PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Wender Pereira Corrêa

CLASSIFICAÇÃO DE POLITICOS CASSAÇÃO

Belo Horizonte

2022

Wender Pereira Corrêa

CLASSIFICAÇÃO DE POLITICOS CASSAÇÃO

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

2022

SUMÁRIO

1. Introdução

1.1. Contextualização

Eleições no Brasil : uma história de 500 anos - CDD 324.981, “Uma tradição portuguesa... Os colonizadores portugueses mal pisavam o território americano, logo realizavam votações para eleger os que iriam governar as vilas e cidades que fundavam, obedecendo à tradição portuguesa de escolher os administradores de seus povoados. Vários cargos eram preenchidos nestes pleitos, dentre eles: vereador, juiz ordinário, procurador e outros oficiais. A primeira eleição de que se tem notícia definiu os membros do Conselho Municipal da Vila de São Vicente1 – atual São Paulo – em 1532 e ocorreu conforme as determinações das Ordenações do Reino(A). Quem podia votar? Só os homens bons tinham o direito de poder escolher os administradores das vilas. Na época do Brasil Colônia, eram homens bons os nobres de linhagem, os senhores de engenho, e os membros da alta burocracia militar, a esses se acrescentando os homens novos, burgueses enriquecidos pelo comércio..”

Uma candidatura a um cargo político é um processo complexo que envolve diversas áreas da vida de um candidato. Como podemos ver em um recorte do site do TSE realizado em 17/04/2022 o TSE realizou a desaprovação das contas do partido PSL, uma vez que a conta do partido foi desaprovada podemos nos perguntar qual o impacto do partido na vida política de um candidato. Pensando neste assunto foi realizada a análise dos dados das eleições a partir de 2014 para analisar os dados do processo eleitoral do inicio ao fim com base nos dados públicos de duas bases de dados disponíveis no site do TSE;

Em 2016 houve eleições ordinárias. Após a data desta eleição ordinária e antes da próxima, houve eleições suplementares em 2017, 2018 e 2019, sabemos que as eleições ordinárias são previstas em Lei, possuem data certa para serem realizadas, ocorrem em anos pares e possuem a periodicidade de 04 em 04 anos. Nas eleições ordinárias nacionais são eleitos os cargos de Presidente, Governadores, Deputados (Federais e Estaduais) e Senadores.

As eleições ordinárias são previstas em Lei, possuem data certa para serem realizadas, ocorrem em anos pares e possuem a periodicidade de 04 em 04 anos. Nas eleições ordinárias nacionais são eleitos os cargos de Presidente, Governadores, Deputados

(Federais e Estaduais) e Senadores.

https://www.tse.jus.br/legislacao/sne/sistematizacao-das-normas-eleitorais

1.2. O problema proposto

Com o objetivo de identificar quais os principais problemas encontrados pelos candidatos junto ao TSE e a necessidade de estruturar as ações de capacitação voltadas para o alcance deste objetivo foi elaborado o estudo e criados modelos para classificação dos dados das eleições que ocorreram no Brasil no período de 2014 à 2020.

O impacto na sociedade da ausência de um estudo que de classificação com base em dados implica na indefinição de processos coerentes, ocasionando falhas no processo de candidatura e, conseqüentemente, em eventuais lacunas de candidatos que representam a parcela da população.

Considerando a disponibilidade de dados confiáveis do TSE optou-se por desenvolver um estudo de base histórica sobre as eleições.

Para o tratamento do problema proposto, foram utilizados datasets relativos às eleições de 2014 a 2020 os dados foram extraídos em 01/04/2022 as 14:00 horas do repositório de dados eleitorais do Tribunal Superior Eleitoral.

2. Coleta de Dados

Uma das primeiras preocupações após a coleta dos dados foi verificar a consistência e a fidedignidade dos mesmos.

URL Base	Arquivos	Tipo
https://cdn.tse.jus.br/estatistica/sead/odsele/	consulta_cand_2014_AC.csv consulta_cand_2014_AL.csv consulta_cand_2014_AM.csv consulta_cand_2014_AP.csv consulta_cand_2014_BA.csv consulta_cand_2014_BR.csv consulta_cand_2014_BRASIL.csv consulta_cand_2014_CE.csv consulta_cand_2014_DF.csv consulta_cand_2014_ES.csv consulta_cand_2014_GO.csv consulta_cand_2014_MA.csv consulta_cand_2014_MG.csv consulta_cand_2014_MS.csv consulta_cand_2014_MT.csv consulta_cand_2014_PA.csv consulta_cand_2014_PB.csv consulta_cand_2014_PE.csv consulta_cand_2014_PI.csv consulta_cand_2014_PR.csv consulta_cand_2014_RJ.csv consulta_cand_2014_RN.csv consulta_cand_2014_RO.csv consulta_cand_2014_RR.csv consulta_cand_2014_RS.csv consulta_cand_2014_SC.csv consulta_cand_2014_SE.csv consulta_cand_2014_SP.csv consulta_cand_2014_TO.csv consulta_cand_2016_AC.csv consulta_cand_2016_AL.csv consulta_cand_2016_AM.csv consulta_cand_2016_AP.csv consulta_cand_2016_BA.csv consulta_cand_2016_BRASIL.csv consulta_cand_2016_CE.csv consulta_cand_2016_ES.csv consulta_cand_2016_GO.csv consulta_cand_2016_MA.csv consulta_cand_2016_MG.csv consulta_cand_2016_MS.csv consulta_cand_2016_MT.csv consulta_cand_2016_PA.csv	A codificação de caracteres do arquivo é “Latin 1”; Os campos estão entre aspas e separados por ponto e vírgula, inclusive os campos numéricos.

	consulta_cand_2016_PB.csv consulta_cand_2016_PE.csv consulta_cand_2016_PI.csv consulta_cand_2016_PR.csv consulta_cand_2016_RJ.csv consulta_cand_2016_RN.csv consulta_cand_2016_RO.csv consulta_cand_2016_RR.csv consulta_cand_2016_RS.csv consulta_cand_2016_SC.csv consulta_cand_2016_SE.csv consulta_cand_2016_SP.csv consulta_cand_2016_TO.csv consulta_cand_2018_AC.csv consulta_cand_2018_AL.csv consulta_cand_2018_AM.csv consulta_cand_2018_AP.csv consulta_cand_2018_BA.csv consulta_cand_2018_BR.csv consulta_cand_2018_BRASIL.csv consulta_cand_2018_CE.csv consulta_cand_2018_DF.csv consulta_cand_2018_ES.csv consulta_cand_2018_GO.csv consulta_cand_2018_MA.csv consulta_cand_2018_MG.csv consulta_cand_2018_MS.csv consulta_cand_2018_MT.csv consulta_cand_2018_PA.csv consulta_cand_2018_PB.csv consulta_cand_2018_PE.csv consulta_cand_2018_PI.csv consulta_cand_2018_PR.csv consulta_cand_2018_RJ.csv consulta_cand_2018_RN.csv consulta_cand_2018_RO.csv consulta_cand_2018_RR.csv consulta_cand_2018_RS.csv consulta_cand_2018_SC.csv consulta_cand_2018_SE.csv consulta_cand_2018_SP.csv consulta_cand_2018_TO.csv consulta_cand_2020_AC.csv consulta_cand_2020_AL.csv consulta_cand_2020_AM.csv consulta_cand_2020_AP.csv consulta_cand_2020_BA.csv consulta_cand_2020_BRASIL.csv consulta_cand_2020_CE.csv consulta_cand_2020_ES.csv consulta_cand_2020_GO.csv	

	consulta_cand_2020_MA.csv consulta_cand_2020_MG.csv consulta_cand_2020_MS.csv consulta_cand_2020_MT.csv consulta_cand_2020_PA.csv consulta_cand_2020_PB.csv consulta_cand_2020_PE.csv consulta_cand_2020_PI.csv consulta_cand_2020_PR.csv consulta_cand_2020_RJ.csv consulta_cand_2020_RN.csv consulta_cand_2020_RO.csv consulta_cand_2020_RR.csv consulta_cand_2020_RS.csv consulta_cand_2020_SC.csv consulta_cand_2020_SE.csv consulta_cand_2020_SP.csv consulta_cand_2020_TO.csv	
https://cdn.tse.jus.br/estatistica/sead/odsele/motivo_cassacao/	motivo_cassacao_2014_AC.csv motivo_cassacao_2014_AL.csv motivo_cassacao_2014_AM.csv motivo_cassacao_2014_AP.csv motivo_cassacao_2014_BA.csv motivo_cassacao_2014_BR.csv motivo_cassacao_2014_BRASIL.csv motivo_cassacao_2014_CE.csv motivo_cassacao_2014_DF.csv motivo_cassacao_2014_ES.csv motivo_cassacao_2014_GO.csv motivo_cassacao_2014_MA.csv motivo_cassacao_2014_MG.csv motivo_cassacao_2014_MS.csv motivo_cassacao_2014_MT.csv motivo_cassacao_2014_PA.csv motivo_cassacao_2014_PB.csv motivo_cassacao_2014_PE.csv motivo_cassacao_2014_PI.csv motivo_cassacao_2014_PR.csv motivo_cassacao_2014_RJ.csv motivo_cassacao_2014_RN.csv motivo_cassacao_2014_RO.csv motivo_cassacao_2014_RR.csv motivo_cassacao_2014_RS.csv motivo_cassacao_2014_SC.csv motivo_cassacao_2014_SE.csv motivo_cassacao_2014_SP.csv motivo_cassacao_2014_TO.csv motivo_cassacao_2016_AC.csv motivo_cassacao_2016_AL.csv motivo_cassacao_2016_AM.csv motivo_cassacao_2016_AP.csv	A codificação de caracteres do arquivo é “Latin 1”; Os campos estão entre aspas e separados por ponto e vírgula, inclusive os campos numéricos.

	motivo_cassacao_2016_BA.csv motivo_cassacao_2016_BRASIL.csv motivo_cassacao_2016_CE.csv motivo_cassacao_2016_ES.csv motivo_cassacao_2016_GO.csv motivo_cassacao_2016_MA.csv motivo_cassacao_2016_MG.csv motivo_cassacao_2016_MS.csv motivo_cassacao_2016_MT.csv motivo_cassacao_2016_PA.csv motivo_cassacao_2016_PB.csv motivo_cassacao_2016_PE.csv motivo_cassacao_2016_PI.csv motivo_cassacao_2016_PR.csv motivo_cassacao_2016_RJ.csv motivo_cassacao_2016_RN.csv motivo_cassacao_2016_RO.csv motivo_cassacao_2016_RR.csv motivo_cassacao_2016_RS.csv motivo_cassacao_2016_SC.csv motivo_cassacao_2016_SE.csv motivo_cassacao_2016_SP.csv motivo_cassacao_2016_TO.csv motivo_cassacao_2018_AC.csv motivo_cassacao_2018_AL.csv motivo_cassacao_2018_AM.csv motivo_cassacao_2018_AP.csv motivo_cassacao_2018_BA.csv motivo_cassacao_2018_BR.csv motivo_cassacao_2018_BRASIL.csv motivo_cassacao_2018_CE.csv motivo_cassacao_2018_DF.csv motivo_cassacao_2018_ES.csv motivo_cassacao_2018_GO.csv motivo_cassacao_2018_MA.csv motivo_cassacao_2018_MG.csv motivo_cassacao_2018_MS.csv motivo_cassacao_2018_MT.csv motivo_cassacao_2018_PA.csv motivo_cassacao_2018_PB.csv motivo_cassacao_2018_PE.csv motivo_cassacao_2018_PI.csv motivo_cassacao_2018_PR.csv motivo_cassacao_2018_RJ.csv motivo_cassacao_2018_RN.csv motivo_cassacao_2018_RO.csv motivo_cassacao_2018_RR.csv motivo_cassacao_2018_RS.csv motivo_cassacao_2018_SC.csv motivo_cassacao_2018_SE.csv motivo_cassacao_2018_SP.csv	

	motivo_cassacao_2018_TO.csv motivo_cassacao_2020_AC.csv motivo_cassacao_2020_AL.csv motivo_cassacao_2020_AM.csv motivo_cassacao_2020_AP.csv motivo_cassacao_2020_BA.csv motivo_cassacao_2020_BRASIL.csv motivo_cassacao_2020_CE.csv motivo_cassacao_2020_ES.csv motivo_cassacao_2020_GO.csv motivo_cassacao_2020_MA.csv motivo_cassacao_2020_MG.csv motivo_cassacao_2020_MS.csv motivo_cassacao_2020_MT.csv motivo_cassacao_2020_PA.csv motivo_cassacao_2020_PB.csv motivo_cassacao_2020_PE.csv motivo_cassacao_2020_PI.csv motivo_cassacao_2020_PR.csv motivo_cassacao_2020_RJ.csv motivo_cassacao_2020_RN.csv motivo_cassacao_2020_RO.csv motivo_cassacao_2020_RR.csv motivo_cassacao_2020_RS.csv motivo_cassacao_2020_SC.csv motivo_cassacao_2020_SE.csv motivo_cassacao_2020_SP.csv motivo_cassacao_2020_TO.csv	

O dataSet contém o leiaute existente das tabelas existentes no repositório de dados eleitorais: 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 #NULO significam que a informação está em branco no banco de dados. O correspondente para #NULO 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. - Os arquivos estão em constante processo de atualização e aperfeiçoamento. Alguns arquivos podem estar em branco ou com mensagem de erro devido a indisponibilidade temporária na base de algum estado ou à inexistência daquele arquivo para a época pretendida.

3. Processamento/Tratamento de Dados

O processamento e o tratamento dos dados foram feitos utilizando a linguagem Python, versão Python 3.6.13 |Anaconda, Inc.| (default, Mar 16 2021, 11:37:27) [MSC v.1916 64 bit (AMD64)] no ambiente do Jupyter Notebook, versão 6.4.3 .

As operações de ETL (extraction, transformation, loading) dos dados, atividades de pré-processamento, representam a etapa mais demorada e trabalhosa de um projeto de data science, consumindo pelo menos 70% do tempo total do projeto.

Importação das dependências do projeto

#Leitura dos dados e importação das bibliotecas utilizadas

In [1]:
```
#IMPORTAÇÃO DAS BIBLIOTECAS
import pandas as pd
import numpy as np
# dependências para importar e descompactar arquivos em zip
import zipfile
import requests
from io import BytesIO
import os
import glob
import seaborn as sns
sns.set_theme(style="whitegrid", palette="muted")
import time
#Importação da função Counter e da biblioteca matplotlib.pyplot
from collections import Counter
import matplotlib.pyplot as plt
#Plotação de um gráfico para mostrar a dispersão dos dados.
#O % e inline significa que o gráfico será mostrado aqui no notebook, e não em um arquivo
%matplotlib inline
```

Supressão dos warnings e definição das variáveis para armazenamento do diretório dos arquivos.

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
#Definindo a pasta de trabalho
dirCandidatoAnalise         = "./candidatoAnalise"
dirCandidatoCassacaoAnalise = "./candidatoCassacaoAnalise"

os.makedirs(dirCandidatoAnalise, exist_ok=True)
os.makedirs(dirCandidatoCassacaoAnalise, exist_ok=True)
```

Definição de função para descompactar os arquivos.

In [4]:
```
def descompactar(nome_arquivo_compactado, diretorio):
    #Lê o arquivo compactado e extrai o conteúdo
    #print(nome_arquivo_compactado)
    filebytes = BytesIO(
        requests.get(nome_arquivo_compactado).content
    )
    myzip = zipfile.ZipFile(filebytes)
    myzip.extractall(diretorio)
    return "Concluído"
```

Estrutura de repetição responsável por acessar, baixar e descompactar os dados do TSE;

1) Dados Politicos

In [5]:
```
ca = 2014
while ca < 2022:
    urlCandidatoAnalise = "https://cdn.tse.jus.br/estatistica/sead/odsele/consulta_cand/consulta_cand_" + str(ca)+ '
    descompactar(urlCandidatoAnalise,dirCandidatoAnalise)
    urlCandidatoCassacaoAnalise = "https://cdn.tse.jus.br/estatistica/sead/odsele/motivo_cassacao/motivo_cassacao_"
    descompactar(urlCandidatoCassacaoAnalise,dirCandidatoCassacaoAnalise)
    ca+=2
```

Foi realizado a importação dos dados relativos ao processo abaixo através de uma estrutura de repetição responsável por acessar o diretório onde contem os arquivos csv, realizar a abertura dos arquivos selecionando o enconding “Latin 1” , separado por “;” agrupando os dados através do comando append. Após o agrupamentos foi elimidado todas a linhas duplicadas através do comando “drop_duplicates()”.

Neste dataframe não apresentam nulos que necessitem de tratamento.

1) Dados Candidato

In [6]:
```
print('Arquivos com extensão csv Candidato:')
concatenar = []
with os.scandir(dirCandidatoAnalise) as arqs:
    for arq in arqs:
        if arq.is_file() and  arq.name.endswith('.csv'):
            print(arq.name)
    path = dirCandidatoAnalise
    # csv files in the path
    files = glob.glob(path + "/*.csv")
    # defining an empty list to store
    # content
    data_frame = pd.DataFrame()
    content = []
    # checking all the csv files in the
    # specified path
    for filename in files:
        df = pd.read_csv(filename, encoding = "Latin 1", sep= ";", decimal = ',',
            error_bad_lines=False,dtype={"SQ_CANDIDATO": int, "ANO_ELEICAO": int,"SG_UF" : "string", "NR_TURNO": int,"HH_G
        df.drop_duplicates()
        content.append(df)
    # converting content to data frame
    data_frame = pd.concat(content).drop_duplicates()
```

```
consulta_cand_2016_AP.csv
consulta_cand_2016_BA.csv
consulta_cand_2016_BRASIL.csv
consulta_cand_2016_CE.csv
consulta_cand_2016_ES.csv
consulta_cand_2016_GO.csv
consulta_cand_2016_MA.csv
consulta_cand_2016_MG.csv
consulta_cand_2016_MS.csv
consulta_cand_2016_MT.csv
consulta_cand_2016_PA.csv
```

Este processo foi responsável por acessar e baixar 112 arquivos csv, e criar o dataframe que contem 1.112.090 entradas, divididas em 63 colunas.

In [122]:
```
data_frame
```

Out[122]:

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	NR_TURNO	CD_ELEICAO	DS_ELEICAO	DT_ELEICAO	TP_ABRANG
0	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FE
1	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FE
2	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FE
3	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FE
4	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FE
...	...	...	...	...	...	...	...	...	...	
558293	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUN
558294	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUN
558296	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUN
558297	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUN
558298	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUN

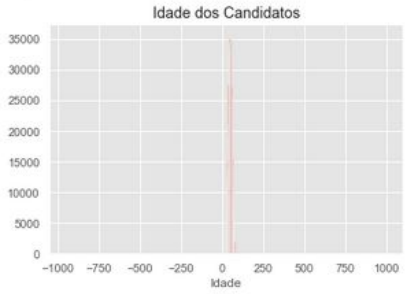
1112090 rows × 63 columns

```
In [7]: 1 data_frame.dtypes

Out[7]: DT_GERACAO                      object
        HH_GERACAO                      object
        ANO_ELEICAO                      int32
        CD_TIPO_ELEICAO                  int32
        NM_TIPO_ELEICAO                 string
                                         ...
        CD_SITUACAO_CANDIDATO_PLEITO    string
        DS_SITUACAO_CANDIDATO_PLEITO    string
        CD_SITUACAO_CANDIDATO_URNA      string
        DS_SITUACAO_CANDIDATO_URNA      string
        ST_CANDIDATO_INSERIDO_URNA      string
        Length: 63, dtype: object
```

Foram realizados a plotagem de diversos dados para identificar a correlação e posteriormente realizar o tratamento das distorções.

```
In [8]: 1 idade_publico = Counter(data_frame['NR_IDADE_DATA_POSSE'])
        2 #Plotagem da idade dos candidatos
        3 plt.style.use('ggplot')
        4 plt.bar(idade_publico.keys(), idade_publico.values())
        5 plt.xlabel('Idade')
        6 plt.title('Idade dos Candidatos')
        7 plt.show()
```



In [9]:
```
categorical_columns_df = [cname for cname in data_frame.columns if data_frame[cname].dtype == "object"]
categorical_columns_df
```

Out[9]:
```
['DT_GERACAO', 'HH_GERACAO', 'DT_ELEICAO', 'DS_COR_RACA']
```

In [9]:
```
#Calcula o total e a % de valores ausentes
num_ausentes = data_frame.isna().sum()
porc_ausentes = data_frame.isna().sum() * 100 / len(data_frame)
df_ausentes = pd.DataFrame({
    'Coluna': data_frame.columns,
    'Dados ausentes': num_ausentes,
    'Porcentagem': porc_ausentes
})
df_ausentes
```

Out[9]:

	Coluna	Dados ausentes	Porcentagem
DT_GERACAO	DT_GERACAO	0	0.0
HH_GERACAO	HH_GERACAO	0	0.0
ANO_ELEICAO	ANO_ELEICAO	0	0.0
CD_TIPO_ELEICAO	CD_TIPO_ELEICAO	0	0.0
NM_TIPO_ELEICAO	NM_TIPO_ELEICAO	0	0.0
...	...	...	...
CD_SITUACAO_CANDIDATO_PLEITO	CD_SITUACAO_CANDIDATO_PLEITO	0	0.0
DS_SITUACAO_CANDIDATO_PLEITO	DS_SITUACAO_CANDIDATO_PLEITO	0	0.0
CD_SITUACAO_CANDIDATO_URNA	CD_SITUACAO_CANDIDATO_URNA	0	0.0
DS_SITUACAO_CANDIDATO_URNA	DS_SITUACAO_CANDIDATO_URNA	0	0.0
ST_CANDIDATO_INSERIDO_URNA	ST_CANDIDATO_INSERIDO_URNA	0	0.0

63 rows × 3 columns

```
In [10]:
display(data_frame.head())
# Imprimindo as últimas linhas
display(data_frame.tail())
# Informações do nosso DataFrame
data_frame.describe()
# Dimensões do df_dados
data_frame.shape
```

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	NR_TURNO	CD_ELEICAO	DS_ELEICAO	DT_ELEICAO	TP_ABRANGENCIA
0	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FEDERAL
1	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FEDERAL
2	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FEDERAL
3	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FEDERAL
4	08/04/2021	11:38:25	2014	2	ELEIÇÃO ORDINÁRIA	1	143	Eleições Gerais 2014	05/10/2014	FEDERAL

5 rows × 63 columns

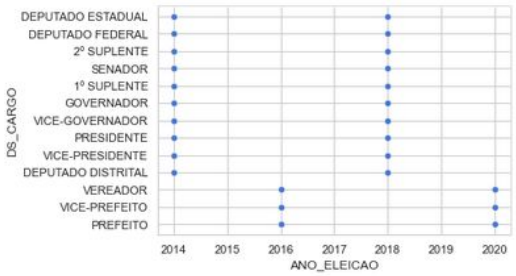
558294	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUNICI
558296	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUNICI
558297	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUNICI
558298	17/04/2022	07:19:43	2020	2	ELEIÇÃO ORDINÁRIA	1	426	Eleições Municipais 2020	15/11/2020	MUNICI

5 rows × 63 columns

```
Out[10]: (1112090, 63)
```

Para melhor compreensão dos registros obtidos inicialmente foram elaboradas representações gráficas sobre os principais conjuntos de dados.

```
In [11]:
res = sns.scatterplot(x=data_frame['ANO_ELEICAO'],y=data_frame['DS_CARGO'])
fig = plt.figure(figsize=(16,8))
plt.show()
```




```
<Figure size 1152x576 with 0 Axes>
```

```
In [12]:
data_frame.values
```

```
Out[12]: array([['08/04/2021', '11:38:25', 2014, ..., '2', 'DEFERIDO', 'SIM'],
                ['08/04/2021', '11:38:25', 2014, ..., '2', 'DEFERIDO', 'SIM'],
                ['08/04/2021', '11:38:25', 2014, ..., '2', 'DEFERIDO', 'SIM'],
                ...,
                ['17/04/2022', '07:19:43', 2020, ..., '2', 'DEFERIDO', 'SIM'],
                ['17/04/2022', '07:19:43', 2020, ..., '2', 'DEFERIDO', 'SIM'],
                ['17/04/2022', '07:19:43', 2020, ..., '2', 'DEFERIDO', 'SIM']],
               dtype=object)
```

Podemos identificar a dispersão dos dados no decorrer dos anos pode perceber a dispersão dos cargos conforme o tipo de eleição.

```
In [13]: 1 data_frame.columns

Out[13]: Index(['DT_GERACAO', 'HH_GERACAO', 'ANO_ELEICAO', 'CD_TIPO_ELEICAO',
       'NM_TIPO_ELEICAO', 'NR_TURNO', 'CD_ELEICAO', 'DS_ELEICAO', 'DT_ELEICAO',
       'TP_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', '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_OCUPACAO',
       'DS_OCUPACAO', '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'],
      dtype='object')

In [14]: 1 data_frame.index

Out[14]: Int64Index([     0,      1,      2,      3,      4,      5,      6,      7,
                  8,      9,
             ...
             558288, 558289, 558290, 558291, 558292, 558293, 558294, 558296,
             558297, 558298],
            dtype='int64', length=1112090)
```

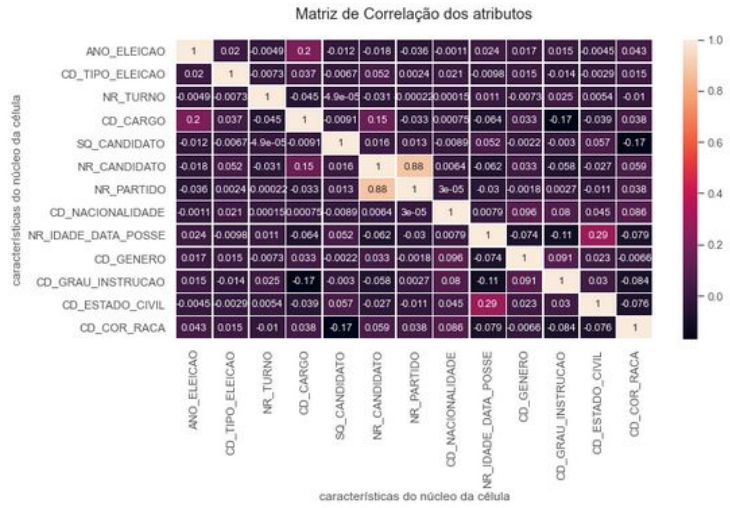
Identificação da matriz de correlação quanto maior a magnitude, mais forte a correlação. Sinal: se positivo, há uma correlação regular. Se negativo, há uma correlação inversa.

Percebeu-se dois valores em destaque nr_partido, nr_candidato com valorer = 0.88

Matriz de correlação de atributos

```
In [16]:
# O corr() método do Pandas DataFrame  é usado para calcular a matriz.
# Por padrão, ele calcula o coeficiente de correlação de Pearson
correlation_mat = data_frame.corr(method='pearson')
# Definindo as configurações do Gráfico
fig = plt.figure(figsize=(8,4))
eixo = fig.add_axes([0, 0, 1, 1])
# Usando o método heatmap para traçar a Matriz
# O parâmetro ' annot=True' exibe os valores do coeficiente de correlação em cada célula.
sns.heatmap(correlation_mat, annot = True, linewidth=0.5)
# Definindo titulo e labels do gráfico
eixo.set_title('Matriz de Correlação dos atributos', fontsize=15, pad=20)
eixo.set_xlabel ("características do núcleo da célula")
eixo.set_ylabel ("características do núcleo da célula")
```

```
Out[16]: Text(-23.500000000000007, 0.5, 'características do núcleo da célula')
```



2 - Dados Cassação

```
In [16]:
print('Arquivos com extensão csv Cassação:')
concatenar = []
with os.scandir(dirCandidatoCassacaoAnalise) as arqs:
    for arq in arqs:
        if arq.is_file() and  arq.name.endswith('.csv'):
            print(arq.name)
    path = dirCandidatoCassacaoAnalise
    # csv files in the path
    files = glob.glob(path + "/*.csv")
    # defining an empty list to store
    # content
    data_frame_cassacao = pd.DataFrame()
    content_situacao = []
    # checking all the csv files in the
    # specified path
    for filename in files:
        dfs = pd.read_csv(filename, encoding = "Latin 1", sep= ";", decimal = ',', error_bad_lines=False, dtype={"SQ_CANDI
        content_situacao.append(dfs)
    # converting content to data frame
    data_frame_cassacao = pd.concat(content_situacao).drop_duplicates()
```

```
motivo_cassacao_2014_BA.csv
motivo_cassacao_2014_BR.csv
motivo_cassacao_2014_BRASIL.csv
motivo_cassacao_2014_CE.csv
motivo_cassacao_2014_DF.csv
motivo_cassacao_2014_ES.csv
motivo_cassacao_2014_GO.csv
motivo_cassacao_2014_MA.csv
motivo_cassacao_2014_MG.csv
motivo_cassacao_2014_MS.csv
motivo_cassacao_2014_MT.csv
motivo_cassacao_2014_PA.csv
motivo_cassacao_2014_PB.csv
motivo_cassacao_2014_PE.csv
motivo_cassacao_2014_PI.csv
motivo_cassacao_2014_PR.csv
motivo_cassacao_2014_RJ.csv
motivo_cassacao_2014_RN.csv
motivo_cassacao_2014_RO.csv
motivo_cassacao_2014_RR.csv
```

Para a remoção, foi utilizado o método “Pandas.DataFrame.dropna”.

```
In [121]: 1 data_frame_cassacao
```

Out[121]:

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	CD_ELEICAO	DS_ELEICAO	SG_UF	SG_UE	NM_UE	SQ_CAND
0	05/04/2021	18:39:43	2014	1	Eleição Suplementar	268	Eleição Suplementar Governador AM	AM	AM	AMAZONAS	1345
1	05/04/2021	18:39:43	2014	1	Eleição Suplementar	268	Eleição Suplementar Governador AM	AM	AM	AMAZONAS	1345
1	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-582
2	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-582
3	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-582
...	...	...	...	...	...	...	...	...	...	...	
18161	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	SP	71218	SÃO VICENTE	892
18162	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	SP	62910	CAMPINAS	892
18164	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	MA	8273	MATA ROMA	1216
18165	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	MA	8273	MATA ROMA	1216
18166	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	MA	8079	ITAPECURU MIRIM	1216

39946 rows × 12 columns

Com o resultado apresentado na imagem acima, foi possível perceber que existem 39.946 linhas e 12 colunas no “DataFrame”

In [17]:
```
#Calcula o total e a % de valores ausentes
num_ausentes = data_frame_cassacao.isna().sum()
porc_ausentes = data_frame_cassacao.isna().sum() * 100 / len(data_frame_cassacao)
df_ausentes = pd.DataFrame({
    'Coluna': data_frame_cassacao.columns,
    'Dados ausentes': num_ausentes,
    'Porcentagem': porc_ausentes
})
df_ausentes
```

Out[17]:

	Coluna	Dados ausentes	Porcentagem
DT_GERACAO	DT_GERACAO	0	0.0
HH_GERACAO	HH_GERACAO	0	0.0
ANO_ELEICAO	ANO_ELEICAO	0	0.0
CD_TIPO_ELEICAO	CD_TIPO_ELEICAO	0	0.0
NM_TIPO_ELEICAO	NM_TIPO_ELEICAO	0	0.0
CD_ELEICAO	CD_ELEICAO	0	0.0
DS_ELEICAO	DS_ELEICAO	0	0.0
SG_UF	SG_UF	0	0.0
SG_UE	SG_UE	0	0.0
NM_UE	NM_UE	0	0.0
SQ_CANDIDATO	SQ_CANDIDATO	0	0.0
DS_MOTIVO_CASSACAO	DS_MOTIVO_CASSACAO	0	0.0

In [18]:
```
display(data_frame_cassacao.head())
# Imprimindo as últimas linhas
display(data_frame_cassacao.tail())
# Informações do nosso DataFrame
data_frame_cassacao.describe()
# Dimensões do df_dados
data_frame_cassacao.shape
```

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	CD_ELEICAO	DS_ELEICAO	SG_UF	SG_UE	NM_UE	SQ_CANDIDA
0	05/04/2021	18:39:43	2014	1	Eleição Suplementar	268	Eleição Suplementar Governador AM	AM	AM	AMAZONAS	13453064
1	05/04/2021	18:39:43	2014	1	Eleição Suplementar	268	Eleição Suplementar Governador AM	AM	AM	AMAZONAS	13453064
1	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-5829292
2	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-5829292
3	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-5829292

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	CD_ELEICAO	DS_ELEICAO	SG_UF	SG_UE	NM_UE	SQ_CAND
18161	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	SP	71218	SÃO VICENTE	892
18162	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais	SP	62910	CAMPINAS	892

```
In [18]:
display(data_frame_cassacao.head())
# Imprimindo os últimas linhas
display(data_frame_cassacao.tail())
# Informações do nosso DataFrame
data_frame_cassacao.describe()
# Dimensões do df_dados
data_frame_cassacao.shape
```

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	CD_ELEICAO	DS_ELEICAO	SG_UF	SG_UE	NM_UE	SQ_CANDIDAT
0	05/04/2021	18:39:43	2014	1	Eleição Suplementar	268	Eleição Suplementar Governador AM	AM	AM	AMAZONAS	134530640
1	05/04/2021	18:39:43	2014	1	Eleição Suplementar	268	Eleição Suplementar Governador AM	AM	AM	AMAZONAS	134530640
1	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-58292920
2	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-58292920
3	05/04/2021	18:39:43	2014	1	Eleição Suplementar	304	Eleição Suplementar Governador - TO	TO	TO	TOCANTINS	-58292920

	DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	CD_ELEICAO	DS_ELEICAO	SG_UF	SG_UE	NM_UE	SQ_CANDI
18161	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	SP	71218	SÃO VICENTE	8927
18162	17/04/2022	07:22:08	2020	2	Eleição Ordinária	426	Eleições Municipais 2020	SP	62910	CAMPINAS	8929

```
In [20]:
data_frame_cassacao.columns
```

```
Out[20]: Index(['DT_GERACAO', 'HH_GERACAO', 'ANO_ELEICAO', 'CD_TIPO_ELEICAO',
       'NM_TIPO_ELEICAO', 'CD_ELEICAO', 'DS_ELEICAO', 'SG_UF', 'SG_UE',
       'NM_UE', 'SQ_CANDIDATO', 'DS_MOTIVO_CASSACAO'],
      dtype='object')
```

Análise dos dados.

ANÁLISE DADOS

Agrupamento de dados selecionando dados de interesse: Dados do candidato

```
In [25]:
#removendo dados com todas linhas faltando dados
data_frame.dropna(how='all', inplace=True)
#descobrindo a idade média
mediaIdade = round(data_frame['NR_IDADE_DATA_POSSE'].mean(),0)
print(mediaIdade)
# Preenchendo a coluna com o valor da média:
data_frame.update(data_frame['NR_IDADE_DATA_POSSE'].fillna(mediaIdade))
# Dimensões do df_dados
data_frame.shape
```

```
45.0
Out[25]: (1112090, 63)
```

```
In [26]:
data_frame['NR_IDADE_DATA_POSSE'].describe().astype('int')
```

```
Out[26]: count    1112090
mean           45
std            11
min          -951
25%            37
50%            45
75%            53
max           999
Name: NR_IDADE_DATA_POSSE, dtype: int32
```

O próximo passo consistiu no tratamento do “DataFrame” para adequação ao projeto. O processo de tratamento seguiu os passos seguintes.

1) Substituição da idade superior a 104 anos pela média da idade do dataset

2) Substituição da idade inferior a 18 anos pela idade mínima para concorrer as eleições.

	SQ_CANDIDATO	ANO_ELEICAO	SG_UF	NR_CPF_CANDIDATO	NR_CANDIDATO	NR_IDADE_DATA_POSSE	DT_NASCIMENTO	NM_CANDIDATO	CD_SITUACA
4575	1345297941	2016	AM	41737288249	12345	944.0	09/04/1072	NAILSON ALVES DE CAMPOS	
5876	1345301017	2016	AM	64157520220	45888	999.0	31/01/0976	MARIA IZABEL PINTO BARBOSA	
6537	1345303268	2016	AM	57808791253	45000	999.0	27/09/0072	LUCAS DOS SANTOS CORREA	
7555	1345301791	2016	AM	79590586287	40333	999.0	30/03/0970	ERISVALDO FERREIRA TAVARES	
72216	1151043192	2016	MG	78684242653	22456	999.0	21/08/0065	REINALDO DUARTE FERNANDES	
168817	-1798690850	2016	PE	99019272420	21021	999.0	28/09/0076	TATIANO PATRICIO DA COSTA CUNHA	
303451	-1928231893	2016	RR	78961890204	55111	999.0	01/04/0082	VANDERSON ANTONIO PORTO CAMPOS	
331410	-453390931	2016	RS	14008459049	15	967.0	27/05/1049	ALAOR PASTORIZA RIBEIRO	
23879	-1733317123	2018	PA	01752740297	10700	825.0	26/12/1193	MAURO CEZAR MELO RIBEIRO	
749	-1538546057	2020	BA	62841106500	19123	999.0	30/01/0972	SIMONE MARIA DOS SANTOS FARIAS	
27656	-1538394234	2020	BA	31698391870	40444	120.0	11/10/1900	ELIZENIA SOUZA OLIVEIRA	
227679	-257983001	2020	MS	36538515134	55123	116.0	22/05/1904	VALDECY LOPES DA SILVA	

Out[28]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	SG_UF	NR_CANDIDATO	NR_IDADE_DATA_POSSE	DT_NASCIMENTO	NM_CANDIDATO	CD_SITUACA
30275	-129533892	2016	57352305300	CE	51200	12.0	13/12/2004	FRANCISCO FABIO GUEDES UCHÔA	
40053	-259077883	2016	48902659100	MS	55222	-951.0	12/12/2968	JAIME CARDOSO DA CRUZ	
6898	1346185022	2020	06290001230	AM	27206	17.0	24/02/2003	FRANCINEIA OLIVEIRA DE SOUZA	
46527	-1797491124	2020	07050764400	PE	36111	16.0	03/03/2004	LUIZ ALMEIDA	
176897	-1732911331	2020	08905421202	PA	10700	17.0	30/06/2003	DEYSE ADRIELE DA SILVA SARMANHO	

Foi identificado que a idade média na data da pose é de 45 anos.

Agrupamento de dados selecionando dados de interesse: Dados do candidato

In [24]:
```
#removendo dados com todas linhas faltando dados
data_frame.dropna(how='all', inplace=True)
#descobrindo a idade média
mediaIdade = round(data_frame['NR_IDADE_DATA_POSSE'].mean(),0)
print(mediaIdade)
# Preenchendo a coluna com o valor da média:
data_frame.update(data_frame['NR_IDADE_DATA_POSSE'].fillna(mediaIdade))
# Dimensões do df_dados
data_frame.shape
```

```
45.0
```

Out[24]: (1112090, 63)

In [25]:
```
data_frame['NR_IDADE_DATA_POSSE'].describe().astype('int')
```

Out[25]:
```
count    1112090
mean          45
std           11
min         -951
25%           37
50%           45
75%           53
max          999
Name: NR_IDADE_DATA_POSSE, dtype: int32
```

Criação de dois datasets para agrupar os dados das distorções para posterior tratamento do dataset pincipal.

```
idade_max = data_frame[data_frame['NR_IDADE_DATA_POSSE'] > 104]
idade_min = data_frame[data_frame['NR_IDADE_DATA_POSSE'] < 18]
```

```
#Seleção das colunas de interesse
df_ida_max = idade_max[['SQ_CANDIDATO','ANO_ELEICAO','SG_UF','NR_CPF_CANDIDATO','NR_CANDIDATO','NR_IDADE_DATA_POSSE','DT_NA
df_ida_max
```

Dataset com dados maiores que 104 anos “Idade_max”

In [27]:
```
#Seleção das colunas de interesse
df_ida_max = idade_max[['SQ_CANDIDATO','ANO_ELEICAO','SG_UF','NR_CPF_CANDIDATO','NR_CANDIDATO','NR_IDADE_DATA_POSSE','DT_NASCIM
df_ida_max
```

Out[27]:

	SQ_CANDIDATO	ANO_ELEICAO	SG_UF	NR_CPF_CANDIDATO	NR_CANDIDATO	NR_IDADE_DATA_POSSE	DT_NASCIMENTO	NM_CANDIDATO	CD_SITUACA
4575	1345297941	2016	AM	41737288249	12345	944.0	09/04/1072	NAILSON ALVES DE CAMPOS	
5876	1345301017	2016	AM	64157520220	45888	999.0	31/01/0976	MARIA IZABEL PINTO BARBOSA	
6537	1345303268	2016	AM	57808791253	45000	999.0	27/09/0072	LUCAS DOS SANTOS CORREA	
7555	1345301791	2016	AM	79590586287	40333	999.0	30/03/0970	ERISVALDO FERREIRA TAVARES	
72216	1151043192	2016	MG	78684242653	22456	999.0	21/08/0065	REINALDO DUARTE FERNANDES	
168817	-1798690850	2016	PE	99019272420	21021	999.0	28/09/0076	TATIANO PATRICIO DA COSTA CUNHA	
303451	-1928231893	2016	RR	78961890204	55111	999.0	01/04/0082	VANDERSON ANTONIO PORTO CAMPOS	
331410	-453390931	2016	RS	14008459049	15	967.0	27/05/1049	ALAOR PASTORIZA RIBEIRO	
23879	-1733317123	2018	PA	01752740297	10700	825.0	26/12/1193	MAURO CEZAR MELO RIBEIRO	
749	-1538546057	2020	BA	62841106500	19123	999.0	30/01/0972	SIMONE MARIA DOS SANTOS FARIAS	
27656	-1538394234	2020	BA	31698391870	40444	120.0	11/10/1900	ELIZENIA SOUZA OLIVEIRA	
227679	-257983001	2020	MS	36538515134	55123	116.0	22/05/1904	VALDECY LOPES DA SILVA	

Estrutura de repetição para tratamento das distorções de idade optou-se por utilizar a idade média de 45 anos.

In [29]:
```
print('--------------------------------------')
for index, row in df_ida_max.iterrows():
    data_frame.at[index , 'NR_IDADE_DATA_POSSE' ] = mediaIdade
    print('Id: ' + str(index))
    print('Ano: ' + str(row['ANO_ELEICAO']))
    print('Candidato: ' + str(row['SQ_CANDIDATO']))
    print('CPF: ' + str(row['NR_CPF_CANDIDATO']))
    print('Nome: ' + str(row['NM_CANDIDATO']))
    print('Data Nascimento: ' + str(row['DT_NASCIMENTO']))
    print('Idade Antiga: ' + str(row['NR_IDADE_DATA_POSSE']))
    print('Idade Aualizada: ' + str(mediaIdade))
    print('--------------------------------------')
```

Detalhamento do tratamento dos dados, Ex: o registro com id 4575 estava reportando uma data de 944 anos e foi realizado o replace para a idade média de 45 anos.

```
--------------------------------------
Id: 4575
Ano: 2016
Candidato: 1345297941
CPF: 41737288249
Nome: NAILSON ALVES DE CAMPOS
Data Nascimento: 09/04/1072
Idade Antiga: 944.0
Idade Aualizada: 45.0
--------------------------------------
Id: 5876
Ano: 2016
Candidato: 1345301017
CPF: 64157520220
Nome: MARIA IZABEL PINTO BARBOSA
Data Nascimento: 31/01/0976
Idade Antiga: 999.0
Idade Aualizada: 45.0
--------------------------------------
Id: 6537
Ano: 2016
Candidato: 1345303268
CPF: 57808791253
Nome: LUCAS DOS SANTOS CORREA
Data Nascimento: 27/09/0072
Idade Antiga: 999.0
Idade Aualizada: 45.0
--------------------------------------
Id: 7555
Ano: 2016
Candidato: 1345301791
CPF: 79590586287
Nome: ERISVALDO FERREIRA TAVARES
Data Nascimento: 30/03/0970
Idade Antiga: 999.0
Idade Aualizada: 45.0
--------------------------------------
Id: 72216
Ano: 2016
Candidato: 1151043192
CPF: 78684242653
Nome: REINALDO DUARTE FERNANDES
Data Nascimento: 21/08/0065
```

Idade Antiga: 999.0
Idade Aualizada: 45.0

Id: 168817
Ano: 2016
Candidato: -1798690850
CPF: 99019272420
Nome: TATIANO PATRICIO DA COSTA CUNHA
Data Nascimento: 28/09/0076
Idade Antiga: 999.0
Idade Aualizada: 45.0

Id: 303451
Ano: 2016
Candidato: -1928231893
CPF: 78961890204
Nome: VANDERSON ANTONIO PORTO CAMPOS
Data Nascimento: 01/04/0082
Idade Antiga: 999.0
Idade Aualizada: 45.0

Id: 331410
Ano: 2016
Candidato: -453390931
CPF: 14008459049
Nome: ALAOR PASTORIZA RIBEIRO
Data Nascimento: 27/05/1049
Idade Antiga: 967.0
Idade Aualizada: 45.0

Id: 23879
Ano: 2018
Candidato: -1733317123
CPF: 01752740297
Nome: MAURO CEZAR MELO RIBEIRO
Data Nascimento: 26/12/1193
Idade Antiga: 825.0
Idade Aualizada: 45.0

Id: 749
Ano: 2020
Candidato: -1538546057
CPF: 62841106500
Nome: SIMONE MARIA DOS SANTOS FARIAS
Data Nascimento: 30/01/0972
Idade Antiga: 999.0
Idade Aualizada: 45.0

Id: 27656
Ano: 2020
Candidato: -1538394234
CPF: 31698391870
Nome: ELIZENIA SOUZA OLIVEIRA
Data Nascimento: 11/10/1900
Idade Antiga: 120.0
Idade Aualizada: 45.0

```
Id: 227679
Ano: 2020
Candidato: -257983001
CPF: 36538515134
Nome: VALDECY LOPES DA SILVA
Data Nascimento: 22/05/1904
Idade Antiga: 116.0
Idade Aualizada: 45.0
---------------------------------------
```

Criação de dataset com idades menores que 18 anos para realizar o tratamento da distorção que retornou uma idade negativa com valode de -951 anos como é possível ver no print abaixo.

Tratamento dos dados da Idade

In [28]:
```
#Seleção das colunas de interesse
df_ida_min = idade_min[['SQ_CANDIDATO','ANO_ELEICAO','NR_CPF_CANDIDATO','SG_UF','NR_CANDIDATO','NR_IDADE_DATA_POSSE','DT_NASCIM
df_ida_min
```

Out[28]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	SG_UF	NR_CANDIDATO	NR_IDADE_DATA_POSSE	DT_NASCIMENTO	NM_CANDIDATO	CD_SITUACA
30275	-129533892	2016	57352305300	CE	51200	12.0	13/12/2004	FRANCISCO FABIO GUEDES UCHÔA	
40053	-259077883	2016	48902659100	MS	55222	-951.0	12/12/2968	JAIME CARDOSO DA CRUZ	
6898	1346185022	2020	06290001230	AM	27206	17.0	24/02/2003	FRANCINEIA OLIVEIRA DE SOUZA	
46527	-1797491124	2020	07050764400	PE	36111	16.0	03/03/2004	LUIZ ALMEIDA	
176897	-1732911331	2020	08905421202	PA	10700	17.0	30/06/2003	DEYSE ADRIELE DA SILVA SARMANHO	

Estrutura de repetição para tratamento das distorções de idade nos casos que a idade era menor que 18 anos optou-se por utilizar a idade mínima para concorrer a eleição.

In [30]:
```
print('---------------------------------------')
for index, row in df_ida_min.iterrows():
    data_frame.at[index , 'NR_IDADE_DATA_POSSE' ] = 18
    print('Id: ' + str(index))
    print('Ano: ' + str(row['ANO_ELEICAO']))
    print('Candidato: ' + str(row['SQ_CANDIDATO']))
    print('CPF: ' + str(row['NR_CPF_CANDIDATO']))
    print('Nome: ' + str(row['NM_CANDIDATO']))
    print('Data Nascimento: ' + str(row['DT_NASCIMENTO']))
    print('Idade Antiga: ' + str(row['NR_IDADE_DATA_POSSE']))
    print('Idade Atualizada: ' + str('18'))
    print('---------------------------------------')
```

Detalhamento dos dados:

```
---------------------------------------
Id: 30275
Ano: 2016
Candidato: -129533892
CPF: 57352305300
Nome: FRANCISCO FABIO GUEDES UCHÔA
Data Nascimento: 13/12/2004
Idade Antiga: 12.0
Idade Atualizada: 18
---------------------------------------
```

Id: 40053
Ano: 2016
Candidato: -259077883
CPF: 48902659100
Nome: JAIME CARDOSO DA CRUZ
Data Nascimento: 12/12/2968
Idade Antiga: -951.0
Idade Atualizada: 18

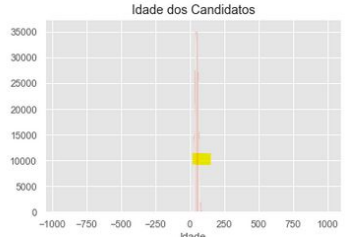
Id: 6898
Ano: 2020
Candidato: 1346185022
CPF: 06290001230
Nome: FRANCINEIA OLIVEIRA DE SOUZA
Data Nascimento: 24/02/2003
Idade Antiga: 17.0
Idade Atualizada: 18

Id: 46527
Ano: 2020
Candidato: -1797491124
CPF: 07050764400
Nome: LUIZ ALMEIDA
Data Nascimento: 03/03/2004
Idade Antiga: 16.0
Idade Atualizada: 18

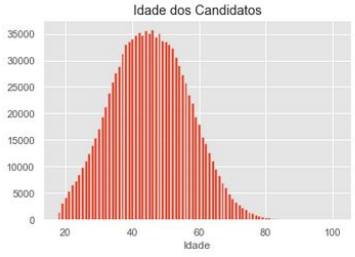
Id: 176897
Ano: 2020
Candidato: -1732911331
CPF: 08905421202
Nome: DEYSE ADRIELE DA SILVA SARMANHO
Data Nascimento: 30/06/2003
Idade Antiga: 17.0
Idade Atualizada: 18

O próximo passo consistiu em avaliar visualmente o dados após o tratamento das distorções da idade.

```
]:
1 idade_publico = Counter(data_frame['NR_IDADE_DATA_POSSE'])
2 #Plotagem da idade dos candidatos
3 plt.style.use('ggplot')
4 plt.bar(idade_publico.keys(), idade_publico.values())
5 plt.xlabel('Idade')
6 plt.title('Idade dos Candidatos')
7 plt.show()
```




```
In [52]:
1 idade_publico = Counter(df_candidato_cassacao['NR_IDADE_DATA_POSSE'])
2 #Plotagem da idade dos candidatos
3 plt.style.use('ggplot')
4 plt.bar(idade_publico.keys(), idade_publico.values())
5 plt.xlabel('Idade')
6 plt.title('Idade dos Candidatos')
7 plt.show()
```



Datasets principal “data_frame” após tratamento das distorções de idade. Como resultado do tratamento da idade não foi localizado nenhuma linha do campo idade na data posse distorcido.

```
In [31]: data_frame[data_frame['NR_IDADE_DATA_POSSE'] > 104]
```

Out[31]:

DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	NR_TURNO	CD_ELEICAO	DS_ELEICAO	DT_ELEICAO	TP_ABRANGENCIA

0 rows × 63 columns

```
In [32]: data_frame[data_frame['NR_IDADE_DATA_POSSE'] < 18]
```

Out[32]:

DT_GERACAO	HH_GERACAO	ANO_ELEICAO	CD_TIPO_ELEICAO	NM_TIPO_ELEICAO	NR_TURNO	CD_ELEICAO	DS_ELEICAO	DT_ELEICAO	TP_ABRANGENCIA

0 rows × 63 columns

```
In [33]: data_frame['NR_IDADE_DATA_POSSE'].describe().astype('int')
```

Out[33]:

```
count    1112090
mean          45
std           11
min           18
25%           37
50%           45
75%           53
max          101
Name: NR_IDADE_DATA_POSSE, dtype: int32
```

Agrupamento de dados de interesse para reunir as informações sobre o motivo de cassação dos candidatos.

Agrupamento de dados selecionando dados de interesse: Dados do Cassação

```
In [34]: #Seleção das colunas de interesse
         df_consulta_candidato = data_frame[['SQ_CANDIDATO','ANO_ELEICAO','NR_CPF_CANDIDATO','NR_CANDIDATO', 'NM_CANDIDATO','CD_CAR(
```

```
profile = pp.ProfileReport(df,n_extreme_obs=5, title='Análise Exploratória',explorative=True)
profile
```

```
In [35]: #Seleção das colunas de interesse
         df_consulta_cassacao = data_frame_cassacao[['SQ_CANDIDATO','ANO_ELEICAO','DS_MOTIVO_CASSACAO']]
         display(df_consulta_cassacao.head())
         # Imprimindo as últimas linhas
         display(df_consulta_cassacao.tail())
         # Informações do nosso DataFrame
         df_consulta_cassacao.describe()
```

	SQ_CANDIDATO	ANO_ELEICAO	DS_MOTIVO_CASSACAO
0	1345306401	2014	Ausência de requisito de registro
1	1345306408	2014	Ficha limpa (LC 64/90)
1	-582929209	2014	Ausência de requisito de registro
2	-582929208	2014	Ausência de requisito de registro
3	-582929203	2014	Ausência de requisito de registro

	SQ_CANDIDATO	ANO_ELEICAO	DS_MOTIVO_CASSACAO
18161	892735756	2020	Ausência de requisito de registro
18162	892917885	2020	Ausência de requisito de registro
18164	1216860705	2020	Indeferimento de partido ou coligação.
18165	1216860709	2020	Indeferimento de partido ou coligação.
18166	1216785583	2020	Ausência de requisito de registro

Para seguir a análise e posterior classificação foi realizar a junção dos dados do candidato com os dados relativos à cassação utilizando o método “pandas.DataFrame.merge”, primeiramente combinando o “DataFrame” “df_consulota_candidato” e o “df_consulta_cassacao” a esqueda (“how=left”), a partira das colunas “sq_candidato,ano_eleicao” presente em ambas as bases.

A segunda etapa consistiu em verificar o resultado da combinação e validar a existência de elementos sem correspondências após a aplicação da função “pandas.DataFrame.head”, dos métodos “pandas.DataFrame.shape”

```
In [36]: m_df_candidato_cassacao = pd.merge(df_consulta_candidato, df_consulta_cassacao, on=['SQ_CANDIDATO','ANO_ELEICAO'], how="le
```

```
In [37]: display(m_df_candidato_cassacao.head())
         # Imprimindo as últimas linhas
         display(m_df_candidato_cassacao.tail())
         # Informações do nosso DataFrame
         m_df_candidato_cassacao.describe()
```

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF	...	DS
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC	...	
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC	...	
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC	...	
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC	...	
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC	...	

5 rows × 28 columns

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_U
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA SILVA	13	VEREADOR	19	PODE	P
1115027	893076343	2020	26951640840	10580	MARISTELA CRISTINA ALBERTO	13	VEREADOR	10	REPUBLICANOS	S
1115028	-128557416	2020	35720638334	55	ANTONIA MARIA DE JESUS ALBUQUERQUE	12	VICE-PREFEITO	55	PSD	C
1115029	-193108551	2020	72054085120	45325	MARIA LÚCIA CUNHA LINHARES	13	VEREADOR	45	PSDB	G
1115030	-193474012	2020	04356899164	12222	RAFAEL SANTOS DANTAS	13	VEREADOR	12	PDT	G

5 rows × 28 columns

4. Análise e Exploração dos Dados

A primeira etapa consistiu em combinar as bases de para gerar a base final que será utilizada nos modelos. Para isso, utilizou-se o método “pandas.DataFrame. merge” Ao plotar o “dataFrame” “m_df_candidato_cassacao” , foi possível perceber que existem 1.115.031 linhas e 28 colunas

```
In [123]: 1 m_df_candidato_cassacao
```

Out[123]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC
1	1410065650	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC
...	...	...	...	...	...	...	...	...	...	...
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA SILVA	13	VEREADOR	19	PODE	PE
1115027	893076343	2020	26951640840	10580	MARISTELA CRISTINA ALBERTO	13	VEREADOR	10	REPUBLICANOS	SP
1115028	-128557416	2020	35720638334	55	ANTONIA MARIA DE JESUS ALBUQUERQUE	12	VICE-PREFEITO	55	PSD	CE
1115029	-193108551	2020	72054085120	45325	MARIA LÚCIA CUNHA LINHARES	13	VEREADOR	45	PSDB	GO
1115030	-193474012	2020	04356899164	12222	RAFAEL SANTOS DANTAS	13	VEREADOR	12	PDT	GO

1115031 rows × 28 columns

In [38]:

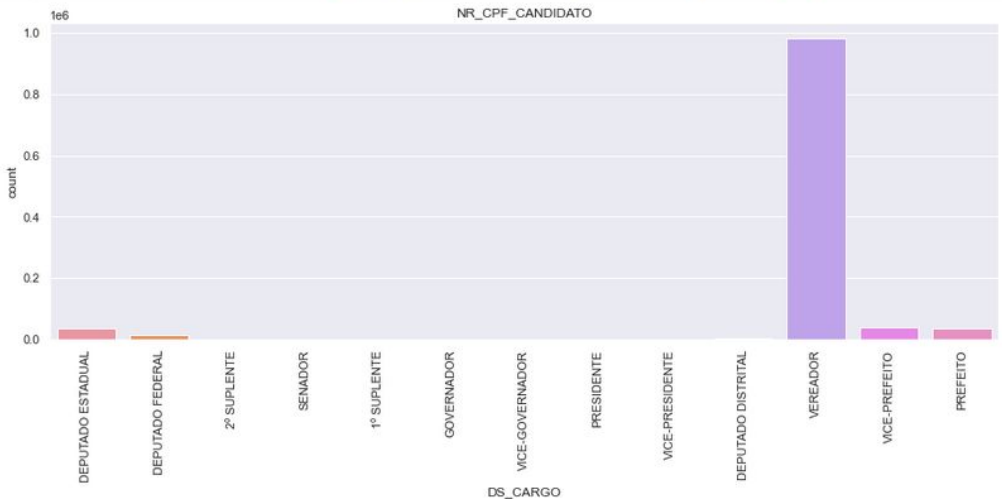
```
#Calcula o total e a % de valores ausentes
num_ausentes = m_df_candidato_cassacao.isna().sum()
porc_ausentes = m_df_candidato_cassacao.isna().sum() * 100 / len(m_df_candidato_cassacao)
df_ausentes = pd.DataFrame({
    'Coluna': m_df_candidato_cassacao.columns,
    'Dados ausentes': num_ausentes,
    'Porcentagem': porc_ausentes
})
df_ausentes
```

	Coluna	Dados ausentes	Porcentagem
SQ_CANDIDATO	SQ_CANDIDATO	0	0.000000
ANO_ELEICAO	ANO_ELEICAO	0	0.000000
NR_CPF_CANDIDATO	NR_CPF_CANDIDATO	0	0.000000
NR_CANDIDATO	NR_CANDIDATO	0	0.000000
NM_CANDIDATO	NM_CANDIDATO	0	0.000000
CD_CARGO	CD_CARGO	0	0.000000
DS_CARGO	DS_CARGO	0	0.000000
NR_PARTIDO	NR_PARTIDO	0	0.000000
SG_PARTIDO	SG_PARTIDO	0	0.000000
SG_UF	SG_UF	0	0.000000
DS_SIT_TOT_TURNO	DS_SIT_TOT_TURNO	0	0.000000
NR_IDADE_DATA_POSSE	NR_IDADE_DATA_POSSE	0	0.000000
CD_GENERO	CD_GENERO	0	0.000000
DS_GENERO	DS_GENERO	0	0.000000
DT_NASCIMENTO	DT_NASCIMENTO	348	0.031210
DS_GRAU_INSTRUCAO	DS_GRAU_INSTRUCAO	0	0.000000
DS_ESTADO_CIVIL	DS_ESTADO_CIVIL	0	0.000000
CD_COR_RACA	CD_COR_RACA	0	0.000000
DS_COR_RACA	DS_COR_RACA	0	0.000000
ST_REELEICAO	ST_REELEICAO	0	0.000000
TP_ABRANGENCIA	TP_ABRANGENCIA	0	0.000000
ST_DECLARAR_BENS	ST_DECLARAR_BENS	0	0.000000
CD_GRAU_INSTRUCAO	CD_GRAU_INSTRUCAO	0	0.000000
DS_SITUACAO_CANDIDATURA	DS_SITUACAO_CANDIDATURA	0	0.000000
CD_SITUACAO_CANDIDATO_URNA	CD_SITUACAO_CANDIDATO_URNA	0	0.000000
ST_CANDIDATO_INSERIDO_URNA	ST_CANDIDATO_INSERIDO_URNA	0	0.000000
VR_DESPESA_MAX_CAMPANHA	VR_DESPESA_MAX_CAMPANHA	0	0.000000
DS_MOTIVO_CASSACAO	DS_MOTIVO_CASSACAO	1075091	96.418037

Aplicando a seqüência de códigos para calcular o total e a % de valores ausentes, identificou que a coluna “ds_motivo_cassacao” possui um total de 1.075.091 representando 96.418037% e a “dt_nascimento” possui 348 linhas ausentes representando 0.031210%.Considerando que a coluna “ds_motivo_cassacao” só vai existir registros se houver alguns registros de cassação ela não será considerada neste momento como umas inconsistências.O campo “dt_nascimento” não será realizado o tratamento uma vez que não possuímos a informação e o custo de obter a data é muito alto na seqüência anterior onde não estava identificada a idade na data da posse foi utilizado duas estratégias diferentes, para as datas superiores a 104 anos utilizou-se a média, e, nos casos de data inferior a 18 anos utilizou a idade mínima de 18 anos.

Verifica-se que o total de cargos no intervalo de 2014 a 2020, como era de se esperar a quantidade de vereadores é muito superior ao restante das vagas com um total de 984.028, já para presidente encontramos um total de 29 registros.

```
In [125]:
sns.set(style="darkgrid")       #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount  = sns.countplot(x="DS_CARGO", data=df_candidato_cassacao).set_title("NR_CPF_CANDIDATO")
```




```
In [41]:
df_candidato_cassacao.groupby('DS_CARGO')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
```

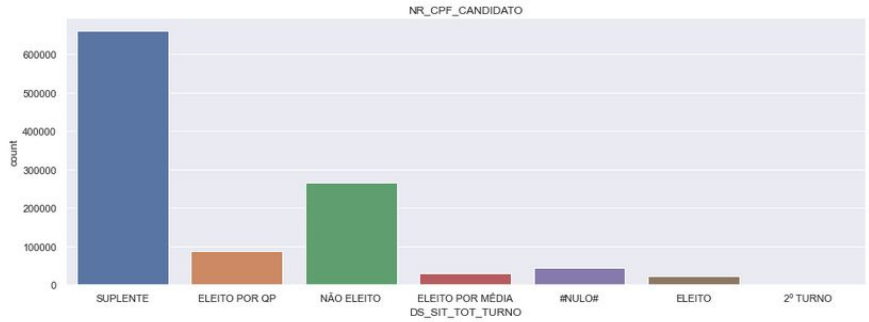
```
Out[41]: DS_CARGO
VEREADOR              984028
VICE-PREFEITO          37945
PREFEITO               37219
DEPUTADO ESTADUAL      35154
DEPUTADO FEDERAL       15855
DEPUTADO DISTRITAL      2037
2º SUPLENTE              626
1º SUPLENTE              613
SENADOR                  562
VICE-GOVERNADOR          474
GOVERNADOR               461
PRESIDENTE                29
VICE-PRESIDENTE           28
Name: NR_CPF_CANDIDATO, dtype: int64
```

Analisando a situação que leva um candidato a ser considerado eleito chegamos ao cenário.

```
In [43]:
df_candidato_cassacao.groupby('DS_SIT_TOT_TURNO')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
```

```
Out[43]: DS_SIT_TOT_TURNO
SUPLENTE            660964
NÃO ELEITO          266648
ELEITO POR QP        88275
#NULO#               44963
ELEITO POR MÉDIA     30758
ELEITO               22847
2º TURNO               576
Name: NR_CPF_CANDIDATO, dtype: int64
```

```
In [127]:
sns.set(style="darkgrid")       #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount  = sns.countplot(x="DS_SIT_TOT_TURNO", data=df_candidato_cassacao).set_title("NR_CPF_CANDIDATO")
```

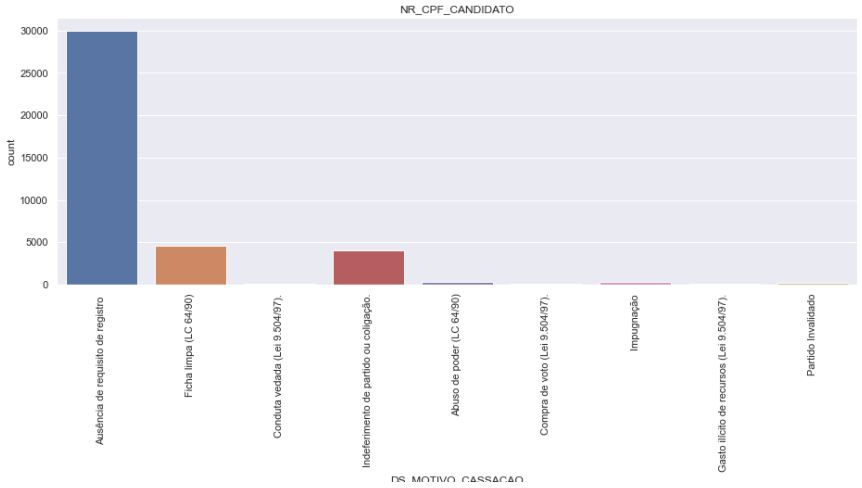


Conforme definição de dados existente no manual do TSE, campos preenchido como “#NULO” significam que a informação está em branco no banco de dados . Ocorrespondente par a # N U L O nos campos numéricos é – 1. Analisando a situações que levam um candidato a ser considerado cassado encontramos o cenário que fica visível que o maior risco de um candidato não participar de uma eleição é “Ausência de requisito de registro” e no segundo maior um fator muito importante e que todo candidato eleito deve ter em mente que ele deve cumprir todos os requisitos para não se enquadrado e ter o nome incrito na Ficha Limpa (LC 64/90) que por sua ver já conta com diversas atualizações, para duvidas aessar o link: http://www.planalto.gov.br/ccivil_03/leis/lcp/lcp64.htm

```
In [44]:
df_candidato_cassacao.groupby('DS_MOTIVO_CASSACAO')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)

Out[44]:
DS_MOTIVO_CASSACAO
Ausência de requisito de registro           29985
Ficha limpa (LC 64/90)                       4639
Indeferimento de partido ou coligação.       4063
Impugnação                                    356
Abuso de poder (LC 64/90)                     356
Partido Invalidado                            223
Conduta vedada (Lei 9.504/97).                122
Compra de voto (Lei 9.504/97).                108
Gasto ilícito de recursos (Lei 9.504/97).      88
Name: NR_CPF_CANDIDATO, dtype: int64
```

```
In [129]:
sns.set(style="darkgrid")       #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount  = sns.countplot(x="DS_MOTIVO_CASSACAO", data=df_candidato_cassacao).set_title("NR_CPF_CANDIDATO")
```



Foi realizado a criação de um dataset para agrupar os dados dos **candidatos eleitos**, considerou que a situação no turno diferente de ('#NULO#','2º TURNO', 'NÃO ELEITO') com eleitos.

```
In [56]:
#Criação de um dataframe final com candidatos eleitos
df_eleitos = df_candidato_cassacao[~df_candidato_cassacao.DS_SIT_TOT_TURNO.isin(['#NULO#','2º TURNO', 'NÃO ELEITO'])]
```

```
In [57]:
df_eleitos
```

Out[57]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC
...	...	...	...	...	...	...	...	...	...	...
1115025	-1992222799	2020	01970617535	23999	ADAILTON SOARES SANTOS	13	VEREADOR	23	CIDADANIA	SE
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA	13	VEREADOR	19	PODE	PE

Foi realizado a criação de um novo dataSet para guardas os dados de todos os candidatos eleitos foram eleitos e posteriormente cassados

```
In [58]:
df_eleitos_cassados = df_eleitos[~df_eleitos.DS_MOTIVO_CASSACAO.isnull()]
```

```
In [59]:
df_eleitos_cassados
```

Out[59]:

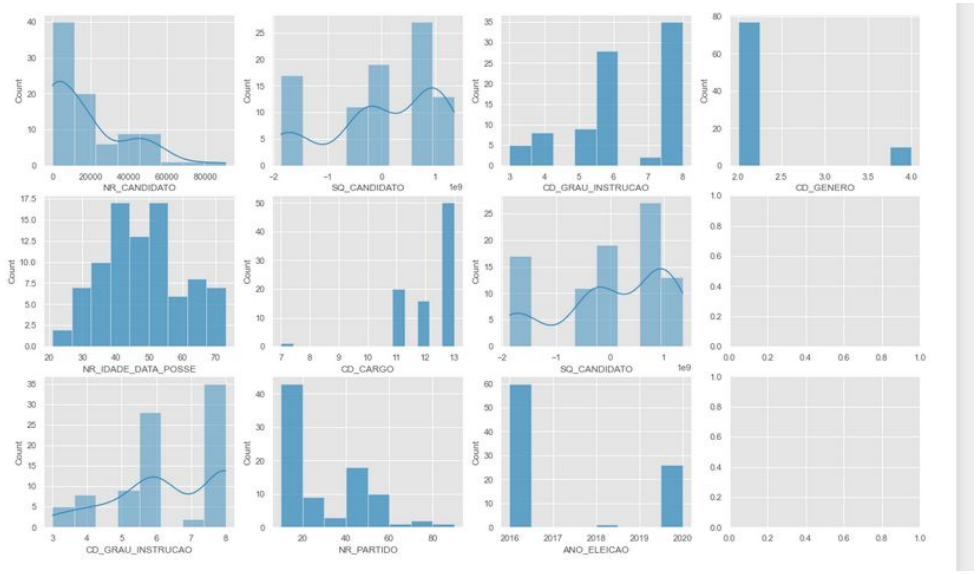
	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF
42114	1345304173	2016	51707152268	19000	ALDIONE SOUZA CORDOVIL	13	VEREADOR	19	PTN	AM
43356	1345303187	2016	14742284291	15	RAIMUNDO PINHEIRO DA SILVA	11	PREFEITO	15	PMDB	AM
43459	1345303188	2016	02705230220	15	ANTÔNIO ARAÚJO COELHO	12	VICE-PREFEITO	77	SD	AM
44630	1345300927	2016	16334060325	90789	JAMY DE CARVALHO CAMPOS	13	VEREADOR	90	PROS	AM
45791	-64768849	2016	46280847691	15007	MARIO RICARDO NASCIMENTO DE OLIVEIRA	13	VEREADOR	15	PMDB	AM
...	...	...	...	...	...	...	...	...	...	...
1066818	893019484	2020	06696211888	40310	ANTONIO CASSEMIRO DA SILVA	13	VEREADOR	40	PSB	SP
1077276	892694587	2020	34590441870	77000	EVERTON APARECIDO DE OLIVEIRA	13	VEREADOR	77	SOLIDARIEDADE	SP

```
In [64]:
cassados = df_eleitos.loc[df_eleitos['DS_MOTIVO_CASSACAO'].notnull()]
#Contagem das opções da coluna DS_GENERO
genero_candidatos_cassados = Counter(cassados['DS_GENERO'])
genero_candidatos_cassados
```

```
Out[64]: Counter({'MASCULINO': 77, 'FEMININO': 10})
```

Análise gráfica

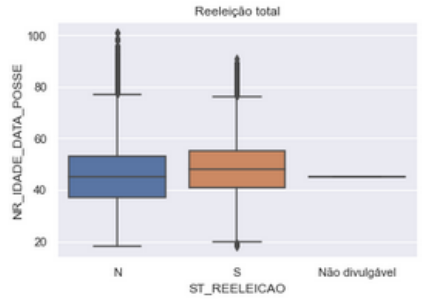
```
In [66]:
# Distributions of the features
fig, ax = plt.subplots(3, 4, figsize=(20, 12))
sns.histplot(df_eleitos_cassados['NR_CANDIDATO'],kde=True, ax=ax[0, 0])
sns.histplot(df_eleitos_cassados['SQ_CANDIDATO'], kde=True, ax=ax[0,1])
sns.histplot(df_eleitos_cassados['CD_GRAU_INSTRUCAO'], ax=ax[0, 2])
sns.histplot(df_eleitos_cassados['CD_GENERO'], ax=ax[0, 3])
sns.histplot(df_eleitos_cassados['NR_IDADE_DATA_POSSE'], ax=ax[1, 0])
sns.histplot(df_eleitos_cassados['CD_CARGO'], ax=ax[1, 1])
sns.histplot(df_eleitos_cassados['SQ_CANDIDATO'], kde=True, ax=ax[1, 2])
sns.histplot(df_eleitos_cassados['CD_GRAU_INSTRUCAO'], kde=True, ax=ax[2, 0])
sns.histplot(df_eleitos_cassados['NR_PARTIDO'], ax=ax[2, 1])
sns.histplot(df_eleitos_cassados['ANO_ELEICAO'], ax=ax[2, 2])
plt.show()
```



In [72]:
```
#Contagem das opções da coluna DS_GENERO
genero_candidatos_eleitos = Counter(df_eleitos['DS_GENERO'])
genero_candidatos_eleitos
```

Out[72]: Counter({'MASCULINO': 540923, 'FEMININO': 261891, 'NÃO DIVULGÁVEL': 30})

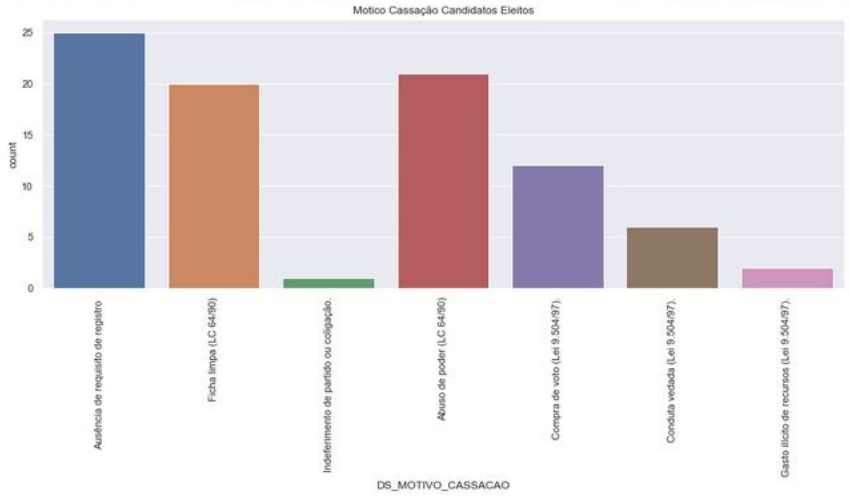
In [73]:
```
sns.boxplot(x='ST_REELEICAO', y='NR_IDADE_DATA_POSSE', data=df_eleitos)
plt.title('Reeleição total')
plt.show()
```



In [68]:
```
df_eleitos_cassados.groupby('DS_MOTIVO_CASSACAO')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
```

Out[68]:
```
DS_MOTIVO_CASSACAO
Ausência de requisito de registro           25
Abuso de poder (LC 64/90)                   21
Ficha limpa (LC 64/90)                      20
Compra de voto (Lei 9.504/97).              12
Conduta vedada (Lei 9.504/97).               6
Gasto ilícito de recursos (Lei 9.504/97).    2
Indeferimento de partido ou coligação.       1
Name: NR_CPF_CANDIDATO, dtype: int64
```

```
In [69]:
1 sns.set(style="darkgrid")       #style the plot background to become a grid
2 fig = plt.figure(figsize=(12,4))
3 eixo = fig.add_axes([0, 0, 1, 1])
4 plt.xticks(rotation=90)
5 genderCount  = sns.countplot(x="DS_MOTIVO_CASSACAO", data=df_eleitos_cassados).set_title("Motico Cassação Candidatos Eleitos")
```



Criou-se um novo dataset para guardar os dados de todos os candidatos que não foram eleitos.

```
In [61]:
1 df_nao_eleitos
```

Out[61]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF
18	1410065775	2014	45689709115	2515	MARIA TERESA GAUNA	6	DEPUTADO FEDERAL	25	DEM	AC
19	1410065778	2014	13813013200	2577	SAULO DE FREITAS RIBEIRO	6	DEPUTADO FEDERAL	25	DEM	AC
25	1410065762	2014	43480063268	333	SANDRA MARIA DA SILVA ROCHA DE AVILA SALDO	10	2º SUPLENTE	43	PV	AC
32	1410065735	2014	65234782204	4321	FRANCISCO ANTONIO LIRA DA SILVA MONTEIRO	6	DEPUTADO FEDERAL	43	PV	AC
51	1410065658	2014	16191463200	54555	CARLOS ROBERTO DE OLIVEIRA	7	DEPUTADO ESTADUAL	54	PPL	AC
...	...	...	...	...	...	...	...	...	...	...
1115011	-581703580	2020	02616167140	28321	ALINE RODRIGUES DA SILVA	13	VEREADOR	28	PRTB	TO
1115012	-1603186288	2020	13709485738	23333	THIARA ALVES MIGUEL	13	VEREADOR	23	CIDADANIA	ES

Semelhando aos candidatos eleitos foi criado um dataset para guardado os candidatos que não foram eleitos e que tiveram sua candidatura cassada.

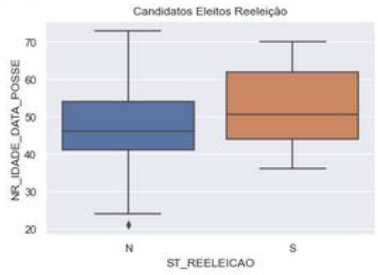
In [62]:
```
df_nao_eleitos_cassados = df_nao_eleitos[~df_nao_eleitos.DS_MOTIVO_CASSACAO.isnull()]
```

In [63]:
```
df_nao_eleitos_cassados
```

Out[63]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_
1163	1345306401	2014	71172092249	23	LILIANE ARAUJO DE ALMEIDA	3	GOVERNADOR	23	PPS	
1498	1345306408	2014	33460825200	11	ABDALA HABIB FRAXE JUNIOR	4	VICE-GOVERNADOR	19	PODE	
8854	-582929203	2014	32303165172	50	MELQUESEDEC MAGALHAES AIRES	4	VICE-GOVERNADOR	50	PSOL	
12396	-582929209	2014	47207868120	50	MARIO LUCIO DE AVELAR	3	GOVERNADOR	50	PSOL	
15527	-582929208	2014	88718298168	50	MAYST MARCOS DE SOUSA SANTOS	4	VICE-GOVERNADOR	50	PSOL	
...	...	...	...	...	...	...	...	...	...	
1114982	-322667151	2020	70308049454	17000	ARTHUR DE SOUSA ESTEVAO	13	VEREADOR	17	PSL	
1114983	1152050704	2020	02691316637	77333	EDINALDO CAETANO CORDEIRO	13	VEREADOR	77	SOLIDARIEDADE	
1115003	-517366380	2020	21822399874	15615	ILOIR ALVES CARNEIRO	13	VEREADOR	15	MDB	
					MARIA DE					

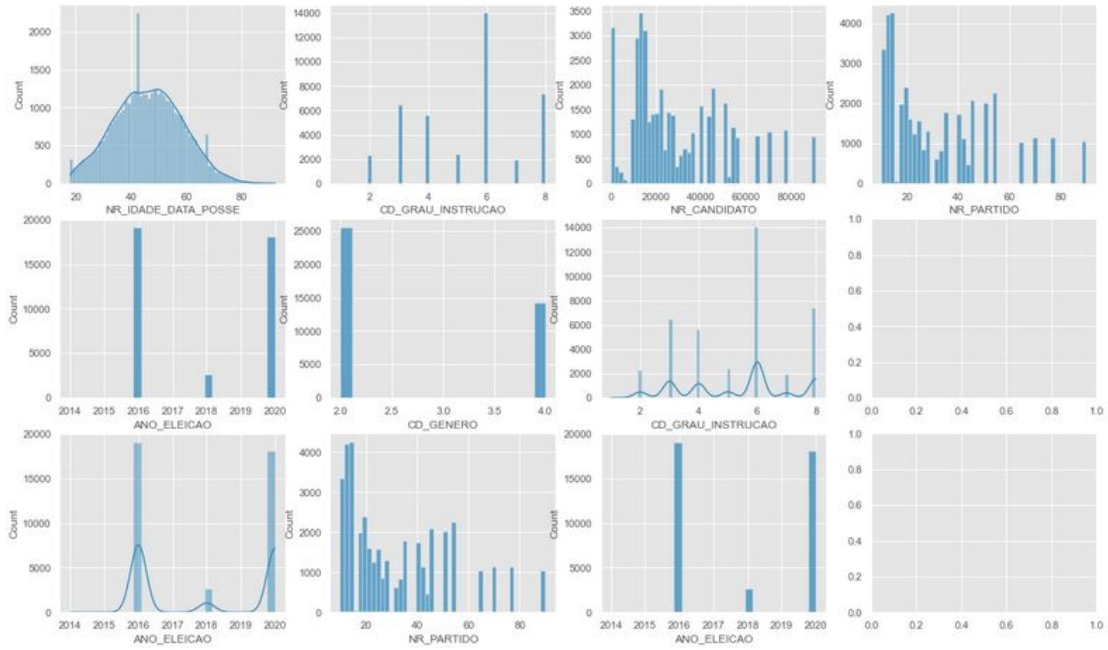
In [74]:
```
sns.boxplot(x='ST_REELEICAO', y='NR_IDADE_DATA_POSSE', data=df_eleitos_cassados)
plt.title('Candidatos Eleitos Reeleição')
plt.show()
```



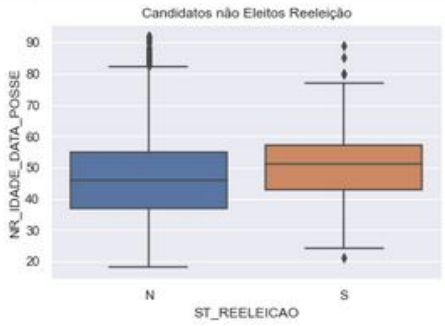
In [65]:
```
cassados_nao_eleito = df_nao_eleitos_cassados.loc[df_nao_eleitos_cassados['DS_MOTIVO_CASSACAO'].notnull()]
#Contagem das opções da coluna DS_GENERO
genero_candidatos_cassados_nao_eleito = Counter(cassados_nao_eleito['DS_GENERO'])
genero_candidatos_cassados_nao_eleito
```

Out[65]:
```
Counter({'FEMININO': 14326, 'MASCULINO': 25527})
```

In [67]:
```
# Distributions of the features
fig, ax = plt.subplots(3, 4, figsize=(20, 12))
sns.histplot(df_nao_eleitos_cassados['NR_IDADE_DATA_POSSE'], kde=True, ax=ax[0, 0])
sns.histplot(df_nao_eleitos_cassados['CD_GRAU_INSTRUCAO'], ax=ax[0, 1])
sns.histplot(df_nao_eleitos_cassados['NR_CANDIDATO'], ax=ax[0, 2])
sns.histplot(df_nao_eleitos_cassados['NR_PARTIDO'], ax=ax[0, 3])
sns.histplot(df_nao_eleitos_cassados['ANO_ELEICAO'], ax=ax[1, 0])
sns.histplot(df_nao_eleitos_cassados['CD_GENERO'], ax=ax[1, 1])
sns.histplot(df_nao_eleitos_cassados['CD_GRAU_INSTRUCAO'], kde=True, ax=ax[1, 2])
sns.histplot(df_nao_eleitos_cassados['ANO_ELEICAO'], kde=True, ax=ax[2, 0])
sns.histplot(df_nao_eleitos_cassados['NR_PARTIDO'], ax=ax[2, 1])
sns.histplot(df_nao_eleitos_cassados['ANO_ELEICAO'], ax=ax[2, 2])
plt.show()
```



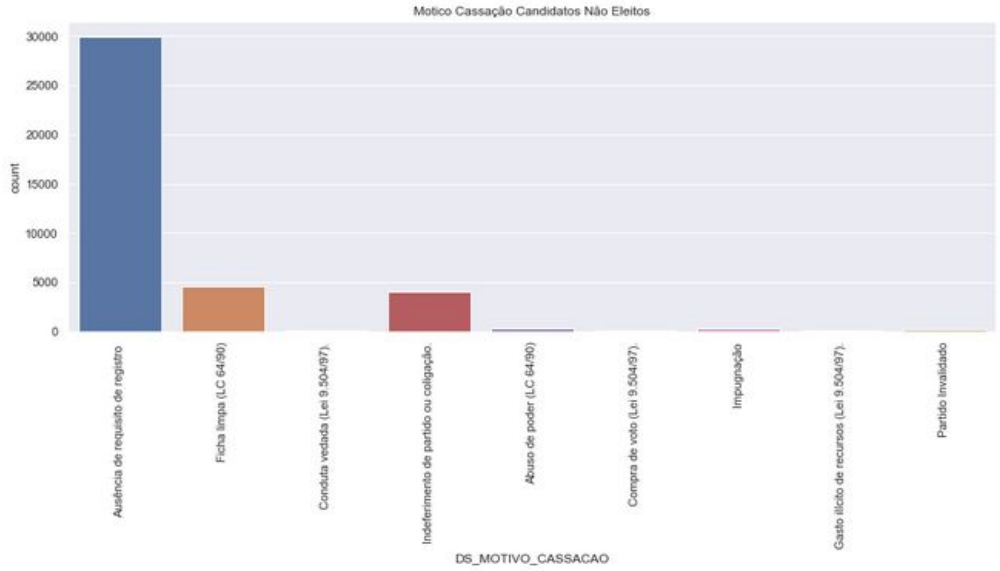
In [75]:
```
sns.boxplot(x='ST_REELEICAO', y='NR_IDADE_DATA_POSSE', data=df_nao_eleitos_cassados)
plt.title('Candidatos não Eleitos Reeleição')
plt.show()
```



In [70]:
```
df_nao_eleitos_cassados.groupby('DS_MOTIVO_CASSACAO')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
```

Out[70]:
```
DS_MOTIVO_CASSACAO
Ausência de requisito de registro          29960
Ficha limpa (LC 64/90)                      4619
Indeferimento de partido ou coligação.      4062
Impugnação                                   356
Abuso de poder (LC 64/90)                    335
Partido Invalidado                           223
Conduta vedada (Lei 9.504/97).               116
Compra de voto (Lei 9.504/97).                96
Gasto ilícito de recursos (Lei 9.504/97).     86
Name: NR_CPF_CANDIDATO, dtype: int64
```

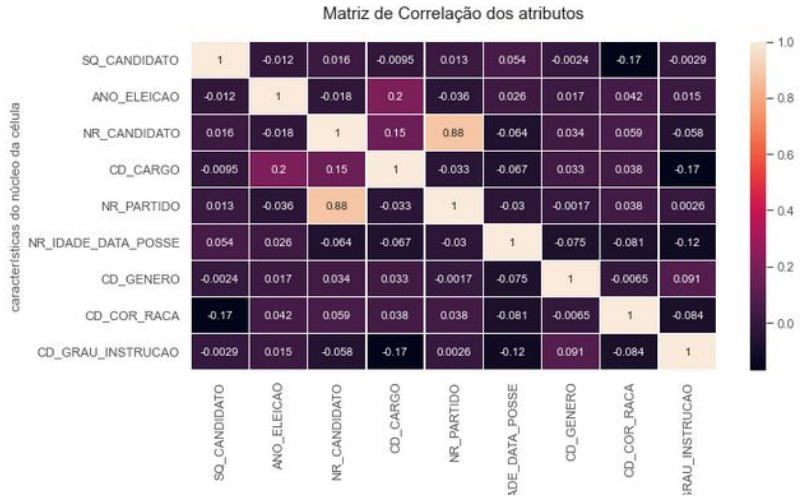
In [71]:
```
sns.set(style="darkgrid")       #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount  = sns.countplot(x="DS_MOTIVO_CASSACAO", data=df_nao_eleitos_cassados).set_title("Motico Cassação Candidatos Não Eleitos"
```



Plotagem da matriz de correlação que é uma tabela que contém coeficientes de correlação entre variáveis. Cada célula da tabela representa a correlação entre duas variáveis. O valor está entre -1 e 1.

```
In [45]:
# Gráfico de Matriz de Correlação
df_small = df_candidato_cassacao.copy()
# O corr() método do Pandas DataFrame  é usado para calcular a matriz.
# Por padrão, ele calcula o coeficiente de correlação de Pearson
correlation_mat = df_small.corr(method='pearson')
# Definindo as configurações do Gráfico
fig = plt.figure(figsize=(8,4))
eixo = fig.add_axes([0, 0, 1, 1])
# Usando o método heatmap para traçar a Matriz
# O parâmetro ' annot=True' exibe os valores do coeficiente de correlação em cada célula.
sns.heatmap(correlation_mat, annot = True, linewidth=0.5)
# Definindo titulo e labels do gráfico
eixo.set_title('Matriz de Correlação dos atributos', fontsize=15, pad=20)
eixo.set_xlabel ("características do núcleo da célula")
eixo.set_ylabel ("características do núcleo da célula")

Out[45]: Text(-23.500000000000007, 0.5, 'características do núcleo da célula')
```



```
In [46]: 1 df_candidato_cassacao['DS_MOTIVO_CASSACAO'].unique()

Out[46]: array([nan, 'Ausência de requisito de registro ',
                'Ficha limpa (LC 64/90)', 'Conduta vedada (Lei 9.504/97).',
                'Indeferimento de partido ou coligação.',
                'Abuso de poder (LC 64/90)', 'Compra de voto (Lei 9.504/97).',
                'Impugnação', 'Gasto ilícito de recursos (Lei 9.504/97).',
                'Partido Invalidado'], dtype=object)
```

Analisando o “dataFrame” “df_candidato_cassacao” identificamos um total de 1115031 linhas , quando consideramos a situação do candidato a ser inserido na urna, nos deparamos com um total de 1050992 considerados Aptos, considerados inaptos um total de 64021 registros.

```
In [131]: 1 df_candidato_cassacao.shape

Out[131]: (1115031, 28)

In [50]: 1 df_candidato_cassacao.groupby('DS_SITUACAO_CANDIDATURA')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)

Out[50]: DS_SITUACAO_CANDIDATURA
         APTO          1050992
         INAPTO          64021
         CADASTRADO         18
         Name: NR_CPF_CANDIDATO, dtype: int64

In [51]: 1 data_frame.groupby('ST_CANDIDATO_INSERIDO_URNA')['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)

Out[51]: ST_CANDIDATO_INSERIDO_URNA
         SIM    1072398
         NÃO      39692
         Name: NR_CPF_CANDIDATO, dtype: int64
```

5. Criação de Modelos de Machine Learning

O objetivo deste estudo é identificar quais os principais motivos que levam um candidato a não estar APTO, optou-se por usar algoritmos de classificação , que são cálculos preditivos usados para atribuir dados em categorias, analisando o conjunto de treinamento.

Para iniciar a continuação do tratamento dos dados foi realizado uma cópia dos dados do “Data-Set” “df_candidato_cassação”

```
In [76]: 1 df_candidato = df_candidato_cassacao.copy()
In [77]: 1 df_candidato
Out[77]:
```

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF	…
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC	…
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC	…
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC	…
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC	…
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC	…
…	…	…	…	…	…	…	…	…	…	…	…
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA SILVA	13	VEREADOR	19	PODE	PE	…
1115027	893076343	2020	26951640840	10580	MARISTELA CRISTINA ALBERTO	13	VEREADOR	10	REPUBLICANOS	SP	…
1115028	-128557416	2020	35720638334	55	ANTONIA MARIA DE JESUS ALBUQUERQUE	12	VICE-PREFEITO	55	PSD	CE	…
1115029	-193108551	2020	72054085120	45325	MARIA LÚCIA CUNHA LINHARES	13	VEREADOR	45	PSDB	GO	…
1115030	-193474012	2020	04356899164	12222	RAFAEL SANTOS DANTAS	13	VEREADOR	12	PDT	GO	…

1115031 rows × 28 columns

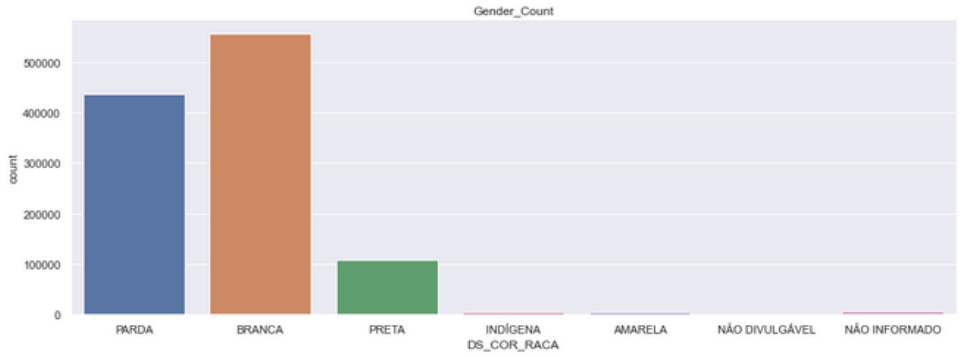
Buscou-se identificar os dados categóricos, através do comando categorical_columns ou seja são as características que não possuem valores quantitativos, mas, ao contrário, são definidas por várias categorias, ou seja, representam uma classificação dos indivíduos. Podem ser nominais ou ordinais. podemos realizar a plotagem dos dados identificados automaticamente.

```
In [78]:
categorical_columns = [cname for cname in df_candidato.columns if df_candidato[cname].dtype == "object"]

In [79]:
categorical_columns

Out[79]:
['DS_MOTIVO_CASSACAO', 'DS_COR_RACA']

In [80]:
sns.set(style="darkgrid")       #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount  = sns.countplot(x="DS_COR_RACA", data=df_candidato).set_title("Gender_Count")
```



Para iniciar a preparação dos dados utilizamos o comando “isnull” Esta função leva um objeto escalar ou semelhante a uma matriz e indica se os valores estão faltando (em matrizes numéricas ou em matrizes de objetos, em datatimelike) e identificar a necessidade de tratamento ou eliminação dos dados não utilizados.

```
In [83]:
df_candidato.isnull().sum()

Out[83]:
SQ_CANDIDATO                    0
ANO_ELEICAO                     0
NR_CPF_CANDIDATO                0
NR_CANDIDATO                    0
NM_CANDIDATO                    0
CD_CARGO                        0
DS_CARGO                        0
NR_PARTIDO                      0
SG_PARTIDO                      0
SG_UF                           0
DS_SIT_TOT_TURNO                0
NR_IDADE_DATA_POSSE             0
CD_GENERO                       0
DS_GENERO                       0
DS_MOTIVO_CASSACAO        1075091
DS_ESTADO_CIVIL                 0
CD_COR_RACA                     0
DS_COR_RACA                     0
ST_REELEICAO                    0
TP_ABRANGENCIA                  0
ST_DECLARAR_BENS                0
DS_GRAU_INSTRUCAO               0
CD_GRAU_INSTRUCAO               0
DS_SITUACAO_CANDIDATURA         0
DT_NASCIMENTO                 348
CD_SITUACAO_CANDIDATO_URNA      0
ST_CANDIDATO_INSERIDO_URNA      0
VR_DESPESA_MAX_CAMPANHA         0
dtype: int64
```

Separando as variáveis numéricas das categóricas Variáveis numéricas são aquelas variáveis que assumem valores numéricos, por exemplo, a variável idade. As variáveis numéricas são classificadas como variáveis contínuas ou discretas.

As variáveis contínuas assumem valores na reta real, como a variável Salário Estimado. E as variáveis discretas são aquelas que assumem valores inteiros, como a variável número de produtos.

Variáveis categóricas são variáveis que não assumem valores numéricos. Por exemplo, a variável estado.

As variáveis categóricas são classificadas como nominais e ordinais. As variáveis categóricas nominais são aquelas que não têm nenhuma ordem envolvida, por exemplo, a variável sexo e ordinais quando temos uma ordem envolvida, como a variável grau de escolaridade.

No pré processamento dos dados separamos as variáveis entre categóricas e numéricas, pois para cada tipo de variável utilizamos técnicas de processamento diferentes.

```
In [84]:
1 data_gd = pd.get_dummies(df_candidato, prefix_sep='_', drop_first=True)
2 data_gd.head(8)
```

Out[84]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF	...	DS_MOTI\ Ilícito de
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC	...	
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC	...	
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC	...	
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC	...	
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC	...	
5	1410065687	2014	21617015253	19012	MARISELVA PEREIRA DOS SANTOS	7	DEPUTADO ESTADUAL	19	PTN	AC	...	
6	1410065510	2014	47768258215	44666	MAGID KASSEN MASTUB NETO	7	DEPUTADO ESTADUAL	44	PRP	AC	...	
7	1410065943	2014	16451724220	15999	ESPERIDIÃO MENEZES JUNIOR	7	DEPUTADO ESTADUAL	15	PMDB	AC	...	

8 rows × 40 columns

Nota-se que anterior ao comando existia 28 colunas, o acréscimo de coluna se deve ao tratamento dos dados categóricos.

```
In [76]:
1 df_candidato = df_candidato_cassacao.copy()
```

```
In [77]:
1 df_candidato
```

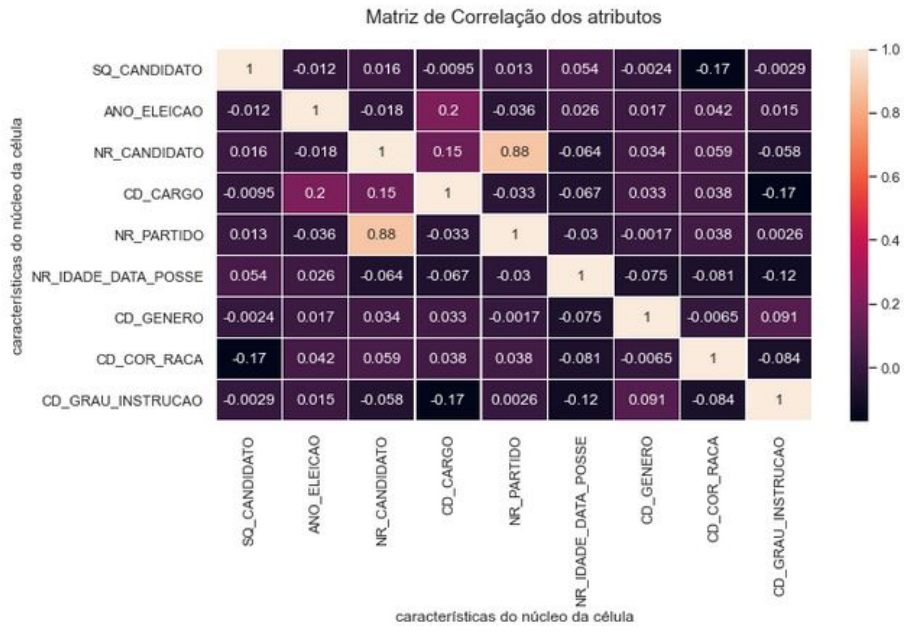
Out[77]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF	...
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC	...
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC	...
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC	...
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC	...
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC	...
...	...	...	...	...	...	...	...	...	...	...	...
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA SILVA	13	VEREADOR	19	PODE	PE	...
1115027	893076343	2020	26951640840	10580	MARISTELA CRISTINA ALBERTO	13	VEREADOR	10	REPUBLICANOS	SP	...
1115028	-128557416	2020	35720638334	55	ANTONIA MARIA DE JESUS ALBUQUERQUE	12	VICE-PREFEITO	55	PSD	CE	...
1115029	-193108551	2020	72054085120	45325	MARIA LÚCIA CUNHA LINHARES	13	VEREADOR	45	PSDB	GO	...
1115030	-193474012	2020	04356899164	12222	RAFAEL SANTOS DANTAS	13	VEREADOR	12	PDT	GO	...

1115031 rows × 28 columns

```
In [85]:
# O corr() método do Pandas DataFrame  é usado para calcular a matriz.
# Por padrão, ele calcula o coeficiente de correlação de Pearson
correlation_mat = df_candidato.corr(method='pearson')
# Definindo as configurações do Gráfico
fig = plt.figure(figsize=(8,4))
eixo = fig.add_axes([0, 0, 1, 1])
# Usando o método heatmap para traçar a Matriz
# O parâmetro ' annot=True' exibe os valores do coeficiente de correlação em cada célula.
sns.heatmap(correlation_mat, annot = True, linewidth=0.5)
# Definindo titulo e labels do gráfico
eixo.set_title('Matriz de Correlação dos atributos', fontsize=15, pad=20)
eixo.set_xlabel ("características do núcleo da célula")
eixo.set_ylabel ("características do núcleo da célula")
```

```
Out[85]: Text(-23.500000000000007, 0.5, 'características do núcleo da célula')
```



Nesta etapa foi realizado a separação das variáveis continuas ('st_reeleicao','ds_cargo','ds_estado_civil','ds_sit_tot_turno','vr_despesa_max_campanha','nr_idade_data_posse','sg_partido','ds_cargo','tp_abrangencia')

Foi realizada a importação de classe reponsável por realizar a matriz de confusão é uma tabela que permite extrair métricas que auxiliam na avaliação de modelos de machine learning para classificação.

```
In [87]:
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

In [88]:
#Variáveis Continuas
x_cont  = ['ST_REELEICAO','DS_CARGO','DS_ESTADO_CIVIL','DS_SIT_TOT_TURNO','VR_DESPESA_MAX_CAMPANHA','NR_IDADE_DATA_POSSE','SG_P

In [89]:
#Variáveis Categóricas
x_cat = list(set(data_gd) - set(x_cont))
x_dummies = data_gd[x_cat]
```

Nas variáveis categóricas optou-se por converter as variáveis categóricas em variáveis fictícias que foram representadas como variáveis numéricas, utilizou-se o comando abaixo para obter os dummies .

```
In [89]:
#Variáveis Categóricas
x_cat = list(set(data_gd) - set(x_cont))
x_dummies = data_gd[x_cat]

In [90]:
data_gd
```

Out[90]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF	…
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC	…
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC	…
2	1410065722	2014	30870966220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC	…
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC	…
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC	…
…	…	…	…	…	…	…	…	…	…	…	…
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA SILVA	13	VEREADOR	19	PODE	PE	…
1115027	893076343	2020	26951640840	10580	MARISTELA CRISTINA ALBERTO	13	VEREADOR	10	REPUBLICANOS	SP	…
1115028	-128557416	2020	35720638334	55	ANTONIA MARIA DE JESUS ALBUQUERQUE	12	VICE-PREFEITO	55	PSD	CE	…
1115029	-193108551	2020	72054085120	45325	MARIA LÚCIA CUNHA LINHARES	13	VEREADOR	45	PSDB	GO	…
1115030	-193474012	2020	04356899164	12222	RAFAEL SANTOS DANTAS	13	VEREADOR	12	PDT	GO	…

1115031 rows × 40 columns

In [91]:
```
##Substituindo a variável sexo para 0 e 1
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_gd['DS_SIT_TOT_TURNO'] = le.fit_transform(data_gd['DS_SIT_TOT_TURNO'])
data_gd.head(10)
```

Out[91]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	DS_CARGO	NR_PARTIDO	SG_PARTIDO	SG_UF	...	DS_MOTIV Ilícito de r
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	DEPUTADO ESTADUAL	54	PPL	AC	...	
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	DEPUTADO ESTADUAL	40	PSB	AC	...	
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	DEPUTADO ESTADUAL	11	PP	AC	...	
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	DEPUTADO ESTADUAL	43	PV	AC	...	
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	DEPUTADO ESTADUAL	43	PV	AC	...	
5	1410065687	2014	21617015253	19012	MARISELVA PEREIRA DOS SANTOS	7	DEPUTADO ESTADUAL	19	PTN	AC	...	
6	1410065510	2014	47768258215	44666	MAGID KASSEN MASTUB NETO	7	DEPUTADO ESTADUAL	44	PRP	AC	...	
7	1410065943	2014	16451724220	15999	ESPERIDIÃO MENEZES JUNIOR	7	DEPUTADO ESTADUAL	15	PMDB	AC	...	
8	1410065831	2014	19691394200	27123	MANOEL LOPES DA SILVA	7	DEPUTADO ESTADUAL	27	PSDC	AC	...	
9	1410065602	2014	52775364268	13420	CAIO CESAR PEREIRA PINHEIRO	7	DEPUTADO ESTADUAL	13	PT	AC	...	

10 rows × 40 columns

Pandas get dummies (pd.get_dummies()) permite que você codifique facilmente seus dados categóricos. A codificação one-hot é uma etapa importante para preparar seu conjunto de dados para uso em aprendizado de máquina. A codificação one-hot transforma seus dados categóricos em uma representação vetorial binária. Pandas pegam manequins torna isso muito fácil!

Isso significa que para cada valor exclusivo em uma coluna, uma nova coluna é criada. Os valores nesta coluna são representados como 1s e 0s, dependendo se o valor corresponde ao cabeçalho da coluna.

Uma codificação a quente pode ser muito útil em termos de trabalho com variáveis categóricas. Uma grande desvantagem, no entanto, é que ele cria significativamente mais dados. Por isso, não deve ser usado quando há muitas categorias

Depois de iniciar a codificação one-hot de várias colunas, pode ficar um pouco confuso. Vamos ver como fazer isso usando o prefix=parâmetro. Por padrão, o parâmetro prefix= será separado por um sublinhado (_).

Neste estudo foi utilizado 4 tipos de algotitimos de classificação (OneVsRest, OneVsOne, MultinomialNB, AdaBoostClassifier)

In [93]:

```
x_final = pd.get_dummies(data = data_gd,
                         columns = ['SG_UF','DS_SIT_TOT_TURNO','DS_GRAU_INSTRUCAO','SG_PARTIDO','DS_CARGO','TP_ABRANGENCIA'],
                         prefix_sep='_',
                         dummy_na=False,
                         sparse=False,
                         drop_first=False,
                         dtype=None
                        )
x_final
```

Out[93]:

	SQ_CANDIDATO	ANO_ELEICAO	NR_CPF_CANDIDATO	NR_CANDIDATO	NM_CANDIDATO	CD_CARGO	NR_PARTIDO	NR_IDADE_DATA_POSSE	CD_GENERO	DS_
0	1410065661	2014	49513427234	54444	GILDOMAR OLIVEIRA GOMES	7	54	42.0	2	MA
1	1410065850	2014	43503594272	40140	WILSON DE MELO LUNA	7	40	43.0	2	MA
2	1410065722	2014	30870968220	11223	KIEFER ROBERTO CAVALCANTE LIMA	7	11	46.0	2	MA
3	1410065737	2014	76492524268	43777	FRANCINEUDO SOUZA DA COSTA	7	43	29.0	2	MA
4	1410065799	2014	51344840230	43250	MICHELE SARAIVA SAMPAIO	7	43	37.0	4	F
...	...	...	...	...	...	...	...	...	...	
1115026	-1797587048	2020	74351850434	19111	MARIA SÔNIA TEIXEIRA DA SILVA	13	19	51.0	4	F
1115027	893076343	2020	26951640840	10580	MARISTELA CRISTINA ALBERTO	13	10	42.0	4	F
1115028	-128557416	2020	35720638334	55	ANTONIA MARIA DE JESUS ALBUQUERQUE	12	55	61.0	4	F
1115029	-193108551	2020	72054085120	45325	MARIA LÚCIA CUNHA LINHARES	13	45	49.0	4	F
1115030	-193474012	2020	04356899164	12222	RAFAEL SANTOS DANTAS	13	12	27.0	2	MA

1115031 rows × 138 columns

Out[93]:

NERO	...	DS_CARGO_PREFEITO	DS_CARGO_PRESIDENTE	DS_CARGO_SENADOR	DS_CARGO_VEREADOR	DS_CARGO_VICE-GOVERNADOR	DS_CARGO_VICE-PREFEITO	DS_CARGO_VICE-PRESIDENTE	TP_A
ULINO	...	0	0	0	0	0	0	0	
ULINO	...	0	0	0	0	0	0	0	
ULINO	...	0	0	0	0	0	0	0	
ULINO	...	0	0	0	0	0	0	0	
MININO	...	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
MININO	...	0	0	0	1	0	0	0	
MININO	...	0	0	0	1	0	0	0	
MININO	...	0	0	0	0	0	1	0	
MININO	...	0	0	0	1	0	0	0	
ULINO	...	0	0	0	1	0	0	0	

```
In [94]: 1 x_dummies.head(8)
```

Out[94]:

	SQ_CANDIDATO	CD_GENERO	DS_COR_RACA_NÃO DIVULGÁVEL	CD_GRAU_INSTRUCAO	DS_MOTIVO_CASSACAO_Ficha limpa (LC 64/90)	NR_CANDIDATO	DS_COR_RACA_BRANCA	DS_MOTIVO_C
0	1410065661	2	0	7	0	54444	0	
1	1410065850	2	0	4	0	40140	0	
2	1410065722	2	0	2	0	11223	1	
3	1410065737	2	0	8	0	43777	0	
4	1410065799	4	0	8	0	43250	1	
5	1410065687	4	0	4	0	19012	0	
6	1410065510	2	0	6	0	44666	0	
7	1410065943	2	0	8	0	15999	0	

8 rows × 32 columns

```
In [95]: 1 x_dummies.columns
```

```
Out[95]: Index(['SQ_CANDIDATO', 'CD_GENERO', 'DS_COR_RACA_NÃO DIVULGÁVEL',
       'CD_GRAU_INSTRUCAO', 'DS_MOTIVO_CASSACAO_Ficha limpa (LC 64/90)',
       'NR_CANDIDATO', 'DS_COR_RACA_BRANCA', 'DS_MOTIVO_CASSACAO_Impugnação',
       'ST_DECLARAR_BENS', 'DS_COR_RACA_PARDA',
       'DS_MOTIVO_CASSACAO_Compra de voto (Lei 9.504/97).',
       'DS_SITUACAO_CANDIDATURA', 'DS_COR_RACA_PRETA',
       'CD_SITUACAO_CANDIDATO_URNA', 'NR_CPF_CANDIDATO', 'NR_PARTIDO',
       'DT_NASCIMENTO', 'CD_CARGO',
       'DS_MOTIVO_CASSACAO_Conduta vedada (Lei 9.504/97).', 'SG_UF',
       'CD_COR_RACA',
       'DS_MOTIVO_CASSACAO_Indeferimento de partido ou coligação.',
       'DS_MOTIVO_CASSACAO_Partido Invalidado', 'NM_CANDIDATO', 'ANO_ELEICAO',
       'DS_GRAU_INSTRUCAO',
       'DS_MOTIVO_CASSACAO_Ausência de requisito de registro ',
       'ST_CANDIDATO_INSERIDO_URNA',
       'DS_MOTIVO_CASSACAO_Gasto ilícito de recursos (Lei 9.504/97).',
       'DS_COR_RACA_INDÍGENA', 'DS_GENERO', 'DS_COR_RACA_NÃO INFORMADO'],
      dtype='object')
```

Realizou-se uma cópia dos dados para o “Data-Frame” “df” para porterior-mente ser aplicado os algoritmos de classificação.

Antes de se aplicar os algoritmos de classificação, é necessário definir qual medida de avaliação será analisada para verificar a eficiência do modelo. Com esse objetivo, serão utilizadas duas funções da biblioteca Scikit-learn: “accuracy_score” e “classification_report”. Essas funções mostram as medidas de acurácia, precisão, revocação (recall) e f1-score.

A acurácia é a quantidade de acertos do modelo dividido pelo total da amostra e indica uma performance geral do modelo:

$$Acurácia = \frac{Verdadeiros\ Positivos + Verdadeiros\ Negativos}{Total\ de\ Amostras}$$

A precisão define os chamados positivos verdadeiros, ou seja, dentre os exemplos classificados como verdadeiros, quantos eram realmente verdadeiros.

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos}$$

A revocação (recall) indica qual a porcentagem de dados classificados como verdadeiros comparado com a quantidade real de resultados verdadeiros que existem na amostra.

$$F1 - score = \frac{2 * Precisão\ * Revocação}{Precisão + Revocação}$$

In [100]:
```
X_df = df[['NR_PARTIDO', 'NR_PARTIDO','CD_CARGO','NR_IDADE_DATA_POSSE','ANO_ELEICAO']]
Y_df = df['ANO_ELEICAO']

Xdummies_df = pd.get_dummies(X_df)
Ydummies_df = Y_df

X = Xdummies_df.values
Y = Ydummies_df.values

porcentagem_de_treino = 0.8
porcentagem_de_teste = 0.1

tamanho_de_treino = int(porcentagem_de_treino * len(Y))
tamanho_de_teste = int(porcentagem_de_teste * len(Y))
tamanho_de_validacao = len(Y) - tamanho_de_treino - tamanho_de_teste

treino_dados = X[:tamanho_de_treino]
treino_marcacoes = Y[:tamanho_de_treino]

fim_de_treino = tamanho_de_treino + tamanho_de_teste

teste_dados = X[tamanho_de_treino:fim_de_treino]
teste_marcacoes = Y[tamanho_de_treino:fim_de_treino]

validacao_dados = X[fim_de_treino:]
validacao_marcacoes = Y[fim_de_treino:]
```

In [101]:
```
from sklearn.metrics import confusion_matrix, classification_report
from pickle import dump, load
```

In [102]:
```
def fit_and_predict(nome, modelo, X_train, Y_train, teste_dados, teste_marcacoes):
    modelo.fit(X_train, Y_train)

    #Testa para ver a sua veracidade
    predictions = modelo.predict(teste_dados)
    print(confusion_matrix(teste_marcacoes, predictions))
    print(classification_report(teste_marcacoes, predictions))

    #Salva para uso futuro
    filename = 'binary_class_model.sav'
    #dump(modelo, open(filename, 'wb'))

    #Le o arquivo do disco
    #Loaded_model = load(open('binary_class_model.sav', 'rb'))

    resultado = modelo.predict(teste_dados)
    acertos = resultado == teste_marcacoes
    total_de_acertos = sum(acertos)
    total_de_elementos = len(teste_dados)
    taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos
    msg = "Taxa de acerto do algoritmo {0}: {1}".format(nome, taxa_de_acerto)
    print(msg)
    return taxa_de_acerto


def teste_real(modelo, validacao_dados, validacao_marcacoes):
    resultado = modelo.predict(validacao_dados)
    acertos = resultado == validacao_marcacoes
    total_de_acertos = sum(acertos)
    total_de_elementos = len(validacao_marcacoes)
    taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos
    msg = "Taxa de acerto do vencedor entre os dois algoritmos no mundo real: {0}".format(taxa_de_acerto)
    print(msg)
```

```
In [103]:
resultados = {}
```

```
In [104]:
#Ajuste do Modelo KNN - https://github.com/alura-cursos/machine-learning-introducao-a-classificacao-2/blob/master/situacao_do_c
from sklearn.neighbors import KNeighborsClassifier #classificador
knn = KNeighborsClassifier()
knn.fit(treino_dados, treino_marcacoes)
resultado_knn = knn.predict(teste_dados)

resultado_knn
```

```
Out[104]: array([2020, 2020, 2020, ..., 2020, 2020, 2020])
```

Definidos os dataframes de treinamento e de teste e escolhidas as medidas de avaliação, o próximo passo é a efetiva utilização dos algoritmos listados anteriormente. O processo para todos será o mesmo, começando pela importação do respectivo algoritmo

```
In [105]:
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
modeloOneVsRest = OneVsRestClassifier(LinearSVC(random_state = 0))
resultadoOneVsRest = fit_and_predict("OneVsRest", modeloOneVsRest, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes
resultados[resultadoOneVsRest] = modeloOneVsRest
```

```
[[    0     0     0     0]
 [    0     0     0     0]
 [    0     0     0     0]
 [16259 33607  3876 57761]]
              precision    recall  f1-score   support

        2014       0.00      0.00      0.00         0
        2016       0.00      0.00      0.00         0
        2018       0.00      0.00      0.00         0
        2020       1.00      0.52      0.68    111503

    accuracy                           0.52    111503
   macro avg       0.25      0.13      0.17    111503
weighted avg       1.00      0.52      0.68    111503

Taxa de acerto do algoritmo OneVsRest: 51.80219366295077
```

```
In [106]:
from sklearn.multiclass import OneVsOneClassifier
modeloOneVsOne = OneVsOneClassifier(LinearSVC(random_state=0))
resultadoOneVsOne = fit_and_predict("OneVsOne", modeloOneVsOne, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)
resultados[resultadoOneVsOne] = modeloOneVsOne
```

```
[[     0      0]
 [111503      0]]
              precision    recall  f1-score   support

        2016       0.00      0.00      0.00       0.0
        2020       0.00      0.00      0.00  111503.0

    accuracy                           0.00  111503.0
   macro avg       0.00      0.00      0.00  111503.0
weighted avg       0.00      0.00      0.00  111503.0

Taxa de acerto do algoritmo OneVsOne: 0.0
```

```
In [107]:
from sklearn.naive_bayes import MultinomialNB
modeloMultinomial = MultinomialNB()
resultadoMultinomial = fit_and_predict("MultinomialNB", modeloMultinomial, treino_dados, treino_marcacoes, teste_dados, teste_m
resultados[resultadoMultinomial] = modeloMultinomial
```

```
[[    0     0     0]
 [    0     0     0]
 [32356 11968 67179]]
              precision    recall  f1-score   support

        2016       0.00      0.00      0.00         0
        2018       0.00      0.00      0.00         0
        2020       1.00      0.60      0.75    111503

    accuracy                           0.60    111503
   macro avg       0.33      0.20      0.25    111503
weighted avg       1.00      0.60      0.75    111503

Taxa de acerto do algoritmo MultinomialNB: 60.24860317659615
```

In [108]:
```
from sklearn.ensemble import AdaBoostClassifier
modeloAdaBoost = AdaBoostClassifier()
resultadoAdaBoost = fit_and_predict("AdaBoostClassifier", modeloAdaBoost, treino_dados, treino_marcacoes, teste_dados, teste_ma
resultados[resultadoAdaBoost] = modeloAdaBoost
```

```
[[111503]]
              precision    recall  f1-score   support

        2020       1.00      1.00      1.00    111503

    accuracy                           1.00    111503
   macro avg       1.00      1.00      1.00    111503
weighted avg       1.00      1.00      1.00    111503

Taxa de acerto do algoritmo AdaBoostClassifier: 100.0
```

In [109]:
```
print (resultados)
maximo = max(resultados)
vencedor = resultados[maximo]
print (vencedor)

resultado = vencedor.predict(validacao_dados)
acertos = (resultado == validacao_marcacoes)

total_de_acertos = sum(acertos)
total_de_elementos = len(validacao_marcacoes)
taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos

msg = "Taxa de acerto do vencedor entre os dois algoritmos no mundo real: {0}".format(taxa_de_acerto)
print(msg)
```

```
{51.80219366295077: OneVsRestClassifier(estimator=LinearSVC(random_state=0)), 0.0: OneVsOneClassifier(estimator=LinearSVC(random_st
ate=0)), 60.24860317659615: MultinomialNB(), 100.0: AdaBoostClassifier()}
AdaBoostClassifier()
Taxa de acerto do vencedor entre os dois algoritmos no mundo real: 100.0
```

In [111]:
```
# precisa fazer o FIT pois o FIT foi excluido pois não fazemos mais o treino
vencedor.fit(treino_dados, treino_marcacoes)

teste_real(vencedor, validacao_dados, validacao_marcacoes)

print("Elementos testados: %d " % len(validacao_dados))
```

```
Taxa de acerto do vencedor entre os dois algoritmos no mundo real: 100.0
Elementos testados: 111504
```

In [112]:
```
#Salva para uso futuro
filename = 'binary_class_model.sav'
dump(resultadoAdaBoost, open(filename, 'wb'))

#Le o arquivo do disco
loaded_model = load(open('binary_class_model.sav', 'rb'))
```

In [113]:
```
import seaborn as sns # statistical visualization
```

6. Interpretação dos Resultados

A construção de um modelo

Algoritimo	precisão	recall	f1-score	suporte
OneVsRest	1.00	0.52	0.68	111503
OneVsOne	0.00	0.00	0.00	0
MultinomialNB	1.00	0.60	0.75	111503
AdaBoostClassifier	1.00	1.00	1.00	111503

Analisando as informações do gráfico percebe-se que 1 que é AdaBoostClassifier algoritmo de aprendizado a ser usado para treinar os modelos fracos. Isso quase sempre não precisará ser alterado porque, de longe, o aluno mais comum para usar com o AdaBoost é uma árvore de decisão – o argumento padrão desse parâmetro.n_estimatorsé o número de modelos para treinar iterativamente. learning_rate é a contribuição de cada modelo para os pesos e padrões para 1. Reduzir a taxa de aprendizado significará que os pesos serão aumentados ou diminuídos em um pequeno grau, forçando o treinamento do modelo mais lento (mas às vezes resultando em melhores pontuações de desempenho).É exclusivo AdaBoostRegressore define a função de perda a ser usada ao atualizar pesos. Esse padrão é uma função de perda linear, mas pode ser alterada para square ou exponential.

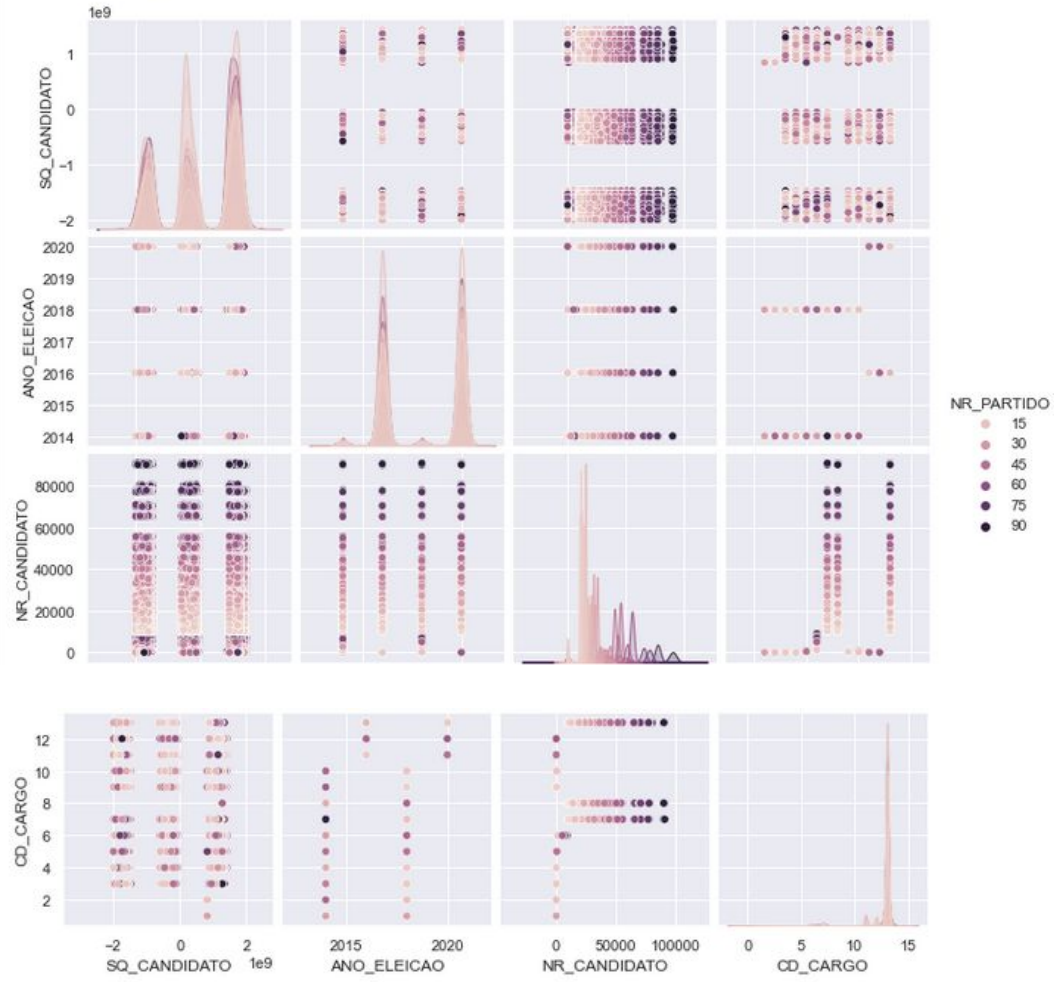
Ao se analisar a precisão dos resultas individuais, o valor obtido na análise do candidato que já que apresenta o valor 1,00, ou seja, o algoritmo não identificou nenhum falso positivo no momento de identificar candidatos que não seriam eleitos. O valor apresentado para os candidatos que seriam eleitos também chama a atenção, já traz o valor de 100 , ou seja, a cada 100 previsões de que o candidato seria eleito nenhumas delas teria um falso positivo

Temo que levar em conta que algoritmo OneVsOne identificou o valor de precisão 0, considerando que este algoritmo é útil para criar modelos que prevejam três ou mais resultados possíveis, quando o resultado depende de variáveis preditoras contínuas ou categóricas. Este método também permite que você use métodos de classificação binária para problemas que requerem várias classes de saída

Considerando o uso do algoritmo AdaBoostClassifier em um cenário em que se obtenha todos os dados de registro do TSE é possivel realizar uma previsão acertiva das chances de um candidato não estar apto a concorrer uma eleição ou nos casos mais severos ter sua candidatura cassada.

Em resumo, se não existirem ações concretas em mudar o grau de instrução e o processo de candidatura, o resultado da eleição é totalmente previsível e o perfil dos grupos que compõe o cenário político do Brasil deve continuar existindo.

Out[118]: <seaborn.axisgrid.PairGrid at 0x1fb2d2b0320>



8. 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://www.youtube.com/watch?v=4aqx8FnyakE

Link para o repositório: https://github.com/wendercorrea/projetoFinalPuc/tree/main

REFERÊNCIAS

CAJADO, ANE FERRARI RAMOS . DORNELLES, THIAGO . PEREIRA, AMANDA CAMYLLA . ELEIÇÕES NO BRASIL : UMA HISTÓRIA DE 500 ANOS, 2014

APÊNDICE

Programação/Scripts

```
# <h1><center>PUC Minas - PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS</center></h1>

<h2><center>Pós-Graduação em Ciência de Dados e Big Data</center></h2>

<h3><center>TRABALHO DE CONCLUSÃO DE CURSO</center><br>
TÍTULO:<br>
<br>
ALUNO: Wender Pereira Corrêa</h3>
Matrícula:
<p>Este notebook é referente ao Trabalho de Conclusão de Curso em Ciência de Dados e Big Data.</p>
#Leitura dos dados e importação das bibliotecas utilizadas
#IMPORTAÇÃO DAS BIBLIOTECAS
import pandas as pd
import numpy as np
# dependências para importar e descompactar arquivos em zip
import zipfile
import requests
from io import BytesIO
import os
import glob
import seaborn as sns
sns.set_theme(style="whitegrid", palette="muted")
import time
#Importação da função Counter e da biblioteca matplotlib.pyplot
from collections import Counter
import matplotlib.pyplot as plt
#Plotação de um gráfico para mostrar a dispersão dos dados.
```

```
#O % e inline significa que o gráfico será mostrado aqui no notebook, e não em um arquivo
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
#Definindo a pasta de trabalho
dirCandidatoAnalise        = "./candidatoAnalise"
dirCandidatoCassacaoAnalise = "./candidatoCassacaoAnalise"

os.makedirs(dirCandidatoAnalise, exist_ok=True)
os.makedirs(dirCandidatoCassacaoAnalise, exist_ok=True)
def descompactar(nome_arquivo_compactado, diretorio):
   #Lê o arquivo compactado e extrai o conteúdo
   #print(nome_arquivo_compactado)
   filebytes = BytesIO(
      requests.get(nome_arquivo_compactado).content
   )
   myzip = zipfile.ZipFile(filebytes)
   myzip.extractall(diretorio)
   return "Concluído"
<b>1) Dados Politicos</b>
ca = 2014
while ca < 2022:
                                                    urlCandidatoAnalise                =
"https://cdn.tse.jus.br/estatistica/sead/odsele/consulta_cand/consulta_cand_"   +   str(ca)+
".zip"
   descompactar(urlCandidatoAnalise,dirCandidatoAnalise)
                                                        urlCandidatoCassacaoAnalise                =
"https://cdn.tse.jus.br/estatistica/sead/odsele/motivo_cassacao/motivo_cassacao_"         +
str(ca)+ ".zip"
   descompactar(urlCandidatoCassacaoAnalise,dirCandidatoCassacaoAnalise)
   ca+=2
<b>1) Dados Candidato</b>
```

```
print('Arquivos com extensão csv Candidato:')
concatenar = []
with os.scandir(dirCandidatoAnalise) as arqs:
   for arq in arqs:
      if arq.is_file() and  arq.name.endswith('.csv'):
         print(arq.name)
   path = dirCandidatoAnalise
   # csv files in the path
   files = glob.glob(path + "/*.csv")
   # defining an empty list to store
   # content
   data_frame = pd.DataFrame()
   content = []
   # checking all the csv files in the
   # specified path
   for filename in files:
      df = pd.read_csv(filename, encoding = "Latin 1", sep= ";", decimal = ',',
          error_bad_lines=False,dtype={"SQ_CANDIDATO": int, "ANO_ELEICAO": int,"SG_UF" :
"string", "NR_TURNO": int,"HH_GERACAO" : object ,"CD_TIPO_ELEICAO" :
int,"NM_TIPO_ELEICAO" : "string","CD_ELEICAO" :
"string","DS_ELEICAO" :"string","TP_ABRANGENCIA" : "string","SG_UE" : "string","NM_UE" :
"string","CD_CARGO" : int,"DS_CARGO" : "string","NR_CANDIDATO" :
int,"NM_CANDIDATO" : "string","NM_URNA_CANDIDATO" :
"string","NM_SOCIAL_CANDIDATO" : "string","NR_CPF_CANDIDATO" :
"string","NM_EMAIL" : "string","CD_SITUACAO_CANDIDATURA" :
"string","DS_SITUACAO_CANDIDATURA" : "string","CD_DETALHE_SITUACAO_CAND" :
"string","DS_DETALHE_SITUACAO_CAND" : "string","TP_AGREMIACAO" :
"string","NR_PARTIDO" : int,"SG_PARTIDO" : "string","NM_PARTIDO" :
"string","SQ_COLIGACAO" : "string","NM_COLIGACAO" :
"string","DS_COMPOSICAO_COLIGACAO" : "string","CD_NACIONALIDADE" :
int,"DS_NACIONALIDADE" : "string","SG_UF_NASCIMENTO" :
"string","CD_MUNICIPIO_NASCIMENTO" : "string","NM_MUNICIPIO_NASCIMENTO" :
```

```
"string","DT_NASCIMENTO" : "string","NR_IDADE_DATA_POSSE" : float ,"NR_TITULO_ELEITORAL_CANDIDATO" : "string","CD_GENERO" : int,"DS_GENERO" : "string","CD_GRAU_INSTRUCAO" : int,"DS_GRAU_INSTRUCAO" : "string","CD_ESTADO_CIVIL" : int,"DS_ESTADO_CIVIL" : "string","CD_OCUPACAO" : "string", "DS_OCUPACAO" : "string","VR_DESPESA_MAX_CAMPANHA" : "string","CD_SIT_TOT_TURNO" : "string","DS_SIT_TOT_TURNO" : "string","ST_REELEICAO" : "string","ST_DECLARAR_BENS" : "string","NR_PROTOCOLO_CANDIDATURA" : "string","NR_PROCESSO" : "string" ,"CD_SITUACAO_CANDIDATO_PLEITO" : "string","DS_SITUACAO_CANDIDATO_PLEITO" : "string","CD_SITUACAO_CANDIDATO_URNA" : "string", "DS_SITUACAO_CANDIDATO_URNA" : "string","ST_CANDIDATO_INSERIDO_URNA" : "string"})
      df.drop_duplicates()
      content.append(df)
   # converting content to data frame
   data_frame = pd.concat(content).drop_duplicates()
data_frame
data_frame.dtypes
idade_publico = Counter(data_frame['NR_IDADE_DATA_POSSE'])
#Plotagem da idade dos candidatos
plt.style.use('ggplot')
plt.bar(idade_publico.keys(), idade_publico.values())
plt.xlabel('Idade')
plt.title('Idade dos Candidatos')
plt.show()
categorical_columns_df = [cname for cname in data_frame.columns if data_frame[cname].dtype == "object"]
categorical_columns_df
#Calcula o total e a % de valores ausentes
num_ausentes = data_frame.isna().sum()
porc_ausentes = data_frame.isna().sum() * 100 / len(data_frame)
df_ausentes = pd.DataFrame({
   'Coluna': data_frame.columns,
```

```
    'Dados ausentes': num_ausentes,
    'Porcentagem': porc_ausentes
})
df_ausentes
#imprimindo as primeiras linhas
display(data_frame.head())
# Imprimindo as últimas linhas
display(data_frame.tail())
# Informações do nosso DataFrame
data_frame.describe()
# Dimensões do df_dados
data_frame.shape
res = sns.scatterplot(x=data_frame['ANO_ELEICAO'],y=data_frame['DS_CARGO'])
fig = plt.figure(figsize=(16,8))
plt.show()
data_frame.values
data_frame.columns
data_frame.index
<b> Matriz de correlação de atributos </b>
# O corr() método do Pandas DataFrame  é usado para calcular a matriz.
# Por padrão, ele calcula o coeficiente de correlação de Pearson
correlation_mat = data_frame.corr(method='pearson')
# Definindo as configurações do Gráfico
fig = plt.figure(figsize=(8,4))
eixo = fig.add_axes([0, 0, 1, 1])
# Usando o método heatmap para traçar a Matriz
# O parâmetro ' annot=True' exibe os valores do coeficiente de correlação em cada célula.
sns.heatmap(correlation_mat, annot = True, linewidth=0.5)
# Definindo titulo e labels do gráfico
eixo.set_title('Matriz de Correlação dos atributos', fontsize=15, pad=20)
eixo.set_xlabel ("características do núcleo da célula")
eixo.set_ylabel ("características do núcleo da célula")
```

```
<b> 2 - Dados Cassação </b>
print('Arquivos com extensão csv Cassação:')
concatenar = []
with os.scandir(dirCandidatoCassacaoAnalise) as arqs:
    for arq in arqs:
        if arq.is_file() and  arq.name.endswith('.csv'):
            print(arq.name)
    path = dirCandidatoCassacaoAnalise
    # csv files in the path
    files = glob.glob(path + "/*.csv")
    # defining an empty list to store
    # content
    data_frame_cassacao = pd.DataFrame()
    content_situacao = []
    # checking all the csv files in the
    # specified path
    for filename in files:
                dfs = pd.read_csv(filename, encoding = "Latin 1", sep= ";", decimal = ',',
error_bad_lines=False, dtype={"SQ_CANDIDATO": int, "ANO_ELEICAO": int})
        content_situacao.append(dfs)
    # converting content to data frame
    data_frame_cassacao = pd.concat(content_situacao).drop_duplicates()
data_frame_cassacao
#Calcula o total e a % de valores ausentes
num_ausentes = data_frame_cassacao.isna().sum()
porc_ausentes = data_frame_cassacao.isna().sum() * 100 / len(data_frame_cassacao)
df_ausentes = pd.DataFrame({
    'Coluna': data_frame_cassacao.columns,
    'Dados ausentes': num_ausentes,
    'Porcentagem': porc_ausentes
})
df_ausentes
```

```
display(data_frame_cassacao.head())
# Imprimindo as últimas linhas
display(data_frame_cassacao.tail())
# Informações do nosso DataFrame
data_frame_cassacao.describe()
# Dimensões do df_dados
data_frame_cassacao.shape
data_frame_cassacao.columns
print('Arquivos com extensão csv Candidato:')
concatenar = []
with os.scandir(dirCandidatoAnalise) as arqs:
   for arq in arqs:
      if arq.is_file() and  arq.name.endswith('.csv'):
         print(arq.name)
   path = dirCandidatoAnalise
   # csv files in the path
   files = glob.glob(path + "/*.csv")
   # defining an empty list to store
   # content
   data_frame = pd.DataFrame()
   content = []
   # checking all the csv files in the
   # specified path
   for filename in files:
      df = pd.read_csv(filename, encoding = "Latin 1", sep= ";", decimal = ',',
          error_bad_lines=False,dtype={"SQ_CANDIDATO": int, "ANO_ELEICAO": int,"SG_UF" :
"string", "NR_TURNO": int,"HH_GERACAO" : object ,"CD_TIPO_ELEICAO" :
int,"NM_TIPO_ELEICAO" : "string","CD_ELEICAO" :
"string","DS_ELEICAO" :"string","TP_ABRANGENCIA" : "string","SG_UE" : "string","NM_UE" :
"string","CD_CARGO" : int,"DS_CARGO" : "string","NR_CANDIDATO" :
int,"NM_CANDIDATO" : "string","NM_URNA_CANDIDATO" :
"string","NM_SOCIAL_CANDIDATO" : "string","NR_CPF_CANDIDATO" :
```

```
"string","NM_EMAIL" : "string","CD_SITUACAO_CANDIDATURA" :
"string","DS_SITUACAO_CANDIDATURA" : "string","CD_DETALHE_SITUACAO_CAND" :
"string","DS_DETALHE_SITUACAO_CAND" : "string","TP_AGREMIACAO" :
"string","NR_PARTIDO" : int,"SG_PARTIDO" : "string","NM_PARTIDO" :
"string","SQ_COLIGACAO" : "string","NM_COLIGACAO" :
"string","DS_COMPOSICAO_COLIGACAO" : "string","CD_NACIONALIDADE" :
int,"DS_NACIONALIDADE" : "string","SG_UF_NASCIMENTO" :
"string","CD_MUNICIPIO_NASCIMENTO" : "string","NM_MUNICIPIO_NASCIMENTO" :
"string","DT_NASCIMENTO" : "string","NR_IDADE_DATA_POSSE" :
float ,"NR_TITULO_ELEITORAL_CANDIDATO" : "string","CD_GENERO" : int,"DS_GENERO" :
"string","CD_GRAU_INSTRUCAO" : int,"DS_GRAU_INSTRUCAO" :
"string","CD_ESTADO_CIVIL" : int,"DS_ESTADO_CIVIL" : "string","CD_OCUPACAO" : "string",
"DS_OCUPACAO" : "string","VR_DESPESA_MAX_CAMPANHA" :
"string","CD_SIT_TOT_TURNO" : "string","DS_SIT_TOT_TURNO" : "string","ST_REELEICAO" :
"string","ST_DECLARAR_BENS" : "string","NR_PROTOCOLO_CANDIDATURA" :
"string","NR_PROCESSO" : "string" ,"CD_SITUACAO_CANDIDATO_PLEITO" :
"string","DS_SITUACAO_CANDIDATO_PLEITO" : "string","CD_SITUACAO_CANDIDATO_URNA"
: "string", "DS_SITUACAO_CANDIDATO_URNA" : "string","ST_CANDIDATO_INSERIDO_URNA"
: "string"})
      df.drop_duplicates()
      content.append(df)
   # converting content to data frame
   data_frame = pd.concat(content).drop_duplicates()
idade_publico = Counter(data_frame['NR_IDADE_DATA_POSSE'])
#Plotagem da idade dos candidatos
plt.style.use('ggplot')
plt.bar(idade_publico.keys(), idade_publico.values())
plt.xlabel('Idade')
plt.title('Idade dos Candidatos')
plt.show()
<b>ANÁLISE DADOS</b>
<b> Agrupamento de dados selecionando dados de interesse: </b> Dados do candidato
```

```
#removendo dados com todas linhas faltando dados
data_frame.dropna(how='all', inplace=True)
#descobrindo a idade média
mediaIdade = round(data_frame['NR_IDADE_DATA_POSSE'].mean(),0)
print(mediaIdade)
# Preenchendo a coluna com o valor da média:
data_frame.update(data_frame['NR_IDADE_DATA_POSSE'].fillna(mediaIdade))
# Dimensões do df_dados
data_frame.shape
data_frame['NR_IDADE_DATA_POSSE'].describe().astype('int')
idade_max = data_frame[data_frame['NR_IDADE_DATA_POSSE'] > 104]
idade_min = data_frame[data_frame['NR_IDADE_DATA_POSSE'] < 18]
#Seleção das colunas de interesse
df_ida_max =
idade_max[['SQ_CANDIDATO','ANO_ELEICAO','SG_UF','NR_CPF_CANDIDATO','NR_CANDIDATO','NR_IDADE_DATA_POSSE','DT_NASCIMENTO','NM_CANDIDATO','CD_SITUACAO_CANDIDATO_URNA','ST_CANDIDATO_INSERIDO_URNA','VR_DESPESA_MAX_CAMPANHA']]
df_ida_max
```

Fonte: Repositório de dados eleitorais (https://www.tse.jus.br/hotsites/catalogo-publicacoes/pdf/relatorio_eleicoes/relatorio-eleicoes-2014.pdf) Relatório das Eleições 2014 Parte III – Um Olhar Infográfico - 52 considerou idades com Inválidas e sem tratamentos

 Criação de DataSet com dados para tratamento de distorções:

 Tratamento dos dados da Idade

```
#Seleção das colunas de interesse
df_ida_min =
idade_min[['SQ_CANDIDATO','ANO_ELEICAO','NR_CPF_CANDIDATO','SG_UF','NR_CANDIDATO','NR_IDADE_DATA_POSSE','DT_NASCIMENTO','NM_CANDIDATO','CD_SITUACAO_CANDIDATO_URNA','ST_CANDIDATO_INSERIDO_URNA','VR_DESPESA_MAX_CAMPANHA']]
df_ida_min
print('--------------------------------------')
for index, row in df_ida_max.iterrows():
   data_frame.at[index , 'NR_IDADE_DATA_POSSE' ] = mediaIdade
```

```
    print('Id: ' + str(index))
    print('Ano: ' + str(row['ANO_ELEICAO']))
    print('Candidato: ' + str(row['SQ_CANDIDATO']))
    print('CPF: ' + str(row['NR_CPF_CANDIDATO']))
    print('Nome: ' + str(row['NM_CANDIDATO']))
    print('Data Nascimento: ' + str(row['DT_NASCIMENTO']))
    print('Idade Antiga: ' + str(row['NR_IDADE_DATA_POSSE']))
    print('Idade Aualizada: ' + str(mediaIdade))
    print('---------------------------------------')
print('---------------------------------------')
for index, row in df_ida_min.iterrows():
    data_frame.at[index , 'NR_IDADE_DATA_POSSE' ] = 18
    print('Id: ' + str(index))
    print('Ano: ' + str(row['ANO_ELEICAO']))
    print('Candidato: ' + str(row['SQ_CANDIDATO']))
    print('CPF: ' + str(row['NR_CPF_CANDIDATO']))
    print('Nome: ' + str(row['NM_CANDIDATO']))
    print('Data Nascimento: ' + str(row['DT_NASCIMENTO']))
    print('Idade Antiga: ' + str(row['NR_IDADE_DATA_POSSE']))
    print('Idade Atualizada: ' + str('18'))
    print('---------------------------------------')
data_frame[data_frame['NR_IDADE_DATA_POSSE'] > 104]
data_frame[data_frame['NR_IDADE_DATA_POSSE'] < 18]
data_frame['NR_IDADE_DATA_POSSE'].describe().astype('int')
<b> Agrupamento de dados selecionando dados de interesse: </b>Dados do Cassação
#Seleção das colunas de interesse
df_consulta_candidato                                                                    =
data_frame[['SQ_CANDIDATO','ANO_ELEICAO','NR_CPF_CANDIDATO','NR_CANDIDATO',
'NM_CANDIDATO','CD_CARGO','DS_CARGO','NR_PARTIDO','SG_PARTIDO','SG_UF','DS_SIT_T
OT_TURNO','NR_IDADE_DATA_POSSE','CD_GENERO','DS_GENERO',
'DT_NASCIMENTO','DS_GRAU_INSTRUCAO',
'DS_ESTADO_CIVIL','CD_COR_RACA','DS_COR_RACA','ST_REELEICAO','TP_ABRANGENCIA','ST
```

```
_DECLARAR_BENS','CD_GRAU_INSTRUCAO','DS_SITUACAO_CANDIDATURA','CD_SITUACAO_
CANDIDATO_URNA','ST_CANDIDATO_INSERIDO_URNA','VR_DESPESA_MAX_CAMPANHA']]
profile = pp.ProfileReport(df,n_extreme_obs=5, title='Análise Exploratória',explorative=True)
profile
#Seleção das colunas de interesse
df_consulta_cassacao =
data_frame_cassacao[['SQ_CANDIDATO','ANO_ELEICAO','DS_MOTIVO_CASSACAO']]
display(df_consulta_cassacao.head())
# Imprimindo as últimas linhas
display(df_consulta_cassacao.tail())
# Informações do nosso DataFrame
df_consulta_cassacao.describe()
m_df_candidato_cassacao = pd.merge(df_consulta_candidato, df_consulta_cassacao,
on=['SQ_CANDIDATO','ANO_ELEICAO'], how="left")
display(m_df_candidato_cassacao.head())
# Imprimindo as últimas linhas
display(m_df_candidato_cassacao.tail())
# Informações do nosso DataFrame
m_df_candidato_cassacao.describe()
m_df_candidato_cassacao
#Calcula o total e a % de valores ausentes
num_ausentes = m_df_candidato_cassacao.isna().sum()
porc_ausentes = m_df_candidato_cassacao.isna().sum() * 100 /
len(m_df_candidato_cassacao)
df_ausentes = pd.DataFrame({
   'Coluna': m_df_candidato_cassacao.columns,
   'Dados ausentes': num_ausentes,
   'Porcentagem': porc_ausentes
})
df_ausentes
<b> Selecionando alguns dados para análise </b>
#Seleção das colunas de interesse
```

```
df_candidato_cassacao =
m_df_candidato_cassacao[['SQ_CANDIDATO','ANO_ELEICAO','NR_CPF_CANDIDATO','NR_CA
NDIDATO',
'NM_CANDIDATO','CD_CARGO','DS_CARGO','NR_PARTIDO','SG_PARTIDO','SG_UF','DS_SIT_T
OT_TURNO','NR_IDADE_DATA_POSSE','CD_GENERO','DS_GENERO','DS_MOTIVO_CASSACAO'
,'DS_ESTADO_CIVIL','CD_COR_RACA','DS_COR_RACA','ST_REELEICAO',
'TP_ABRANGENCIA','ST_DECLARAR_BENS','DS_GRAU_INSTRUCAO','CD_GRAU_INSTRUCAO','
DS_SITUACAO_CANDIDATURA','DT_NASCIMENTO','CD_SITUACAO_CANDIDATO_URNA','ST_
CANDIDATO_INSERIDO_URNA','VR_DESPESA_MAX_CAMPANHA']]
df_candidato_cassacao
df_candidato_cassacao.groupby('DS_CARGO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount = sns.countplot(x="DS_CARGO",
data=df_candidato_cassacao).set_title("NR_CPF_CANDIDATO")
cols_dados = ["ANO_ELEICAO",
"NR_PARTIDO","SG_PARTIDO","SG_UF","ST_REELEICAO","DS_SIT_TOT_TURNO","DS_MOTIV
O_CASSACAO"]
df_candidato_cassacao[cols_dados]
df_candidato_cassacao.groupby('DS_SIT_TOT_TURNO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount = sns.countplot(x="DS_SIT_TOT_TURNO",
data=df_candidato_cassacao).set_title("NR_CPF_CANDIDATO")
df_candidato_cassacao.groupby('DS_MOTIVO_CASSACAO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
sns.set(style="darkgrid")      #style the plot background to become a grid
```

```
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount = sns.countplot(x="DS_MOTIVO_CASSACAO",
data=df_candidato_cassacao).set_title("NR_CPF_CANDIDATO")
# Gráfico de Matriz de Correlação
df_small = df_candidato_cassacao.copy()
# O corr() método do Pandas DataFrame  é usado para calcular a matriz.
# Por padrão, ele calcula o coeficiente de correlação de Pearson
correlation_mat = df_small.corr(method='pearson')
# Definindo as configurações do Gráfico
fig = plt.figure(figsize=(8,4))
eixo = fig.add_axes([0, 0, 1, 1])
# Usando o método heatmap para traçar a Matriz
# O parâmetro ' annot=True' exibe os valores do coeficiente de correlação em cada célula.
sns.heatmap(correlation_mat, annot = True, linewidth=0.5)
# Definindo titulo e labels do gráfico
eixo.set_title('Matriz de Correlação dos atributos', fontsize=15, pad=20)
eixo.set_xlabel ("características do núcleo da célula")
eixo.set_ylabel ("características do núcleo da célula")
df_candidato_cassacao['DS_MOTIVO_CASSACAO'].unique()
Para iniciar a exploração dos dados será apresentado um sumário estatístico
das variáveis com dados numéricos
df_candidato_cassacao.groupby('DS_MOTIVO_CASSACAO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
data_frame.groupby('DS_SIT_TOT_TURNO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount = sns.countplot(x="DS_SIT_TOT_TURNO",
data=data_frame).set_title("NR_CPF_CANDIDATO")
```

```
df_candidato_cassacao.shape
df_candidato_cassacao.groupby('DS_SITUACAO_CANDIDATURA')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
data_frame.groupby('ST_CANDIDATO_INSERIDO_URNA')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
idade_publico = Counter(df_candidato_cassacao['NR_IDADE_DATA_POSSE'])
#Plotagem da idade dos candidatos
plt.style.use('ggplot')
plt.bar(idade_publico.keys(), idade_publico.values())
plt.xlabel('Idade')
plt.title('Idade dos Candidatos')
plt.show()
idade_cal = Counter(df_candidato_cassacao['NR_PARTIDO'])
totalValor = np.array(df_candidato_cassacao['ANO_ELEICAO'])
df_candidato_cassacao.head()
#Criação de um dataframe final com candidatos eleitos
df_eleitos =
df_candidato_cassacao[~df_candidato_cassacao.DS_SIT_TOT_TURNO.isin(['#NULO#','2º
TURNO', 'NÃO ELEITO'])]
df_eleitos
df_eleitos_cassados = df_eleitos[~df_eleitos.DS_MOTIVO_CASSACAO.isnull()]
df_eleitos_cassados
#Criação de um dataframe final com candidatos eleitos
df_nao_eleitos =
df_candidato_cassacao[df_candidato_cassacao.DS_SIT_TOT_TURNO.isin(['#NULO#','2º
TURNO', 'NÃO ELEITO'])]
df_nao_eleitos
df_nao_eleitos_cassados = df_nao_eleitos[~df_nao_eleitos.DS_MOTIVO_CASSACAO.isnull()]
df_nao_eleitos_cassados
cassados = df_eleitos.loc[df_eleitos['DS_MOTIVO_CASSACAO'].notnull()]
#Contagem das opções da coluna DS_GENERO
genero_candidatos_cassados = Counter(cassados['DS_GENERO'])
```

```
genero_candidatos_cassados
cassados_nao_eleito =
df_nao_eleitos_cassados.loc[df_nao_eleitos_cassados['DS_MOTIVO_CASSACAO'].notnull()]
#Contagem das opções da coluna DS_GENERO
genero_candidatos_cassados_nao_eleito = Counter(cassados_nao_eleito['DS_GENERO'])
genero_candidatos_cassados_nao_eleito
# Distributions of the features
fig, ax = plt.subplots(3, 4, figsize=(20, 12))
sns.histplot(df_eleitos_cassados['NR_CANDIDATO'],kde=True, ax=ax[0, 0])
sns.histplot(df_eleitos_cassados['SQ_CANDIDATO'], kde=True, ax=ax[0,1])
sns.histplot(df_eleitos_cassados['CD_GRAU_INSTRUCAO'], ax=ax[0, 2])
sns.histplot(df_eleitos_cassados['CD_GENERO'], ax=ax[0, 3])
sns.histplot(df_eleitos_cassados['NR_IDADE_DATA_POSSE'], ax=ax[1, 0])
sns.histplot(df_eleitos_cassados['CD_CARGO'], ax=ax[1, 1])
sns.histplot(df_eleitos_cassados['SQ_CANDIDATO'], kde=True, ax=ax[1, 2])
sns.histplot(df_eleitos_cassados['CD_GRAU_INSTRUCAO'], kde=True, ax=ax[2, 0])
sns.histplot(df_eleitos_cassados['NR_PARTIDO'], ax=ax[2, 1])
sns.histplot(df_eleitos_cassados['ANO_ELEICAO'], ax=ax[2, 2])
plt.show()
# Distributions of the features
fig, ax = plt.subplots(3, 4, figsize=(20, 12))
sns.histplot(df_nao_eleitos_cassados['NR_IDADE_DATA_POSSE'], kde=True, ax=ax[0, 0])
sns.histplot(df_nao_eleitos_cassados['CD_GRAU_INSTRUCAO'], ax=ax[0, 1])
sns.histplot(df_nao_eleitos_cassados['NR_CANDIDATO'], ax=ax[0, 2])
sns.histplot(df_nao_eleitos_cassados['NR_PARTIDO'], ax=ax[0, 3])
sns.histplot(df_nao_eleitos_cassados['ANO_ELEICAO'], ax=ax[1, 0])
sns.histplot(df_nao_eleitos_cassados['CD_GENERO'], ax=ax[1, 1])
sns.histplot(df_nao_eleitos_cassados['CD_GRAU_INSTRUCAO'], kde=True, ax=ax[1, 2])
sns.histplot(df_nao_eleitos_cassados['ANO_ELEICAO'], kde=True, ax=ax[2, 0])
sns.histplot(df_nao_eleitos_cassados['NR_PARTIDO'], ax=ax[2, 1])
sns.histplot(df_nao_eleitos_cassados['ANO_ELEICAO'], ax=ax[2, 2])
plt.show()
```

```
df_eleitos_cassados.groupby('DS_MOTIVO_CASSACAO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount                    =              sns.countplot(x="DS_MOTIVO_CASSACAO",
data=df_eleitos_cassados).set_title("Motico Cassação Candidatos Eleitos")
df_nao_eleitos_cassados.groupby('DS_MOTIVO_CASSACAO')
['NR_CPF_CANDIDATO'].count().sort_values(ascending=False)
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
plt.xticks(rotation=90)
genderCount                    =              sns.countplot(x="DS_MOTIVO_CASSACAO",
data=df_nao_eleitos_cassados).set_title("Motico Cassação Candidatos Não Eleitos")
#Contagem das opções da coluna DS_GENERO
genero_candidatos_eleitos = Counter(df_eleitos['DS_GENERO'])
genero_candidatos_eleitos
sns.boxplot(x='ST_REELEICAO', y='NR_IDADE_DATA_POSSE', data=df_eleitos)
plt.title('Reeleição total')
plt.show()
sns.boxplot(x='ST_REELEICAO', y='NR_IDADE_DATA_POSSE', data=df_eleitos_cassados)
plt.title('Candidatos Eleitos Reeleição')
plt.show()
sns.boxplot(x='ST_REELEICAO', y='NR_IDADE_DATA_POSSE', data=df_nao_eleitos_cassados)
plt.title('Candidatos não Eleitos Reeleição')
plt.show()
df_candidato = df_candidato_cassacao.copy()
df_candidato
categorical_columns    =    [cname    for    cname    in    df_candidato.columns    if
df_candidato[cname].dtype == "object"]
```

```
categorical_columns
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount                    =          sns.countplot(x="DS_COR_RACA",
data=df_candidato).set_title("Gender_Count")
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount                    =          sns.countplot(x="DS_COR_RACA",
data=df_nao_eleitos_cassados).set_title("Gender_Count")
sns.set(style="darkgrid")      #style the plot background to become a grid
fig = plt.figure(figsize=(12,4))
eixo = fig.add_axes([0, 0, 1, 1])
genderCount                    =          sns.countplot(x="DS_COR_RACA",
data=df_eleitos_cassados).set_title("Gender_Count")
df_candidato.isnull().sum()
```

https://www.flai.com.br/juscudilio/parte-iii-como-utilizar-modelos-de-machine-learning-para-reduzir-o-churn/

Separando as variáveis numéricas das categóricas

Variáveis numéricas são aquelas variáveis que assumem valores numéricos, por exemplo a variável idade. As variáveis numéricas são classificadas como variáveis contínuas ou discreta.

As variáveis contínuas assumem valores na reta real, como a variável Salário Estimado. E as variáveis discretas são aquelas que assumem valores inteiros, como a variável número de produtos.

Variáveis categóricas são variáveis que não assumem valores numéricos. Por exemplo, a variável país.

As variáveis categóricas são classificadas como nominais e ordinais. As variáveis categóricas nominais são aquelas que não tem nenhuma ordem envolvida, por exemplo, a variável sexo e ordinais quando temos uma ordem envolvida, como a variável grau de escolaridade.

No pré processamento dos dados separamos as variáveis entre categóricas e numéricas, pois para cada tipo de variável utilizamos técnicas de processamento diferentes.

```
data_gd = pd.get_dummies(df_candidato, prefix_sep='_', drop_first=True)
data_gd.head(8)
# O corr() método do Pandas DataFrame  é usado para calcular a matriz.
# Por padrão, ele calcula o coeficiente de correlação de Pearson
correlation_mat = df_candidato.corr(method='pearson')
# Definindo as configurações do Gráfico
fig = plt.figure(figsize=(8,4))
eixo = fig.add_axes([0, 0, 1, 1])
# Usando o método heatmap para traçar a Matriz
# O parâmetro ' annot=True' exibe os valores do coeficiente de correlação em cada célula.
sns.heatmap(correlation_mat, annot = True, linewidth=0.5)
# Definindo titulo e labels do gráfico
eixo.set_title('Matriz de Correlação dos atributos', fontsize=15, pad=20)
eixo.set_xlabel ("características do núcleo da célula")
eixo.set_ylabel ("características do núcleo da célula")
data_gd.columns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
#Variáveis Continuas
x_cont =
['ST_REELEICAO','DS_CARGO','DS_ESTADO_CIVIL','DS_SIT_TOT_TURNO','VR_DESPESA_MAX_
CAMPANHA','NR_IDADE_DATA_POSSE','SG_PARTIDO','DS_CARGO','TP_ABRANGENCIA']

#Variáveis Categóricas
x_cat = list(set(data_gd) - set(x_cont))
```

```
x_dummies = data_gd[x_cat]
data_gd
##Substituindo a variável sexo para 0 e 1
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_gd['DS_SIT_TOT_TURNO'] = le.fit_transform(data_gd['DS_SIT_TOT_TURNO'])
data_gd.head(10)
#https://datagy.io/pandas-get-dummies/
x_final = pd.get_dummies(data = data_gd,
                                                                    columns =
['SG_UF','DS_SIT_TOT_TURNO','DS_GRAU_INSTRUCAO','SG_PARTIDO','DS_CARGO','TP_ABR
ANGENCIA'],
                         prefix_sep='_',
                         dummy_na=False,
                         sparse=False,
                         drop_first=False,
                         dtype=None
                        )
x_final
x_dummies.head(8)
x_dummies.columns
# Distributions of the features
fig, ax = plt.subplots(3, 4, figsize=(20, 12))
sns.histplot(df_candidato_cassacao['NR_IDADE_DATA_POSSE'], kde=True, ax=ax[0, 0])
sns.histplot(df_candidato_cassacao['CD_GRAU_INSTRUCAO'], ax=ax[0, 1])
sns.histplot(df_candidato_cassacao['NR_CANDIDATO'], ax=ax[0, 2])
sns.histplot(df_candidato_cassacao['NR_PARTIDO'], ax=ax[0, 3])
sns.histplot(df_candidato_cassacao['ANO_ELEICAO'], ax=ax[1, 0])
sns.histplot(df_candidato_cassacao['CD_GENERO'], ax=ax[1, 1])
sns.histplot(df_candidato_cassacao['CD_GRAU_INSTRUCAO'], kde=True, ax=ax[1, 2])
sns.histplot(df_candidato_cassacao['ANO_ELEICAO'], kde=True, ax=ax[2, 0])
sns.histplot(df_candidato_cassacao['NR_PARTIDO'], ax=ax[2, 1])
```

```
sns.histplot(df_candidato_cassacao['CD_CARGO'], ax=ax[2, 2])
plt.show()
<b> DADOS </b>
df = x_final.copy()
df.columns
df
X_df = df[['NR_PARTIDO', 'NR_PARTIDO','CD_CARGO','NR_IDADE_DATA_POSSE','ANO_ELEICAO']]
Y_df = df['ANO_ELEICAO']

Xdummies_df = pd.get_dummies(X_df)
Ydummies_df = Y_df

X = Xdummies_df.values
Y = Ydummies_df.values

porcentagem_de_treino = 0.8
porcentagem_de_teste = 0.1

tamanho_de_treino = int(porcentagem_de_treino * len(Y))
tamanho_de_teste = int(porcentagem_de_teste * len(Y))
tamanho_de_validacao = len(Y) - tamanho_de_treino - tamanho_de_teste

treino_dados = X[:tamanho_de_treino]
treino_marcacoes = Y[:tamanho_de_treino]

fim_de_treino = tamanho_de_treino + tamanho_de_teste

teste_dados = X[tamanho_de_treino:fim_de_treino]
teste_marcacoes = Y[tamanho_de_treino:fim_de_treino]

validacao_dados = X[fim_de_treino:]
```

```
validacao_marcacoes = Y[fim_de_treino:]
from sklearn.metrics import confusion_matrix, classification_report
from pickle import dump, load
def fit_and_predict(nome, modelo, X_train, Y_train, teste_dados, teste_marcacoes):
    modelo.fit(X_train, Y_train)

    #Testa para ver a sua veracidade
    predictions = modelo.predict(teste_dados)
    print(confusion_matrix(teste_marcacoes, predictions))
    print(classification_report(teste_marcacoes, predictions))

    #Salva para uso futuro
    filename = 'binary_class_model.sav'
    #dump(modelo, open(filename, 'wb'))

    #Le o arquivo do disco
    #loaded_model = load(open('binary_class_model.sav', 'rb'))

    resultado = modelo.predict(teste_dados)
    acertos = resultado == teste_marcacoes
    total_de_acertos = sum(acertos)
    total_de_elementos = len(teste_dados)
    taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos
    msg = "Taxa de acerto do algoritmo {0}: {1}".format(nome, taxa_de_acerto)
    print(msg)
    return taxa_de_acerto


def teste_real(modelo, validacao_dados, validacao_marcacoes):
    resultado = modelo.predict(validacao_dados)
    acertos = resultado == validacao_marcacoes
    total_de_acertos = sum(acertos)
```

```
    total_de_elementos = len(validacao_marcacoes)
    taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos
        msg = "Taxa de acerto do vencedor entre os dois algoritmos no mundo real: {0}".format(taxa_de_acerto)
    print(msg)
resultados = {}
#Ajuste do Modelo KNN - https://github.com/alura-cursos/machine-learning-introducao-a-classificacao-2/blob/master/situacao_do_cliente.py
from sklearn.neighbors import KNeighborsClassifier #classificador
knn = KNeighborsClassifier()
knn.fit(treino_dados, treino_marcacoes)
resultado_knn = knn.predict(teste_dados)

resultado_knn
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
modeloOneVsRest = OneVsRestClassifier(LinearSVC(random_state = 0))
resultadoOneVsRest = fit_and_predict("OneVsRest", modeloOneVsRest, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)
resultados[resultadoOneVsRest] = modeloOneVsRest
from sklearn.multiclass import OneVsOneClassifier
modeloOneVsOne = OneVsOneClassifier(LinearSVC(random_state=0))
resultadoOneVsOne = fit_and_predict("OneVsOne", modeloOneVsOne, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)
resultados[resultadoOneVsOne] = modeloOneVsOne
from sklearn.naive_bayes import MultinomialNB
modeloMultinomial = MultinomialNB()
resultadoMultinomial = fit_and_predict("MultinomialNB", modeloMultinomial, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)
resultados[resultadoMultinomial] = modeloMultinomial
from sklearn.ensemble import AdaBoostClassifier
modeloAdaBoost = AdaBoostClassifier()
```

```
resultadoAdaBoost = fit_and_predict("AdaBoostClassifier", modeloAdaBoost, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)
resultados[resultadoAdaBoost] = modeloAdaBoost
print (resultados)
maximo = max(resultados)
vencedor = resultados[maximo]
print (vencedor)

resultado = vencedor.predict(validacao_dados)
acertos = (resultado == validacao_marcacoes)

total_de_acertos = sum(acertos)
total_de_elementos = len(validacao_marcacoes)
taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos

msg = "Taxa de acerto do vencedor entre os dois algoritmos no mundo real: {0}".format(taxa_de_acerto)
print(msg)
print(resultados)
maximo = max(resultados)
vencedor = resultados[maximo]
print("VENCEDOR:", vencedor)
# precisa fazer o FIT pois o FIT foi excluido pois não fazemos mais o treino
vencedor.fit(treino_dados, treino_marcacoes)

teste_real(vencedor, validacao_dados, validacao_marcacoes)

print("Elementos testados: %d " % len(validacao_dados))
#Salva para uso futuro
filename = 'binary_class_model.sav'
dump(resultadoAdaBoost, open(filename, 'wb'))
```

```
#Le o arquivo do disco
loaded_model = load(open('binary_class_model.sav', 'rb'))
import seaborn as sns # statistical visualization
(df_candidato_cassacao.isnull().sum() / df_candidato_cassacao.shape[0] * 100).sort_values(ascending=False)
#quantidade de valores únicos por cada coluna.
df_candidato_cassacao.nunique().sort_values()
# describing categorical features
df_candidato_cassacao.describe(include=['O'])
# plotting this correlation between features
fig, ax = plt.subplots(figsize=(17, 17))
sns.heatmap(data=df_candidato_cassacao.corr().round(2), annot=True, cmap="PiYG", ax=ax)
# pair plot diagnosis + 10 features (mean)
sns.pairplot(data=df_candidato_cassacao.iloc[:,0:11], hue='NR_PARTIDO', diag_kind='kde')
# plotting this correlation between features
fig, ax = plt.subplots(figsize=(17, 17))
sns.heatmap(data=df_candidato.corr().round(2), annot=True, cmap="PiYG", ax=ax)
df.values
```

https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/67/292/544-1?inline=1

Segundo o exemplo anterior, o conjunto de dados está aparentemente completo. Porém, tal resposta não é suficiente para descartar a hipótese de que existem dados ausentes. Para uma melhor averiguação, pode-se resumir cada coluna no DataFrame booleano somando os valores False=0 e True=1. Tal processo retorna o número total de valores ausentes. Também pode-se dividir cada valor pelo número total de linhas no conjunto de dados, resultando na porcentagem de tais ausências, conforme o exemplo a seguir.

```
[4]: # Calcula o total e a % de valores ausentes
num_ausentes = df.isna().sum()
porc_ausentes = df.isna().sum() * 100 / len(df)
# DataFrame com as informações computadas acima
df_ausentes = pd.DataFrame({
    'Coluna': df.columns,
    'Dados ausentes': num_ausentes,
    'Porcentagem': porc_ausentes
})
df_ausentes
```

[4]:

	Coluna	Dados ausentes	Porcentagem
artist_id	artist_id	0	0.00
name	name	0	0.00
followers	followers	0	0.00
popularity	popularity	62	9.92
genres	genres	40	6.40
image_url	image_url	10	1.60