricas .....	50
4.4 Análise de Variáveis Numéricas .....	56
4.4.1 Plotagem de Matrizes de Correlação .....	60
4.5 Plotagem de Boxplots .....	64
4.6 Estatísticas Descritivas da Coluna VR_DESPESA_MAX_CAMPANHA .....	65

5	Criação de Modelos de Machine Learning .....	67
5.1	Gradient Boosting Regressor.....	67
5.2	analise_de_modelo_VR_DESPESA_MAX_CAMPANHA.....	68
5.3	calcular_melhores_parametros .....	69
5.4	remover_outliers_VR_DESPESA_MAX_CAMPANHA.....	70
5.5	Análise do modelo de regressão .....	71
6	Interpretação e Apresentação dos Resultados .....	73
6.1	Saídas durante Criação de Modelo de Machine Learning .....	73
6.2	Análise do modelo de regressão e Outliers - df_eleitos_dados_demograficos .....	75
6.3	Análise do modelo de regressão e Outliers - df_eleitos_prefeitos .....	76
6.3.1	Interpretação dos Resultados: .....	76
6.3.2	Conclusão:.....	77
6.4	Análise do modelo de regressão e Outliers - df_eleitos_vereadores .....	77
6.4.1	Interpretação dos Resultados .....	78
6.4.2	Conclusão .....	78
6.5	Análise dos Outliers .....	79
6.5.1	Análise da Relação entre Reeleição e Outliers.....	79
6.5.2	Distribuição dos Eleitos por Cargo e Agremiação .....	80
6.5.3	Disparidade nas Despesas de Campanha: Capitais vs. Não Capitais.....	81
6.5.4	Relação Entre Média de Despesas de Campanha, PIB e UF.....	83
6.5.5	Variação das Despesas de Campanha e População Estimada por UF .....	84
6.5.6	Variação nas Despesas de Campanha de Candidatos Eleitos por Estado e Região	
6.5.7	85 Variação nas Médias de Gastos de Campanha por Partido .....	86
6.6	Considerações Finais.....	88
7	Links .....	89
	APÊNDICE .....	90

## 1 Introdução

### 1.1 Contextualização

Em 2020, o cenário das eleições municipais no Brasil foi profundamente impactado por uma conjuntura desafiadora, com a pandemia da COVID-19 impondo tanto desafios à saúde pública quanto abalos à economia. Os candidatos eleitos nesse pleito enfrentaram uma pressão sem precedentes, em que a gestão eficiente de recursos financeiros se tornou crucial para o sucesso das campanhas.

A análise do valor das campanhas políticas dos candidatos eleitos nas eleições municipais de 2020 no Brasil adquire uma relevância inegável. Ela nos permite adentrar na intrincada dinâmica eleitoral do país e compreender de que forma os recursos financeiros moldaram os desfechos eleitorais em diferentes municípios. Além disso, essa análise se desdobra em uma investigação profunda da interconexão entre esses dados financeiros e dois pilares fundamentais: o Produto Interno Bruto (PIB) municipal e o porte populacional dos municípios.

#### **Valor das Campanhas Políticas:**

No ano de 2020, o Brasil testemunhou um marco nas eleições municipais, caracterizado por um notável aumento nos gastos de campanha em relação a pleitos anteriores. Esse fenômeno foi motivado, em parte, pela proibição das doações empresariais em 2015, que impeliu os candidatos a dependerem mais fortemente de recursos próprios e de apoio financeiro obtido por meio de financiamento coletivo (crowdfunding).

#### **PIB Municipal:**

O PIB municipal, um indicador crítico, espelha a atividade econômica de cada localidade. Em 2020, muitos municípios brasileiros enfrentaram desafios econômicos iminentes devido aos impactos da pandemia de COVID-19, refletindo-se diretamente no desempenho do PIB de diversas regiões. Além disso, é imprescindível destacar a profunda desigualdade econômica que permeia o Brasil, com municípios de variados tamanhos e níveis de desenvolvimento.

#### **Porte Populacional de Município:**

O porte de um município está intrinsecamente ligado a seu tamanho populacional e à sua importância econômica. No contexto brasileiro, os municípios podem ser categorizados como pequenos, médios e grandes, baseados em critérios populacionais e de PIB. Essa classificação exerce uma influência direta na quantidade de recursos disponíveis para as campanhas políticas, bem como na complexidade das estratégias eleitorais adotadas pelos candidatos.

Em síntese, a relação entre o montante investido nas campanhas políticas, o Produto Interno Bruto municipal e o porte populacional de cada município revela-se como um enigma multifacetado e complexo. Aprofundar essa contextualização é imperativo para uma compreensão holística de como os fatores econômicos e demográficos moldaram as eleições municipais de 2020 no Brasil, influenciando resultados alcançados pelos candidatos eleitos e lançando luz sobre a dinâmica política do país.

## 1.2 O Problema Proposto e Objetivos

Este projeto é crucial devido ao cenário desafiador das eleições municipais brasileiras de 2020, que aconteceram em meio à pandemia de COVID-19. Entender como o financiamento das campanhas políticas afetou os resultados eleitorais é essencial para analisar as dinâmicas eleitorais.

Este trabalho irá analisar dados das eleições municipais de 2020 no Brasil, abrangendo candidatos eleitos em diferentes municípios. Nossas fontes principais de dados incluem informações eleitorais, econômicas, como o Produto Interno Bruto (PIB) municipal, e dados populacionais.

O objetivo principal é investigar como fatores econômicos (PIB municipal) e demográficos (porte populacional dos municípios) influenciaram estratégias de campanha e resultados eleitorais em 2020. Concentraremos nossa análise na identificação e análise de outliers (valores atípicos nos) gastos de campanha e em entender como esses valores atípicos afetaram os resultados das eleições.

## 2 Coleta de Dados

Para atingir o objetivo, coletou-se uma variedade de dados pertinentes a partir de fontes confiáveis. Nossa principal motivação é entender como fatores econômicos, representados pelo Produto Interno Bruto (PIB) municipal, e fatores demográficos, expressos pelo porte populacional dos municípios, influenciaram as estratégias de campanha e, consequentemente, os resultados eleitorais de 2020.

Esta seção irá focar nas informações detalhadas sobre a origem e a estrutura dos dados utilizados em nossa análise. Exploraremos as fontes de dados, o formato dos conjuntos de dados, e estabeleceremos conexões entre eles para uma compreensão mais clara do nosso processo de coleta e análise conforme a seguir.

### 2.1 Candidatos - 2020

Os dados do dataset "Candidatos - 2020" foram obtidos no site oficial do Tribunal Superior Eleitoral (TSE) nos seguintes links: <https://dadosabertos.tse.jus.br/dataset/candidatos-2020-subtemas/resource/8187b1aa-5026-4908-a15a-0bf777ee6701>

Link para o recurso direto do arquivo CSV:

[https://cdn.tse.jus.br/estatistica/sead/odsele/consulta\\_cand/consulta\\_cand\\_2020.zip](https://cdn.tse.jus.br/estatistica/sead/odsele/consulta_cand/consulta_cand_2020.zip)

A coleta desses dados foi realizada em 18/08/2023. O arquivo CSV referente aos candidatos chama-se "consulta\_cand\_2020\_BRASIL.csv".

Aqui está a tabela com a descrição de cada campo/coluna do dataset "Candidatos – 2020":

Nome da Coluna/Campo	Descrição	Tipo
DT_GERACAO	Data da extração dos dados para geração do arquivo	Datetime
HH_GERACAO	Hora da extração dos dados para geração do arquivo no horário de Brasília.	Datetime
ANO_ELEICAO	Ano de referência da eleição para geração do arquivo.	Integer
CD_TIPO_ELEICAO	Código do tipo de eleição. Pode assumir os valores: 1 - Eleição Suplementar, 2 - Eleição Ordinária e 3 - Consulta Popular.	Integer
NM_TIPO_ELEICAO	Nome do tipo de eleição.	String
NR_TURNO	Número do turno da eleição.	Integer
CD_ELEICAO	Código único da eleição no âmbito da Justiça Eleitoral.	Integer
DS_ELEICAO	Descrição da eleição.	String
DT_ELEICAO	Data em que ocorreu a eleição.	Date
TP_ABRANGENCIA	Abrangência da eleição. Pode assumir os valores: Municipal, Estadual e Federal.	String
SG_UF	Sigla da Unidade da Federação em que ocorreu a eleição.	String
SG_UE	Sigla da Unidade Eleitoral em que a candidata ou o candidato concorre na eleição.	String
NM_UE	Nome da Unidade Eleitoral.	String
CD_CARGO	Código do cargo da candidata ou candidato.	Integer

DS_CARGO	Descrição do cargo da candidata ou candidato.	String
SQ_CANDIDATO	Número sequencial da candidata ou candidato, gerado internamente pelos sistemas eleitorais para cada eleição.	Integer
NR_CANDIDATO	Número da candidata ou candidato na urna.	Integer
NM_CANDIDATO	Nome completo da candidata ou candidato.	String
NM_URNA_CANDIDATO	Nome da candidata ou candidato que aparece na urna.	String
NM_SOCIAL_CANDIDATO	Nome social da candidata ou candidato.	String
NR CPF_CANDIDATO	Número do CPF da candidata ou candidato.	String
NM_EMAIL	Endereço de e-mail da candidata ou candidato.	String
CD_SITUACAO_CANDIDATURA	Código da situação do registro de candidatura da candidata ou candidato.	Integer
DS_SITUACAO_CANDIDATURA	Descrição da situação do registro da candidatura da candidata ou candidato.	String
CD_DETALHE_SITUA-CAO_CAND	Código do detalhe da situação do registro da candidatura da candidata ou candidato.	Integer
DS_DETALHE_SITUA-CAO_CAND	Descrição do detalhe da situação do registro de candidatura da candidata ou candidato.	String
TP AGREMIACAO	Tipo de agremiação da candidatura da candidata ou candidato.	String
NR_PARTIDO	Número do partido de origem da candidata ou candidato.	Integer
SG_PARTIDO	Sigla do partido de origem da candidata ou candidato.	String
NM_PARTIDO	Nome do partido de origem da candidata ou candidato.	String
NR_FEDERACAO	Número da Federação a qual concorre a candidata ou candidato. Caso a candidatura não esteja em uma Federação, o arquivo retornará o valor '#NULL#'.	Integer
NM_FEDERACAO	Nome da Federação a qual concorre a candidata ou candidato. Caso a candidatura não esteja em uma Federação, o arquivo retornará o valor '#NULL#'.	String
SG_FEDERACAO	Sigla da Federação a qual concorre a candidata ou candidato. Caso a candidatura não esteja em uma Federação, o arquivo retornará o valor '#NULL#'.	String
DS_COMPOSICAO_FEDERA-CAO	Lista dos partidos que compõem a federação a qual o partido da candidata ou candidato está associado. Nesta lista, os partidos estão concatenados e separados por /. Caso o partido da candidata ou candidato não esteja associado a uma federação, a coluna virá com o valor '#NULL#'.	String
SQ_COLIGACAO	Sequencial da coligação da qual a candidata ou candidato pertence, gerado pela Justiça Eleitoral.	Integer
NM_COLIGACAO	Nome da coligação da qual a candidata ou candidato pertence.	String
DS_COMPOSICAO_COLIGA-CAO	Composição da coligação da qual a candidata ou candidato pertence. Observação: Coligação é a união de dois ou mais partidos a fim de disputarem eleições. A informação da coligação no arquivo está composta pela concatenação das siglas dos partidos intercaladas com o símbolo /.	String
CD_NACIONALIDADE	Nacionalidade da candidata ou candidato.	Integer
DS_NACIONALIDADE	Nacionalidade da candidata ou candidato.	String
SG_UF_NASCIMENTO	Sigla da Unidade da Federação de nascimento da candidata ou candidato.	String
CD_MUNICIPIO_NASCIMENTO	Código de identificação do município de nascimento da candidata ou candidato.	Integer
NM_MUNICIPIO_NASCI-MENTO	Nome do município de nascimento da candidata ou candidato.	String
DT_NASCIMENTO	Data de nascimento da candidata ou candidato.	Date

NR_IDADE_DATA_POSSE	Idade da candidata ou candidato na data da posse. A idade é calculada com base na data da posse da referida candidata ou candidato para o cargo e unidade eleitoral constantes no arquivo de vagas.	Integer
NR_TITULO_ELEITORAL_CANDIDATO	Número do título eleitoral da candidata ou candidato.	String
CD_GENERO	Código do gênero da candidata ou candidato.	Integer
DS_GENERO	Gênero da candidata ou candidato.	String
CD_GRAU_INSTRUCAO	Código do grau de instrução da candidata ou candidato.	Integer
DS_GRAU_INSTRUCAO	Grau de instrução da candidata ou candidato.	String
CD_ESTADO_CIVIL	Código do estado civil da candidata ou candidato.	Integer
DS_ESTADO_CIVIL	Estado civil da candidata ou candidato.	String
CD_COR_RACA	Código da cor/raça da candidata ou candidato (autodeclaração).	Integer
DS_COR_RACA	Cor/raça da candidata ou candidato (autodeclaração).	String
CD_OCUPACAO	Código da ocupação da candidata ou candidato.	Integer
DS_OCUPACAO	Ocupação da candidata ou candidato.	String
VR_DESPESA_MAX_CAMPANHA	Valor máximo, em Reais, de despesas de campanha declarada pelo partido para aquela candidata ou candidato.	Decimal
CD_SIT_TOT_TURNO	Código da situação de totalização da candidata ou candidato naquele turno.	Integer
DS_SIT_TOT_TURNO	Descrição da situação de totalização da candidata ou candidato naquele turno.	String
ST_REELEICAO	Indica se a candidata ou candidato está concorrendo ou não à reeleição. Pode assumir os valores: S - Sim e N - Não. Informação autodeclarada pela candidata ou candidato. Reeleição é a renovação do mandato para o mesmo cargo eletivo, por mais um período, na mesma circunscrição eleitoral na qual a representante ou o representante, no pleito imediatamente anterior, se elegeu. Pelo sistema eleitoral brasileiro, o presidente da República, os governadores de estado e os prefeitos podem ser reeleitos para um único período subsequente, o que se aplica também ao vice-presidente da República, aos vice-governadores e aos vice-prefeitos. Já os parlamentares (senadores, deputados federais e estaduais/distritais e vereadores) podem se reeleger ilimitadas vezes. A possibilidade da reeleição compreende algumas regras mais específicas detalhadas no sistema eleitoral brasileiro.	String
ST_DECLARAR_BENS	Indica se a candidata ou candidato tem ou não bens a declarar. Pode assumir os valores: S - Sim e N - Não. Esta informação é fornecida pela própria candidata ou candidato no momento do pedido da candidatura.	String
NR_PROTOCOLO_CANDIDATURA	Número do protocolo de registro de candidatura da candidata ou candidato.	String
NR_PROCESSO	Número do processo de registro de candidatura da candidata ou candidato.	String
CD_SITUACAO_CANDIDATO_PLEITO	Código da situação da candidatura no dia do Pleito.	Integer
DS_SITUACAO_CANDIDATO_PLEITO	Situação da candidatura no dia do Pleito.	String
CD_SITUACAO_CANDIDATO_URNNA	Código da situação da candidatura na urna.	Integer
DS_SITUACAO_CANDIDATO_URNNA	Situação da candidatura na urna.	String
ST_CANDIDATO_INSERIDO_URNNA	Informa se a candidata ou candidato foi inserido na urna eletrônica. Valores possíveis: S - Sim e N - Não.	String
NM_TIPO_DESTINACAO_VOTOS	Nome do tipo da destinação dos votos.	String
CD_SITUACAO_CANDIDATO_TOT	Código da situação atual da candidata ou candidato no banco de totalização.	Integer
DS_SITUACAO_CANDIDATO_TOT	Descrição da situação atual da candidata ou candidato no banco de totalização.	String
ST_PREST_CONTAS	Indica se o candidato prestou contas. Valores possíveis: S - Sim e N - Não. Se existe um ou mais registros para a candidata ou candidato na tabela de prestação de contas,	String

	significa que a mesma ou o mesmo prestou contas. Se não existe, significa que não prestou contas.	
ST_SUBSTITUIDO	É facultado ao partido político ou à coligação substituir candidata ou candidato que tiver seu registro indeferido, inclusive por inelegibilidade, cancelado ou cassado, ou, ainda, que renunciar ou falecer após o termo final do prazo do registro. Esse campo informará se a candidata ou o candidato foi substituído por outro. Valores possíveis: S - Sim e N - Não.	String
SQ_SUBSTITUIDO	O sequencial da candidata ou do candidato que foi substituído. Esse campo identifica a candidata ou candidato que está sendo substituído pela candidata ou candidato corrente. O valor de ST_SUBSTITUIDO da candidata ou candidato ao qual esta coluna se refere deve ser "S".	Integer
SQ_ORDEM_SUPLENCIA	Ordem da suplência do cargo da candidata ou candidato para cargos proporcionais (deputado federal, deputado estadual e vereador). No caso de senador, temos 1º suplente e 2º suplente. Só é preenchido para candidatas e candidatos suplentes.	Integer
DT_ACEITE_CANDIDATURA	Data em que foi aceito o registro de candidatura pela Justiça Eleitoral. No caso de haver mais de uma candidatura com o mesmo número, para fins de destinação de votos, considerar somente o registro de candidatura com essa data mais recente, ou seja, o registro de candidatura substituto.	Date

### Seguem abaixo as observações do dataset:

Para o caso de candidatos não divulgáveis, seus dados pessoais são preenchidos com:

- 4, em caso de campos numéricos, exceto campo de idade;
- "NÃO DIVULGÁVEL", em caso de campos textuais;
- "", no caso de campos relativos a idade do candidato e sua data de nascimento.

Este arquivo contém o leiaute das tabelas existentes no repositório de dados eleitorais. Antes de

trabalhar os dados é importante ler as seguintes considerações:

- A codificação de caracteres dos arquivos é "Latin 1";
- Os campos estão entre aspas e separados por ponto e vírgula, inclusive os campos numéricos;
- Campos preenchidos com #NULL significam que a informação está em branco no banco de dados.

O correspondente para #NULL nos campos numéricos é -1;

- Campos preenchidos com #NE significam que naquele ano a informação não era registrada em

banco de dados pelos sistemas eleitorais. O correspondente para #NE nos campos numéricos é -3;

- O campo UF, além das unidades da federação, pode conter alguma das seguintes situações:

o BR: quando se tratar de informação a nível nacional;

o VT: quando se tratar de voto em trânsito;

o ZZ: quando se tratar de Exterior

## 2.2 Produto Interno Bruto dos Municípios - Base 2010-2020

Os dados do dataset "Produto Interno Bruto dos Municípios - Base 2010-2020" foram obtidos diretamente do Instituto Brasileiro de Geografia e Estatística (IBGE) no dia 18/08/2023. O arquivo pode ser encontrado no seguinte link:

<https://www.ibge.gov.br/estatisticas/economicas/contas-nacionais/9088-produto-interno-bruto-dos-municipios.html?t=resultados>

ou diretamente em : [https://ftp.ibge.gov.br/Pib\\_Municípios/2020/base/base\\_de\\_dados\\_2010\\_2020.xls.zip](https://ftp.ibge.gov.br/Pib_Municípios/2020/base/base_de_dados_2010_2020.xls.zip)

O arquivo utilizado é o "PIB dos Municípios - base de dados 2010-2020.xls".

Esses dados são referentes ao período de 2010 a 2020 e contêm informações sobre o PIB e outros aspectos econômicos dos municípios brasileiros.

Aqui está a descrição das colunas presentes no dataset:

Nome da coluna/campo	Descrição	Tipo
Ano	Ano de referência do dado	Numérico
Código da Grande Região	Código da Grande Região geográfica do município	Numérico
Nome da Grande Região	Nome da Grande Região geográfica do município	Texto
Código da Unidade da Federação	Código da Unidade da Federação do município	Numérico
Sigla da Unidade da Federação	Sigla da Unidade da Federação do município	Texto
Nome da Unidade da Federação	Nome da Unidade da Federação do município	Texto
Código do Município	Código do município	Numérico
Nome do Município	Nome do município	Texto
Região Metropolitana	Indicação se o município pertence a uma Região Metropolitana	Texto
Código da Mesorregião	Código da Mesorregião geográfica do município	Numérico
Nome da Mesorregião	Nome da Mesorregião geográfica do município	Texto
Código da Microrregião	Código da Microrregião geográfica do município	Numérico
Nome da Microrregião	Nome da Microrregião geográfica do município	Texto
Código da Região Geográfica Imediata	Código da Região Geográfica Imediata do município	Numérico
Nome da Região Geográfica Imediata	Nome da Região Geográfica Imediata do município	Texto
Município da Região Geográfica Imediata	Nome do município da Região Geográfica Imediata	Texto
Código da Região Geográfica Intermediária	Código da Região Geográfica Intermediária do município	Numérico
Nome da Região Geográfica Intermediária	Nome da Região Geográfica Intermediária do município	Texto
Município da Região Geográfica Intermediária	Nome do município da Região Geográfica Intermediária	Texto
Código Concentração Urbana	Código da Concentração Urbana do município	Numérico
Nome Concentração Urbana	Nome da Concentração Urbana do município	Texto
Tipo Concentração Urbana	Tipo da Concentração Urbana do município	Texto
Código Arranjo Populacional	Código do Arranjo Populacional do município	Numérico
Nome Arranjo Populacional	Nome do Arranjo Populacional do município	Texto
Hierarquia Urbana	Hierarquia Urbana do município	Texto
Hierarquia Urbana (principais categorias)	Principais categorias da Hierarquia Urbana do município	Texto
Código da Região Rural	Código da Região Rural do município	Numérico
Nome da Região Rural	Nome da Região Rural do município	Texto
Região rural (segundo classificação do núcleo)	Classificação da região rural do município segundo núcleo	Texto
Amazônia Legal	Indicação se o município está na Amazônia Legal	Texto
Semiárido	Indicação se o município está no Semiárido	Texto
Cidade-Região de São Paulo	Indicação se o município faz parte da Cidade-Região de São Paulo	Texto
Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)	Valor adicionado bruto da Agropecuária a preços correntes em mil reais	Numérico
Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)	Valor adicionado bruto da Indústria a preços correntes em mil reais	Numérico

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)	Valor adicionado bruto dos Serviços (excluindo setores específicos) a preços correntes em mil reais	Numérico
Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)	Valor adicionado bruto dos setores de Administração, defesa, educação e saúde públicas e seguridade social a preços correntes em mil reais	Numérico
Valor adicionado bruto total, a preços correntes (R\$ 1.000)	Valor adicionado bruto total a preços correntes em mil reais	Numérico
Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)	Impostos líquidos de subsídios sobre produtos a preços correntes em mil reais	Numérico
Produto Interno Bruto, a preços correntes (R\$ 1.000)	Produto Interno Bruto a preços correntes em mil reais	Numérico
Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)	Produto Interno Bruto per capita a preços correntes em reais por habitante	Numérico
Atividade com maior valor adicionado bruto	Atividade econômica com o maior valor adicionado bruto	Texto
Atividade com segundo maior valor adicionado bruto	Atividade econômica com o segundo maior valor adicionado bruto	Texto
Atividade com terceiro maior valor adicionado bruto	Atividade econômica com o terceiro maior valor adicionado bruto	Texto

### 2.3 Estimativas da População 2020

O dataset "Estimativas da População 2020" foi obtido a partir do site oficial do Instituto Brasileiro de Geografia e Estatística (IBGE) no seguinte link:

<https://www.ibge.gov.br/estatisticas/sociais/populacao/9103-estimativas-de-populacao.html?edicao=28674>

ou diretamente em : [https://ftp.ibge.gov.br/Estimativas\\_de\\_Populacao/Estimativas\\_2020/estimativa\\_dou\\_2020.xls](https://ftp.ibge.gov.br/Estimativas_de_Populacao/Estimativas_2020/estimativa_dou_2020.xls)

A data de acesso aos dados foi em 18 de agosto de 2023.

O arquivo "estimativa\_dou\_2020.xls" contém as estimativas de população publicadas no Diário Oficial da União (DOU) para os municípios e para as Unidades da Federação brasileiros em 2020.

A seguir, apresento uma tabela com a descrição de cada campo/coluna do dataset:

Nome da Coluna	Descrição	Tipo de Dado
UF	Sigla da Unidade da Federação (Estado)	String
COD. UF	Código da Unidade da Federação (Estado)	Integer
COD. MUNIC	Código do município	Integer
NOME DO MUNICÍPIO	Nome do município	String
POPULAÇÃO ESTIMADA	Estimativa da população para o ano de 2020	Integer

Essas são as informações referentes ao dataset "Estimativas da População 2020" obtido do IBGE. O arquivo contém dados de estimativas populacionais para todos os municípios e unidades federativas do Brasil no ano de 2020.

## 2.4 Municípios Brasileiros TSE

O dataset "Municípios Brasileiros TSE" contém informações sobre os municípios brasileiros, incluindo a correspondência entre os códigos do Tribunal Superior Eleitoral (TSE) e os códigos do Instituto Brasileiro de Geografia e Estatística (IBGE), bem como outras informações relevantes.

Os dados foram obtidos no repositório do GitHub do usuário "betafcc" no seguinte link: [https://github.com/betafcc/Municipios-Brasileiros-TSE/blob/master/municipios\\_brasileiros\\_tse.csv](https://github.com/betafcc/Municipios-Brasileiros-TSE/blob/master/municipios_brasileiros_tse.csv) em 07/09/2023

Abaixo, está a descrição dos campos/colunas presentes no dataset:

Nome da Coluna	Descrição	Tipo de Dado
codigo_tse	Código do município atribuído pelo Tribunal Superior Eleitoral (TSE).	Integer
uf	Sigla da unidade federativa (Estado) à qual o município pertence.	String
nome_municipio	Nome do município.	String
capital	Indicador se o município é uma capital (0 para não, 1 para sim).	Integer
codigo_ibge	Código do município atribuído pelo Instituto Brasileiro de Geografia e Estatística (IBGE).	Integer

Este dataset é útil para relacionar informações eleitorais e geográficas dos municípios brasileiros, facilitando análises e cruzamento de dados relacionados às eleições no Brasil.

### 3 Processamento/Tratamento de Dados

#### 3.1 Tecnologias e Bibliotecas

No desenvolvimento deste projeto, foram utilizadas as seguintes tecnologias:

**Python 3.9:** Python é uma linguagem de programação amplamente utilizada em projetos de ciência de dados devido à sua simplicidade, flexibilidade e uma vasta gama de bibliotecas específicas para análise de dados e aprendizado de máquina. A versão 3.9 do Python foi escolhida, aproveitando as últimas atualizações e melhorias da linguagem.

**PyCharm 2022.1 (Community Edition):** O PyCharm é um ambiente de desenvolvimento integrado (IDE) desenvolvido pela JetBrains, amplamente utilizado por desenvolvedores Python. A versão Community Edition é gratuita e oferece um conjunto poderoso de ferramentas para programação Python, incluindo realce de sintaxe, depuração, gerenciamento de projetos e integração com controle de versão. Foi escolhido como ambiente de desenvolvimento para facilitar a codificação, depuração e gerenciamento de projetos Python de forma eficiente.

Abaixo estão as importações de bibliotecas essenciais no projeto:

```
# Importando a biblioteca pandas para manipulação de dados
import pandas as pd

# Importando as bibliotecas matplotlib e seaborn para visualização de dados
import matplotlib.pyplot as plt
import seaborn as sns

# Configurando a paleta de cores do Seaborn para uma visualização agradável
sns.set_palette("viridis", 30)

# Definindo a paleta de cores do Seaborn como um mapa de cores (cmap) para uso posterior
sns.color_palette("viridis", as_cmap=True)

# Importando bibliotecas do scikit-learn para preparação de dados e modelagem
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor # Importando o modelo GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Importando a biblioteca statsmodels para análise estatística
import statsmodels.api as sm

# Importando a função ols (Ordinary Least Squares) do statsmodels para ajuste de modelos lineares
from statsmodels.formula.api import ols

# Exibe todas as colunas do DataFrame sem truncamento
pd.set_option('display.max_columns', None)

# Configuração para exibir os números com precisão fixa de 6 casas decimais
pd.set_option('display.float_format', '{:.4f}'.format)

pd.set_option('display.max_rows', None)
```

#### 3.2 Leitura dos dados de Municípios Brasileiros TSE

O código carrega um DataFrame chamado df\_TSE\_IBGE a partir de um arquivo CSV chamado 'municipios\_brasileiros\_tse.csv' usando a função pd.read\_csv().

São impressas as primeiras linhas e informações do DataFrame, como as colunas presentes e o tipo de dados de cada coluna.

Algumas colunas específicas ('codigo\_ibge' e 'codigo\_tse') são convertidas para o tipo de dado 'str'.

```
# Carrega o DataFrame com dados do TSE e IBGE
df_TSE_IBGE = pd.read_csv('DATASETS/municipios_brasileiros_tse.csv', encoding='UTF-8', delimiter=',')

# Imprime as primeiras linhas do DataFrame
print(df_TSE_IBGE.head())
# Imprime informações do DataFrame
print(df_TSE_IBGE.info())
# Imprime estatísticas descritivas do DataFrame
print(df_TSE_IBGE.describe())
# Converte colunas específicas para o tipo de dado 'str'
df_TSE_IBGE['codigo_ibge'] = df_TSE_IBGE['codigo_ibge'].astype(str)
df_TSE_IBGE['codigo_tse'] = df_TSE_IBGE['codigo_tse'].astype(str)
```

Saídas do código acima, em destaque para o total de 5570 linhas com `codigo_tse` e `codigo_ibge`:

```
*****
INICIO Coleta de Dados - Processamento/Tratamento de Dados
*****
***** df_TSE_IBGE *****
***** 
      codigo_tse  uf  nome_municipio  capital  codigo_ibge
0           1120  AC     ACRELÂNDIA          0    1200013
1           1570  AC      ASSIS BRASIL          0    1200054
2           1058  AC      BRASILEÍTA          0    1200104
3           1007  AC       BUJARI           0    1200138
4           1015  AC      CAPIXABA          0    1200179
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
---  -- 
 0   codigo_tse        5570 non-null   int64  
 1   uf                5570 non-null   object  
 2   nome_municipio    5570 non-null   object  
 3   capital           5570 non-null   int64  
 4   codigo_ibge       5570 non-null   int64  
dtypes: int64(3), object(2)
memory usage: 217.7+ KB
None
      codigo_tse  capital  codigo_ibge
count  5570.0000  5570.0000  5570.0000
mean   51469.3993  0.0048  3253590.7707
std    29375.0802  0.0695  984910.3394
min    19.0000  0.0000  1100015.0000
25%   24258.7500  0.0000  2512125.7500
50%   51241.5000  0.0000  3146280.0000
75%   79213.5000  0.0000  4119189.5000
max    99074.0000  1.0000  5300108.0000
*****
FIM df_TSE_IBGE
*****
```

### 3.3 Leitura dos dados Candidatos – 2020

Um segundo DataFrame chamado `df_eleitos` é carregado a partir de um arquivo CSV chamado '`consulta_cand_2020_BRASIL.csv`'.

São exibidas as colunas do DataFrame e informações básicas, como as colunas e o tipo de dados de cada coluna.

```
df_eleitos = pd.read_csv('DATASETS/consulta_cand_2020_BRASIL.csv', encoding='latin1', delimiter=';')

# Exibe as colunas do dataframe e informações básicas
print("Colunas do df_eleitos:")
print(df_eleitos.columns)
print("\nInformações do df_eleitos:")
print(df_eleitos.info())
```

Uma lista de colunas desejadas é definida para manter apenas as colunas relevantes no DataFrame `df_eleitos`.

O DataFrame é filtrado para manter apenas as colunas desejadas.

```
# Lista das colunas desejadas
eleitos_colunas_desejadas = [
    'CD_TIPO_ELEICAO',
    'NM_TIPO_ELEICAO',
    'SG_UF',
    'SG_UE',
    'NM_UE',
    'CD_CARGO',
    'DS_CARGO',
    'CD_SITUACAO_CANDIDATURA',
    'DS_SITUACAO_CANDIDATURA',
    'TP_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'CD_SIT_TOT_TURNO',
    'DS_SIT_TOT_TURNO',
    'ST_REELEICAO',
    'VR_DESPESA_MAX_CAMPANHA'
]

# Filtra o dataframe para manter apenas as colunas desejadas
df_eleitos = df_eleitos[eleitos_colunas_desejadas]
```

Algumas colunas ('SG\_UE') são convertidas para o tipo de dado 'str'.

```
# Converte a coluna 'SG_UE' para o tipo string, se necessário
df_eleitos['SG_UE'] = df_eleitos['SG_UE'].astype(str)
```

É exibido o número de valores únicos em cada coluna.

```
# Exibe o número de valores únicos em cada coluna
print("\nNúmero de valores únicos em cada coluna:")
print(df_eleitos.nunique())
```

Saída do código acima:

Número de valores únicos em cada coluna:

CD_TIPO_ELEICAO	2
NM_TIPO_ELEICAO	2
SG_UF	26
SG_UE	5568
NM_UE	5295
CD_CARGO	3
DS_CARGO	3
CD_SITUACAO_CANDIDATURA	3
DS_SITUACAO_CANDIDATURA	3
TP_AGREMIACAO	3
SG_PARTIDO	34
NM_PARTIDO	34
CD_SIT_TOT_TURNO	7

```
DS_SIT_TOT_TURNO          7
ST_REELEICAO                3
VR_DESPESA_MAX_CAMPANHA    3811
dtype: int64
```

### 3.3.1 Filtragem de dados Candidatos – 2020

Uma lista de colunas a serem analisadas é definida.

```
# Lista das colunas a serem analisadas
colunas_para_analisar = [
    'CD_TIPO_ELEICAO',
    'NM_TIPO_ELEICAO',
    'CD_CARGO',
    'DS_CARGO',
    'CD_SITUACAO_CANDIDATURA',
    'DS_SITUACAO_CANDIDATURA',
    'TP_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'CD_SIT_TOT_TURNO',
    'DS_SIT_TOT_TURNO',
    'ST_REELEICAO'
]

# Itera sobre as colunas para exibir os valores únicos em cada uma
for coluna in colunas_para_analisar:
    valores_unicos = df_eleitos[coluna].unique()
    print(f"Valores únicos em {coluna}: \n{valores_unicos}\n")
```

O código itera sobre essas colunas e exibe os valores únicos em cada uma.

Saídas do código citado acima:

```
Valores únicos em CD_TIPO_ELEICAO:
[2 1]

Valores únicos em NM_TIPO_ELEICAO:
['ELEIÇÃO ORDINÁRIA' 'ELEIÇÃO SUPLEMENTAR']

Valores únicos em CD_CARGO:
[11 13 12]

Valores únicos em DS_CARGO:
['PREFEITO' 'VEREADOR' 'VICE-PREFEITO']

Valores únicos em CD_SITUACAO_CANDIDATURA:
[12 3 1]

Valores únicos em DS_SITUACAO_CANDIDATURA:
['APTO' 'INAPTO' 'CADASTRADO']

Valores únicos em TP_AGREMIACAO:
['COLIGAÇÃO' 'PARTIDO ISOLADO' 'FEDERAÇÃO']
```

Valores únicos em SG\_PARTIDO:

```
[MDB' 'REPUBLICANOS' 'DEM' 'PMN' 'PTC' 'PODE' 'PSDB' 'PSC' 'AVANTE' 'PT'
'PC do B' 'PSL' 'PROS' 'PDT' 'PSD' 'PSB' 'PV' 'PTB' 'PL' 'PRTB' 'PP'
'PATRIOTA' 'SOLIDARIEDADE' 'PSOL' 'CIDADANIA' 'DC' 'REDE' 'PMB' 'NOVO'
'UP' 'PSTU' 'PCB' 'PCO' 'UNIÃO']
```

Valores únicos em NM\_PARTIDO:

```
['MOVIMENTO DEMOCRÁTICO BRASILEIRO' 'REPUBLICANOS' 'DEMOCRATAS'
'PARTIDO DA MOBILIZAÇÃO NACIONAL' 'PARTIDO TRABALHISTA CRISTÃO' 'PODEMOS'
'PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA' 'PARTIDO SOCIAL CRISTÃO'
'AVANTE' 'PARTIDO DOS TRABALHADORES' 'PARTIDO COMUNISTA DO BRASIL'
'PARTIDO SOCIAL LIBERAL' 'PARTIDO REPUBLICANO DA ORDEM SOCIAL'
'PARTIDO DEMOCRÁTICO TRABALHISTA' 'PARTIDO SOCIAL DEMOCRÁTICO'
'PARTIDO SOCIALISTA BRASILEIRO' 'PARTIDO VERDE'
'PARTIDO TRABALHISTA BRASILEIRO' 'PARTIDO LIBERAL'
'PARTIDO RENOVADOR TRABALHISTA BRASILEIRO' 'PROGRESSISTAS' 'PATRIOTA'
'SOLIDARIEDADE' 'PARTIDO SOCIALISMO E LIBERDADE' 'CIDADANIA'
'DEMOCRACIA CRISTÃ' 'REDE SUSTENTABILIDADE'
'PARTIDO DA MULHER BRASILEIRA' 'PARTIDO NOVO' 'UNIDADE POPULAR'
'PARTIDO SOCIALISTA DOS TRABALHADORES UNIFICADO'
'PARTIDO COMUNISTA BRASILEIRO' 'PARTIDO DA CAUSA OPERÁRIA' 'UNIÃO BRASIL']
```

Valores únicos em CD\_SIT\_TOT\_TURNO:

```
[1 5 4 2 3 -1 6]
```

Valores únicos em DS\_SIT\_TOT\_TURNO:

```
['ELEITO' 'SUPLENTE' 'NÃO ELEITO' 'ELEITO POR QP' 'ELEITO POR MÉDIA'
'#NULL!' '2º TURNO']
```

Valores únicos em ST\_REELEICAO:

```
['S' 'N' 'Não divulgável']
```

Uma lista de valores desejados para a coluna 'DS\_SIT\_TOT\_TURNO' é definida.

O DataFrame df\_eleitos é filtrado para manter apenas as linhas com as situações desejadas na coluna 'DS\_SIT\_TOT\_TURNO' relacionadas a um candidato eleito.

```
# Lista dos valores de 'DS_SIT_TOT_TURNO' desejados
eleitos_selecionados_eleito = ['ELEITO', 'ELEITO POR QP', 'ELEITO POR MÉDIA']

# Filtra o dataframe para manter apenas as linhas com as situações desejadas
df_eleitos = df_eleitos[df_eleitos['DS_SIT_TOT_TURNO'].isin(eleitos_selecionados_eleito)]

# Exibe informações do dataframe após a primeira filtragem
print("\nInformações do dataframe após a primeira filtragem DS_SIT_TOT_TURNO:")
print(df_eleitos.info())
```

Saídas do código acima, com destaque para a redução para 69167 linhas no data frame:

```
Informações do dataframe após a primeira filtragem DS_SIT_TOT_TURNO:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69167 entries, 0 to 558555
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CD_TIPO_ELEICAO  69167 non-null   int64
```

```

1  NM_TIPO_ELEICAO      69167 non-null  object
2  SG_UF                 69167 non-null  object
3  SG_UE                 69167 non-null  object
4  NM_UE                 69167 non-null  object
5  CD_CARGO               69167 non-null  int64
6  DS_CARGO               69167 non-null  object
7  CD_SITUACAO_CANDIDATURA 69167 non-null  int64
8  DS_SITUACAO_CANDIDATURA 69167 non-null  object
9  TP_AGREMIACAO          69167 non-null  object
10 SG_PARTIDO              69167 non-null  object
11 NM_PARTIDO              69167 non-null  object
12 CD_SIT_TOT_TURNO        69167 non-null  int64
13 DS_SIT_TOT_TURNO        69167 non-null  object
14 ST_REELEICAO             69167 non-null  object
15 VR_DESPESA_MAX_CAMPANHA 69167 non-null  float64
dtypes: float64(1), int64(4), object(11)
memory usage: 9.0+ MB

```

O DataFrame é novamente filtrado para manter apenas as linhas com a situação de candidatura desejada na coluna 'DS\_SITUACAO\_CANDIDATURA' seja como 'APTO'.

```

# Lista dos valores de 'DS_SITUACAO_CANDIDATURA' desejados
eleitos_selecionados_apto = ['APTO']

# Filtra o dataframe para manter apenas as situações de candidatura desejadas
df_eleitos = df_eleitos[df_eleitos['DS_SITUACAO_CANDIDATURA'].isin(eleitos_selecionados_apto)]

# Exibe informações do dataframe após a segunda filtragem
print("\nInformações do dataframe após a segunda filtragem DS_SITUACAO_CANDIDATURA:")
print(df_eleitos.info())

```

Saídas do código acima, com destaque para a redução para 69146 linhas no data frame:

```

Informações do dataframe após a segunda filtragem DS_SITUACAO_CANDIDATURA:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69146 entries, 0 to 558555
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CD_TIPO_ELEICAO    69146 non-null  int64  
 1   NM_TIPO_ELEICAO    69146 non-null  object 
 2   SG_UF              69146 non-null  object 
 3   SG_UE              69146 non-null  object 
 4   NM_UE              69146 non-null  object 
 5   CD_CARGO            69146 non-null  int64  
 6   DS_CARGO            69146 non-null  object 
 7   CD_SITUACAO_CANDIDATURA 69146 non-null  int64  
 8   DS_SITUACAO_CANDIDATURA 69146 non-null  object 
 9   TP_AGREMIACAO       69146 non-null  object 
 10  SG_PARTIDO           69146 non-null  object 
 11  NM_PARTIDO           69146 non-null  object 
 12  CD_SIT_TOT_TURNO     69146 non-null  int64  
 13  DS_SIT_TOT_TURNO     69146 non-null  object 
 14  ST_REELEICAO          69146 non-null  object 
 15  VR_DESPESA_MAX_CAMPANHA 69146 non-null  float64
dtypes: float64(1), int64(4), object(11)
memory usage: 9.0+ MB
None

```

Uma lista de colunas a serem removidas é definida e essas colunas são removidas do Data-Frame.

```
# Lista das colunas a serem removidas
colunas_para_remover = [
    "NM_UE",
    "CD_CARGO",
    "CD_TIPO_ELEICAO",
    "NM_TIPO_ELEICAO",
    "CD_SITUACAO_CANDIDATURA",
    "DS_SITUACAO_CANDIDATURA",
    "CD_SIT_TOT_TURNO",
    "DS_SIT_TOT_TURNO"
]

# Remove as colunas especificadas do dataframe
df_eleitos = df_eleitos.drop(columns=colunas_para_remover)

# Exibe informações do dataframe após a remoção de colunas
print("\nInformações do dataframe após a remoção de colunas:")
print(df_eleitos.info())

print('*' * 100)
print('FIM df_eleitos')
print('*' * 100)
```

Saídas do código acima, com destaque para a redução do número de colunas final no data frame:

```
Informações do dataframe após a remoção de colunas:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69146 entries, 0 to 558555
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   SG_UF            69146 non-null   object  
 1   SG_UE            69146 non-null   object  
 2   DS_CARGO         69146 non-null   object  
 3   TP_AGREMIACAO   69146 non-null   object  
 4   SG_PARTIDO       69146 non-null   object  
 5   NM_PARTIDO       69146 non-null   object  
 6   ST_REELEICAO     69146 non-null   object  
 7   VR_DESPESA_MAX_CAMPANHA 69146 non-null   float64 
dtypes: float64(1), object(7)
memory usage: 4.7+ MB
None
*****
FIM df_eleitos
*****
```

### 3.4 Leitura dos dados Estimativas da População 2020

O código carrega um terceiro DataFrame chamado `df_populacao` a partir de um arquivo Excel chamado 'estimativa\_dou\_2020.xls'.

São exibidas as primeiras linhas das colunas 'COD. UF' e 'COD. MUNIC'.

```
print('*' * 100)
print('INICIO df_populacao')
print('*' * 100)
# Carregar o DataFrame a partir do arquivo CSV
# Usar skiprows=1 para ignorar o cabeçalho
df_populacao = pd.read_excel('DATASETS/estimativa_dou_2020.xls',
                             skiprows=1, header=0, sheet_name='Municípios')

# Exibir as 10 primeiras linhas das colunas 'COD. UF' e 'COD. MUNIC'
print(df_populacao['COD. UF'].head(10))
print(df_populacao['COD. MUNIC'].head(10))
```

Saídas do código acima, no arquivo original, há um título na primeira linha:

```
*****
INICIO df_populacao
*****
0    11.0000
1    11.0000
2    11.0000
3    11.0000
4    11.0000
5    11.0000
6    11.0000
7    11.0000
8    11.0000
9    11.0000
Name: COD. UF, dtype: float64
0    15.0000
1    23.0000
2    31.0000
3    49.0000
4    56.0000
5    64.0000
6    72.0000
7    80.0000
8    98.0000
9   106.0000
Name: COD. MUNIC, dtype: float64
```

As colunas categóricas e numéricas são identificadas e exibidas.

```
# Identificar as colunas categóricas e numéricas
categorical_columns = df_populacao.select_dtypes(include=['object']).columns
numeric_columns = df_populacao.select_dtypes(include=['int', 'float']).columns
```

```
# Análise das variáveis categóricas
print('Análise das variáveis categóricas')

for col in categorical_columns:
    print(f'Análise de {col}:')
    print(f'Total de valores únicos: {df_populacao[col].nunique()}')
    print(f'Valores únicos: {df_populacao[col].unique()}')
    print("===")
```

Saídas do código acima, com destaque nos valores estranhos em UF e nulos em todas:

```
Análise das variáveis categóricas
Análise de UF:
Total de valores únicos: 41
Valores únicos: ['RO' 'AC' 'AM' 'RR' 'PA' 'AP' 'TO' 'MA' 'PI' 'CE' 'RN' 'PB' 'PE' 'AL'
 'SE' 'BA' 'MG' 'ES' 'RJ' 'SP' 'PR' 'SC' 'RS' 'MS' 'MT' 'GO' 'DF' nan
 'Fonte: IBGE. Diretoria de Pesquisas - DPE - Coordenação de População e Indicadores Sociais -
 COPIS.'
 'Notas:'
 '(1) População judicial do município de Porto Velho-RO: 494.013 habitantes. Processo Judicial nº
 12316-40.2016.4.01.4100 - Seção Judiciária de Rondônia.'
 '(2) População judicial do município de Manaus-AM: entre 30.565 e 37.356 habitantes. Parecer de
 Força Executória nº 00010/2017/NUCOB-GEAC/PFAM/PGF/AGU, em trâmite na 3ª VF/AM.'
 '(3) População judicial do município de Santa Isabel do Rio Negro-AM: entre 23.773 e 30.564
 habitantes. Parecer de Força Executória nº 00007/2017/NUCOB-GEAC/PFAM/PGF/AGU, em trâmite na 3ª VF/AM.'
 '(4) População judicial do município de Urucará-AM: entre 16.981 e 23.772 habitantes. Parecer de
 Força Executória nº 00004/2017/NUCOB-GEAC/PFAM/PGF/AGU, em trâmite na 3ª VF/AM.'
 '(5) População judicial do município de Jacareacanga-PA: 41.487 habitantes. Processo Judicial nº
 798-41.2011.4.01.3902, Seção Judiciária de Itaituba-PA.'
 '(6) População judicial do município de Paço do Lumiar - MA: superior a 156.216 habitantes. Processo
 Judicial nº13916-98.2017.4.01.3700 - Seção Judiciária do Maranhão- MA.'
 '(7) População judicial do município de São Gonçalo do Amarante-RN: entre 101.881 e 115.464
 habitantes. Processo Judicial nº: 0813188-75.2017.4.05.8400 (4ª Vara Federal - RN).'
 '(8) População judicial do município de Ferreiros -PE: entre 13.585 e 16.980 habitantes. Processo
 Judicial nº 0805921-61.2019.4.05.0000 (1ª Turma do Tribunal Regional Federal da 5ª Região).'
 '(9) População judicial do município Coronel João Sá-BA: 17.422 habitantes. Processo Judicial nº
 0002222-53.2017.4.01.3306 - Vara Única de Paulo Afonso-BA.'
 '(10) População judicial do município Ibiassucê-BA: entre 10.189 e 13.584 habitantes. Processo
 Judicial\x00 nº 1018608-53.2017.4.01.3400 (14ª VARA FEDERAL CÍVEL DA SJDF).'
 '(11) População judicial do município de Rodelas - BA: entre 10.189 a 13.584 habitantes. Processo
 Judicial nº 1002950-09.2019.4.01.3306 (Procuradoria Federal no Estado da Bahia, Núcleo Finalístico
 Ambiental)..'
 '(12) População judicial do município de Vera Cruz - BA: entre 44.149 a 50.940 habitantes. Processo
 Judicial nº 8000464-59.2018.8.05.0124 (Vara dos Feitos Relativos às Relações de Consumo, Cíveis e Comerciais,
 Registros Públicos e Acidentes de Trabalho e Juizado Especial Adjunto da Comarca de Itaparica).'
 ===
 Análise de NOME DO MUNICÍPIO:
 Total de valores únicos: 5297
 Valores únicos: ['Alta Floresta D'Oeste' 'Ariquemes' 'Cabixi' ... 'Vila Propício'
 'Brasília' nan]
 ===
 Análise de POPULAÇÃO ESTIMADA:
 Total de valores únicos: 5125
 Valores únicos: [22728 109523 5188 ... 8873 3055149 nan]
 ===
```

```

print('# Análise das variáveis numéricas')
# Análise das variáveis numéricas
for col in numeric_columns:
    print(f'Análise de {col}:')
    print(f'Média: {df_populacao[col].mean()}')
    print(f'Desvio padrão: {df_populacao[col].std()}')
    print(f'Valor mínimo: {df_populacao[col].min()}')
    print(f'Valor máximo: {df_populacao[col].max()}')
    print("====")

# Exibir informações gerais sobre o DataFrame
print(df_populacao.info())

```

Saídas do código acima:

```

# Análise das variáveis numéricas
Análise de COD. UF:
Média: 32.37773788150808
Desvio padrão: 9.833861774653943
Valor mínimo: 11.0
Valor máximo: 53.0
=====
Análise de COD. MUNIC:
Média: 15816.982585278276
Desvio padrão: 15997.29977960963
Valor mínimo: 13.0
Valor máximo: 72202.0
=====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5586 entries, 0 to 5585
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
 ---  --  
 0   UF                5584 non-null    object  
 1   COD. UF           5570 non-null    float64 
 2   COD. MUNIC        5570 non-null    float64 
 3   NOME DO MUNICÍPIO 5570 non-null    object  
 4   POPULAÇÃO ESTIMADA 5570 non-null    object  
dtypes: float64(2), object(3)
memory usage: 218.3+ KB
None

```

### 3.4.1 Tratamento dos Dados de Estimativas da População 2020

O DataFrame então é filtrado para manter apenas as linhas cujos estados ('UF') estão na lista 'uf\_correcao'.

```

# Filtrar as linhas onde a coluna 'UF' está na lista 'uf_correcao'
uf_correcao = ['RO', 'AC', 'AM', 'RR', 'PA', 'AP',
                'TO', 'MA', 'PI', 'CE', 'RN', 'PB',
                'PE', 'AL', 'SE', 'BA', 'MG', 'ES',
                'RJ', 'SP', 'PR', 'SC', 'RS', 'MS',
                'MT', 'GO', 'DF']

# Filtre as linhas onde a coluna 'UF' está na lista 'uf_correcao'.
df_populacao = df_populacao[df_populacao['UF'].isin(uf_correcao)]

# Análise da coluna 'UF' após a filtragem
print(f"Análise de 'UF':")
print(f"Total de valores únicos: {df_populacao['UF'].nunique()}")
print(f"Valores únicos: {df_populacao['UF'].unique()}")
print("====")

# Exibir informações atualizadas sobre o DataFrame após a filtragem
print(df_populacao.info())
# Exibir estatísticas descritivas do DataFrame
print(df_populacao.describe())

```

Saídas do código acima:

```

Análise de 'UF':
Total de valores únicos: 27
Valores únicos: ['RO' 'AC' 'AM' 'RR' 'PA' 'AP' 'TO' 'MA' 'PI' 'CE' 'RN' 'PB' 'PE' 'AL'
                  'SE' 'BA' 'MG' 'ES' 'RJ' 'SP' 'PR' 'SC' 'RS' 'MS' 'MT' 'GO' 'DF']
=====
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   UF                5570 non-null    object  
 1   COD. UF           5570 non-null    float64 
 2   COD. MUNIC        5570 non-null    float64 
 3   NOME DO MUNICÍPIO 5570 non-null    object  
 4   POPULAÇÃO ESTIMADA 5570 non-null    object  
dtypes: float64(2), object(3)
memory usage: 261.1+ KB
None
      COD. UF  COD. MUNIC
count  5570.0000  5570.0000
mean    32.3777  15816.9826
std     9.8339  15997.2998
min    11.0000   13.0000
25%   25.0000   4507.2500
50%   31.0000  10400.5000
75%   41.0000  20853.0000
max    53.0000  72202.0000

```

A coluna 'POPULAÇÃO ESTIMADA' é tratada para lidar com valores não numéricos (como '(1)', '(2)', etc.).

Os valores não numéricos são identificados e armazenados em uma lista chamada 'população\_estimada\_correcao'.

```
# Tratar a coluna 'POPULAÇÃO ESTIMADA'
# Use a função `pd.to_numeric` com `errors='coerce'` para converter a coluna em números
# e lidar com valores não numéricos.
valores_nao_numericos = df_populacao[df_populacao['POPULAÇÃO ESTIMADA'].apply(pd.to_numeric, errors='coerce').isna()]

# Exibir os registros não numéricos
print(valores_nao_numericos)

# Lista de indices de valores não numéricos
lista_valores_nao_numericos = valores_nao_numericos.index.tolist()
print(lista_valores_nao_numericos)

# Valores a serem corrigidos na coluna 'POPULAÇÃO ESTIMADA'
população_estimada_correcao = ['(1)', '(2)', '(3)', '(4)',
                                '(5)', '(6)', '(7)', '(8)',
                                '(9)', '(10)', '(11)', '(12)']
```

Saídas do código acima, com destaque nos valores estranhos de POPULAÇÃO ESTIMADA:

	UF	COD. UF	COD. MUNIC	NOME DO MUNICÍPIO	POPULAÇÃO ESTIMADA
16	RO	11.0000	205.0000	Porto Velho	539354(1)
110	AM	13.0000	2553.0000	Manaquiri	33049 (2)
123	AM	13.0000	3601.0000	Santa Isabel do Rio Negro	25865(3)
134	AM	13.0000	4302.0000	Urucará	16130 (4)
210	PA	15.0000	3754.0000	Jacareacanga	7590 (5)
579	MA	21.0000	7506.0000	Paço do Lumiar	123747(6)
1205	RN	24.0000	12005.0000	São Gonçalo do Amarante	103672(7)
1526	PE	26.0000	5509.0000	Ferreiros	12170(8)
1937	BA	29.0000	9208.0000	Coronel João Sá	15717 (9)
1973	BA	29.0000	12004.0000	Ibiassucê	9031(10)
2159	BA	29.0000	27101.0000	Rodelas	9442 (11)
2237	BA	29.0000	33208.0000	Vera Cruz	43716 (12)
[16, 110, 123, 134, 210, 579, 1205, 1526, 1937, 1973, 2159, 2237]					

A função substituir\_valor é chamada para substituir os valores não numéricos por valores válidos.

```
# Aplicar a substituição apenas nos índices especificados em 'lista_valores_nao_numericos'
df_populacao.loc[lista_valores_nao_numericos, 'POPULAÇÃO ESTIMADA'] = df_populacao.loc[
    lista_valores_nao_numericos, 'POPULAÇÃO ESTIMADA'].apply(substituir_valor)
print("df_populacao.loc[lista_valores_nao_numericos, 'POPULAÇÃO ESTIMADA']")
print(df_populacao.loc[lista_valores_nao_numericos])

# Função para substituir os valores da lista 'população_estimada_correcao' por uma string vazia
def substituir_valor(valor):
    for item in população_estimada_correcao:
        valor = valor.replace(item, '')
        valor = valor.replace(" ", "")

    return valor
```

A coluna 'POPULAÇÃO ESTIMADA' é convertida para o tipo de dado numérico.

Outras colunas são convertidas para tipos de dados apropriados (por exemplo, 'UF' para string e 'COD. MUNIC' é formatada).

```

# Usar a função pd.to_numeric para converter a coluna em números, definindo errors='coerce'
# para tratar não numéricos como NaN.
df_populacao['POPULAÇÃO ESTIMADA'] = pd.to_numeric(df_populacao['POPULAÇÃO ESTIMADA'], errors='coerce')

# Converter as colunas para o tipo de dados 'strings'
df_populacao['UF'] = df_populacao['UF'].astype(str)
df_populacao['COD. UF'] = df_populacao['COD. UF'].astype(str).str.split('.').str[0]

# Converter a coluna 'COD. MUNIC' em strings, dividir por '.', pegar a parte inteira
# e aplicar o preenchimento com zeros à esquerda
df_populacao['COD. MUNIC'] = df_populacao['COD. MUNIC'].astype(str)
df_populacao['COD. MUNIC'] = df_populacao['COD. MUNIC'].apply(lambda x: str(x).replace('.', '').zfill(6))

df_populacao['NOME DO MUNICÍPIO'] = df_populacao['NOME DO MUNICÍPIO'].astype(str)

# Verificar os tipos de dados atualizados
print(df_populacao.dtypes)

# Exibir estatísticas descritivas atualizadas do DataFrame
print(df_populacao.describe())
# Exibir informações atualizadas sobre o DataFrame
print(df_populacao.info())
# Exibir as colunas do DataFrame
print(df_populacao.columns)

print('*' * 100)
print('FIM df_populacao')
print('*' * 100)

```

Saídas do código acima, com destaque na quantidade total de linhas 5570 do df\_população:

```

df_populacao.loc[[listas_valores_nao_numericos, 'POPULAÇÃO ESTIMADA']]

      UF  COD. UF  COD. MUNIC           NOME DO MUNICÍPIO  POPULAÇÃO ESTIMADA
16    RO   11.0000  205.0000        Porto Velho            539354
110   AM   13.0000 2553.0000       Manaquiri             33049
123   AM   13.0000 3601.0000  Santa Isabel do Rio Negro     25865
134   AM   13.0000 4302.0000       Urucará              16130
210   PA   15.0000 3754.0000  Jacareacanga            7590
579   MA   21.0000 7506.0000       Paço do Lumiar          123747
1205  RN   24.0000 12005.0000  São Gonçalo do Amarante        103672
1526  PE   26.0000 5509.0000       Ferreiros             12170
1937  BA   29.0000 9208.0000  Coronel João Sá            15717
1973  BA   29.0000 12004.0000      Ibiassucê              9031
2159  BA   29.0000 27101.0000      Rodelas                9442
2237  BA   29.0000 33208.0000      Vera Cruz              43716
UF                  object
COD. UF               object
COD. MUNIC             object
NOME DO MUNICÍPIO      object
POPULAÇÃO ESTIMADA      int64
dtype: object

      POPULAÇÃO ESTIMADA
count            5570.0000
mean            38017.1799
std             222892.9921
min             776.0000
25%            5442.2500
50%            11665.5000
75%            25663.7500
max            12325232.0000
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  -- 
 0   UF               5570 non-null   object 

```

```

1 COD. UF           5570 non-null   object
2 COD. MUNIC        5570 non-null   object
3 NOME DO MUNICÍPIO 5570 non-null   object
4 POPULAÇÃO ESTIMADA 5570 non-null   int64
dtypes: int64(1), object(4)
memory usage: 390.1+ KB
None
Index(['UF', 'COD. UF', 'COD. MUNIC', 'NOME DO MUNICÍPIO',
       'POPULAÇÃO ESTIMADA'],
      dtype='object')
*****
FIM df_populacao
*****

```

### 3.5 Leitura dos Dados Produto Interno Bruto dos Municípios - Base 2010-2020

De início, carrega-se um quarto DataFrame chamado df\_renda a partir de um arquivo Excel chamado 'PIB dos Municípios - base de dados 2010-2020.xls'.

São exibidas as colunas do DataFrame e informações básicas:

Saídas do código acima, com destaque nos valores estranhos em UF e nulos em todas:

```

print('*' * 100)
print('INICIO df_renda')
print('*' * 100)

# Carrega o conjunto de dados do PIB dos Municípios
df_renda = pd.read_excel('DATASETS/PIB dos Municípios - base de dados 2010-2020.xls',
                         skiprows=0, header=0, sheet_name='PIB_dos_Municípios')

# Exibe as colunas do DataFrame
print("Colunas do df_renda:")
print(df_renda.columns)
# Exibe informações sobre o DataFrame
print("Informações do df_renda:")
print(df_renda.info())

```

Saídas do código acima, com destaque na quantidade inicial de linhas (61255) e colunas (43) do data frame:

```

*****
INICIO df_renda
*****
Colunas do df_renda:
Index(['Ano', 'Código da Grande Região', 'Nome da Grande Região',
       'Código da Unidade da Federação', 'Sigla da Unidade da Federação',
       'Nome da Unidade da Federação', 'Código do Município',
       'Nome do Município', 'Região Metropolitana', 'Código da Mesorregião',
       'Nome da Mesorregião', 'Código da Microrregião', 'Nome da Microrregião',
       'Código da Região Geográfica Imediata',
       'Nome da Região Geográfica Imediata',
       'Município da Região Geográfica Imediata',
       'Código da Região Geográfica Intermediária',
       'Nome da Região Geográfica Intermediária',
       'Município da Região Geográfica Intermediária'],
      dtype='object')

```

```

'Código Concentração Urbana', 'Nome Concentração Urbana',
'Tipo Concentração Urbana', 'Código Arranjo Populacional',
'Nome Arranjo Populacional', 'Hierarquia Urbana',
'Hierarquia Urbana (principais categorias)', 'Código da Região Rural',
'Nome da Região Rural',
'Região rural (segundo classificação do núcleo)', 'Amazônia Legal',
'Semiárido', 'Cidade-Região de São Paulo',
'Valor adicionado bruto da Agropecuária, \na preços correntes\n(R$ 1.000)',
'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)',
'Valor adicionado bruto dos Serviços,\nnos preços correntes \n- exceto Administração, defesa, educação e saúde públicas e seguridade social\n(R$ 1.000)',
'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, \nnos preços correntes\n(R$ 1.000)',
'Valor adicionado bruto total, \nnos preços correntes\n(R$ 1.000)',
'Impostos, líquidos de subsídios, sobre produtos, \nnos preços correntes\n(R$ 1.000)',
'Produto Interno Bruto, \nnos preços correntes\n(R$ 1.000)',
'Produto Interno Bruto per capita, \nnos preços correntes\n(R$ 1,00)',
'Atividade com maior valor adicionado bruto',
'Atividade com segundo maior valor adicionado bruto',
'Atividade com terceiro maior valor adicionado bruto'],
dtype='object')

Informações do df_renda:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61255 entries, 0 to 61254
Data columns (total 43 columns):
#
Non-Null Count Dtype   Column
--- 
----- 
0          Ano
61255 non-null int64
1          Código
61255 non-null int64
2          Nome
61255 non-null object
3          Código
61255 non-null int64
4          Sigla
61255 non-null object
5          Nome
61255 non-null object
6          Código
61255 non-null int64
7          Nome
61255 non-null object
8          Região
15740 non-null object
9          Código
61255 non-null int64
10         Nome
61255 non-null object
11         Código
61255 non-null int64
12         Nome
61255 non-null object
13         Código
61255 non-null int64
14         Nome
61255 non-null object

```

	Município	da	Região	Geográfica	Imediata
15					
61255 non-null object					
16	Código	da	Região	Geográfica	Intermediária
61255 non-null int64					
17	Nome	da	Região	Geográfica	Intermediária
61255 non-null object					
18	Município	da	Região	Geográfica	Intermediária
61255 non-null object					
19		Código		Concentração	Urbana
7251 non-null float64					
20		Nome		Concentração	Urbana
7251 non-null object					
21		Tipo		Concentração	Urbana
7251 non-null object					
22		Código		Arranjo	Populacional
10507 non-null float64					
23		Nome		Arranjo	Populacional
10507 non-null object					
24			Hierarquia		Urbana
61255 non-null object					
25		Hierarquia	Urbana	(principais	categorias)
61255 non-null object					
26		Código	da	Região	Rural
61255 non-null int64					
27		Nome	da	Região	Rural
61255 non-null object					
28	Região	rural	(segundo	classificação	do
61255 non-null object					núcleo)
29				Amazônia	Legal
61255 non-null object					
30					Semiárido
61255 non-null object					
31		Cidade-Região	de	São	Paulo
61255 non-null object					
32	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)				61255 non-null float64
33	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)				61255 non-null
float64					
34	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)				
61255 non-null float64					
35	Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)		61255 non-null float64		
36	Valor adicionado bruto total, a preços correntes (R\$ 1.000)				61255 non-
null float64					
37	Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)			61255 non-null float64	
38	Produto Interno Bruto, a preços correntes (R\$ 1.000)				61255
non-null float64					

```

39 Produto Interno Bruto per capita,
a preços correntes
(R$ 1,00)
float64
40 Atividade com maior valor adicionado bruto
61255 non-null object
41 Atividade com segundo maior valor adicionado bruto
61255 non-null object
42 Atividade com terceiro maior valor adicionado bruto
61255 non-null object
dtypes: float64(10), int64(9), object(24)
memory usage: 20.1+ MB
None

```

### 3.5.1 Filtragem de Dados Produto Interno Bruto dos Municípios - Base 2010-2020

O DataFrame é filtrado para manter apenas os dados do ano de 2020.

```

# Filtra o DataFrame para manter apenas os dados do ano de 2020
df_renda = df_renda[df_renda['Ano'] == 2020]

# Exibe informações atualizadas do DataFrame
print("Informações do DataFrame após filtragem:")
print(df_renda.info())

```

Saídas do código acima, com destaque na quantidade de linhas (5570) e colunas (43) do data frame:

```

Informações do DataFrame após filtragem:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 55685 to 61254
Data columns (total 43 columns):
 #                                                 Column
Non-Null Count Dtype   #                                                 Column
---  -----
-----  -----
0          Ano
5570 non-null int64
1          Código
5570 non-null int64
2          Nome
5570 non-null object
3          Código
5570 non-null int64
4          Sigla
5570 non-null object
5          Nome
5570 non-null object
6          Código
5570 non-null int64
7          Nome
5570 non-null object
8          Região
1432 non-null object
9          Código
5570 non-null int64

```

10		Nome	da	Mesorregião	
5570 non-null	object				
11		Código	da	Microrregião	
5570 non-null	int64				
12		Nome	da	Microrregião	
5570 non-null	object				
13		Código	da	Geográfica	Imediata
5570 non-null	int64				
14		Nome	da	Geográfica	Imediata
5570 non-null	object				
15		Município	da	Geográfica	Imediata
5570 non-null	object				
16		Código	da	Geográfica	Intermediária
5570 non-null	int64				
17		Nome	da	Geográfica	Intermediária
5570 non-null	object				
18		Município	da	Geográfica	Intermediária
5570 non-null	object				
19		Código		Concentração	Urbana
660 non-null	float64				
20		Nome		Concentração	Urbana
660 non-null	object				
21		Tipo		Concentração	Urbana
660 non-null	object				
22		Código		Arranjo	Populacional
956 non-null	float64				
23		Nome		Arranjo	Populacional
956 non-null	object				
24		Hierarquia			Urbana
5570 non-null	object				
25		Hierarquia	Urbana	(principais	categorias)
5570 non-null	object				
26		Código	da	Região	Rural
5570 non-null	int64				
27		Nome	da	Região	Rural
5570 non-null	object				
28		Região	rural	(segundo	classificação
5570 non-null	object				do
29					núcleo)
5570 non-null	object				
30				Amazônia	Legal
5570 non-null	object				
31		Cidade-Região	de	São	Paulo
5570 non-null	object				
32	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)				5570 non-null
float64					
33	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)				5570 non-
null float64					
34	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R\$ 1.000) 5570 non-null float64				
35	Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R\$ 1.000) 5570 non-null float64				

```

36  Valor adicionado bruto total,
a preços correntes
(R$ 1.000) 5570
non-null float64
37  Impostos, líquidos de subsídios, sobre produtos,
a preços correntes
(R$ 1.000) 5570 non-null float64
38  Produto Interno Bruto,
a preços correntes
(R$ 1.000) 5570
non-null float64
39  Produto Interno Bruto per capita,
a preços correntes
(R$ 1,00) 5570
non-null float64
40          Atividade      com     maior     valor    adicionado     bruto
5570 non-null object
41          Atividade      com     segundo    maior     valor    adicionado     bruto
5570 non-null object
42          Atividade      com     terceiro    maior     valor    adicionado     bruto
5570 non-null object
dtypes: float64(10), int64(9), object(24)
memory usage: 1.9+ MB
None

```

São selecionadas as colunas desejadas no DataFrame. O DataFrame é então atualizado para manter apenas as colunas listadas e renomeia as colunas para nomes mais legíveis.

A coluna 'Código do Município' é convertida para o tipo de dados *string* para evitar problemas de formatação.

```
# Mantém apenas as colunas de interesse no DataFrame
df_renda = df_renda[renda_colunas_selecionadas]
# Renomeia as colunas para nomes mais legíveis
df_renda.rename(columns={
    'Valor adicionado bruto da Agropecuária, \nna preços correntes\n(R$ 1.000)': 'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',

    'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)': 'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',

    'Valor adicionado bruto dos Serviços,\nna preços correntes \n- exceto Administração, ' +
    'defesa, educação e saúde públicas e seguridade social\n(R$ 1.000)': 'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa,' +
    'educação e saúde públicas e seguridade social (R$ 1.000)',

    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social,' +
    '\nna preços correntes\n(R$ 1.000)': 'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social,' +
    'a preços correntes (R$ 1.000)',

    'Valor adicionado bruto total, \nna preços correntes\n(R$ 1.000)': 'Valor adicionado bruto total, a preços correntes (R$ 1.000)',

    'Impostos, líquidos de subsídios, sobre produtos, \nna preços correntes\n(R$ 1.000)': 'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',

    'Produto Interno Bruto per capita, \nna preços correntes\n(R$ 1,00)': 'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'
}, inplace=True)

# Renomeia a coluna 'Nome da Grande Região'
df_renda = df_renda.rename(columns={'Nome da Grande Região': 'Nome_da_Grande_Região'})
```

```
# Exibe informações atualizadas do DataFrame
print("Informações do df_renda após seleção e renomeação de colunas:")
print(df_renda.info())

# Exibe estatísticas descritivas do DataFrame
print("Estatísticas Descritivas do df_renda:")
print(df_renda.describe())

print('*' * 100)
print('FIM df_renda')
print('*' * 100)
```

Saídas do código acima, contendo o data frame df renda final:

```
Informações do df_renda após seleção e renomeação de colunas:  
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 5570 entries, 55685 to 61254  
Data columns (total 9 columns):  
 #                                         Column  
Non-Null Count Dtype     
---  
-----  
0                           Nome_da_Grande_Região  
5570 non-null  object  
1                           Código  
5570 non-null  object  
do  
Município
```

```

2      Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
5570 non-null float64
3      Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
5570 non-null float64
4  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) 5570 non-null float64
5  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000) 5570 non-null float64
6      Valor adicionado bruto total, a preços correntes (R$ 1.000)
5570 non-null float64
7  Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
5570 non-null float64
8      Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
5570 non-null float64
dtypes: float64(7), object(2)
memory usage: 435.2+ KB
None
Estatísticas Descritivas do df_renda:
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \
count          5570.0000
mean           78028.9048
std            163678.1503
min            0.0000
25%           13648.5425
50%           33817.6590
75%           79504.5878
max          3533041.0140

Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \
count          5570.0000
mean           266487.7917
std            1502496.6284
min            440.6840
25%           4577.6773
50%           15101.2145
75%           86468.6263
max          58077783.6010

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) \
count          5570.0000
mean           633586.8941
std            7911585.4041
min            2545.3280
25%           22929.6870
50%           58109.2225
75%           182307.4985
max          520357968.8930

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000) \
count          5570.0000
mean           205906.6427
std            1842211.5109
min            8946.7740
25%           30534.5590
50%           58373.7205
75%           125388.7250
max          111267001.3810

```

```

Valor adicionado bruto total, a preços correntes (R$ 1.000) \
count                      5570.0000
mean                       1184010.2334
std                        10379905.6585
min                        16344.8180
25%                         92311.7995
50%                         202598.8070
75%                         536598.2435
max                        624409861.1530

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
count                      5570.0000
mean                       182165.1706
std                        2027286.1168
min                        395.8550
25%                         4092.6925
50%                         11776.5065
75%                         45373.4298
max                        124349145.8120

Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
count                      5570.0000
mean                       27458.3039
std                        28114.5488
min                        4924.0400
25%                         11573.5175
50%                         20159.7350
75%                         33931.1250
max                        591101.1100
*****
FIM df_renda
*****

```

### 3.6 Junção de dados de População e PIB

As colunas 'COD. UF' e 'COD. MUNIC' são usadas para criar uma coluna temporária para facilitar a junção entre os DataFrames df\_renda e df\_populacao. Neste trecho, duas colunas temporárias são criadas no DataFrame df\_populacao.

Essas colunas são criadas para facilitar a junção posterior entre os DataFrames df\_renda e df\_populacao. A primeira linha concatena as colunas 'COD. UF' e 'COD. MUNIC', transformando-as em uma única string e removendo o último caractere da string resultante. A segunda linha garante que o tipo de dados da nova coluna seja uma string.

```

# Cria uma coluna temporária em ambos os DataFrames para facilitar a junção
df_populacao['COD. UF + COD. MUNIC'] = (df_populacao['COD. UF'] + df_populacao['COD. MUNIC']).astype(str).str[:-1]
df_populacao['COD. UF + COD. MUNIC'] = df_populacao['COD. UF + COD. MUNIC'].astype(str)
print("Exemplo de dados nas colunas 'COD_UF', 'COD_MUNIC' e 'COD. UF + COD. MUNI':")
print(df_populacao['COD. UF'].head(5))
print(df_populacao['COD. MUNIC'].head(5))
print(df_populacao['COD. UF + COD. MUNIC'].head(5))

```

Saídas do código acima, com destaque nos valores de 'COD. UF + COD. MUNIC':

```

*****
Inicio df_dados_demograficos = df_renda + df_populacao
*****
Exemplo de dados nas colunas 'COD_UF', 'COD_MUNIC' e 'COD. UF + COD. MUNI':

```

```

0    11
1    11
2    11
3    11
4    11
Name: COD. UF, dtype: object
0    000150
1    000230
2    000310
3    000490
4    000560
Name: COD. MUNIC, dtype: object
0    1100015
1    1100023
2    1100031
3    1100049
4    1100056
Name: COD. UF + COD. MUNIC, dtype: object

```

Neste trecho, são definidas duas listas, colunas\_df\_populacao e colunas\_df\_renda, que especificam quais colunas serão mantidas nos DataFrames após a junção.

A lista colunas\_df\_populacao contém as colunas desejadas do DataFrame df\_populacao, enquanto colunas\_df\_renda é criada a partir das colunas do DataFrame df\_renda.

```

# Seleciona as colunas desejadas em cada DataFrame
colunas_df_populacao = ['UF', 'NOME DO MUNICÍPIO', 'POPULAÇÃO ESTIMADA']
colunas_df_renda = df_renda.columns.tolist()

```

Neste trecho, os DataFrames df\_renda e df\_populacao são unidos (juntados) usando a função pd.merge(). A junção é realizada usando as colunas 'Código do Município' do DataFrame df\_renda e 'COD. UF + COD. MUNIC' do DataFrame df\_populacao como chaves. O tipo de junção é 'inner', o que significa que apenas as linhas que têm correspondência em ambas as chaves são incluídas no DataFrame resultante.

O resultado final é armazenado no DataFrame df\_dados\_demograficos, que contém apenas as colunas especificadas nas listas colunas\_df\_renda e colunas\_df\_populacao.

```

# Realiza a junção entre os DataFrames usando a coluna temporária como chave
df_dados_demograficos = pd.merge(df_renda, df_populacao, left_on='Código do Município', right_on='COD. UF + COD. MUNIC', how='inner')[colunas_df_renda + colunas_df_populacao]

```

Neste trecho abaixo, são exibidas informações sobre o DataFrame resultante, incluindo o número de registros, o nome das colunas e os tipos de dados de cada coluna e as primeiras 5 linhas do DataFrame df\_dados\_demograficos são impressas na saída. Isso permite que o usuário veja uma amostra dos dados após o processamento e junção.

```

print("Informações sobre o df_dados_demograficos:")
print(df_dados_demograficos.info())

# Imprime as 10 primeiras linhas do DataFrame resultante
print("As 5 Primeiras Linhas do DataFrame Combinado:")
print(df_dados_demograficos.head(5))

print('*' * 100)
print('FIM df_dados_demograficos = df_renda + df_populacao')
print('*' * 100)

```

Saídas do código acima, com destaque na quantidade de linhas (5570) de df\_dados\_demograficos:

```

Informações sobre o df_dados_demograficos:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 12 columns):
 #                                         Column
Non-Null Count Dtype   Column Name
---  -----
0      non-null   object  Nome_da_Grande_Região
1      non-null   object  Código
2      non-null   float64 Município
3      non-null   float64 (R$ 1.000)
4      non-null   float64 (R$ 1.000)
5      non-null   float64 (R$ 1.000)
6      non-null   float64 (R$ 1.000)
7      non-null   float64 (R$ 1.000)
8      non-null   float64 (R$ 1,00)
9      non-null   object  UF
10     non-null  int64   MUNICÍPIO
11     non-null  int64   ESTIMADA
dtypes: float64(7), int64(1), object(4)
memory usage: 565.7+ KB
None

As 5 Primeiras Linhas do DataFrame Combinado:
Nome_da_Grande_Região Código do Município \
0          Norte      1100015
1          Norte      1100023
2          Norte      1100031
3          Norte      1100049

```

4 Norte 110005

	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000) \
0	203394.4190
1	199722.7100
2	81177.1010
3	236214.9520
4	94757.6390

	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000) \
0	20716.3980
1	404751.7570
2	5438.0330
3	275536.8230
4	23582.3300

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \

0	150192.4340
1	1207405.0680
2	28666.9660
3	1157343.9830
4	276755.1060

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

0	160859.6300
1	710513.4070
2	44671.0740
3	575806.3910
4	115651.8540

	Valor adicionado bruto total, a preços correntes (R\$ 1.000) \
0	535162.8810
1	2522392.9430
2	159953.1750
3	2244902.1480
4	510746.9290

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \

0	35109.4450
1	295655.6750
2	7236.6360
3	274450.5290
4	89923.1460

	Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)	UF	\
0	25091.1800	RO	
1	25730.2000	RO	
2	32226.2500	RO	
3	29331.2900	RO	
4	37069.2500	RO	

NOME DO MUNICÍPIO POPULAÇÃO ESTIMADA

0	Alta Floresta D'Oeste	22728
1	Ariquemes	109523
2	Cabixi	5188
3	Cacoal	85893
4	Cerejeiras	16204

```
*****
FIM df_dados_demograficos = df_renda + df_populacao
*****
```

Uma lista chamada nova\_ordem\_colunas é criada para definir a ordem desejada das colunas no DataFrame df\_dados\_demográficos.

Cada elemento da lista corresponde ao nome de uma coluna que deve aparecer no DataFrame resultante, e a ordem dos elementos determina a ordem das colunas.

Essa etapa é útil quando se deseja reorganizar as colunas de um DataFrame de acordo com uma ordem específica. Após definir a ordem desejada das colunas na lista nova\_ordem\_colunas, o código reordena as colunas do DataFrame df\_dados\_demográficos com base nessa ordem.

```
print('*' * 100)
print('Início df_dados_demograficos completo')
print('*' * 100)

# Define a ordem desejada das colunas em uma lista
nova_ordem_colunas = ['NOME DO MUNICÍPIO',
                      'Código do Município',
                      'UF',
                      'Nome_da_Grande_Região',
                      'POPULAÇÃO ESTIMADA',
                      'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
                      'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
                      'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
                      'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
                      'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
                      'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
                      'Produto Interno Bruto per capita, a preços correntes (R$ 1.000)']

# Reordena as colunas do DataFrame
df_dados_demograficos = df_dados_demograficos[nova_ordem_colunas]
```

Este trecho imprime os nomes das colunas do DataFrame após a reordenação para que o usuário possa verificar se a reordenação foi bem-sucedida. código imprime informações sobre o DataFrame df\_dados\_demograficos. As informações incluem detalhes sobre as colunas, tipos de dados, quantidade de valores não nulos, uso de memória, entre outros.

```
df_dados_demograficos = df_dados_demograficos[nova_ordem_colunas]
print('Nomes das colunas após reordenamento:')
print(df_dados_demograficos.columns)

# Imprime informações sobre o DataFrame
print('Informações sobre o DataFrame:')
print(df_dados_demograficos.info())

print('*' * 100)
print('FIM df_dados_demograficos completo')
print('*' * 100)
```

Saídas do código acima, com destaque na quantidade de linhas (5570) e a nova ordem das colunas na versão final de df\_dados\_demograficos:

```
*****
Início df_dados_demograficos completo
*****
Nomes das colunas após reordenamento:
Index(['NOME DO MUNICÍPIO', 'Código do Município', 'UF',
       'Nome_da_Grande_Região', 'POPULAÇÃO ESTIMADA',
       'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
       'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
       'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
       'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
       'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
       'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
```

```

'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'],
dtype='object')
Informações sobre o DataFrame:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 12 columns):
 #                                         Column
Non-Null Count Dtype   Column
---  -----
0                           NOME
5570 non-null  object  MUNICÍPIO
1                           Código  Município
5570 non-null  object  UF
2                           Nome_da_Grande_Região
5570 non-null  object  POPULAÇÃO
3                           ESTIMADA
4                           (R$ 1.000)
5      Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
5570 non-null  float64
6      Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
5570 non-null  float64
7  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000) 5570 non-null  float64
8  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000) 5570 non-null  float64
9      Valor adicionado bruto total, a preços correntes (R$ 1.000)
5570 non-null  float64
10     Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
5570 non-null  float64
11     Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
5570 non-null  float64
dtypes: float64(7), int64(1), object(4)
memory usage: 565.7+ KB
None
*****
FIM df_dados_demograficos completo
*****

```

### 3.7 Junção de Candidatos – 2020 (Eleitos) e Municípios Brasileiros TSE

Neste trecho, são selecionadas as colunas relevantes dos DataFrames `df_eleitos` e `df_TSE_IBGE` e armazenadas em `colunas_df_eleitos` e `colunas_df_TSE_IBGE`, respectivamente. Isso é importante para reduzir a quantidade de dados a serem analisados, focando apenas nas informações necessárias.

```

print('*' * 100)
print('Início df_eleitos')
print('*' * 100)

# Selecionando as colunas desejadas para o DataFrame df_eleitos
colunas_df_eleitos = [
    'DS_CARGO',
    'TP_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'ST_REELEICAO',
    'VR_DESPESA_MAX_CAMPANHA']

# Selecionando as colunas desejadas para o DataFrame df_TSE_IBGE
colunas_df_TSE_IBGE = ['capital', 'codigo_ibge']

```

O código então realiza uma junção (merge) entre os DataFrames `df_eleitos` e `df_TSE_IBGE` usando as colunas 'SG\_UE' e 'codigo\_tse' como chave de junção (`left_on='SG_UE'`, `right_on='codigo_tse'`).

É especificado que a junção deve ser "inner", o que significa que apenas as linhas com chaves correspondentes em ambos os DataFrames serão mantidas.

O resultado da junção é armazenado no DataFrame `df_eleitos_IBGE`, contendo apenas as colunas selecionadas previamente.

```

# Realizando a junção entre os DataFrames usando a coluna 'SG_UE' e 'codigo_tse' como chave
df_eleitos_IBGE = pd.merge(df_eleitos, df_TSE_IBGE, left_on='SG_UE', right_on='codigo_tse',
                           how='inner')[colunas_df_eleitos + colunas_df_TSE_IBGE]

```

É mencionado que campos preenchidos com #NULO significam que a informação está em branco no banco de dados. E para campos numéricos, o correspondente para #NULO é -1.

Os dados são então separados em duas categorias com base no valor da coluna 'VR\_DESPESA\_MAX\_CAMPANHA':

**valores\_positivos:** Registros em que 'VR\_DESPESA\_MAX\_CAMPANHA' é maior ou igual a 0.

**valores\_negativos:** Registros em que 'VR\_DESPESA\_MAX\_CAMPANHA' é menor que 0.

A contagem de valores em cada categoria é realizada e armazenada em `contagem_positivos` e `contagem_negativos`.

Um gráfico de barras é criado para visualizar a contagem de valores positivos e negativos. As categorias são definidas como 'Valores >= 0' e 'Valores < 0', e as contagens são usadas como alturas das barras.

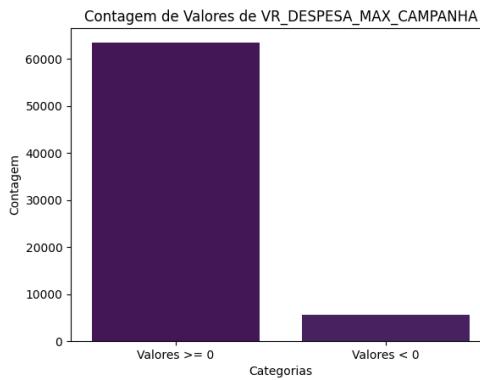
```
# Campos preenchidos com #NULL significam que a informação está em branco no banco de dados.
# O correspondente para #NULL nos campos numéricos é -1;

# Separando os dados em duas categorias: >= 0 e < 0
valores_positivos = df_eleitos_IBGE[df_eleitos_IBGE['VR_DESPESA_MAX_CAMPANHA'] >= 0]
valores_negativos = df_eleitos_IBGE[df_eleitos_IBGE['VR_DESPESA_MAX_CAMPANHA'] < 0]

# Contando os valores em cada categoria
contagem_positivos = len(valores_positivos)
contagem_negativos = len(valores_negativos)

# Criando um gráfico de barras para visualizar a contagem
categorias = ['Valores >= 0', 'Valores < 0']
contagem = [contagem_positivos, contagem_negativos]

plt.bar(categorias, contagem)
plt.xlabel('Categorias')
plt.ylabel('Contagem')
plt.title('Contagem de Valores de VR_DESPESA_MAX_CAMPANHA')
plt.show()
```



É filtrado o DataFrame `df_eleitos_IBGE` para remover os registros em que '`VR_DESPESA_MAX_CAMPANHA`' é menor que 0. Isso significa que os registros com valores negativos são descartados.

É feita uma nova filtragem no DataFrame `df_eleitos_vices`, que contém apenas os dados relacionados aos vice-prefeitos, tanto antes quanto depois da remoção de valores negativos.

```
# Filtrando apenas os dados relacionados aos vice-prefeitos antes da remoção de valores negativos
print("DataFrame df_eleitos_vices antes da remoção de valores negativos:")
df_eleitos_vices = df_eleitos_IBGE[df_eleitos_IBGE['DS_CARGO'] == 'VICE-PREFEITO']
print(df_eleitos_vices.info())

# Removendo valores negativos do DataFrame df_eleitos com IBGE
df_eleitos_IBGE = df_eleitos_IBGE[df_eleitos_IBGE['VR_DESPESA_MAX_CAMPANHA'] >= 0]

# Filtrando apenas os dados relacionados aos vice-prefeitos após a remoção de valores negativos
print("DataFrame df_eleitos_vices depois da remoção de valores negativos:")
df_eleitos_vices = df_eleitos_IBGE[df_eleitos_IBGE['DS_CARGO'] == 'VICE-PREFEITO']
print(df_eleitos_vices.info())
```

Saídas do código acima, com destaque na quantidade de registros antes e depois da remoção, contatando que todos os vice-prefeitos foram eliminados:

```
*****
Inicio df_eleitos
*****
DataFrame df_eleitos_vices antes da remoção de valores negativos:
<class 'pandas.core.frame.DataFrame'>
```

```

Int64Index: 5587 entries, 4 to 69144
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DS_CARGO         5587 non-null    object  
 1   TP AGREMIACAO   5587 non-null    object  
 2   SG_PARTIDO       5587 non-null    object  
 3   NM_PARTIDO       5587 non-null    object  
 4   ST_REELEICAO    5587 non-null    object  
 5   VR_DESPESA_MAX_CAMPANHA  5587 non-null    float64
 6   capital          5587 non-null    int64   
 7   codigo_ibge      5587 non-null    object  
dtypes: float64(1), int64(1), object(6)
memory usage: 392.8+ KB
None
DataFrame df_eleitos_vices depois da remoção de valores negativos:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DS_CARGO         0 non-null     object  
 1   TP AGREMIACAO   0 non-null     object  
 2   SG_PARTIDO       0 non-null     object  
 3   NM_PARTIDO       0 non-null     object  
 4   ST_REELEICAO    0 non-null     object  
 5   VR_DESPESA_MAX_CAMPANHA  0 non-null    float64
 6   capital          0 non-null     int64   
 7   codigo_ibge      0 non-null     object  
dtypes: float64(1), int64(1), object(6)
memory usage: 0.0+ bytes
None

```

São impressas informações sobre o DataFrame `df_eleitos_IBGE` após as transformações, incluindo informações sobre as colunas e os tipos de dados presentes no DataFrame.

```

# Exibindo informações sobre o DataFrame df_eleitos_IBGE
print("Informações sobre o DataFrame df_eleitos_IBGE:")
print(df_eleitos_IBGE.info())

print('*' * 100)
print('FIM df_eleitos')
print('*' * 100)

```

Saídas do código acima, com a versão final de `df_eleitos_IBGE`:

```

Informações sobre o DataFrame df_eleitos_IBGE:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63448 entries, 0 to 69145
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DS_CARGO         63448 non-null    object  
 1   TP AGREMIACAO   63448 non-null    object  
 2   SG_PARTIDO       63448 non-null    object  
 3   NM_PARTIDO       63448 non-null    object  

```

```

4    ST_REELEICAO           63448 non-null  object
5    VR_DESPESA_MAX_CAMPANHA 63448 non-null  float64
6    capital                63448 non-null  int64
7    codigo_ibge             63448 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 4.4+ MB
None
*****
FIM df_eleitos
*****

```

### 3.8 Junção de Dados dos Eleitos com Dados Demográficos

Nesse trecho, são obtidas as listas de colunas dos DataFrames `df_eleitos_IBGE` e `df_dados_demograficos`.

Essas listas serão usadas posteriormente para manipular as colunas dos DataFrames. Os DataFrames `df_eleitos_IBGE` e `df_dados_demograficos` são mesclados com base nas colunas "codigo\_ibge" e "Código do Município", respectivamente.

O resultado é um novo DataFrame chamado `df_eleitos_dados_demograficos`, que contém as colunas de ambos os DataFrames originais.

```

# Obtém as listas de colunas dos DataFrames
colunas_df_eleitos_IBGE = df_eleitos_IBGE.columns.tolist()
colunas_df_dados_demograficos = df_dados_demograficos.columns.tolist()

# Realiza a junção entre os DataFrames
df_eleitos_dados_demograficos = pd.merge(df_eleitos_IBGE, df_dados_demograficos, left_on='codigo_ibge',
                                           right_on='Código do Município',
                                           how='inner')[colunas_df_eleitos_IBGE + colunas_df_dados_demograficos]

```

Aqui, a coluna "codigo\_ibge" é removida do DataFrame resultante, uma vez que já foi utilizada na junção dos DataFrames.

```

# Remove a coluna 'codigo_ibge'
df_eleitos_dados_demograficos = df_eleitos_dados_demograficos.drop(columns='codigo_ibge')

```

#### 3.8.1 Conversão de Tipos de Dados e Reordenação de Colunas

Neste trecho, as colunas que representam variáveis categóricas são convertidas para o tipo de dados "category", enquanto as colunas que representam variáveis numéricas são convertidas para o tipo numérico usando `pd.to_numeric`.

O argumento `errors='coerce'` permite que valores não numéricos sejam convertidos para NaN (caso não possam ser convertidos para números).

As colunas do DataFrame são reordenadas de acordo com a lista `nova_ordem_colunas`.

```

# Lista de colunas para definir como categóricas
colunas_categoricas = [
    'DS_CARGO',
    'TP_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'ST_REELEICAO',
    'capital',
    'UF',
    'Nome_da_Grande_Região'
]

# Define as colunas como categóricas
df_eleitos_dados_demograficos[colunas_categoricas] = df_eleitos_dados_demograficos[colunas_categoricas].astype('category')

# Lista de colunas para definir como numéricas
colunas_numericas = [
    'VR_DESPESA_MAX_CAMPANHA',
    'POPULAÇÃO ESTIMADA',
    'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
    'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
    'Produto Interno Bruto per capita, a preços correntes (R$ 1.000)'
]

# Define a ordem desejada das colunas
nova_ordem_colunas = [
    'NOME DO MUNICÍPIO',
    'Código do Município',
    'UF',
    'Nome_da_Grande_Região',
    'POPULAÇÃO ESTIMADA',
    'capital',
    'DS_CARGO',
    'VR_DESPESA_MAX_CAMPANHA',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'TP_AGREMIACAO',
    'ST_REELEICAO',
    'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
    'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
    'Produto Interno Bruto per capita, a preços correntes (R$ 1.000)'
]

# Reordena as colunas do DataFrame
df_eleitos_dados_demograficos = df_eleitos_dados_demograficos[nova_ordem_colunas]

# Reordena as colunas do DataFrame
df_eleitos_dados_demograficos = df_eleitos_dados_demograficos[nova_ordem_colunas]

# Usa pd.to_numeric para definir as colunas como numéricas
df_eleitos_dados_demograficos[colunas_numericas] = df_eleitos_dados_demograficos[colunas_numericas].apply(pd.to_numeric,
                                                       errors='coerce')

```

Informações sobre o DataFrame são impressas, incluindo informações sobre tipos de dados e estatísticas resumidas.

É exibida também uma amostra aleatória de 5 linhas do df\_eleitos\_dados\_demograficos para realizar uma análise exploratória inicial.

```

# Imprime informações sobre o DataFrame após a definição dos tipos de colunas
print("Informações sobre o df_eleitos_dados_demograficos após a definição dos tipos das colunas:")
print(df_eleitos_dados_demograficos.info())

# Verifica os tipos de dados das colunas após a conversão
print("Tipos de dados das colunas após a conversão:")
print(df_eleitos_dados_demograficos.dtypes)

## Realiza uma análise exploratória básica
## Exibe algumas linhas aleatórias do DataFrame
print("Amostra aleatória do df_eleitos_dados_demograficos:")
print(df_eleitos_dados_demograficos.sample(5))

print('*' * 100)
print('FIM df_eleitos_dados_demograficos = df_eleitos_IBGE + df_dados_demograficos')
print('*' * 100)

print('*' * 100)
print('FIM Coleta de Dados - Processamento/Tratamento de Dados')
print('*' * 100)

```

Saídas do código acima, com a versão final do df\_eleitos\_dados\_demograficos, o data frame que será realizado a análise e exploração dos dados a seguir:

```

*****+
INICIO df_eleitos_dados_demograficos = df_eleitos_IBGE + df_dados_demograficos
*****+
Informações sobre o df_eleitos_dados_demograficos após a definição dos tipos das colunas:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63448 entries, 18569 to 32662
Data columns (total 19 columns):
 #   Column           Non-Null Count Dtype  
 ---  --  
 0   NOME DO MUNICÍPIO    63448 non-null  object  
 1   Código do Município  63448 non-null  object  
 2   UF                 63448 non-null  category 
 3   Nome_da_Grande_Região 63448 non-null  category 
 4   POPULAÇÃO ESTIMADA   63448 non-null  int64  
 5   capital             63448 non-null  category 
 6   DS_CARGO            63448 non-null  category 
 7   VR_DESPESA_MAX_CAMPANHA 63448 non-null  float64 
 8   SG_PARTIDO           63448 non-null  category 
 9   NM_PARTIDO           63448 non-null  category 
 10  TP_AGRREMICACAO     63448 non-null  category 
 11  ST_REELEICAO         63448 non-null  category 
 12  Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) 63448 non-null  float64 
 13  Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) 63448 non-null  float64 
 14  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000) 63448 non-null  float64 
 15  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000) 63448 non-null  float64 
 16  Valor adicionado bruto total, a preços correntes (R$ 1.000) 63448 non-null  float64 
 17  Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) 63448 non-null  float64 
 18  Produto Interno Bruto per capita, a preços correntes (R$ 1.000) 63448 non-null  float64 
dtypes: category(8), float64(8), int64(1), object(2)
memory usage: 6.3+ MB
None

None
Tipos de dados das colunas após a conversão:
NOME DO MUNICÍPIO          object  
Código do Município        object  
UF                          category 
Nome_da_Grande_Região      category 
POPULAÇÃO ESTIMADA         int64  
capital                     category 
DS_CARGO                   category 
VR_DESPESA_MAX_CAMPANHA   float64 
SG_PARTIDO                  category 
NM_PARTIDO                  category 
TP_AGRREMICACAO            category 
ST_REELEICAO                category 
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) float64 
Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) float64 
Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000) float64 
Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000) float64 
Valor adicionado bruto total, a preços correntes (R$ 1.000) float64 
Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) float64 
Produto Interno Bruto per capita, a preços correntes (R$ 1.000) float64 
dtype: object

```

Amostra aleatória do df\_eleitos\_dados\_demograficos:

	NOME DO MUNICÍPIO	Código do Município	UF	Nome_da_Grande_Região	\
37906	Pouso Redondo	4213708	SC	Sul	
27020	Vila Pavão	3205150	ES	Sudeste	
26278	Medeiros	3141306	MG	Sudeste	
45534	Barra do Rocha	2903102	BA	Nordeste	
28249	Flores da Cunha	4308201	RS	Sul	

	POPULAÇÃO ESTIMADA	capital	DS_CARGO	VR_DESPESA_MAX_CAMPANHA	\
37906	17712	0	VEREADOR	12307.7500	
27020	9244	0	VEREADOR	20446.3900	
26278	3832	0	VEREADOR	12307.7500	
45534	5612	0	PREFEITO	123077.4200	
28249	31063	0	VEREADOR	12307.7500	

	SG_PARTIDO		NM_PARTIDO	TP_AGREMIACAO	\
37906	PSDB	PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA	PARTIDO ISOLADO		
27020	DEM		DEMOCRATAS	PARTIDO ISOLADO	
26278	AVANTE		AVANTE	PARTIDO ISOLADO	
45534	PDT	PARTIDO DEMOCRÁTICO TRABALHISTA	COLIGAÇÃO		
28249	PP		PROGRESSISTAS	PARTIDO ISOLADO	

	ST_REELEICAO	\
37906	N	
27020	S	
26278	N	
45534	N	
28249	S	

	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)	\
37906	68044.0520	
27020	32130.9850	
26278	108380.5100	
45534	21037.3690	
28249	113372.4830	

	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)	\
37906	204286.2630	
27020	22359.7110	
26278	8041.7360	
45534	3074.3650	
28249	709892.3060	

	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)	\
37906	228279.3630	
27020	40168.9430	
26278	33735.0180	
45534	13932.3600	
28249	587793.6480	

	Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)	\
37906	81032.0010	
27020	47763.3220	
26278	22869.4480	

45534	28027.9090
28249	167995.1570

Valor adicionado bruto total, a preços correntes (R\$ 1.000) \

37906	581641.6790
27020	142422.9600
26278	173026.7120
45534	66072.0020
28249	1579053.5940

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \

37906	62293.2800
27020	7895.5900
26278	5487.7310
45534	1685.8800
28249	369899.7670

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

37906	36355.8600
27020	16261.2000
26278	46585.1900
45534	12073.7500
28249	62741.9600

\*\*\*\*\*  
FIM df\_eleitos\_dados\_demograficos = df\_eleitos\_IBGE + df\_dados\_demograficos  
\*\*\*\*\*

## 4 Análise e Exploração dos Dados

### 4.1 Separação dos Dados em DataFrames Diferentes

Aqui, os dados são divididos em dois DataFrames diferentes, df\_eleitos\_prefeitos e df\_eleitos\_vereadores, com base no valor da coluna 'DS\_CARGO'. Essa separação permite analisar separadamente os prefeitos e vereadores.

```
# Criar DataFrames df_eleitos_prefeitos e df_eleitos_vereadores separados com base na coluna 'DS_CARGO'
df_eleitos_prefeitos = df_eleitos_dados_demograficos[df_eleitos_dados_demograficos['DS_CARGO'] == 'PREFEITO']
df_eleitos_vereadores = df_eleitos_dados_demograficos[df_eleitos_dados_demograficos['DS_CARGO'] == 'VEREADOR']
```

### 4.2 Exibição do Tamanho e Informações dos DataFrames

Esses comandos abaixo imprimem o número de linhas e colunas dos DataFrames df\_eleitos\_prefeitos e df\_eleitos\_vereadores.

Aqui, são exibidas informações detalhadas sobre os DataFrames, incluindo o tipo de dado de cada coluna, a quantidade de valores não nulos e o uso de memória.

```
# Exibir o tamanho dos DataFrames
print("Tamanho de df_eleitos_prefeitos:", df_eleitos_prefeitos.shape)

print("Tamanho de df_eleitos_vereadores:", df_eleitos_vereadores.shape)

# Exibir informações sobre os DataFrames
print("\nInformações sobre df_prefeitos:")
print(df_eleitos_prefeitos.info())
print("\nInformações sobre df_vereadores:")
print(df_eleitos_vereadores.info())

*****  

INICIO Análise e Exploração dos Dados - Criação de Modelos de Machine Learning  

*****  

Tamanho de df_eleitos_prefeitos: (5492, 19)  

Tamanho de df_eleitos_vereadores: (57956, 19)

Informações sobre df_prefeitos:  

<class 'pandas.core.frame.DataFrame'>  

Int64Index: 5492 entries, 18569 to 32667  

Data columns (total 19 columns):  

 #   Column          Non-Null Count  Dtype     

---  --  

 0   NOME DO MUNICÍPIO      5492 non-null   object    

 1   Código do Município    5492 non-null   object    

 2   UF                   5492 non-null   category    

 3   Nome_da_Grande_Região 5492 non-null   category    

 4   POPULAÇÃO ESTIMADA    5492 non-null   int64     

 5   capital              5492 non-null   category    

 6   DS_CARGO              5492 non-null   category    

 7   VR_DESPESA_MAX_CAMPANHA 5492 non-null   float64   

 8   SG_PARTIDO             5492 non-null   category    

 9   NM_PARTIDO             5492 non-null   category    

 10  TP_AGRIMIACAO         5492 non-null   category    

 11  ST_REELEICAO           5492 non-null   category    

 12  Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) 5492 non-null   float64   

 13  Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) 5492 non-null   float64   

 14  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000) 5492 non-null   float64   

 15  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000) 5492 non-null   float64   

 16  Valor adicionado bruto total, a preços correntes (R$ 1.000) 5492 non-null   float64   

 17  Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) 5492 non-null   float64   

 18  Produto Interno Bruto per capita, a preços correntes (R$ 1.000) 5492 non-null   float64  

dtypes: category(8), float64(8), int64(1), object(2)  

memory usage: 562.3+ KB  

None
```

```

Informações sobre df_vereadores:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57956 entries, 18571 to 32662
Data columns (total 19 columns):
 #   Column          Non-Null Count Dtype  
0   NOME DO MUNICÍPIO      57956 non-null  object  
1   Código do Município    57956 non-null  object  
2   UF                   57956 non-null  category 
3   Nome_da_Grande_Região 57956 non-null  category 
4   POPULAÇÃO ESTIMADA    57956 non-null  int64   
5   capital              57956 non-null  category 
6   DS_CARGO              57956 non-null  category 
7   VR_DESPESA_MAX_CAMPANHA 57956 non-null  float64 
8   SG_PARTIDO             57956 non-null  category 
9   NM_PARTIDO             57956 non-null  category 
10  TP_AGRÉMIAÇÃO          57956 non-null  category 
11  ST_Reeleicao            57956 non-null  category 
12  Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) 57956 non-null  float64 
13  Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)    57956 non-null  float64 
14  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000) 57956 non-null  float64 
15  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000)           57956 non-null  float64 
16  Valor adicionado bruto total, a preços correntes (R$ 1.000)             57956 non-null  float64 
17  Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) 57956 non-null  float64 
18  Produto Interno Bruto per capita, a preços correntes (R$ 1,00)         57956 non-null  float64 
dtypes: category(8), float64(8), int64(1), object(2)
memory usage: 5.8+ MB
None

```

### 4.3 Análise de Variáveis Categóricas

Neste ponto, são identificadas as colunas do DataFrame df\_eleitos\_dados\_demograficos que contêm dados categóricos. É feita uma contagem para cada valor distinto de cada coluna encontrada:

```

#Variáveis Categóricas:
# Identificar e imprimir as variáveis categóricas
categorical_columns = df_eleitos_dados_demograficos.select_dtypes(include='category').columns
print("Variáveis Categóricas: ")
for col in categorical_columns:
    print(f"Coluna: {col}")
    print(df_eleitos_dados_demograficos[col].value_counts())
    print("\n")

```

Saídas do código acima:

Variáveis Categóricas:

Coluna: UF

MG 9322

SP 7585

RS 5382

BA 4971

PR 4264

SC 3184

GO 2762

MA 2642

PB 2428

CE 2371

PI 2366

PE 2330

PA 1906

RN 1819

MT 1477

TO 1446

RJ 1272

AL 1187

ES 936

MS 921

SE 878

AM 799

```

RO      585
AC      253
AP      190
RR      172
Name: UF, dtype: int64

```

```

Coluna: Nome_da_Grande_Região
Nordeste      20992
Sudeste       19115
Sul          12830
Norte         5351
Centro-oeste  5160
Name: Nome_da_Grande_Região, dtype: int64

```

```

Coluna: capital
0      62607
1      841
Name: capital, dtype: int64

```

```

Coluna: DS_CARGO
VEREADOR      57956
PREFEITO      5492
Name: DS_CARGO, dtype: int64

```

```

Coluna: SG_PARTIDO
MDB           8137
PP            7032
PSD           6338
PSDB          4911
DEM           4789
PL             3799
PDT           3748
PSB           3257
PT             2839
REPUBLICANOS  2802
PTB           2675
CIDADANIA     1732
PODE          1634
PSC           1617
SOLIDARIEDADE 1442
PSL           1300
AVANTE        1125
PV             853
PROS          802
PATRIOTA       769
PC do B        741
PRTB          222
PTC           219
PMN           213
REDE          152
DC             125
PSOL          96
PMB           49
NOVO          30
Name: SG_PARTIDO, dtype: int64

```

Coluna: NM\_PARTIDO

MOVIMENTO DEMOCRÁTICO BRASILEIRO	8137
PROGRESSISTAS	7032
PARTIDO SOCIAL DEMOCRÁTICO	6338
PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA	4911
DEMOCRATAS	4789
PARTIDO LIBERAL	3799
PARTIDO DEMOCRÁTICO TRABALHISTA	3748
PARTIDO SOCIALISTA BRASILEIRO	3257
PARTIDO DOS TRABALHADORES	2839
REPUBLICANOS	2802
PARTIDO TRABALHISTA BRASILEIRO	2675
CIDADANIA	1732
PODEMOS	1634
PARTIDO SOCIAL CRISTÃO	1617
SOLIDARIEDADE	1442
PARTIDO SOCIAL LIBERAL	1300
AVANTE	1125
PARTIDO VERDE	853
PARTIDO REPUBLICANO DA ORDEM SOCIAL	802
PATRIOTA	769
PARTIDO COMUNISTA DO BRASIL	741
PARTIDO RENOVADOR TRABALHISTA BRASILEIRO	222
PARTIDO TRABALHISTA CRISTÃO	219
PARTIDO DA MOBILIZAÇÃO NACIONAL	213
REDE SUSTENTABILIDADE	152
DEMOCRACIA CRISTÃ	125
PARTIDO SOCIALISMO E LIBERDADE	96
PARTIDO DA MULHER BRASILEIRA	49
PARTIDO NOVO	30

Name: NM\_PARTIDO, dtype: int64

Coluna: TP\_AGREMIACAO

PARTIDO ISOLADO	58503
COLIGAÇÃO	4945

Name: TP\_AGREMIACAO, dtype: int64

Coluna: ST\_REELEICAO

N	51777
S	11671

Name: ST\_REELEICAO, dtype: int64

A seguir, será feita uma análise Variável Categórica | Hipótese | Padrões e Insights para as informações apresentadas acima, e, em seguida, uma visão geral:

#### Variável Categórica: UF (Unidade da Federação):

Hipótese: Essa variável representa os estados brasileiros.

Padrões e Insights: Minas Gerais (MG) possui o maior número de observações, seguido por São Paulo (SP) e Rio Grande do Sul (RS). Isso pode indicar que a amostra de dados tem uma forte representação desses estados. É importante levar em consideração o tamanho da população de cada estado ao interpretar esses números.

**Variável Categórica: Nome\_da\_Grande\_Região:**

Hipótese: Essa variável indica a região geográfica do Brasil.

Padrões e Insights: A região Nordeste tem o maior número de observações, seguida pela região Sudeste e Sul. Isso pode sugerir que a amostra de dados está desbalanceada em relação às regiões brasileiras.

**Variável Categórica: capital:**

Hipótese: Esta variável binária indica se a cidade onde ocorreu a eleição é uma capital (1) ou não (0).

Padrões e Insights: A grande maioria das eleições ocorreu em cidades que não são capitais (0). Isso pode ser esperado, pois o Brasil tem muitas cidades que não são capitais.

**Variável Categórica: DS\_CARGO:**

Hipótese: Essa variável indica o cargo para o qual a eleição ocorreu.

Padrões e Insights: A maioria das eleições foi para o cargo de VEREADOR, com um número significativamente menor de eleições para PREFEITO. Isso é esperado, pois em uma eleição municipal, geralmente há mais candidatos a vereador do que a prefeito.

**Variável Categórica: SG\_PARTIDO:**

Hipótese: Essa variável representa os partidos políticos.

Padrões e Insights: O MDB (Movimento Democrático Brasileiro) é o partido com o maior número de observações, seguido pelo PP (Progressistas) e PSD (Partido Social Democrático). Essa distribuição pode refletir a popularidade ou a representatividade desses partidos em relação a outros nas eleições.

**Variável Categórica: NM\_PARTIDO:**

Hipótese: Essa variável também representa os partidos políticos, mas com os nomes completos dos partidos.

Padrões e Insights: Os padrões observados aqui são semelhantes aos da variável anterior, uma vez que ambas representam os mesmos dados, apenas em diferentes formatos.

**Variável Categórica: TP\_AGREMIACAO:**

Hipótese: Esta variável indica o tipo de agremiação partidária.

Padrões e Insights: A maioria das eleições envolveu partidos isolados (PARTIDO ISOLADO) em vez de coligações (COLIGAÇÃO). Isso pode refletir a estratégia de alguns partidos de concorrerem sozinhos em eleições municipais.

**Variável Categórica: ST\_REEELEICAO:**

Hipótese: Esta variável indica se o candidato estava concorrendo à reeleição (S) ou não (N).

Padrões e Insights: A maioria dos candidatos não estava concorrendo à reeleição (N), o que é consistente com o fato de que muitos cargos nas eleições municipais não permitem reeleição.

**Visão Geral da Análise:**

Os dados parecem estar desbalanceados em relação às unidades da federação e às regiões geográficas, com uma concentração significativa de observações em alguns estados e regiões.

A maioria das eleições é para o cargo de vereador, envolvendo partidos isolados.

A maioria dos candidatos não estava concorrendo à reeleição.

Há uma variedade de partidos políticos representados, com alguns deles tendo mais observações do que outros.

```
# Realizar análise ANOVA nas colunas categóricas
print("Análise ANOVA para df_eleitos_dados_demograficos (colunas categóricas)")
analisar_ANOVA(df_eleitos_dados_demograficos,colunas_categoricas)

print("Análise ANOVA para df_prefeitos (colunas categóricas)")
analisar_ANOVA(df_eleitos_prefeitos,colunas_categoricas)

print("Análise ANOVA para df_vereadores (colunas categóricas)")
analisar_ANOVA(df_eleitos_vereadores,colunas_categoricas)
```

O código acima faz três chamadas da função analisar\_ANOVA para realizar a análise ANOVA em diferentes conjuntos de dados, representados pelos DataFrames df\_eleitos\_dados\_demograficos, df\_eleitos\_prefeitos e df\_eleitos\_vereadores, todos usando a mesma lista de colunas\_categoricas.

```
def analisar_ANOVA(df,colunas_categoricas):
    # Criar uma fórmula para a ANOVA
    formula_anova = 'VR_DESPESA_MAX_CAMPANHA ~ ' + ' + '.join(['C(' + coluna + ')' for coluna in colunas_categoricas])
    modelo_anova = ols(formula_anova, data=df).fit()

    tabela_anova = sm.stats.anova_lm(modelo_anova, typ=2)

    # Exibir os resultados da ANOVA
    print(f"Resultados da Análise de Variância (ANOVA) para as colunas: {', '.join(colunas_categoricas)}")
    print(tabela_anova)
```

A análise ANOVA é uma técnica estatística comumente usada para determinar se existem diferenças significativas entre grupos de dados categóricos em relação a uma variável numérica, como os gastos de campanha neste caso. Essa análise pode ajudar a identificar padrões e insights relacionados às eleições e aos gastos de campanha. Os resultados da ANOVA são frequentemente usados para validar ou refutar hipóteses sobre como as variáveis categóricas afetam a variável dependente.

Esta função recebe dois parâmetros: df (um DataFrame) e colunas\_categoricas (uma lista de nomes de colunas categóricas).

A função realiza uma análise de variância (ANOVA) para entender como as variáveis categóricas afetam a variável dependente VR\_DESPESA\_MAX\_CAMPANHA em um modelo estatístico.

Esta função recebe dois parâmetros: df (um DataFrame) e colunas\_categoricas (uma lista de nomes de colunas categóricas).

A função realiza uma análise de variância (ANOVA) para entender como as variáveis categóricas afetam a variável dependente VR\_DESPESA\_MAX\_CAMPANHA em um modelo estatístico.

### **Construção da Fórmula da ANOVA:**

A função cria uma fórmula para a ANOVA usando a variável dependente VR\_DESPESA\_MAX\_CAMPANHA e todas as variáveis categóricas presentes na lista colunas\_categoricas. Isso é feito através da concatenação de strings.

A fórmula criada será usada para ajustar o modelo estatístico.

### **Ajuste do Modelo ANOVA:**

Utilizando a fórmula criada, a função ajusta um modelo de análise de variância (ANOVA) aos dados do DataFrame df usando a biblioteca estatística statsmodels. O modelo é ajustado usando o método das Mínimas Quadradas Ordinárias (OLS).

### Cálculo da Tabela ANOVA:

Após ajustar o modelo, a função calcula a tabela ANOVA utilizando o método sm.stats.anova\_lm. Esta tabela contém estatísticas importantes para avaliar a influência das variáveis categóricas na variável dependente.

### Impressão dos Resultados da ANOVA:

A função imprime os resultados da análise ANOVA, incluindo a tabela ANOVA, para fornecer informações sobre a relação entre as variáveis categóricas e a variável dependente.

Os resultados são exibidos na saída padrão, mostrando as estatísticas de interesse abaixo:

Análise ANOVA para df\_eleitos\_dados\_demograficos (colunas categóricas):

	sum_sq	df	F	PR(>F)
C(DS_CARGO)	10132758441105.1270	1.0000	198.0481	0.0000
C(TP_AGREMIACAO)	4994050246960.6172	1.0000	97.6103	0.0000
C(SG_PARTIDO)	95691006891.2541	28.0000	0.0668	1.0000
C(NM_PARTIDO)	95691006866.0873	28.0000	0.0668	1.0000
C(ST_REELEICAO)	45377846450.3708	1.0000	0.8869	0.3463
C(capital)	618254698413249.5000	1.0000	12083.9888	0.0000
C(UF)	49889644749.0736	25.0000	0.0388	0.9971
C(Nome_da_Grande_Região)	7950343159.7408	4.0000	0.0388	0.9971
Residual	3243179687725583.0000	63389.0000	NaN	NaN

Os resultados mostram estatísticas importantes da ANOVA, incluindo a soma dos quadrados (sum\_sq), os graus de liberdade (df), a estatística F (F), e o valor p (PR(>F)).

As variáveis DS\_CARGO, TP\_AGREMIACAO e capital têm valores de p (PR(>F)) muito baixos (próximos a 0), o que sugere que essas variáveis têm um impacto significativo nas despesas de campanha. Isso pode ser um insight importante.

Análise ANOVA para df\_prefeitos (colunas categóricas):

	sum_sq	df	F	PR(>F)
C(DS_CARGO)	7137591147470.2236	1.0000	19.9889	0.0000
C(TP_AGREMIACAO)	1159007861312.9319	1.0000	3.2475	0.0716
C(SG_PARTIDO)	95253070224708.0312	28.0000	9.5271	0.0000
C(NM_PARTIDO)	95253070224675.9219	28.0000	9.5271	0.0000
C(ST_REELEICAO)	1454791684499.3262	1.0000	4.0742	0.0436
C(capital)	828412094980748.8750	1.0000	2319.9785	0.0000
C(UF)	167700197045675.0425	25.0000	18.7859	0.0000
C(Nome_da_Grande_Região)	26832031527285.1328	4.0000	18.7859	0.0000
Residual	1940716183076238.2500	5435.0000	NaN	NaN

Nota-se que a variável C(TP\_AGREMIACAO) tem um valor de p (PR(>F)) próximo a 0.05, o que indica que ela pode ter um impacto marginalmente significativo nas despesas de campanha, embora não seja tão forte quanto as outras variáveis significativas.

Análise ANOVA para df\_vereadores (colunas categóricas):

	sum_sq	df	F	PR(>F)
C(DS_CARGO)	267123213.3903	1.0000	0.0215	0.8836
C(TP_AGREMIACAO)	210413192.7098	1.0000	0.0169	0.8966
C(SG_PARTIDO)	45577407456.1106	28.0000	0.1307	1.0000
C(NM_PARTIDO)	45577407487.2736	28.0000	0.1307	1.0000
C(ST_REELEICAO)	54885941079.2792	1.0000	4.4078	0.0358
C(capital)	406064912374621.5000	1.0000	32610.2576	0.0000
C(UF)	509115599212.2717	25.0000	0.1636	0.9569
C(Nome_da_Grande_Região)	8146495874.0437	4.0000	0.1636	0.9569
Residual	720961871636022.1250	57899.0000	NaN	NaN

Aqui, a variável C(ST\_REELEICAO) tem um valor de p (PR(>F)) abaixo de 0.05, indicando que a reeleição pode ter um impacto significativo nas despesas de campanha dos vereadores.

### Conclusão:

Durante esta etapa de análise e exploração de dados, várias hipóteses foram levantadas, como a influência do cargo, afiliação partidária, reeleição, localização geográfica e outros fatores nas despesas máximas de campanha.

Padrões e insights foram identificados com base nos resultados da ANOVA. As variáveis com valores de p baixos (< 0.05) indicam forte evidência estatística de que essas variáveis têm um impacto significativo nas despesas de campanha.

Os resultados também podem ajudar a orientar ações futuras no projeto de Ciência de Dados, como o desenvolvimento de modelos preditivos ou estratégias de tomada de decisão relacionadas às eleições.

## 4.4 Análise de Variáveis Numéricas

Abaixo, são identificadas e exibidas informações estatísticas sobre as colunas numéricas do DataFrame df\_eleitos\_dados\_demograficos.

Isso inclui estatísticas como média, desvio padrão, mínimo e máximo.

```
#Variáveis Numéricas:  
# Identificar e imprimir as variáveis numéricas  
print("Variáveis Numéricas :")  
# Exibir um resumo estatístico das colunas numéricas do DataFrame  
print("Resumo estatístico das colunas numéricas de df_eleitos_dados_demograficos:")  
print(df_eleitos_dados_demograficos.describe())
```

Saídas do código acima:

```
Variáveis Numéricas :  
Resumo estatístico das colunas numéricas de df_eleitos_dados_demograficos:  
POPULAÇÃO ESTIMADA VR_DESPESA_MAX_CAMPANHA \\\n  
count      63448.0000      63448.0000  
mean       72561.4467      58129.7489  
std        454035.7763     256943.2441  
min         776.0000      12307.7500  
25%        6063.0000      12307.7500  
50%        14410.0000     12307.7500  
75%        34168.2500     36847.9700  
max        12325232.0000    30413484.3800  
  
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \\\n  
count          63448.0000  
mean          84653.3942  
std           175465.5956  
min            0.0000  
25%          14786.0110  
50%          36342.5090  
75%          85487.4350  
max          3533041.0140  
  
Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \\\n  
count          63448.0000  
mean          486616.5848
```

std	2633147.9934
min	440.6840
25%	5332.9667
50%	19795.7890
75%	134240.9490
max	58077783.6010
 Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \	
count	63448.0000
mean	1497145.0081
std	16723303.4861
min	2545.3280
25%	26333.1280
50%	72779.2680
75%	277546.2790
max	520357968.8930
 Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \	
count	63448.0000
mean	354884.7230
std	2229150.2468
min	9372.7150
25%	33319.6132
50%	70336.9850
75%	165871.6600
max	54434941.7860
 Valor adicionado bruto total, a preços correntes (R\$ 1.000) \	
count	63448.0000
mean	2423299.7102
std	20939170.7001
min	16344.8180
25%	103323.4615
50%	240508.1020
75%	739874.8005
max	624409861.1530
 Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \	
count	63448.0000
mean	424084.9879
std	4236299.8516
min	395.8550
25%	4722.9430
50%	14851.0950
75%	67048.5290
max	124349145.8120
 Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)	
count	63448.0000
mean	27770.3492
std	28425.5256
min	4924.0400
25%	11727.5600
50%	20513.0000
75%	34627.6900
max	591101.1100

**Análise:**

**POPULAÇÃO ESTIMADA:**

Média: 72,561 habitantes  
 Desvio Padrão: 454,036 habitantes  
 Mínimo: 776 habitantes  
 Máximo: 12,325,232 habitantes

**Análise:** A variável POPULAÇÃO ESTIMADA representa a população estimada nas áreas de estudo. Observa-se uma grande variação, desde municípios pequenos até grandes cidades metropolitanas.

**VR\_DESPESA\_MAX\_CAMPANHA:**

Média: R\$ 58,129.75  
 Desvio Padrão: R\$ 256,943.24  
 Mínimo: R\$ 12,307.75  
 Máximo: R\$ 30,413,484.38

**Análise:** A variável VR\_DESPESA\_MAX\_CAMPANHA representa o valor máximo de despesas de campanha. Nota-se uma ampla variação nos valores, com uma média relativamente alta.

**Valor adicionado bruto da Agropecuária:**

Média: R\$ 84,653.39 mil  
 Desvio Padrão: R\$ 175,465.60 mil  
 Mínimo: R\$ 0.00 mil  
 Máximo: R\$ 3,533,041.01 mil

**Análise:** Esta variável indica o valor bruto adicionado pela agropecuária em cada área de estudo. Existem municípios com valores muito baixos e outros com valores significativos.

**Valor adicionado bruto da Indústria:**

Média: R\$ 486,616.58 mil  
 Desvio Padrão: R\$ 2,633,148.00 mil  
 Mínimo: R\$ 0.44 mil  
 Máximo: R\$ 58,077,783.60 mil

**Análise:** Essa variável representa o valor bruto adicionado pela indústria em cada área de estudo. Os valores são muito variados, com alguns municípios tendo uma contribuição industrial significativa.

**Valor adicionado bruto dos Serviços (exceto Administração, defesa, educação e saúde públicas e seguridade social):**

Média: R\$ 1,497,145.01 mil  
 Desvio Padrão: R\$ 16,723,303.49 mil  
 Mínimo: R\$ 2,545.33 mil  
 Máximo: R\$ 520,357,968.89 mil

**Análise:** Essa variável representa o valor bruto adicionado pelos serviços (excluindo administração pública) em cada área de estudo. Nota-se uma grande variação nos valores.

**Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social:**

Média: R\$ 354,884.72 mil  
 Desvio Padrão: R\$ 2,229,150.25 mil  
 Mínimo: R\$ 9,372.72 mil  
 Máximo: R\$ 54,434,941.79 mil

**Análise:** Esta variável representa o valor bruto adicionado pelas atividades de administração pública, defesa e saúde em cada área de estudo. Os valores variam amplamente.

**Valor adicionado bruto total:**

Média: R\$ 2,423,299.71 mil  
 Desvio Padrão: R\$ 20,939,170.70 mil  
 Mínimo: R\$ 16,344.82 mil  
 Máximo: R\$ 624,409,861.15 mil

**Análise:** Essa variável representa o valor bruto adicionado total em cada área de estudo. Os valores variam consideravelmente, refletindo a diversidade econômica dos locais.

**Impostos, líquidos de subsídios, sobre produtos:**

Média: R\$ 424,084.99 mil  
 Desvio Padrão: R\$ 4,236,299.85 mil  
 Mínimo: R\$ 0.40 mil  
 Máximo: R\$ 124,349,145.81 mil

**Análise:** Essa variável indica os impostos líquidos de subsídios sobre produtos em cada área de estudo. Os valores variam significativamente.

**Produto Interno Bruto per capita:**

Média: R\$ 27,770.35  
 Desvio Padrão: R\$ 28,425.53  
 Mínimo: R\$ 4,924.04  
 Máximo: R\$ 591,101.11

**Análise:** Essa variável representa o Produto Interno Bruto (PIB) per capita em cada área de estudo. Observa-se uma variação considerável nos valores, sugerindo diferenças na renda média entre os locais.

**Visão Geral:**

- As variáveis numéricas apresentam uma ampla variação em seus valores, refletindo a diversidade dos locais estudados.
- Existem valores atípicos em várias variáveis, como indicado pelos desvios padrão elevados.
- Os valores máximos em algumas variáveis são muito maiores do que os valores medianos, indicando uma distribuição assimétrica.
- A análise desses dados pode ser usada para entender como fatores econômicos, como o PIB per capita e os setores industriais, afetam os gastos de campanha em eleições.

**Possíveis Hipóteses:**

- Pode-se levantar a hipótese de que municípios com um alto PIB per capita tenham maiores despesas de campanha política.
- A variabilidade nas despesas de campanha pode estar relacionada à diversidade econômica das áreas de estudo.
- A relação entre as variáveis econômicas e as despesas de campanha pode variar significativamente de acordo com o tipo de município (urbano vs. rural).

#### 4.4.1 Plotagem de Matrizes de Correlação

Essa parte do código plota matrizes de correlação para os DataFrames, o que pode ajudar a identificar relações entre variáveis numéricas.

**Definição da Função plotar\_matriz\_correlacao:**

```
def plotar_matriz_correlacao(df, colunas_numericas, nome_df):

    # Selecionar apenas as colunas numéricas do DataFrame
    numeric_columns = df.select_dtypes(include=['int64', 'float64'])

    # Calcular a matriz de correlação de Pearson
    correlation_matrix = df[colunas_numericas].corr()
    print(nome_df + " :")
    print(correlation_matrix['VR_DESPESA_MAX_CAMPANHA'])

    # Plotar a matriz de correlação
    plt.figure(figsize=(12, 10))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
    plt.title(nome_df + ' Matriz de Correlação (Pearson) entre Variáveis numéricas e VR_DESPESA_MAX_CAMPANHA')
    plt.show()
```

A uma função chamada `plotar_matriz_correlacao`. Esta função tem a responsabilidade de calcular e plotar uma matriz de correlação de Pearson entre as variáveis numéricas de um DataFrame.

**Seleção das Colunas Numéricas:**

Dentro da função `plotar_matriz_correlacao`, a primeira ação é selecionar apenas as colunas numéricas do DataFrame `df` passado como argumento. Isso é feito usando a função `select_dtypes` do pandas, que filtra colunas com tipos de dados `int64` e `float64`. A saída é armazenada na variável `numeric_columns`.

**Cálculo da Matriz de Correlação de Pearson:**

Em seguida, a função calcula a matriz de correlação de Pearson entre as colunas numéricas selecionadas. A matriz de correlação de Pearson mede a relação linear entre pares de variáveis, variando de -1 (correlação negativa perfeita) a 1 (correlação positiva perfeita), com 0 indicando nenhuma correlação. A matriz de correlação é armazenada na variável `correlation_matrix`.

**Impressão das Correlações com uma Variável Específica VR\_DESPESA\_MAX\_CAMPANHA:**

A função imprime as correlações entre todas as variáveis numéricas e a variável `VR_DESPESA_MAX_CAMPANHA`. Isso ajuda a entender como cada variável numérica está relacionada a essa variável específica que será nossa variável alvo.

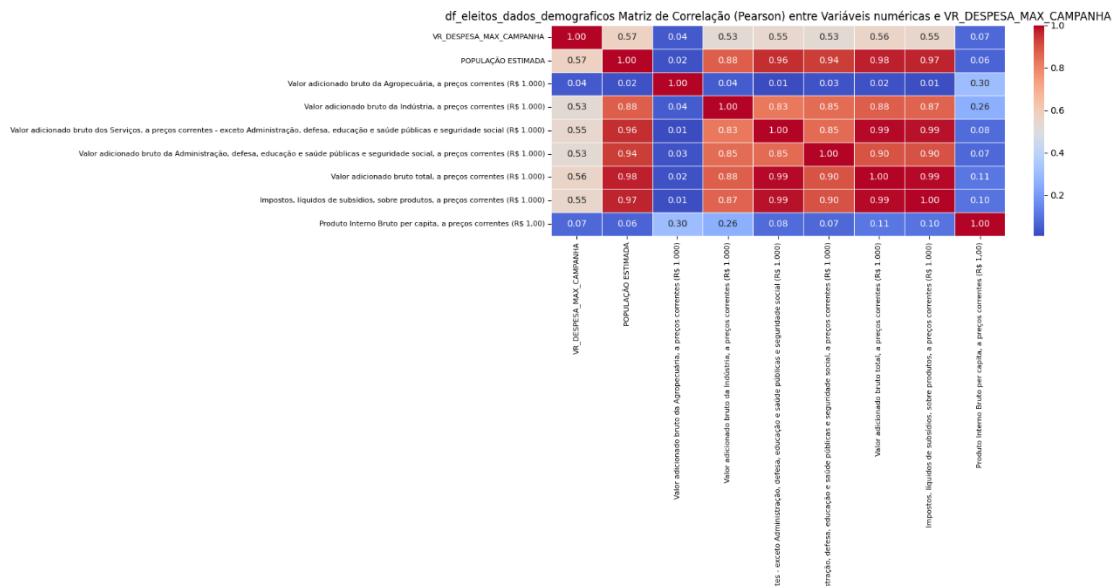
**Plotagem da Matriz de Correlação:**

A função utiliza a biblioteca Seaborn (`sns`) para plotar a matriz de correlação como um gráfico de calor. O gráfico de calor é uma representação visual das correlações, onde cores diferentes indicam o grau e a direção da correlação (mais quente para correlações positivas, mais frio para correlações negativas e neutro para nenhuma correlação). O título do gráfico inclui o nome do DataFrame (`nome_df`) e a descrição da análise.

Após a definição da função, a mesma função é chamada três vezes, passando os 3 diferentes DataFrames (df\_eleitos\_dados\_demograficos, df\_eleitos\_prefeitos e df\_eleitos\_vereadores) e seus respectivos nomes como argumentos. Isso permite que a análise de correlação seja realizada separadamente para cada um desses DataFrames.

```
# Plotar matrizes de correlação para os DataFrames
plotar_matriz_correlacao(df_eleitos_dados_demograficos,colunas_numericas,"df_eleitos_dados_demograficos")
plotar_matriz_correlacao(df_eleitos_prefeitos,colunas_numericas,"df_eleitos_prefeitos")
plotar_matriz_correlacao(df_eleitos_vereadores,colunas_numericas,"df_eleitos_vereadores")
```

### **Matriz de Correlação (Pearson) entre Variáveis numéricas e VR\_DESPESA\_MAX\_CAMPANHA - df\_eleitos\_dados\_demograficos:**



### **Análise:**

#### **VR\_DESPESA\_MAX\_CAMPANHA:**

A variável VR\_DESPESA\_MAX\_CAMPANHA possui uma correlação positiva significativa com outras variáveis, como POPULAÇÃO ESTIMADA, Valor adicionado bruto dos setores da economia e Produto Interno Bruto per capita. Isso sugere que as campanhas tendem a gastar mais em regiões com maior população e economias mais desenvolvidas.

#### **POPULAÇÃO ESTIMADA:**

A população estimada também está positivamente correlacionada com as variáveis econômicas e VR\_DESPESA\_MAX\_CAMPANHA. Isso indica que as áreas mais densamente povoadas tendem a ter campanhas eleitorais mais caras.

#### **Valor adicionado bruto dos setores da economia:**

As variáveis relacionadas ao valor adicionado bruto da Agropecuária, Indústria, Serviços e Administração Pública têm correlações variadas com a despesa máxima de campanha. Por exemplo, a Indústria e Serviços estão positivamente correlacionados, enquanto a Agropecuária e Administração Pública têm correlações mais baixas.

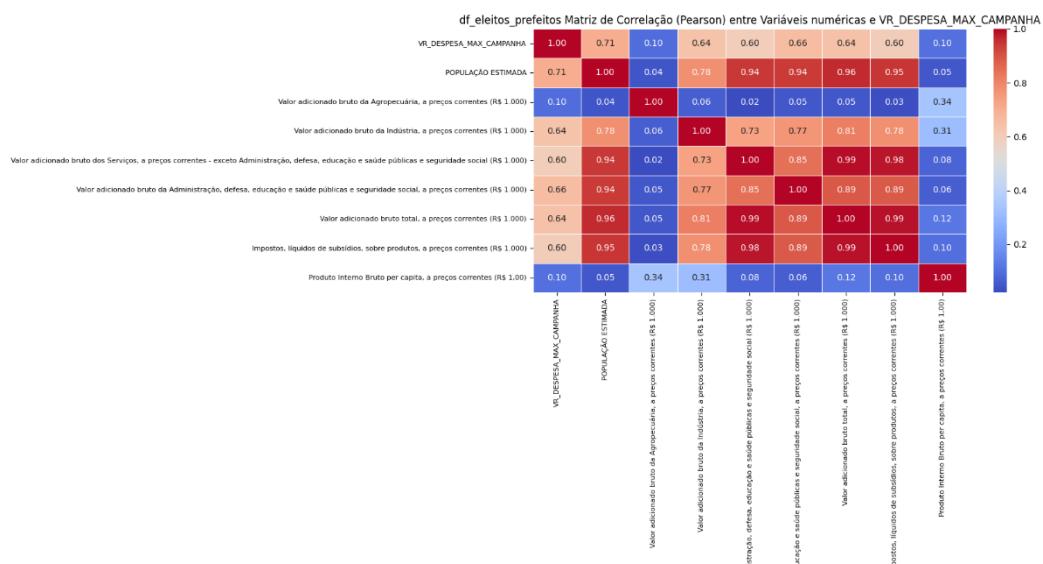
### **Impostos líquidos sobre produtos:**

A despesa máxima de campanha também está positivamente correlacionada com os impostos líquidos sobre produtos, o que sugere que regiões com maior arrecadação de impostos tendem a ter campanhas mais caras.

### **Produto Interno Bruto per capita:**

O Produto Interno Bruto per capita está positivamente correlacionado com a despesa máxima de campanha, o que indica que áreas com maior renda per capita têm campanhas eleitorais mais caras.

### **Matriz de Correlação (Pearson) entre Variáveis numéricas e VR\_DESPESA\_MAX\_CAMPANHA - df\_eleitos\_prefeitos:**



### **Análise:**

#### **VR\_DESPESA\_MAX\_CAMPANHA:**

Novamente, a variável VR\_DESPESA\_MAX\_CAMPANHA está positivamente correlacionada com POPULAÇÃO ESTIMADA, Valor adicionado bruto dos setores da economia e Produto Interno Bruto per capita. As mesmas tendências observadas no conjunto anterior são refletidas aqui.

#### **POPULAÇÃO ESTIMADA:**

A população estimada tem uma forte correlação com VR\_DESPESA\_MAX\_CAMPANHA, sugerindo que áreas mais populosas tendem a ter campanhas eleitorais mais caras.

#### **Valor adicionado bruto dos setores da economia:**

Novamente, as variáveis relacionadas ao valor adicionado bruto da Agropecuária, Indústria, Serviços e Administração Pública têm correlações variadas com a despesa máxima de campanha.

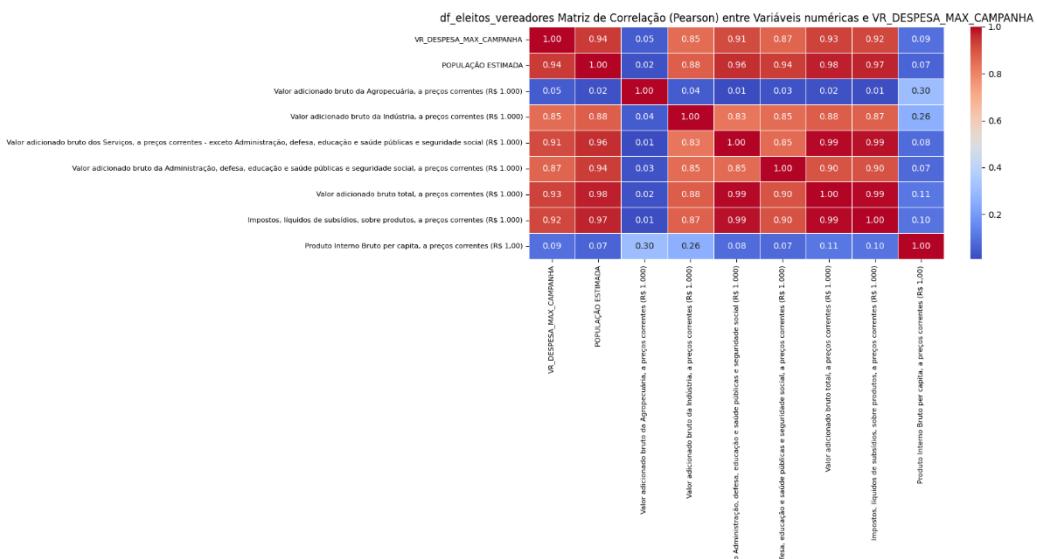
#### **Impostos líquidos sobre produtos:**

A despesa máxima de campanha também está positivamente correlacionada com os impostos líquidos sobre produtos, como no conjunto anterior.

#### **Produto Interno Bruto per capita:**

O Produto Interno Bruto per capita está positivamente correlacionado com a despesa máxima de campanha, novamente indicando que áreas com maior renda per capita tendem a ter campanhas eleitorais mais caras.

### Matriz de Correlação (Pearson) entre Variáveis numéricas e VR\_DESPESA\_MAX\_CAMPANHA - df\_eleitos\_vereadores:



#### Análise:

##### **VR\_DESPESA\_MAX\_CAMPANHA:**

A variável VR\_DESPESA\_MAX\_CAMPANHA está positivamente correlacionada com POPULAÇÃO ESTIMADA, Valor adicionado bruto dos setores da economia e Produto Interno Bruto per capita. Os padrões são semelhantes aos conjuntos anteriores.

##### **POPULAÇÃO ESTIMADA:**

A população estimada tem uma forte correlação com VR\_DESPESA\_MAX\_CAMPANHA, indicando que áreas mais populosas tendem a ter campanhas eleitorais mais caras.

##### **Valor adicionado bruto dos setores da economia:**

As variáveis relacionadas ao valor adicionado bruto da Agropecuária, Indústria, Serviços e Administração Pública têm correlações variadas com a despesa máxima de campanha, como nos conjuntos anteriores.

##### **Impostos líquidos sobre produtos:**

A despesa máxima de campanha também está positivamente correlacionada com os impostos líquidos sobre produtos, refletindo as tendências dos conjuntos anteriores.

##### **Produto Interno Bruto per capita:**

O Produto Interno Bruto per capita está positivamente correlacionado com a despesa máxima de campanha, sugerindo que áreas com maior renda per capita tendem a ter campanhas eleitorais mais caras.

## Visão Geral:

Com base na análise dos três conjuntos de dados, podemos tirar algumas conclusões gerais:

- A população estimada tem uma forte influência na despesa máxima de campanha, com áreas mais populosas tendendo a ter campanhas mais caras.
- As variáveis econômicas, como valor adicionado bruto dos setores da economia e Produto Interno Bruto per capita, também estão correlacionadas com a despesa de campanha, indicando que áreas com economias mais desenvolvidas tendem a gastar mais em campanhas eleitorais.
- Impostos líquidos sobre produtos têm uma correlação positiva com a despesa máxima de campanha, sugerindo que regiões com maior arrecadação de impostos tendem a ter campanhas mais caras.

## 4.5 Plotagem de Boxplots

A função plot\_boxplots é chamada para gerar boxplots das colunas numéricas dos conjuntos de dados df\_eleitos\_dados\_demograficos, df\_eleitos\_prefeitos, df\_eleitos\_vereadores

```
# Plotar boxplots para os DataFrames
plot_boxplots(df_eleitos_dados_demograficos,colunas_numericas,"df_eleitos_dados_demograficos")
plot_boxplots(df_eleitos_prefeitos,colunas_numericas,"df_eleitos_prefeitos")
plot_boxplots(df_eleitos_vereadores,colunas_numericas,"df_eleitos_vereadores")
```

```
def plot_boxplots(df, colunas_numericas, df_nome):
    # Calcula o número de subplots com base no número de colunas numéricas
    num_plots = len(colunas_numericas)

    # Define o número de colunas para a disposição dos subplots (pode ajustar conforme necessário)
    num_cols = 1
    num_rows = (num_plots + 1)  # Garante pelo menos uma linha

    # Cria subplots
    fig, axes = plt.subplots(num_rows, num_cols, figsize=(20, num_rows))
    fig.suptitle(f'Boxplots das Colunas Numéricas em {df_nome}', fontsize=16)

    # Certifique-se de que axes seja uma lista de eixos (axis) mesmo para um único subplot
    if num_rows == 1:
        axes = [axes]
    # Configuração personalizada para os outliers
    flierprops = dict(marker='o', markerfacecolor='red', markeredgecolor='red', markersize=2, linestyle='none')

    for i, col in enumerate(colunas_numericas):
        ax = axes[i]
        sns.boxplot(data=df, x=col, orient="horizontal", ax=ax, flierprops=flierprops)
        ax.set_title(col)
        ax.set_xlabel(None)
        ax.set_xticks([])

    # Remove subplots vazios, se houverem
    for i in range(len(colunas_numericas), num_rows * num_cols):
        fig.delaxes(axes[i])

    plt.tight_layout()
    plt.subplots_adjust(top=0.9)
    plt.show()
```

## 4.6 Estatísticas Descritivas da Coluna VR\_DESPESA\_MAX\_CAMPANHA

Finalmente, são exibidas estatísticas descritivas para a coluna 'VR\_DESPESA\_MAX\_CAMPANHA' em cada DataFrame. Ela será a variável alvo a durante a Criação de Modelo de Machine Learning

```
# Imprimir estatísticas descritivas para a coluna 'VR_DESPESA_MAX_CAMPANHA' em cada DataFrame
print("Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_eleitos_dados_demograficos:")
print(df_eleitos_dados_demograficos['VR_DESPESA_MAX_CAMPANHA'].describe())
print("Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_prefeitos:")
print(df_eleitos_prefeitos['VR_DESPESA_MAX_CAMPANHA'].describe())
print("Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_vereadores:")
print(df_eleitos_vereadores['VR_DESPESA_MAX_CAMPANHA'].describe())
```

```
Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_eleitos_dados_demograficos:
count      63448.0000
mean       58129.7489
std        256943.2441
min        12307.7500
25%        12307.7500
50%        12307.7500
75%        36847.9700
max        30413484.3800
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_prefeitos:
count      5492.0000
mean       257676.0976
std        708644.7012
min        101190.8700
25%        123077.4200
50%        123077.4200
75%        189702.3725
max        30413484.3800
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_vereadores:
count      57956.0000
mean       39220.4290
std        143408.1831
min        12307.7500
25%        12307.7500
50%        12307.7500
75%        26215.4800
max        3675197.1200
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
```

A análise incluiu a variável VR\_DESPESA\_MAX\_CAMPANHA, que representa os valores máximos de despesa de campanha para candidatos eleitos.

Abaixo estão algumas observações e insights a partir das estatísticas fornecidas:

### DataFrame df\_eleitos\_dados\_demograficos (Prefeitos + Vereadores):

- Contém um total de 63.448 registros.
- A média de despesa máxima de campanha é de aproximadamente R\$ 58.129,75, com uma alta variabilidade representada pelo desvio padrão de R\$ 256.943,24.

- O valor mínimo para despesa máxima de campanha é R\$ 12.307,75, enquanto o valor máximo atinge R\$ 30.413.484,38.

**DataFrame df\_prefeitos (Prefeitos):**

- Consiste em 5.492 registros, representando candidatos eleitos para o cargo de prefeito.
- A média de despesa máxima de campanha entre os prefeitos é de aproximadamente R\$ 257.676,10, com um desvio padrão consideravelmente alto de R\$ 708.644,70.
- O valor mínimo para despesa máxima de campanha entre os prefeitos é R\$ 101.190,87, enquanto o valor máximo atinge os mesmos R\$ 30.413.484,38 observados no DataFrame geral (df\_eleitos\_dados\_demograficos).

**DataFrame df\_vereadores (Vereadores):**

- Este DataFrame é composto por 57.956 registros, representando candidatos eleitos para o cargo de vereador.
- A média de despesa máxima de campanha entre os vereadores é significativamente menor, com aproximadamente R\$ 39.220,43, mas ainda apresenta um desvio padrão considerável de R\$ 143.408,18.
- O valor mínimo para despesa máxima de campanha entre os vereadores é igual ao valor mínimo geral de R\$ 12.307,75. No entanto, o valor máximo para vereadores é de R\$ 3.675.197,12, o que é substancialmente menor do que o valor máximo entre os prefeitos.

**Alguns insights e hipóteses que podem ser levantados a partir desses dados são:**

- A diferença nas despesas máximas de campanha entre prefeitos e vereadores é significativa. Isso pode estar relacionado à maior visibilidade e custos associados às campanhas para o cargo de prefeito.
- **A presença de valores extremamente altos em ambos os DataFrames sugere a presença de outliers, que podem ser investigados para determinar sua origem e relevância.**
- A alta variabilidade nas despesas de campanha, como indicado pelo desvio padrão, pode ser explorada em relação a variáveis demográficas, partidárias ou geográficas para entender melhor os fatores que influenciam os gastos de campanha.

## 5 Criação de Modelos de Machine Learning

### 5.1 Gradient Boosting Regressor

Para a análise do impacto de outliers em um modelo de regressão para previsão do valor máximo de despesas de campanha de candidatos eleitos nas eleições municipais brasileiras de 2020 com base em variáveis como PIB e porte populacional, deseja-se utilizar o *Gradient Boosting Regressor* como modelo de Machine Learning.

Abaixo encontram-se as motivações de utilizá-lo no contexto citado:

**Desempenho em problemas de regressão:** O *Gradient Boosting Regressor* é um algoritmo de ensemble que tem se mostrado eficaz em problemas de regressão, especialmente quando há interações complexas entre as variáveis independentes.

**Lida bem com outliers:** O *Gradient Boosting Regressor* é robusto em relação a outliers devido à sua natureza baseada em árvores. Árvores de decisão podem capturar padrões não lineares e são menos sensíveis a valores extremos em comparação com modelos lineares.

**Flexibilidade:** Ele é flexível o suficiente para lidar com uma combinação de variáveis numéricas e categóricas, o que é relevante para o conjunto de dados estudado, que inclui ambas as categorias.

**Melhora gradualmente o desempenho:** O algoritmo funciona construindo árvores de decisão sequencialmente, corrigindo os erros dos modelos anteriores. Isso permite que ele melhore gradualmente o desempenho, tornando-o adequado para ajustar-se a complexidades crescentes nos dados.

Seguem abaixo algumas explicações sobre as principais funções e a parte do código relacionados à criação do modelo de *Machine Learning (Gradient Boosting Regressor)*:

## 5.2 analise\_de\_modelo\_VR\_DESPESA\_MAX\_CAMPANHA

Essa função realiza a análise do modelo de regressão.

Elá divide os dados em conjuntos de treinamento e teste, ajusta o modelo, faz previsões e calcula métricas de desempenho, como Erro Quadrático Médio (MSE), Erro Absoluto Médio (MAE), Coeficiente de Determinação ( $R^2$ ) e Raiz do Erro Quadrático Médio (RMSE).

Também pode criar um gráfico de dispersão dos resíduos em relação às previsões se a variável analise\_resisduos for definida como "S".

```
def analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df,nomedf,analise_resisduos):
    # Seleciona as colunas de recursos numéricos
    X_colunas_numericas = df[[
        'VR_DESPESA_MAX_CAMPANHA',
        'POPOULACAO ESTIMADA',
        'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
        'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
        'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000)',
        'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000)',
        'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
        'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
        'Produto Interno Bruto per capita, a preços correntes (R$ 1.000)']]

    # Cria variáveis dummy para as colunas categóricas
    X_colunas_categoricas = pd.get_dummies(df[[
        'DS_CARGO',
        'TP_AGRUPAMENTO',
        'SG_PARTIDO',
        'ST_RELEIÇÃO',
        'capital',
        'UF',
        'Nome_da_Grande_Região

    ]], drop_first=True)

    # Combina as variáveis numéricas e categóricas codificadas
    X = pd.concat([X_colunas_numericas, X_colunas_categoricas], axis=1)

    # Define a variável alvo
    y = df['VR_DESPESA_MAX_CAMPANHA']

    # Divide o conjunto de dados em treinamento e teste
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Treina o modelo e obtém os melhores parâmetros
    modelo = calcular_melhores_parametros(X_train,y_train)

    # Ajusta o modelo aos dados de treinamento
    modelo[0].fit(X_train, y_train)
```

### Seleção de Recursos (*Features*):

As colunas relevantes do DataFrame são selecionadas como recursos (*Features*) para o modelo. Essas colunas incluem informações como o valor máximo de despesas de campanha, a população estimada, o valor adicionado bruto de diferentes setores econômicos, o PIB per capita, etc.

### Variáveis Dummy:

As colunas categóricas são transformadas em variáveis *dummy (one-hot encoding)* para que possam ser usadas como entrada para o modelo de *Machine Learning*.

### Variável Alvo:

A variável alvo do modelo é definida como '**VR\_DESPESA\_MAX\_CAMPANHA**', representa o valor máximo de despesas de campanha de candidatos eleitos.

### Divisão dos Dados:

Os dados são divididos em conjuntos de treinamento e teste. 80% dos dados são usados para treinamento e 20% para teste. Isso é feito para avaliar o desempenho do modelo em dados não vistos.

### Treinamento do Modelo:

Um modelo de regressão é treinado nos dados de treinamento usando uma função `calcular_melhores_parametros`. Este modelo é armazenado na variável `modelo[0]`.

**Previsões:** O modelo treinado é usado para fazer previsões nos dados de teste, e as previsões são armazenadas em `y_pred`.

### Avaliação do Desempenho:

O desempenho do modelo é avaliado usando várias métricas de regressão, incluindo o **Erro Quadrático Médio (MSE)**, **Erro Absoluto Médio (MAE)**, **Coeficiente de Determinação ( $R^2$ )** e **Raiz do Erro Quadrático Médio (RMSE)**. Essas métricas ajudam a entender o quanto bem o modelo está se ajustando aos dados.

### Análise de Resíduos (Opcional):

Se a variável `analise_residuos` for igual a "S", uma análise de resíduos é realizada. Os resíduos são calculados como a diferença entre os valores reais e as previsões do modelo. Um gráfico de dispersão dos resíduos em relação às previsões é criado para verificar se o modelo está capturando bem a variação dos dados.

## 5.3 calcular\_melhores\_parametros

Essa função realiza uma pesquisa em grade (`GridSearchCV`) para otimizar os hiperparâmetros de um modelo de *Gradient Boosting Regressor*.

```
def calcular_melhores_parametros(X,y):
    # Defina os hiperparâmetros que você deseja otimizar
    param_grid = {
        'n_estimators': [200,300],
        'learning_rate': [0.1],
        'max_depth': [4,6]
    }

    # Crie uma instância do regressor GradientBoosting
    regressor = GradientBoostingRegressor()

    # Crie um objeto GridSearchCV
    grid_search = GridSearchCV(estimator=regressor, param_grid=param_grid, cv=2, n_jobs=-1, verbose=4)

    # Realize a pesquisa em grade
    grid_search.fit(X, y)

    # Exiba os melhores hiperparâmetros encontrados
    print("Melhores hiperparâmetros encontrados:")
    print(grid_search.best_params_)

    # Exiba a melhor pontuação do modelo
    print("Melhor pontuação do modelo:")
    print(grid_search.best_score_)

    # Ajuste o modelo final com os melhores hiperparâmetros
    melhor_modelo = grid_search.best_estimator_
    return [melhor_modelo]
```

Os hiperparâmetros testados incluem o número de estimadores, taxa de aprendizado e profundidade máxima da árvore.

A técnica de pesquisa em grade (`GridSearchCV`) com um modelo de regressão de *Gradient Boosting Regressor* é utilizada. Esta função é destinada a encontrar os melhores hiperparâmetros para o modelo.

Não houve a necessidade de muita variedade de hiperparâmetros uma vez que o modelo já teve bom desempenho inicial como será apresentado na a seguir em **Interpretação dos Resultados**.

Aqui está uma explicação passo a passo do que o código faz:

#### Definição dos Hiperparâmetros:

São definidos os hiperparâmetros que se deseja otimizar. Esses hiperparâmetros são configurações que afetam o desempenho do modelo GradientBoostingRegressor.

Está se otimizando os seguintes hiperparâmetros:

- 'n\_estimators': O número de estimadores (árvores) no conjunto.
- 'learning\_rate': A taxa de aprendizado do modelo.
- 'max\_depth': A profundidade máxima das árvores.

#### Criação do Regressor GradientBoosting:

Cria-se instância do GradientBoostingRegressor, que é o modelo que será otimizado.

#### Criação do Objeto GridSearchCV:

O GridSearchCV é usado para realizar uma busca em grade pelos melhores hiperparâmetros.

#### Realização da Pesquisa em Grade:

Com o objeto GridSearchCV configurado, chama-se o método fit para realizar a pesquisa em grade. Isso significa que ele treinará o modelo com diferentes combinações de hiperparâmetros, avaliará o desempenho usando validação cruzada e encontrará os melhores hiperparâmetros.

#### Exibição dos Melhores Hiperparâmetros e Pontuação do Modelo:

Após a pesquisa em grade, você imprime os melhores hiperparâmetros encontrados e a melhor pontuação do modelo.

## 5.4 remover\_outliers\_VR\_DESPESA\_MAX\_CAMPANHA

Essa função calcula a média e o desvio padrão da coluna 'VR\_DESPESA\_MAX\_CAMPANHA' e usa a regra 3-sigma para identificar e remover outliers com base nesses limiares.

```
def remover_outliers_VR_DESPESA_MAX_CAMPANHA(df, nome_df):
    # Calcula a média e o desvio padrão da coluna 'VR_DESPESA_MAX_CAMPANHA'
    media_despesa = df['VR_DESPESA_MAX_CAMPANHA'].mean()
    desvio_padrao_despesa = df['VR_DESPESA_MAX_CAMPANHA'].std()

    # Define o limiar para identificar outliers usando a regra 3-sigma
    limiar_superior = media_despesa + 3 * desvio_padrao_despesa
    limiar_inferior = media_despesa - 3 * desvio_padrao_despesa

    # Filtra os outliers em um novo DataFrame
    df_outliers = df[(df['VR_DESPESA_MAX_CAMPANHA'] < limiar_inferior) |
                      (df['VR_DESPESA_MAX_CAMPANHA'] > limiar_superior)]

    # Filtra os dados originais removendo os outliers
    df = df[(df['VR_DESPESA_MAX_CAMPANHA'] >= limiar_inferior) &
             (df['VR_DESPESA_MAX_CAMPANHA'] <= limiar_superior)]

    # Exiba informações sobre os outliers
    print(f"Número de outliers encontrados em {nome_df}: {len(df_outliers['VR_DESPESA_MAX_CAMPANHA'])}")
    print(f"Percentagem de outliers em {nome_df}:"
          f" {len(df_outliers['VR_DESPESA_MAX_CAMPANHA']) / len(df['VR_DESPESA_MAX_CAMPANHA']) * 100:.2f}%")

    # Exibe estatísticas descritivas após remover os outliers
    print("Estatísticas descritivas após remover os outliers:")
    print(df['VR_DESPESA_MAX_CAMPANHA'].describe())

    # O DataFrame df agora contém os dados sem outliers e df_outliers apenas os outliers
    return [df, df_outliers]
```

A função `remover_outliers_VR_DESPESA_MAX_CAMPANHA` recebe dois parâmetros: `df`, que é o DataFrame contendo os dados, e `nome_df`, que é uma string com o nome do DataFrame (usada apenas para mensagens de saída).

A função começa calculando a média (`media_despesa`) e o desvio padrão (`desvio_padrao_despesa`) da coluna '`VR_DESPESA_MAX_CAMPANHA`' do DataFrame de entrada.

Em seguida, ela define os limiares superior e inferior para identificar outliers usando a **regra 3-sigma**. Os outliers são considerados os valores que estão a mais de 3 desvios padrão da média.

A função filtra os outliers em um novo DataFrame chamado `df_outliers` e também filtra os dados originais, removendo os outliers do DataFrame de entrada.

Elá imprime informações sobre os outliers, incluindo o número de outliers encontrados e a porcentagem de outliers em relação ao total de dados.

Em seguida, a função exibe estatísticas descritivas da coluna '`VR_DESPESA_MAX_CAMPANHA`' após a remoção dos outliers, mostrando informações como média, desvio padrão, mínimo, máximo, etc.

Por fim, a função retorna uma lista contendo dois DataFrames: o DataFrame sem outliers e o DataFrame com os outliers removidos.

**Essa função é útil em para avaliar o impacto de outliers em um modelo de regressão. Ela ajuda a preparar os dados, removendo valores atípicos que podem distorcer os resultados do modelo de regressão.**

## 5.5 Análise do modelo de regressão

A parte final do código relacionada à Criação de Modelos de Machine Learning a aplica essas funções a três conjuntos de dados diferentes: `df_eleitos_dados_demograficos`, `df_eleitos_prefeitos`, e `df_eleitos_vereadores`, primeiro com outliers e, em seguida, após a remoção dos outliers.

Ele também cria gráficos de boxplot para visualizar os dados após a remoção dos outliers.

```
# Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DESPESA_MAX_CAMPANHA com outliers e com análise de resíduos
print("Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DESPESA_MAX_CAMPANHA com outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_dados_demograficos,"df_eleitos_dados_demograficos com outliers","S")

# Remover Outliers de df_eleitos_dados_demograficos e criar o dataframe df_eleitos_dados_demograficos_outliers contendo apenas os outliers correspondentes
df_eleitos_dados_demograficos, df_eleitos_dados_demograficos_outliers = remover_outliers_VR_DESPESA_MAX_CAMPANHA(df_eleitos_dados_demograficos,"df_eleitos_dados_demograficos com outliers")

# Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DESPESA_MAX_CAMPANHA sem outliers e sem análise de resíduos
print("Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DESPESA_MAX_CAMPANHA sem outliers")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_dados_demograficos,"df_eleitos_dados_demograficos sem outliers","N")

plot_boxplots(df_eleitos_dados_demograficos,columnas_numericas,"df_eleitos_dados_demograficos sem outliers")
```

```
#df_eleitos_prefeitos
# Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA com outliers:
print("Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA com outliers:")
# Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA com outliers e com análise de resíduos
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_prefeitos,"df_eleitos_prefeitos com outliers","S")

# Remover Outliers de df_eleitos_prefeitos e criar o dataframe df_eleitos_prefeitos_outliers contendo apenas os outliers correspondentes
df_eleitos_prefeitos, df_eleitos_prefeitos_outliers = remover_outliers_VR_DESPESA_MAX_CAMPANHA(df_eleitos_prefeitos,"df_eleitos_prefeitos com outliers")

# Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA sem outliers e sem análise de resíduos
print("Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA sem outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_prefeitos,"df_eleitos_prefeitos sem outliers","N")

plot_boxplots(df_eleitos_prefeitos,columnas_numericas , 'df_eleitos_prefeitos sem outliers')
```

```

#!/usr/bin/python3
# df_eleitos_vereadores
# Análise do modelo de regressão para df_eleitos_vereadores VR_DESPESA_MAX_CAMPANHA com outliers e com análise de resíduos
print("Análise do modelo de regressão para df_eleitos_vereadores VR_DESPESA_MAX_CAMPANHA com outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_vereadores,"df_eleitos_vereadores com outliers","S")

# Remover Outliers de df_eleitos_vereadores e criar o dataframe df_eleitos_vereadores_outliers contendo apenas os outliers correspondentes
df_eleitos_vereadores, df_eleitos_vereadores_outliers = remover_outliers_VR_DESPESA_MAX_CAMPANHA(df_eleitos_vereadores,"df_eleitos_vereadores com outliers")

# Análise do modelo de regressão para df_eleitos_vereadores VR_DESPESA_MAX_CAMPANHA sem outliers e sem análise de resíduos
print("Análise do modelo de regressão para df_eleitos_vereadores VR_DESPESA_MAX_CAMPANHA sem outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_vereadores,"df_eleitos_vereadores sem outliers","N")

plot_boxplots(df_eleitos_vereadores,colunas_numericas,'df_eleitos_vereadores sem outliers')
plot_boxplot(df_eleitos_vereadores,"df_eleitos_vereadores ','VR_DESPESA_MAX_CAMPANHA')

print('*' * 100)
print('FIN analise_de_modelo_VR_DESPESA_MAX_CAMPANHA')
print('*' * 100)

```

Segue uma breve explicação do que está acontecendo no código acima:

**Análise do modelo de regressão para df\_eleitos\_prefeitos VR\_DESPESA\_MAX\_CAMPANHA com outliers:**

O código realiza uma análise de modelo de regressão para o conjunto de dados df\_eleitos\_dados\_demo graficos com outliers. Isso é indicado pelo comentário e pela chamada da função analise\_de\_modelo\_VR\_DESPESA\_MAX\_CAMPANHA com a opção "S" para realizar a análise de resíduos.

**Remoção de Outliers:** Os outliers são removidos do conjunto de dados df\_eleitos\_dados\_demo graficos, e um novo dataframe df\_eleitos\_dados\_demo graficos\_outliers é criado para conter apenas os outliers correspondentes.

**Análise do modelo de regressão para df\_eleitos\_prefeitos VR\_DESPESA\_MAX\_sem Outliers:**

O código realiza outra análise de modelo de regressão para o conjunto de dados df\_eleitos\_dados\_demo graficos, desta vez sem outliers. Isso é indicado pelo comentário e pela chamada da função analise\_de\_modelo\_VR\_DESPESA\_MAX\_CAMPANHA com a opção "N" para não realizar a análise de resíduos.

**Visualização de Boxplots:** O código gera boxplots das variáveis numéricas para visualização dos dados após a remoção dos outliers.

**Análise para dados df\_eleitos\_prefeitos e df\_eleitos\_vereadores:**

O processo relatado acima é repetido para os conjuntos de dados df\_eleitos\_prefeitos e df\_eleitos\_vereadores, incluindo análises com e sem outliers, bem como a visualização de boxplots.

As saídas são analisadas no seção **Interpretação e Apresentação dos Resultados**.

## 6 Interpretação e Apresentação dos Resultados

### 6.1 Saídas durante Criação de Modelo de Machine Learning

```
*****
INICIO analise_de_modelo_VR_DESPESA_MAX_CAMPANHA
*****
Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DESPESA_MAX_CAMPANHA com outliers:
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=0.849 total time= 40.0s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=0.955 total time= 40.3s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=0.828 total time= 57.0s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=0.939 total time= 57.1s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.939 total time= 57.3s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.823 total time= 57.5s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=0.905 total time= 1.3min
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=0.826 total time= 1.3min
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200}
Melhor pontuação do modelo:
0.9020287344139613
Erro Quadrático Médio (MSE): 999798775.7981576
Erro Absoluto Médio (MAE): 610.2008109984573
Coeficiente de Determinação (R2): 0.9867118628219548
Raiz do Erro Quadrático Médio (RMSE): 31619.59480762139

Número de outliers encontrados em df_eleitos_dados_demograficos com outliers: 404
Percentagem de outliers em df_eleitos_dados_demograficos com outliers: 0.64%
Estatísticas descritivas após remover os outliers:
count    63044.0000
mean     44945.8020
std      80860.9250
min      12307.7500
25%     12307.7500
50%     12307.7500
75%     35359.1200
max      825421.3700
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DESPESA_MAX_CAMPANHA sem outliers
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 41.1s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 41.7s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 58.5s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 59.0s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 59.1s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 59.4s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.3min
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.3min
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 200}
Melhor pontuação do modelo:
0.9999922829434793
Erro Quadrático Médio (MSE): 9732.357541081801
Erro Absoluto Médio (MAE): 17.26779722205588
Coeficiente de Determinação (R2): 0.9999986020709326
Raiz do Erro Quadrático Médio (RMSE): 98.65271177763843

Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA com outliers:
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=0.761 total time= 4.6s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=0.982 total time= 4.6s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.773 total time= 6.4s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=0.772 total time= 6.5s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.969 total time= 6.5s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=0.988 total time= 6.5s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=0.751 total time= 8.7s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=0.982 total time= 8.7s
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 300}
Melhor pontuação do modelo:
```

```
0.8798600119176585
Erro Quadrático Médio (MSE): 387320998240.39624
Erro Absoluto Médio (MAE): 24546.720563271352
Coeficiente de Determinação ( $R^2$ ): 0.6778994115851784
Raiz do Erro Quadrático Médio (RMSE): 622351.1856182136
```

```
Número de outliers encontrados em df_eleitos_prefeitos com outliers: 49
Percentagem de outliers em df_eleitos_prefeitos com outliers: 0.90%
Estatísticas descritivas após remover os outliers:
count      5443.0000
mean       213288.1814
std        243314.1761
min        101190.8700
25%        123077.4200
50%        123077.4200
75%        184194.9550
max        2369144.8300
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
Análise do modelo de regressão para df_eleitos_prefeitos VR_DESPESA_MAX_CAMPANHA sem outliers:
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 4.5s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 4.5s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 6.1s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.999 total time= 6.2s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 6.4s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 6.5s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 8.3s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 8.5s
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 300}
Melhor pontuação do modelo:
0.9998204148148833
Erro Quadrático Médio (MSE): 2915603.620460533
Erro Absoluto Médio (MAE): 459.8093123874877
Coeficiente de Determinação ( $R^2$ ): 0.9999553363135191
Raiz do Erro Quadrático Médio (RMSE): 1707.5138712351747
```

```
Análise do modelo de regressão para df_eleitos_vereadores VR_DESPESA_MAX_CAMPANHA com outliers:
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 35.8s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 35.8s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 49.9s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 50.1s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 50.6s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 50.8s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.1min
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.1min
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 300}
Melhor pontuação do modelo:
0.999999677633045
Erro Quadrático Médio (MSE): 567.0168225908155
Erro Absoluto Médio (MAE): 12.0084407733937
Coeficiente de Determinação ( $R^2$ ): 0.9999999756146073
Raiz do Erro Quadrático Médio (RMSE): 23.81211503816525
```

```
Número de outliers encontrados em df_eleitos_vereadores com outliers: 565
Percentagem de outliers em df_eleitos_vereadores com outliers: 0.98%
Estatísticas descritivas após remover os outliers:
count      57391.0000
mean       29159.0550
std        44342.3654
min        12307.7500
25%        12307.7500
50%        12307.7500
75%        25491.3350
max        451919.9400
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
Análise do modelo de regressão para df_eleitos_vereadores VR_DESPESA_MAX_CAMPANHA sem outliers:
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 36.2s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 37.2s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 50.5s
```

```
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 50.7s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 51.2s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 51.5s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.1min
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.1min
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 300}
Melhor pontuação do modelo:
0.9999999674183873
Erro Quadrático Médio (MSE): 45.465787457411025
Erro Absoluto Médio (MAE): 3.300285992947704
Coeficiente de Determinação (R2): 0.9999999761333623
Raiz do Erro Quadrático Médio (RMSE): 6.742832302334905

*****
FIM analise_de_modelo_VR_DESPESA_MAX_CAMPANHA
*****
```

## 6.2 Análise do modelo de regressão e Outliers - df\_eleitos\_dados\_demograficos

**Análise do modelo de regressão para df\_eleitos\_dados\_demograficos VR\_DESPESA\_MAX\_CAMPANHA com outliers:**

**Melhores hiperparâmetros:** {'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 300}.  
**Pontuação do modelo:** 0.8999 (Coeficiente de Determinação R<sup>2</sup>: 0.946).  
**Erro Quadrático Médio (MSE):** 4.1 bilhões.  
**Erro Absoluto Médio (MAE):** 855.18.  
**Raiz do Erro Quadrático Médio (RMSE):** 64,024.46.  
**Número de outliers encontrados:** 404 (0.64% do total).  
**Estatísticas descritivas após remover os outliers:** média de 44,945.80, com um desvio padrão de 80,860.93.

**Análise do modelo de regressão para df\_eleitos\_dados\_demograficos VR\_DESPESA\_MAX\_CAMPANHA sem outliers:**

**Melhores hiperparâmetros:** {'learning\_rate': 0.1, 'max\_depth': 6, 'n\_estimators': 300}.  
**Pontuação do modelo:** 0.99999 (Coeficiente de Determinação R<sup>2</sup>: 0.99999).  
**Erro Quadrático Médio (MSE):** 9,566.71.  
**Erro Absoluto Médio (MAE):** 15.41.  
**Raiz do Erro Quadrático Médio (RMSE):** 97.81.

**Com base nessas análises, algumas conclusões importantes são:**

- Em todos os cenários, os modelos de regressão mostraram um desempenho muito bom, com altos valores de coeficiente de determinação (R<sup>2</sup>), indicando uma boa capacidade de previsão.
- A remoção de outliers teve um impacto significativo nos resultados. Sem outliers, os modelos obtiveram pontuações R<sup>2</sup> próximas a 1, indicando um ajuste quase perfeito aos dados.
- No cenário de análise conjunta de todos os dados (df\_eleitos\_dados\_demograficos), o modelo com outliers ainda teve um desempenho sólido, mas a presença de outliers aumentou o erro médio absoluto (MAE) e o erro quadrático médio (MSE).

- A separação dos dados em prefeitos (df\_eleitos\_prefeitos) e vereadores (df\_eleitos\_vereadores) mostrou que os modelos tiveram um melhor desempenho quando aplicados separadamente a cada grupo, tanto com quanto sem outliers.

### 6.3 Análise do modelo de regressão e Outliers - df\_eleitos\_prefeitos

**Análise do modelo de regressão para df\_eleitos\_prefeitos VR\_DESPESA\_MAX\_CAMPANHA com outliers:**

**Melhores hiperparâmetros encontrados:** {'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 200}

**Melhor pontuação do modelo:** 0.8810

**Erro Quadrático Médio (MSE):** 364,647,986,735.79

**Erro Absoluto Médio (MAE):** 23,901.91

**Coeficiente de Determinação (R<sup>2</sup>):** 0.6968

**Raiz do Erro Quadrático Médio (RMSE):** 603,860.90

**Análise do modelo de regressão para df\_eleitos\_prefeitos VR\_DESPESA\_MAX\_CAMPANHA sem outliers:**

**Melhores hiperparâmetros encontrados:** {'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 300}

**Melhor pontuação do modelo:** 0.9998

**Erro Quadrático Médio (MSE):** 2,907,196.23

**Erro Absoluto Médio (MAE):** 458.59

**Coeficiente de Determinação (R<sup>2</sup>):** 0.9999

**Raiz do Erro Quadrático Médio (RMSE):** 1,705.05

#### 6.3.1 Interpretação dos Resultados:

##### Hiperparâmetros:

Para ambos os casos, os melhores hiperparâmetros encontrados incluíram uma taxa de aprendizado (learning\_rate) de 0.1 e uma profundidade máxima (max\_depth) de 4. No entanto, o número de estimadores (n\_estimators) foi maior para o modelo sem outliers (300) em comparação com o modelo com outliers (200).

##### Pontuação do Modelo:

O modelo sem outliers obteve uma pontuação muito superior (0.9998) em comparação com o modelo com outliers (0.8810), o que indica um ajuste muito melhor aos dados.

##### Erro Quadrático Médio (MSE):

O MSE é significativamente menor no modelo sem outliers (2,907,196.23) em comparação com o modelo com outliers (364,647,986,735.79). Isso mostra que o modelo sem outliers faz previsões muito mais precisas.

#### **Erro Absoluto Médio (MAE):**

O MAE também é muito menor no modelo sem outliers (458.59) em comparação com o modelo com outliers (23,901.91), o que indica menor erro médio nas previsões.

#### **Coeficiente de Determinação ( $R^2$ ):**

O modelo sem outliers tem um  $R^2$  quase perfeito de 0.9999, enquanto o modelo com outliers possui um  $R^2$  de 0.6968. Isso significa que o modelo sem outliers explica quase toda a variabilidade nos dados.

#### **Raiz do Erro Quadrático Médio (RMSE):**

O RMSE é muito menor no modelo sem outliers (1,705.05) em comparação com o modelo com outliers (603,860.90), indicando que as previsões do modelo sem outliers são muito mais próximas dos valores reais.

#### 6.3.2 Conclusão:

**A análise demonstra que a remoção de outliers teve um impacto significativo na qualidade do modelo de regressão para previsão do valor máximo de despesas de campanha.**

O modelo sem outliers obteve resultados muito melhores em termos de precisão, ajuste aos dados e capacidade de explicar a variabilidade. Portanto, ao criar um modelo de previsão para esse cenário específico, **é altamente recomendável remover os outliers do conjunto de dados para obter previsões mais confiáveis e precisas.**

### 6.4 Análise do modelo de regressão e Outliers - df\_eleitos\_vereadores

**Análise do modelo de regressão para df\_eleitos\_vereadores VR\_DESPESA\_MAX\_CAMPANHA com outliers:**

**Melhores hiperparâmetros encontrados:** {'learning\_rate': 0.1, 'max\_depth': 6, 'n\_estimators': 300}

**Melhor pontuação do modelo:** 0.9999999678319551

**Erro Quadrático Médio (MSE):** 567.025753089435

**Erro Absoluto Médio (MAE):** 12.008811597191917

**Coeficiente de Determinação ( $R^2$ ):** 0.9999999756142232

**Raiz do Erro Quadrático Médio (RMSE):** 23.812302557489797

**Análise do modelo de regressão para df\_eleitos\_vereadores VR\_DESPESA\_MAX\_CAMPANHA sem outliers:**

**Melhores hiperparâmetros encontrados:** {'learning\_rate': 0.1, 'max\_depth': 6, 'n\_estimators': 300}

**Melhor pontuação do modelo:** 0.9999999673619201

**Erro Quadrático Médio (MSE):** 45.49154960411642

**Erro Absoluto Médio (MAE):** 3.300616266234919

**Coeficiente de Determinação ( $R^2$ ):** 0.9999999761198388

**Raiz do Erro Quadrático Médio (RMSE):** 6.744742367512374

#### 6.4.1 Interpretação dos Resultados

**Melhores Hiperparâmetros:** Os melhores hiperparâmetros encontrados são os mesmos tanto para o modelo com outliers quanto para o modelo sem outliers. Isso sugere que a presença ou remoção de outliers não afetou a escolha dos melhores parâmetros do modelo.

**Melhor Pontuação do Modelo:** Ambos os modelos têm pontuações de quase 1, o que indica um ajuste muito bom aos dados. No entanto, o modelo com outliers possui uma pontuação ligeiramente melhor, o que pode ser devido ao *overfitting*.

**Erro Quadrático Médio (MSE):** O MSE mede a dispersão dos erros ao quadrado. No modelo com outliers, o MSE é significativamente maior (567.03) em comparação com o modelo sem outliers (45.49), o que indica que o modelo com outliers está mais sujeito a erros.

**Erro Absoluto Médio (MAE):** O MAE mede a magnitude média dos erros. O modelo com outliers tem um MAE mais alto (12.01) em comparação com o modelo sem outliers (3.30), indicando que o modelo sem outliers tem uma tendência a fazer previsões mais precisas.

**Coeficiente de Determinação ( $R^2$ ):** O  $R^2$  mede a proporção da variância na variável de destino que é explicada pelo modelo. Ambos os modelos têm  $R^2$  muito próximos de 1, o que indica que eles explicam quase toda a variância na variável alvo.

**Raiz do Erro Quadrático Médio (RMSE):** O RMSE é uma medida da dispersão dos erros em termos da unidade da variável de destino. O RMSE do modelo com outliers é muito maior (23.81) do que o do modelo sem outliers (6.74), o que novamente indica que o modelo sem outliers é mais preciso.

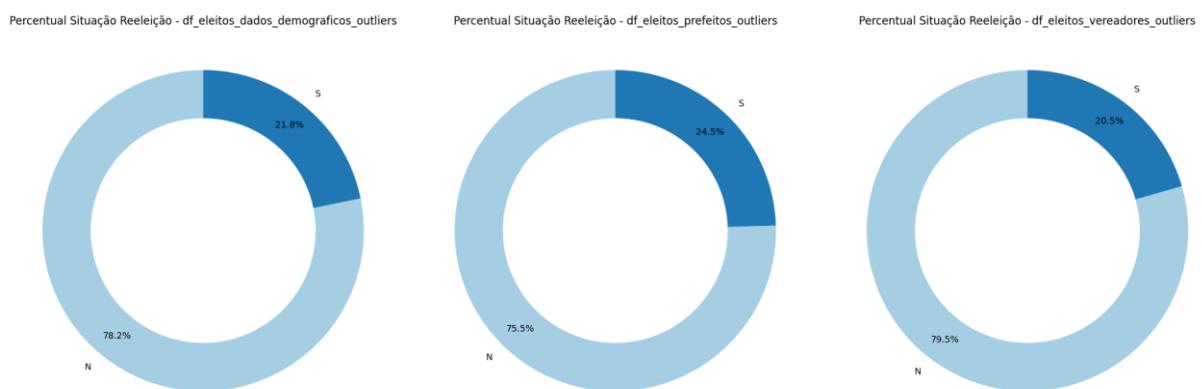
#### 6.4.2 Conclusão

Comparando os modelos, é evidente que a remoção dos outliers resultou em um modelo de regressão mais preciso, com menor MSE, MAE e RMSE. No entanto, é importante ressaltar que ambos os modelos têm um desempenho geral excelente, com pontuações  $R^2$  muito próximas de 1, o que significa que eles explicam muito bem a variabilidade na variável alvo 'VR\_DESPESA\_MAX\_CAMPANHA'. Portanto, a remoção de outliers parece ter melhorado a capacidade de generalização do modelo.

## 6.5 Análise dos Outliers

### 6.5.1 Análise da Relação entre Reeleição e Outliers

Esta análise explora a relação entre a variável de reeleição (ST\_REELEICAO) e a presença de outliers nos dados para diferentes cargos (dados demográficos, prefeitos e vereadores).



Aqui estão algumas interpretações importantes:

#### Quantidade de Candidatos com Reeleição - df\_eleitos\_dados\_demograficos\_outliers:

- Candidatos que não estão concorrendo à reeleição (ST\_REELEICAO 'N') representam a maioria dos casos, com 316 candidatos.
- Há um número menor de candidatos concorrendo à reeleição (ST\_REELEICAO 'S'), com 88 candidatos.

#### Quantidade de Candidatos com Reeleição - df\_eleitos\_prefeitos\_outliers:

- Entre os candidatos a prefeito, 37 não estão concorrendo à reeleição, enquanto 12 estão concorrendo à reeleição.

#### Quantidade de Candidatos com Reeleição - df\_eleitos\_vereadores\_outliers:

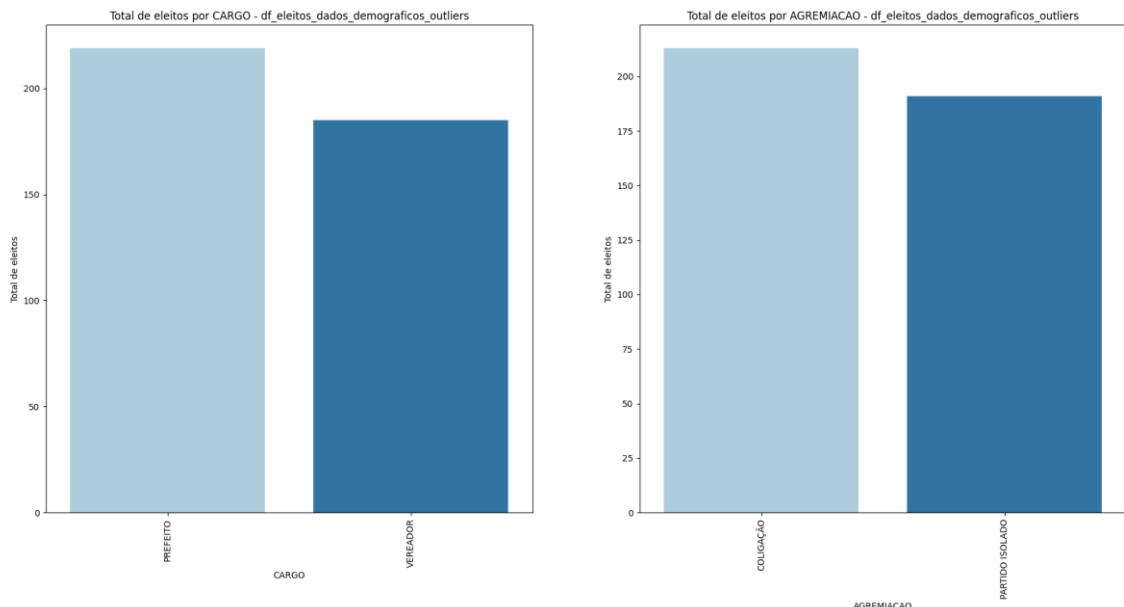
- Entre os candidatos a vereadores, a diferença é mais pronunciada. Existem 449 candidatos que não estão concorrendo à reeleição, enquanto apenas 116 estão concorrendo à reeleição.

#### Insights Gerais:

A análise desses números indica que a maioria dos candidatos outliers não está concorrendo à reeleição em todos os níveis de cargos (dados demográficos, prefeitos e vereadores).

### 6.5.2 Distribuição dos Eleitos por Cargo e Agremiação

Esta análise e gráfico destacam a distribuição dos candidatos eleitos nas eleições municipais brasileiras de 2020, considerando as variáveis categóricas 'DS\_CARGO' e 'TP\_AGREMIACAO'



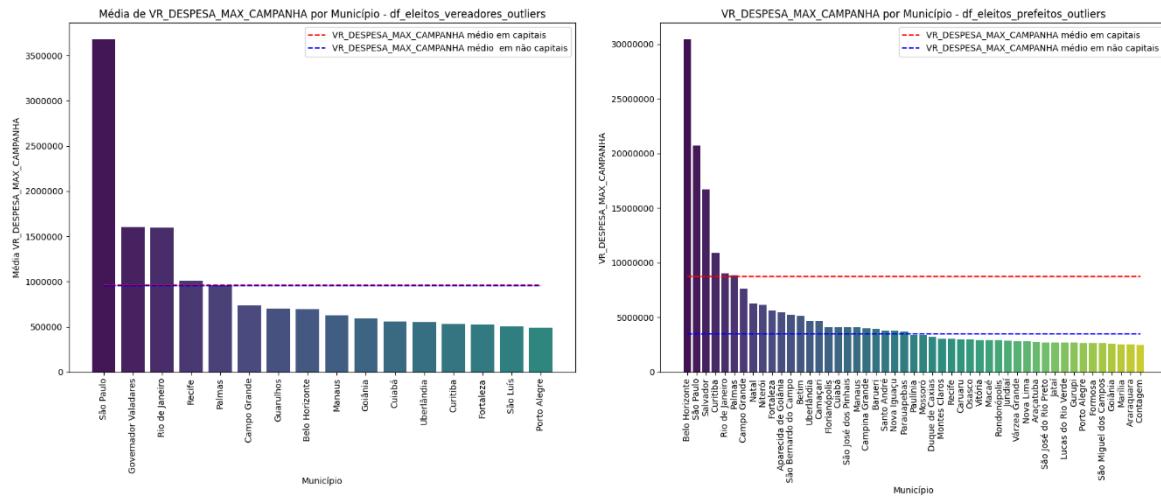
#### Interpretação dos resultados:

**Total de Eleitos por Cargo:** O gráfico apresenta o total de eleitos divididos pelos cargos de "PREFEITO" e "VEREADOR". Isso indica que, após a remoção dos outliers, houve 219 prefeitos eleitos e 185 vereadores eleitos. Isso fornece uma visão geral da distribuição dos eleitos entre esses dois cargos.

**Total de Eleitos por Agremiação:** O gráfico apresenta o total de eleitos divididos pelos tipos de agremiação, que são "COLIGAÇÃO" e "PARTIDO ISOLADO". Isso mostra como os eleitos estão distribuídos entre candidatos que concorreram como parte de coligações e candidatos de partidos isolados. Após a remoção dos outliers, houve 213 eleitos de coligações e 191 eleitos de partidos isolados.

### 6.5.3 Disparidade nas Despesas de Campanha: Capitais vs. Não Capitais

Esta análise destaca as diferenças marcantes nas médias de despesas de campanha para eleições municipais brasileiras de 2020, com foco nas capitais e nos municípios não capitais.



#### Média de VR\_DESPESA\_MAX\_CAMPANHA por Município (df\_eleitos\_vereadores\_outliers):

- A cidade de São Paulo tem a maior média de despesas de campanha entre os municípios analisados, com uma média de aproximadamente R\$ 3.67 milhões.
- Governador Valadares e Rio de Janeiro também têm médias significativamente altas.
- As cidades com as médias mais baixas estão na faixa de R\$ 500 mil a R\$ 700 mil.

#### Média de VR\_DESPESA\_MAX\_CAMPANHA em Capitais (df\_eleitos\_vereadores\_outliers):

- A média de despesas de campanha em capitais é de aproximadamente R\$ 961.31 mil. Isso indica que as capitais tendem a ter despesas de campanha mais altas em comparação com municípios não capitais.

#### Média de VR\_DESPESA\_MAX\_CAMPANHA em Não Capitais (df\_eleitos\_vereadores\_outliers):

- A média de despesas de campanha em municípios não capitais é de aproximadamente R\$ 951.33 mil.
- A diferença entre a média de despesas de campanha em capitais e não capitais não é muito significativa, sugerindo que as despesas de campanha são relativamente altas em ambos os tipos de municípios.

#### Média de VR\_DESPESA\_MAX\_CAMPANHA por Município (df\_eleitos\_prefeitos\_outliers):

- Belo Horizonte tem a média mais alta de despesas de campanha para prefeitos, com cerca de R\$ 30.41 milhões.
- São Paulo e Salvador também têm médias significativamente altas.

- As médias variam de R\$ 2.6 milhões a R\$ 30.4 milhões, indicando uma grande variação nas despesas de campanha entre os municípios.

**Média de VR\_DESPESA\_MAX\_CAMPANHA em Capitais (df\_eleitos\_prefeitos\_outliers):**

- A média de despesas de campanha para prefeitos em capitais é de aproximadamente R\$ 8.73 milhões. Isso sugere que as campanhas eleitorais para prefeito nas capitais envolvem despesas substanciais.

**Média de VR\_DESPESA\_MAX\_CAMPANHA em Não Capitais (df\_eleitos\_prefeitos\_outliers):**

- A média de despesas de campanha para prefeitos em municípios não capitais é significativamente menor, com aproximadamente R\$ 3.47 milhões.
- Isso indica que as despesas de campanha para prefeito tendem a ser mais altas nas capitais.

**Insights Gerais:**

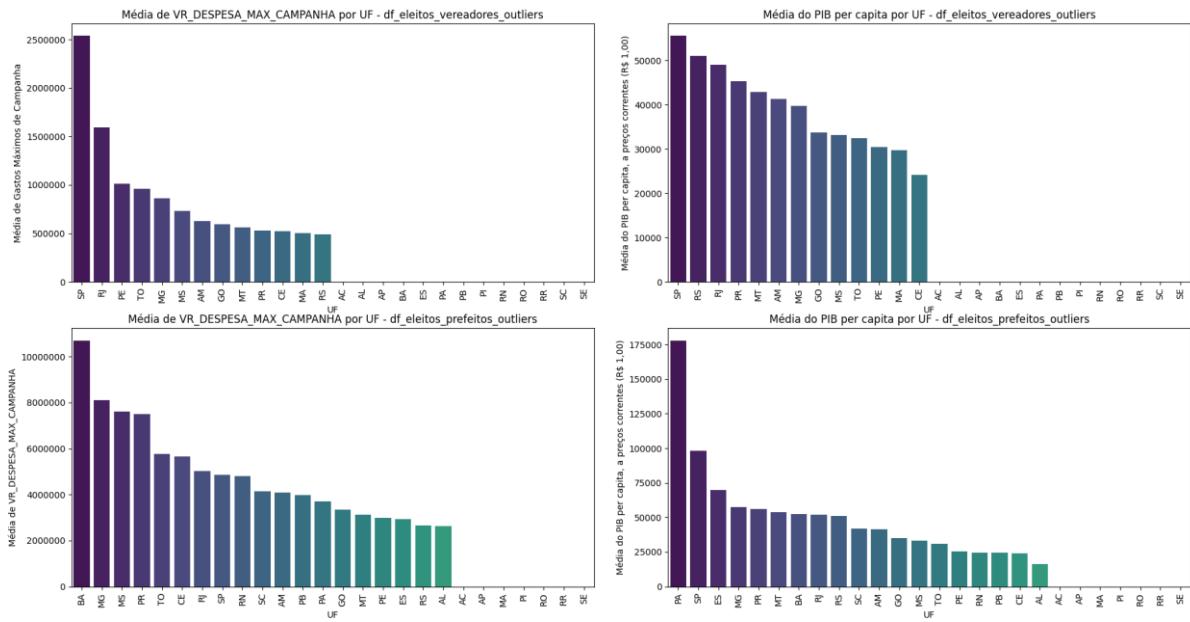
Os resultados mostram uma grande variação nas despesas de campanha entre os municípios brasileiros, tanto para vereadores quanto para prefeitos.

Capitais geralmente têm despesas de campanha mais altas do que municípios não capitais.

A análise fornece informações valiosas sobre como fatores como PIB e porte populacional podem influenciar as despesas de campanha em eleições municipais, e esses insights podem ser úteis para estratégias políticas futuras e alocação de recursos de campanha destes outliers.

### 6.5.4 Relação Entre Média de Despesas de Campanha, PIB e UF

Nesta análise é explorada a relação entre a média de despesas de campanha, o Produto Interno Bruto (PIB) per capita e as Unidades Federativas (UF) nas eleições municipais brasileiras de 2020.



Com base nos resultados da análise, podemos tirar algumas conclusões importantes relacionadas à variável alvo 'VR\_DESPESA\_MAX\_CAMPANHA':

#### Média de VR\_DESPESA\_MAX\_CAMPANHA por UF - df\_eleitos\_vereadores\_outliers:

- O estado de São Paulo (SP) tem a maior média de despesas de campanha, seguido por Rio de Janeiro (RJ), Pernambuco (PE), e Tocantins (TO).
- O Produto Interno Bruto (PIB) per capita varia consideravelmente entre esses estados, com São Paulo e Rio de Janeiro tendo os valores mais altos.

#### Média de VR\_DESPESA\_MAX\_CAMPANHA por UF - df\_eleitos\_prefeitos\_outliers:

- O estado da Bahia (BA) tem a maior média de despesas de campanha para prefeitos, seguido por Minas Gerais (MG), Mato Grosso do Sul (MS), Paraná (PR), e Tocantins (TO).
- O PIB per capita também varia entre esses estados, sendo a Bahia o estado com a média mais alta.

#### Média de PIB per capita por UF - df\_eleitos\_vereadores\_outliers:

- O PIB per capita não parece estar diretamente relacionado com a média de despesas de campanha para vereadores, já que estados com PIB per capita alto e baixo têm médias de despesas de campanha variáveis.

#### Média de PIB per capita por UF - df\_eleitos\_prefeitos\_outliers:

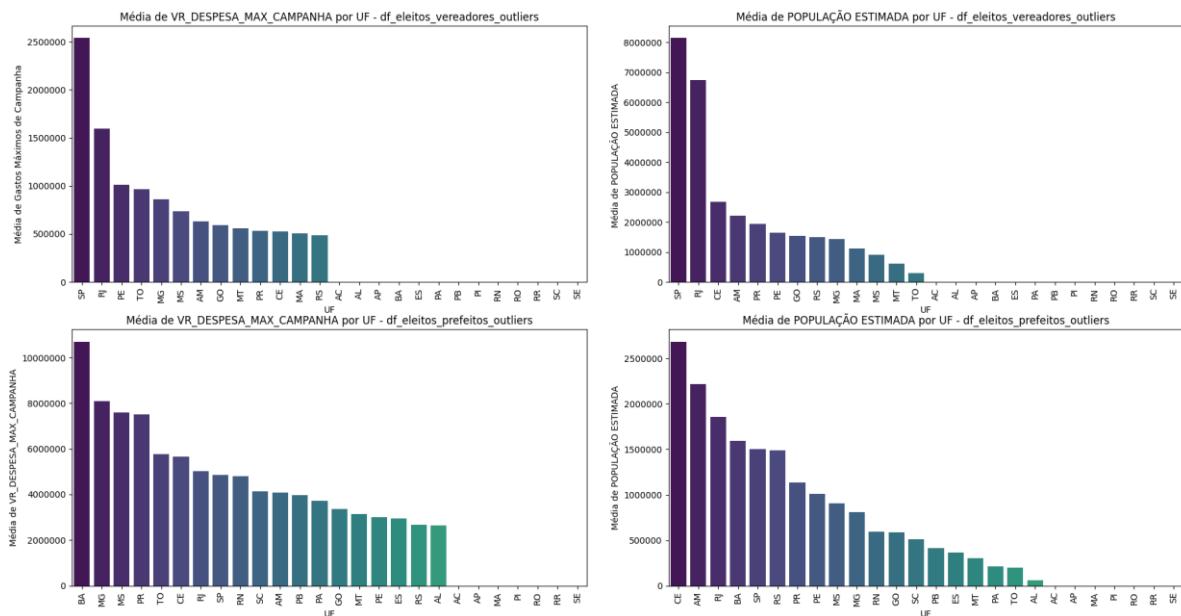
- O PIB per capita pode estar mais correlacionado com a média de despesas de campanha para prefeitos, mas ainda há variações significativas.

### Insights Gerais:

Esses resultados sugerem que, embora haja uma tendência de que estados com PIB per capita mais alto tenham maiores médias de despesas de campanha, **essa relação não é tão direta e clara**. Outros fatores, como estratégia de campanha, competição eleitoral e características políticas locais, também podem desempenhar um papel importante na determinação das despesas de campanha deste *outliers*.

#### 6.5.5 Variação das Despesas de Campanha e População Estimada por UF

Esta análise examina a média das despesas de campanha (VR\_DESPESA\_MAX\_CAMPANHA) e da população estimada por unidade federativa (UF) no contexto das eleições municipais de 2020.



Abaixo estão algumas interpretações dos resultados obtidos

Para o conjunto de dados 'df\_eleitos\_vereadores\_outliers':

- A maior média de 'VR\_DESPESA\_MAX\_CAMPANHA' ocorreu em São Paulo (SP) com aproximadamente R\$ 2.539.233,01, seguido pelo Rio de Janeiro (RJ) com R\$ 1.594.577,38.
- A maior média de 'POPULAÇÃO ESTIMADA' também foi em São Paulo (SP) com cerca de 8.148.537,91 habitantes.
- Observa-se uma variabilidade significativa nos valores de 'VR\_DESPESA\_MAX\_CAMPANHA' entre diferentes estados (UFs).

**Para o conjunto de dados 'df\_eleitos\_prefeitos\_outliers':**

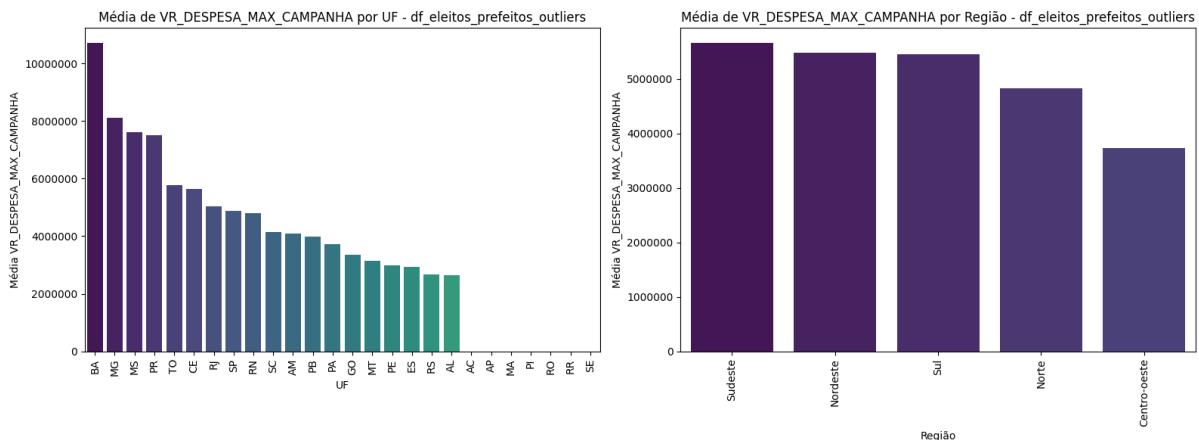
- A maior média de 'VR\_DESPESA\_MAX\_CAMPANHA' ocorreu na Bahia (BA) com aproximadamente R\$ 10.706.462,74, seguido por Minas Gerais (MG) com cerca de R\$ 8.100.372,73 e Mato Grosso do Sul (MS) com R\$ 7.609.782,17.
- A maior média de 'POPULAÇÃO ESTIMADA' também foi na Bahia (BA) com cerca de 1.595.500 habitantes.
- Novamente, há uma variabilidade significativa nos valores de 'VR\_DESPESA\_MAX\_CAMPANHA' entre diferentes estados (UFs).
- Além disso, é importante notar que alguns estados (UFs) têm valores ausentes (NaN) para 'VR\_DESPESA\_MAX\_CAMPANHA' e 'POPULAÇÃO ESTIMADA'.

#### Insights Gerais:

Essas análises iniciais fornecem insights sobre a distribuição das despesas de campanha e a população estimada em diferentes estados (UFs) no contexto das eleições municipais brasileiras de 2020. **A variabilidade nos valores sugere que fatores regionais e econômicos podem desempenhar um papel importante na determinação das despesas de campanha destes outliers.**

#### 6.5.6 Variação nas Despesas de Campanha de Candidatos Eleitos por Estado e Região

Esta análise explora a variação nas despesas de campanha de candidatos eleitos a prefeito em diferentes estados e regiões do Brasil após a identificação dos outliers.



Abaixo estão algumas interpretações dos resultados obtidos:

#### Média de VR\_DESPESA\_MAX\_CAMPANHA por Estado - df\_eleitos\_prefeitos\_outliers:

- Bahia (BA) tem a maior média de despesas de campanha, com um valor médio de aproximadamente R\$ 10.7 milhões.
- Minas Gerais (MG) segue com a segunda maior média, em torno de R\$ 8.1 milhões.
- Mato Grosso do Sul (MS), Paraná (PR) e Tocantins (TO) têm médias em torno de R\$ 7.6 a 7.5 milhões.
- Outros estados também têm médias significativas, incluindo Ceará (CE), Rio de Janeiro (RJ), São Paulo (SP), Rio Grande do Norte (RN), Santa Catarina (SC), e assim por diante.

- Alguns estados, como Acre (AC), Amapá (AP), Amazonas (AM), Pará (PA), Rondônia (RO), Roraima (RR), Tocantins (TO), não têm informações disponíveis (NaN) na média de despesas de campanha dos outliers.

Média de VR\_DESPESA\_MAX\_CAMPANHA por Região - df\_eleitos\_prefeitos\_outliers:

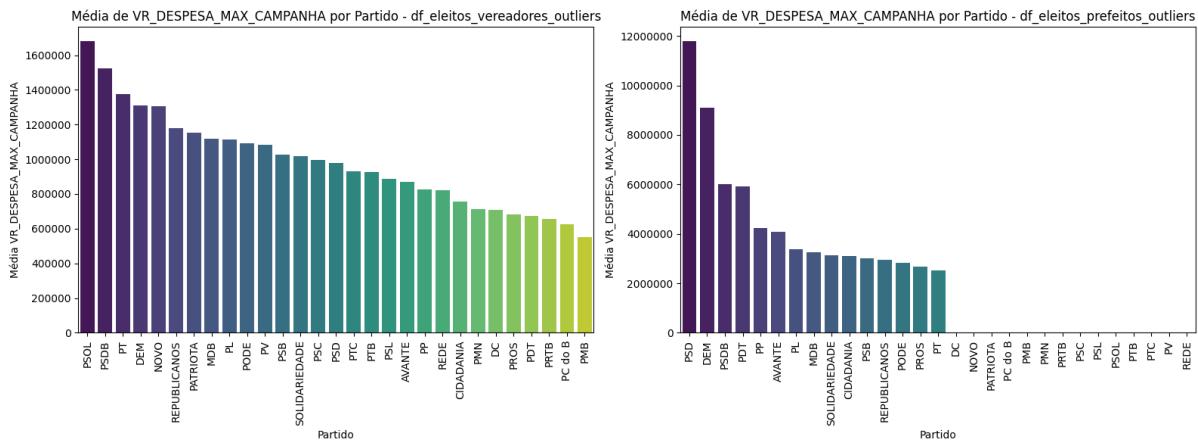
- Sudeste tem a maior média de despesas de campanha entre todas as regiões, com um valor médio de aproximadamente R\$ 5.7 milhões.
  - Nordeste vem em segundo lugar, com uma média de cerca de R\$ 5.5 milhões.
  - Sul está próxima da região Nordeste, com média de cerca de R\$ 5.5 milhões.
  - Norte possui a menor média, mas ainda significativa, de aproximadamente R\$ 4.8 milhões.
  - Centro-Oeste tem a menor média entre todas as regiões, com um valor médio de cerca de R\$ 3.7 milhões.

## **Insights Gerais:**

Esses resultados indicam que as despesas de campanha outliers variam significativamente entre estados e regiões do Brasil dentre candidatos eleitos a prefeito.

#### 6.5.7 Variação nas Médias de Gastos de Campanha por Partido

Nesta análise, é explorada a variação nas médias de gastos de campanha por partido político em relação às eleições municipais brasileiras de 2020.



Aqui estão algumas das conclusões e insights que podem ser extraídos desses resultados:

**Diferenças nas Médias de Gastos por Partido:** Os resultados mostram que os partidos políticos têm médias de gastos de campanha bastante variáveis. Por exemplo, para vereadores, o partido PSOL tem a maior média de gastos, enquanto para prefeitos, o PSD lidera em termos de média de gastos. Essa diferença pode ser explorada para entender as estratégias de financiamento de diferentes partidos em eleições municipais.

**Impacto dos Outliers:** É evidente que a presença de outliers afeta significativamente as médias de gastos por partido. Quando outliers estão presentes, as médias podem ser distorcidas, o que pode

levar a conclusões errôneas. Portanto, a remoção ou tratamento adequado de outliers é importante para obter uma visão mais precisa das tendências nos gastos de campanha.

**Partidos sem Dados:** Alguns partidos não têm dados disponíveis para prefeitos ou vereadores, como DC, NOVO, PATRIOTA, entre outros. Isso pode ser devido à falta de candidatos eleitos desses partidos em algumas áreas, e essa ausência de dados deve ser considerada ao realizar análises ou previsões.

**Variação nas Estratégias de Gastos:** A variação nas médias de gastos entre partidos pode indicar diferentes estratégias de financiamento e campanha. Partidos com médias de gastos mais altas podem estar investindo mais em campanhas eleitorais, enquanto partidos com médias mais baixas podem estar seguindo estratégias de gastos mais conservadoras.

#### **Insights Gerais:**

Em resumo, a análise das médias de gastos de campanha por partido político e a consideração do impacto dos outliers são passos importantes na compreensão do financiamento de campanhas eleitorais nas eleições municipais de 2020 no Brasil.

## 6.6 Considerações Finais

O estudo em questão proporcionou uma análise abrangente sobre o impacto dos outliers em modelos de regressão para a previsão do valor máximo de despesas de campanha de candidatos eleitos nas eleições municipais brasileiras de 2020, considerando como variáveis explicativas o Produto Interno Bruto (PIB) municipal e o porte populacional dos municípios.

A seguir, destacam-se as principais conclusões e implicações dessas análises.

**Contextualização Significativa:** A contextualização inicial foi essencial para compreender a relevância desse estudo, considerando o contexto desafiador das eleições municipais de 2020 no Brasil, marcadas pela pandemia de COVID-19 e pela mudança no financiamento das campanhas políticas.

**Impacto dos Outliers:** O estudo demonstrou que a presença de outliers nos dados de despesas de campanha teve um impacto significativo nos modelos de regressão. A remoção desses valores atípicos resultou em modelos mais precisos e confiáveis, com melhor capacidade de previsão.

**Modelos de Regressão Sólidos:** Independentemente da presença de outliers, os modelos de regressão utilizados neste estudo mostraram um desempenho geral excelente, com altos valores de coeficiente de determinação ( $R^2$ ), indicando uma boa capacidade de explicar a variabilidade nos gastos de campanha.

**Influência de Fatores Econômicos e Demográficos:** Ficou claro que fatores econômicos, representados pelo PIB municipal, e demográficos, como o porte populacional dos municípios, desempenharam um papel significativo na determinação das estratégias de campanha e dos resultados eleitorais em 2020.

**Diferenças Regionais e Partidárias:** O estudo revelou variações significativas nas despesas de campanha entre diferentes estados, regiões e partidos políticos. Essas diferenças podem ser valiosas para orientar estratégias políticas futuras e alocação de recursos de campanha.

**Recomendações para Modelagem:** Com base nas análises, uma recomendação importante é a remoção de outliers ao criar modelos de previsão para gastos de campanha. Isso pode melhorar a qualidade e a confiabilidade das previsões.

**Contribuição para a Compreensão Eleitoral:** O estudo contribuiu significativamente para a compreensão da dinâmica política brasileira, ao mostrar como fatores econômicos, demográficos e a presença de outliers afetaram as eleições municipais de 2020. Essas informações são valiosas para futuros estudos e estratégias políticas.

**Recomendações Futuras:** Para pesquisas futuras, seria interessante explorar mais a fundo as causas por trás das diferenças regionais e partidárias nas despesas de campanha, bem como considerar outros fatores que possam influenciar os resultados eleitorais.

Em suma, este insights valiosos sobre as eleições municipais brasileiras de 2020, destacando a importância da gestão eficiente de recursos financeiros para o sucesso das campanhas políticas. Além disso, ressaltou a relevância de modelos de regressão robustos e da identificação e tratamento de outliers ao analisar esses dados complexos. Os resultados aqui apresentados podem orientar estratégias políticas futuras e aprofundar nossa compreensão da dinâmica eleitoral no Brasil.

## 7 Links

Aqui você deve disponibilizar os links para o vídeo com sua apresentação de 5 minutos e para o repositório contendo os dados utilizados no projeto, scripts criados, etc.

Link para o vídeo: [https://youtu.be/GGy\\_JDm2wj4](https://youtu.be/GGy_JDm2wj4)

Link para o repositório (SEM DATASETS): <https://github.com/arthurmmota/TCC>

Link para o repositório (COM DATASETS):

<https://drive.google.com/drive/folders/1D1mRICRBevrZoTqqpdH0Rz39EhzN12Tk?usp=sharing>

## APÊNDICE

### Programação/Scripts

```
# Importando a biblioteca pandas para manipulação de dados
import pandas as pd

# Importando as bibliotecas matplotlib e seaborn para visualização de dados
import matplotlib.pyplot as plt
import seaborn as sns

# Configurando a paleta de cores do Seaborn para uma visualização agradável
sns.set_palette("viridis", 30)

# Definindo a paleta de cores do Seaborn como um mapa de cores (cmap) para
# uso posterior
sns.color_palette("viridis", as_cmap=True)

# Importando bibliotecas do scikit-learn para preparação de dados e modelagem
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor # Importando o modelo GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Importando a biblioteca statsmodels para análise estatística
import statsmodels.api as sm

# Importando a função ols (Ordinary Least Squares) do statsmodels para
# ajuste de modelos lineares
from statsmodels.formula.api import ols

# Exibe todas as colunas do DataFrame sem truncamento
pd.set_option('display.max_columns', None)

# Configuração para exibir os números com precisão fixa de 6 casas decimais
pd.set_option('display.float_format', '{:.4f}'.format)

pd.set_option('display.max_rows', None)

def analisar_ANOVA(df,colunas_categoricas):
    # Criar uma fórmula para a ANOVA
    formula_anova = 'VR_DESPESA_MAX_CAMPANHA ~ ' + ' + '.join(['C(' + coluna + ')' for coluna in colunas_categoricas])
    modelo_anova = ols(formula_anova, data=df).fit()

    tabela_anova = sm.stats.anova_lm(modelo_anova, typ=2)

    # Exibir os resultados da ANOVA
    print(f"Resultados da Análise de Variância (ANOVA) para as colunas: {', '.join(colunas_categoricas)}")
    print(tabela_anova)
```

```

# Função para substituir os valores da lista 'populacao_estimada_correcao'
# por uma string vazia
def substituir_valor (valor):
    for item in populacao_estimada_correcao:
        valor = valor.replace(item, '')
        valor = valor.replace(" ", "")

    return valor


def plot_boxplot(df, nome_df, coluna_nome):
    # Normaliza os dados da coluna
    coluna_normalizada = (df[coluna_nome] - df[coluna_nome].mean()) / df[coluna_nome].std()

    # Cria um boxplot da coluna normalizada usando Seaborn
    plt.figure(figsize=(8, 6))
    sns.boxplot(y=coluna_normalizada, color='red', flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markeredgecolor': 'red', 'markersize': 5})

    # Configurações adicionais do gráfico
    plt.title(f'Boxplot da Coluna {coluna_nome} (Normalizada) - {nome_df}')
    plt.ylabel(coluna_nome + ' (Normalizada)')
    plt.ticklabel_format(axis='y', style='plain', useOffset=False)

    # Mostra o gráfico
    plt.show()


def plot_boxplots(df, colunas_numericas, df_nome):
    # Calcula o número de subplots com base no número de colunas numéricas
    num_plots = len(colunas_numericas)

    # Define o número de colunas para a disposição dos subplots (pode ajustar conforme necessário)
    num_cols = 1
    num_rows = (num_plots + 1)    # Garante pelo menos uma linha

    # Cria subplots
    fig, axes = plt.subplots(num_rows, num_cols, figsize=(20, num_rows))
    fig.suptitle(f'Boxplots das Colunas Numéricas em {df_nome}', fontsize=16)

    # Certifique-se de que axes seja uma lista de eixos (axis) mesmo para
    # um único subplot
    if num_rows == 1:
        axes = [axes]
    # Configuração personalizada para os outliers
    flierprops = dict(marker='o', markerfacecolor='red', markeredgecolor='red', markersize=2, linestyle='none')

    for i, col in enumerate(colunas_numericas):
        ax = axes[i]
        sns.boxplot(data=df, x=col, orient="horizontal", ax=ax, flierprops=flierprops)
        ax.set_title(col)
        ax.set_xlabel(None)
        ax.set_xticks([])
```

```

# Remove subplots vazios, se houverem
for i in range(len(colunas_numericas), num_rows * num_cols):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()

def plotar_matriz_correlacao(df, colunas_numericas, nome_df):

    # Selecionar apenas as colunas numéricas do DataFrame
    numeric_columns = df.select_dtypes(include=['int64', 'float64'])

    # Calcular a matriz de correlação de Pearson
    correlation_matrix = df[colunas_numericas].corr()
    print(nome_df + " :")
    print(correlation_matrix['VR_DESPESA_MAX_CAMPANHA'])

    # Plotar a matriz de correlação
    plt.figure(figsize=(8, 6))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
                fmt=".2f", linewidths=0.5)
    plt.title(nome_df + ' Matriz de Correlação (Pearson) entre Variáveis
numéricas e VR_DESPESA_MAX_CAMPANHA')
    plt.yticks(fontsize=8)
    plt.xticks(fontsize=8)
    #tick_params(axis='x', rotation=90)
    plt.show()

def remover_outliers_VR_DESPESA_MAX_CAMPANHA(df, nome_df):
    # Calcula a média e o desvio padrão da coluna 'VR_DESPESA_MAX_CAMPANHA'
    media_despesa = df['VR_DESPESA_MAX_CAMPANHA'].mean()
    desvio_padrao_despesa = df['VR_DESPESA_MAX_CAMPANHA'].std()

    # Define o limiar para identificar outliers usando a regra 3-sigma
    limiar_superior = media_despesa + 3 * desvio_padrao_despesa
    limiar_inferior = media_despesa - 3 * desvio_padrao_despesa

    # Filtra os outliers em um novo DataFrame
    df_outliers = df[(df['VR_DESPESA_MAX_CAMPANHA'] < limiar_inferior) |
                      (df['VR_DESPESA_MAX_CAMPANHA'] > limiar_superior)]

    # Filtra os dados originais removendo os outliers
    df = df[(df['VR_DESPESA_MAX_CAMPANHA'] >= limiar_inferior) &
             (df['VR_DESPESA_MAX_CAMPANHA'] <= limiar_superior)]

    # Exiba informações sobre os outliers
    print(f"Número de outliers encontrados em {nome_df}: {len(df_outliers['VR_DESPESA_MAX_CAMPANHA'])}")
    print(f"Percentagem de outliers em {nome_df}:"
          f" {len(df_outliers['VR_DESPESA_MAX_CAMPANHA'])} / "
          f" {len(df['VR_DESPESA_MAX_CAMPANHA'])} * 100:.2f %")

    # Exibe estatísticas descritivas após remover os outliers
    print("Estatísticas descritivas após remover os outliers:")
    print(df['VR_DESPESA_MAX_CAMPANHA'].describe())

```

```

# O DataFrame df agora contém os dados sem outliers e df_outliers apenas os outliers
return [df, df_outliers]

def calcular_melhores_parametros(X, y):
    # Defina os hiperparâmetros que você deseja otimizar
    param_grid = {
        'n_estimators': [200, 300],
        'learning_rate': [0.1],
        'max_depth': [4, 6]
    }

    # Crie uma instância do regressor GradientBoosting
    regressor = GradientBoostingRegressor()

    # Crie um objeto GridSearchCV
    grid_search = GridSearchCV(estimator=regressor, param_grid=param_grid,
                               cv=2, n_jobs=-1, verbose=4)

    # Realize a pesquisa em grade
    grid_search.fit(X, y)

    # Exiba os melhores hiperparâmetros encontrados
    print("Melhores hiperparâmetros encontrados:")
    print(grid_search.best_params_)

    # Exiba a melhor pontuação do modelo
    print("Melhor pontuação do modelo:")
    print(grid_search.best_score_)

    # Ajuste o modelo final com os melhores hiperparâmetros
    melhor_modelo = grid_search.best_estimator_
    return [melhor_modelo]

def analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df, nome_df, analise_residuos):
    # Seleciona as colunas de recursos numéricos
    X_colunas_numericas = df[[
        'VR_DESPESA_MAX_CAMPANHA',
        'POPULAÇÃO ESTIMADA',
        'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
        'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
        'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000)',
        'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000)',
        'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
        'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
        'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)']]

    # Cria variáveis dummy para as colunas categóricas
    X_colunas_categoricas = pd.get_dummies(df[[
        'DS_CARGO',
        'TP AGREMIACAO',
        'SG_PARTIDO',
        'ST_REELEICAO',
        'capital',

```

```

'UF',
'Nome_da_Grande_Região'

]], drop_first=True)

# Combina as variáveis numéricas e categóricas codificadas
X = pd.concat([X_colunas_numericas, X_colunas_categoricas], axis=1)

# Define a variável alvo
y = df['VR_DESPESA_MAX_CAMPANHA']

# Divide o conjunto de dados em treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Treina o modelo e obtém os melhores parâmetros
modelo = calcular_melhores_parametros(X_train,y_train)

# Ajusta o modelo aos dados de treinamento
modelo[0].fit(X_train, y_train)

# Faz previsões no conjunto de teste
y_pred = modelo[0].predict(X_test)

# Avalia o desempenho do modelo

# Calcular o Erro Quadrático Médio (MSE)
erro_quadratico_medio = mean_squared_error(y_test, y_pred)
print(f"Erro Quadrático Médio (MSE): {erro_quadratico_medio}")

# Calcular o Erro Absoluto Médio (MAE)
erro_absoluto_medio = mean_absolute_error(y_test, y_pred)
print(f"Erro Absoluto Médio (MAE): {erro_absoluto_medio}")

# Calcular o Coeficiente de Determinação ( $R^2$ )
coeficiente_determinacao = r2_score(y_test, y_pred)
print(f"Coeficiente de Determinação ( $R^2$ ): {coeficiente_determinacao}")

# Calcular a Raiz do Erro Quadrático Médio (RMSE)
raiz_erro_quadratico_medio = mean_squared_error(y_test, y_pred, squared=False)
print(f"Raiz do Erro Quadrático Médio (RMSE): {raiz_erro_quadratico_medio}")

print("\n")

if analise_resisduos == "S":
    # Calcula os resíduos do modelo
    resíduos = y_test - y_pred

    # Cria um DataFrame para os resíduos
    resíduos_df = pd.DataFrame({'Previsões': y_pred, 'Resíduos': resíduos})

    # Gráfico de dispersão dos resíduos em relação às previsões usando Seaborn
    plt.figure(figsize=(8, 6))
    sns.scatterplot(data=resíduos_df, x='Previsões', y='Resíduos', color='green', alpha=0.7)
    plt.axhline(y=0, color='red', linestyle='--', linewidth=2)

```

```

plt.xlabel('Previsões')
plt.ylabel('Resíduos')
plt.title(f'Gráfico de Dispersão dos Resíduos em Função das Previsões - {nomedf}')
plt.ticklabel_format(axis='y', style='plain', useOffset=False)
plt.ticklabel_format(axis='x', style='plain', useOffset=False)

plt.show()

# Função para criar um gráfico de donut a partir dos dados fornecidos
def plot_donut(df, titulo, ax):
    # Extrai os rótulos e tamanhos dos dados
    labels = df['ST_REELEICAO']
    sizes = df['VR_DESPESA_MAX_CAMPANHA']

    # Cria o gráfico de pizza (donut)
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, pctdistance=0.85)
    ax.set_title(titulo)

    # Adiciona um círculo no centro para criar o efeito de donut
    centre_circle = plt.Circle((0, 0), 0.70, fc='white')
    ax.add_artist(centre_circle)

print('*' * 100)
print('INICIO Coleta de Dados - Processamento/Tratamento de Dados')
print('*' * 100)
print('*' * 100)
print('INICIO df_TSE_IBGE')
print('*' * 100)

# Carrega o DataFrame com dados do TSE e IBGE
df_TSE_IBGE = pd.read_csv('DATASETS/municipios_brasileiros_tse.csv', encoding='UTF-8', delimiter=';')

# Imprime as primeiras linhas do DataFrame
print(df_TSE_IBGE.head())
# Imprime informações do DataFrame
print(df_TSE_IBGE.info())
# Imprime estatísticas descritivas do DataFrame
print(df_TSE_IBGE.describe())
# Converte colunas específicas para o tipo de dado 'str'
df_TSE_IBGE['codigo_ibge'] = df_TSE_IBGE['codigo_ibge'].astype(str)
df_TSE_IBGE['codigo_tse'] = df_TSE_IBGE['codigo_tse'].astype(str)

print('*' * 100)
print('FIM df_TSE_IBGE')
print('*' * 100)

print('*' * 100)
print('INICIO df_eleitos')
print('*' * 100)

df_eleitos = pd.read_csv('DATASETS/consulta_cand_2020_BRASIL.csv', encoding='latin1', delimiter=';')
# Exibe as colunas do dataframe e informações básicas
print("Colunas do df_eleitos:")

```

```

print(df_eleitos.columns)
print("\nInformações do df_eleitos:")
print(df_eleitos.info())

# Lista das colunas desejadas
eleitos_colunas_desejadas = [
    'CD_TIPO_ELEICAO',
    'NM_TIPO_ELEICAO',
    'SG_UF',
    'SG_UE',
    'NM_UE',
    'CD_CARGO',
    'DS_CARGO',
    'CD_SITUACAO_CANDIDATURA',
    'DS_SITUACAO_CANDIDATURA',
    'TP_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'CD_SIT_TOT_TURNO',
    'DS_SIT_TOT_TURNO',
    'ST_REELEICAO',
    'VR_DESPESA_MAX_CAMPANHA'
]

# Filtra o dataframe para manter apenas as colunas desejadas
df_eleitos = df_eleitos[eleitos_colunas_desejadas]

# Converte a coluna 'SG_UE' para o tipo string, se necessário
df_eleitos['SG_UE'] = df_eleitos['SG_UE'].astype(str)

# Exibe o número de valores únicos em cada coluna
print("\nNúmero de valores únicos em cada coluna:")
print(df_eleitos.nunique())

# Lista das colunas a serem analisadas
colunas_para_analizar = [
    'CD_TIPO_ELEICAO',
    'NM_TIPO_ELEICAO',
    'CD_CARGO',
    'DS_CARGO',
    'CD_SITUACAO_CANDIDATURA',
    'DS_SITUACAO_CANDIDATURA',
    'TP_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'CD_SIT_TOT_TURNO',
    'DS_SIT_TOT_TURNO',
    'ST_REELEICAO'
]

# Itera sobre as colunas para exibir os valores únicos em cada uma
for coluna in colunas_para_analizar:
    valores_unicos = df_eleitos[coluna].unique()
    print(f"Valores únicos em {coluna}:\n{valores_unicos}\n")

# Lista dos valores de 'DS_SIT_TOT_TURNO' desejados
eleitos_selecionados_eleito = ['ELEITO', 'ELEITO POR QP', 'ELEITO POR MÉ-DIA']

```

```

# Filtra o dataframe para manter apenas as linhas com as situações desejadas
df_eleitos = df_eleitos[df_eleitos['DS_SIT_TOT_TURNO'].isin(eleitos_selecionados_eleito)]

# Exibe informações do dataframe após a primeira filtragem
print("\nInformações do dataframe após a primeira filtragem")
print("DS_SIT_TOT_TURNO:")
print(df_eleitos.info())

# Lista dos valores de 'DS_SITUACAO_CANDIDATURA' desejados
eleitos_selecionados_apto = ['APTO']

# Filtra o dataframe para manter apenas as linhas com as situações de candidatura desejadas
df_eleitos = df_eleitos[df_eleitos['DS_SITUACAO_CANDIDATURA'].isin(eleitos_selecionados_apto)]

# Exibe informações do dataframe após a segunda filtragem
print("\nInformações do dataframe após a segunda filtragem DS_SITUACAO_CANDIDATURA:")
print(df_eleitos.info())

# Lista das colunas a serem removidas
colunas_para_remover = [
    "NM_UE",
    "CD_CARGO",
    "CD_TIPO_ELEICAO",
    "NM_TIPO_ELEICAO",
    "CD_SITUACAO_CANDIDATURA",
    "DS_SITUACAO_CANDIDATURA",
    "CD_SIT_TOT_TURNO",
    "DS_SIT_TOT_TURNO"
]

# Remove as colunas especificadas do dataframe
df_eleitos = df_eleitos.drop(columns=colunas_para_remover)

# Exibe informações do dataframe após a remoção de colunas
print("\nInformações do dataframe após a remoção de colunas:")
print(df_eleitos.info())

print('*' * 100)
print('FIM df_eleitos')
print('*' * 100)

print('*' * 100)
print('INICIO df_populacao')
print('*' * 100)
# Carregar o DataFrame a partir do arquivo CSV
# Usar skiprows=1 para ignorar o cabeçalho
df_populacao = pd.read_excel('DATASETS/estimativa dou 2020.xls',
                             skiprows=1, header=0, sheet_name='Municípios')

# Exibir as 10 primeiras linhas das colunas 'COD. UF' e 'COD. MUNIC'
print(df_populacao['COD. UF'].head(10))
print(df_populacao['COD. MUNIC'].head(10))

```

```

# Identificar as colunas categóricas e numéricas
categorical_columns = df_populacao.select_dtypes(include=['object']).columns
numeric_columns = df_populacao.select_dtypes(include=['int', 'float']).columns

# Análise das variáveis categóricas
print('Análise das variáveis categóricas')

for col in categorical_columns:
    print(f"Análise de {col}:")
    print(f"Total de valores únicos: {df_populacao[col].nunique()}")
    print(f"Valores únicos: {df_populacao[col].unique()}")
    print("====")

print('# Análise das variáveis numéricas')
# Análise das variáveis numéricas
for col in numeric_columns:
    print(f"Análise de {col}:")
    print(f"Média: {df_populacao[col].mean()}")
    print(f"Desvio padrão: {df_populacao[col].std()}")
    print(f"Valor mínimo: {df_populacao[col].min()}")
    print(f"Valor máximo: {df_populacao[col].max()}")
    print("====")

# Exibir informações gerais sobre o DataFrame
print(df_populacao.info())

# Filtrar as linhas onde a coluna 'UF' está na lista 'uf_correcao'
uf_correcao = ['RO', 'AC', 'AM', 'RR', 'PA', 'AP',
                'TO', 'MA', 'PI', 'CE', 'RN', 'PB',
                'PE', 'AL', 'SE', 'BA', 'MG', 'ES',
                'RJ', 'SP', 'PR', 'SC', 'RS', 'MS',
                'MT', 'GO', 'DF']

# Filtre as linhas onde a coluna 'UF' está na lista 'uf_correcao'.
df_populacao = df_populacao[df_populacao['UF'].isin(uf_correcao)]

# Análise da coluna 'UF' após a filtragem
print(f"Análise de 'UF':")
print(f"Total de valores únicos: {df_populacao['UF'].nunique()}")
print(f"Valores únicos: {df_populacao['UF'].unique()}")
print("====")

# Exibir informações atualizadas sobre o DataFrame após a filtragem
print(df_populacao.info())
# Exibir estatísticas descritivas do DataFrame
print(df_populacao.describe())

# Tratar a coluna 'POPULAÇÃO ESTIMADA'
# Use a função `pd.to_numeric` com `errors='coerce'` para converter a coluna em números
# e lidar com valores não numéricos.
valores_nao_numericos = df_populacao[df_populacao['POPULAÇÃO ESTIMADA'].apply(pd.to_numeric, errors='coerce').isna()]

# Exibir os registros não numéricos
print(valores_nao_numericos)

```

```

# Lista de índices de valores não numéricos
lista_valores_nao_numericos = valores_nao_numericos.index.tolist()
print(lista_valores_nao_numericos)

# Valores a serem corrigidos na coluna 'POPULAÇÃO ESTIMADA'
populacao_estimada_correcao = ['(1)', '(2)', '(3)', '(4)', '(5)', '(6)', '(7)', '(8)', '(9)', '(10)', '(11)', '(12)']

# Aplicar a substituição apenas nos índices especificados em 'lista_valores_nao_numericos'
df_populacao.loc[lista_valores_nao_numericos, 'POPULAÇÃO ESTIMADA'] =
df_populacao.loc[
    lista_valores_nao_numericos, 'POPULAÇÃO ESTIMADA'].apply(substituir_valor)
print("df_populacao.loc[lista_valores_nao_numericos, 'POPULAÇÃO ESTIMADA']")
print(df_populacao.loc[lista_valores_nao_numericos])

# Usar a função pd.to_numeric para converter a coluna em números, definindo
errors='coerce'
# para tratar não numéricos como NaN.
df_populacao['POPULAÇÃO ESTIMADA'] = pd.to_numeric(df_populacao['POPULAÇÃO ESTIMADA'], errors='coerce')

# Converter as colunas para o tipo de dados 'strings'
df_populacao['UF'] = df_populacao['UF'].astype(str)
df_populacao['COD. UF'] = df_populacao['COD. UF'].astype(str).str.split('.').str[0]

# Converter a coluna 'COD. MUNIC' em strings, dividir por '.', pegar a
parte inteira
# e aplicar o preenchimento com zeros à esquerda
df_populacao['COD. MUNIC'] = df_populacao['COD. MUNIC'].astype(str)
df_populacao['COD. MUNIC'] = df_populacao['COD. MUNIC'].apply(lambda x:
str(x).replace('.', '').zfill(6))

df_populacao['NOME DO MUNICÍPIO'] = df_populacao['NOME DO MUNICÍPIO'].astype(str)

# Verificar os tipos de dados atualizados
print(df_populacao.dtypes)

# Exibir estatísticas descritivas atualizadas do DataFrame
print(df_populacao.describe())
# Exibir informações atualizadas sobre o DataFrame
print(df_populacao.info())
# Exibir as colunas do DataFrame
print(df_populacao.columns)

print('*' * 100)
print('FIM df_populacao')
print('*' * 100)

print('*' * 100)
print('INICIO df_renda')
print('*' * 100)

```

```

# Carrega o conjunto de dados do PIB dos Municípios
df_renda = pd.read_excel('DATASETS/PIB dos Municípios - base de dados 2010-2020.xls',
                         skiprows=0, header=0, sheet_name='PIB_dos_Municípios')

# Exibe as colunas do DataFrame
print("Colunas do df_renda:")
print(df_renda.columns)
# Exibe informações sobre o DataFrame
print("Informações do df_renda:")
print(df_renda.info())


# Filtra o DataFrame para manter apenas os dados do ano de 2020
df_renda = df_renda[df_renda['Ano'] == 2020]

# Exibe informações atualizadas do DataFrame
print("Informações do DataFrame após filtragem:")
print(df_renda.info())


# Seleciona as colunas de interesse
renda_colunas_selecionadas = [
    'Nome da Grande Região',
    'Código do Município',
    'Valor adicionado bruto da Agropecuária, \nna preços correntes\n(R$ 1.000)',
    'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)',
    'Valor adicionado bruto dos Serviços,\nna preços correntes \n- exceto Administração, defesa, educação e saúde públicas e segurança social\n(R$ 1.000)',
    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social,\nna preços correntes\n(R$ 1.000)',
    'Valor adicionado bruto total,\nna preços correntes\n(R$ 1.000)',
    'Impostos, líquidos de subsídios, sobre produtos,\nna preços correntes\n(R$ 1.000)',
    'Produto Interno Bruto per capita,\nna preços correntes\n(R$ 1,00)'
]
# Mantém apenas as colunas de interesse no DataFrame
df_renda = df_renda[renda_colunas_selecionadas]
# Renomeia as colunas para nomes mais legíveis
df_renda.rename(columns={
    'Valor adicionado bruto da Agropecuária, \nna preços correntes\n(R$ 1.000)': 'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',

    'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)': 'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',

    'Valor adicionado bruto dos Serviços,\nna preços correntes \n- exceto Administração, ':
        'defesa, educação e saúde públicas e segurança social\n(R$ 1.000)': 'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa,':
            'educação e saúde públicas e segurança social (R$ 1.000)',

})

```

```

'Valor adicionado bruto da Administração, defesa, educação e saúde pú-
blicas e seguridade social,'
    ' \na preços correntes\n(R$ 1.000)':
        'Valor adicionado bruto da Administração, defesa, educação e saúde
públicas e seguridade social,'
            ' a preços correntes (R$ 1.000)',

'Valor adicionado bruto total, \na preços correntes\n(R$ 1.000)':
    'Valor adicionado bruto total, a preços correntes (R$ 1.000)',

'Impostos, líquidos de subsídios, sobre produtos, \na preços corren-
tes\n(R$ 1.000)':
    'Impostos, líquidos de subsídios, sobre produtos, a preços corren-
tes (R$ 1.000)',

'Produto Interno Bruto per capita, \na preços correntes\n(R$ 1,00)':
    'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'
}, inplace=True)

# Renomeia a coluna 'Nome da Grande Região'
df_renda = df_renda.rename(columns={'Nome da Grande Região':
'Nome_da_Grande_Região'})

# Converte a coluna 'Código do Município' para tipo string
df_renda['Código do Município'] = df_renda['Código do Município'].as-
type(str)

# Exibe informações atualizadas do DataFrame
print("Informações do df_renda após seleção e renomeação de colunas:")
print(df_renda.info())

# Exibe estatísticas descritivas do DataFrame
print("Estatísticas Descritivas do df_renda:")
print(df_renda.describe())

print('*' * 100)
print('FIM df_renda')
print('*' * 100)

print('*' * 100)
print('Inicio df_dados_demograficos = df_renda + df_populacao')
print('*' * 100)

# Cria uma coluna temporária em ambos os DataFrames para facilitar a junção
df_populacao['COD. UF + COD. MUNIC'] = (df_populacao['COD. UF'] + df_popu-
lacao['COD. MUNIC']).astype(str).str[:-1]
df_populacao['COD. UF + COD. MUNIC'] = df_populacao['COD. UF + COD. MU-
NIC'].astype(str)
print("Exemplo de dados nas colunas 'COD_UF', 'COD_MUNIC' e 'COD. UF + COD.
MUNI':")
print(df_populacao['COD. UF'].head(5))
print(df_populacao['COD. MUNIC'].head(5))
print(df_populacao['COD. UF + COD. MUNIC'].head(5))

# Seleciona as colunas desejadas em cada DataFrame
colunas_df_populacao = ['UF', 'NOME DO MUNICÍPIO', 'POPULAÇÃO ESTIMADA']
colunas_df_renda = df_renda.columns.tolist()

```

```

# Realiza a junção entre os DataFrames usando a coluna temporária como
chave
df_dados_demograficos = pd.merge(df_renda, df_populacao, left_on='Código do
Município', right_on='COD. UF + COD. MUNIC', how='inner')[

    colunas_df_renda + colunas_df_populacao]

print("Informações sobre o df_dados_demograficos:")
print(df_dados_demograficos.info())

# Imprime as 10 primeiras linhas do DataFrame resultante
print("As 5 Primeiras Linhas do DataFrame Combinado:")
print(df_dados_demograficos.head(5))

print('*' * 100)
print('FIM df_dados_demograficos = df_renda + df_populacao')
print('*' * 100)

print('*' * 100)
print('Início df_dados_demograficos completo')
print('*' * 100)

# Define a ordem desejada das colunas em uma lista
nova_ordem_colunas = [
    'NOME DO MUNICÍPIO',
    'Código do Município',
    'UF',
    'Nome_da_Grande_Região',
    'POPULAÇÃO ESTIMADA',
    'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
    'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
    'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'
]

# Reordena as colunas do DataFrame
df_dados_demograficos = df_dados_demograficos[nova_ordem_colunas]
print('Nomes das colunas após reordenamento:')
print(df_dados_demograficos.columns)

# Imprime informações sobre o DataFrame
print('Informações sobre o DataFrame:')
print(df_dados_demograficos.info())

print('*' * 100)
print('FIM df_dados_demograficos completo')
print('*' * 100)

print('*' * 100)

```

```

print('Início df_eleitos')
print('*' * 100)

# Selecionando as colunas desejadas para o DataFrame df_eleitos
colunas_df_eleitos = [
    'DS_CARGO',
    'TP AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'ST_REELEICAO',
    'VR_DESPESA_MAX_CAMPANHA']
# Selecionando as colunas desejadas para o DataFrame df_TSE_IBGE
colunas_df_TSE_IBGE = ['capital', 'codigo_ibge']

# Realizando a junção entre os DataFrames usando a coluna 'SG_UE' e 'codigo_tse' como chave
df_eleitos_IBGE = pd.merge(df_eleitos, df_TSE_IBGE, left_on='SG_UE',
                           right_on='codigo_tse', how='inner')[colunas_df_eleitos + colunas_df_TSE_IBGE]

# Campos preenchidos com #NULO significam que a informação está em branco no banco de dados.
# O correspondente para #NULO nos campos numéricos é -1;

# Separando os dados em duas categorias: >= 0 e < 0
valores_positivos = df_eleitos_IBGE[df_eleitos_IBGE['VR_DESPESA_MAX_CAMPANHA'] >= 0]
valores_negativos = df_eleitos_IBGE[df_eleitos_IBGE['VR_DESPESA_MAX_CAMPANHA'] < 0]

# Contando os valores em cada categoria
contagem_positivos = len(valores_positivos)
contagem_negativos = len(valores_negativos)

print("Contagem de Valores de VR_DESPESA_MAX_CAMPANHA")
print(f"Contagem_positivos: {contagem_positivos}")
print(f"Contagem_negativos: {contagem_negativos}")

# Criando um gráfico de barras para visualizar a contagem
categorias = ['Valores >= 0', 'Valores < 0']
contagem = [contagem_positivos, contagem_negativos]

# Cria um gráfico de barras usando Seaborn
sns.barplot(x=categorias, y=contagem)

# Define rótulos e título
plt.xlabel('Categorias')
plt.ylabel('Contagem')
plt.title('Contagem de Valores de VR_DESPESA_MAX_CAMPANHA')
plt.ticklabel_format(axis='y', style='plain', useOffset=False)

# Exibe o gráfico
plt.show()

# Filtrando apenas os dados relacionados aos vice-prefeitos antes da remoção de valores negativos
print("DataFrame df_eleitos_vices antes da remoção de valores negativos:")
df_eleitos_vices = df_eleitos_IBGE[df_eleitos_IBGE['DS_CARGO'] == 'VICE-PREFEITO']

```

```

print(df_eleitos_vices.info())

# Removendo valores negativos do DataFrame df_eleitos_com_IBGE
df_eleitos_IBGE = df_eleitos_IBGE[df_eleitos_IBGE['VR_DESPESA_MAX_CAMPA-
NHA'] >= 0]

# Filtrando apenas os dados relacionados aos vice-prefeitos após a remoção
# de valores negativos
print("DataFrame df_eleitos_vices depois da remoção de valores negativos:")
df_eleitos_vices = df_eleitos_IBGE[df_eleitos_IBGE['DS_CARGO'] == 'VICE-
PREFEITO']
print(df_eleitos_vices.info())

# Exibindo informações sobre o DataFrame df_eleitos_IBGE
print("Informações sobre o DataFrame df_eleitos_IBGE:")
print(df_eleitos_IBGE.info())

print('*' * 100)
print('FIM df_eleitos')
print('*' * 100)

print('*' * 100)
print('INICIO df_eleitos_dados_demograficos = df_eleitos_IBGE + df_da-
dos_demograficos')
print('*' * 100)

# Obtém as listas de colunas dos DataFrames
colunas_df_eleitos_IBGE = df_eleitos_IBGE.columns.tolist()
colunas_df_dados_demograficos = df_dados_demograficos.columns.tolist()

# Realiza a junção entre os DataFrames
df_eleitos_dados_demograficos = pd.merge(df_eleitos_IBGE, df_dados_demogra-
ficos, left_on='codigo_ibge',
                                         right_on='Código do Município',
                                         how='inner')[colunas_df_elei-
tos_IBGE + colunas_df_dados_demograficos]

# Remove a coluna 'codigo_ibge'
df_eleitos_dados_demograficos = df_eleitos_dados_demograficos.drop(co-
lumns='codigo_ibge')

# Lista de colunas para definir como categóricas
colunas_categoricas = [
    'DS_CARGO',
    'TF_AGREMIACAO',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'ST_REELEICAO',
    'capital',
    'UF',
    'Nome_da_Grande_Região'
]

# Define as colunas como categóricas
df_eleitos_dados_demograficos[colunas_categoricas] = df_eleitos_dados_demo-
graficos[colunas_categoricas].astype('category')

# Lista de colunas para definir como numéricas

```

```

colunas_numericas = [
    'VR_DESPESA_MAX_CAMPANHA',
    'POPULAÇÃO ESTIMADA',
    'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
    'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
    'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'
]

# Define a ordem desejada das colunas
nova_ordem_colunas = [
    'NOME DO MUNICÍPIO',
    'Código do Município',
    'UF',
    'Nome_da_Grande_Região',
    'POPULAÇÃO ESTIMADA',
    'capital',
    'DS_CARGO',
    'VR_DESPESA_MAX_CAMPANHA',
    'SG_PARTIDO',
    'NM_PARTIDO',
    'TP_AGREMIACAO',
    'ST_REELEICAO',
    'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000)',
    'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)',
    'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
    'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
    'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'
]

# Reordena as colunas do DataFrame
df_eleitos_dados_demograficos = df_eleitos_dados_demograficos[nova_ordem_colunas]

# Usa pd.to_numeric para definir as colunas como numéricas
df_eleitos_dados_demograficos[colunas_numericas] = df_eleitos_dados_demograficos[colunas_numericas].apply(pd.to_numeric,
errors='coerce')

# Ordenar o DataFrame com base em várias colunas

```

```

df_eleitos_dados_demograficos = df_eleitos_dados_demograficos.sort_values(by=['UF', 'NOME DO MUNICÍPIO', 'SG_PARTIDO',
'DS_CARGO'], ascending=[True, True, True, True])

# Imprime informações sobre o DataFrame após a definição dos tipos de colunas
print("Informações sobre o df_eleitos_dados_demograficos após a definição dos tipos das colunas:")
print(df_eleitos_dados_demograficos.info())

# Verifica os tipos de dados das colunas após a conversão
print("Tipos de dados das colunas após a conversão:")
print(df_eleitos_dados_demograficos.dtypes)

# Realiza uma análise exploratória básica
# Exibe algumas linhas aleatórias do DataFrame
print("Amostra aleatória do df_eleitos_dados_demograficos:")
print(df_eleitos_dados_demograficos.sample(5))

print('*' * 100)
print('FIM df_eleitos_dados_demograficos = df_eleitos_IBGE + df_dados_demograficos')
print('*' * 100)

print('*' * 100)
print('FIM Coleta de Dados - Processamento/Tratamento de Dados')
print('*' * 100)

print('*' * 100)
print('INICIO Análise e Exploração dos Dados - Criação de Modelos de Machine Learning')
print('*' * 100)

# Criar DataFrames df_eleitos_prefeitos e df_eleitos_vereadores separados com base na coluna 'DS_CARGO'
df_eleitos_prefeitos = df_eleitos_dados_demograficos[df_eleitos_dados_demograficos['DS_CARGO'] == 'PREFEITO']
df_eleitos_vereadores = df_eleitos_dados_demograficos[df_eleitos_dados_demograficos['DS_CARGO'] == 'VEREADOR']

# Exibir o tamanho dos DataFrames
print("Tamanho de df_eleitos_prefeitos:", df_eleitos_prefeitos.shape)
print("Tamanho de df_eleitos_vereadores:", df_eleitos_vereadores.shape)

# Exibir informações sobre os DataFrames
print("\nInformações sobre df_prefeitos:")
print(df_eleitos_prefeitos.info())
print("\nInformações sobre df_vereadores:")
print(df_eleitos_vereadores.info())

# Amostra de 5 registros aleatórios de cada DataFrame
print("\nAmostra de 5 registros de df_prefeitos:")

```

```

print(df_eleitos_prefeitos.sample(5))
print("\nAmostra de 5 registros de df_vereadores:")
print(df_eleitos_vereadores.sample(5))

#Variáveis Categóricas:
# Identificar e imprimir as variáveis categóricas
categorical_columns = df_eleitos_dados_demograficos.select_dtypes(include='category').columns
print("Variáveis Categóricas: ")
for col in categorical_columns:
    print(f"Coluna: {col}")
    print(df_eleitos_dados_demograficos[col].value_counts())
    print("\n")

# Realizar análise ANOVA nas colunas categóricas
print("Análise ANOVA para df_eleitos_dados_demograficos (colunas categóricas)")
analisar_ANOVA(df_eleitos_dados_demograficos,colunas_categoricas)

print("Análise ANOVA para df_prefeitos (colunas categóricas)")
analisar_ANOVA(df_eleitos_prefeitos,colunas_categoricas)

print("Análise ANOVA para df_vereadores (colunas categóricas)")
analisar_ANOVA(df_eleitos_vereadores,colunas_categoricas)

#Variáveis Numéricas:
# Identificar e imprimir as variáveis numéricas
print("Variáveis Numéricas :")
# Exibir um resumo estatístico das colunas numéricas do DataFrame
print("Resumo estatístico das colunas numéricas de df_eleitos_dados_demograficos:")
print(df_eleitos_dados_demograficos.describe())

# Plotar matrizes de correlação para os DataFrames
plotar_matriz_correlacao(df_eleitos_dados_demograficos,colunas_numericas,"df_eleitos_dados_demograficos")
plotar_matriz_correlacao(df_eleitos_prefeitos,colunas_numericas,"df_eleitos_prefeitos")
plotar_matriz_correlacao(df_eleitos_vereadores,colunas_numericas,"df_eleitos_vereadores")

# Plotar boxplots para os DataFrames
plot_boxplots(df_eleitos_dados_demograficos,colunas_numericas,"df_eleitos_dados_demograficos")
plot_boxplots(df_eleitos_prefeitos,colunas_numericas,"df_eleitos_prefeitos")
plot_boxplots(df_eleitos_vereadores,colunas_numericas,"df_eleitos_vereadores")

# Plotar boxplot específico para a coluna 'VR_DESPESA_MAX_CAMPANHA'
plot_boxplot(df_eleitos_dados_demograficos,"df_eleitos_dados_demograficos",'VR_DESPESA_MAX_CAMPANHA')
plot_boxplot(df_eleitos_prefeitos,"df_eleitos_prefeitos",'VR_DESPESA_MAX_CAMPANHA')
plot_boxplot(df_eleitos_vereadores,"df_eleitos_vereadores",'VR_DESPESA_MAX_CAMPANHA')

```

```

# Imprimir estatísticas descritivas para a coluna 'VR_DESPESA_MAX_CAMPANHA'
em cada DataFrame
print("Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_eleitos_dados_demo-
graficos:")
print(df_eleitos_dados_demograficos['VR_DESPESA_MAX_CAMPANHA'].describe())
print("Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_prefeitos:")
print(df_eleitos_prefeitos['VR_DESPESA_MAX_CAMPANHA'].describe())
print("Estatísticas para VR_DESPESA_MAX_CAMPANHA em df_vereadores:")
print(df_eleitos_vereadores['VR_DESPESA_MAX_CAMPANHA'].describe())

print('*' * 100)
print('INICIO analise_de_modelo_VR_DESPESA_MAX_CAMPANHA')
print('*' * 100)

#df_eleitos_dados_demograficos

# Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DES-
PESA_MAX_CAMPANHA com outliers e com análise de resíduos
print("Análise do modelo de regressão para df_eleitos_dados_demograficos
VR_DESPESA_MAX_CAMPANHA com outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_dados_demografi-
cos,"df_eleitos_dados_demograficos com outliers","S")

#Remover Outliers de df_eleitos_dados_demograficos e criar o dataframe
df_eleitos_dados_demograficos_outliers = re-
mover_outliers_VR_DESPESA_MAX_CAMPANHA(df_eleitos_dados_demografi-
cos,"df_eleitos_dados_demograficos com outliers")

# Análise do modelo de regressão para df_eleitos_dados_demograficos VR_DES-
PESA_MAX_CAMPANHA sem outliers e sem análise de resíduos
print("Análise do modelo de regressão para df_eleitos_dados_demograficos
VR_DESPESA_MAX_CAMPANHA sem outliers")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_dados_demografi-
cos,"df_eleitos_dados_demograficos sem outliers","N")

plot_boxplots(df_eleitos_dados_demograficos,colunas_numericas,"df_elei-
tos_dados_demograficos sem outliers")
plot_boxplot(df_eleitos_dados_demograficos,"df_eleitos_dados_demograficos
sem outliers",'VR_DESPESA_MAX_CAMPANHA')

#df_eleitos_prefeitos

print("Análise do modelo de regressão para df_eleitos_prefeitos VR_DES-
PESA_MAX_CAMPANHA com outliers:")
# Análise do modelo de regressão para df_eleitos_prefeitos VR_DES-
PESA_MAX_CAMPANHA com outliers e com análise de resíduos
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_prefeitos,"df_elei-
tos_prefeitos com outliers","S")

#Remover Outliers de df_eleitos_prefeitos e criar o dataframe df_elei-
tos_prefeitos_outliers contendo apenas o outliers correspondentes
df_eleitos_prefeitos, df_eleitos_prefeitos_outliers = remover_outli-
ers_VR_DESPESA_MAX_CAMPANHA(df_eleitos_prefeitos,"df_eleitos_prefeitos com
outliers")

```

```

# Análise do modelo de regressão para df_eleitos_prefeitos VR_DES-
PESA_MAX_CAMPANHA sem outliers e sem análise de resíduos
print("Análise do modelo de regressão para df_eleitos_prefeitos VR_DES-
PESA_MAX_CAMPANHA sem outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_prefeitos,"df_elei-
tos_prefeitos sem outliers","N")

plot_boxplots(df_eleitos_prefeitos,colunas_numericas , 'df_eleitos_prefeitos
sem outliers')
plot_boxplot(df_eleitos_prefeitos, "df_eleitos_prefeitos sem outliers"
,'VR_DESPESA_MAX_CAMPANHA')

#df_eleitos_vereadores
# Análise do modelo de regressão para df_eleitos_vereadores VR_DES-
PESA_MAX_CAMPANHA com outliers e com análise de resíduos
print("Análise do modelo de regressão para df_eleitos_vereadores VR_DES-
PESA_MAX_CAMPANHA com outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_vereadores,"df_elei-
tos_vereadores com outliers","S")

#Remover Outliers de df_eleitos_vereadores e criar o dataframe df_elei-
tos_vereadores_outliers contendo apenas os outliers correspondentes
df_eleitos_vereadores, df_eleitos_vereadores_outliers = remover_outli-
ers_VR_DESPESA_MAX_CAMPANHA(df_eleitos_vereadores,"df_eleitos_vereadores
com outliers")

# Análise do modelo de regressão para df_eleitos_vereadores VR_DES-
PESA_MAX_CAMPANHA sem outliers e sem análise de resíduos
print("Análise do modelo de regressão para df_eleitos_vereadores VR_DES-
PESA_MAX_CAMPANHA sem outliers:")
analise_de_modelo_VR_DESPESA_MAX_CAMPANHA(df_eleitos_vereadores,"df_elei-
tos_vereadores sem outliers","N")

plot_boxplots(df_eleitos_vereadores,colunas_numericas,'df_eleitos_veread-
ores sem outliers')
plot_boxplot(df_eleitos_vereadores,"df_eleitos_vereadores" , 'VR_DES-
PESA_MAX_CAMPANHA')

print('*' * 100)
print('FIM analise_de_modelo_VR_DESPESA_MAX_CAMPANHA')
print('*' * 100)

print('*' * 100)
print('INICIO Análise Outliers')
print('*' * 100)

# Calcula a média de 'VR_DESPESA_MAX_CAMPANHA' por município
agrupado_por_municipio_vereadores = df_eleitos_vereadores_outli-
ers.groupby('NOME DO MUNICÍPIO')[['VR_DESPESA_MAX_CAMPANHA']].mean().re-
set_index().copy()

# Ordena o DataFrame resultante em ordem decrescente com base na média de
despesas de campanha
agrupado_por_municipio_vereadores = agrupado_por_municipio_vereado-
res.sort_values(by='VR DESPESA MAX CAMPANHA', ascending=False)

```

```

# Calcula a média de 'VR_DESPESA_MAX_CAMPANHA' por município, distinguindo
entre capitais e não capitais
agrupado_por_municipio_vereadores_capital = df_eleitos_vereadores_outliers.groupby(['capital', 'NOME DO MUNICÍPIO'])['VR_DESPESA_MAX_CAMPANHA'].mean().reset_index()

# Calcula a média de 'VR_DESPESA_MAX_CAMPANHA' em capitais
agrupado_por_municipio_vereadores_capital_1 = agrupado_por_municipio_vereadores_capital[
    (agrupado_por_municipio_vereadores_capital['capital'] == 1) & agrupado_por_municipio_vereadores_capital['VR_DESPESA_MAX_CAMPANHA'].notnull()]
].groupby(['capital', 'NOME DO MUNICÍPIO'])['VR_DESPESA_MAX_CAMPANHA'].mean().reset_index()

agrupado_por_municipio_vereadores_capital['Média Capital'] = agrupado_por_municipio_vereadores_capital_1['VR_DESPESA_MAX_CAMPANHA'].mean()

# Calcula a média de 'VR_DESPESA_MAX_CAMPANHA' em não capitais
agrupado_por_municipio_vereadores_capital_0 = agrupado_por_municipio_vereadores_capital[
    (agrupado_por_municipio_vereadores_capital['capital'] == 0) & agrupado_por_municipio_vereadores_capital['VR_DESPESA_MAX_CAMPANHA'].notnull()]
].groupby(['capital', 'NOME DO MUNICÍPIO'])['VR_DESPESA_MAX_CAMPANHA'].mean().reset_index()

agrupado_por_municipio_vereadores_capital['Média Não Capital'] = agrupado_por_municipio_vereadores_capital_0['VR_DESPESA_MAX_CAMPANHA'].mean()

# Calcula e exibe a média geral de 'VR_DESPESA_MAX_CAMPANHA' por município
print('Média de VR_DESPESA_MAX_CAMPANHA por Município - df_eleitos_vereadores_outliers:')
print(agrupado_por_municipio_vereadores)

# Calcula e exibe a média de 'VR_DESPESA_MAX_CAMPANHA' em capitais
print("VR_DESPESA_MAX_CAMPANHA médio em capitais - df_eleitos_vereadores_outliers")
print(agrupado_por_municipio_vereadores_capital_1['VR_DESPESA_MAX_CAMPANHA'].mean())

# Calcula e exibe a média de 'VR_DESPESA_MAX_CAMPANHA' em não capitais
print("VR_DESPESA_MAX_CAMPANHA médio em não capitais - df_eleitos_vereadores_outliers")
print(agrupado_por_municipio_vereadores_capital_0['VR_DESPESA_MAX_CAMPANHA'].mean())

# Agrupa os dados por 'NOME DO MUNICÍPIO' e calcula a média da despesa máxima de campanha (VR DESPESA MAX CAMPANHA)
# para prefeitos eleitos, removendo outliers e reiniciando o índice.
agrupado_por_municipio_prefeitos = df_eleitos_prefeitos_outliers.groupby('NOME DO MUNICÍPIO')['VR_DESPESA_MAX_CAMPANHA'].mean().reset_index().copy()

```

```

# Ordena o DataFrame resultante em ordem decrescente com base na média da
despesa máxima de campanha.
agrupado_por_municipio_prefeitos = agrupado_por_municipio_prefei-
tos.sort_values(by='VR_DESPESA_MAX_CAMPANHA', ascending=False)

# Calcula a média da despesa máxima de campanha por município, distinguindo
entre capitais e não capitais.
agrupado_por_municipio_prefeitos_capital = df_eleitos_prefeitos_outli-
ers.groupby(['capital', 'NOME DO MUNICÍPIO'])['VR_DESPESA_MAX_CAMPA-
NHA'].mean().reset_index()

# Calcula a média da despesa máxima de campanha para capitais.
agrupado_por_municipio_prefeitos_capital_1 = agrupado_por_municipio_prefei-
tos_capital[
    (agrupado_por_municipio_prefeitos_capital['capital'] == 1) &
    (agrupado_por_municipio_prefeitos_capital['VR_DESPESA_MAX_CAMPA-
NHA'].notnull())
].groupby(['capital', 'NOME DO MUNICÍPIO'])['VR_DESPESA_MAX_CAMPA-
NHA'].mean().reset_index()

# Adiciona a média da despesa máxima de campanha para capitais ao DataFrame
principal.
agrupado_por_municipio_prefeitos_capital['Média Capital'] = agru-
pado_por_municipio_prefeitos_capital_1['VR_DESPESA_MAX_CAMPANHA'].mean()

# Calcula a média da despesa máxima de campanha para não capitais.
agrupado_por_municipio_prefeitos_capital_0 = agrupado_por_municipio_prefei-
tos_capital[
    (agrupado_por_municipio_prefeitos_capital['capital'] == 0) &
    (agrupado_por_municipio_prefeitos_capital['VR_DESPESA_MAX_CAMPA-
NHA'].notnull())
].groupby(['capital', 'NOME DO MUNICÍPIO'])['VR_DESPESA_MAX_CAMPA-
NHA'].mean().reset_index()

# Adiciona a média da despesa máxima de campanha para não capitais ao Data-
Frame principal.
agrupado_por_municipio_prefeitos_capital['Média Não Capital'] = agru-
pado_por_municipio_prefeitos_capital_0['VR_DESPESA_MAX_CAMPANHA'].mean()

# Imprime a média da despesa máxima de campanha por município.
print('VR_DESPESA_MAX_CAMPANHA por Município - df_eleitos_prefeitos_outli-
ers:')
print(agrupado_por_municipio_prefeitos)

# Imprime a média da despesa máxima de campanha em capitais e não capitais.
print("Média da VR_DESPESA_MAX_CAMPANHA em Capitais - df_eleitos_prefei-
tos_outliers")
print(agrupado_por_municipio_prefeitos_capital_1['VR_DESPESA_MAX_CAMPA-
NHA'].mean())
print("Média da VR_DESPESA_MAX_CAMPANHA em Não Capitais - df_eleitos_pre-
feitos_outliers")
print(agrupado_por_municipio_prefeitos_capital_0['VR_DESPESA_MAX_CAMPA-
NHA'].mean())

# Define a paleta de cores para os gráficos usando o esquema "viridis" com
30 cores diferentes.
sns.set_palette("viridis", 30)

# Cria uma figura com dois subplots de tamanho personalizado (16 unidades
de largura por 6 unidades de altura).
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

```

```

# Primeiro gráfico de barras (por UF)
# Utiliza um gráfico de barras para representar os gastos máximos de campanha por município.
# Os municípios são ordenados pela ordem original.
sns.barplot(data=agrupado_por_municipio_vereadores, x='NOME DO MUNICÍPIO',
y='VR_DESPESA_MAX_CAMPANHA', ax=axes[0],
            order=agrupado_por_municipio_vereadores['NOME DO MUNICÍPIO'])

# Adiciona um gráfico de linha representando a média dos gastos máximos de campanha em capitais.
sns.lineplot(data=agrupado_por_municipio_vereadores_capital, x='NOME DO MUNICÍPIO', y='Média Capital',
             label='VR_DESPESA_MAX_CAMPANHA médio em capitais', linestyle='dashed', color='Red', ax=axes[0])

# Adiciona um gráfico de linha representando a média dos gastos máximos de campanha em não capitais.
sns.lineplot(data=agrupado_por_municipio_vereadores_capital, x='NOME DO MUNICÍPIO', y='Média Não Capital',
             label='VR_DESPESA_MAX_CAMPANHA médio em não capitais', linestyle='dashed', color='Blue', ax=axes[0])

# Define o título, rótulos dos eixos e formatação dos ticks para o primeiro gráfico.
axes[0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por Município - df_eleitos_vereadores_outliers')
axes[0].set_xlabel('Município')
axes[0].set_ylabel('Média VR_DESPESA_MAX_CAMPANHA')
axes[0].tick_params(axis='x', rotation=90)
axes[0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Define uma nova paleta de cores para o segundo gráfico com 50 cores diferentes.
sns.set_palette("viridis", 50)

# Segundo gráfico de barras (por Região)
# Utiliza um gráfico de barras para representar os gastos máximos de campanha por município.
# Os municípios são ordenados pela ordem original.
sns.barplot(data=agrupado_por_municipio_prefeitos, x='NOME DO MUNICÍPIO',
y='VR_DESPESA_MAX_CAMPANHA', ax=axes[1],
            order=agrupado_por_municipio_prefeitos['NOME DO MUNICÍPIO'])

# Adiciona um gráfico de linha representando a média dos gastos máximos de campanha em capitais.
sns.lineplot(data=agrupado_por_municipio_prefeitos_capital, x='NOME DO MUNICÍPIO', y='Média Capital',
             label='VR_DESPESA_MAX_CAMPANHA médio em capitais', linestyle='dashed', color='Red', ax=axes[1])

# Adiciona um gráfico de linha representando a média dos gastos máximos de campanha em não capitais.
sns.lineplot(data=agrupado_por_municipio_prefeitos_capital, x='NOME DO MUNICÍPIO', y='Média Não Capital',
             label='VR DESPESA MAX CAMPANHA médio em não capitais', linestyle='dashed', color='Blue', ax=axes[1])

# Define o título, rótulos dos eixos e formatação dos ticks para o segundo gráfico.

```

```

axes[1].set_title('VR_DESPESA_MAX_CAMPANHA por Município - df_eleitos_preefeitos_outliers')
axes[1].set_xlabel('Município')
axes[1].set_ylabel('VR_DESPESA_MAX_CAMPANHA')
axes[1].tick_params(axis='x', rotation=90)
axes[1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Ajusta o layout e exibe os gráficos.
plt.tight_layout()
plt.show()

# Configurando a paleta de cores para o gráfico
sns.set_palette("viridis", 30)

# Agrupando os dados de vereadores por unidade federativa (UF) e calculando
# a média das despesas máximas de campanha e PIB per capita
agrupado_por_uf_pib_vereadores = df_eleitos_vereadores_outliers.groupby('UF')[['VR_DESPESA_MAX_CAMPANHA', 'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)']].mean().reset_index()

# Classificando os dados por despesas máximas de campanha de vereadores em
# ordem decrescente
agrupado_por_uf_pib_vereadores = agrupado_por_uf_pib_vereadores.sort_values(by=['VR_DESPESA_MAX_CAMPANHA'], ascending=False)

# Classificando os dados por PIB per capita em ordem decrescente
agrupado_por_uf_pib_vereadores_ord = agrupado_por_uf_pib_vereadores.sort_values(by=['Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'], ascending=False)

# Imprimindo a média das despesas máximas de campanha por UF para vereadores
print('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_vereadores_outliers:')
print(agrupado_por_uf_pib_vereadores)

# Imprimindo a média do PIB per capita por UF para vereadores
print('Média de PIB, a preços correntes (R$ 1,00) por UF - df_eleitos_vereadores_outliers:')
print(agrupado_por_uf_pib_vereadores_ord)

# Agrupando os dados de prefeitos por unidade federativa (UF) e calculando
# a média das despesas máximas de campanha e PIB per capita
agrupado_por_uf_pib_prefeitos = df_eleitos_prefeitos_outliers.groupby('UF')[['VR_DESPESA_MAX_CAMPANHA', 'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)']].mean().reset_index()

# Classificando os dados por despesas máximas de campanha de prefeitos em
# ordem decrescente
agrupado_por_uf_pib_prefeitos = agrupado_por_uf_pib_prefeitos.sort_values(by=['VR_DESPESA_MAX_CAMPANHA'], ascending=False)

# Classificando os dados por PIB per capita em ordem decrescente
agrupado_por_uf_pib_prefeitos_ord = agrupado_por_uf_pib_prefeitos.sort_values(by=['Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'], ascending=False)

# Imprimindo a média das despesas máximas de campanha por UF para prefeitos
print('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers:')

```

```

print(agrupado_por_uf_pib_prefeitos)

# Imprimindo a média do PIB per capita por UF para prefeitos
print('Média de PIB per capita, a preços correntes (R$ 1,00) por UF -')
df_eleitos_prefeitos_outliers:')
print(agrupado_por_uf_pib_prefeitos_ord)

# Cria uma figura com dois subplots em duas linhas e duas colunas
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Primeiro gráfico de barras (vereadores) - Média do VR_DESPESA_MAX_CAMPANHA por UF
sns.barplot(data=agrupado_por_uf_pib_vereadores, x='UF', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[0, 0], order=agrupado_por_uf_pib_vereadores['UF'])
axes[0, 0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_vereadores_outliers')
axes[0, 0].set_xlabel('UF')
axes[0, 0].set_ylabel('Média de Gastos Máximos de Campanha')
axes[0, 0].tick_params(axis='x', rotation=90)
axes[0, 0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Segundo gráfico de barras (vereadores) - Média do PIB per capita por UF
sns.barplot(data=agrupado_por_uf_pib_vereadores_ord, x='UF', y='Produto Interno Bruto per capita, a preços correntes (R$ 1,00)', ax=axes[0, 1], order=agrupado_por_uf_pib_vereadores_ord['UF'])
axes[0, 1].set_title('Média do PIB per capita por UF - df_eleitos_vereadores_outliers')
axes[0, 1].set_xlabel('UF')
axes[0, 1].set_ylabel('Média do PIB per capita, a preços correntes (R$ 1,00)')
axes[0, 1].tick_params(axis='x', rotation=90)
axes[0, 1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Terceiro gráfico de barras (prefeitos) - Média do VR_DESPESA_MAX_CAMPANHA por UF
sns.barplot(data=agrupado_por_uf_pib_prefeitos, x='UF', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[1, 0], order=agrupado_por_uf_pib_prefeitos['UF'])
axes[1, 0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers')
axes[1, 0].set_xlabel('UF')
axes[1, 0].set_ylabel('Média de VR_DESPESA_MAX_CAMPANHA')
axes[1, 0].tick_params(axis='x', rotation=90)
axes[1, 0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Quarto gráfico de barras (prefeitos) - Média do PIB per capita por UF
sns.barplot(data=agrupado_por_uf_pib_prefeitos_ord, x='UF', y='Produto Interno Bruto per capita, a preços correntes (R$ 1,00)', ax=axes[1, 1], order=agrupado_por_uf_pib_prefeitos_ord['UF'])
axes[1, 1].set_title('Média do PIB per capita por UF - df_eleitos_prefeitos_outliers')
axes[1, 1].set_xlabel('UF')
axes[1, 1].set_ylabel('Média do PIB per capita, a preços correntes (R$ 1,00)')
axes[1, 1].tick_params(axis='x', rotation=90)
axes[1, 1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Ajuste de layout para evitar sobreposição
plt.tight_layout()

```

```

# Exiba os gráficos
plt.show()

# Para o primeiro gráfico (vereadores):

# Agrupa os dados por 'UF' e calcula a média das colunas 'VR_DESPESA_MAX_CAMPANHA' e 'POPULAÇÃO ESTIMADA'.
agrupado_por_uf_pop_vereadores = df_eleitos_vereadores_outliers.groupby('UF')[['VR_DESPESA_MAX_CAMPANHA', 'POPULAÇÃO ESTIMADA']].mean().reset_index()

# Ordena o DataFrame pela média de 'VR_DESPESA_MAX_CAMPANHA' em ordem decrescente.
agrupado_por_uf_pop_vereadores = agrupado_por_uf_pop_vereadores.sort_values(by=['VR_DESPESA_MAX_CAMPANHA'], ascending=False)

# Ordena o DataFrame pela média de 'POPULAÇÃO ESTIMADA' em ordem decrescente.
agrupado_por_uf_pop_vereadores_ord = agrupado_por_uf_pop_vereadores.sort_values(by=['POPULAÇÃO ESTIMADA'], ascending=False)

# Exibe as médias de gastos de campanha para vereadores por UF.
print('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_vereadores_outliers:')
print(agrupado_por_uf_pop_vereadores)

# Exibe as médias de população estimada por UF, ordenadas por gastos de campanha para vereadores.
print('Média de POPULAÇÃO ESTIMADA por UF - df_eleitos_vereadores_outliers')
print(agrupado_por_uf_pop_vereadores_ord)

# Para o segundo gráfico (prefeitos):

# Agrupa os dados por 'UF' e calcula a média das colunas 'VR_DESPESA_MAX_CAMPANHA' e 'POPULAÇÃO ESTIMADA'.
agrupado_por_uf_pop_prefeitos = df_eleitos_prefeitos_outliers.groupby('UF')[['VR_DESPESA_MAX_CAMPANHA', 'POPULAÇÃO ESTIMADA']].mean().reset_index()

# Ordena o DataFrame pela média de 'VR_DESPESA_MAX_CAMPANHA' em ordem decrescente.
agrupado_por_uf_pop_prefeitos = agrupado_por_uf_pop_prefeitos.sort_values(by=['VR_DESPESA_MAX_CAMPANHA'], ascending=False)

# Ordena o DataFrame pela média de 'POPULAÇÃO ESTIMADA' em ordem decrescente.
agrupado_por_uf_pop_prefeitos_ord = agrupado_por_uf_pop_prefeitos.sort_values(by=['POPULAÇÃO ESTIMADA'], ascending=False)

# Exibe as médias de gastos de campanha para prefeitos por UF.
print('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers:')
print(agrupado_por_uf_pop_prefeitos)

# Exibe as médias de população estimada por UF, ordenadas por gastos de campanha para prefeitos.

```

```

print('Média de POPULAÇÃO ESTIMADA por UF - df_eleitos_prefeitos_outliers:')
print(agrupado_por_uf_pop_prefeitos_ord)

# Cria uma figura com dois subplots em duas linhas e duas colunas
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Primeiro gráfico de barras (vereadores) - Média VR_DESPESA_MAX_CAMPANHA
# por UF
sns.barplot(data=agrupado_por_uf_pop_vereadores, x='UF', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[0, 0], order=agrupado_por_uf_pop_vereadores['UF'])
axes[0, 0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_vereadores_outliers')
axes[0, 0].set_xlabel('UF')
axes[0, 0].set_ylabel('Média de Gastos Máximos de Campanha')
axes[0, 0].tick_params(axis='x', rotation=90)
axes[0, 0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Segundo gráfico de barras (vereadores) - Média POPULAÇÃO ESTIMADA por UF
sns.barplot(data=agrupado_por_uf_pop_vereadores_ord, x='UF', y='POPULAÇÃO ESTIMADA', ax=axes[0, 1], order=agrupado_por_uf_pop_vereadores_ord['UF'])
axes[0, 1].set_title('Média de POPULAÇÃO ESTIMADA por UF - df_eleitos_vereadores_outliers')
axes[0, 1].set_xlabel('UF')
axes[0, 1].set_ylabel('Média de POPULAÇÃO ESTIMADA ')
axes[0, 1].tick_params(axis='x', rotation=90)
axes[0, 1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Terceiro gráfico de barras (prefeitos) - Média VR_DESPESA_MAX_CAMPANHA
# por UF
sns.barplot(data=agrupado_por_uf_pop_prefeitos, x='UF', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[1, 0], order=agrupado_por_uf_pop_prefeitos['UF'])
axes[1, 0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers')
axes[1, 0].set_xlabel('UF')
axes[1, 0].set_ylabel('Média de VR_DESPESA_MAX_CAMPANHA')
axes[1, 0].tick_params(axis='x', rotation=90)
axes[1, 0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Quarto gráfico de barras (prefeitos) - Média POPULAÇÃO ESTIMADA por UF
sns.barplot(data=agrupado_por_uf_pop_prefeitos_ord, x='UF', y='POPULAÇÃO ESTIMADA', ax=axes[1, 1], order=agrupado_por_uf_pop_prefeitos_ord['UF'])
axes[1, 1].set_title('Média de POPULAÇÃO ESTIMADA por UF - df_eleitos_prefeitos_outliers')
axes[1, 1].set_xlabel('UF')
axes[1, 1].set_ylabel('Média de POPULAÇÃO ESTIMADA ')
axes[1, 1].tick_params(axis='x', rotation=90)
axes[1, 1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Ajusta o layout para evitar sobreposição de elementos
plt.tight_layout()

# Exibe os gráficos
plt.show()

# Agrupamento de dados por 'UF' e cálculo da média de 'VR_DESPESA_MAX_CAMPANHA' para o primeiro gráfico

```

```

agrupado_por_uf = df_eleitos_prefeitos_outliers.groupby('UF')[['VR_DESPESA_MAX_CAMPANHA']].mean().reset_index()
# Ordenação do DataFrame resultante em ordem decrescente pela média de despesas máximas de campanha
agrupado_por_uf = agrupado_por_uf.sort_values(by='VR_DESPESA_MAX_CAMPANHA', ascending=False)

# Exibição da média de 'VR_DESPESA_MAX_CAMPANHA' por UF - DataFrame:
df_eleitos_prefeitos_outliers
print("Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers")
print(agrupado_por_uf)

# Agrupamento de dados por 'Nome_da_Grande_Região' e cálculo da média de 'VR_DESPESA_MAX_CAMPANHA' para o segundo gráfico
agrupado_por_regiao = df_eleitos_prefeitos_outliers.groupby('Nome_da_Grande_Região')[['VR_DESPESA_MAX_CAMPANHA']].mean().reset_index()
# Ordenação do DataFrame resultante em ordem decrescente pela média de despesas máximas de campanha
agrupado_por_regiao = agrupado_por_regiao.sort_values(by='VR_DESPESA_MAX_CAMPANHA', ascending=False)

# Exibição da média de 'VR_DESPESA_MAX_CAMPANHA' por Região - DataFrame:
df_eleitos_prefeitos_outliers
print("Média de VR_DESPESA_MAX_CAMPANHA por Região - df_eleitos_prefeitos_outliers")
print(agrupado_por_regiao)

# Criação de uma figura com dois subplots para os gráficos
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Primeiro gráfico de barras (por UF)
sns.barplot(data=agrupado_por_uf, x='UF', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[0], order=agrupado_por_uf['UF'])
axes[0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers')
axes[0].set_xlabel('UF')
axes[0].set_ylabel('Média VR_DESPESA_MAX_CAMPANHA')
axes[0].tick_params(axis='x', rotation=90)
axes[0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Segundo gráfico de barras (por Região)
sns.barplot(data=agrupado_por_regiao, x='Nome_da_Grande_Região', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[1], order=agrupado_por_regiao['Nome_da_Grande_Região'])
axes[1].set_title('Média de VR_DESPESA_MAX_CAMPANHA por Região - df_eleitos_prefeitos_outliers')
axes[1].set_xlabel('Região')
axes[1].set_ylabel('Média VR_DESPESA_MAX_CAMPANHA')
axes[1].tick_params(axis='x', rotation=90)
axes[1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Ajuste de layout e exibição dos gráficos
plt.tight_layout()
plt.show()

# Realiza o agrupamento e ordenação dos dados para o primeiro gráfico de barras (vereadores)

```

```

agrupado_por_partido_vereadores = df_eleitos_vereadores_outliers.groupby('SG_PARTIDO')['VR_DESPESA_MAX_CAMPANHA'].mean().reset_index()
agrupado_por_partido_vereadores = agrupado_por_partido_vereadores.sort_values(by='VR_DESPESA_MAX_CAMPANHA', ascending=False)

# Imprime a média de VR_DESPESA_MAX_CAMPANHA por Partido para vereadores
print("Média de VR_DESPESA_MAX_CAMPANHA por Partido - df_eleitos_vereadores_outliers")
print(agrupado_por_partido_vereadores)

# Realiza o agrupamento e ordenação dos dados para o segundo gráfico de barras (prefeitos)
agrupado_por_partido_prefeitos = df_eleitos_prefeitos_outliers.groupby('SG_PARTIDO')['VR_DESPESA_MAX_CAMPANHA'].mean().reset_index()
agrupado_por_partido_prefeitos = agrupado_por_partido_prefeitos.sort_values(by='VR_DESPESA_MAX_CAMPANHA', ascending=False)

# Imprime a média de VR_DESPESA_MAX_CAMPANHA por Partido para prefeitos
print("Média de VR_DESPESA_MAX_CAMPANHA por Partido - df_eleitos_prefeitos_outliers")
print(agrupado_por_partido_prefeitos)

# Cria uma figura com dois subgráficos (um para cada gráfico de barras)
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Configura o primeiro gráfico de barras (vereadores)
sns.barplot(data=agrupado_por_partido_vereadores, x='SG_PARTIDO',
y='VR_DESPESA_MAX_CAMPANHA', ax=axes[0],
order=agrupado_por_partido_vereadores['SG_PARTIDO'])
axes[0].set_title('Média de VR_DESPESA_MAX_CAMPANHA por Partido - df_eleitos_vereadores_outliers')
axes[0].set_xlabel('Partido')
axes[0].set_ylabel('Média VR_DESPESA_MAX_CAMPANHA')
axes[0].tick_params(axis='x', rotation=90)
axes[0].ticklabel_format(axis='y', style='plain', useOffset=False)

# Configura o segundo gráfico de barras (prefeitos)
sns.barplot(data=agrupado_por_partido_prefeitos, x='SG_PARTIDO', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[1],
order=agrupado_por_partido_prefeitos['SG_PARTIDO'])
axes[1].set_title('Média de VR_DESPESA_MAX_CAMPANHA por Partido - df_eleitos_prefeitos_outliers')
axes[1].set_xlabel('Partido')
axes[1].set_ylabel('Média VR_DESPESA_MAX_CAMPANHA')
axes[1].tick_params(axis='x', rotation=90)
axes[1].ticklabel_format(axis='y', style='plain', useOffset=False)

# Ajusta o layout para garantir que os gráficos não se sobreponham
plt.tight_layout()

# Exibe a figura com os dois gráficos de barras
plt.show()

# Define a paleta de cores a ser usada nos gráficos
sns.set_palette("Paired", 30)

# Cria uma figura com três subgráficos
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Agrupa os dados por 'ST REELEICAO' para diferentes conjuntos de dados

```

```

agrupado_por_reeleicao_demograficos = df_eleitos_dados_demograficos_outliers.groupby(
    'ST_REELEICAO').count().reset_index()
agrupado_por_reeleicao_prefeitos = df_eleitos_prefeitos_outliers.groupby('ST_REELEICAO').count().reset_index()
agrupado_por_reeleicao_vereadores = df_eleitos_vereadores_outliers.groupby('ST_REELEICAO').count().reset_index()

# Plota os gráficos de donut para diferentes conjuntos de dados
plot_donut(agrupado_por_reeleicao_demograficos,
            'Percentual Situação Reeleição - df_eleitos_dados_demograficos_outliers', axes[0])
plot_donut(agrupado_por_reeleicao_prefeitos, 'Percentual Situação Reeleição - df_eleitos_prefeitos_outliers', axes[1])
plot_donut(agrupado_por_reeleicao_vereadores, 'Percentual Situação Reeleição - df_eleitos_vereadores_outliers', axes[2])

# Imprime informações sobre os dados agrupados
print('Percentual Situação Reeleição - df_eleitos_dados_demograficos_outliers')
print(agrupado_por_reeleicao_demograficos)

print('Percentual Situação Reeleição - df_eleitos_prefeitos_outliers')
print(agrupado_por_reeleicao_prefeitos)

print('Percentual Situação Reeleição - df_eleitos_vereadores_outliers')
print(agrupado_por_reeleicao_vereadores)

# Ajusta o layout para evitar sobreposições
plt.tight_layout()

# Exibe o gráfico de donut contendo os três gráficos de pizza
plt.show()

#Agrupa os dados por 'DS_CARGO' e calcula a contagem de ocorrências (eleitos) para o primeiro gráfico
agrupado_por_cargo = df_eleitos_dados_demograficos_outliers.groupby('DS_CARGO').count().reset_index()

#Ordena o DataFrame resultante em ordem decrescente com base na coluna 'VR_DESPESA_MAX_CAMPANHA'
agrupado_por_cargo = agrupado_por_cargo.sort_values(by='VR_DESPESA_MAX_CAMPANHA', ascending=False)

# Exibe o total de eleitos por cargo
print("Total de eleitos por CARGO - df_eleitos_dados_demograficos_outliers:")
print(agrupado_por_cargo)

#Agrupa os dados por 'TP AGREMIACAO' e calcula a contagem de ocorrências (eleitos) para o segundo gráfico
agrupado_por_agremiacao = df_eleitos_dados_demograficos_outliers.groupby('TP AGREMIACAO').count().reset_index()

#Ordena o DataFrame resultante em ordem decrescente com base na coluna 'VR DESPESA MAX CAMPANHA'
agrupado_por_agremiacao = agrupado_por_agremiacao.sort_values(by='VR DESPESA_MAX_CAMPANHA', ascending=False)

# Exibe o total de eleitos por agremiação

```

```

print("Total de eleitos por AGREMIACAO - df_eleitos_dados_demograficos_outliers:")
print(agrupado_por_agremiacao)

#Cria uma figura com dois subplots (gráficos lado a lado)
fig, axes = plt.subplots(1, 2, figsize=(8, 10))

#Plota o primeiro gráfico de barras (por cargo)
sns.barplot(data=agrupado_por_cargo, x='DS_CARGO', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[0],
             order=agrupado_por_cargo['DS_CARGO'])
axes[0].set_title('Total de eleitos por CARGO - df_eleitos_dados_demograficos_outliers')
axes[0].set_xlabel('CARGO')
axes[0].set_ylabel('Total de eleitos')
axes[0].tick_params(axis='x', rotation=90)
axes[0].ticklabel_format(axis='y', style='plain', useOffset=False)

#Plota o segundo gráfico de barras (por agremiação)
sns.barplot(data=agrupado_por_agremiacao, x='TP AGREMIACAO', y='VR_DESPESA_MAX_CAMPANHA', ax=axes[1],
             order=agrupado_por_agremiacao['TP AGREMIACAO'])
axes[1].set_title('Total de eleitos por AGREMIACAO - df_eleitos_dados_demograficos_outliers')
axes[1].set_xlabel('AGREMIACAO')
axes[1].set_ylabel('Total de eleitos')
axes[1].tick_params(axis='x', rotation=90)
axes[1].ticklabel_format(axis='y', style='plain', useOffset=False)

#Ajusta o layout dos subplots para melhor visualização
plt.tight_layout()

#Exibe os gráficos
plt.show()

print('*' * 100)
print('FIM Análise Outliers')
print('*' * 100)

print('*' * 100)
print('FIM Análise e Exploração dos Dados - Criação de Modelos de Machine Learning')
print('*' * 100)

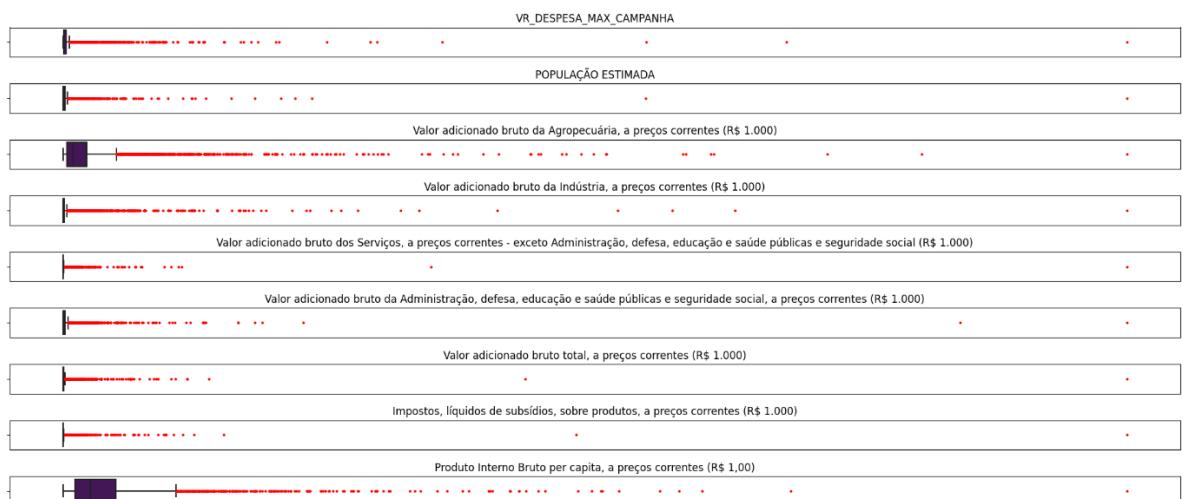
```

## Gráficos

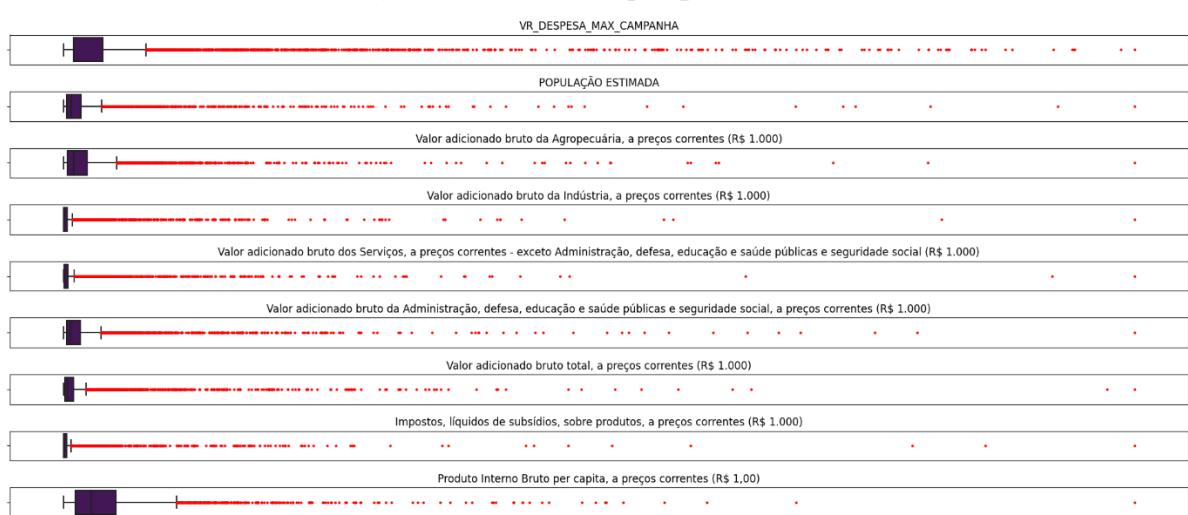
Boxplots das Colunas Numéricas em df\_eleitos\_vereadores sem outliers



Boxplots das Colunas Numéricas em df\_eleitos\_prefeitos



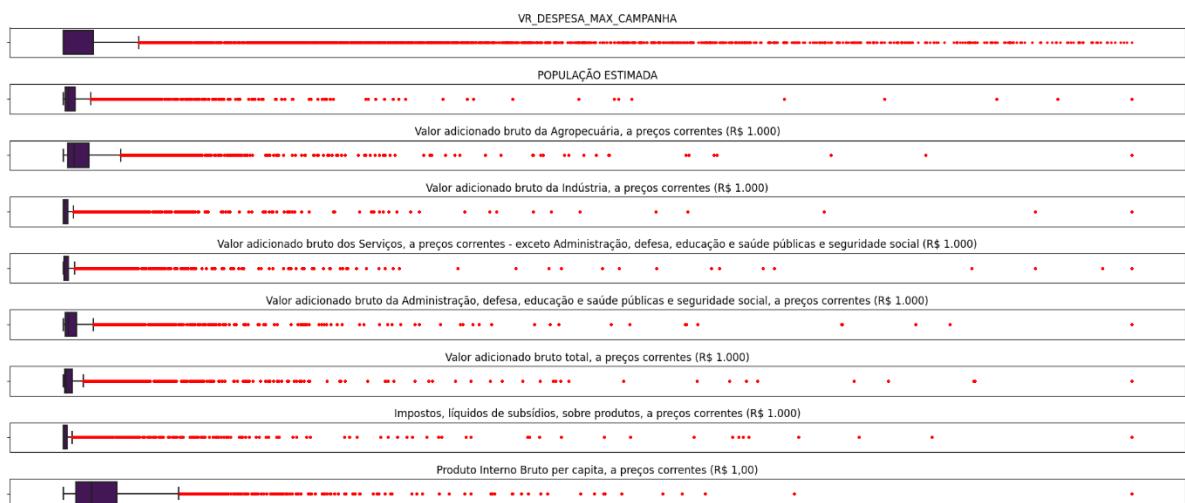
Boxplots das Colunas Numéricas em df\_eleitos\_prefeitos sem outliers



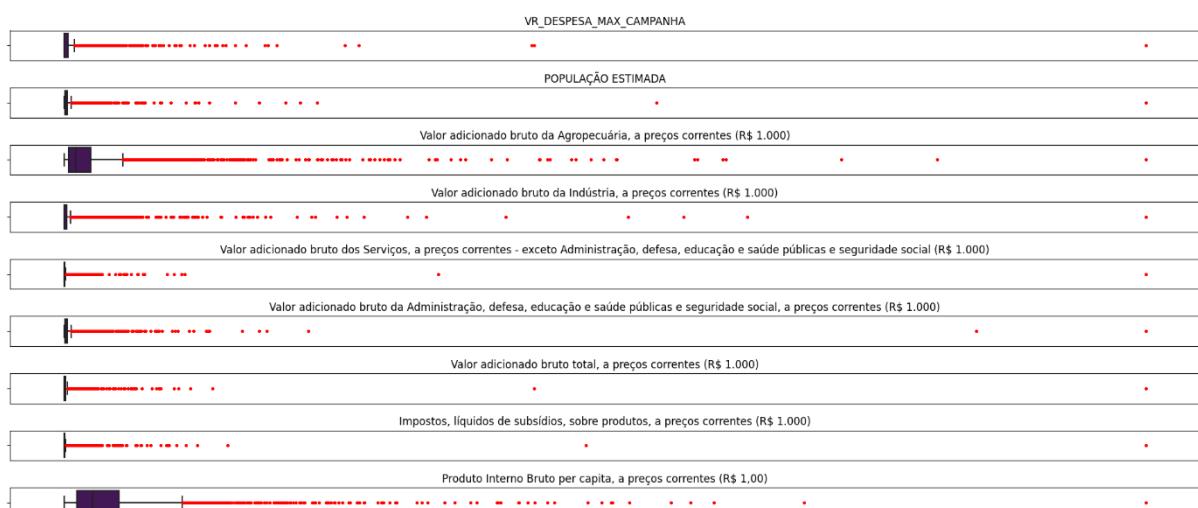
Boxplots das Colunas Numéricas em df\_eleitos\_dados\_demograficos



Boxplots das Colunas Numéricas em df\_eleitos\_dados\_demograficos sem outliers



Boxplots das Colunas Numéricas em df\_eleitos\_vereadores



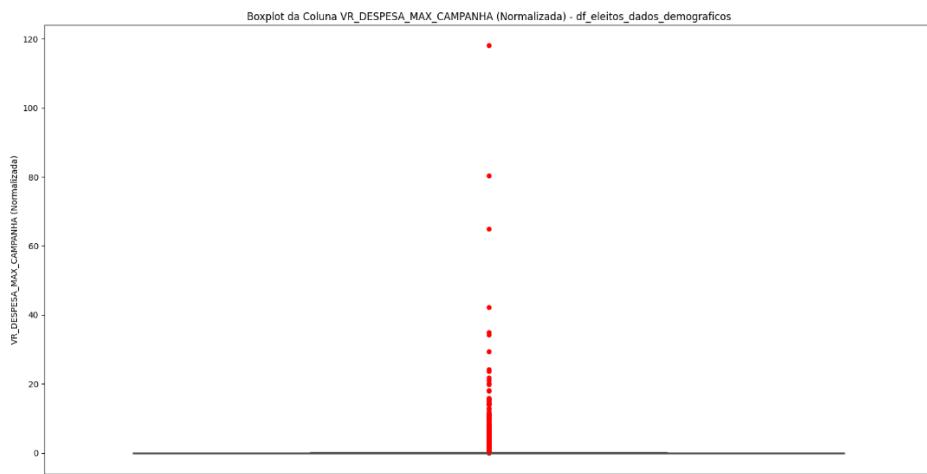
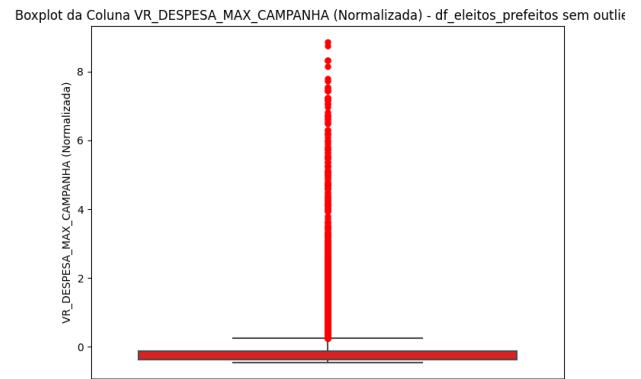
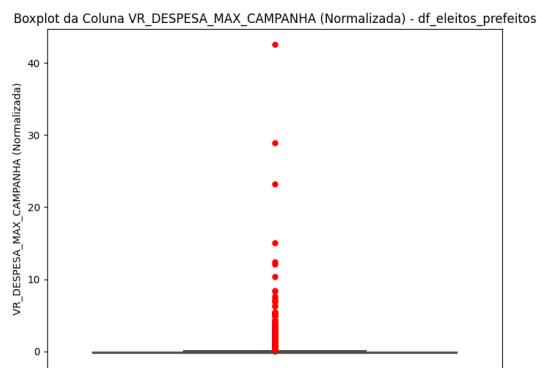
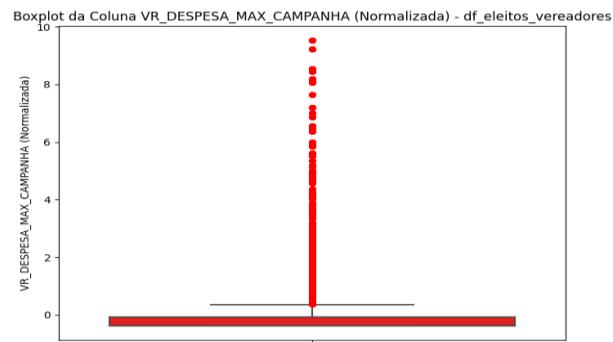
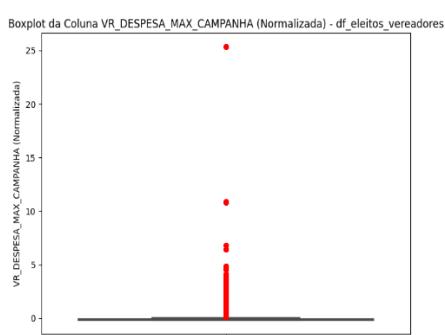


Gráfico de Dispersão dos Resíduos em Função das Previsões - df\_eleitos\_vereadores com outliers

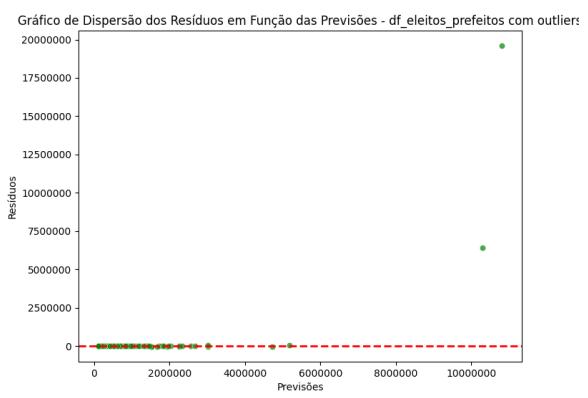
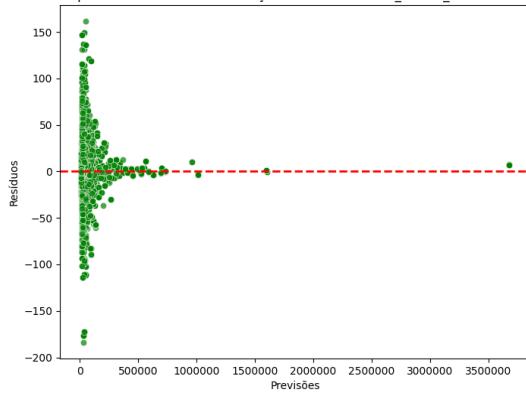
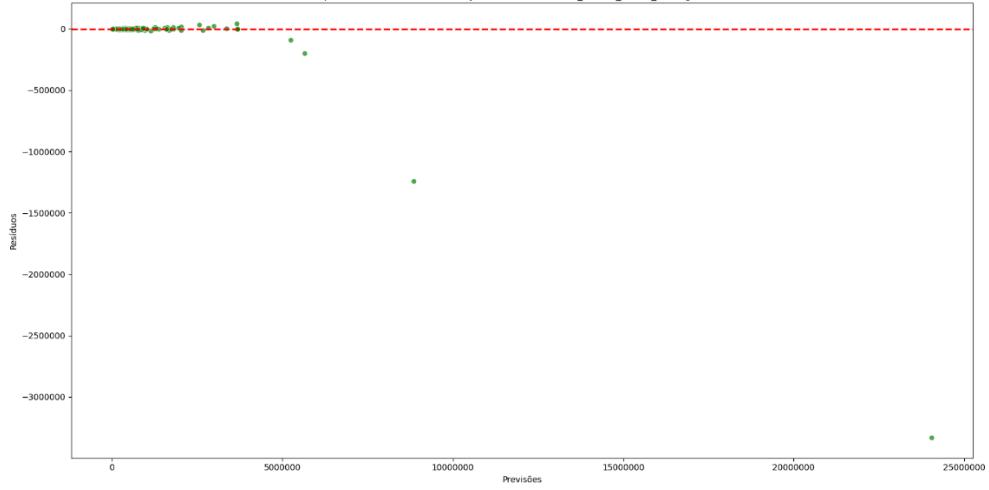


Gráfico de Dispersão dos Resíduos em Função das Previsões - df\_eleitos\_dados\_demograficos com outliers



## Saída completa do código

```
*****
INICIO Coleta de Dados - Processamento/Tratamento de Dados
*****
***** df_TSE_IBGE *****
*****
  codigo_tse  uf nome_municipio  capital  codigo_ibge
0          1120  AC      ACRELÂNDIA        0   1200013
1          1570  AC      ASSIS BRASIL       0   1200054
2          1058  AC      BRASILÉIA        0   1200104
3          1007  AC      BUJARI          0   1200138
4          1015  AC      CAPIXABA        0   1200179
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   codigo_tse        5570 non-null    int64  
 1   uf                5570 non-null    object  
 2   nome_municipio     5570 non-null    object  
 3   capital           5570 non-null    int64  
 4   codigo_ibge       5570 non-null    int64  
dtypes: int64(3), object(2)
memory usage: 217.7+ KB
None
  codigo_tse  capital  codigo_ibge
count    5570.0000  5570.0000  5570.0000
mean    51469.3993   0.0048  3253590.7707
std     29375.0802   0.0695  984910.3394
min     19.0000   0.0000  1100015.0000
25%    24258.7500   0.0000  2512125.7500
50%    51241.5000   0.0000  3146280.0000
75%    79213.5000   0.0000  4119189.5000
max    99074.0000   1.0000  5300108.0000
*****
FIM df_TSE_IBGE
*****
***** df_eleitos *****
*****
INICIO df_eleitos
*****
Colunas do df_eleitos:
Index(['DT_GERACAO', 'HH_GERACAO', 'ANO_ELEICAO', 'CD_TIPO_ELEICAO',
       'NM_TIPO_ELEICAO', 'NR_TURNO', 'CD_ELEICAO', 'DS_ELEICAO', 'DT_ELEICAO',
       'TE_ABRANGENCIA', 'SG_UF', 'SG_UE', 'NM_UE', 'CD_CARGO', 'DS_CARGO',
       'SQ_CANDIDATO', 'NR_CANDIDATO', 'NM_CANDIDATO', 'NM_URNA_CANDIDATO',
       'NM_SOCIAL_CANDIDATO', 'NR_CPF_CANDIDATO', 'NM_EMAIL',
       'CD_SITUACAO_CANDIDATURA', 'DS_SITUACAO_CANDIDATURA',
       'CD_DETALHE_SITUACAO_CAND', 'DS_DETALHE_SITUACAO_CAND', 'TP_AGREMIACAO',
       'NR_PARTIDO', 'SG_PARTIDO', 'NM_PARTIDO', 'NR_FEDERACAO',
       'NM_FEDERACAO', 'SG_FEDERACAO', 'DS_COMPOSICAO_FEDERACAO',
       'SQ_COLIGACAO', 'NM_COLIGACAO', 'DS_COMPOSICAO_COLIGACAO',
       'CD_NACIONALIDADE', 'DS_NACIONALIDADE', 'SG_UF_NASCIMENTO',
       'CD_MUNICIPIO_NASCIMENTO', 'NM_MUNICIPIO_NASCIMENTO', 'DT_NASCIMENTO',
       'NR_IDADE_DATA_POSSE', 'NR_TITULO_ELEITORAL_CANDIDATO', 'CD_GENERO',
       'DS_GENERO', 'CD_GRAU_INSTRUCAO', 'DS_GRAU_INSTRUCAO',
       'CD_ESTADO_CIVIL', 'DS_ESTADO_CIVIL', 'CD_COR_RACA', 'DS_COR_RACA',
       'CD_OCUPLICAO', 'DS_OCUPLICAO', 'VR_DESPESA_MAX_CAMPANHA',
       'CD_SIT_TOT_TURNO', 'DS_SIT_TOT_TURNO', 'ST_REELEICAO',
       'ST DECLARAR_BENS', 'NR_PROTOCOLO_CANDIDATURA', 'NR_PROCESSO',
       'CD_SITUACAO_CANDIDATO_PLEITO', 'DS_SITUACAO_CANDIDATO_PLEITO',
       'CD_SITUACAO_CANDIDATO_URNA', 'DS_SITUACAO_CANDIDATO_URNA',
       'ST_CANDIDATO_INSERIDO_URNA', 'NM_TIPO_DESTINACAO_VOTOS',
       'CD_SITUACAO_CANDIDATO_TOT', 'DS_SITUACAO_CANDIDATO_TOT',
       'ST_PREST_CONTAS'],
      dtype='object')

Informações do df_eleitos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558585 entries, 0 to 558584
Data columns (total 71 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   DT_GERACAO        558585 non-null  object  
 1   HH_GERACAO        558585 non-null  object  
 2   ANO_ELEICAO       558585 non-null  int64  
 3   CD_TIPO_ELEICAO   558585 non-null  int64  
 4   NM_TIPO_ELEICAO   558585 non-null  object  
 5   NR_TURNO          558585 non-null  int64  
 6   CD_ELEICAO         558585 non-null  int64  
 7   DS_ELEICAO         558585 non-null  object  
 8   DT_ELEICAO         558585 non-null  object  
 9   TP_ABRANGENCIA   558585 non-null  object  
 10  SG_UF             558585 non-null  object  
 11  SG_UE             558585 non-null  int64  
 12  NM_UE             558585 non-null  object
```

```

13  CD_CARGO           558585 non-null int64
14  DS_CARGO           558585 non-null object
15  SQ_CANDIDATO       558585 non-null int64
16  NR_CANDIDATO       558585 non-null int64
17  NM_CANDIDATO       558585 non-null object
18  NM_URNA_CANDIDATO  558582 non-null object
19  NM_SOCIAL_CANDIDATO 558585 non-null object
20  NR_CPF_CANDIDATO   558585 non-null int64
21  NM_EMAIL            558585 non-null object
22  CD_SITUACAO_CANDIDATURA 558585 non-null int64
23  DS_SITUACAO_CANDIDATURA 558585 non-null object
24  CD_DETALHE_SITUACAO_CAND 558585 non-null int64
25  DS_DETALHE_SITUACAO_CAND 558585 non-null object
26  TP_AGREMIACAO       558585 non-null object
27  NR_PARTIDO          558585 non-null int64
28  SG_PARTIDO          558585 non-null object
29  NM_PARTIDO          558585 non-null object
30  NR_FEDERACAO        558585 non-null int64
31  NM_FEDERACAO        558585 non-null object
32  SG_FEDERACAO        558585 non-null object
33  DS_COMPOSICAO_FEDERACAO 558585 non-null object
34  SQ_COLIGACAO         558585 non-null int64
35  NM_COLIGACAO         558585 non-null object
36  DS_COMPOSICAO_COLIGACAO 558585 non-null object
37  CD_NACIONALIDADE    558585 non-null int64
38  DS_NACIONALIDADE    558585 non-null object
39  SG_UF_NASCIMENTO    558585 non-null object
40  CD_MUNICIPIO_NASCIMENTO 558585 non-null int64
41  NM_MUNICIPIO_NASCIMENTO 558585 non-null object
42  DT_NASCIMENTO       558303 non-null object
43  NR_IDADE_DATA_POSSE 558303 non-null float64
44  NR_TITULO_ELEITORAL_CANDIDATO 558585 non-null int64
45  CD_GENERO           558585 non-null int64
46  DS_GENERO           558585 non-null object
47  CD_GRAU_INSTRUCAO   558585 non-null int64
48  DS_GRAU_INSTRUCAO   558585 non-null object
49  CD_ESTADO_CIVIL     558585 non-null int64
50  DS_ESTADO_CIVIL     558585 non-null object
51  CD_COR_RACA         558585 non-null int64
52  DS_COR_RACA         558585 non-null object
53  CD_OCUPACAO         558585 non-null int64
54  DS_OCUPACAO         558585 non-null object
55  VR_DESPESA_MAX_CAMPANHA 558585 non-null float64
56  CD_SIT_TOT_TURNO    558585 non-null int64
57  DS_SIT_TOT_TURNO    558585 non-null object
58  ST_REELEICAO         558585 non-null object
59  ST_DECLARAR_BENS    558585 non-null object
60  NR_PROTOCOLO_CANDIDATURA 558585 non-null int64
61  NR_PROCESSO          558585 non-null int64
62  CD_SITUACAO_CANDIDATO_PLEITO 558585 non-null int64
63  DS_SITUACAO_CANDIDATO_PLEITO 558585 non-null object
64  CD_SITUACAO_CANDIDATO_URNA 558585 non-null int64
65  DS_SITUACAO_CANDIDATO_URNA 558585 non-null object
66  ST_CANDIDATO_INSERIDO_URNA 558585 non-null object
67  NM_TIPO_DESTINACAO_VOTOS 558585 non-null object
68  CD_SITUACAO_CANDIDATO_TOT 558585 non-null int64
69  DS_SITUACAO_CANDIDATO_TOT 558585 non-null object
70  ST_PREST_CONTAS      558585 non-null object
dtypes: float64(2), int64(28), object(41)
memory usage: 302.6+ MB
None

```

Número de valores únicos em cada coluna:

CD_TIPO_ELEICAO	2
NM_TIPO_ELEICAO	2
SG_UF	26
SG_UE	5568
NM_UE	5295
CD_CARGO	3
DS_CARGO	3
CD_SITUACAO_CANDIDATURA	3
DS_SITUACAO_CANDIDATURA	3
TP_AGREMIACAO	3
SG_PARTIDO	34
NM_PARTIDO	34
CD_SIT_TOT_TURNO	7
DS_SIT_TOT_TURNO	7
ST_REELEICAO	3
VR_DESPESA_MAX_CAMPANHA	3811

dtype: int64

Valores únicos em CD\_TIPO\_ELEICAO:  
[2 1]

Valores únicos em NM\_TIPO\_ELEICAO:

['ELEIÇÃO ORDINÁRIA' 'ELEIÇÃO SUPLEMENTAR']

Valores únicos em CD\_CARGO:

[11 13 12]

Valores únicos em DS\_CARGO:  
['PREFEITO' 'VEREADOR' 'VICE-PREFEITO']

Valores únicos em CD\_SITUACAO\_CANDIDATURA:  
[12 3 1]

Valores únicos em DS\_SITUACAO\_CANDIDATURA:  
['APTO' 'INAPTO' 'CADASTRADO']

Valores únicos em TP\_AGREMIACAO:  
['COLIGAÇÃO' 'PARTIDO ISOLADO' 'FEDERAÇÃO']

Valores únicos em SG\_PARTIDO:  
['MOVIMENTO DEMOCRÁTICO BRASILEIRO' 'REPUBLICANOS' 'DEMOCRATAS'  
'PARTIDO DA MOBILIZAÇÃO NACIONAL' 'PARTIDO TRABALHISTA CRISTÃO' 'PODEMOS'  
'PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA' 'PARTIDO SOCIAL CRISTÃO'  
'AVANTE' 'PARTIDO DOS TRABALHADORES' 'PARTIDO COMUNISTA DO BRASIL'  
'PARTIDO SOCIAL LIBERAL' 'PARTIDO REPUBLICANO DA ORDEM SOCIAL'  
'PARTIDO DEMOCRÁTICO TRABALHISTA' 'PARTIDO SOCIAL DEMOCRÁTICO'  
'PARTIDO SOCIALISTA BRASILEIRO' 'PARTIDO VERDE'  
'PARTIDO TRABALHISTA BRASILEIRO' 'PARTIDO LIBERAL'  
'PARTIDO RENOVADOR TRABALHISTA BRASILEIRO' 'PROGRESSISTAS' 'PATRIOTA'  
'SOLIDARIEDADE' 'PARTIDO SOCIALISMO E LIBERDADE' 'CIDADANIA'  
'DEMOCRACIA CRISTÃ' 'REDE SUSTENTABILIDADE'  
'PARTIDO DA MULHER BRASILEIRA' 'PARTIDO NOVO' 'UNIDADE POPULAR'  
'PARTIDO SOCIALISTA DOS TRABALHADORES UNIFICADO'  
'PARTIDO COMUNISTA BRASILEIRO' 'PARTIDO DA CAUSA OPERÁRIA' 'UNIÃO BRASIL']

Valores únicos em CD\_SIT\_TOT\_TURNO:  
[ 1 5 4 2 3 -1 6]

Valores únicos em DS\_SIT\_TOT\_TURNO:  
['ELEITO' 'SUPLENTE' 'NÃO ELEITO' 'ELEITO POR QP' 'ELEITO POR MÉDIA'  
'#NULO#' '2º TURNO']

Valores únicos em ST\_REELEICAO:  
['S' 'N' 'Não divulgável']

Informações do dataframe após a primeira filtragem DS\_SIT\_TOT\_TURNO:  
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 69167 entries, 0 to 558555  
Data columns (total 16 columns):  
 # Column Non-Null Count Dtype  
---  
 0 CD\_TIPO\_ELEICAO 69167 non-null int64  
 1 NM\_TIPO\_ELEICAO 69167 non-null object  
 2 SG\_UF 69167 non-null object  
 3 SG\_UE 69167 non-null object  
 4 NM\_UE 69167 non-null object  
 5 CD\_CARGO 69167 non-null int64  
 6 DS\_CARGO 69167 non-null object  
 7 CD\_SITUACAO\_CANDIDATURA 69167 non-null int64  
 8 DS\_SITUACAO\_CANDIDATURA 69167 non-null object  
 9 TP\_AGREMIACAO 69167 non-null object  
10 SG\_PARTIDO 69167 non-null object  
11 NM\_PARTIDO 69167 non-null object  
12 CD\_SIT\_TOT\_TURNO 69167 non-null int64  
13 DS\_SIT\_TOT\_TURNO 69167 non-null object  
14 ST\_REELEICAO 69167 non-null object  
15 VR\_DESPESA\_MAX\_CAMPANHA 69167 non-null float64  
dtypes: float64(1), int64(4), object(11)  
memory usage: 9.0+ MB  
None

Informações do dataframe após a segunda filtragem DS\_SITUACAO\_CANDIDATURA:  
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 69146 entries, 0 to 558555  
Data columns (total 16 columns):  
 # Column Non-Null Count Dtype  
---  
 0 CD\_TIPO\_ELEICAO 69146 non-null int64  
 1 NM\_TIPO\_ELEICAO 69146 non-null object  
 2 SG\_UF 69146 non-null object  
 3 SG\_UE 69146 non-null object  
 4 NM\_UE 69146 non-null object  
 5 CD\_CARGO 69146 non-null int64  
 6 DS\_CARGO 69146 non-null object  
 7 CD\_SITUACAO\_CANDIDATURA 69146 non-null int64  
 8 DS\_SITUACAO\_CANDIDATURA 69146 non-null object  
 9 TP\_AGREMIACAO 69146 non-null object

```

10 SG_PARTIDO           69146 non-null   object
11 NM_PARTIDO          69146 non-null   object
12 CD_SIT_TOT_TURNO   69146 non-null   int64
13 DS_SIT_TOT_TURNO   69146 non-null   object
14 ST_REELEICAO        69146 non-null   object
15 VR_DESPESA_MAX_CAMPANHA 69146 non-null   float64
dtypes: float64(1), int64(4), object(11)
memory usage: 9.0+ MB
None

```

Informações do dataframe após a remoção de colunas:

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 69146 entries, 0 to 558555
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   SG_UF              69146 non-null   object 
 1   SG_UE              69146 non-null   object 
 2   DS_CARGO           69146 non-null   object 
 3   TP_AGRIMIACAO     69146 non-null   object 
 4   SG_PARTIDO         69146 non-null   object 
 5   NM_PARTIDO         69146 non-null   object 
 6   ST_REELEICAO       69146 non-null   object 
 7   VR_DESPESA_MAX_CAMPANHA 69146 non-null   float64
dtypes: float64(1), object(7)
memory usage: 4.7+ MB
None
*****
```

FIM df\_eleitos

\*\*\*\*\*

\*\*\*\*\*

INICIO df\_populacao

\*\*\*\*\*

```

0   11.0000
1   11.0000
2   11.0000
3   11.0000
4   11.0000
5   11.0000
6   11.0000
7   11.0000
8   11.0000
9   11.0000
Name: COD. UF, dtype: float64
0   15.0000
1   23.0000
2   31.0000
3   49.0000
4   56.0000
5   64.0000
6   72.0000
7   80.0000
8   98.0000
9   106.0000
Name: COD. MUNIC, dtype: float64

```

Análise das variáveis categóricas

Análise de UF:

Total de valores únicos: 41

Valores únicos: ['RO' 'AC' 'AM' 'RR' 'PA' 'AP' 'TO' 'MA' 'PI' 'CE' 'RN' 'PB' 'PE' 'AL' 'SE' 'BA' 'MG' 'ES' 'RJ' 'SP' 'PR' 'SC' 'RS' 'MS' 'MT' 'GO' 'DF' nan

'Fonte: IBGE. Diretoria de Pesquisas - DPE - Coordenação de População e Indicadores Sociais - COPIS.'

'Notas:'

'(1) População judicial do município de Porto Velho-RO: 494.013 habitantes. Processo Judicial nº 12316-40.2016.4.01.4100 - Seção Judiciária de Rondônia.'

'(2) População judicial do município de Manaus-AM: entre 30.565 e 37.356 habitantes. Parecer de Força Executória nº 00010/2017/NUCOB-GEAC/PFAM/PGF/AGU, em trâmite na 3ª VF/AM.'

'(3) População judicial do município de Santa Isabel do Rio Negro-AM: entre 23.773 e 30.564 habitantes. Parecer de Força Executória nº 00007/2017/NUCOB-GEAC/PFAM/PGF/AGU, em trâmite na 3ª VF/AM.'

'(4) População judicial do município de Urucará-AM: entre 16.981 e 23.772 habitantes. Parecer de Força Executória nº 00004/2017/NUCOB-GEAC/PFAM/PGF/AGU, em trâmite na 3ª VF/AM.'

'(5) População judicial do município de Jacareacanga-PA: 41.487 habitantes. Processo Judicial nº 798-41.2011.4.01.3902, Seção Judiciária de Itaituba-PA.'

'(6) População judicial do município de Paço do Lumiar - MA: superior a 156.216 habitantes. Processo Judicial nº 13916-98.2017.4.01.3700 - Seção Judiciária do Maranhão- MA.'

'(7) População judicial do município de São Gonçalo do Amarante-RN: entre 101.881 e 115.464 habitantes. Processo Judicial nº: 0813188-75.2017.4.05.8400 (4ª Vara Federal - RN).'

'(8) População judicial do município de Ferreiros -PE: entre 13.585 e 16.980 habitantes. Processo Judicial nº 0805921-61.2019.4.05.0000 (1ª Turma do Tribunal Regional Federal da 5ª Região).'

'(9) População judicial do município Coronel João Sá-BA: 17.422 habitantes. Processo Judicial nº 0002222-53.2017.4.01.3306 - Vara Única de Paulo Afonso-BA.'

'(10) População judicial do município Ibiassucê-BA: entre 10.189 e 13.584 habitantes. Processo Judicial\xao nº 1018608-53.2017.4.01.3400 (14ª VARA FEDERAL CÍVEL DA SJDF).'

'(11) População judicial do município de Rodelas - BA: entre 10.189 a 13.584 habitantes. Processo Judicial nº 1002950-09.2019.4.01.3306 (Procuradoria Federal no Estado da Bahia, Núcleo Finalístico Ambiental).'

'(12) População judicial do município de Vera Cruz - BA: entre 44.149 a 50.940 habitantes. Processo Judicial nº 8000464-59.2018.8.05.0124 (Vara dos Feitos Relativos às Relações de Consumo, Civis e Comerciais, Registros Públicos e Acidentes de Trabalho e Juizado Especial Adjunto da Comarca de Itaparica).'

==

```
Análise de NOME DO MUNICÍPIO:
Total de valores únicos: 5297
Valores únicos: ['Alta Floresta D'Oeste' 'Ariquemes' 'Cabixi' ... 'Vila Propício'
 'Brasília' nan]
===
Análise de POPULAÇÃO ESTIMADA:
Total de valores únicos: 5125
Valores únicos: [22728 109523 5188 ... 8873 3055149 nan]
===
# Análise das variáveis numéricas
Análise de COD. UF:
Média: 32.37773788150808
Desvio padrão: 9.833861774653943
Valor mínimo: 11.0
Valor máximo: 53.0
===
Análise de COD. MUNIC:
Média: 15816.982585278276
Desvio padrão: 15997.29977960963
Valor mínimo: 13.0
Valor máximo: 72202.0
===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5586 entries, 0 to 5585
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
---  --  
 0   UF                5584 non-null    object  
 1   COD. UF           5570 non-null    float64 
 2   COD. MUNIC        5570 non-null    float64 
 3   NOME DO MUNICÍPIO 5570 non-null    object  
 4   POPULAÇÃO ESTIMADA 5570 non-null    object  
dtypes: float64(2), object(3)
memory usage: 218.3+ KB
None
Análise de 'UF':
Total de valores únicos: 27
Valores únicos: ['RO' 'AC' 'AM' 'RR' 'PA' 'AP' 'TO' 'MA' 'PI' 'CE' 'RN' 'PB' 'PE' 'AL'
 'SE' 'BA' 'MG' 'ES' 'RJ' 'SP' 'PR' 'SC' 'RS' 'MS' 'MT' 'GO' 'DF']
===
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
---  --  
 0   UF                5570 non-null    object  
 1   COD. UF           5570 non-null    float64 
 2   COD. MUNIC        5570 non-null    float64 
 3   NOME DO MUNICÍPIO 5570 non-null    object  
 4   POPULAÇÃO ESTIMADA 5570 non-null    object  
dtypes: float64(2), object(3)
memory usage: 261.1+ KB
None
      COD. UF  COD. MUNIC
count  5570.0000  5570.0000
mean    32.3777  15816.9826
std     9.8339  15997.2998
min     11.0000  13.0000
25%    25.0000  4507.2500
50%    31.0000  10400.5000
75%    41.0000  20853.0000
max    53.0000  72202.0000
      UF  COD. UF  COD. MUNIC          NOME DO MUNICÍPIO  POPULAÇÃO ESTIMADA
16   RO  11.0000  205.0000          Porto Velho          539354 (1)
110  AM  13.0000  2553.0000         Manaus             33049 (2)
123  AM  13.0000  3601.0000  Santa Isabel do Rio Negro  25865 (3)
134  AM  13.0000  4302.0000          Urucará           16130 (4)
210  PA  15.0000  3754.0000  Jacareacanga          7590 (5)
579  MA  21.0000  7506.0000          Paço do Lumiar       123747 (6)
1205 RN  24.0000  12005.0000  São Gonçalo do Amarante  103672 (7)
1526 PE  26.0000  5509.0000          Ferreiros          12170 (8)
1937 BA  29.0000  9208.0000  Coronel João Sá          15717 (9)
1973 BA  29.0000  12004.0000          Ibiassucê          9031 (10)
2159 BA  29.0000  27101.0000         Rodelas           9442 (11)
2237 BA  29.0000  33208.0000          Vera Cruz          43716 (12)
[16, 110, 123, 134, 210, 579, 1205, 1526, 1937, 1973, 2159, 2237]
df_populacao.loc[[list_valores_nao_numericos, 'POPULAÇÃO ESTIMADA']]
      UF  COD. UF  COD. MUNIC          NOME DO MUNICÍPIO  POPULAÇÃO ESTIMADA
16   RO  11.0000  205.0000          Porto Velho          539354
110  AM  13.0000  2553.0000         Manaus             33049
123  AM  13.0000  3601.0000  Santa Isabel do Rio Negro  25865
134  AM  13.0000  4302.0000          Urucará           16130
210  PA  15.0000  3754.0000  Jacareacanga          7590
579  MA  21.0000  7506.0000          Paço do Lumiar       123747
1205 RN  24.0000  12005.0000  São Gonçalo do Amarante  103672
1526 PE  26.0000  5509.0000          Ferreiros          12170
1937 BA  29.0000  9208.0000  Coronel João Sá          15717
1973 BA  29.0000  12004.0000          Ibiassucê          9031
```

```

2159 BA 29.0000 27101.0000           Rodelas          9442
2237 BA 29.0000 33208.0000           Vera Cruz         43716
UF          object
COD. UF      object
COD. MUNIC    object
NOME DO MUNICÍPIO object
POPULAÇÃO ESTIMADA int64
dtype: object
    POPULAÇÃO ESTIMADA
count      5570.0000
mean       38017.1799
std        222892.9921
min        776.0000
25%       5442.2500
50%       11665.5000
75%       25663.7500
max       12325232.0000
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   UF          5570 non-null   object  
 1   COD. UF     5570 non-null   object  
 2   COD. MUNIC  5570 non-null   object  
 3   NOME DO MUNICÍPIO 5570 non-null   object  
 4   POPULAÇÃO ESTIMADA 5570 non-null   int64  
dtypes: int64(1), object(4)
memory usage: 390.1+ KB
None
Index(['UF', 'COD. UF', 'COD. MUNIC', 'NOME DO MUNICÍPIO',
       'POPULAÇÃO ESTIMADA'],
      dtype='object')
*****
FIM df_populacao
*****
*****
INICIO df_renda
*****
Colunas do df_renda:
Index(['Ano', 'Código da Grande Região', 'Nome da Grande Região',
       'Código da Unidade da Federação', 'Sigla da Unidade da Federação',
       'Nome da Unidade da Federação', 'Código do Município',
       'Nome do Município', 'Região Metropolitana', 'Código da Mesorregião',
       'Nome da Mesorregião', 'Código da Microrregião', 'Nome da Microrregião',
       'Código da Região Geográfica Imediata',
       'Nome da Região Geográfica Imediata',
       'Município da Região Geográfica Imediata',
       'Código da Região Geográfica Intermediária',
       'Nome da Região Geográfica Intermediária',
       'Município da Região Geográfica Intermediária',
       'Código Concentração Urbana', 'Nome Concentração Urbana',
       'Tipo Concentração Urbana', 'Código Arranjo Populacional',
       'Nome Arranjo Populacional', 'Hierarquia Urbana',
       'Hierarquia Urbana (principais categorias)', 'Código da Região Rural',
       'Nome da Região Rural',
       'Região rural (segundo classificação do núcleo)', 'Amazônia Legal',
       'Semiárido', 'Cidade-Região de São Paulo',
       'Valor adicionado bruto da Agropecuária, \na preços correntes\n(R$ 1.000)',
       'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)',
       'Valor adicionado bruto dos Serviços,\nna preços correntes \n- exceto Administração, defesa, educação e saúde públicas e seguridade social\n(R$ 1.000)',
       'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, \nna preços correntes\n(R$ 1.000)',
       'Valor adicionado bruto total, \nna preços correntes\n(R$ 1.000)',
       'Impostos, líquidos de subsídios, sobre produtos, \nna preços correntes\n(R$ 1.000)',
       'Produto Interno Bruto, \nna preços correntes\n(R$ 1.000)',
       'Produto Interno Bruto per capita, \nna preços correntes\n(R$ 1,00)',
       'Atividade com maior valor adicionado bruto',
       'Atividade com segundo maior valor adicionado bruto',
       'Atividade com terceiro maior valor adicionado bruto'],
      dtype='object')
Informações do df_renda:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61255 entries, 0 to 61254
Data columns (total 43 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Ano          61255 non-null   int64  
 1   Código       61255 non-null   object  
 2   Nome         61255 non-null   object  
 3   Unidade      61255 non-null   object  
 4   Federação    61255 non-null   int64  
 5   ...
 6   ...
 7   ...
 8   ...
 9   ...
 10  ...
 11  ...
 12  ...
 13  ...
 14  ...
 15  ...
 16  ...
 17  ...
 18  ...
 19  ...
 20  ...
 21  ...
 22  ...
 23  ...
 24  ...
 25  ...
 26  ...
 27  ...
 28  ...
 29  ...
 30  ...
 31  ...
 32  ...
 33  ...
 34  ...
 35  ...
 36  ...
 37  ...
 38  ...
 39  ...
 40  ...
 41  ...
 42  ...

```

4		Sigla	da	Unidade	da	Federação
61255 non-null	object	Nome	da	Unidade	da	Federação
5			Código		do	Município
61255 non-null	object		Nome		do	Município
6				Região		Metropolitana
61255 non-null	int64		Código		da	Mesorregião
7						
61255 non-null	object		Nome		da	Mesorregião
8				Região		
15740 non-null	object		Código		da	Mesorregião
9						
61255 non-null	int64		Nome		da	Microrregião
10						
61255 non-null	object		Código		da	Microrregião
11						
61255 non-null	int64		Nome		da	Microrregião
12						
61255 non-null	object		Código	da	Região	Geográfica
13						Imediata
61255 non-null	int64		Nome	da	Região	Geográfica
14						Imediata
61255 non-null	object		Município	da	Região	Geográfica
15						Imediata
61255 non-null	object		Código	da	Região	Geográfica
16						Intermediária
61255 non-null	int64		Nome	da	Região	Geográfica
17						Intermediária
61255 non-null	object		Município	da	Região	Geográfica
18						Intermediária
61255 non-null	object		Código		Concentração	Urbana
19						
7251 non-null	float64		Nome		Concentração	Urbana
20						
7251 non-null	object		Tipo		Concentração	Urbana
21						
7251 non-null	object		Código		Arranjo	Populacional
22						
10507 non-null	float64		Nome		Arranjo	Populacional
23						
10507 non-null	object		Hierarquia			Urbana
24						
61255 non-null	object		Hierarquia	Urbana	(principais	categorias)
25						
61255 non-null	object		Código	da	Região	Rural
26						
61255 non-null	int64		Nome	da	Região	Rural
27						
61255 non-null	object		Região	rural	(segundo	classificação
28						do
61255 non-null	object					núcleo)
29					Amazônia	
61255 non-null	object					Legal
30						
61255 non-null	object		Cidade-Região	de	São	Semiárido
31						
61255 non-null	object					
32	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)				61255 non-null	float64
33	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)				61255 non-null	
float64						
34	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)				61255 non-	
35	Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)		61255 non-null	float64		
36	Valor adicionado bruto total, a preços correntes (R\$ 1.000)				61255 non-	
float64						
37	Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)				61255 non-null	float64
38	Produto Interno Bruto, a preços correntes (R\$ 1.000)				61255 non-null	
non-null	float64					
39	Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)				61255 non-null	
float64						
40	Atividade	com	maior	valor	adicionado	bruto
61255 non-null	object					

```

41          Atividade      com     segundo     maior     valor adicionado     bruto
61255 non-null   object      com     terceiro     maior     valor adicionado     bruto
42          Atividade      com     terceiro     maior     valor adicionado     bruto
61255 non-null   object
dtypes: float64(10), int64(9), object(24)
memory usage: 20.1+ MB
None
Informações do DataFrame após filtragem:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 55685 to 61254
Data columns (total 43 columns):
 #           Column
Non-Null Count Dtype   Column
---          -----
0            Ano
5570 non-null int64
1            Código
5570 non-null int64
2            Nome
5570 non-null object
3            Código
5570 non-null int64
4            Sigla
5570 non-null object
5            Nome
5570 non-null object
6            Código
5570 non-null int64
7            Nome
5570 non-null object
8            Região
1432 non-null object
9            Código
5570 non-null int64
10           Nome
5570 non-null object
11           Código
5570 non-null int64
12           Nome
5570 non-null object
13           Código
5570 non-null int64
14           Nome
5570 non-null object
15           Município
5570 non-null object
16           Código
5570 non-null int64
17           Nome
5570 non-null object
18           Município
5570 non-null object
19           Código
660 non-null  float64
20           Nome
660 non-null  object
21           Tipo
660 non-null  object
22           Código
956 non-null  float64
23           Nome
956 non-null  object
24           Hierarquia
5570 non-null object
25           Hierarquia
5570 non-null object
26           Código
5570 non-null int64
27           Nome
5570 non-null object
28           Região
5570 non-null object
29           rural
5570 non-null object
30           (segundo
5570 non-null object
31           Cidade-Região
5570 non-null object
32           Valor adicionado bruto da Agropecuária,
a preços correntes
(R$ 1.000)
33           Valor adicionado bruto da Indústria,
a preços correntes
(R$ 1.000)
float64
34           Valor adicionado bruto dos Serviços,
a preços correntes
5570 non-null  float64
5570 non-null

```

```

- exceto Administração, defesa, educação e saúde públicas e seguridade social
(R$ 1.000) 5570 non-null float64
35 Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social,
a preços correntes
(R$ 1.000) 5570 non-null float64
36 Valor adicionado bruto total,
a preços correntes
(R$ 1.000) 5570 non-null float64
37 Impostos, líquidos de subsídios, sobre produtos,
a preços correntes
(R$ 1.000) 5570 non-null float64
38 Produto Interno Bruto,
a preços correntes
(R$ 1.000) 5570 non-null float64
39 Produto Interno Bruto per capita,
a preços correntes
(R$ 1,00) 5570 non-null float64
40 Atividade com maior valor adicionado bruto
5570 non-null object com segundo maior valor adicionado bruto
5570 non-null object com terceiro maior valor adicionado bruto
5570 non-null object
dtypes: float64(10), int64(9), object(24)
memory usage: 1.9+ MB
None
Informações do df_renda após seleção e renomeação de colunas:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 55685 to 61254
Data columns (total 9 columns):
 #                                         Column
Non-Null Count Dtype   ---  -----
0                                         Nome_da_Grande_Região
5570 non-null object
1                                         Código          do        Município
5570 non-null object
2     Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
5570 non-null float64
3     Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
5570 non-null float64
4     Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde
públicas e seguridade social (R$ 1.000) 5570 non-null float64
5     Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços
correntes (R$ 1.000) 5570 non-null float64
6     Valor adicionado bruto total, a preços correntes (R$ 1.000)
5570 non-null float64
7     Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
5570 non-null float64
8     Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
5570 non-null float64
dtypes: float64(7), object(2)
memory usage: 435.2+ KB
None
Estatísticas Descritivas do df_renda:
    Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \
count           5570.0000
mean            78028.9048
std             163678.1503
min              0.0000
25%            13648.5425
50%            33817.6590
75%            79504.5878
max            3533041.0140

    Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \
count           5570.0000
mean            266487.7917
std             1502496.6284
min             440.6840
25%            4577.6773
50%            15101.2145
75%            86468.6263
max            58077783.6010

    Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e
saúde públicas e seguridade social (R$ 1.000) \
count           5570.0000
mean            633586.8941
std             7911585.4041
min             2545.3280
25%            22929.6870
50%            58109.2225
75%            182307.4985

```

```

max                         520357968.8930

    Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a
    preços correntes (R$ 1.000) \
count                      5570.0000
mean                       205906.6427
std                        1842211.5109
min                        8946.7740
25%                        30534.5590
50%                        58373.7205
75%                        125388.7250
max                        111267001.3810

    Valor adicionado bruto total, a preços correntes (R$ 1.000) \
count                      5570.0000
mean                       1184010.2334
std                        10379905.6585
min                        16344.8180
25%                        92311.7995
50%                        202598.8070
75%                        536598.2435
max                        624409861.1530

    Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
count                      5570.0000
mean                       182165.1706
std                        2027286.1168
min                        395.8550
25%                        4092.6925
50%                        11776.5065
75%                        45373.4298
max                        124349145.8120

    Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
count                      5570.0000
mean                       27458.3039
std                        28114.5488
min                        4924.0400
25%                        11573.5175
50%                        20159.7350
75%                        33931.1250
max                        591101.1100
*****
FIM df_renda
*****
*****df_dados_demograficos = df_renda + df_populacao
*****
Exemplo de dados nas colunas 'COD_UF', 'COD_MUNIC' e 'COD. UF + COD. MUNI':
0   11
1   11
2   11
3   11
4   11
Name: COD. UF, dtype: object
0  000150
1  000230
2  000310
3  000490
4  000560
Name: COD. MUNIC, dtype: object
0  1100015
1  1100023
2  1100031
3  1100049
4  1100056
Name: COD. UF + COD. MUNIC, dtype: object
Informações sobre o df_dados_demograficos:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 12 columns):
 #                                         Column
Non-Null Count Dtype
---             -----
0                           Nome_da_Grande_Região
5570 non-null  object
1                           Código
5570 non-null  object
2                           Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
5570 non-null  float64
3                           Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
5570 non-null  float64
4   Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde
públicas e seguridade social (R$ 1.000) 5570 non-null  float64
5   Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços
correntes (R$ 1.000)                  5570 non-null  float64

```

```

6          Valor adicionado bruto total, a preços correntes (R$ 1.000)
5570 non-null float64
7          Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
5570 non-null float64
8          Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
5570 non-null float64
9                                         UF
5570 non-null object
10                                     NOME
5570 non-null object
11                                     POPULAÇÃO
12                                         ESTIMADA
5570 non-null int64
dtypes: float64(7), int64(1), object(4)
memory usage: 565.7+ KB
None
As 5 Primeiras Linhas do DataFrame Combinado:
   Nome_da_Grande_Região Código do Município \
0           Norte            1100015
1           Norte            1100023
2           Norte            1100031
3           Norte            1100049
4           Norte            1100056

   Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \
0                  203394.4190
1                  199722.7100
2                  81177.1010
3                  236214.9520
4                  94757.6390

   Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \
0                  20716.3980
1                  404751.7570
2                  5438.0330
3                  275536.8230
4                  23582.3300

   Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde
públicas e segurança social (R$ 1.000) \
0                  150192.4340
1                  1207405.0680
2                  28666.9660
3                  1157343.9830
4                  276755.1060

   Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços
correntes (R$ 1.000) \
0                  160859.6300
1                  710513.4070
2                  44671.0740
3                  575806.3910
4                  115651.8540

   Valor adicionado bruto total, a preços correntes (R$ 1.000) \
0                  535162.8810
1                  2522392.9430
2                  159953.1750
3                  2244902.1480
4                  510746.9290

   Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
0                  35109.4450
1                  295655.6750
2                  7236.6360
3                  274450.5290
4                  89923.1460

   Produto Interno Bruto per capita, a preços correntes (R$ 1,00) UF \
0                  25091.1800      RO
1                  25730.2000      RO
2                  32226.2500      RO
3                  29331.2900      RO
4                  37069.2500      RO

   NOME DO MUNICÍPIO  POPULAÇÃO ESTIMADA
0 Alta Floresta D'Oeste        22728
1 Ariquemes                   109523
2 Cabixi                       5188
3 Cacoal                      85893
4 Cerejeiras                  16204
*****
FIM df_dados_demograficos = df_renda + df_populacao
*****
***** Início df_dados_demograficos completo *****
***** Nomes das colunas após reordenamento:
Index(['NOME DO MUNICÍPIO', 'Código do Município', 'UF',
       'POPULAÇÃO', 'ESTIMADA'],
      dtype='object')

```

```

'Nome_da_Grande_Região', 'POPULAÇÃO ESTIMADA',
'Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)',
'Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)',
'Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e
saúde públicas e seguridade social (R$ 1.000)',
'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a
preços correntes (R$ 1.000)',
'Valor adicionado bruto total, a preços correntes (R$ 1.000)',
'Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)',
'Produto Interno Bruto per capita, a preços correntes (R$ 1,00)'],
dtype='object')
Informações sobre o DataFrame:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   NOME            5570 non-null    object  
 1   Código           5570 non-null    object  
 2   do               5570 non-null    object  
 3   MUNICÍPIO        5570 non-null    object  
 4   POPULAÇÃO       5570 non-null    int64  
 5   Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)  5570 non-null    float64 
 6   Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)    5570 non-null    float64 
 7   Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde
públicas e seguridade social (R$ 1.000)  5570 non-null    float64 
 8   Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços
correntes (R$ 1.000)                   5570 non-null    float64 
 9   Valor adicionado bruto total, a preços correntes (R$ 1.000)             5570 non-null    float64 
 10  Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)  5570 non-null    float64 
 11  Produto Interno Bruto per capita, a preços correntes (R$ 1,00)            5570 non-null    float64 
dtypes: float64(7), int64(1), object(4)
memory usage: 565.7+ KB
None
*****
FIM df_dados_demoograficos completo
*****
***** Início df_eleitos *****
Contagem de Valores de VR_DESPESA_MAX_CAMPANHA
Contagem_positivos: 63448
Contagem_negativos: 5698
DataFrame df_eleitos_vices antes da remoção de valores negativos:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5587 entries, 4 to 69144
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   DS_CARGO         5587 non-null    object  
 1   TP_AGREMIACAO   5587 non-null    object  
 2   SG_PARTIDO       5587 non-null    object  
 3   NM_PARTIDO       5587 non-null    object  
 4   ST_REELEICAO     5587 non-null    object  
 5   VR_DESPESA_MAX_CAMPANHA 5587 non-null    float64 
 6   capital          5587 non-null    int64  
 7   codigo_ibge      5587 non-null    object  
dtypes: float64(1), int64(1), object(6)
memory usage: 392.8+ KB
None
DataFrame df_eleitos_vices depois da remoção de valores negativos:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   DS_CARGO         0 non-null     object  
 1   TP_AGREMIACAO   0 non-null     object  
 2   SG_PARTIDO       0 non-null     object  
 3   NM_PARTIDO       0 non-null     object  
 4   ST_REELEICAO     0 non-null     object  
 5   VR_DESPESA_MAX_CAMPANHA 0 non-null     float64 
 6   capital          0 non-null     int64  
 7   codigo_ibge      0 non-null     object  
dtypes: float64(1), int64(1), object(6)
memory usage: 0.0+ bytes

```

```

None
Informações sobre o DataFrame df_eleitos_IBGE:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63448 entries, 0 to 69145
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   DS_CARGO          63448 non-null   object  
 1   TP AGREMIACAO    63448 non-null   object  
 2   SG_PARTIDO        63448 non-null   object  
 3   NM_PARTIDO        63448 non-null   object  
 4   ST_REELEICAO     63448 non-null   object  
 5   VR_DESPESA_MAX_CAMPANHA 63448 non-null   float64 
 6   capital           63448 non-null   int64   
 7   codigo_ibge       63448 non-null   object  
dtypes: float64(1), int64(1), object(6)
memory usage: 4.4+ MB
None
*****
FIM df_eleitos
*****
*****INICIO df_eleitos_dados_demograficos = df_eleitos_IBGE + df_dados_demograficos*****
*****Informações sobre o df_eleitos_dados_demograficos após a definição dos tipos das colunas:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63448 entries, 18569 to 32662
Data columns (total 19 columns):
 #           Column
Non-Null Count  Dtype
--- 
 0           NOME      DO      MUNICÍPIO
 1           Código    do      Município
 2           UF        Nome_da_Grande_Região
 3           POPULAÇÃO ESTIMADA
 4           capital
 5           DS_CARGO
 6           category
 7           VR_DESPESA_MAX_CAMPANHA
 8           SG_PARTIDO
 9           NM_PARTIDO
 10          category
 11          TP AGREMIACAO
 12          ST_REELEICAO
 13          Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
 14          Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e segurança social (R$ 1.000) 63448 non-null float64
 15          Valor adicionado bruto da Administração, defesa, educação e saúde públicas e segurança social, a preços correntes (R$ 1.000) 63448 non-null float64
 16          Valor adicionado bruto total, a preços correntes (R$ 1.000)
 17          Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
 18          Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
dtypes: category(8), float64(8), int64(1), object(2)
memory usage: 6.3+ MB
None
Tipos de dados das colunas após a conversão:
NOME           DO      MUNICÍPIO
object
Código         do      Município
object
UF
category
Nome_da_Grande_Região
category
POPULAÇÃO     ESTIMADA
int64
capital
category
DS_CARGO
category

```

```

VR_DESPESA_MAX_CAMPANHA
float64
SG_PARTIDO
category
NM_PARTIDO
category
TP_AGREMIAÇÃO
category
ST_REELEICAO
category
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
float64
Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
float64
Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) float64
Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000)
float64
Valor adicionado bruto total, a preços correntes (R$ 1.000)
float64
Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
float64
Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
float64
dtype: object
Amostra aleatória do df_eleitos_dados_demograficos:
   NOME DO MUNICÍPIO Código do Município UF Nome_da_Grande_Região \
7401    Caldas Novas          5204508 GO Centro-oeste
18439   Tabuleiro            3167905 MG Sudeste
19387   Marcelândia          5105580 MT Centro-oeste
33735   Vespasiano           3171204 MG Sudeste
37701   Nazária              2206720 PI Nordeste

   POPULAÇÃO ESTIMADA capital DS_CARGO VR_DESPESA_MAX_CAMPANHA \
7401             93196     0 VEREADOR      87179.6200
18439            3708     0 VEREADOR      12307.7500
19387            10301     0 VEREADOR      12307.7500
33735            129765     0 VEREADOR      69439.5500
37701            8602     0 VEREADOR      29484.3500

   SG_PARTIDO          NM_PARTIDO      TP_AGREMIAÇÃO \
7401      PDT        PARTIDO DEMOCRÁTICO TRABALHISTA PARTIDO ISOLADO
18439      MDB        MOVIMENTO DEMOCRÁTICO BRASILEIRO PARTIDO ISOLADO
19387      DEM        DEMOCRATAS PARTIDO ISOLADO
33735      PSDB       PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA PARTIDO ISOLADO
37701      PT         PARTIDO DOS TRABALHADORES PARTIDO ISOLADO

   ST_REELEICAO \
7401      N
18439      N
19387      S
33735      N
37701      N

   Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \
7401                  220701.7030
18439                  6269.9240
19387                  209208.9010
33735                  826.7390
37701                  5941.7780

   Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \
7401                  458566.5090
18439                  9497.4890
19387                  46813.2950
33735                  705887.0120
37701                  12737.6730

   Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) \
7401                  1198885.6690
18439                  22602.1510
19387                  113098.0900
33735                  989792.9840
37701                  19147.9930

   Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000) \
7401                  479922.9460
18439                  20092.0970
19387                  68825.3920
33735                  582553.7620
37701                  50779.9800

   Valor adicionado bruto total, a preços correntes (R$ 1.000) \
7401                  2358076.8270
18439                  58461.6610
19387                  437945.6770

```



```

1                                     Código          do        Município
57956 non-null  object
2                                     UF
57956 non-null  category
3                                     Nome_da_Grande_Região
57956 non-null  category
4                                     POPULAÇÃO      ESTIMADA
57956 non-null  int64
5                                     capital
57956 non-null  category
6                                     DS_CARGO
57956 non-null  category
7                                     VR_DESPESA_MAX_CAMPANHA
57956 non-null  float64
8                                     SG_PARTIDO
57956 non-null  category
9                                     NM_PARTIDO
57956 non-null  category
10                                    TP_AGREMIACAO
57956 non-null  category
11                                    ST_REELEICAO
57956 non-null  category
12    Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
57956 non-null float64
13    Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
57956 non-null float64
14 Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) 57956 non-null float64
15 Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000) 57956 non-null float64
16    Valor adicionado bruto total, a preços correntes (R$ 1.000)
57956 non-null float64
17    Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
57956 non-null float64
18    Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
57956 non-null float64
dtypes: category(8), float64(8), int64(1), object(2)
memory usage: 5.8+ MB
None

```

Amostra de 5 registros de df\_prefeitos:

	NOME DO MUNICÍPIO	Código do Município	UF	Nome_da_Grande_Região
9008	São Jorge do Patrocínio	4125357	PR	Sul
9284	Anajás	1500701	PA	Norte
29434	Ponte Nova	3152105	MG	Sudeste
34864	Campos Altos	3111507	MG	Sudeste
59251	Dom Feliciano	4306502	RS	Sul

	POPULAÇÃO	ESTIMADA	capital	DS_CARGO	VR_DESPESA_MAX_CAMPANHA
9008	5586	0	PREFEITO		123077.4200
9284	29688	0	PREFEITO		123077.4200
29434	59875	0	PREFEITO		207981.5400
34864	15563	0	PREFEITO		123077.4200
59251	15487	0	PREFEITO		123077.4200

	SG_PARTIDO	PARTIDO SOCIAL DEMOCRÁTICO	NM_PARTIDO	TP_AGREMIACAO
9008	PSD	PARTIDO SOCIAL DEMOCRÁTICO	COLIGAÇÃO	
9284	PSDB	PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA	COLIGAÇÃO	
29434	PSB	PARTIDO SOCIALISTA BRASILEIRO	COLIGAÇÃO	
34864	PV	PARTIDO VERDE	COLIGAÇÃO	
59251	PTB	PARTIDO TRABALHISTA BRASILEIRO	COLIGAÇÃO	

	ST_REELEICAO
9008	N
9284	N
29434	S
34864	S
59251	S

	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)
9008	40424.5120
9284	35373.7720
29434	45439.5330
34864	128979.7220
59251	92878.0810

	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)
9008	21698.2450
9284	6753.0110
29434	408049.3060
34864	15389.6120
59251	8593.7680

	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)
9008	74693.6000
9284	33371.9140
29434	988301.4870

34864	132467.3110
59251	73200.1750

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

9008	36986.7260
9284	142404.0520
29434	284968.0620
34864	74131.0580
59251	83067.1050

Valor adicionado bruto total, a preços correntes (R\$ 1.000) \

9008	173803.0830
9284	217902.7500
29434	1726758.3870
34864	350967.7030
59251	257739.1290

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \

9008	17036.5300
9284	10791.2740
29434	257040.3630
34864	20614.4460
59251	12539.2090

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

9008	34163.9100
9284	7703.2500
29434	33132.3400
34864	23876.0000
59251	17451.9500

Amostra de 5 registros de df\_vereadores:

	NOME DO MUNICÍPIO	Código do Município	UF	Nome_da_Grande_Região	\
324	São José da Tapera	2708402	AL	Nordeste	
6606	Mathias Lobato	3171501	MG	Sudeste	
18286	Remanso	2926004	BA	Nordeste	
14781	Bandeira	3105202	MG	Sudeste	
58926	Santa Cruz	2513208	PB	Nordeste	

POPULAÇÃO ESTIMADA capital DS\_CARGO VR\_DESPESA\_MAX\_CAMPANHA \

324	32405	0	VEREADOR	37215.3500
6606	3179	0	VEREADOR	12307.7500
18286	41170	0	VEREADOR	45546.3700
14781	4766	0	VEREADOR	12307.7500
58926	6581	0	VEREADOR	12307.7500

SG\_PARTIDO NM\_PARTIDO TP\_AGRÉMIAÇÃO \

324	MDB	MOVIMENTO DEMOCRÁTICO BRASILEIRO	PARTIDO ISOLADO
6606	PSDB	PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA	PARTIDO ISOLADO
18286	PC do B	PARTIDO COMUNISTA DO BRASIL	PARTIDO ISOLADO
14781	AVANTE	AVANTE	PARTIDO ISOLADO
58926	PL	PARTIDO LIBERAL	PARTIDO ISOLADO

ST\_REELEICAO \

324	S
6606	N
18286	N
14781	N
58926	N

Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000) \

324	33181.2000
6606	4143.2570
18286	69984.1550
14781	10160.4680
58926	3967.6170

Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000) \

324	20930.5050
6606	1864.6430
18286	13533.7940
14781	1878.3280
58926	2725.0230

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \

324	69295.5810
6606	13224.4980
18286	159617.8660
14781	12766.6470
58926	15125.4140

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

324	150620.2060
6606	20992.6060
18286	159333.2720

```

14781           24937.9220
58926           33408.3870

    Valor adicionado bruto total, a preços correntes (R$ 1.000) \
324           274027.4900
6606          40225.0030
18286          402469.0880
14781          49743.3650
58926          55226.4420

    Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
324           9894.5010
6606          1979.6230
18286          22659.2560
14781          1602.1370
58926          3121.5560

    Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
324           8761.6700
6606          13276.0700
18286          10326.1700
14781          10773.2900
58926          8866.1300

Variáveis Categóricas:
Coluna: UF
MG      9322
SP      7585
RS      5382
BA      4971
PR      4264
SC      3184
GO      2762
MA      2642
PB      2428
CE      2371
PI      2366
PE      2330
PA      1906
RN      1819
MT      1477
TO      1446
RJ      1272
AL      1187
ES      936
MS      921
SE      878
AM      799
RO      585
AC      253
AP      190
RR      172
Name: UF, dtype: int64

Coluna: Nome_da_Grande_Região
Nordeste     20992
Sudeste     19115
Sul        12830
Norte       5351
Centro-oeste 5160
Name: Nome_da_Grande_Região, dtype: int64

Coluna: capital
0          62607
1           841
Name: capital, dtype: int64

Coluna: DS_CARGO
VEREADOR    57956
PREFEITO    5492
Name: DS_CARGO, dtype: int64

Coluna: SG_PARTIDO
MDB         8137
PP          7032
PSD         6338
PSDB        4911
DEM         4789
PL          3799
PDT         3748
PSB         3257
PT          2839
REPUBLICANOS 2802
PTB         2675
CIDADANIA   1732

```

```

PODE      1634
PSC       1617
SOLIDARIEDADE 1442
PSL       1300
AVANTE    1125
PV        853
PROS      802
PATRIOTA  769
PC do B   741
PRTB      222
PTC       219
PMN       213
REDE      152
DC        125
PSOL      96
PMB       49
NOVO     30
Name: SG_PARTIDO, dtype: int64

```

```

Coluna: NM_PARTIDO
MOVIMENTO DEMOCRÁTICO BRASILEIRO      8137
PROGRESSISTAS                         7032
PARTIDO SOCIAL DEMOCRÁTICO            6338
PARTIDO DA SOCIAL DEMOCRACIA BRASILEIRA 4911
DEMOCRATAS                            4789
PARTIDO LIBERAL                         3799
PARTIDO DEMOCRÁTICO TRABALHISTA        3748
PARTIDO SOCIALISTA BRASILEIRO          3257
PARTIDO DOS TRABALHADORES             2839
REPUBLICANOS                          2802
PARTIDO TRABALHISTA BRASILEIRO         2675
CIDADANIA                             1732
PODEMOS                               1634
PARTIDO SOCIAL CRISTÃO                1617
SOLIDARIEDADE                         1442
PARTIDO SOCIAL LIBERAL                1300
AVANTE                                1125
PARTIDO VERDE                           853
PARTIDO REPUBLICANO DA ORDEM SOCIAL   802
PATRIOTA                               769
PARTIDO COMUNISTA DO BRASIL           741
PARTIDO RENOVADOR TRABALHISTA BRASILEIRO 222
PARTIDO TRABALHISTA CRISTÃO           219
PARTIDO DA MOBILIZAÇÃO NACIONAL       213
REDE SUSTENTABILIDADE                 152
DEMOCRACIA CRISTÃ                     125
PARTIDO SOCIALISMO E LIBERDADE        96
PARTIDO DA MULHER BRASILEIRA          49
PARTIDO NOVO                           30
Name: NM_PARTIDO, dtype: int64

```

```

Coluna: TP_AGREMIACAO
PARTIDO ISOLADO      58503
COLIGAÇÃO           4945
Name: TP_AGREMIACAO, dtype: int64

```

```

Coluna: ST_REELEICAO
N      51777
S      11671
Name: ST_REELEICAO, dtype: int64

```

```

Análise ANOVA para df_eleitos_dados_demoograficos (colunas categóricas)
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  ValueWarning: covariance of constraints does not have full rank. The number of constraints is 28, but rank is 26
    warnings.warn('covariance of constraints does not have full ')
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  ValueWarning: covariance of constraints does not have full rank. The number of constraints is 28, but rank is 26
    warnings.warn('covariance of constraints does not have full ')
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  ValueWarning: covariance of constraints does not have full rank. The number of constraints is 25, but rank is 4
    warnings.warn('covariance of constraints does not have full ')
Resultados da Análise de Variância (ANOVA) para as colunas: DS_CARGO, TP_AGREMIACAO, SG_PARTIDO, NM_PARTIDO, ST_REELEICAO, capital, UF, Nome_da_Grande_Região
              sum_sq      df      F  PR(>F)
C(DS_CARGO)      10132758441165.1270  1.0000  198.0481  0.0000
C(TP_AGREMIACAO)  4994050246960.6172  1.0000  97.6103  0.0000
C(SG_PARTIDO)     95691006891.2541  28.0000  0.0668  1.0000
C(NM_PARTIDO)     95691006866.0873  28.0000  0.0668  1.0000
C(ST_REELEICAO)   45377846450.3708  1.0000  0.8869  0.3463
C(capital)        618254698413249.5000  1.0000  12083.9888  0.0000
C(UF)             49689644749.0736  25.0000  0.0388  0.9971
C(Nome_da_Grande_Região) 7950343159.7408  4.0000  0.0388  0.9971
Residual          3243179687725583.0000 63389.0000      NaN      NaN
Análise ANOVA para df_prefeitos (colunas categóricas)

```

```

D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  Value-
Warning: covariance of constraints does not have full rank. The number of constraints is 28, but rank is 11
    warnings.warn('covariance of constraints does not have full ')
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  Value-
Warning: covariance of constraints does not have full rank. The number of constraints is 28, but rank is 11
    warnings.warn('covariance of constraints does not have full ')
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  Value-
Warning: covariance of constraints does not have full rank. The number of constraints is 25, but rank is 4
    warnings.warn('covariance of constraints does not have full ')
Resultados da Análise de Variância (ANOVA) para as colunas: DS_CARGO, TP_AGREMIACAO, SG_PARTIDO, NM_PARTIDO,
ST_REELEICAO, capital, UF, Nome_da_Grande_Região
      sum_sq        df         F   PR(>F)
C(DS_CARGO)      7137591147470.2236    1.0000  19.9889  0.0000
C(TP_AGREMIACAO) 1159607861312.9319    1.0000  3.2475  0.0716
C(SG_PARTIDO)    95253070224708.0312   28.0000  9.5271  0.0000
C(NM_PARTIDO)    95253070224675.9219   28.0000  9.5271  0.0000
C(ST_REELEICAO)  1454791684499.3262    1.0000  4.0742  0.0436
C(capital)       828412094980748.8750   1.0000 2319.9785  0.0000
C(UF)            167700197045675.0625  25.0000 18.7859  0.0000
C(Nome_da_Grande_Região) 26832031527285.1328   4.0000 18.7859  0.0000
Residual         1940716183076238.2500  5435.0000   NaN    NaN
Análise ANOVA para df_vereadores (colunas categóricas)
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  Value-
Warning: covariance of constraints does not have full rank. The number of constraints is 28, but rank is 25
    warnings.warn('covariance of constraints does not have full ')
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  Value-
Warning: covariance of constraints does not have full rank. The number of constraints is 28, but rank is 25
    warnings.warn('covariance of constraints does not have full ')
D:\Users\arthu\Documents\Desenvolvimento\py\venv\lib\site-packages\statsmodels\base\model.py:1871:  Value-
Warning: covariance of constraints does not have full rank. The number of constraints is 25, but rank is 4
    warnings.warn('covariance of constraints does not have full ')
Resultados da Análise de Variância (ANOVA) para as colunas: DS_CARGO, TP_AGREMIACAO, SG_PARTIDO, NM_PARTIDO,
ST_REELEICAO, capital, UF, Nome_da_Grande_Região
      sum_sq        df         F   PR(>F)
C(DS_CARGO)      267123218.3903    1.0000  0.0215  0.8836
C(TP_AGREMIACAO) 210413192.7098    1.0000  0.0169  0.8966
C(SG_PARTIDO)    45577407456.1106   28.0000  0.1307  1.0000
C(NM_PARTIDO)    45577407487.2736   28.0000  0.1307  1.0000
C(ST_REELEICAO)  54885941079.2792    1.0000  4.4078  0.0358
C(capital)       406064912374621.5000  1.0000 32610.2576  0.0000
C(UF)            50915599212.2717   25.0000  0.1636  0.9569
C(Nome_da_Grande_Região) 8146495874.0437   4.0000  0.1636  0.9569
Residual         720961871636022.1250  57899.0000   NaN    NaN
Variáveis Numéricas :
Resumo estatístico das colunas numéricas de df_eleitos_dados_demograficos:
  POPULAÇÃO ESTIMADA VR_DESPESA_MAX_CAMPANHA \
count          63448.0000          63448.0000
mean           72561.4467          58129.7489
std             454035.7763        256943.2441
min            776.0000          12307.7500
25%           6063.0000          12307.7500
50%           14410.0000         12307.7500
75%           34168.2500          36847.9700
max          12325232.0000        30413484.3800

  Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \
count          63448.0000          63448.0000
mean           84653.3942          84653.3942
std             175465.5956          0.0000
min            0.0000
25%           14786.0110
50%           36342.5090
75%           85487.4350
max          3533041.0140

  Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \
count          63448.0000          63448.0000
mean           486616.5848          486616.5848
std             2633147.9934          440.6840
min            5332.9667
25%           19795.7890
50%           134240.9490
75%           58077783.6010
max          58077783.6010

  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e
saúde públicas e seguridade social (R$ 1.000) \
count          63448.0000          63448.0000
mean           1497145.0081          1497145.0081
std             16723303.4861          2545.3280
min            26333.1280
25%           72779.2680
50%           277546.2790
75%           520357968.8930
max          520357968.8930

  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a
preços correntes (R$ 1.000) \

```

```

count                                63448.0000
mean                                 354884.7230
std                                  2229150.2468
min                                  9372.7150
25%                                 33319.6132
50%                                 70336.9850
75%                                 165871.6600
max                                  54434941.7860

    Valor adicionado bruto total, a preços correntes (R$ 1.000) \
count                                63448.0000
mean                                 2423299.7102
std                                  20939170.7001
min                                  16344.8180
25%                                 103323.4615
50%                                 240508.1020
75%                                 739874.8005
max                                  624409861.1530

    Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
count                                63448.0000
mean                                 424084.9879
std                                  4236299.8516
min                                  395.8550
25%                                 4722.9430
50%                                 14851.0950
75%                                 67048.5290
max                                  124349145.8120

    Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
count                                63448.0000
mean                                 27770.3492
std                                  28425.5256
min                                  4924.0400
25%                                 11727.5600
50%                                 20513.0000
75%                                 34627.6900
max                                  591101.1100

df_eleitos_dados_demograficos :
VR_DESPESA_MAX_CAMPANHA
1.0000
POPULAÇÃO                           ESTIMADA
0.5724
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
0.0418
Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
0.5281
Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) 0.5459
Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000) 0.5285
Valor adicionado bruto total, a preços correntes (R$ 1.000)
0.5590
Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
0.5498
Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
0.0713
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
df_eleitos_prefeitos :
VR_DESPESA_MAX_CAMPANHA
1.0000
POPULAÇÃO                           ESTIMADA
0.7053
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
0.0952
Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
0.6402
Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R$ 1.000) 0.6010
Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R$ 1.000) 0.6560
Valor adicionado bruto total, a preços correntes (R$ 1.000)
0.6429
Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000)
0.6039
Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
0.1040
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
df_eleitos_vereadores :
VR_DESPESA_MAX_CAMPANHA
1.0000
POPULAÇÃO                           ESTIMADA
0.9401
Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000)
0.0459
Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000)
0.8503

```

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) 0.9139  
 Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) 0.8665  
 Valor adicionado bruto total, a preços correntes (R\$ 1.000)  
 0.9291  
 Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)  
 0.9195  
 Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)  
 0.0933  
 Name: VR\_DESPESA\_MAX\_CAMPANHA, dtype: float64  
 Estatísticas para VR\_DESPESA\_MAX\_CAMPANHA em df\_eleitos\_dados\_demograficos:  
 count 63448.0000  
 mean 58129.7489  
 std 256943.2441  
 min 12307.7500  
 25% 12307.7500  
 50% 12307.7500  
 75% 36847.9700  
 max 30413484.3800  
 Name: VR\_DESPESA\_MAX\_CAMPANHA, dtype: float64  
 Estatísticas para VR\_DESPESA\_MAX\_CAMPANHA em df\_prefeitos:  
 count 5492.0000  
 mean 257676.0976  
 std 708644.7012  
 min 101190.8700  
 25% 123077.4200  
 50% 123077.4200  
 75% 189702.3725  
 max 30413484.3800  
 Name: VR\_DESPESA\_MAX\_CAMPANHA, dtype: float64  
 Estatísticas para VR\_DESPESA\_MAX\_CAMPANHA em df\_vereadores:  
 count 57956.0000  
 mean 39220.4290  
 std 143408.1831  
 min 12307.7500  
 25% 12307.7500  
 50% 12307.7500  
 75% 26215.4800  
 max 3675197.1200  
 Name: VR\_DESPESA\_MAX\_CAMPANHA, dtype: float64  
 \*\*\*\*  
**INICIO analise\_de\_modelo\_VR\_DESPESA\_MAX\_CAMPANHA**  
 \*\*\*\*  
 Análise do modelo de regressão para df\_eleitos\_dados\_demograficos VR\_DESPESA\_MAX\_CAMPANHA com outliers:  
 Fitting 2 folds for each of 4 candidates, totalling 8 fits  
 [CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=0.839 total time= 39.3s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=0.952 total time= 39.5s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=0.915 total time= 54.8s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=0.954 total time= 55.1s  
 [CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=0.829 total time= 55.4s  
 [CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=0.824 total time= 55.9s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=0.901 total time= 1.2min  
 [CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=0.829 total time= 1.2min  
 Melhores hiperparâmetros encontrados:  
 {'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 200}  
 Melhor pontuação do modelo:  
 0.8955521919577892  
 Erro Quadrático Médio (MSE): 1652121185.475804  
 Erro Absoluto Médio (MAE): 695.463504329042  
 Coeficiente de Determinação (R<sup>2</sup>): 0.9780419685652934  
 Raiz do Erro Quadrático Médio (RMSE): 40646.29362532092  
  
 Número de outliers encontrados em df\_eleitos\_dados\_demograficos com outliers: 404  
 Percentagem de outliers em df\_eleitos\_dados\_demograficos com outliers: 0.64%  
 Estatísticas descritivas após remover os outliers:  
 count 63044.0000  
 mean 44945.8020  
 std 80860.9250  
 min 12307.7500  
 25% 12307.7500  
 50% 12307.7500  
 75% 35359.1200  
 max 825421.3700  
 Name: VR\_DESPESA\_MAX\_CAMPANHA, dtype: float64  
 Análise do modelo de regressão para df\_eleitos\_dados\_demograficos VR\_DESPESA\_MAX\_CAMPANHA sem outliers  
 Fitting 2 folds for each of 4 candidates, totalling 8 fits  
 [CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=1.000 total time= 39.5s  
 [CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=1.000 total time= 39.5s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=1.000 total time= 56.2s  
 [CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=1.000 total time= 56.3s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=1.000 total time= 56.3s  
 [CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=1.000 total time= 56.4s  
 [CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=1.000 total time= 1.2min  
 [CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=1.000 total time= 1.2min  
 Melhores hiperparâmetros encontrados:  
 {'learning\_rate': 0.1, 'max\_depth': 6, 'n\_estimators': 200}

Melhor pontuação do modelo:  
0.9999923664076009  
Erro Quadrático Médio (MSE): 9635.930306967815  
Erro Absoluto Médio (MAE): 17.22693488745662  
Coeficiente de Determinação ( $R^2$ ): 0.9999986159214753  
Raiz do Erro Quadrático Médio (RMSE): 98.16277454803229

Análise do modelo de regressão para df\_eleitos\_prefeitos VR\_DESPESA\_MAX\_CAMPANHA com outliers:  
Fitting 2 folds for each of 4 candidates, totalling 8 fits  
[CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=0.992 total time= 4.8s  
[CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=0.770 total time= 4.9s  
[CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=0.977 total time= 6.8s  
[CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=0.782 total time= 6.9s  
[CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=0.755 total time= 7.0s  
[CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=0.991 total time= 7.2s  
[CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=0.760 total time= 9.4s  
[CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=0.968 total time= 9.5s  
Melhores hiperparâmetros encontrados:  
{'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 200}  
Melhor pontuação do modelo:  
0.881043279383122  
Erro Quadrático Médio (MSE): 364647986735.79047  
Erro Absoluto Médio (MAE): 23901.909926113996  
Coeficiente de Determinação ( $R^2$ ): 0.6967545482288077  
Raiz do Erro Quadrático Médio (RMSE): 603860.9001548208

Número de outliers encontrados em df\_eleitos\_prefeitos com outliers: 49  
Percentagem de outliers em df\_eleitos\_prefeitos com outliers: 0.90%

Estatísticas descritivas após remover os outliers:  
count 5443.0000  
mean 213288.1814  
std 243314.1761  
min 101190.8700  
25% 123077.4200  
50% 123077.4200  
75% 184194.9550  
max 2369144.8300

Name: VR\_DESPESA\_MAX\_CAMPANHA, dtype: float64  
Análise do modelo de regressão para df\_eleitos\_prefeitos VR\_DESPESA\_MAX\_CAMPANHA sem outliers:  
Fitting 2 folds for each of 4 candidates, totalling 8 fits  
[CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=1.000 total time= 4.9s  
[CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=1.000 total time= 5.3s  
[CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=1.000 total time= 7.3s  
[CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=1.000 total time= 7.4s  
[CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=1.000 total time= 7.5s  
[CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=1.000 total time= 7.7s  
[CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=1.000 total time= 9.9s  
[CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=1.000 total time= 10.0s  
Melhores hiperparâmetros encontrados:  
{'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 300}  
Melhor pontuação do modelo:  
0.9997952252395954  
Erro Quadrático Médio (MSE): 2907196.225806929  
Erro Absoluto Médio (MAE): 458.5921115829642  
Coeficiente de Determinação ( $R^2$ ): 0.9999554651051135  
Raiz do Erro Quadrático Médio (RMSE): 1705.0502121072357

Análise do modelo de regressão para df\_eleitos\_vereadores VR\_DESPESA\_MAX\_CAMPANHA com outliers:  
Fitting 2 folds for each of 4 candidates, totalling 8 fits  
[CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=1.000 total time= 38.7s  
[CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=200;, score=1.000 total time= 39.2s  
[CV 1/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=1.000 total time= 52.7s  
[CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=1.000 total time= 52.9s  
[CV 2/2] END learning\_rate=0.1, max\_depth=4, n\_estimators=300;, score=1.000 total time= 53.6s  
[CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=200;, score=1.000 total time= 53.9s  
[CV 2/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=1.000 total time= 1.1min  
[CV 1/2] END learning\_rate=0.1, max\_depth=6, n\_estimators=300;, score=1.000 total time= 1.2min  
Melhores hiperparâmetros encontrados:  
{'learning\_rate': 0.1, 'max\_depth': 6, 'n\_estimators': 300}  
Melhor pontuação do modelo:  
0.999999678319551  
Erro Quadrático Médio (MSE): 567.025753089435  
Erro Absoluto Médio (MAE): 12.008811597191917  
Coeficiente de Determinação ( $R^2$ ): 0.9999999756142232  
Raiz do Erro Quadrático Médio (RMSE): 23.812302557489797

Número de outliers encontrados em df\_eleitos\_vereadores com outliers: 565  
Percentagem de outliers em df\_eleitos\_vereadores com outliers: 0.98%

Estatísticas descritivas após remover os outliers:  
count 57391.0000  
mean 29159.0550  
std 44342.3654  
min 12307.7500  
25% 12307.7500

```

50%      12307.7500
75%      25491.3350
max      451919.9400
Name: VR_DESPESA_MAX_CAMPANHA, dtype: float64
Análise do modelo de regressão para df_eleitos vereadores VR_DESPESA_MAX_CAMPANHA sem outliers:
Fitting 2 folds for each of 4 candidates, totalling 8 fits
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 36.0s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=200;, score=1.000 total time= 36.6s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 51.3s
[CV 2/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 51.3s
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=200;, score=1.000 total time= 51.9s
[CV 1/2] END learning_rate=0.1, max_depth=4, n_estimators=300;, score=1.000 total time= 52.4s
[CV 2/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.1min
[CV 1/2] END learning_rate=0.1, max_depth=6, n_estimators=300;, score=1.000 total time= 1.1min
Melhores hiperparâmetros encontrados:
{'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 300}
Melhor pontuação do modelo:
0.9999999673619201
Erro Quadrático Médio (MSE): 45.49154960411642
Erro Absoluto Médio (MAE): 3.300616266234919
Coeficiente de Determinação (R2): 0.9999999761198388
Raiz do Erro Quadrático Médio (RMSE): 6.744742367512374

```

```

*****
FIM analise_de_modelo_VR_DESPESA_MAX_CAMPANHA
*****
***** INICIO Análise Outliers *****
***** Média de VR_DESPESA_MAX_CAMPANHA por Município - df_eleitos_vereadores_outliers: *****
    NOME DO MUNICÍPIO   VR_DESPESA_MAX_CAMPANHA
14      São Paulo          3675197.1200
6       Governador Valadares 1602918.4400
12      Rio de Janeiro     1594577.3800
11      Recife                1011149.6500
9       Palmas                962451.1500
1       Campo Grande          732621.6600
7       Guarulhos             701644.0300
0       Belo Horizonte        692183.0200
8       Manaus                628500.4700
5       Goiânia                591983.4600
2       Cuiabá                560511.1900
15      Uberlândia            549424.6600
3       Curitiba              530525.5800
4       Fortaleza              524050.7000
13      São Luís              504192.2600
10      Porto Alegre          489142.8400
VR_DESPESA_MAX_CAMPANHA médio em capitais - df_eleitos_vereadores_outliers
961314.3446153847
VR_DESPESA_MAX_CAMPANHA médio em não capitais - df_eleitos_vereadores_outliers
951329.0433333335
***** VR_DESPESA_MAX_CAMPANHA por Município - df_eleitos_prefeitos_outliers: *****
    NOME DO MUNICÍPIO   VR_DESPESA_MAX_CAMPANHA
4       Belo Horizonte        30413484.3800
45      São Paulo              20719753.4700
39      Salvador              16722661.9900
12      Curitiba              10903325.6700
37      Rio de Janeiro         9049004.2200
32      Palmas                8846132.1400
8       Campo Grande           7609782.1700
27      Natal                  6254508.5200
28      Niterói                6164915.8300
16      Fortaleza              5654269.7400
0       Aparecida de Goiânia  5454813.7100
41      São Bernardo do Campo 5245993.6800
5       Betim                  5154458.6900
46      Uberlândia              4701654.9600
6       Camapuã                 4690263.4900
14      Florianópolis          4133220.9800
11      Cuiabá                  4103087.4600
43      São José dos Pinhais  4101496.7300
23      Manaus                  4090982.3600
7       Campina Grande          3981974.7900
3       Barueri                  3954040.8400
40      Santo André              3792580.6800
29      Nova Iguaçu              3770832.7100
33      Parauapebas              3712311.1100
34      Paulínia                  3365884.6100
26      Mossoró                  3364740.1900
13      Duque de Caxias          3224308.3700
25      Montes Claros             3042080.5900
36      Recife                  3010863.1700
9       Caruaru                  2983811.9400
31      Osasco                  2982657.6400
47      Vitória                  2942611.2700
22      Macaé                   2936736.0400
38      Rondonópolis              2924437.9900

```

20 Jundiaí 2861007.0400  
 48 Várzea Grande 2839285.3400  
 30 Nova Lima 2826120.3500  
 2 Araçatuba 2781588.8000  
 42 São José do Rio Preto 2722689.8600  
 19 Jataí 2716607.5800  
 21 Lucas do Rio Verde 2686807.4200  
 18 Gurupi 2678796.4400  
 35 Porto Alegre 2665432.6700  
 15 Formosa 2649828.6100  
 44 São Miguel dos Campos 2633359.6700  
 17 Goiânia 2589653.4400  
 24 Marília 2557227.0100  
 1 Araraquara 2553032.8400  
 10 Contagem 2464437.4000  
 Média da VR\_DESPESA\_MAX\_CAMPANHA em Capitais - df\_eleitos\_prefeitos\_outliers  
 8731798.353125  
 Média da VR\_DESPESA\_MAX\_CAMPANHA em Não Capitais - df\_eleitos\_prefeitos\_outliers  
 3470326.756060603  
 Média de VR\_DESPESA\_MAX\_CAMPANHA por UF - df\_eleitos\_vereadores\_outliers:  
 UF VR\_DESPESA\_MAX\_CAMPANHA \  
 24 SP 2539233.0182  
 17 RJ 1594577.3800  
 14 PE 1011149.6500  
 25 TO 962451.1500  
 9 MG 863766.9312  
 10 MS 732621.6600  
 2 AM 628500.4700  
 7 GO 591983.4600  
 11 MT 560511.1900  
 16 PR 530525.5800  
 5 CE 524050.7000  
 8 MA 504192.2600  
 21 RS 489142.8400  
 0 AC NaN  
 1 AL NaN  
 3 AP NaN  
 4 BA NaN  
 6 ES NaN  
 12 PA NaN  
 13 PB NaN  
 15 PI NaN  
 18 RN NaN  
 19 RO NaN  
 20 RR NaN  
 22 SC NaN  
 23 SE NaN  
 Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)  
 24 55612.3997  
 17 49094.4000  
 14 30427.6900  
 25 32452.5600  
 9 39790.9094  
 10 33243.6300  
 2 41345.1100  
 7 33826.8400  
 11 42918.3100  
 16 45318.4600  
 5 24253.9300  
 8 29823.9500  
 21 51116.7200  
 0 NaN  
 1 NaN  
 3 NaN  
 4 NaN  
 6 NaN  
 12 NaN  
 13 NaN  
 15 NaN  
 18 NaN  
 19 NaN  
 20 NaN  
 22 NaN  
 23 NaN  
 Média de PIB, a preços correntes (R\$ 1,00) por UF - df\_eleitos\_vereadores\_outliers:  
 UF VR\_DESPESA\_MAX\_CAMPANHA \  
 24 SP 2539233.0182  
 21 RS 489142.8400  
 17 RJ 1594577.3800  
 16 PR 530525.5800  
 11 MT 560511.1900  
 2 AM 628500.4700  
 9 MG 863766.9312  
 7 GO 591983.4600  
 10 MS 732621.6600  
 25 TO 962451.1500  
 14 PE 1011149.6500

8	MA	504192.2600
5	CE	524050.7000
0	AC	NaN
1	AL	NaN
3	AP	NaN
4	BA	NaN
6	ES	NaN
12	PA	NaN
13	PB	NaN
15	PI	NaN
18	RN	NaN
19	RO	NaN
20	RR	NaN
22	SC	NaN
23	SE	NaN

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

24		55612.3997
21		51116.7200
17		49094.4000
16		45318.4600
11		42918.3100
2		41345.1100
9		39790.9094
7		33826.8400
10		33243.6300
25		32452.5600
14		30427.6900
8		29823.9500
5		24253.9300
0		NaN
1		NaN
3		NaN
4		NaN
6		NaN
12		NaN
13		NaN
15		NaN
18		NaN
19		NaN
20		NaN
22		NaN
23		NaN

Média de VR\_DESPESA\_MAX\_CAMPANHA por UF - df\_eleitos\_prefeitos\_outliers:

UF	VR_DESPESA_MAX_CAMPANHA	
4	BA	10706462.7400
9	MG	8100372.7283
10	MS	7609782.1700
16	PR	7502411.2000
25	TO	5762464.2900
5	CE	5654269.7400
17	RJ	5029159.4340
24	SP	4866950.5882
18	RN	4809624.3550
22	SC	4133220.9800
2	AM	4090982.3600
13	PB	3981974.7900
12	PA	3712311.1100
7	GO	3352725.8350
11	MT	3138404.5525
14	PE	2997337.5550
6	ES	2942611.2700
21	RS	2665432.6700
1	AL	2633359.6700
0	AC	NaN
3	AP	NaN
8	MA	NaN
15	PI	NaN
19	RO	NaN
20	RR	NaN
23	SE	NaN

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

4		52431.8650
9		57662.6250
10		33243.6300
16		56050.9300
25		30752.5800
5		24253.9300
17		51814.1100
24		98346.0709
18		24682.1050
22		41885.5300
2		41345.1100
13		24481.8100
12		177992.2100
7		34938.3900
11		53707.2325

14 25504.9700  
 6 69628.4000  
 21 51116.7200  
 1 16062.6000  
 0 NaN  
 3 NaN  
 8 NaN  
 15 NaN  
 19 NaN  
 20 NaN  
 23 NaN

Média de PIB per capita, a preços correntes (R\$ 1,00) por UF - df\_eleitos\_prefeitos\_outliers:

UF	VR_DESPESA_MAX_CAMPANHA
12 PA	3712311.1100
24 SP	4866950.5882
6 ES	2942611.2700
9 MG	8100372.7283
16 PR	7502411.2000
11 MT	3138404.5525
4 BA	10706462.7400
17 RJ	5029159.4340
21 RS	2665432.6700
22 SC	4133220.9800
2 AM	4090982.3600
7 GO	3352725.8350
10 MS	7609782.1700
25 TO	5762464.2900
14 PE	2997337.5550
18 RN	4809624.3550
13 PB	3981974.7900
5 CE	5654269.7400
1 AL	2633359.6700
0 AC	NaN
3 AP	NaN
8 MA	NaN
15 PI	NaN
19 RO	NaN
20 RR	NaN
23 SE	NaN

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

UF	VR_DESPESA_MAX_CAMPANHA
12	177992.2100
24	98346.0709
6	69628.4000
9	57662.6250
16	56050.9300
11	53707.2325
4	52431.8650
17	51814.1100
21	51116.7200
22	41885.5300
2	41345.1100
7	34938.3900
10	33243.6300
25	30752.5800
14	25504.9700
18	24682.1050
13	24481.8100
5	24253.9300
1	16062.6000
0	NaN
3	NaN
8	NaN
15	NaN
19	NaN
20	NaN
23	NaN

Média de VR\_DESPESA\_MAX\_CAMPANHA por UF - df\_eleitos\_vereadores\_outliers:

UF	VR_DESPESA_MAX_CAMPANHA	POPULAÇÃO ESTIMADA
24 SP	2539233.0182	8148537.9101
17 RJ	1594577.3800	6747815.0000
14 PE	1011149.6500	1653461.0000
25 TO	962451.1500	306296.0000
9 MG	863766.9312	1440019.2022
10 MS	732621.6600	906092.0000
2 AM	628500.4700	2219580.0000
7 GO	591983.4600	1536097.0000
11 MT	560511.1900	618124.0000
16 PR	530525.5800	1948626.0000
5 CE	524050.7000	2686612.0000
8 MA	504192.2600	1108975.0000
21 RS	489142.8400	1488252.0000
0 AC	NaN	NaN
1 AL	NaN	NaN
3 AP	NaN	NaN
4 BA	NaN	NaN
6 ES	NaN	NaN
12 PA	NaN	NaN

13	PB		NaN	NaN
15	PI		NaN	NaN
18	RN		NaN	NaN
19	RO		NaN	NaN
20	RR		NaN	NaN
22	SC		NaN	NaN
23	SE		NaN	NaN
Média de POPULAÇÃO ESTIMADA por UF - df_eleitos_vereadores_outliers:				
	UF	VR_DESPESA_MAX_CAMPANHA	POPULAÇÃO ESTIMADA	
24	SP	2539233.0182	8148537.9101	
17	RJ	1594577.3800	6747815.0000	
5	CE	524050.7000	2686612.0000	
2	AM	628500.4700	2219580.0000	
16	PR	530525.5800	1948626.0000	
14	PE	1011149.6500	1653461.0000	
7	GO	591983.4600	1536097.0000	
21	RS	489142.8400	1488252.0000	
9	MG	863766.9312	1440019.2022	
8	MA	504192.2600	1108975.0000	
10	MS	732621.6600	906092.0000	
11	MT	560511.1900	618124.0000	
25	TO	962451.1500	306296.0000	
0	AC		NaN	NaN
1	AL		NaN	NaN
3	AP		NaN	NaN
4	BA		NaN	NaN
6	ES		NaN	NaN
12	PA		NaN	NaN
13	PB		NaN	NaN
15	PI		NaN	NaN
18	RN		NaN	NaN
19	RO		NaN	NaN
20	RR		NaN	NaN
22	SC		NaN	NaN
23	SE		NaN	NaN
Média de VR_DESPESA_MAX_CAMPANHA por UF - df_eleitos_prefeitos_outliers:				
	UF	VR_DESPESA_MAX_CAMPANHA	POPULAÇÃO ESTIMADA	
4	BA	10706462.7400	1595500.0000	
9	MG	8100372.7283	807339.6667	
10	MS	7609782.1700	906092.0000	
16	PR	7502411.2000	1138842.0000	
25	TO	5762464.2900	196920.5000	
5	CE	5654269.7400	2686612.0000	
17	RJ	5029159.4340	1854511.8000	
24	SP	4866950.5882	1504096.2727	
18	RN	4809624.3550	595549.0000	
22	SC	4133220.9800	508826.0000	
2	AM	4090982.3600	2219580.0000	
13	PB	3981974.7900	411807.0000	
12	PA	3712311.1100	213576.0000	
7	GO	3352725.8350	587998.0000	
11	MT	3138404.5525	302328.0000	
14	PE	2997337.5550	1009369.5000	
6	ES	2942611.2700	365855.0000	
21	RS	2665432.6700	1488252.0000	
1	AL	2633359.6700	61797.0000	
0	AC		NaN	NaN
3	AP		NaN	NaN
8	MA		NaN	NaN
15	PI		NaN	NaN
19	RO		NaN	NaN
20	RR		NaN	NaN
23	SE		NaN	NaN
Média de POPULAÇÃO ESTIMADA por UF - df_eleitos_prefeitos_outliers:				
	UF	VR_DESPESA_MAX_CAMPANHA	POPULAÇÃO ESTIMADA	
5	CE	5654269.7400	2686612.0000	
2	AM	4090982.3600	2219580.0000	
17	RJ	5029159.4340	1854511.8000	
4	BA	10706462.7400	1595500.0000	
24	SP	4866950.5882	1504096.2727	
21	RS	2665432.6700	1488252.0000	
16	PR	7502411.2000	1138842.0000	
14	PE	2997337.5550	1009369.5000	
10	MS	7609782.1700	906092.0000	
9	MG	8100372.7283	807339.6667	
18	RN	4809624.3550	595549.0000	
7	GO	3352725.8350	587998.0000	
22	SC	4133220.9800	508826.0000	
13	PB	3981974.7900	411807.0000	
6	ES	2942611.2700	365855.0000	
11	MT	3138404.5525	302328.0000	
12	PA	3712311.1100	213576.0000	
25	TO	5762464.2900	196920.5000	
1	AL	2633359.6700	61797.0000	
0	AC		NaN	NaN
3	AP		NaN	NaN
8	MA		NaN	NaN
15	PI		NaN	NaN

19 RO NaN NaN  
 20 RR NaN NaN  
 23 SE NaN NaN  
 Média de VR\_DESPESA\_MAX\_CAMPANHA por UF - df\_eleitos\_prefeitos\_outliers  
 UF VR\_DESPESA\_MAX\_CAMPANHA  
 4 BA 10706462.7400  
 9 MG 8100372.7283  
 10 MS 7609782.1700  
 16 PR 7502411.2000  
 25 TO 5762464.2900  
 5 CE 5654269.7400  
 17 RJ 5029159.4340  
 24 SP 4866950.5882  
 18 RN 4809624.3550  
 22 SC 4133220.9800  
 2 AM 4090982.3600  
 13 PB 3981974.7900  
 12 PA 3712311.1100  
 7 GO 3352725.8350  
 11 MT 3138404.5525  
 14 PE 2997337.5550  
 6 ES 2942611.2700  
 21 RS 2665432.6700  
 1 AL 2633359.6700  
 0 AC NaN  
 3 AP NaN  
 8 MA NaN  
 15 PI NaN  
 19 RO NaN  
 20 RR NaN  
 23 SE NaN  
 Média de VR\_DESPESA\_MAX\_CAMPANHA por Região - df\_eleitos\_prefeitos\_outliers  
 Nome\_da\_Grande\_Região VR\_DESPESA\_MAX\_CAMPANHA  
 3 Sudeste 5662047.8817  
 1 Nordeste 5477383.7222  
 4 Sul 5450869.0125  
 2 Norte 4832055.5125  
 0 Centro-oeste 3730478.1911  
 Média de VR\_DESPESA\_MAX\_CAMPANHA por Partido - df\_eleitos\_vereadores\_outliers  
 SG\_PARTIDO VR\_DESPESA\_MAX\_CAMPANHA  
 21 PSOL 1680258.6046  
 19 PSDB 1525902.3440  
 22 PT 1374953.5950  
 3 DEM 1309631.8142  
 5 NOVO 1306085.7520  
 27 REPUBLICANOS 1178139.3078  
 6 PATRIOTA 1152481.9665  
 4 MDB 1118067.3504  
 9 PL 1115752.0687  
 12 PODE 1092339.4696  
 25 PV 1083512.4800  
 16 PSB 1026680.2196  
 28 SOLIDARIEDADE 1019938.3267  
 17 PSC 997016.0967  
 18 PSD 978363.1697  
 24 PTC 932220.5560  
 23 PTB 926829.5027  
 20 PSL 888863.4668  
 0 AVANTE 870925.7047  
 13 PP 827229.6597  
 26 REDE 820239.6960  
 1 CIDADANIA 756007.3700  
 11 PMN 710604.8237  
 2 DC 710497.2589  
 14 PROS 683039.3065  
 8 PDT 672515.1719  
 15 PRTB 658246.3050  
 7 PC do B 627317.1656  
 10 PMB 552518.7800  
 Média de VR\_DESPESA\_MAX\_CAMPANHA por Partido - df\_eleitos\_prefeitos\_outliers  
 SG\_PARTIDO VR\_DESPESA\_MAX\_CAMPANHA  
 18 PSD 11789925.0075  
 3 DEM 9099695.2700  
 19 PSDB 5999664.4120  
 8 PDT 5909592.7850  
 13 PP 4236243.8350  
 0 AVANTE 4090982.3600  
 9 PL 3365884.6100  
 4 MDB 3266154.9210  
 28 SOLIDARIEDADE 3144589.0900  
 1 CIDADANIA 3118648.2260  
 16 PSB 3010863.1700  
 27 REPUBLICANOS 2942611.2700  
 12 PODE 2816243.1250  
 14 PROS 2678796.4400  
 22 PT 2508735.1200  
 2 DC NaN  
 5 NOVO NaN

6 PATRIOTA NaN  
 7 PC do B NaN  
 10 PMB NaN  
 11 PMN NaN  
 15 PRTB NaN  
 17 PSC NaN  
 20 PSL NaN  
 21 PSOL NaN  
 23 PTB NaN  
 24 PTC NaN  
 25 PV NaN  
 26 REDE NaN

Percentual Situação Reeleição - df\_eleitos\_dados\_demograficos\_outliers  
 ST\_REELEICAO NOME DO MUNICÍPIO Código do Município UF \

	N	316	316	316
0				
1	S	88	88	88

Nome\_da\_Grande\_Região POPULAÇÃO ESTIMADA capital DS\_CARGO \

	316	316	316	316
0				
1		88	88	88

VR\_DESPESA\_MAX\_CAMPANHA SG\_PARTIDO NM\_PARTIDO TP\_AGREMIACAO \

	316	316	316	316
0				
1		88	88	88

Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000) \

	316	316
0		
1		88

Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000) \

	316	316
0		
1		88

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \

	316	316
0		
1		88

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

	316	316
0		
1		88

Valor adicionado bruto total, a preços correntes (R\$ 1.000) \

	316	316
0		
1		88

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \

	316	316
0		
1		88

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

	316	316
0		
1		88

Percentual Situação Reeleição - df\_eleitos\_prefeitos\_outliers  
 ST\_REELEICAO NOME DO MUNICÍPIO Código do Município UF \

	N	37	37	37
0				
1	S	12	12	12

Nome\_da\_Grande\_Região POPULAÇÃO ESTIMADA capital DS\_CARGO \

	37	37	37	37
0				
1		12	12	12

VR\_DESPESA\_MAX\_CAMPANHA SG\_PARTIDO NM\_PARTIDO TP\_AGREMIACAO \

	37	37	37	37
0				
1		12	12	12

Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000) \

	37	37
0		
1		12

Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000) \

	37	37
0		
1		12

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \

	37	37
0		
1		12

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

	37	37
0		
1		12

Valor adicionado bruto total, a preços correntes (R\$ 1.000) \

	37	37
0		
1		12

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \

0		37
1		12

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

0		37
1		12

Percentual Situação Reeleição - df\_eleitos\_vereadores\_outliers

ST_REELEICAO	NOME DO MUNICÍPIO	Código do Município	UF
0	N	449	449 449
1	S	116	116 116

Nome\_da\_Grande\_Região POPULAÇÃO ESTIMADA capital DS\_CARGO \

0	449	449	449	449
1	116	116	116	116

VR\_DESPESA\_MAX\_CAMPANHA SG\_PARTIDO NM\_PARTIDO TP\_AGREMIACAO \

0	449	449	449	449
1	116	116	116	116

Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000) \

0		449
1		116

Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000) \

0		449
1		116

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \

0		449
1		116

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

0		449
1		116

Valor adicionado bruto total, a preços correntes (R\$ 1.000) \

0		449
1		116

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000) \

0		449
1		116

Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)

0		449
1		116

Total de eleitos por CARGO - df\_eleitos\_dados\_demograficos\_outliers:

DS_CARGO	NOME DO MUNICÍPIO	Código do Município	UF
0 PREFEITO	219	219	219
1 VEREADOR	185	185	185

Nome\_da\_Grande\_Região POPULAÇÃO ESTIMADA capital \

0	219	219	219
1	185	185	185

VR\_DESPESA\_MAX\_CAMPANHA SG\_PARTIDO NM\_PARTIDO TP\_AGREMIACAO \

0	219	219	219	219
1	185	185	185	185

ST\_REELEICAO \

0	219
1	185

Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000) \

0		219
1		185

Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000) \

0		219
1		185

Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000) \

0		219
1		185

Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000) \

0		219
1		185

Valor adicionado bruto total, a preços correntes (R\$ 1.000) \

0		219
1		185

```

Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
0                               219
1                               185

Produto Interno Bruto per capita, a preços correntes (R$ 1,00)
0                               219
1                               185
Total de eleitos por AGREMIACAO - df_eleitos_dados_demograficos_outliers:
  TP_AGREMIACAO  NOME DO MUNICÍPIO  Código do Município  UF \
0      COLIGAÇÃO          213              213   213
1  PARTIDO ISOLADO         191              191   191

  Nome_da_Grande_Região  POPULAÇÃO ESTIMADA  capital  DS_CARGO \
0                  213              213       213     213
1                  191              191       191     191

  VR_DESPESA_MAX_CAMPANHA  SG_PARTIDO  NM_PARTIDO  ST_REELEICAO \
0                  213              213       213     213
1                  191              191       191     191

  Valor adicionado bruto da Agropecuária, a preços correntes (R$ 1.000) \
0                               213
1                               191

  Valor adicionado bruto da Indústria, a preços correntes (R$ 1.000) \
0                               213
1                               191

  Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde
públicas e seguridade social (R$ 1.000) \
0                               213
1                               191

  Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços
correntes (R$ 1.000) \
0                               213
1                               191

  Valor adicionado bruto total, a preços correntes (R$ 1.000) \
0                               213
1                               191

  Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R$ 1.000) \
0                               213
1                               191
*****
FIM Análise Outliers
*****
***** FIM Análise e Exploração dos Dados - Criação de Modelos de Machine Learning *****
***** Process finished with exit code 0

```