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

Henrique Sander de Lima Fernandes

ANÁLISE DE IMPACTO DO NOME DE URNA NAS ELEIÇÕES DE 2018

Belo Horizonte
2022

Henrique Sander de Lima Fernandes

ANÁLISE DE IMPACTO DO NOME DE URNA NAS ELEIÇÕES DE 2018

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

Participar do processo eleitoral brasileiro enquanto candidato é uma prerrogativa prevista na Constituição Federal aos de nacionalidade brasileira e em o pleno exercício dos direitos políticos. Dentre os critérios previstos ainda constam o alistamento eleitoral, domicílio eleitoral na circunscrição e filiação partidária, desde que o candidato possua a idade mínima exigida para o cargo pleiteado (Constituição federal/1988 título II - dos direitos e garantias fundamentais - capítulo IV - dos direitos políticos - art. 14.). Se este candidato for militar alistável a legislação ainda orienta que caso possua menos de dez anos de serviço, deverá afastar-se da atividade, e se contar mais de dez anos de serviço, será agregado pela autoridade superior e, se eleito, passará automaticamente, no ato da diplomação, para a inatividade. (Constituição federal/1988 título II - dos direitos e garantias fundamentais - capítulo IV - dos direitos políticos - art. 14.)

Esse processo eleitoral ainda estabelece a possibilidade de o candidato escolher o nome que irá constar na urna eletrônica. Esse nome, que pode ou não coincidir com o nome do registro civil, é utilizado pelo candidato durante todo o período eleitoral e o identifica na urna eletrônica. Dentre as opções para essa escolha estão apelido ou nome pelo qual o candidato é mais conhecido, desde que não se estabeleça dúvida quanto a sua identidade, não atente contra o pudor e não seja ridículo ou irreverente (Resolução do TSEn°23609/2019 – Art.25). Os nomes de urna integram as estratégias que os candidatos dispõem para atrair e persuadir os eleitores, além se facilitar o reconhecimento e a identificação.

Nesse sentido alguns candidatos optam por agregar ao nome de urna suas patentes e títulos. Em levantamento anterior realizado referente a eleição municipal verificou que o acréscimo mais utilizado é o de professor ou professora. As denominações ligadas à área de saúde também apareceram de forma significativa com sendo doutor/doutora os mais utilizados, seguidos de “da saúde” e enfermeiros. Nesse levantamento, os cargos religiosos também estiveram entre os mais utilizados com as denominações de “pastor” e “irmão”

sendo as mais prevalentes. Outra titulação muito utilizada foi em relação as denominações militares em que o título de “sargento” se sobressaiu.

Em outro estudo denominado “O aumento da inclusão de postos e graduações militares em nomes de urna como um indicador de mudanças no imaginário social brasileiro” realizado acerca do assunto e com enfoque na eleição de 2018 foi observado que o surgimento de discursos favoráveis ao regime militar encorajou os militares a utilizarem a identificação da patente na campanha eleitoral. Impulsionados ainda por temas como a segurança pública e o combate à corrupção, esses candidatos acabaram se destacando em todos os estados e para todos os cargos em disputa. Os dados também revelaram que, até 2014, a maior parte dos militares optavam por não incluir postos e graduações em nomes de urna. Contudo em 2018, houve uma ruptura, uma mudança brusca nesse padrão de comportamento e, pela primeira vez em 20 anos (1998 a 2018), houve mais militares que optaram pela inclusão.

1.2. O problema proposto

Tendo em vista o contexto histórico descrito sobre a eleição de 2018 e a relevância histórica e social dos processos eleitorais, torna-se importante verificar o impacto que o uso de títulos agregados ao nome de urna pode causar na eleição dos candidatos que optam por essa estratégia.

Para responder as questões suscitadas neste trabalho, foram utilizados conjunto de dados do Portal de Dados Abertos do TSE contendo informações sobre os nomes de urna dos candidatos a cargos de deputado estadual, deputado federal e senador na eleição nacional de 2018. O resultado da candidatura (se eleito ou não) foi utilizado como desfecho sendo determinante para que as análises de impacto pudessem ser desenvolvidas. Além disso, também foram utilizadas informações que agregassem a análise como o investimento declarado, grau de instrução, gênero, estado civil e raça. Essas informações foram relevantes para que a comparação pudesse ser mais embasada e assertiva.

1.3. Objetivos

Medir e comparar o desempenho de diferentes grupos caracterizados por títulos agregados ao nome de urna do candidato e por fim implementar um modelo preditivo que evidencie a tendência para as próximas eleições.

2. Coleta de Dados

Os conjuntos de dados se encontram no Portal de Dados Abertos do TSE (https://dadosabertos.tse.jus.br/). De acordo com o próprio site "O portal disponibiliza à sociedade os dados gerados ou custodiados pelo TSE, de forma a garantir o acesso a informações e aprimorar a cultura de transparência. Ele substitui o antigo Repositório de Dados Eleitorais, descontinuado em janeiro de 2022. Os dados aqui disponíveis podem ser livremente acessados, utilizados, modificados e compartilhados por qualquer pessoa, com vistas à geração de novas informações e iniciativas da sociedade que busquem estimular o controle social e contribuir com a melhoria da gestão pública."

O primeiro conjunto de dados pode ser baixado através do link (https://cdn.tse.jus.br/estatistica/sead/odsele/consulta_cand/consulta_cand_2018.zip). Este conjunto traz informações dos candidatos em relação as eleições de 2018.

Coluna / Tipo	**Descrição**
DT_GERACAO	Data da extração dos dados para geração do arquivo.
HH_GERACAO	Hora da extração dos dados para geração do arquivo com base no horário de Brasília.
ANO_ELEICAO	Ano de referência da eleição para geração do arquivo. Observação: Para eleições suplementares o ano de referência da eleição é o da eleição ordinária correspondente. Por exemplo: Em 2016 houve eleições ordinárias. Após a data desta eleição ordinária e antes da próxima, houve eleições suplementares em 2017, 2018 e 2019. As informações destas eleições suplementares estarão divulgadas no arquivo gerado para as Eleições 2016.
CD_TIPO_ELEICAO	Código do tipo de eleição. Pode assumir os valores: 1 - Eleição Suplementar, 2 - Eleição Ordinária e 3 - Consulta Popular.

NM_TIPO_ELEICAO	Nome do tipo de eleição.
NR_TURNO	Número do turno da eleição.
CD_ELEICAO	Código único da eleição no âmbito da Justiça Eleitoral. Observação: Este código é único por eleição e por turno, ou seja, cada turno possui seu código de eleição.
DS_ELEICAO	Descrição da eleição.
DT_ELEICAO	Data em que ocorreu a eleição.
TP_ABRANGENCIA	Abrangência da eleição. Pode assumir os valores: Municipal, Estadual e Federal.
SG_UF	Sigla da Unidade da Federação em que ocorreu a eleição.
SG_UE	Sigla da Unidade Eleitoral em que o candidato concorre na eleição.
NM_UE	Nome da Unidade Eleitoral do candidato.
CD_CARGO	Código do cargo ao qual o candidato concorre na eleição.
DS_CARGO	Cargo ao qual o candidato concorre na eleição.
SQ_CANDIDATO	Número sequencial do candidato, gerado internamente pelos sistemas eleitorais para cada eleição.
NR_CANDIDATO	Número do candidato na urna.
NM_CANDIDATO	Nome completo do candidato.
NM_URNA_CANDIDATO	Nome do candidato que aparece na urna. (CONTA MAIS)
NM_SOCIAL_CANDIDATO	Nome social do candidato.
NR_CPF_CANDIDATO	Número do CPF do candidato.
NM_EMAIL	Endereço de e-mail do candidato.
CD_SITUACAO_CANDIDATURA	Código da situação do registro de candidatura do candidato.
DS_SITUACAO_CANDIDATURA	Situação do registro da candidatura do candidato.
CD_DETALHE_SITUACAO_CAND	Código do detalhe da situação do registro de candidatura do candidato.
DS_DETALHE_SITUACAO_CAND	Detalhe da situação do registro de candidatura do candidato que especifica o motivo pelo qual a candidatura foi julgada como 'Apta' ou 'Inapta'

TP_AGREMIACAO	Tipo de agremiação da candidatura do candidato, ou seja, forma como o candidato concorrerá nas eleições. Pode assumir os valores: Coligação (quando o candidato concorre por coligação) e Partido Isolado (quando o candidato concorre somente pelo partido).
NR_PARTIDO	Número do partido de origem do candidato. Mesmo que o candidato participe de uma coligação, este número é o número do seu partido de origem.
SG_PARTIDO	Sigla do partido de origem do candidato.
NM_PARTIDO	Nome do partido de origem do candidato.
SQ_COLIGACAO	Sequencial da coligação da qual o candidato pertence, gerado pela Justiça Eleitoral.
NM_COLIGACAO	Nome da coligação da qual o candidato pertence.
DS_COMPOSICAO_COLIGACAO	Composição da coligação da qual o candidato pertence.
CD_NACIONALIDADE	Código da nacionalidade do candidato.
DS_NACIONALIDADE	Nacionalidade do candidato.
SG_UF_NASCIMENTO	Sigla da Unidade da Federação de nascimento do candidato.
CD_MUNICIPIO_NASCIMENTO	Código de identificação do município de nascimento do candidato.
NM_MUNICIPIO_NASCIMENTO	Nome do município de nascimento do candidato.
DT_NASCIMENTO	Data de nascimento do candidato.
NR_IDADE_DATA_POSSE	Idade do candidato na data da posse.
NR_TITULO_ELEITORAL_CANDIDATO	Número do título eleitoral do candidato.
CD_GENERO	Código do gênero do candidato.
DS_GENERO	Gênero do candidato.
CD_GRAU_INSTRUCAO	Código do grau de instrução do candidato.
DS_GRAU_INSTRUCAO	Grau de instrução do candidato.
CD_ESTADO_CIVIL	Código do estado civil do candidato.
DS_ESTADO_CIVIL	Estado civil do candidato.
CD_COR_RACA	Código da cor/raça do candidato. (autodeclaração)

DS_COR_RACA	Cor/raça do candidato. (autodeclaração)
CD_OCUPACAO	Código da ocupação do candidato.
DS_OCUPACAO	Ocupação do candidato.
VR_DESPESA_MAX_CAMPANHA	Valor máximo, em reais, de despesas de campanha declarada pelo partido para aquele candidato.
CD_SIT_TOT_TURNO	Código da situação de totalização do candidato, naquele turno da eleição, após a totalização dos votos.
DS_SIT_TOT_TURNO	Situação de totalização do candidato, naquele turno da eleição, após a totalização dos votos.
ST_REELEICAO	Indica se o candidato está concorrendo ou não à reeleição.
ST_DECLARAR_BENS	Indica se o candidato tem ou não bens a declarar.
NR_PROTOCOLO_CANDIDATURA	Número do protocolo de registro de candidatura do candidato.
NR_PROCESSO	Número do processo de registro de candidatura do candidato.
CD_SITUACAO_CANDIDATO_PLEITO	Código da situação da candidatura no dia do Pleito.
DS_SITUACAO_CANDIDATO_PLEITO	Situação da candidatura no dia do Pleito.
CD_SITUACAO_CANDIDATO_URNA	Código da situação da candidatura na urna.
DS_SITUACAO_CANDIDATO_URNA	Situação da candidatura na urna.
ST_CANDIDATO_INSERIDO_URNA	Informa se o candidato foi inserido na urna eletrônica.

O segundo conjunto de dados pode ser baixado através do link (https://cdn.tse.jus.br/estatistica/sead/odsele/prestacao_contas/prestacao_de_contas_eleitorais_candidatos_2018.zip)

Este conjunto traz informações sobre despesas contratadas e despesas pagas pelos candidatos participantes das eleições de 2018.

Coluna / Tipo	**Descrição**
DT_GERACAO	Data de geração das informações (data da extração dos da-

	dos).
HH_GERACAO	Hora de geração das informações (hora da extração dos dados) - Horário de Brasília.
ANO_ELEICAO	Ano da eleição referente ao ano eleitoral de pesquisa.
CD_TIPO_ELEICAO	Código do tipo de eleição.
NM_TIPO_ELEICAO	Nome do tipo de eleição.
CD_ELEICAO	Código da eleição.
DS_ELEICAO	Descrição da eleição.
DT_ELEICAO	Data em que ocorreu a eleição.
ST_TURNO	Indica prestação de contas para 2º turno.
TP_PRESTACAO_CONTAS	Tipo de entrega da prestação de contas.
DT_PRESTACAO_CONTAS	Data de entrega da prestação de contas junto ao TSE.
SQ_PRESTADOR_CONTAS	Sequencial de identificação do prestador de contas junto ao TSE.
SG_UF	Sigla da unidade da federação de abrangência do prestador de contas.
SG_UE	Sigla da Unidade Eleitoral do candidato.
NM_UE	Nome de Unidade Eleitoral do candidato.
NR_CNPJ_PRESTADOR_CONTA	Numero do CNPJ do prestador de contas.
CD_CARGO	Código do cargo do candidato prestador de contas.
DS_CARGO	Descrição do cargo do candidato prestador de contas.
SQ_CANDIDATO	Sequencial do candidato prestador de contas.
NR_CANDIDATO	Número do candidato prestador de contas.
NM_CANDIDATO	Nome completo do candidato.
NR_CPF_CANDIDATO	CPF do candidato registrado na Justiça Eleitoral.
NR_CPF_VICE_CANDIDATO	CPF do candidato à vice/suplente do titular, se houver.
NR_PARTIDO	Número do partido do candidato.
SG_PARTIDO	Sigla do partido do candidato.

NM_PARTIDO	Nome do partido do candidato.
CD_TIPO_FORNECEDOR	Código de identificação do tipo de fornecedor informada pelo prestador de contas em relação à despesa.
DS_TIPO_FORNECEDOR	Descrição do tipo de fornecedor informada pelo prestador de contas em relação à despesa.
CD_CNAE_FORNECEDOR	Código CNAE do fornecedor de bens e/ou serviços, se pessoa jurídica.
DS_CNAE_FORNECEDOR	Descrição do CNAE (Código do Setor Econômico) do fornecedor de bens e/ou serviços, se pessoa jurídica.
NR_CPF_CNPJ_FORNECEDOR	Número do CPF/CNPJ do fornecedor de bens e/ou serviços informada pelo prestador de contas em relação à despesa.
NM_FORNECEDOR	Nome do fornecedor de bens e/ou serviços declarado a Justiça Eleitoral, informada pelo prestador de contas em relação à despesa.
NM_FORNECEDOR_RFB	Nome do fornecedor cadastrado na Receita Federal do Brasil, informada pelo prestador de contas em relação à despesa.
CD_ESFERA_PART_FORNECEDOR	Código do tipo de esfera partidária do fornecedor, quando fornecedor 'Órgão partidário'.
DS_ESFERA_PART_FORNECEDOR	Descrição do tipo de esfera partidária do fornecedor.
SG_UF_FORNECEDOR	Sigla da unidade da federação do fornecedor, quando fornecedor candidato ou órgão partidário.
CD_MUNICIPIO_FORNECEDOR	Código do município do fornecedor, quando a esfera partidária do fornecedor for municipal.
NM_MUNICIPIO_FORNECEDOR	Descrição do município do fornecedor, quando a esfera partidária do fornecedor for municipal.
SQ_CANDIDATO_FORNECEDOR	Sequencial do candidato fornecedor, quando fornecedor candidato.
NR_CANDIDATO_FORNECEDOR	Número do candidato declarado pelo prestador de contas, quando fornecedor candidato.
CD_CARGO_FORNECEDOR	Código do cargo do candidato declarado pelo prestador de contas, quando fornecedor.
DS_CARGO_FORNECEDOR	Descrição do cargo do candidato declarado pelo prestador de contas, quando fornecedor.

NR_PARTIDO_FORNECEDOR	Número do partido declarado pelo prestador de contas, quando fornecedor candidato ou órgão partidário.
SG_PARTIDO_FORNECEDOR	Sigla do partido declarado pelo prestador de contas, quando fornecedor candidato ou órgão partidário.
NM_PARTIDO_FORNECEDOR	Nome do partido declarado pelo prestador de contas, quando fornecedor candidato ou órgão partidário.
DS_TIPO_DOCUMENTO	Tipo de documento.
NR_DOCUMENTO	Número de documento que comprove a despesa.
SQ_DESPESA	Sequencial de identificação do registro da despesa declarada pelo prestador de contas.
DT_DESPESA	Data da despesa declarada à Justiça Eleitoral.
DS_DESPESA	Descrição do gasto no elenco de aplicações informada pelo prestador de contas em relação à despesa.
VR_DESPESA_CONTRATADA	Valor da despesa contratada em Reais (R$), informada pelo prestador de contas em relação à despesa.

3. Processamento/Tratamento de Dados

O processamento se inicia com a importação do primeiro dataset csv (consulta_cand_2018_BRASIL) utilizando a biblioteca pandas, função read_csv.

```
df_candidatos = pd.read_csv("consulta_cand_2018_BRASIL.csv",
    encoding = "Latin 1", sep = ";", decimal = ',', low_memory=False)
```

Além do caminho do próprio arquivo, foram utilizados os parâmetros encoding contemplando os caracteres especiais da língua portuguesa e low_memory permitindo maior utilização de memória em favor de maior assertividade na inferência do tipo de dado. Nesta etapa obtivemos 29.180 registros, sem ausência nas 25 colunas.

Em seguida, por meio da função isin, as colunas DS_CARGO, DS_SITUACAO_CANDIDATURA e DS_SIT_TOT_TURNO foram filtradas para contemplar candidatos do legislativo, com candidatura apta, e totalização de votos.

```
df_candidatos = df_candidatos[
    df_candidatos.DS_CARGO.isin(['DEPUTADO ESTADUAL','DEPUTADO FEDERAL', 'SENADOR'])]

df_candidatos = df_candidatos[
    df_candidatos.DS_SITUACAO_CANDIDATURA.isin(['APTO'])]

df_candidatos = df_candidatos[
    df_candidatos.DS_SIT_TOT_TURNO.isin(['ELEITO', 'ELEITO POR MÉDIA','ELEITO POR QP','NÃO ELEITO'])]

df_candidatos = df_candidatos[
    ~df_candidatos.DS_COR_RACA.isin(['NÃO DIVULGÁVEL'])]

df_candidatos = df_candidatos[['SQ_CANDIDATO','NM_URNA_CANDIDATO','NR_IDADE_DATA_POSSE','DS_GENERO',
    'DS_GRAU_INSTRUCAO', 'DS_ESTADO_CIVIL','DS_COR_RACA','DS_SIT_TOT_TURNO']]

df_candidatos = df_candidatos.set_index('SQ_CANDIDATO')
df_candidatos.info()
```

✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6230 entries, 180000614026 to 80000611871
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   NM_URNA_CANDIDATO    6230 non-null   object
 1   NR_IDADE_DATA_POSSE  6230 non-null   float64
 2   DS_GENERO            6230 non-null   object
 3   DS_GRAU_INSTRUCAO    6230 non-null   object
 4   DS_ESTADO_CIVIL      6230 non-null   object
 5   DS_COR_RACA          6230 non-null   object
 6   DS_SIT_TOT_TURNO     6230 non-null   object
dtypes: float64(1), object(6)
memory usage: 389.4+ KB
```

Nesta mesma etapa também foi definido um índice para o DataFrame utilizando a função set_index passando como parâmetro a coluna a ser utilizada como índice, que no caso é a SQ_CANDIDATO, um número único de cada candidato.

Na sequência o segundo dataset foi importado (despesas_contratadas_candidatos_2018_BRASIL) utilizando a biblioteca pandas, função read_csv.

```
df_despesas = pd.read_csv("despesas_contratadas_candidatos_2018_BRASIL.csv",
    encoding = "Latin 1", sep = ";", decimal = ',', low_memory=False)
```

Como este dataset lista despesas discriminadas de cada candidato, optou-se por agrupar somando todas as despesas do mesmo, restando apenas um registro por candidato permitindo assim o join com o primeiro dataset.

```
df_despesas = df_despesas[['SQ_CANDIDATO','VR_DESPESA_CONTRATADA']]
df_despesas = df_despesas.groupby(['SQ_CANDIDATO']).sum()
df_despesas.info()
```

✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19370 entries, 10000600001 to 280000629808
Data columns (total 1 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   VR_DESPESA_CONTRATADA  19370 non-null  float64
dtypes: float64(1)
memory usage: 302.7 KB
```

```
df_candidatos = df_candidatos.join(df_despesas)
```

✓ 0.4s

Neste ponto restam sete colunas no DataFrame sendo elas: nome de urna do candidato, idade na posse, gênero, grau de instrução, estado civil, cor/raça, situação na totalização dos votos e total da despesa contratada. Apenas as colunas de idade e despesa são quantitativas. Considerando a intenção de produzir um modelo de aprendizado de máquina, aplicamos o dummy encoding para agregar as informações qualitativas no mesmo. As variáveis dummies, também conhecidas como variáveis indicadoras, assumem valores binários indicando se um elemento possui ou não determinada característica.

A primeira coluna, DS_SIT_TOT_TURNO foi tratada utilizando uma função lambda para gerar um mapa onde toda situação de totalização diferente de 'NÃO ELEITO' assume o valor 1 (true) e na situação do não eleito assume o valor 0 (false). Esta função gera uma lista que por sua vez passa a ser a coluna 'ELEITO'. A função lambda foi utilizada ao invés da pandas.get_dummies por conta do interesse em manter as 4 situações de totalização em uma única coluna, caso contrário seriam geradas 4 colunas para representar tal situação.

```
df_candidatos['ELEITO'] = list(map(lambda x:
int('NÃO ELEITO' not in x),
df_candidatos['DS_SIT_TOT_TURNO']))
```

Já no caso das colunas gênero, estado civil e raça, optou-se pela utilização da função pandas.get_dummies. Com isso a coluna gênero desdobrou-se em duas, sendo elas 'GENERO_FEMININO' e 'GENERO_MASCULINO'; da mesma forma estado civil gerou outras cinco colunas e por fim a coluna raça que produziu mais cinco colunas.

```
df_genero = pd.get_dummies(df_candidatos.DS_GENERO, prefix='GENERO')
df_candidatos = df_candidatos.join(df_genero)

df_estado_civil = pd.get_dummies(df_candidatos.DS_ESTADO_CIVIL, prefix='ESTADO_CIVIL')
df_candidatos = df_candidatos.join(df_estado_civil)

df_raca = pd.get_dummies(df_candidatos.DS_COR_RACA, prefix='RACA')
df_candidatos = df_candidatos.join(df_raca)
```

```
0   NM_URNA_CANDIDATO                       6230 non-null   object
1   NR_IDADE_DATA_POSSE                     6230 non-null   float64
2   DS_GENERO                               6230 non-null   object
3   DS_GRAU_INSTRUCAO                       6230 non-null   int64
4   DS_ESTADO_CIVIL                         6230 non-null   object
5   DS_COR_RACA                             6230 non-null   object
6   DS_SIT_TOT_TURNO                        6230 non-null   object
7   VR_DESPESA_CONTRATADA                   6230 non-null   float64
8   ELEITO                                  6230 non-null   int64
9   GENERO_FEMININO                         6230 non-null   uint8
10  GENERO_MASCULINO                        6230 non-null   uint8
11  ESTADO_CIVIL_CASADO(A)                  6230 non-null   uint8
12  ESTADO_CIVIL_DIVORCIADO(A)              6230 non-null   uint8
13  ESTADO_CIVIL_SEPARADO(A) JUDICIALMENTE  6230 non-null   uint8
14  ESTADO_CIVIL_SOLTEIRO(A)                6230 non-null   uint8
15  ESTADO_CIVIL_VIÚVO(A)                   6230 non-null   uint8
16  RACA_AMARELA                            6230 non-null   uint8
17  RACA_BRANCA                             6230 non-null   uint8
18  RACA_INDÍGENA                           6230 non-null   uint8
19  RACA_PARDA                              6230 non-null   uint8
20  RACA_PRETA                              6230 non-null   uint8
```

Para o grau de instrução, foi utilizada uma abordagem de codificação ordinal através da geração de um dicionário ordenando o grau de instrução. Em seguida esse dicionário foi passado como parâmetro na função map.

```
recode = {'LÊ E ESCREVE': 1,
    'ENSINO FUNDAMENTAL INCOMPLETO': 2,
    'ENSINO FUNDAMENTAL COMPLETO': 3,
    'ENSINO MÉDIO INCOMPLETO': 4,
    'ENSINO MÉDIO COMPLETO': 5,
    'SUPERIOR INCOMPLETO': 6,
    'SUPERIOR COMPLETO': 7}
df_candidatos['DS_GRAU_INSTRUCAO'] = df_candidatos['DS_GRAU_INSTRUCAO'].map(recode)
```

Em função do objetivo de analizar o desempenho de diferentes grupos caracterizados por títulos agregados ao nome de urna, foi necessário criar variáveis indicadoras da presença de título. Para tal foi implementada uma função lambda para gerar um mapa que indique o valor 1 (true) sempre que identificar a presença de uma determinada sequência de caracteres no nome de urna do candidato. A presença destes caracteres classifica o candidato como doutor, militar, professor ou figura religiosa.

```
df_candidatos['DOUTOR'] = list(map(lambda x:
int('DOUTOR' in x or
'DR ' in x or
'DR.' in x or
'DRA.' in x),
df_candidatos['NM_URNA_CANDIDATO']))
```

```
df_candidatos['MILITAR'] = list(map(lambda x:
int('AGENTE' in x or
'AGT.' in x or
'BOMBEIRO' in x or
'CABO' in x or
'CAP.' in x or
'CAPITAO' in x or
'CB' in x or
'CORONEL' in x or
'DELE.' in x or
'DELEGAD' in x or
'MAJOR' in x or
'POLICIAL' in x or
'SARG.' in x or
'SARGENTO' in x or
'SGT' in x or
'SOLDADO' in x or
'SUB TENENTE' in x or
'SUBOFICIAL' in x or
'SUBTEN' in x or
'TEN.' in x or
'TENENTE' in x),
df_candidatos['NM_URNA_CANDIDATO']))
```

```
df_candidatos['PROFESSOR'] = list(map(lambda x:
int('PROF' in x), df_candidatos['NM_URNA_CANDIDATO']))

df_candidatos['RELIGIOSO'] = list(map(lambda x:
int('AP.' in x or
'APOSTOL' in x or
'APÓSTOLO' in x or
'BISP' in x or
'DIACONO' in x or
'MISSIONÁRI' in x or
'MISSIONARI' in x or
'PADRE' in x or
'PAI ' in x or
'PASTOR' in x or
'PR ' in x or
'PR.' in x or
'REV.' in x),
df_candidatos['NM_URNA_CANDIDATO']))
```

Com isso temos o DataFrame df_candidatos preparado para fundar as próximas etapas.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6230 entries, 180000614026 to 80000611871
Data columns (total 25 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   NM_URNA_CANDIDATO                      6230 non-null   object
 1   NR_IDADE_DATA_POSSE                    6230 non-null   float64
 2   DS_GENERO                              6230 non-null   object
 3   DS_GRAU_INSTRUCAO                      6230 non-null   int64
 4   DS_ESTADO_CIVIL                        6230 non-null   object
 5   DS_COR_RACA                            6230 non-null   object
 6   DS_SIT_TOT_TURNO                       6230 non-null   object
 7   VR_DESPESA_CONTRATADA                  6230 non-null   float64
 8   ELEITO                                 6230 non-null   int64
 9   GENERO_FEMININO                        6230 non-null   uint8
 10  GENERO_MASCULINO                       6230 non-null   uint8
 11  ESTADO_CIVIL_CASADO(A)                 6230 non-null   uint8
 12  ESTADO_CIVIL_DIVORCIADO(A)             6230 non-null   uint8
 13  ESTADO_CIVIL_SEPARADO(A) JUDICIALMENTE 6230 non-null   uint8
 14  ESTADO_CIVIL_SOLTEIRO(A)               6230 non-null   uint8
 15  ESTADO_CIVIL_VIÚVO(A)                  6230 non-null   uint8
 16  RACA_AMARELA                           6230 non-null   uint8
 17  RACA_BRANCA                            6230 non-null   uint8
 18  RACA_INDÍGENA                          6230 non-null   uint8
 19  RACA_PARDA                             6230 non-null   uint8
 20  RACA_PRETA                             6230 non-null   uint8
 21  DOUTOR                                 6230 non-null   int64
 22  MILITAR                                6230 non-null   int64
 23  PROFESSOR                              6230 non-null   int64
 24  RELIGIOSO                              6230 non-null   int64
dtypes: float64(2), int64(6), object(5), uint8(12)
memory usage: 883.5+ KB
```

4. Análise e Exploração dos Dados

Considerando o DataFrame com a distinção de grupos caracterizados por títulos agregados ao nome de urna, foram construídas tabelas de referência cruzada (crosstabs).

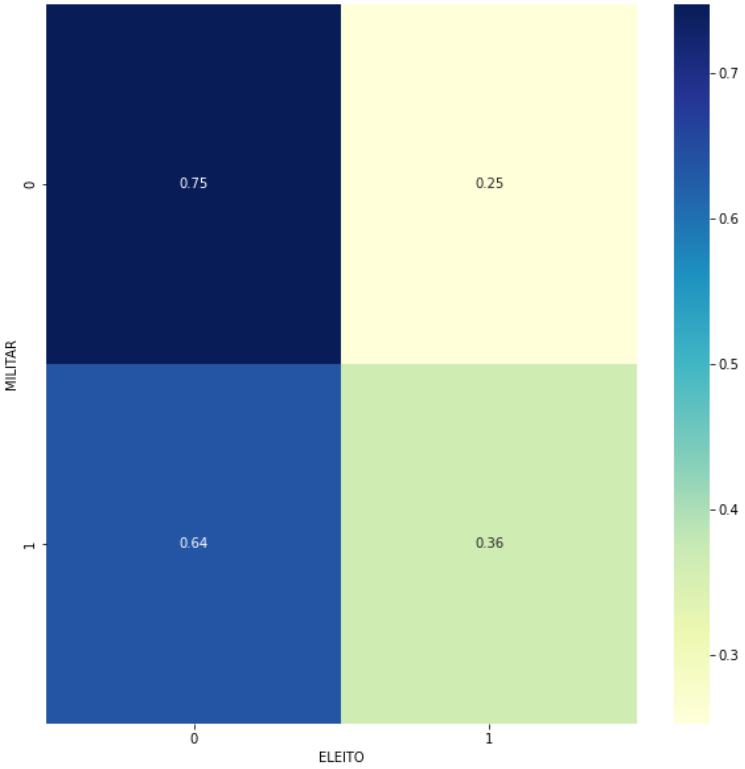
```
ct_pct_militar = pd.crosstab(df_candidatos['MILITAR'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_professor = pd.crosstab(df_candidatos['PROFESSOR'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_doutor = pd.crosstab(df_candidatos['DOUTOR'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_religioso = pd.crosstab(df_candidatos['RELIGIOSO'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_genero = pd.crosstab(df_candidatos['DS_GENERO'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_militar_fem = pd.crosstab(df_candidatos_fem['MILITAR'],
    df_candidatos_fem['ELEITO'], normalize='index')
```

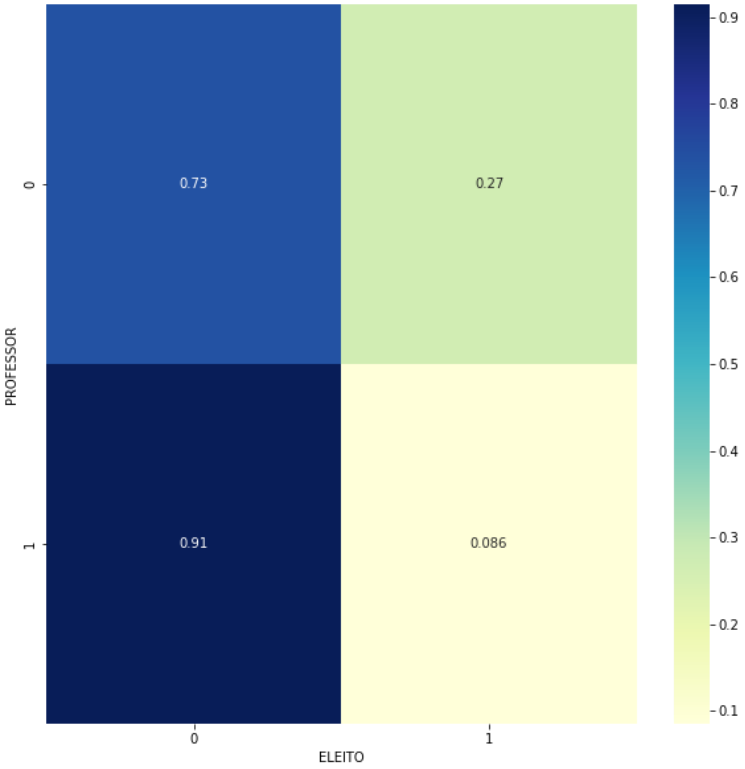
✓ 0.7s

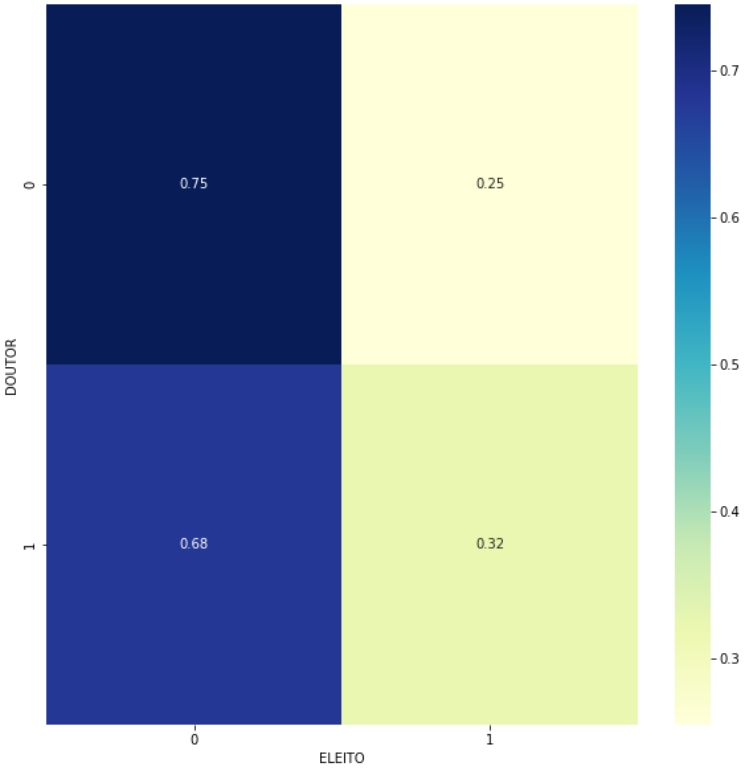
```
ct_militar = pd.crosstab(df_candidatos['MILITAR'],
    df_candidatos['ELEITO'])

ct_professor = pd.crosstab(df_candidatos['PROFESSOR'],
    df_candidatos['ELEITO'])

ct_doutor = pd.crosstab(df_candidatos['DOUTOR'],
    df_candidatos['ELEITO'])

ct_religioso = pd.crosstab(df_candidatos['RELIGIOSO'],
    df_candidatos['ELEITO'])

ct_genero = pd.crosstab(df_candidatos['DS_GENERO'],
    df_candidatos['ELEITO'])

ct_militar_fem = pd.crosstab(df_candidatos_fem['MILITAR'],
    df_candidatos_fem['ELEITO'])
```

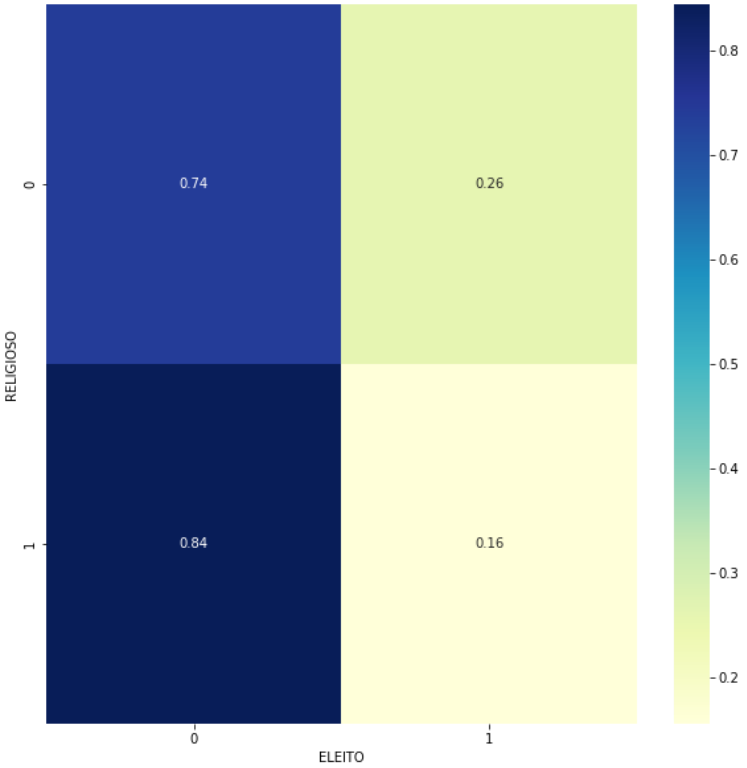
✓ 0.7s

As tabelas demonstram que aproximadamente um quarto dos candidatos se elegem contudo essa proporção não é observada nos candidatos agrupados por títulos.


0.75
0.25
0.64
0.36
MILITAR
0
1
0
1
ELEITO
0.7
0.6
0.5
0.4
0.3


0.73
0.27
0.91
0.086
PROFESSOR
0
1
0
1
ELEITO
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


0.75
0.25
0.68
0.32
DOUTOR
0
1
0
1
ELEITO
0.7
0.6
0.5
0.4
0.3


0.74
0.26
0.84
0.16
RELIGIOSO
0
1
0
1
ELEITO
0.8
0.7
0.6
0.5
0.4
0.3
0.2

Em seguida foi gerada a matriz de correlação utilizando a biblioteca pandas.DataFrame, função corr.

```
mask = np.zeros_like(df_candidatos.corr())
triangle_indices = np.triu_indices_from(mask)
mask[triangle_indices] = True
```

0.3s

```
plt.figure(figsize=(40,25))
sns.heatmap(df_candidatos.corr(), mask=mask,
    annot=True, annot_kws={"size": 14})
sns.set_style('white')
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

Considerando uma possível influência do valor das despesas contratadas no resultado do candidato, o próximo ítem a ser analisado é a despesa contratada dentro de cada grupo. Para isso foi utilizada a biblioteca pandas.DataFrame função query e describe.

```
df_candidatos_mil = df_candidatos.query("MILITAR == 1")
df_candidatos_mil['VR_DESPESA_CONTRATADA'].describe()
```

✓ 0.3s

```
count    2.220000e+02
mean     8.446023e+04
std      1.915924e+05
min      0.000000e+00
25%      1.000000e+03
50%      1.345153e+04
75%      6.944747e+04
max      1.509484e+06
Name: VR_DESPESA_CONTRATADA, dtype: float64
```

```
df_candidatos_prof = df_candidatos.query("PROFESSOR == 1")
df_candidatos_prof['VR_DESPESA_CONTRATADA'].describe()
```

✓ 0.3s

```
count    2.800000e+02
mean     8.090007e+04
std      3.499494e+05
min      0.000000e+00
25%      0.000000e+00
50%      2.555175e+03
75%      1.074670e+04
max      3.376756e+06
Name: VR_DESPESA_CONTRATADA, dtype: float64
```

```
df_candidatos_dr = df_candidatos.query("DOUTOR == 1")
df_candidatos_dr['VR_DESPESA_CONTRATADA'].describe()
```

✓ 0.3s

```
count    1.990000e+02
mean     1.713730e+05
std      3.569801e+05
min      0.000000e+00
25%      3.000000e+03
50%      2.471787e+04
75%      1.592099e+05
max      2.185890e+06
Name: VR_DESPESA_CONTRATADA, dtype: float64
```

```
df_candidatos_reli = df_candidatos.query("RELIGIOSO == 1")
df_candidatos_reli['VR_DESPESA_CONTRATADA'].describe()
```

✓ 0.4s

```
count    1.150000e+02
mean     1.049522e+05
std      3.225269e+05
min      0.000000e+00
25%      0.000000e+00
50%      6.450000e+02
75%      2.773121e+04
max      2.747750e+06
Name: VR_DESPESA_CONTRATADA, dtype: float64
```

5. Criação de Modelos de Machine Learning

Para a geração dos modelos, o DataFrame df_candidatos foi preparado com dados qualitativos tendo passado pelo processo de dummy encoding. Sendo assim, o próximo passo foi o ajuste da proporção dos candidatos eleitos e não eleitos. Esta etapa foi importante em vista do objetivo do modelo que é a prever a situação do candidato após a totalização dos votos levando em consideração o grupo no qual ele está inserido.

```
df_candidatos_majority = df_candidatos[df_candidatos.ELEITO==0]
df_candidatos_minority = df_candidatos[df_candidatos.ELEITO==1]

df_candidatos_minority_upsampled = skl_utils.resample(df_candidatos_minority, replace=True,
    n_samples=len(df_candidatos[df_candidatos.ELEITO==0]))

df_candidatos_upsampled = pd.concat([df_candidatos_majority, df_candidatos_minority_upsampled])
```

Ao estabelecer o número de amostras (n_samples) sendo o tamanho do DataFrame dos candidatos não eleitos por meio da função len() temos como resultado um DataFrame de candidatos eleitos do mesmo tamanho que os não eleitos. Nesta condição seguimos com a remoção das colunas qualitativas e geração do DataFrame contendo as features do modelo. Separadamente temos o target. Os dois DataFrame foram utilizados como parâmetro na geração de bases de treinamento e de teste.

```
df_candidatos_upsampled = df_candidatos_upsampled.drop(['NM_URNA_CANDIDATO'], axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_GENERO'], axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_ESTADO_CIVIL'], axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_COR_RACA'], axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_SIT_TOT_TURNO'], axis=1)

X_features = df_candidatos_upsampled.drop(['ELEITO'], axis = 1)
y_target = df_candidatos_upsampled.ELEITO
X_features.info()
```

✓ 0.4s

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9258 entries, 70000602710 to 260000615075
Data columns (total 19 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   NR_IDADE_DATA_POSSE                     9258 non-null   float64
 1   DS_GRAU_INSTRUCAO                       9258 non-null   int64
 2   VR_DESPESA_CONTRATADA                   9258 non-null   float64
 3   GENERO_FEMININO                         9258 non-null   uint8
 4   GENERO_MASCULINO                        9258 non-null   uint8
 5   ESTADO_CIVIL_CASADO(A)                  9258 non-null   uint8
 6   ESTADO_CIVIL_DIVORCIADO(A)              9258 non-null   uint8
 7   ESTADO_CIVIL_SEPARADO(A) JUDICIALMENTE  9258 non-null   uint8
 8   ESTADO_CIVIL_SOLTEIRO(A)                9258 non-null   uint8
 9   ESTADO_CIVIL_VIÚVO(A)                   9258 non-null   uint8
 10  RACA_AMARELA                            9258 non-null   uint8
 11  RACA_BRANCA                             9258 non-null   uint8
 12  RACA_INDÍGENA                           9258 non-null   uint8
 13  RACA_PARDA                              9258 non-null   uint8
 14  RACA_PRETA                              9258 non-null   uint8
 15  DOUTOR                                  9258 non-null   int64
 16  MILITAR                                 9258 non-null   int64
 17  PROFESSOR                               9258 non-null   int64
 18  RELIGIOSO                               9258 non-null   int64
dtypes: float64(2), int64(5), uint8(12)
memory usage: 687.1 KB
```

A função train_test_split da biblioteca sklearn.model_selection foi utilizada para dividir o DataFrame sendo uma parte para treinamento e a outra para teste numa proporção de 75-25%

```
xtreinamento, xteste, ytreinamento, yteste = (
    skl_model.train_test_split(X_features, y_target))
```
✓ 0.3s

```
len(xtreinamento)
```
✓ 0.4s

```
6943
```

```
len(xteste)
```
✓ 0.4s

```
2315
```

Em seguida o primeiro modelo foi instanciado e treinado recebendo como parâmetro os DataFrame xtreinamento e ytreinamento.

```
rfm = skl_ensemble.RandomForestClassifier()
rfm = rfm.fit(xtreinamento, ytreinamento)
```
✓ 0.3s

O Random Forest da biblioteca sklearn.ensemble foi escolhido por trabalhar bem com mais variáveis e por lidar com valores discrepantes por conta própria. Como resultado obtivemos um modelo com boa precisão e score de 95%, sendo o score a proporção das previsões corretas sobre o total das previsões somando corretas e incorretas.

```
tp_rfm = rfm.predict(xteste)
print("Accuracy score: ", skl_metrics.accuracy_score(yteste, tp_rfm))
print(skl_metrics.classification_report(yteste, tp_rfm))
```

✓ 0.6s

```
Accuracy score:  0.9511879049676026
              precision    recall  f1-score   support

           0       0.98      0.92      0.95      1130
           1       0.92      0.98      0.95      1185

    accuracy                           0.95      2315
   macro avg       0.95      0.95      0.95      2315
weighted avg       0.95      0.95      0.95      2315
```

Em seguida o segundo modelo foi instanciado e treinado recebendo como parâmetro os DataFrame xtreinamento e ytreinamento.

```
lr = skl_lm.LogisticRegression()
lr = lr.fit(xtreinamento, ytreinamento)
```

✓ 0.4s

A Logistic Regression da biblioteca sklearn.linear_model foi escolhido pela natureza da pergunta que se buscou responder; se um candidato será ou não eleito. Ainda que a pergunta seja adequada, o modelo é mais apropriado para dados contínuos e ordinais. Neste sentido, como resultado obtivemos um modelo com baixa precisão e score de 83%.

```
tp_lr = lr.predict(xteste)
print("Accuracy score: ", skl_metrics.accuracy_score(yteste, tp_lr))
print(skl_metrics.classification_report(yteste, tp_lr))
```

✓ 0.3s

```
Accuracy score:  0.8384449244060476
              precision    recall  f1-score   support

           0       0.77      0.96      0.85      1130
           1       0.95      0.72      0.82      1185

    accuracy                           0.84      2315
   macro avg       0.86      0.84      0.84      2315
weighted avg       0.86      0.84      0.84      2315
```

Por fim o modelo melhor avaliado, o random forest, foi exportado no formato pickle permitindo assim que ele seja utilizado em outro contexto.

```
joblib.dump(rfm, 'rfm.pkl')
```

✓ 0.3s

6. Interpretação dos Resultados

Ao aplicar a função scipy.stats.chi2_contingency na tabela de referência cruzada dos candidatos militares obtivemos um resultado que indica que as variáveis observadas não são independentes. Este comportamento também se repete para o grupo dos professores.

```
chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_militar)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if chi > percentile:
    print("Hipótese nula rejeitada.")
else:
    print("Hipótese nula não pode ser rejeitada.")
```

✓ 0.4s

```
p-value:  0.0002447508280365537
chi = 13.451956, percentile = 3.841459

Hipótese nula rejeitada.
```

```
chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_professor)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if chi > percentile:
    print("Hipótese nula rejeitada.")
else:
    print("Hipótese nula não pode ser rejeitada.")
```

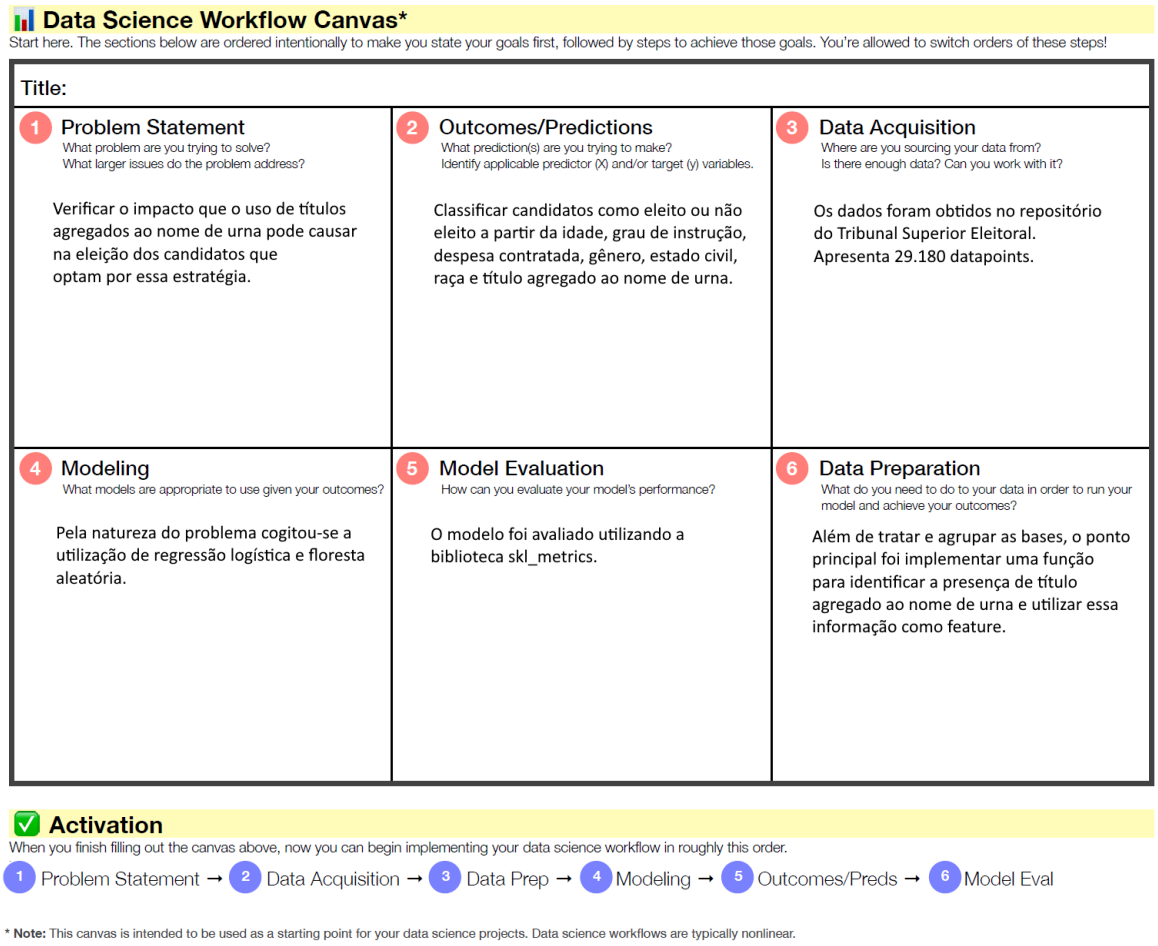
✓ 0.3s

```
p-value:  3.114221969154514e-11
chi = 44.103760, percentile = 3.841459

Hipótese nula rejeitada.
```

O modelo Random Forest apresentou melhor capacidade de classificação para o problema apresentado considerando sua média. Foi possível observar também que o nome de urna do candidato é indicativo de fatores que influenciam no voto do eleitor principalmente no caso de professores e militares.

7. Apresentação dos Resultados

Considerando o workflow proposto por Vasandani é possível observar as etapas propostas e como foram executadas nesse cenário.



Data Science Workflow Canvas*

Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!

Title:

1 Problem Statement
What problem are you trying to solve?
What larger issues do the problem address?

Verificar o impacto que o uso de títulos agregados ao nome de urna pode causar na eleição dos candidatos que optam por essa estratégia.

2 Outcomes/Predictions
What prediction(s) are you trying to make?
Identify applicable predictor (X) and/or target (y) variables.

Classificar candidatos como eleito ou não eleito a partir da idade, grau de instrução, despesa contratada, gênero, estado civil, raça e título agregado ao nome de urna.

3 Data Acquisition
Where are you sourcing your data from?
Is there enough data? Can you work with it?

Os dados foram obtidos no repositório do Tribunal Superior Eleitoral. Apresenta 29.180 datapoints.

4 Modeling
What models are appropriate to use given your outcomes?

Pela natureza do problema cogitou-se a utilização de regressão logística e floresta aleatória.

5 Model Evaluation
How can you evaluate your model's performance?

O modelo foi avaliado utilizando a biblioteca skl_metrics.

6 Data Preparation
What do you need to do to your data in order to run your model and achieve your outcomes?

Além de tratar e agrupar as bases, o ponto principal foi implementar uma função para identificar a presença de título agregado ao nome de urna e utilizar essa informação como feature.

Activation

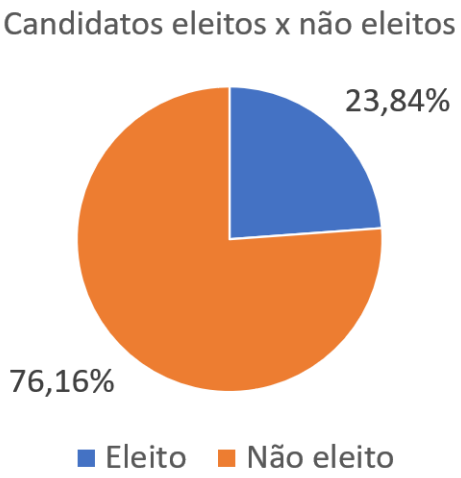
When you finish filling out the canvas above, now you can begin implementing your data science workflow in roughly this order.

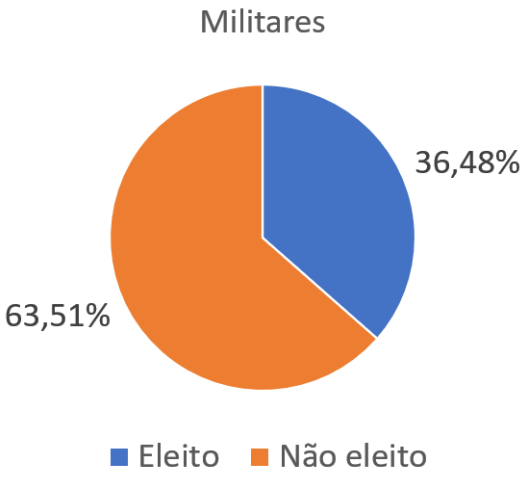
1 Problem Statement → 2 Data Acquisition → 3 Data Prep → 4 Modeling → 5 Outcomes/Preds → 6 Model Eval

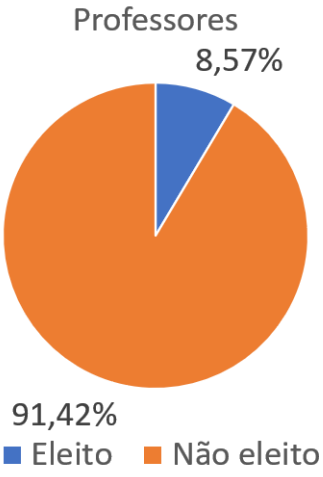
* **Note:** This canvas is intended to be used as a starting point for your data science projects. Data science workflows are typically nonlinear.

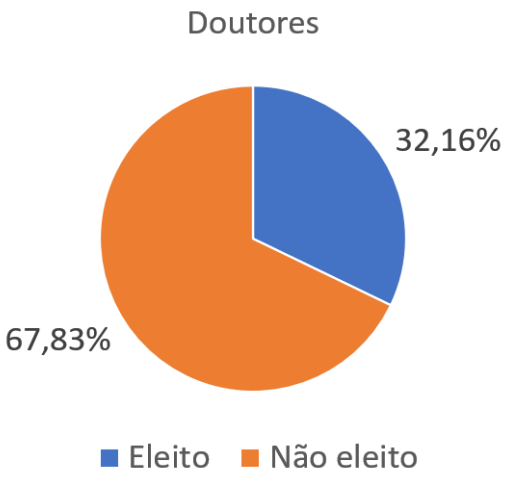
Conceptualized by Jasmine Vasandani using notes from General Assembly's Data Science Immersive. Format inspired by Business Model Canvas.

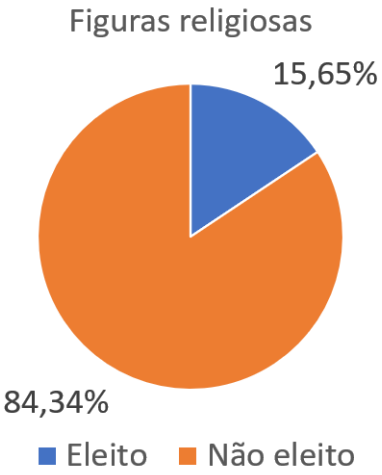
As tabelas de referência cruzada revelam que enquanto aproximadamente um quarto dos candidatos, sem nenhuma distinção, se elegem; esta proporção não se mantém quando observamos os candidatos agrupados por títulos.



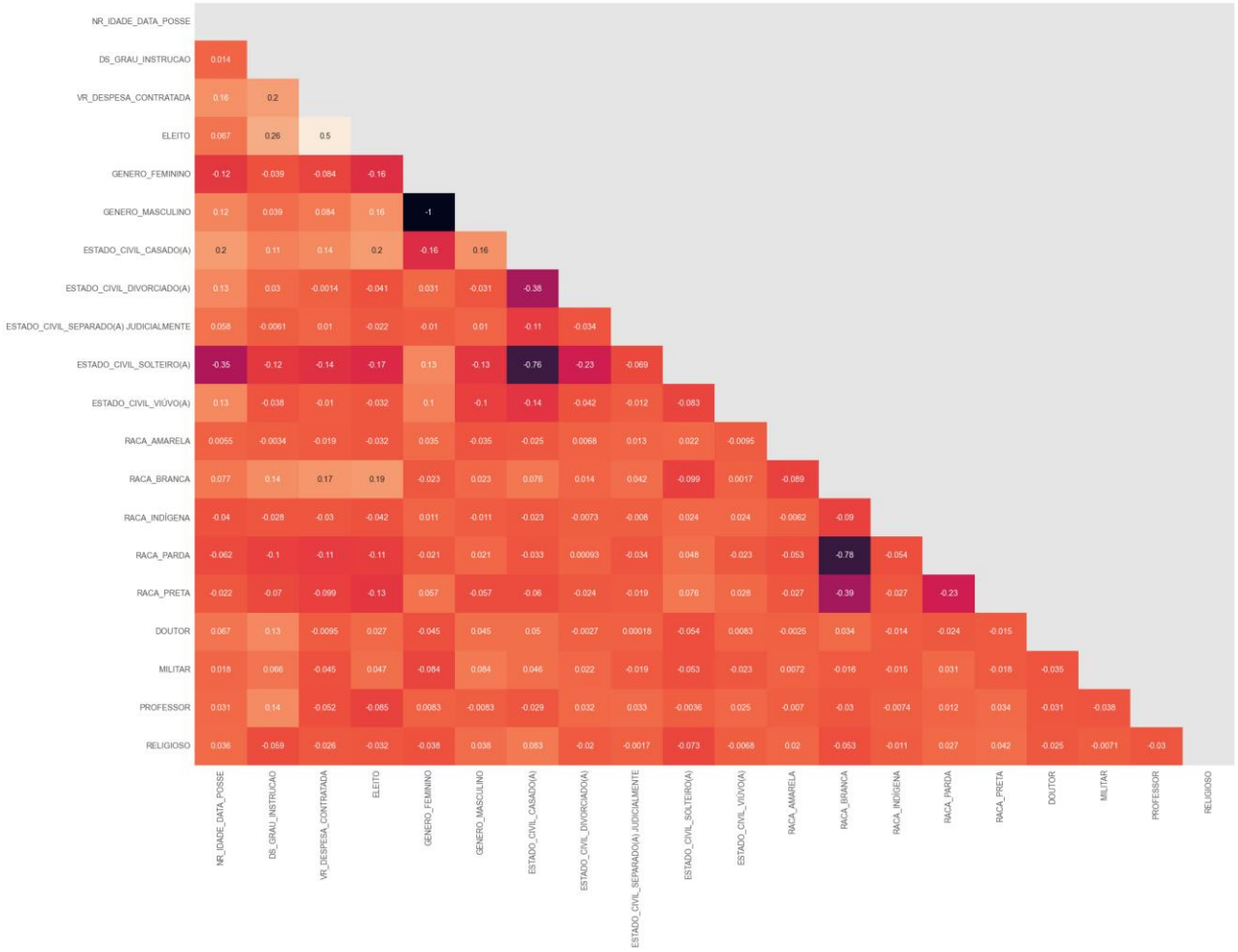








Analisando a matiz de correlação, percebe-se uma forte influência do valor das despesas contratadas no resultado do candidato.


NR_IDADE_DATA_POSSE
DS_GRAU_INSTRUCAO
VR_DESPESA_CONTRATADA
ELEITO
GENERO_FEMININO
GENERO_MASCULINO
ESTADO_CIVIL_CASADO(A)
ESTADO_CIVIL_DIVORCIADO(A)
ESTADO_CIVIL_SEPARADO(A) JUDICIALMENTE
ESTADO_CIVIL_SOLTEIRO(A)
ESTADO_CIVIL_VIÚVO(A)
RACA_AMARELA
RACA_BRANCA
RACA_INDÍGENA
RACA_PARDA
RACA_PRETA
DOUTOR
MILITAR
PROFESSOR
RELIGIOSO


Despesa média contratada por grupo de candidatos
R$ 250.000,00
R$ 200.000,00
R$ 150.000,00
R$ 100.000,00
R$ 50.000,00
R$ -
Militares
Professores
Doutores
Religiosos
Todos

Pode-se observar que, ainda que a matriz de correlação aponte influência do valor das despesas no resultado eleitoral, os militares e professores foram os dois grupos que declararam as menores despesas (ainda que a despesa seja similar) com resultado nas urnas se apresentando de forma diferente entre esses grupos, conforme pode ser observado nos gráficos de setores supracitados. Também observa-se que os grupos doutores e religiosos apresentaram comportamento dentro do esperado de acordo com a matriz de correlação. E por fim, se comparado a média geral, os grupos declararam despesas menores.

Dessa forma, após a apresentação e compreensão dos resultados é possível inferir que o nome de urna do candidato tem potencial para influenciar no voto do eleitor principalmente no caso de professores e militares. Também é importante notar que a afirmação de que o modelo apresenta 95% de precisão considera o contexto de 2018. É esperado que o sentimento da população em relação a certos grupos apresente variações com o tempo e o contexto sociocultural, além de ser possível o surgimento de novos grupos ou ainda ascensão e declínio de outros.

Contudo, ainda assim o modelo continua relevante e representa uma análise pertinente referente ao processo eleitoral brasileiro, e o mais importante é a constatação de vantagem para determinados grupos e a reflexão para criarmos no futuro mecanismos e condições capazes de proporcionar uma disputa mais paritária entre candidatos.

8. Links

Link para o vídeo: https://youtu.be/G9MUnkxhz5s

Link para o repositório: https://github.com/hqsander/nome-de-urna

APÊNDICE

Programação/Scripts

eleitoral.ipynb

```
import collections
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats
import sklearn.model_selection as skl_model
import sklearn.utils as skl_utils
import sklearn.metrics as skl_metrics
import sklearn.ensemble as skl_ensemble
import sklearn.linear_model as skl_lm
import joblib


df_candidatos = pd.read_csv("consulta_cand_2018_BRASIL.csv",
    encoding = "Latin 1", sep = ";", decimal = ',', low_memory=False)


df_candidatos.info()


df_candidatos.DS_CARGO.unique()


df_candidatos.DS_SITUACAO_CANDIDATURA.unique()


df_candidatos.DS_SIT_TOT_TURNO.unique()


df_candidatos = df_candidatos[
    df_candidatos.DS_CARGO.isin(['DEPUTADO ESTADUAL','DEPUTADO FEDERAL', 'SE-
NADOR'])]

df_candidatos = df_candidatos[
    df_candidatos.DS_SITUACAO_CANDIDATURA.isin(['APTO'])]

df_candidatos = df_candidatos[
    df_candidatos.DS_SIT_TOT_TURNO.isin(['ELEITO', 'ELEITO POR MÉDIA','ELEITO
POR QP','NÃO ELEITO'])]

df_candidatos = df_candidatos[
```

```
    ~df_candidatos.DS_COR_RACA.isin(['NÃO DIVULGÁVEL'])]

df_candidatos =
df_candidatos[['SQ_CANDIDATO','NM_URNA_CANDIDATO','NR_IDADE_DATA_POSSE','DS_GE
NERO',
    'DS_GRAU_INSTRUCAO', 'DS_ESTADO_CIVIL','DS_COR_RACA','DS_SIT_TOT_TURNO']]

df_candidatos = df_candidatos.set_index('SQ_CANDIDATO')
df_candidatos.info()


df_candidatos.DS_GENERO.unique()


df_candidatos.DS_COR_RACA.unique()


df_despesas = pd.read_csv("despesas_contratadas_candidatos_2018_BRASIL.csv",
    encoding = "Latin 1", sep = ";", decimal = ',', low_memory=False)


df_despesas = df_despesas[['SQ_CANDIDATO','VR_DESPESA_CONTRATADA']]
df_despesas = df_despesas.groupby(['SQ_CANDIDATO']).sum()
df_despesas.info()


df_candidatos = df_candidatos.join(df_despesas)


df_candidatos.info()


df_candidatos['VR_DESPESA_CONTRATADA'] =
df_candidatos['VR_DESPESA_CONTRATADA'].fillna(0)
df_candidatos['VR_DESPESA_CONTRATADA'] =
df_candidatos['VR_DESPESA_CONTRATADA'].apply(np.float64)
df_candidatos['NR_IDADE_DATA_POSSE'] =
df_candidatos['NR_IDADE_DATA_POSSE'].apply(np.float64)


df_candidatos['ELEITO'] = list(map(lambda x:
int('NÃO ELEITO' not in x),
df_candidatos['DS_SIT_TOT_TURNO']))

recode = {'LÊ E ESCREVE': 1,
    'ENSINO FUNDAMENTAL INCOMPLETO': 2,
    'ENSINO FUNDAMENTAL COMPLETO': 3,
    'ENSINO MÉDIO INCOMPLETO': 4,
    'ENSINO MÉDIO COMPLETO': 5,
    'SUPERIOR INCOMPLETO': 6,
    'SUPERIOR COMPLETO': 7}
```

```
df_candidatos['DS_GRAU_INSTRUCAO'] =
df_candidatos['DS_GRAU_INSTRUCAO'].map(recode)

df_genero = pd.get_dummies(df_candidatos.DS_GENERO, prefix='GENERO')
df_candidatos = df_candidatos.join(df_genero)

df_estado_civil = pd.get_dummies(df_candidatos.DS_ESTADO_CIVIL, pre-
fix='ESTADO_CIVIL')
df_candidatos = df_candidatos.join(df_estado_civil)

df_raca = pd.get_dummies(df_candidatos.DS_COR_RACA, prefix='RACA')
df_candidatos = df_candidatos.join(df_raca)


df_candidatos['DOUTOR'] = list(map(lambda x:
int('DOUTOR' in x or
'DR ' in x or
'DR.' in x or
'DRA.' in x),
df_candidatos['NM_URNA_CANDIDATO']))

df_candidatos['MILITAR'] = list(map(lambda x:
int('AGENTE' in x or
'AGT.' in x or
'BOMBEIRO' in x or
'CABO' in x or
'CAP.' in x or
'CAPITAO' in x or
'CB' in x or
'CORONEL' in x or
'DELE.' in x or
'DELEGAD' in x or
'MAJOR' in x or
'POLICIAL' in x or
'SARG.' in x or
'SARGENTO' in x or
'SGT' in x or
'SOLDADO' in x or
'SUB TENENTE' in x or
'SUBOFICIAL' in x or
'SUBTEN' in x or
'TEN.' in x or
'TENENTE' in x),
df_candidatos['NM_URNA_CANDIDATO']))

df_candidatos['PROFESSOR'] = list(map(lambda x:
int('PROF' in x), df_candidatos['NM_URNA_CANDIDATO']))

df_candidatos['RELIGIOSO'] = list(map(lambda x:
```

```
int('AP.' in x or
'APOSTOL' in x or
'APÓSTOLO' in x or
'BISP' in x or
'DIACONO' in x or
'MISSIONÁRI' in x or
'MISSIONARI' in x or
'PADRE' in x or
'PAI ' in x or
'PASTOR' in x or
'PR ' in x or
'PR.' in x or
'REV.' in x),
df_candidatos['NM_URNA_CANDIDATO']))

df_candidatos.info()


df_candidatos_fem = df_candidatos[df_candidatos.DS_GENERO.isin(['FEMININO'])]
# df_candidatos_fem.info()


ct_pct_militar = pd.crosstab(df_candidatos['MILITAR'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_professor = pd.crosstab(df_candidatos['PROFESSOR'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_doutor = pd.crosstab(df_candidatos['DOUTOR'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_religioso = pd.crosstab(df_candidatos['RELIGIOSO'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_genero = pd.crosstab(df_candidatos['DS_GENERO'],
    df_candidatos['ELEITO'], normalize='index')

ct_pct_militar_fem = pd.crosstab(df_candidatos_fem['MILITAR'],
    df_candidatos_fem['ELEITO'], normalize='index')


ct_militar = pd.crosstab(df_candidatos['MILITAR'],
    df_candidatos['ELEITO'])

ct_professor = pd.crosstab(df_candidatos['PROFESSOR'],
    df_candidatos['ELEITO'])

ct_doutor = pd.crosstab(df_candidatos['DOUTOR'],
    df_candidatos['ELEITO'])
```

```
ct_religioso = pd.crosstab(df_candidatos['RELIGIOSO'],
    df_candidatos['ELEITO'])

ct_genero = pd.crosstab(df_candidatos['DS_GENERO'],
    df_candidatos['ELEITO'])

ct_militar_fem = pd.crosstab(df_candidatos_fem['MILITAR'],
    df_candidatos_fem['ELEITO'])


df_candidatos['VR_DESPESA_CONTRATADA'].describe()


df_candidatos.VR_DESPESA_CONTRATADA.hist(bins=5)
plt.style.use('seaborn-pastel')
plt.title("Despesa contratada por candidatos")
plt.xlabel("Despesa")
plt.ylabel("Número de candidatos")
plt.show()


df_candidatos_mil = df_candidatos.query("MILITAR == 1")
df_candidatos_mil['VR_DESPESA_CONTRATADA'].describe()


df_candidatos_mil.VR_DESPESA_CONTRATADA.hist(bins=5)
plt.style.use('seaborn-pastel')
plt.title("Despesa contratada por militares")
plt.xlabel("Despesa")
plt.ylabel("Número de candidatos")
plt.show()


df_candidatos_prof = df_candidatos.query("PROFESSOR == 1")
df_candidatos_prof['VR_DESPESA_CONTRATADA'].describe()


df_candidatos_prof.VR_DESPESA_CONTRATADA.hist(bins=5)
plt.style.use('seaborn-pastel')
plt.title("Despesa contratada por professores")
plt.xlabel("Despesa")
plt.ylabel("Número de candidatos")
plt.show()


df_candidatos_dr = df_candidatos.query("DOUTOR == 1")
df_candidatos_dr['VR_DESPESA_CONTRATADA'].describe()


df_candidatos_dr.VR_DESPESA_CONTRATADA.hist(bins=5)
plt.style.use('seaborn-pastel')
plt.title("Despesa contratada por doutores")
```

```
plt.xlabel("Despesa")
plt.ylabel("Número de candidatos")
plt.show()


df_candidatos_reli = df_candidatos.query("RELIGIOSO == 1")
df_candidatos_reli['VR_DESPESA_CONTRATADA'].describe()


df_candidatos_reli.VR_DESPESA_CONTRATADA.hist(bins=5)
plt.style.use('seaborn-pastel')
plt.title("Despesa contratada por fig. religiosas")
plt.xlabel("Despesa")
plt.ylabel("Número de candidatos")
plt.show()


significance = 0.05
lower_tail = 1 - significance
chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_doutor)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if chi > percentile:
    print("Hipótese nula rejeitada.")
else:
    print("Hipótese nula não pode ser rejeitada.")


chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_militar)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if chi > percentile:
    print("Hipótese nula rejeitada.")
else:
    print("Hipótese nula não pode ser rejeitada.")


chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_professor)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if chi > percentile:
    print("Hipótese nula rejeitada.")
```

```
else:
    print("Hipótese nula não pode ser rejeitada.")


chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_religioso)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if chi > percentile:
    print("Hipótese nula rejeitada.")
else:
    print("Hipótese nula não pode ser rejeitada.")


chi, pval, dof, expected = scipy.stats.chi2_contingency(ct_genero)
print("p-value: ", pval)
percentile = scipy.stats.chi2.ppf(lower_tail, dof)

print("chi = %.6f, percentile = %.6f\n" % (chi, percentile))

if pval < significance:
    print("Hipótese nula rejeitada.")
else:
    print("Hipótese nula não pode ser rejeitada.")


df_militares = df_candidatos[df_candidatos.MILITAR.isin([1])]
df_militares.head()
genero_candidatos_militares = collections.Counter(df_militares['DS_GENERO'])
genero_candidatos_militares

plt.style.use('ggplot')
plt.pie(genero_candidatos_militares.values(), labels = gene-
ro_candidatos_militares.keys(),
        autopct = '%1.1f%%', textprops={'fontsize': 16})
plt.title("Gênero dos candidatos militares", fontsize=18)
plt.axis("image")
plt.show()


df_professores = df_candidatos[df_candidatos.PROFESSOR.isin([1])]
df_professores.head()
genero_candidatos_professores = collec-
tions.Counter(df_professores['DS_GENERO'])
genero_candidatos_professores

plt.style.use('ggplot')
```

```
plt.pie(genero_candidatos_professores.values(), labels = gene-
ro_candidatos_professores.keys(),
        autopct = '%1.1f%%', textprops={'fontsize': 16})
plt.title("Gênero dos candidatos professores", fontsize=18)
plt.axis("image")
plt.show()


mask = np.zeros_like(df_candidatos.corr())
triangle_indices = np.triu_indices_from(mask)
mask[triangle_indices] = True



plt.figure(figsize=(40,25))
sns.heatmap(df_candidatos.corr(), mask=mask,
    annot=True, annot_kws={"size": 14})
sns.set_style('white')
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()


df_candidatos[df_candidatos.ELEITO==0].info()


df_candidatos[df_candidatos.ELEITO==1].info()


df_candidatos_majority = df_candidatos[df_candidatos.ELEITO==0]
df_candidatos_minority = df_candidatos[df_candidatos.ELEITO==1]

df_candidatos_minority_upsampled = skl_utils.resample(df_candidatos_minority,
replace=True,
    n_samples=len(df_candidatos[df_candidatos.ELEITO==0]))

df_candidatos_upsampled = pd.concat([df_candidatos_majority,
df_candidatos_minority_upsampled])


df_candidatos_upsampled = df_candidatos_upsampled.drop(['NM_URNA_CANDIDATO'],
axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_GENERO'], axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_ESTADO_CIVIL'],
axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_COR_RACA'],
axis=1)
df_candidatos_upsampled = df_candidatos_upsampled.drop(['DS_SIT_TOT_TURNO'],
axis=1)

X_features = df_candidatos_upsampled.drop(['ELEITO'], axis = 1)
```

```
y_target = df_candidatos_upsampled.ELEITO
X_features.info()


xtreinamento, xteste, ytreinamento, yteste = (
    skl_model.train_test_split(X_features, y_target))


rfm = skl_ensemble.RandomForestClassifier()
rfm = rfm.fit(xtreinamento, ytreinamento)


tp_rfm = rfm.predict(xteste)
print("Accuracy score: ", skl_metrics.accuracy_score(yteste, tp_rfm))
print(skl_metrics.classification_report(yteste, tp_rfm))


lr = skl_lm.LogisticRegression()
lr = lr.fit(xtreinamento, ytreinamento)


tp_lr = lr.predict(xteste)
print("Accuracy score: ", skl_metrics.accuracy_score(yteste, tp_lr))
print(skl_metrics.classification_report(yteste, tp_lr))


joblib.dump(rfm, 'rfm.pkl')
```