

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

Daniel Cirino do Vale

**CLASSIFICAÇÃO TEMÁTICA DE PROPOSTAS LEGISLATIVAS APRESENTADAS
À CÂMARA DOS DEPUTADOS POR MEIO DA APLICAÇÃO DE MODELOS DE
MACHINE LEARNING**

Belo Horizonte
2022

Daniel Cirino do Vale

**CLASSIFICAÇÃO TEMÁTICA DE PROPOSTAS LEGISLATIVAS APRESENTADAS
À CÂMARA DOS DEPUTADOS POR MEIO DA APLICAÇÃO DE MODELOS DE
MACHINE LEARNING**

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	04
1.1. Contextualização	05
1.2. O problema proposto	05
1.3. Objetivos	06
2. Coleta de dados	07
3. Processamento/Tratamento de Dados	16
4. Análise e Exploração dos Dados	35
5. Criação de Modelos de Machine Learning	40
6. Interpretação dos Resultados	57
7. Apresentação dos Resultados	59
8. Links	65
REFERÊNCIAS	66

1. Introdução

O presidencialismo é a forma de governo adotada no Brasil desde a Constituição brasileira de 1988. Nesta forma de governo, o poder público é constituído pelo Poder Executivo, Poder Legislativo e Poder Judiciário que são formados por grupos diferentes e atribuídos de funções distintas mas com igual autonomia e independência entre eles gerando uma convivência harmônica, impedindo abusos para garantir o equilíbrio constitucional.

O Poder Executivo é exercido pelo Presidente da República no âmbito federal, pelo Governador na esfera estadual e pelo prefeito na municipal, todos eles eleitos por voto popular. A eles cabe a observância das necessidades coletivas e atendê-las de acordo com as leis, propondo planos e ações para diversas áreas como saúde, educação, cultura, segurança, programas sociais e infraestrutura.

O Poder Judiciário tem a função de aplicar a lei por meio de julgamento e interpretação de fatos. Os órgãos da justiça garantem direitos individuais, coletivos e sociais da população, instituições e Estado. O órgão superior da justiça no Brasil é o Supremo Tribunal Federal, composto por 11 ministros nomeados pelo Presidente da República.

Por fim, o Poder Legislativo tem suas principais funções relacionadas à criação de leis e fiscalização do Poder Executivo. Na esfera federal, este poder é exercido pelo Congresso Nacional, por meio de senadores e deputados federais, a nível estadual pelas Assembleias Legislativas, por deputados estaduais e a nível municipal o poder exercido pelas Câmaras Municipais onde atuam os vereadores.

Neste trabalho, o foco será a atividade legislativa realizada pela Câmara dos Deputados, buscando por meio da aplicação das técnicas de Aprendizagem de Máquina e Ciência de Dados avaliar as propostas legislativas apresentadas para votação e fazer a classificação temática das mesmas.

1.1. Contextualização

A atividade legislativa compreende a elaboração, análise e votação de vários tipos de propostas: leis ordinárias, medidas provisórias, emendas à Constituição, decretos legislativos e resoluções, entre outras. As propostas são redigidas pelos autores, que podem ser deputados, senadores, comissões ou até mesmo os cidadãos, e tratam dos mais variados temas de interesse da sociedade.

O processo de monitoramento da atividade legislativa é uma atividade que se torna cada vez mais relevante para a garantia dos interesses da sociedade e dos diversos grupos que a compõem. Esta importância se justifica no fato de que quase todas as atividades exercidas no país, seja na iniciativa privada, pública ou sem fins lucrativos, são regidas pelo conteúdo das propostas de lei.

Visando então contribuir para o processo de monitoramento da atividade legislativa, em particular na esfera federal, este trabalho busca empregar técnicas, tecnologia e ferramentas de aprendizado de máquina e ciência de dados para fazer a classificação temática das propostas de lei escritas pelos deputados federais com base no conteúdo das mesmas.

Considerando o volume de dados das proposições criadas pelos parlamentares, a variabilidade do formato dos dados e pelos dados não serem estruturados, a análise do conteúdo e classificação destas informações, de forma a gerar algum valor, caracteriza um problema problema de dados que pode ser resolvido aplicando técnicas da ciência de dados, especialmente as técnicas de machine learning.

1.2. O problema proposto

Na esfera federal, a atividade parlamentar gerou nos anos 2019, 2020 e 2021 um total de 71.640¹ (setenta e um mil, seiscentos e quarenta) proposições de diversos tipos tais como leis ordinárias, medidas provisórias, emendas à Constituição, decretos legislativos e

¹ De acordo com os dados extraídos do site da Câmara Legislativa, em <https://dadosabertos.camara.leg.br>.

resoluções, entre outras, o que representa cerca de 14 GB (gigabytes) de documentos no período. Para se ter uma idéia da quantidade de texto, são aproximadamente 215.000 páginas de texto². Este é um volume de informação que torna para o ser humano as tarefas de ler e extrair algum tipo de análise em um tempo aceitável, dispendiosas e quase impossíveis.

De todas as proposições feitas no período acima, apenas 26.863 (37%) possuem a informação de classificação temática. Este índice tão baixo pode ser justificado pelo processo manual de classificação feito pelo Centro de Documentação e Informação da Câmara dos Deputados no momento da apresentação das propostas. A falta de informação sobre a área temática que uma proposição está relacionada, dificulta a análise e acompanhamento pela sociedade.

Buscando uma forma de classificação mais eficiente e que permita uma melhor análise das áreas temáticas, foram analisados os dados de proposições apresentadas à Câmara dos Deputados, dos temas atribuídos às proposições e dos documentos que contém o inteiro teor das propostas nos anos 2019, 2020 e 2021. Todos estes dados estão disponíveis no portal de dados abertos da Câmara, no endereço eletrônico <https://dadosabertos.camara.leg.br/swagger/api.html>, portanto são dados oficiais do Governo, regidos pela Lei de Acesso a Informação Pública (Lei 12.527/2011).

1.3. Objetivos

O objetivo principal do trabalho é criar uma máquina de classificação que permita com base nas proposições que já possuem a classificação temática feita manualmente, nos documentos que contém o inteiro teor das propostas, atribuir às proposições sem temas uma classificação com base no inteiro teor das mesmas. Com a classificação de todas as propostas, o acompanhamento da atividade legislativa será facilitado permitindo identificação de pautas de interesse seja da iniciativa pública, privada ou sem fins lucrativos, podendo identificar padrões na atuação dos deputados.

² Os documentos possuem em média 300kb de tamanho e apresentam em média 3 páginas.

1.4. Linguagem de programação e ferramentas

Para realização do trabalho foi escolhida a linguagem de programação Python. Trata-se de uma linguagem de código aberto (Open Source), considerada de alto nível e bastante utilizada em projetos de Data Science e Machine Learning. Para o desenvolvimento foi utilizado o Jupyter Notebook, que é uma plataforma de computação baseada na web que permite a programação iterativa facilitando a escrita e compilação dos códigos Python.

A comunidade Python disponibiliza uma série de bibliotecas que auxiliam nos processos de Data Science e Machine Learning, como por exemplo obtenção de dados da internet, processamento de arquivos, visualização e pré-processamento de dados, treinamento e validação de modelos de aprendizagem de máquina. Neste projeto foram utilizadas as bibliotecas abaixo:

1. **requests** - utilizada para obter dados da internet.
2. **pandas** - utilizada para manipulação e análise de dados.
3. **numpy** - utilizada para o processamento de grandes, multidimensionais arrays e matrizes.
4. **PyPDF2** - utilizada para extrair texto de arquivos PDF.
5. **matplotlib** e **seaborn** - utilizada para gerar visualização dos dados.
6. **sklearn** - utilizada para as funções de aprendizado de máquina, mais especificamente para tarefas de classificação.

Outras ferramentas foram utilizadas de forma a auxiliar na análise de dados, com destaque para o MS Excel, software de planilha eletrônica da Microsoft e Power BI, ferramenta para criação de dashboards.

2. Coleta de Dados

Foi selecionada como fonte para obtenção dos dados o portal de dados abertos da Câmara Legislativa (<https://dadosabertos.camara.leg.br/swagger/api.html#staticfile>). O portal disponibiliza uma API (Application Programming Interface) para extração de dados em

volume pequeno utilizando o protocolo HTTP e uma seção de download de conjuntos de dados completos em diversos formatos separados anualmente. Para o propósito deste trabalho foi escolhido o formato CSV (comma separated values) dos anos de 2018, 2019, 2020 e 2021.

2.1. Proposições por ano de apresentação

Este *dataset* contém os dados sobre as proposições apresentadas à Câmara dos Deputados para deliberação em determinado ano. A atualização deste conjunto de dados é diária, e pode ser obtido no endereço eletrônico <http://dadosabertos.camara.leg.br/arquivos/proposicoes/csv/proposicoes-{ano}.csv>.

Campos do arquivo de proposições:

#	Nome da coluna/campo	Descrição	Tipo
1	id	Identificador único da proposição	Número
2	uri	Endereço eletrônico da proposição	Texto
3	siglaTipo	Sigla que identifica o tipo de proposição.	Texto
4	numero	Número da proposição	Número
5	ano	Ano da proposição	Número
6	codTipo	Código identificador do tipo de proposição	Número
7	descricaoTipo	Descrição do tipo de proposição	Texto
8	ementa	Resumo da proposição	Texto
9	ementaDetalhada	Complemento do resumo da proposição	Texto
10	keywords	Lista de palavras chaves relacionadas à proposição	Texto
11	dataApresentacao	Data em que foi apresentada a proposição	Data e Hora

12	uriOrgaoNumerador	Endereço eletrônico do órgão numerador	Texto
13	uriPropAnterior	Endereço eletrônico da proposição anterior	Texto
14	uriPropPrincipal	Endereço eletrônico da proposição principal	Texto
15	uriPropPosterior	Endereço eletrônico da proposição posterior	Texto
16	urlInteiroTeor	Endereço eletrônico do inteiro teor da proposição	Texto
17	urnFinal	??	Texto
18	ultimoStatus_dataHora	Data e hora do último status da proposição	Texto
19	ultimoStatus_sequencia	Sequência do último status da proposição	Número
20	ultimoStatus_uriRelator	Endereço eletrônico do relator do último status	Texto
21	ultimoStatus_idOrgao	Código do órgão do último status	Número
22	ultimoStatus_siglaOrgao	Sigla do órgão do último status	Texto
23	ultimoStatus_uriOrgao	Endereço eletrônico do órgão do último status	Texto
24	ultimoStatus_regime	Regime do último status	Texto
25	ultimoStatus_descricaoTramitacao	Descrição da tramitação do último status	Texto
26	ultimoStatus_idTipoTramitacao	Código do tipo de tramitação do último status	Número
27	ultimoStatus_descricaoSituacao	Descrição da situação do último status	Texto
28	ultimoStatus_idSituacao	Código da situação do último status	Número
29	ultimoStatus_despacho	Despacho do último status	Texto

30	ultimoStatus_url	Endereço eletrônico do último status	Texto
----	------------------	--------------------------------------	-------

2.2. Classificação temática das proposições por ano

Contém o relacionamento entre uma preposição e o tema ao qual ela foi atribuída. Uma proposição pode ser classificada em mais de um tema, logo ela pode aparecer mais de uma vez neste dataset. Este conjunto de dados é atualizado diariamente e fica disponível no endereço eletrônico

<http://dadosabertos.camara.leg.br/arquivos/proposicoesTemas/{csv}/proposicoesTemas-{ano}.{csv}>

Campos do arquivo de classificação temática das proposições

#	Nome da coluna/campo	Descrição	Tipo
1	uriProposicao	Endereço eletrônico da proposição	Texto
2	siglaTipo	Sigla que identifica o tipo de proposição.	Texto
3	numero	Número da proposição no ano	Número
4	ano	Ano da proposição	Número
5	codTema	Código que identifica o tema atribuído à proposição	Número
6	tema	Nome do tema atribuído à proposição	Texto
7	relevancia	Valor de relevância atribuído à proposição	Número

2.3. Dataset inteiro teor das proposições

Para poder fazer a análise do conteúdo do texto das proposições, foi necessário obter o inteiro teor das mesmas. O portal da Câmara dos deputados não oferece um arquivo que contenha esta informação, contudo disponibiliza uma *API (Application Programming Interface)* que permite a consulta dos documentos de forma unitária. Desta forma, utilizando

o endereço do inteiro teor de cada proposição informado do *dataset* de PROPOSIÇÕES, foi possível construir este conjunto de dados.

Campos do *dataset* inteiro teor:

Posição	Nome da coluna/campo	Descrição	Tipo
1	id	Identificador da proposição	Número
2	urlInteiroTeor	Endereço eletrônico do documento da proposição	Texto
3	statusExtracao	Indicador do status do processo de extração	Número
4	tipoDocumento	Tipo de documento do inteiro teor da proposição	Texto
5	tamanhoEmBytes	Tamanho do documento do inteiro teor da proposição	Número
6	inteiroTeor	Texto do inteiro teor	Texto
7	tokensInteiroTeor	Palavras chaves fruto da tokenização do inteiro teor	Texto

2.4. Programa de extração dos dados

Para automatizar a extração dos datasets de proposições e classificação temática do portal de dados da Câmara dos deputados foi criada a função abaixo, que recebe como parâmetros o ano de início e fim que se deseja extrair. A função faz o download dos arquivos em um diretório pré configurado.

Figura 001 - Código da função de extração

```

1 def extrairDatasetsPorAno(anoInicio:int, anoFim:int):
2     anosParaExtracao = range(anoInicio, anoFim+1)
3     for ano in anosParaExtracao:
4         urlArquivoProposicao = f'{URL_ARQUIVO_PROPOSICOES}proposicoes-{ano}.csv'
5         urlArquivoTemas = f'{URL_ARQUIVO_TEMAS}proposicoesTemas-{ano}.csv'
6
7         arquivoProposicao = requests.get(urlArquivoProposicao)
8         with open(f'{DIR_DATASET}/proposicoes/proposicoes-{ano}.csv', 'wb') as arquivo:
9             arquivo.write(arquivoProposicao.content)
10            print(f'[OK] Arquivo baixado: {urlArquivoProposicao} ({len(arquivoProposicao.content)} bytes)')
11
12         arquivoTemas = requests.get(urlArquivoTemas)
13         with open(f'{DIR_DATASET}/temas/proposicoesTemas-{ano}.csv', 'wb') as arquivo:
14             arquivo.write(arquivoTemas.content)
15
16         print(f'[OK] Arquivo baixado: {urlArquivoTemas} ({len(arquivoTemas.content)} bytes)')
17         print(f'[OK] Finalizada extração dos datasets de proposição e classificação temática de {anoInicio} a {anoFim}')

```

Fonte: Elaboração própria

Figura 002 -Resultado da extração

```

1 extrairDatasetsPorAno(2018,2021)
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoes/csv/proposicoes-2018.csv (15727867) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoesTemas/csv/proposicoesTemas-2018.csv (614772) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoes/csv/proposicoes-2019.csv (31523314) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoesTemas/csv/proposicoesTemas-2019.csv (1715368) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoes/csv/proposicoes-2020.csv (24161880) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoesTemas/csv/proposicoesTemas-2020.csv (2246413) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoes/csv/proposicoes-2021.csv (33474989) bytes
[OK] Arquivo baixado: http://dadosabertos.camara.leg.br/arquivos/proposicoesTemas/csv/proposicoesTemas-2021.csv (1772951) bytes
[OK] Finalizada extração dos datasets de proposição e classificação temática de 2018 a 2021

```

Fonte: Elaboração própria

Outras duas funções foram criadas para fazer a compilação dos arquivos de proposições e temas baixados em arquivos únicos, permitindo que fosse possível trabalhar com *datasets* unificados, contendo os dados de todos os anos de interesse, na etapa de processamento dos dados.

Figura 003 - Código da função para unificar arquivos de proposição

```

1 def gerarDataframeProposicoes():
2     dfProposicoesList = []
3
4     for arquivo in os.listdir(f'{DIR_DATASET}proposicoes/'):
5         try:
6             anoDataset = arquivo.split('.')[0][-4:]
7             df = pd.read_csv(f'{DIR_DATASET}proposicoes/{arquivo}', sep=';', low_memory=False)
8             df.loc[:, 'anoDataset'] = anoDataset
9             dfProposicoesList.append(df)
10            print(f'[OK] [{anoDataset}] {arquivo}')
11        except Exception as e:
12            print(f'[ERRO] {arquivo}')
13
14    return pd.concat(dfProposicoesList)

```

Fonte: Elaboração Própria

Figura 004 - Código da função para unificar arquivos de classificação temática

```

1 def gerarDataframeTemas():
2     dfTemasList = []
3     for arquivo in os.listdir(f'{DIR_DATASET}temas/'):
4         try:
5             anoDataset = arquivo.split('.')[0][-4:]
6             df = pd.read_csv(f'{DIR_DATASET}temas/{arquivo}', sep=';', low_memory=False)
7             df.loc[:, 'anoDataset'] = anoDataset
8             dfTemasList.append(df)
9             print(f'[OK] [{anoDataset}] {arquivo}')
10        except Exception as e:
11            print(f'[ERRO] {arquivo}')
12
13    return pd.concat(dfTemasList)

```

Fonte: Elaboração Própria

Tomando como base os campos “id” e “urlInteiroTeor” do dataset de proposições unificado, foi criado o *dataset* que recebeu as informações do inteiro teor após o processo de extração. Neste *dataset* foi incluído uma coluna para controle da extração (“statusExtracao”), 3 colunas para armazenar informações sobre o conteúdo extraído (“tipoDocumento”, “tamanhoEmBytes”, “inteiroTeor”).

Para obter o inteiro teor de cada proposição, foi necessário consultar cada documento de forma unitária no portal de dados abertos da Câmara dos Deputados. Para esta tarefa foi implementada a função “extrairInteiroTeor” em Python, que lê o arquivo “inteiroTeor.csv” e para cada registro no *dataset* faz uma requisição HTTP por meio da biblioteca “*requests*” utilizando o campo “urlInteiroTeor”.

O retorno destas requisições normalmente foram documentos no formato PDF (*Portable Document Format*), contudo foram observados casos em que o retorno foi uma página *.htm, que contém marcações HTML (*Hypertext Markup Language*), e casos onde foram retornados um arquivo *.doc (documento de processamento de texto). Devido esta

variedade de tipos, foram processados os arquivos *.pdf e os *.htm. Os arquivos *.doc, por exigirem uma quantidade de processamento adicional que não se justifica em função do volume que é baixo, foram desconsiderados.

De forma geral o processo de obtenção dos dados pela internet pode apresentar erros temporários ou até mesmo erros nos arquivos que se desejava extrair e para tentar contornar esta situação a função que faz a extração foi construída para ser resiliente a falhas de modo que possa registar o resultado da extração (status e detalhes do conteúdo) e fazer nova tentativa de extração se for necessário.

A estratégia adotada para extração do inteiro teor das proposições foi de não armazenar arquivos localmente, mas sim tratar o retorno obtido na requisição HTTP. As respostas HTTP possuem cabeçalhos (*headers*) que contêm informações sobre os dados retornados, uma destas informações é o tipo de conteúdo das respostas (*content-type*). Desta forma foi possível saber qual documento é um *.pdf (*content-type* igual a “*application/pdf*”) e qual é um *.htm (*content-type* igual a “*text/html;charset=utf-8*”) e aplicar o processamento ideal a cada formato de documento.

Os documentos *.pdf são processados com uma biblioteca do Python chamada PyPDF2, utilizada para extrair texto deste formato de arquivo. Já os documentos *.htm são processados utilizando uma técnica de *webscraping*, que é uma forma de mineração de dados da web. Devido ao fato de estarmos interessados apenas no texto do documento e não em algum elemento específico de HTML, não foi necessário adotar uma biblioteca específica de *webscraping* como *BeautifulSoap*, sendo utilizada uma técnica de expressão regular para remover todas as TAGs HTML do documento, restando apenas o texto.

Figura 005 - Função de extração do inteiro teor

```

1 def extrairInteiroTeor(df, qtdLimite=0):
2     dfParaExtracao = df.loc[df['statusExtracao']==0]
3     print(f'[INFO] Proposicoes para extracao de inteiro teor: {len(dfParaExtracao)}')
4
5     sessao = requests.Session()
6     contador = 0
7     t0 = time()
8     for index,linha in dfParaExtracao.iterrows():
9         try:
10             url = linha['urlInteiroTeor']
11             idPreposicao = linha['id']
12             indexPreposicao = dfInteiroTeor.loc[df['id']==idPreposicao].index.item()
13
14             if contador%100==0:
15                 print(f'[INFO] [{indexPreposicao}][{idPreposicao}][{url}]')
16
17             if type(url) == float:
18                 df.loc[indexPreposicao,['statusExtracao','tipoDocumento']] = [4,'Proposicao s/ url inteiro teor']
19                 continue
20
21             response = sessao.get(url)
22             headers = response.headers
23             tipoDocumento = str(headers['content-type'])
24             statusExtracao=0
25             inteiroTeor =''
26             tokensInteiroTeor = ''
27             tamanhoConteudo = 0
28
29             if tipoDocumento =='application/pdf':
30                 inteiroTeor = extrairInteiroTeorPdf(response.content)
31                 tamanhoConteudo = len(response.content)
32                 statusExtracao = 1
33             elif tipoDocumento=='text/html;charset=utf-8':
34                 idTeor = url.split('=')[-1]
35                 response = sessao.get(f'https://www.camara.leg.br/internet/ordemdodia/integras/{idTeor}.htm')
36                 inteiroTeor = extrairInteiroTeorHtml(response.text)
37                 tamanhoConteudo = len(response.content)
38                 statusExtracao = 1

```

Fonte: Elaboração Própria

Figura 006 - Função de tratamento de PDF

```

1 def extrairInteiroTeorPdf(conteudo):
2     textoPaginas = ''
3     pdf = io.BytesIO(conteudo)
4     pdfReader = PyPDF2.PdfFileReader(pdf)
5     numeroPaginas = pdfReader.numPages
6
7     for p in range(numeroPaginas):
8         pagina = pdfReader.getPage(p)
9         textoPaginas += pagina.extractText()
10
11     return textoPaginas

```

Fonte: Elaboração Própria

Figura 007 - Função de tratamento de HTML

```

1 def extrairInteiroTeorHtml(conteudo):
2     TAG_HTML = re.compile(r'<[^>]+>')
3     return TAG_HTML.sub(' ',conteudo)

```

Fonte: Elaboração Própria

Uma vez que os dados foram obtidos foi preciso determinar qual o relacionamento entre eles. O *dataset* de PROPOSIÇÕES se relaciona com o *dataset* de TEMAS por meio dos campos “uri”, do primeiro *dataset*, e “uriProposicao” do segundo *dataset*. O relacionamento

do dataset de PROPOSIÇÕES com o dataset de INTEIRO TEOR é feito por meio dos campos “id” de ambos os datasets.

3. Processamento/Tratamento de Dados

Com os dados obtidos, foi necessário realizar o processamento e tratamentos dos dados com o objetivo de verificar a coerência e integridade dos mesmos, problemas comuns que impactam o processamento tais como codificação de caracteres, a maldição da dimensionalidade, dados nulos ou ausentes e outras mais específicas do problema de classificação temática das proposições.

Para esta tarefa, foi utilizado a linguagem de programação *Python*, na versão 3.9 com a biblioteca *pandas* (versão 1.4.1) para análise dos dados, a biblioteca *matplotlib* (versão 3.5.1) e *seaborn* (versão 0.11.2) para visualização dos dados.

Na etapa de obtenção de dados foi feita a consolidação dos arquivos anuais baixados do portal da Câmara dos Deputados, em outros dois arquivos de acordo com o conteúdo. Considerando isto, nesta etapa foram analisados os arquivos consolidados de PROPOSIÇÕES e CLASSIFICAÇÃO TEMÁTICA e o arquivo de INTEIRO TEOR que foi gerado utilizando o programa de extração criado para este fim. Este processo de extraiu em média 1 documento de inteiro teor a cada 2 segundos o que levou cerca de 70 horas para extrair 126.398 documentos de inteiro teor das proposições, que é a quantidade de proposições analisadas no período considerado para o estudo.

Figura 008 - Arquivos consolidados

Name	Tamanho	Tipo
proposicoes	--	Pasta
classificacaoFinal.csv	20,2 MB	Documento CSV
stop_words.csv	6 KB	Documento CSV
proposicoesUnificado.csv	1,64 GB	Documento CSV
inteiroTeor.csv	980,7 MB	Documento CSV
temas.csv	6,7 MB	Documento CSV
proposicoes.csv	106 MB	Documento CSV

Fonte: Elaboração própria

Considerando o objetivo do trabalho de fazer a classificação temática de uma proposição automaticamente com base em seu conteúdo, foi feita a análise para cada *dataset* para identificar quais campos contribuem para este objetivo ou que podem dar insumo para alguma análise sobre os conjuntos de dados. Desta forma foi possível determinar para cada *dataset* os campos que serão utilizados nas próximas fases do trabalho, conforme detalhado abaixo.

O dataset PROPOSIÇÕES apresentou 126.398 registros com 30 colunas, sendo que muitas delas apresentaram valores nulos ou que não agregariam ao objetivo do trabalho. Foi avaliado também a existência de registros duplicados e os mesmos são inexistentes.

Figura 009 - Análise valores nulos e registros duplicados no dataset de proposições

```
1 dfProposicoes = pd.read_csv(f'{DIR_DATASET}proposicoes.csv', sep=';', low_memory=False)

1 dfProposicoes.info()
2 qtdIdsDuplicados = dfProposicoes.duplicated(subset='id').sum()
3 print(f'\n[INFO ]Quantidade de IDs duplicados: {qtdIdsDuplicados}')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126398 entries, 0 to 126397
Data columns (total 31 columns):
```

#	Column	Non-Null Count	Dtype
0	id	126398	non-null
1	uri	126398	non-null
2	siglaTipo	126398	non-null
3	numero	126398	non-null
4	ano	126398	non-null
5	codTipo	126398	non-null
6	descricaoTipo	126398	non-null
7	ementa	122242	non-null
8	ementaDetalhada	1467	non-null
9	keywords	20047	non-null
10	dataApresentacao	126398	non-null
11	uriOrgaoNumerador	126398	non-null
12	uriPropAnterior	0	non-null
13	uriPropPrincipal	91030	non-null
14	uriPropPosterior	21	non-null
15	urlInteiroTeor	125762	non-null
16	urnFinal	0	non-null
17	ultimoStatus_dataHora	126398	non-null
18	ultimoStatus_sequencia	126398	non-null
19	ultimoStatus_uriRelator	13805	non-null
20	ultimoStatus_idOrgao	51943	non-null
21	ultimoStatus_siglaOrgao	51940	non-null
22	ultimoStatus_uriOrgao	51943	non-null
23	ultimoStatus_regime	23775	non-null
24	ultimoStatus_descricaoTramitacao	126398	non-null
25	ultimoStatus_idTipoTramitacao	126398	non-null
26	ultimoStatus_descricaoSituacao	51943	non-null
27	ultimoStatus_idSituacao	51943	non-null
28	ultimoStatus_despacho	126389	non-null
29	ultimoStatus_url	82658	non-null
30	anoDataset	126398	non-null

dtypes: float64(4), int64(7), object(20)
memory usage: 29.9+ MB

[INFO] Quantidade de IDs duplicados: 0

Fonte: Elaboração Própria

Em relação às colunas que apresentaram valores nulos ou irrelevantes, o tratamento aplicado foi a exclusão dos mesmos, sendo consideradas apenas os campos abaixo:

#	Nome da coluna/campo	Descrição	Tipo
1	id	Identificador único da proposição	Número
2	uri	Endereço eletrônico da proposição	Texto
3	siglaTipo	Sigla que identifica o tipo de proposição.	Texto

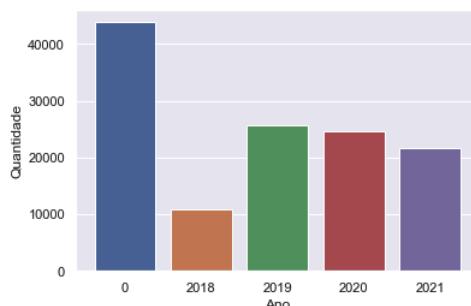
5	ano	Ano da proposição	Número
6	codTipo	Código identificador do tipo de proposição	Número
7	descricaoTipo	Descrição do tipo de proposição	Texto
8	ementa	Resumo da proposição	Texto
11	dataApresentacao	Data em que foi apresentada a proposição	Data e Hora
16	urlInteiroTeor	Endereço eletrônico do inteiro teor da proposição	Texto

Analizando os valores do campo "ano" do *dataset* de PROPOSIÇÕES, foi identificado que o mesmo não reflete o período estudado, pois apresenta valores zerados conforme demonstrado abaixo:

Figura 010 - Gráfico quantidade de proposições por ano

```
1 ax = sns.countplot(x='ano', data=dfProposicoes)
2 ax.set_xlabel('Ano')
3 ax.set_ylabel('Quantidade')
4 ax.plot()
```

[]



```
1 dfProposicoes.groupby(['ano', 'anoDataset']).size()
```

```
ano  anoDataset
0    2018        7792
     2019       12681
     2020        5173
     2021       18167
2018 2018       10920
2019 2019       25565
2020 2020       24503
2021 2021       21597
dtype: int64
```

Fonte: Elaboração própria

Para contornar esta situação e garantir maior assertividade nas análises, foi criado um novo campo no *dataset* chamado "anoDataset" que foi preenchido com o ano presente no nome dos arquivos. Para isto a função "gerarDatasetProposicoes" conforme abaixo e o resultado pode ser observado no gráfico de quantidade de proposições por ano:

Figura 011 - Função para gerar o dataset de proposições consolidado

```
def gerarDataframeProposicoes():
    dfProposicoesList = []

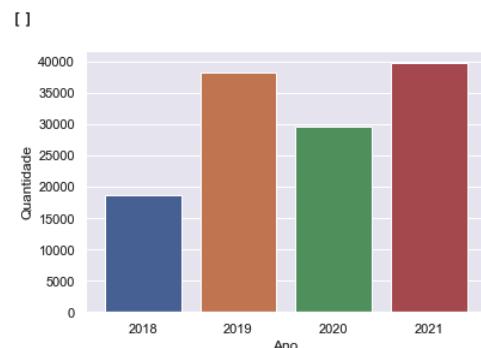
    for arquivo in os.listdir(f'{DIR_DATASET}proposicoes/'):
        try:
            anoDataset = arquivo.split('.')[0][-4:]
            df = pd.read_csv(f'{DIR_DATASET}proposicoes/{arquivo}', sep=';', low_memory=False)
            df.loc[:, 'anoDataset'] = anoDataset
            dfProposicoesList.append(df)
            print(f'[OK] [{anoDataset}] {arquivo}')
        except Exception as e:
            print(f'[ERRO] {arquivo}')

    return pd.concat(dfProposicoesList)
```

Fonte:Elaboração própria

Figura 012 - Gráfico de proposições por ano do dataset

```
1 ax = sns.countplot(x='anoDataset', data=dfProposicoes)
2 ax.set_xlabel('Ano')
3 ax.set_ylabel('Quantidade')
4 ax.plot()
```



```
1 dfProposicoes.groupby(['anoDataset']).size()
anoDataset
2018    18712
2019    38246
2020    29676
2021    39764
dtype: int64
```

Fonte: Elaboração própria

O dataset de classificação temática apresentou 53.268 registros com 8 colunas e nenhuma delas continham valores nulos, contudo algumas informações como "codTema" e

"tema" são redundantes podendo assim algumas delas serem desconsideradas para o modelo de machine learning.

Figura 012 - Campos do dataset temas das proposições

```
1 | dfTemasProposicoes.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53268 entries, 0 to 53267
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   uriProposicao    53268 non-null   object  
 1   siglaTipo        53268 non-null   object  
 2   numero          53268 non-null   int64   
 3   ano              53268 non-null   int64   
 4   codTema          53268 non-null   int64   
 5   tema             53268 non-null   object  
 6   relevancia       53268 non-null   int64   
 7   anoDataset       53268 non-null   int64   
dtypes: int64(5), object(3)
memory usage: 3.3+ MB
```

Fonte: Elaboração própria

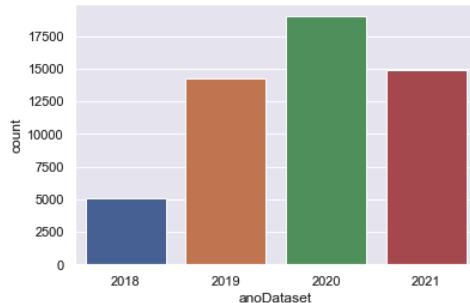
Algumas informações foram consideradas irrelevantes para o objetivo proposto, então abaixo estão listados os campos deste *dataset* que foram utilizados:

#	Nome da coluna/campo	Descrição	Tipo
1	uriProposicao	Endereço eletrônico da proposição	Texto
2	siglaTipo	Sigla que identifica o tipo de proposição.	Texto
4	ano	Ano da proposição	Número
6	tema	Nome do tema atribuído à proposição	Texto

Ao contrário do dataset de PROPOSIÇÕES, o de CLASSIFICAÇÃO TEMÁTICA não apresentou divergência entre o ano informado e o ano de extração, ainda assim foi incluído no dataset a coluna "anoDataset" para fins de análise e comparação.

Figura 013 - Gráfico temas de proposição por ano

```
1 ax = sns.countplot(x='anoDataset', data=dfTemasProposicoes)
```



```
1 dfTemasProposicoes.groupby(['ano', 'anoDataset']).size()
```

Ano	Ano Dataset	Count
2018	2018	5097
2019	2019	14251
2020	2020	19004
2021	2021	14916

Fonte: Elaboração própria

A informação mais relevante para este trabalho são os temas que foram atribuídos à uma proposição, que foram utilizados para classificar proposições que estão sem tema definido. Constam no dataset 32 temas possíveis no período de estudo e foi possível identificar que uma mesma proposição pode ser classificada em mais de uma tema. Esta situação não configura um problema para o modelo, visto que ele se baseará na frequência dos termos para fazer a classificação.

Figura 014 - Exemplo de quantidade de temas de uma proposição

```
1 dfTemasProposicoes.groupby(['uriProposicao']).size()
```

URI Proposicao	Count
https://dadosabertos.camara.leg.br/api/v2/proposicoes/1299572	3
https://dadosabertos.camara.leg.br/api/v2/proposicoes/1299577	1
https://dadosabertos.camara.leg.br/api/v2/proposicoes/1555295	1
https://dadosabertos.camara.leg.br/api/v2/proposicoes/2057848	2
https://dadosabertos.camara.leg.br/api/v2/proposicoes/2057934	2
..	..
https://dadosabertos.camara.leg.br/api/v2/proposicoes/593065	1
https://dadosabertos.camara.leg.br/api/v2/proposicoes/601739	2
https://dadosabertos.camara.leg.br/api/v2/proposicoes/614512	1
https://dadosabertos.camara.leg.br/api/v2/proposicoes/946475	2
https://dadosabertos.camara.leg.br/api/v2/proposicoes/946514	1

Fonte: Elaboração própria

Figura 015 - Lista de temas e quantidade classificada em cada um

```

1 dfTemasProposicoes.groupby(['tema']).size()

tema
Administração Pública          4957
Agricultura, Pecuária, Pesca e Extrativismo 1030
Arte, Cultura e Religião        651
Cidades e Desenvolvimento Urbano 993
Ciência, Tecnologia e Inovação   544
Ciências Exatas e da Terra      3
Ciências Sociais e Humanas      6
Comunicações                     1979
Defesa e Segurança               2210
Direito Civil e Processual Civil 1007
Direito Constitucional            159
Direito Penal e Processual Penal 2033
Direito e Defesa do Consumidor   1183
Direito e Justiça                 296
Direitos Humanos e Minorias       5193
Economia                         1727
Educação                          3011
Energia, Recursos Hídricos e Minerais 1361
Esporte e Lazer                  451
Estrutura Fundiária              253
Finanças Públicas e Orçamento    4529
Homenagens e Datas Comemorativas 975
Indústria, Comércio e Serviços     1748
Meio Ambiente e Desenvolvimento Sustentável 1725
Política, Partidos e Eleições      537
Previdência e Assistência Social   1565
Processo Legislativo e Atuação Parlamentar 535
Relações Internacionais e Comércio Exterior 636
Saúde                            6633
Trabalho e Emprego                3019
Turismo                           218
Viação, Transporte e Mobilidade    2101
dtype: int64

```

Fonte: Elaboração própria

A estratégia inicial do projeto foi trabalhar com o campo "ementa" do *dataset* de PROPOSIÇÕES que é um resumo da proposição, contudo este campo apresentou 122.442 registros com o campo preenchido, ou seja algumas proposições não possuíam a informação. Somando isto ao fato de que o *dataset* de CLASSIFICAÇÃO TEMÁTICA possuir apenas 30.491 proposições classificadas (24%), foi adotada uma nova abordagem, que consistiu em utilizar todo o texto da proposição presente no documento de inteiro teor. Por meio da função demonstrada anteriormente, tentou-se extrair o inteiro teor de 125.762 documentos e o resultado desta extração será detalhado na sequência. Embora não altere o total de proposições classificadas, trabalhar com o inteiro teor das proposições aumentou o volume de texto a ser analisado para proceder com a classificação, aumentando assim a eficiência do modelo de classificação aplicado.

Figura 016 - Quantidade de proposições classificadas

```

1 dfPreposicoesTemas = pd.merge(dfTemasProposicoes, dfProposicoes, left_on='uriProposicao', right_on='uri')
2 qtdProposicoesClassificadas = len(dfPreposicoesTemas.groupby(['id']).nunique())
3 print(f'[INFO] Total de proposições classificadas: {qtdProposicoesClassificadas}')

```

[INFO] Total de proposições classificadas: 30491

Fonte: Elaboração própria

O dataset INTEIRO TEOR apresentou a mesma quantidade de registros do dataset PROPOSIÇÕES, 126.398, com seis colunas.

Figura 017 - Informações dataset inteiro teor

```
1 dfInteiroTeor.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126398 entries, 0 to 126397
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          126398 non-null   int64  
 1   urlInteiroTeor  125762 non-null   object 
 2   statusExtracao 126398 non-null   int64  
 3   tipoDocumento  126398 non-null   object 
 4   tamanhoEmBytes 126398 non-null   int64  
 5   inteiroTeor    118204 non-null   object 
dtypes: int64(3), object(3)
memory usage: 5.8+ MB
```

Fonte: Elaboração própria

No processo de extração, foi identificado que 636 proposições não apresentavam valor no campo "urlInteiroTeor", impossibilitando assim o processo de extração dos dados necessários. A estes casos foi atribuído o valor 4 ao campo "statusExtracao" e ao "tipoDocumento" o valor "Proposicao s/ url inteiro teor".

Eram esperados erros no processo de extração, tais como erro de conexão de rede, erro de indisponibilidade do portal da Câmara dos Deputados, porém foram identificados outros problemas durante o processo e a estes erros foram atribuídos os valores:

- 2 → para o caso do tipo de documento retornado do portal fosse diferente de um arquivo *.pdf ou *.htm.
- 3 → para erros não mapeados.
- 5 → imagem salva como pdf.

Para os casos em que a extração pôde ser realizada com sucesso, foi atribuído o valor 1 ao campo "statusExtracao", o tamanho do documento em *bytes* e conteúdo em si foram gravados nos campos "tamanhoEmBytes" e "inteiroTeor" respectivamente.

Figura 018 - Resultado extração do inteiro teor

```
1 dfInteiroTeor.groupby(['statusExtracao']).count()
```

	id	urlInteiroTeor	tipoDocumento	tamanhoEmBytes	inteiroTeor
statusExtracao					
1	125595	125595	125595	125595	118204
2	97	97	97	97	0
3	70	70	70	70	0
4	636	0	636	636	0

Fonte: Elaboração própria

Ao analisar o *dataset* gerado com o conteúdo dos documentos, foi identificado que 7.391 proposições embora apresentasse a URL para extração do inteiro teor, e o tipo de documento retornado foi um arquivo *.pdf, elas não tiveram o campo "inteiroTeor" preenchido. Investigando o ocorrido, foi identificado que se trata de documentos escaneados, o que os torna como uma imagem. A extração de texto destes documentos necessitaria do uso da tecnologia de Reconhecimento Ótico de Caracteres, ou OCR (Optical Character Recognition), contudo esta abordagem não foi utilizada e decidiu-se por atribuir o valor 5 ao campo "statusExtracao".

Figura 019 - Proposições com url de inteiro teor e sem conteúdo

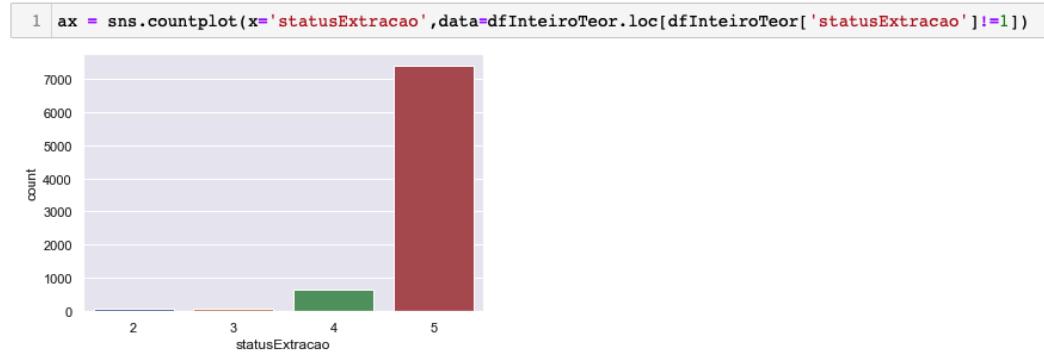
```
dfInteiroTeor.loc[(dfInteiroTeor['statusExtracao']==1)&(dfInteiroTeor['inteiroTeor'].isnull())]
```

id	urlInteiroTeor	statusExtracao	tipoDocumento	tamanhoEmBytes	inteiroTeor	
18	2080812	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	53091	NaN
19	2111588	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	968013	NaN
326	2190692	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	572179	NaN
2304	2193133	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	25333	NaN
2373	2193216	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	695994	NaN
...	
126393	2316058	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	143370	NaN
126394	2316178	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	319115	NaN
126395	2316179	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	154366	NaN
126396	2316200	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	175666	NaN
126397	2316251	http://www.camara.gov.br/proposicoesWeb/prop_m...	1	application/pdf	101789	NaN

7391 rows x 6 columns

Fonte: Elaboração própria

Figura 020 - Gráfico status da extração diferente de sucesso



Fonte: Elaboração própria

Figura 021 -Tabela de status de extração e quantidade

```
1 dfInteiroTeor.groupby(['statusExtracao']).count()
```

	id	urlInteiroTeor	tipoDocumento	tamanhoEmBytes	inteiroTeor
statusExtracao					
1	118204	118204	118204	118204	118204
2	97	97	97	97	0
3	70	70	70	70	0
4	636	0	636	636	0
5	7391	7391	7391	7391	0

Fonte: Elaboração própria

Figura 022 - Tabela tipos de documento e quantidades

```
1 dfInteiroTeor.groupby(['tipoDocumento']).count()
```

		id	urlInteiroTeor	statusExtracao	tamanhoEmBytes	inteiroTeor
	tipoDocumento					
	('/Contents',)	4	4	4	4	0
	('Cannot read an empty file',)	52	52	52	52	0
	('Error -5 while decompressing data: incomplete or truncated stream',)	1	1	1	1	0
	('Multiple definitions in dictionary at byte 0x2c8 for key /PageLabels',)	1	1	1	1	0
	('Multiple definitions in dictionary at byte 0x2e8 for key /PageLabels',)	1	1	1	1	0
	(ProtocolError('Connection aborted.', OSError(50, 'Network is down')),)	1	1	1	1	0
	(ProtocolError('Connection aborted.', RemoteDisconnected('Remote end closed connection without response')),)	7	7	7	7	0
	(ReadTimeoutError("HTTPConnectionPool(host='www.camara.gov.br', port=80): Read timed out. (read timeout=None)",),	1	1	1	1	0
	(ReadTimeoutError("HTTPSConnectionPool(host='www.camara.leg.br', port=443): Read timed out. (read timeout=None)",),	1	1	1	1	0
	([<class 'decimal.ConversionSyntax'\>],)	1	1	1	1	0
	Proposicoes s/ url inteiro teor	636	0	636	636	0
	application/msword	97	97	97	97	0
	application/pdf	118108	118108	118108	118108	118108
	imagem salva como pdf	7391	7391	7391	7391	0
	text/html;charset=utf-8	96	96	96	96	96

Fonte: Elaboração própria

A quantidade de erros encontrados no processo de extração foi relativamente baixa em relação à quantidade que teve sucesso na extração, permitindo assim o uso dos dados extraídos.

Figura 023 - Tabela percentual status da extração

```
1 dfPercentualStatus = dfInteiroTeor.groupby(['statusExtracao']).count()
2 dfPercentualStatus['%'] = dfPercentualStatus['id']/len(dfInteiroTeor)*100
3 dfPercentualStatus[['id','%']]
```

	id	%
	statusExtracao	
1	118204	93.517302
2	97	0.076742
3	70	0.055381
4	636	0.503173
5	7391	5.847403

Fonte: Elaboração própria

Após a análise e tratamentos dos dados do dataset de INTEIRO TEOR, chegou-se ao total de 118.204 proposições que tiveram o texto conteúdo do documento extraído e processado. Estes documentos totalizaram um total de 108.38 Gigabytes de dados.

Figura 024 - Tamanho dos conteúdos extraídos

```

1 dfTamanhoConteudo = dfInteiroTeor.loc[dfInteiroTeor['statusExtracao']==1]\n2 .groupby(['tipoDocumento'])\n3 .agg({'tamanhoEmBytes': 'sum'})\n4\n5 tamanhoEmGigas = dfTamanhoConteudo.agg({'tamanhoEmBytes': 'sum'})/1e+9\n6 tamanhoEmGigas = tamanhoEmGigas['tamanhoEmBytes']\n7\n8 print(f'[INFO      ] Tamanho dos documentos extraídos e processados: {tamanhoEmGigas:.2f} GB')\n9 dfTamanhoConteudo

```

[INFO] Tamanho dos documentos extraídos e processados: 108.38 GB

tamanhoEmBytes	
tipoDocumento	
application/pdf	108380121493
text/html;charset=utf-8	266750

Fonte: Elaboração própria

Ao se trabalhar com processamento de linguagem natural em machine learning, uma etapa importante do processo é remover palavras pouco relevantes, também conhecidas por *stopwords* como por exemplo artigos, preposições, pronomes e também palavras irrelevantes dentro do contexto estudado. No caso deste estudo, além das "stopwords" em português (cerca de 550 palavras) foram incluídas para remoção palavras muitos comuns como "projeto", "lei", "deputado" com o objetivo de otimizar a etapa de processamento do modelo de machine learning.

Figura 025 - Arquivo de stopwords em português

stop_words.csv	
1	"stop_words"
2	"a"
3	"à"
4	"adeus"
5	"agora"
6	"é"
7	"ainda"
8	"além"
9	"algo"
10	"algum"
11	"algum"
12	"alguma"
13	"algumas"
14	"alguns"
15	"ali"
581	"p"
582	"q"
583	"r"
584	"s"
585	"t"
586	"u"
587	"v"
588	"x"
589	"z"
590	"g"
591	"ç"
592	"á"
593	"projeto"
594	"lei"
595	"deputado"
596	"câmara"
597	"deputados"

Fonte: Elaboração própria

Outros procedimentos relevantes para o tratamento de textos é trabalhar com palavras "tokenizadas", ou seja uma lista de palavras relevantes contidas no texto e com todas as letras em minúsculo para evitar ambiguidade. Para este trabalho foi criada uma função que recebe como parâmetro de entrada o texto e retorno o texto livre das stopwords (indicadas no arquivo "stop_words.csv"), pontuação, caracteres especiais, quebras de linha e números.

Figura 026: Código da função “tokenizarTexto”

```
1 def tokenizarTexto(texto: str, listaStopWords=[]):
2     try:
3         tamanhoTexto = len(texto.strip().split())
4
5         pontuacao = ['{', '}', '(', ')', ';', ':', '[', ']', '#', '%', '*', '<', '>', '=', '?',
6                     ',', '/', '!', '$', '?', '!', 'o', 'S', '+', '@',
7                     'g', '!', '!', '\\", \\", E', '.', '&', '&', '&', '&', '&', '&', '&', '&', '&', '&']
8         texto = texto.replace('\\n', ' ')
9
10        for ponto in pontuacao:
11            texto = texto.replace(ponto, ' ')
12
13        for numero in range(0, 10):
14            texto = texto.replace(str(numero), ' ')
15
16        tokens = texto.strip().split()
17        palavrasChave = [palavra for palavra in tokens if not palavra.lower() in listaStopWords and len(palavra)>3]
18
19
20        return (' '.join(palavrasChave).lower(), tamanhoTexto, len(palavrasChave))
21    except Exception as e:
22        print(f'[ERRO] Erro ao tokenizar texto: "{texto}" \n{e}')
23        raise e
```

Fonte: Elaboração própria

Visando facilitar a etapa de construção do modelo de *machine learning*, decidiu-se por criar um *dataset* específico para ser usado nesta etapa, contendo todas as proposições, a classificação temática e o inteiro teor das proposições.

Os datasets de PROPOSIÇÕES, CLASSIFICAÇÃO TEMÁTICA e INTEIRO TEOR apresentavam os seguintes relacionamentos:

- PROPOSIÇÕES x CLASSIFICAÇÃO TEMÁTICA: campo "uri" com campo "uriProposicao" respectivamente.
 - PROPOSIÇÕES x INTEIRO TEOR: campo "id" com campo "id"
 - CLASSIFICAÇÃO TEMÁTICA x INTEIRO TEOR: não possuíam relação direta, contudo foi possível fazer a referência cruzada utilizando os relacionamentos com PROPOSIÇÕES.

Com base nestes relacionamentos, e nos campos mais relevantes dos *datasets* foi possível construir o dataset PROPOSIÇÕES UNIFICADO que foi composto por todos os registros do dataset PROPOSIÇÕES e INTEIRO TEOR e a CLASSIFICAÇÃO TEMÁTICA existente. Para realização desta tarefa foi utilizado a função "merge" da biblioteca *pandas*, que permite fazer a junção de dois *datasets* que possuem algum campo chave em comum.

Como já foram identificadas as colunas principais dos datasets, por questão de performance optou-se por abrir os *datasets* novamente trazendo apenas estes campos e aqueles que são chaves entre os datasets.

Figura 027: Código para abertura dos arquivos com os campos importantes

```

1 colunasProposicao=['id','uri','ementa','descricaoTipo','anoDataset']
2 colunasTemas=['uriProposicao','tema']
3 colunasInteiroTeor=['id','urlInteiroTeor','statusExtracao','tipoDocumento','tamanhoEmBytes','inteiroTeor']
4
5 dfProposicoes = pd.read_csv(f'{DIR_DATASET}proposicoes.csv', sep=';', low_memory=False, usecols=colunasProposicao)
6 dfTemasProposicoes = pd.read_csv(f'{DIR_DATASET}temas.csv', sep=';', low_memory=False, usecols=colunasTemas)
7 dfInteiroTeor = pd.read_csv(f'{DIR_DATASET}inteiroTeor.csv', sep=';', low_memory=False, usecols=colunasInteiroTeor)
8
9 dfProposicoes.info()
10 dfTemasProposicoes.info()
11 dfInteiroTeor.info()

```

Fonte: Elaboração própria

Figura 028: Informações *data frame* proposições

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126398 entries, 0 to 126397
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          126398 non-null   int64  
 1   uri         126398 non-null   object  
 2   descricaoTipo 126398 non-null   object  
 3   ementa      122242 non-null   object  
 4   anoDataset  126398 non-null   int64  
dtypes: int64(2), object(3)
memory usage: 4.8+ MB

```

Fonte: Elaboração própria

Figura 029: Informações *data frame* temas da proposição

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53268 entries, 0 to 53267
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   uriProposicao 53268 non-null   object  
 1   tema         53268 non-null   object  
dtypes: object(2)
memory usage: 832.4+ KB

```

Fonte: Elaboração própria

Figura 030: Informações *data frame* inteiro teor

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126398 entries, 0 to 126397
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          126398 non-null   int64  
 1   urlInteiroTeor 125762 non-null   object  
 2   statusExtracao 126398 non-null   int64  
 3   tipoDocumento  126398 non-null   object  
 4   tamanhoEmBytes 126398 non-null   int64  
 5   inteiroTeor    118204 non-null   object  
dtypes: int64(3), object(3)
memory usage: 5.8+ MB
```

Fonte: Elaboração própria

A função "merge" da biblioteca *pandas* permite fazer a junção de dois datasets por vez, então a estratégia adotada foi criar um *data frame* auxiliar que é resultado da combinação dos datasets PROPOSIÇÃO e INTEIRO TEOR, usando como campos chave o "id" de ambos os datasets.

Figura 031: Informações *data frame* auxiliar

```
1 dfAuxiliar = pd.merge(dfProposicoes,dfInteiroTeor, left_on='id',right_on='id')
2 dfAuxiliar.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 126398 entries, 0 to 126397
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          126398 non-null   int64  
 1   uri         126398 non-null   object  
 2   descricaoTipo 126398 non-null   object  
 3   ementa      122242 non-null   object  
 4   anoDataset  126398 non-null   int64  
 5   urlInteiroTeor 125762 non-null   object  
 6   statusExtracao 126398 non-null   int64  
 7   tipoDocumento 126398 non-null   object  
 8   tamanhoEmBytes 126398 non-null   int64  
 9   inteiroTeor    118204 non-null   object  
dtypes: int64(4), object(6)
memory usage: 10.6+ MB
```

Fonte: Elaboração própria

Com o *data frame* auxiliar criado, o próximo passo foi combiná-lo com o *data frame* CLASSIFICAÇÃO TEMÁTICA, usando como chave os campos "uri" e "uriProposicao" respectivamente. Já era sabido que nem todas as proposições estão classificadas, então quando a função "merge" é utilizada, por padrão ela retorna apenas os registros que coincidem, contudo para objetivo do trabalho foi preciso que a junção mantivesse todas as proposições e incluisse as informações dos temas e para aquelas que não estão classificadas

mantivesse os valores nulos. Para isto foi preciso usar o parâmetro "how" da função "merge" com o valor "left".

Figura 032: Informações *data frame* proposições

```

1 dfUnificado = pd.merge(dfAuxiliar,dfTemasProposicoes, left_on='uri',right_on='uriProposicao',how='left')
2 dfUnificado['tokens'] = np.NaN
3 dfUnificado['semConteudo'] = pd.isna(dfUnificado['ementa']) & pd.isna(dfUnificado['inteiroTeor'])
4 dfUnificado.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 149175 entries, 0 to 149174
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          149175 non-null   int64  
 1   uri         149175 non-null   object  
 2   descricaoTipo 149175 non-null   object  
 3   ementa       145019 non-null   object  
 4   anoDataset   149175 non-null   int64  
 5   urlInteiroTeor 148537 non-null   object  
 6   statusExtracao 149175 non-null   int64  
 7   tipoDocumento 149175 non-null   object  
 8   tamanhoEmBytes 149175 non-null   int64  
 9   inteiroTeor    140584 non-null   object  
10  uriProposicao  53268 non-null   object  
11  tema          53268 non-null   object  
12  tokens         0 non-null     float64 
13  semConteudo    149175 non-null   bool    
dtypes: bool(1), float64(1), int64(4), object(8)
memory usage: 16.1+ MB

```

Fonte: Elaboração própria

Como é possível constatar pela imagem acima , os campos "ementa" e "inteiroTeor" possuem valores nulos, logo os registros que não apresentam as duas informações não podem ser utilizados no modelo de *machine learning*. Para facilitar a identificação destes casos, foi criada uma coluna com o nome "semConteudo" que foi preenchida com o valor "True" quando ambos os campos são nulos, ou com "False" quando pelo menos um deles não é nulo.

Figura 033: Resumo da informação sobre o conteúdo da proposição

```

1 dfUnificado.groupby(['semConteudo']).size().sort_values(ascending=False).head(50)

semConteudo
False    148186
True     989
dtype: int64

```

Fonte: Elaboração própria

A última etapa do processamento de dados consistiu em sanitizar o texto contido nos campos "ementa" e/ou "inteiroTeor". Para isto, a estratégia adotada, para garantir uma resiliência a erros garantindo o processo de retomada do processamento em caso de falhas, foi criar uma função chamada "tokenizarDataset" que iterou por todos os 148.186 registros

que possuem conteúdo e para cada um deles fez a "tokenização" do texto, eliminando tudo que foi considerado irrelevante (pontuações, caracteres especiais, números), e gravando o resultado no campo chamado "tokens" e a quantidade de palavras no campo chamado "qtdTokens" gerando ao final um arquivo chamado "proposicoesUnificado.csv" com o resultado.

Figura 034: Código da função “tokenizarDataset”

```

1 def tokenizarDataset(df,qtdLimite=0):
2     try:
3         dfStopWords = pd.read_csv(f'{DIR_DATASET}stop_words.csv', usecols=['stop_words']) \
4             .replace(to_replace=[r"\t|\n|\r", "\t|\n|\r"], value=["", ""])
5
6         listaStopWords = dfStopWords['stop_words'].values.tolist()
7         print(f'[INFO] Quantidade de stop words: {len(listaStopWords)})')
8
9         dfParaExtracao = df.loc[df['tokens'].isnull()]
10        print(f'[INFO] Proposicoes para tokenização: {len(dfParaExtracao)})')
11
12        contador = 0
13        contadorNulos = 0
14        t0 = time()

15        for index,linha in dfParaExtracao.iterrows():
16            if not linha['semConteudo']:
17                ementa = '' if pd.isna(linha['ementa']) else linha['ementa']
18                inteiroTeor = '' if pd.isna(linha['inteiroTeor']) else linha['inteiroTeor']
19                idProposicao = linha['id']
20
21                tokens, tamanhoInicial, tamanhoFinal = tokenizarTexto(f'{ementa} {inteiroTeor}',listaStopWords)
22
23                df.loc[df['id']==idProposicao,['ementa',
24                                              'inteiroTeor',
25                                              'tokens']] = [ementa.strip(),inteiroTeor.strip(),tokens]
26
27
28            if contador%1000==0:
29                zeros = (6-len(str(contador)))*'0'
30                print(f'[INFO] [{zeros}{contador}] {idProposicao} Palavras: {tamanhoInicial} \
31 | Tokens: {tamanhoFinal} \
32 | Redução: {(tamanhoInicial-tamanhoFinal)/tamanhoInicial*100:.0f}%')
33
34            if contador%10000==0:
35                df.to_csv(f'{DIR_DATASET}/proposicoesUnificado.csv',
36                          index=False,sep=';',quoting=csv.QUOTE_ALL, escapechar="\\")
37                print('.' * 80 )
38
39            if qtdLimite>0 and contador ==qtdLimite:
40                break
41
42            contador+=1
43        else:
44            contadorNulos+=1

45        duracao = time()-t0
46        print(f'[INFO] Tokenizado {contador} proposicoes em {duracao:.0f} segundos. Media={(contador/duracao):.2f}')
47        print(f'[INFO] Proposicoes sem ementa e sem inteiro teor: {contadorNulos} ')
48
49
50
51    except Exception as e:
52        print(f'[ERRO] Erro ao tokenizar dataset: {e.args}')
53        print(f'[ERRO] Ementa: {ementa} \n Teor: {inteiroTeor}')
54        raise e
55
56    finally:
57        df.to_csv(f'{DIR_DATASET}/proposicoesUnificado.csv',index=False,sep=';',quoting=csv.QUOTE_ALL, escapechar="\\")

```

Fonte: Elaboração própria

Este processamento reduziu em média de 50% a 80% a quantidade de palavras dos textos dos campos "ementa" e "inteiroTeor", melhorando assim a performance para etapa de construção do modelo de *machine learning*.

Figura 035: Amostra do resultado da função “tokenizarDataset”

```
1 tokenizarDataset(dfUnificado)

[INFO] Quantidade de stop words: 596
[INFO] Proposicoes para tokenização: 129168
[INFO] [000000][2211661]Palavras: 252 | Tokens: 124 | Redução: 51%
.....
[INFO] [001000][2212825]Palavras: 8 | Tokens: 1 | Redução: 88%
[INFO] [002000][2214015]Palavras: 542 | Tokens: 227 | Redução: 58%
[INFO] [003000][2215028]Palavras: 720 | Tokens: 278 | Redução: 61%
[INFO] [004000][2216065]Palavras: 1385 | Tokens: 567 | Redução: 59%
[INFO] [005000][2217390]Palavras: 571 | Tokens: 274 | Redução: 52%
[INFO] [006000][2218457]Palavras: 429 | Tokens: 177 | Redução: 59%
[INFO] [007000][2219564]Palavras: 165 | Tokens: 69 | Redução: 58%
[INFO] [008000][2220644]Palavras: 407 | Tokens: 188 | Redução: 54%
[INFO] [009000][2221719]Palavras: 453 | Tokens: 218 | Redução: 52%
[INFO] [010000][2222863]Palavras: 106 | Tokens: 12 | Redução: 89%
.....
[INFO] [121000][2306865]Palavras: 61 | Tokens: 33 | Redução: 46%
[INFO] [122000][2307881]Palavras: 513 | Tokens: 131 | Redução: 74%
[INFO] [123000][2308754]Palavras: 85 | Tokens: 20 | Redução: 76%
[INFO] [124000][2309795]Palavras: 2015 | Tokens: 903 | Redução: 55%
[INFO] [125000][2310788]Palavras: 186 | Tokens: 17 | Redução: 91%
[INFO] [126000][2311765]Palavras: 34 | Tokens: 2 | Redução: 94%
[INFO] [127000][2312719]Palavras: 104 | Tokens: 14 | Redução: 87%
[INFO] [128000][2313641]Palavras: 51 | Tokens: 1 | Redução: 98%
[INFO] Tokenizado 128179 proposicoes em 6870 segundos. Media=18.66 prop/seg
[INFO] Proposições sem ementa e sem inteiro teor: 989
```

Fonte: Elaboração própria

Figura 036: Exemplo de proposição após tokenização

dfUnificado.loc[dfUnificado['id']==2211661][['id', 'tokens', 'descricaoTipo', 'tema', 'inteiroTeor']]						
	Id	tokens	descricaoTipo	tema	inteiroTeor	
20327	2211661	requer senhor ministro ministério cidadania informação acerca planejamento estratégico ministerio requerimento deputada tabata amaral requer senhor ministro ministério cidadania informação acerca planejamento estratégico ministéri senior presidente senhor presidente equerido termos constituição federal regime interno prestadas senhor ministro ministério cidadania informações planejamento estratég ministério termos requisita informações prazo formulação planejamento estratégico ministério caso formulado esclarecimentos respeito metas definidas programa secretaria órgão vinculado disponibilização documento contendo tais informações justificação elaboração execução planejamento estra tégo coaduna princípio constitucional eficiência gestão pública constituição federal planejamentos estratégicos ministérios governo federal ferramentas essenciais definição priorização atividades serem desenvolvidas órgão período delimitado tempo acordo cultura prática planejamento estratégico setorial contribui melhorar desenho programas ações conferir clareza objetivos serem perseguidos órgãos aperfeiçoar capacidade gestão tomada decisão sala sessões julho deputad tabata amaral	Requerimento de Informação	Administração Pública	CÂMARA/\nDOS/\nDEPUTADOS/\n\nREQUERIMENTO Nº ,\nDE 2019/\nDa Sra. Deputada Tabata Amaral PDT/SP) /\nRequer a Senhor Ministro d'vno/\nMinistério da Cidadania, informação vñacerca do Planejamento Estratégico do \nMinistério.\n\n\n vñSenhor(a) Presidente,\nvñSenhor Presidente,\n\nnequeiro, nos termos do art. 50, § 2º, da Constituição Federal e do art. 115 e 116 do Regimento Interno Câmara dos Deputados, que vñsejam prestadas, pelo Senhor Ministro do Ministério da Cidadania, informações vñsobre o Planejamento Estratégico do Ministério. Nesses termos, requisita/vñm/vñse:\n\n\n. Informações sobre o prazo para a formulação do Planejamento Estratégico do Ministério;\n\n\n. Caso já tenha sido formulado, esclarecimentos a respeito das metas vñdefinidas para cada programa, Secretaria e órgão vinculado e a vñdisponibilização do documento contendo tais informações.\n\n\n.JUSTIFICAÇÃO\nA elaboração e execução de planejamento estratégico coadunav\nse com vñ princípio constitucional de eficiência na gestão Pública (Art. 37 da Constituição Federal).\n\nOs Planejamentos Estratégicos dos Ministérios do Governo Federal são vñferramentas essenciais para definição e priorização de atividades vñ serem vñdesenvolvidas pelo órgão em um período delimitado no tempo. De acordo com vñnde cultura e prática de planejamento estratégico setorial contribui para: vñ vñ\n(vñ) melhorar desenho dos vñprogramas e ações;\n(vñb) conferir maior clareza aos objetivos a serem perseguidos pelos vñórgãos;\n(vñc) aperfeiçoe a capacidade de gestão e a tomada de decisão.\n\n\n.Sala das vñSessões\n, em vñ08vñ /de vñJulho/vñ /de vñ2019vñ.\n\nDeputad\nna Tabata Amaral\n\nPDT\nvñSP	

Fonte: Elaboração própria

4. Análise e Exploração dos Dados

Uma vez findada a etapa de processamento dos dados, foi iniciada a análise e exploração dos dados gerados na etapa anterior, mais especificamente o *dataset* "proposicoesUnificado.csv". Neste dataset foram mantidas as colunas "ementa" e "inteiroTeor", contudo não foram utilizadas na análise pois elas possuem dados brutos, antes do tratamento realizado. Iremos trabalhar com os seguintes campos:

Figura 037: Código para abertura do arquivo proposições unificado

```

1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 %matplotlib inline
7 #pd.set_option('display.max_colwidth', -1)
8 sns.set_theme(style="darkgrid")
9
10 DIR_DATASET = r'/Users/dev-rocks/Documents/TCC - Data Science e Big Data/Projeto/datasets/'
11 # DIR_DATASET = r'C:\Users\Daniel.Vale\Personal\PBDC\TCC\Projeto_v2\datasets\\'

1 colunas = ['id','descricaoTipo','anoDataset','statusExtracao',
2             'tipoDocumento','tamanhoEmBytes','tema','tokens',
3             'qtdTokens','semConteudo']
4
5 dfProposicoes = pd.read_csv(f'{DIR_DATASET}proposicoesUnificado.csv', sep=';', low_memory=False,usecols=colunas)
6

```

Fonte: Elaboração própria

Figura 038: Informações *data frame* proposições unificado

```

1 dfProposicoes.info()
2

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149175 entries, 0 to 149174
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
0    id          149175 non-null   int64  
1    descricaoTipo 149175 non-null   object  
2    anoDataset   149175 non-null   int64  
3    statusExtracao 149175 non-null   int64  
4    tipoDocumento 149175 non-null   object  
5    tamanhoEmBytes 149175 non-null   int64  
6    tema         53268 non-null    object  
7    tokens       147784 non-null    object  
8    semConteudo  149175 non-null   bool    
9    qtdTokens    149175 non-null   int64  
dtypes: bool(1), int64(5), object(4)
memory usage: 10.4+ MB

```

Fonte: Elaboração própria

A primeira análise realizada foi entender o comportamento da quantidade de proposições por ano. Foi possível perceber um aumento significativo do ano de 2018 para o ano de 2019. Considerando que em 2019 se iniciou a pandemia do novo coronavírus, é plausível deduzir que esta é a causa do aumento da atividade parlamentar.

Figura 039: Gráfico de proposições por ano

```

1 proposicoesPorAno = dfProposicoes.groupby(['anoDataset'], as_index=False)[['id']].count()
2 f, ax = plt.subplots(figsize=(10, 5))
3
4 ax = sns.lineplot(x='anoDataset',y='id',data=proposicoesPorAno)
5 ax.set_title('Proposições por ano')
6 ax.set_xlabel('Ano')
7 ax.set_ylabel('Quantidade')
8 ax.plot()
9
10 proposicoesPorAno

```

	anoDataset	id
0	2018	20192
1	2019	42596
2	2020	40033
3	2021	46354



Fonte: Elaboração própria

Analisando os temas que possuem a maior quantidade de proposições nos anos, foi possível observar que o tema "Saúde" foi o tema que apresentou a maior quantidade de propostas no período analisado, contudo no anos de 2019 e 2021 não foi o tema principal, dando lugar a "Administração Pública" e "Direitos Humanos e Minorias" respectivamente.

Figura 040: Tabela temas por ano

```

1 proposicoesPorAno = dfProposicoes.groupby(['tema','anoDataset'], as_index=False)[['id']].count()
2 temasPorAno = proposicoesPorAno.pivot_table(index='tema',columns='anoDataset',values='id')
3 temasPorAno.loc[:, 'Total'] = temasPorAno.sum(numeric_only=True, axis=1)
4 temasPorAno.sort_values(by='Total', ascending=False).head(10)
5

```

	anoDataset	2018	2019	2020	2021	Total
	tema					
	Saúde	515.0	1299.0	3089.0	1730.0	6633.0
	Direitos Humanos e Minorias	422.0	1252.0	1728.0	1791.0	5193.0
	Administração Pública	400.0	1310.0	1987.0	1260.0	4957.0
	Finanças Públicas e Orçamento	377.0	992.0	1712.0	1448.0	4529.0
	Trabalho e Emprego	228.0	628.0	1331.0	832.0	3019.0
	Educação	365.0	975.0	850.0	821.0	3011.0
	Defesa e Segurança	174.0	647.0	771.0	618.0	2210.0
	Viação, Transporte e Mobilidade	252.0	793.0	542.0	514.0	2101.0
	Direito Penal e Processual Penal	239.0	720.0	621.0	453.0	2033.0
	Comunicações	217.0	413.0	355.0	994.0	1979.0

Fonte: Elaboração própria

Levando em consideração o evento global da pandemia do novo Coronavírus em 2019, era de se esperar que houvesse um aumento da atividade parlamentar relacionada ao tema "Saúde", contudo é possível observar que foi no ano de 2020 que isto ocorreu.

Conforme detalhado anteriormente, nem todas as proposições possuem classificação temática e nem todas possuem conteúdo que possa ser analisado, logo não podem ser classificadas. Estas proposições precisaram ser desconsideradas pois não agregavam ao objetivo do projeto seja para treinamento do modelo de classificação, seja para classificação em si. Isto posto, o processo de limpeza consiste em utilizar apenas proposições que possuam um tema atribuído, ou seja não nulo, apresentando estas uma quantidade de 53.268 registros.

Figura 041: Informações data frame proposições classificadas

```

1 dfProposicoesClassificadas = dfProposicoes.loc[dfProposicoes['tema'].isna() == False]
2 dfProposicoesClassificadas.info()
3 dfProposicoesClassificadas.describe()
4
5

<class 'pandas.core.frame.DataFrame'>
Int64Index: 53268 entries, 0 to 149154
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          53268 non-null   int64  
 1   descricaoTipo 53268 non-null   object  
 2   anoDataset    53268 non-null   int64  
 3   statusExtracao 53268 non-null   int64  
 4   tipoDocumento 53268 non-null   object  
 5   tamanhoEmBytes 53268 non-null   int64  
 6   tema         53268 non-null   object  
 7   tokens        53268 non-null   object  
 8   semConteudo   53268 non-null   bool   
 9   qtdTokens    53268 non-null   int64  
dtypes: bool(1), int64(5), object(4)
memory usage: 4.1+ MB

```

Fonte: Elaboração própria

Com as proposições classificadas devidamente filtradas, foi necessário remover aquelas que não possuíam conteúdo, ou seja, que apresentavam no campo “semConteudo” o valor “True”. Contudo não foram identificadas proposições classificadas que não possuem conteúdo, seja de ementa, seja de inteiro teor.

Figura 042: Resumo do campo “semConteudo” proposições classificadas

```

1 print(dfProposicoesClassificadas.groupby(['semConteudo'], as_index=False)[['id']].count())
2

0   semConteudo   id
    False       53268

```

Fonte: Elaboração própria

Considerando o dataset das proposições classificadas, foi avaliado por meio de uma nuvem de palavras (*word cloud*) os termos que mais se destacam no conteúdo das proposições. Embora fora feita uma higienização do conteúdo das proposições para eliminar palavras muito comuns, artigos e etc, foi possível identificar na nuvem de palavras gerada a ocorrência de termos muito comuns no contexto da atividade parlamentar, tais como “congresso”, “nacional”, “vigor”, “publicação”, “sala” entre outras. Neste caso decidiu-se por retirar estas palavras com o objetivo de destacar termos mais relevantes como “saúde”, “segurança pública”, “energia elétrica”, “saúde pública” etc .

Figura 043: Nuvem de palavras com os termos comuns

```
1 f, ax = plt.subplots(figsize=(18, 9))
2 plt.axis('off')
3 plt.imshow(nuvemPalavras, interpolation='bilinear')
4 plt.show()
```



Fonte: Elaboração própria

Figura 044: Nuvem de palavras sem os termos comuns



Fonte: Elaboração própria

Neste ponto do trabalho, estavam devidamente separados os datasets que seriam utilizados no modelo de machine learning, para treinamento e testes (proposições que possuem conteúdo e classificação) e para classificação (proposições que possuem conteúdo, mas não apresenta tema atribuído).

Figura 045: Informações proposições classificadas

```

1 dfProposicoesClassificadas = dfProposicoes.loc[dfProposicoes['tema'].isna() == False]
2 dfProposicoesClassificadas.info()
3 dfProposicoesClassificadas.describe()
4
5

<class 'pandas.core.frame.DataFrame'>
Int64Index: 53268 entries, 0 to 149154
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          53268 non-null   int64  
 1   descricaoTipo 53268 non-null   object  
 2   anoDataset   53268 non-null   int64  
 3   statusExtracao 53268 non-null   int64  
 4   tipoDocumento 53268 non-null   object  
 5   tamanhoEmBytes 53268 non-null   int64  
 6   tema         53268 non-null   object  
 7   tokens       53268 non-null   object  
 8   semConteudo  53268 non-null   bool   
 9   qtdTokens    53268 non-null   int64  
dtypes: bool(1), int64(5), object(4)
memory usage: 4.1+ MB

```

Fonte: Elaboração própria

Figura 046: Informações proposições não classificadas

```

1 dfProposicoesNaoClassificadas = dfProposicoes.loc[
2     (dfProposicoes['tema'].isna()) & (dfProposicoes['tokens'].isna() == False)
3 ]
4 dfProposicoesNaoClassificadas.info()
5 dfProposicoesNaoClassificadas.describe()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 94516 entries, 41 to 149174
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          94516 non-null   int64  
 1   descricaoTipo 94516 non-null   object  
 2   anoDataset   94516 non-null   int64  
 3   statusExtracao 94516 non-null   int64  
 4   tipoDocumento 94516 non-null   object  
 5   tamanhoEmBytes 94516 non-null   int64  
 6   tema         0 non-null     object  
 7   tokens       94516 non-null   object  
 8   semConteudo  94516 non-null   bool   
 9   qtdTokens    94516 non-null   int64  
dtypes: bool(1), int64(5), object(4)
memory usage: 7.3+ MB

```

Fonte: Elaboração própria

5. Criação de Modelos de Machine Learning

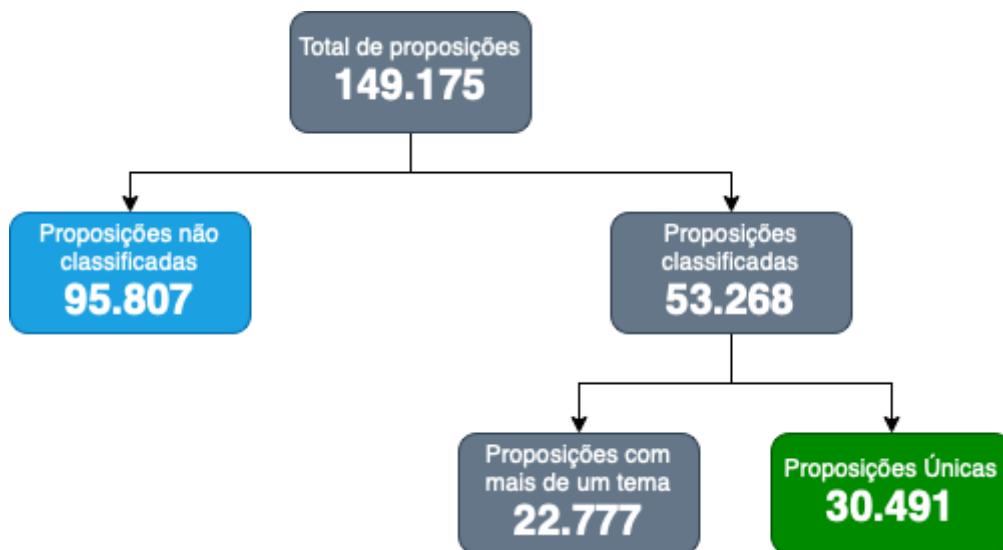
Agora que os dados foram obtidos, processados e analisados, seguiu-se para o alcance do objetivo principal do projeto, que foi criar uma máquina de classificação das proposições utilizando modelos de *machine learning*. Dentre os tipos de aprendizagem de máquina, o que mais se adapta ao nosso problema é o de aprendizado supervisionado que pode ser aplicado a um conjunto de dados rotulados para induzir um modelo preditivo capaz de predizer, para um novo um novo objeto representado pelos seus valores preditivos, o

valor do seu atributo alvo. No caso deste projeto de dados, com base no conteúdo de uma proposição, tentou-se definir qual o tema (ou temas) que ela pode ser classificada.

Para construção do modelo de machine learning foi utilizada principalmente a biblioteca para aprendizagem de máquina *scikit-learn* para a linguagem de programação *Python*, em especial seus módulos para métricas e algoritmos de classificação. O *scikit-learn* fornece um conjunto de ferramentas que facilitam muito a realização das etapas necessárias para um problema de classificação de dados.

Até este ponto ficou claro que estamos lidando com um problema de classificação que basicamente consiste em agrupar os dados com base em suas características predeterminadas. Neste processo, os dados podem ser classificados em duas ou mais classes de forma que quando há duas classes possíveis para a classificação trata-se de um problema de classificação binária e se o problema apresenta mais de duas classes é um problema de classificação multiclasse. O problema de classificação deste trabalho se encaixou em um subgrupo que é classificação *multi-label*, no qual os dados podem ser classificados em mais de uma classe conforme demonstra o diagrama abaixo.

Figura 047: Diagrama de proposições classificadas em um e mais temas



Fonte: Elaboração própria

O dataset de proposições unificadas apresenta todas as proposições que possuem conteúdo (ementa e/ou inteiro teor) com a informação e se tem alguma classificação temática atribuída a ela.

Figura 048: Informações data frame proposições unificado

```

1 #matplotlib inline
2 sns.set_theme(style="darkgrid")
3
4 DIR_DATASET = r'/Users/dev-rocks/Documents/TCC - Data Science e Big Data/Projeto/datasets/'
5 # DIR_DATASET = r'C:\Users\Daniel.Vale\Personal\PBDC\TCC\Projeto_v2\datasets\\'
6
7 colunas = ['id', 'descricaoTipo', 'tipoDocumento', 'tema', 'tokens']
8
9 dfProposicoes = pd.read_csv(f'{DIR_DATASET}proposicoesUnificado.csv', sep=';', low_memory=False, usecols=colunas)
10 dfProposicoes.info()

```

#	Column	Non-Null Count	Dtype
0	id	149175 non-null	int64
1	descricaoTipo	149175 non-null	object
2	tipoDocumento	149175 non-null	object
3	tema	53268 non-null	object
4	tokens	147784 non-null	object

dtypes: int64(1), object(4)
memory usage: 5.7+ MB

Fonte: Elaboração própria

Para aplicar o aprendizado supervisionado ao problema proposto, que exige uma base dados rotulada, o dataset unificado foi filtrado de forma a considerar apenas as proposições classificadas.

Figura 049: Informações proposições classificadas

```

1 dfProposicoesClassificadas = dfProposicoes.loc[dfProposicoes['tema'].isna() == False]
2 dfProposicoesClassificadas.info()

```

#	Column	Non-Null Count	Dtype
0	id	53268 non-null	int64
1	descricaoTipo	53268 non-null	object
2	tipoDocumento	53268 non-null	object
3	tema	53268 non-null	object
4	tokens	53268 non-null	object

dtypes: int64(1), object(4)
memory usage: 2.4+ MB

Fonte: Elaboração própria

As proposições podem ser classificadas em mais de um tema, quando isto ocorre o "id" da proposição aparece mais de uma vez no dataset.

Figura 050 - Descrição do data frame de proposições classificadas

```

1 dfProposicoesClassificadas['id'].astype('str').describe()

```


Fonte: Elaboração própria

Para poder aplicar os algoritmos de classificação, foi necessário transformar os dados do dataset de proposições classificadas de forma que cada proposição se tornasse um registro e cada tema uma coluna do dataset, sendo que caso a proposição esteja classificada no tema, na coluna correspondente ao tema o valor 1 seja atribuído e 0 caso contrário. Esta tarefa foi realizada utilizando a função "crosstab" da biblioteca *pandas*.

Figura 051 - Código para transformação do *data frame* de classificação

```

1 dfClassificacao = pd.crosstab(index=dfProposicoesClassificadas['id'],columns=dfProposicoesClassificadas['tema'])
2 dfClassificacao.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 30491 entries, 308880 to 2314680
Data columns (total 32 columns):
 #   Column          Non-Null Count Dtype  
 --- 
 0   Administração Pública      30491 non-null int64  
 1   Agricultura, Pecuária, Pesca e Extrativismo 30491 non-null int64  
 2   Arte, Cultura e Religião    30491 non-null int64  
 3   Cidades e Desenvolvimento Urbano 30491 non-null int64  
 4   Ciência, Tecnologia e Inovação 30491 non-null int64  
 5   Ciências Exatas e da Terra 30491 non-null int64  
 6   Ciências Sociais e Humanas 30491 non-null int64  
 7   Comunicações               30491 non-null int64  
 8   Defesa e Segurança        30491 non-null int64  
 9   Direito Civil e Processual Civil 30491 non-null int64  
 10  Direito Constitucional     30491 non-null int64  
 11  Direito Penal e Processual Penal 30491 non-null int64  
 12  Direito e Defesa do Consumidor 30491 non-null int64  
 13  Direito e Justiça         30491 non-null int64  
 14  Direitos Humanos e Minorias 30491 non-null int64  
 15  Economia                  30491 non-null int64  
 16  Educação                  30491 non-null int64  
 17  Energia, Recursos Hídricos e Minerais 30491 non-null int64  
 18  Esporte e Lazer           30491 non-null int64  
 19  Estrutura Fundiária       30491 non-null int64  
 20  Finanças Públicas e Orçamento 30491 non-null int64  
 21  Homenagens e Data Comemorativas 30491 non-null int64  
 22  Indústria, Comércio e Serviços 30491 non-null int64  
 23  Meio Ambiente e Desenvolvimento Sustentável 30491 non-null int64  
 24  Política. Partidos e Eleições   30491 non-null int64

```

Fonte: Elaboração própria

Figura 052 - Data frame de classificação após transformação

tema	Administração Pública	Agricultura, Pecuária, Pesca e Extrativismo	Arte, Cultura e Religião	Cidades e Desenvolvimento Urbano	Ciência, Tecnologia e Inovação	Ciências Exatas e da Terra	Ciências Sociais e Humanas	Comunicações	Defesa e Segurança	Direito Civil e Processual Civil	Indústria, Comércio e Serviços	Meio Ambiente e Desenvolvimento Sustentável
id												
308880	0	0	0	0	0	0	0	0	0	0	0	0
317970	1	0	0	0	0	0	0	0	0	0	0	0
427339	0	0	0	0	0	0	0	0	0	0	0	1
459323	0	0	0	0	0	0	0	0	0	0	0	0
498347	0	0	0	0	0	0	0	0	0	0	0	0
...
2314507	0	1	0	0	0	0	0	0	0	0	0	0
2314512	0	0	0	0	0	0	0	0	0	0	0	0
2314519	0	0	0	0	0	0	0	0	0	0	0	0
2314678	0	1	0	0	0	0	0	0	0	0	0	0
2314680	0	0	0	0	0	0	0	0	0	0	0	0

30491 rows x 32 columns

Fonte: Elaboração própria

Com a classificação temática no formato de vetor, foi construído um novo *dataset* para ser utilizado nos modelos de classificação. Este novo *dataset* possui todas as proposições classificadas, com as informações de "id", "descriçãoTipo", "tokens" (palavras do conteúdo) e o novo formato de classificação temática, indicando em qual tema elas foram classificadas.

Figura 053 - Código geração *dataframe* de treino

```

1 dfTreino = dfProposicoesClassificadas[['id', 'descricaoTipo', 'tokens']].drop_duplicates()
2 dfTreino = pd.merge(dfTreino, dfClassificacao, left_on='id', right_index=True)
3
4 print(f'[INFO ] Quantidade de instâncias: {len(dfTreino)}\n[INFO ] Número de atributos: {len(dfTreino.columns)})')
5 print('*'*80)
6 dfTreino.info()

```

#	Column	Non-Null Count	Dtype
0	id	30491	non-null int64
1	descricaoTipo	30491	non-null object
2	tokens	30491	non-null object
3	Administração Pública	30491	non-null int64
4	Agricultura, Pecuária, Pesca e Extrativismo	30491	non-null int64
5	Arte, Cultura e Religião	30491	non-null int64
6	Cidades e Desenvolvimento Urbano	30491	non-null int64
7	Ciência, Tecnologia e Inovação	30491	non-null int64
8	Ciências Exatas e da Terra	30491	non-null int64
9	Ciências Sociais e Humanas	30491	non-null int64
10	Comunicações	30491	non-null int64
11	Defesa e Segurança	30491	non-null int64
12	Direito Civil e Processual Civil	30491	non-null int64
13	Direito Constitucional	30491	non-null int64
14	Direito Penal e Processual Penal	30491	non-null int64
15	Direito e Defesa do Consumidor	30491	non-null int64

Fonte: Elaboração própria

Em problemas de classificação de texto em machine learning, é importante analisar as palavras do vocabulário dos textos. Após analisar todos os textos das proposições foi possível determinar que a quantidade de palavras chegou a 6.296.814, sendo 135.042 delas únicas. Foram identificadas proposições que possuem mais de 22.000 palavras e a média de palavras de uma proposição ficou em 207 palavras.

Figura 054 - Código para geração do vocabulário

```

1 tokens = []
2 for t in dfTreino['tokens'].values:
3     tokens.extend(t.split())
4
5 palavrasVocabulario = (list(set(tokens)))
6 print(f'[INFO ] Quantidade de palavras: {len(tokens)})')
7 print(f'[INFO ] Tamanho do vocabulário: {len(palavrasVocabulario)})')
8

```

[INFO] Quantidade de palavras: 6296814
[INFO] Tamanho do vocabulário: 135042

Fonte: Elaboração própria

Figura 055 - 10 maiores quantidade de palavras de uma proposição

```
1 dfTreino['qtdPalavras'] = dfTreino['tokens'].apply(lambda t:len(t.split()))
2 maiorQtdPalavras = dfTreino['qtdPalavras'].max()
3 dfTreino[['tokens','qtdPalavras']].nlargest(10,'qtdPalavras')
```

		tokens	qtdPalavras
55681	redação setembro alterou julho código eleitora...	22455	
23	estabelece medidas corrupção crimes patrimônio...	13557	
42248	dispõe regimes resolução instituições autoriza...	13521	
62041	dispõe transferência imóveis fundo regime prev...	13425	
48970	altera fevereiro julho atualizar legislação re...	12557	
2304	modifica previdência social estabelece regras ...	12472	
46931	redefine traçado parque joaquim altera parque ...	11271	
37894	institui plano regional desenvolvimento nordes...	10711	
50810	dispõe futebol profissional providências comis...	9147	
10446	dispõe futebol profissional providências felip...	9078	

Fonte: Elaboração própria

Figura 056 - Estatísticas de quantidade de palavras

```
1 dfTreino['qtdPalavras'].describe()
```

count	30491.000000
mean	206.513857
std	422.207921
min	1.000000
25%	17.000000
50%	130.000000
75%	276.000000
max	22455.000000
Name:	qtdPalavras, dtype: float64

Fonte: Elaboração própria

Antes de seguir para as etapas de treinamento e avaliação dos modelos de classificação, foi necessário representar as palavras do texto bruto em um formato que os algoritmos de machine learning compreendam. Existem várias formas de se fazer a representação, neste trabalho foi utilizada uma técnica simples mas eficiente chamada TF-IDF, que significa Frequência do Termo-Frequência Inversa. Basicamente trata-se de uma medida estatística que atribui um valor numérico à importância de uma palavra dentro de um conjunto de texto agindo como fator ponderador.

A biblioteca do scikit-learn disponibiliza o método "CountVectorizer", que foi utilizado em combinação com o método "TfidfTransformer" para transformar as palavras das proposições em uma matriz esparsa, um formato que os modelos de classificação pudessem utilizar.

Figura 057: Código de vetorização TF-IDF

```

1 from sklearn.feature_extraction.text import CountVectorizer
2 from sklearn.feature_extraction.text import TfidfTransformer
3
4 vetorizador = CountVectorizer()
5 tfidf = TfidfTransformer()
6

```

Fonte: Elaboração própria

Os modelos de classificação, para fazer o processo de treinamento e teste, trabalham com dois conjuntos de dados: um conjunto contendo o texto vetorizado (comumente chamado de X) e o conjunto dos rótulos atribuídos ao texto (conhecido como y). Assim sendo, foi atribuído à variável X os tokens da proposição vetorizados com *TF-IDF* e à variável y foi atribuída a lista (*array*) de classificação temática contendo 32 colunas. Interessante observar que após a transformação das palavras em valores pelo TF-IDF, o resultado é um array de mais de 130.000 colunas, pois o algoritmo considera o total de palavras do conjunto de texto analisado e atribui um valor de importância para cada uma delas.

Figura 057: Código de vetorização TF-IDF

```

1 rotulos = list(dfTreino.iloc[:,3:-2].columns.values)
2
3 X_counts = vetorizador.fit_transform(dfTreino['tokens'])
4 X = tfidf.fit_transform(X_counts)
5 y = dfTreino[rotulos].values
6
7 print(f'[INFO] Dimensões do X: {X.shape}')
8 print(f'[INFO] Dimensões do y: {y.shape}')

```

```
[INFO] Dimensões do X: (30491, 133814)
[INFO] Dimensões do y: (30491, 31)
```

Fonte: Elaboração própria

A próxima etapa necessária para um modelo de classificação foi a divisão dos dados em dados para treinamento e dados para teste do modelo. Levando em consideração o volume de dados (*dataset*) disponíveis para o modelo, os mesmos foram divididos na proporção de 80% para treino (*trainset*) e 20% para teste (*testset*). Esta divisão foi feita utilizando o método "iterative_train_test_split" da biblioteca *scikit-multilearn*. Esta função se diferencia da função equivalente do *scikit-learn*, "train_test_split", por fazer a estratificação iterativa dos dados para entregar uma distribuição balanceada entre as classes.

Figura 058: Código para divisão dos dados para treino e teste

```

1 from skmultilearn.model_selection import iterative_train_test_split
2 np.random.seed(45)
3 X_treino,y_treino,X_teste,y_teste = iterative_train_test_split(X,y,test_size=0.2)
4
5 print(f'[INFO ] Dimensões do X_treino: {X_treino.shape}')
6 print(f'[INFO ] Dimensões do y_treino: {y_treino.shape}')
7
8 print(f'[INFO ] Dimensões do X_teste: {X_teste.shape}')
9 print(f'[INFO ] Dimensões do y_teste: {y_teste.shape}')

[INFO ] Dimensões do X_treino: (24392, 133814)
[INFO ] Dimensões do y_treino: (24392, 31)
[INFO ] Dimensões do X_teste: (6099, 133814)
[INFO ] Dimensões do y_teste: (6099, 31)

```

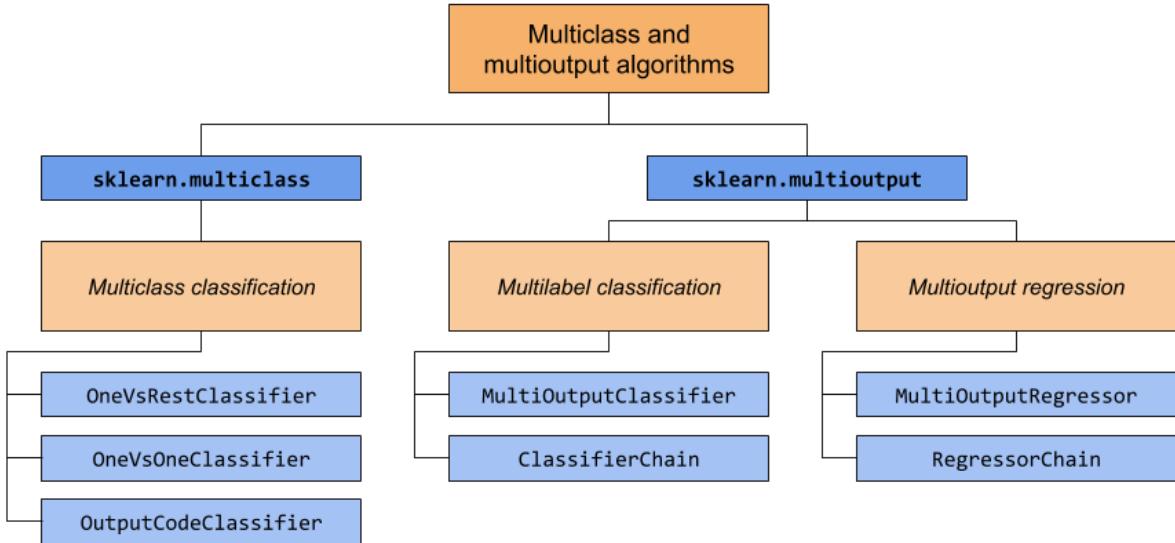
Fonte: Elaboração própria

Para o problema de classificação de dados existem diversos algoritmos que podem ser utilizados para executar a tarefa. Isto posto, para decidir qual o modelo mais adequado aos nossos dados e ao objetivo do projeto, foi necessário criar uma forma de fazer uma análise comparativa de alguns algoritmos. Foram escolhidos os algoritmos abaixo para realização do benchmark:

1. SGD Classifier (com variações de hiperparâmetros)
2. Linear SVC (com variações de hiperparâmetros)
3. Linear SVC com Feature Selection
4. Ridge Classifier
5. Perceptron
6. Passive Aggressive Classifier
7. Nearest Centroid
8. Naive Bayes - Multinomial NB
9. Naive Bayes - Bernoulli NB
10. Naive Bayes - Complement NB
11. Random Forest Classifier
12. MLP Classifier
13. Logistic Regression
14. K Neighbors Classifier

Como dito anteriormente, os dados utilizados configuraram um problema de classificação *multi-label*, ou seja, uma proposição pode ser classificada em mais de um tema. A biblioteca *scikit-learn* dispõe de dois módulos para se trabalhar com este tipo de problema sendo o mais adequado para o projeto a classe MultiOutputClassifier.

Figura 059 - Diagramas algoritmos *Multiclass* e *Multioutput* scikit-learn



Fonte: https://scikit-learn.org/stable/_images/multi_org_chart.png

Como métricas de para avaliação dos modelos foi utilizada a acurácia, que indica o percentual de acerto do modelo, considerando as instâncias classificadas como verdadeiro positivo (*True Positive*), as classificadas como verdadeiro negativo (*True Negative*) e o total de instâncias testadas. Como medidas secundárias foram considerados os tempos de treinamento e teste do modelo que podem inviabilizar a utilização do modelo dadas as limitações computacionais.

Figura 060 - Fórmula de cálculo da acurácia

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Fonte:

<https://www.mydatamodels.com/wp-content/uploads/2020/10/2.-Accuracy-formula-machine-learning-algorith.ms.png>

Com as medidas de desempenho dos modelos definidas, foi criada uma lista com a lista dos modelos que serão avaliados. Esta lista é composta pela instância do classificador,

contendo seus hiperparâmetros, e um nome amigável para facilitar a identificação do modelo.

Figura 060 - Fórmula de cálculo da acurácia

```

16 listaModelos = [
17     (SGDClassifier(alpha=0.0001, max_iter=50, penalty='l2'), 'SGD penalidade l2'),
18     (SGDClassifier(alpha=0.0001, max_iter=50, penalty='l1'), 'SGD penalidade l1'),
19     (SGDClassifier(alpha=0.0001, max_iter=50, penalty="elasticnet"),'SGD elasticnet'),
20     (LinearSVC(penalty='l2', dual=False, tol=1e-3), 'Linear SVC penalidade l2'),
21     (LinearSVC(penalty='l1', dual=False, tol=1e-3), 'Linear SVC penalidade l1'),
22     (RidgeClassifier(tol=1e-2, solver="sag"), "Ridge Classifier"),
23     (Perceptron(max_iter=50), "Perceptron"),
24     (PassiveAggressiveClassifier(max_iter=50), "Passive-Aggressive"),
25     (NearestCentroid(), 'NearestCentroid '),
26     (MultinomialNB(alpha=0.01), ' Naive Bayes - MultinomialNB'),
27     (BernoulliNB(alpha=0.01), ' Naive Bayes - BernoulliNB'),
28     (ComplementNB(alpha=0.01), ' Naive Bayes - ComplementNB'),
29     (Pipeline(
30         [
31             (
32                 "feature_selection",
33                 SelectFromModel(LinearSVC(penalty="l2", dual=False, tol=1e-3)),
34             ),
35             ("classificador", LinearSVC(penalty="l2"))
36         ]
37     ), 'LinearSVC c/ feature selection'),
38     (RandomForestClassifier(), "Random forest"),
39     (MLPClassifier(max_iter=100, random_state=1, verbose=False), 'MLP Classifier'),
40     (LogisticRegression(solver='sag'), 'Logistic Regression - SAG'),
41     (LogisticRegression(solver='lbfgs',random_state=0), 'Logistic Regression - LBFGS'),
42     (KNeighborsClassifier(n_neighbors=10), "kNN")
43   )
44 ]

```

Fonte: Elaboração própria

A avaliação dos algoritmos de classificação consistiu basicamente em 1) executar a função “fit” de cada classificador utilizando os *datasets* de treino (*X_treino* e *y_treino*) para que o modelo seja treinado, 2) executar a função “predict” de cada classificador utilizando o *dataset* de teste (*X_teste*) para fazer a classificação dos dados de teste utilizando o modelo treinado, 3) fazer a medição da acurácia do modelo utilizando o método “accuracy_score” do módulo de métricas do *scikit-learn* utilizando a o *dataset* de teste (*y_teste*) e a predição realizada pelo classificador.

Figura 060 - Código que faz o treino, predição dos classificadores (abordagem multi-label)

```

5  for modelo in listaModelos:
6      print('=' * 80)
7      print(f'[PROCESSANDO MODELO]: {modelo[1]}')
8
9      estatisticas = {
10         'modelo':modelo[1],
11         'acuracia':0,
12         'tempoTreino':0,
13         'tempoTeste':0
14     }
15
16     classificador = modelo[0]
17     multioutput = MultiOutputClassifier(classificador)
18     inicio = time()
19
20     multioutput.fit(X_treino,y_treino)
21     tempoTreino = time()-inicio
22
23
24     inicio = time()
25     predicao = multioutput.predict(X_teste)
26     tempoTeste = time()-inicio
27     acuracia = metrics.accuracy_score(y_teste, predicao)
28
29     estatisticas['acuracia'] = acuracia
30     estatisticas['tempoTeste'] = tempoTeste
31     estatisticas['tempoTreino'] = tempoTreino
32
33
34     resultadoMultiLabel.append(estatisticas)
35
36
37     print(f'\n[INFO ] Acurácia ===== {estatisticas["acuracia"]*100:.2f}%')
38     print(f'[INFO ] Tempo treino = {estatisticas["tempoTreino"]:.2f} segundos')
39     print(f'[INFO ] Tempo teste == {estatisticas["tempoTeste"]:.2f} segundos')
40     print('_'* 80)
--
```

Fonte: Elaboração própria

Figura 061 - Outputs do treinamento e teste da abordagem multi-label

```

=====
[PROCESSANDO MODELO]: SGD panalidade 12
[INFO  ] Acurácia ===== 42.66%
[INFO  ] Tempo treino = 4.56 segundos
[INFO  ] Tempo teste == 0.15 segundos
=====

[PROCESSANDO MODELO]: SGD panalidade 11
[INFO  ] Acurácia ===== 39.32%
[INFO  ] Tempo treino = 7.18 segundos
[INFO  ] Tempo teste == 0.14 segundos
=====

[PROCESSANDO MODELO]: SGD elasticnet
[INFO  ] Acurácia ===== 40.76%
[INFO  ] Tempo treino = 10.45 segundos
[INFO  ] Tempo teste == 0.14 segundos
=====

[PROCESSANDO MODELO]: Linear SVC penalidade 12
[INFO  ] Acurácia ===== 46.66%
[INFO  ] Tempo treino = 23.71 segundos
[INFO  ] Tempo teste == 0.15 segundos
=====

[PROCESSANDO MODELO]: Linear SVC penalidade 11
[INFO  ] Acurácia ===== 45.68%
[INFO  ] Tempo treino = 40.88 segundos
[INFO  ] Tempo teste == 0.15 segundos
=====
```

Fonte: Elaboração própria

Após a execução dos modelos de classificação, foi criado um *dataset* composto pelas métricas de cada classificador testado com o objetivo de facilitar a análise comparativa dos mesmos, tanto na questão da acurácia quanto dos tempos para treino e teste dos modelos.

Figura 062 - Benchmark dos modelos abordagem multi-label

	modelo	acuracia	tempoTreino	tempoTeste
12	LinearSVC c/ feature selection	0.467618	38.933285	0.603588
3	Linear SVC penalidade l2	0.466634	23.709083	0.145989
4	Linear SVC penalidade l1	0.456796	40.875750	0.149341
5	Ridge Classifier	0.438433	38.711639	0.141190
7	Passive-Aggressive	0.429251	9.359465	0.141411
0	SGD penalidade l2	0.426627	4.559749	0.145773
6	Perceptron	0.410231	3.215560	0.138166
2	SGD elasticnet	0.407608	10.449692	0.140792
1	SGD penalidade l1	0.393179	7.182517	0.138109
13	Logistic Regression - SAG	0.379242	51.519686	0.171776
14	Logistic Regression - LBFGS	0.379242	81.386913	0.159516
9	Naive Bayes - MultinomialNB	0.373012	1.355876	0.401861
11	Naive Bayes - ComplementNB	0.266929	1.971424	0.420516
8	NearestCentroid	0.243646	2.468234	0.475513
10	Naive Bayes - BernoulliNB	0.241843	1.765003	0.835257

Fonte: Elaboração própria

O modelo “ Linear SVC c/ feature selection” apresentou o maior desempenho do ponto de vista de assertividade, apresentando uma acurácia de 44,36% e os tempos de treinamento e teste do modelo não são discrepantes da maioria dos outros modelos.

Os modelos “Random Forest” e “MLP Classifier” em função da capacidade computacional do computador utilizado, não puderam ter o treinamento e teste concluídos, desta forma foram desconsiderados por serem inviáveis . Os modelos “Logistic Regression - SAG” e “Logistic Regression - LBFGS” apresentaram a acurácia de 32,12%, porém o segundo modelo apresentou um tempo de treino muito elevado em relação ao primeiro, logo o segundo classificador foi desconsiderado também. O modelo “KNN” se destaca por apresentar um alto tempo de treinamento (404 segundos) o que eleva a média geral nesta métrica, além disto a acurácia apurada, de 32,66%, não o colocou como um dos melhores modelos para o problema de classificação deste projeto.

De forma geral, não foi observada uma discrepância significativa da acurácia dos classificadores analisados, visto que o valor médio desta medida ficou em 38,05% e o desvio

padrão apresentou um valor de 0,007 indicando uma concentração dos valores em torno da média. Foi observado também os tempos médios de 21 segundos para treino e 0.28 segundos para treinamento.

Figura 063 - Estatísticas resultado abordagem multi-label

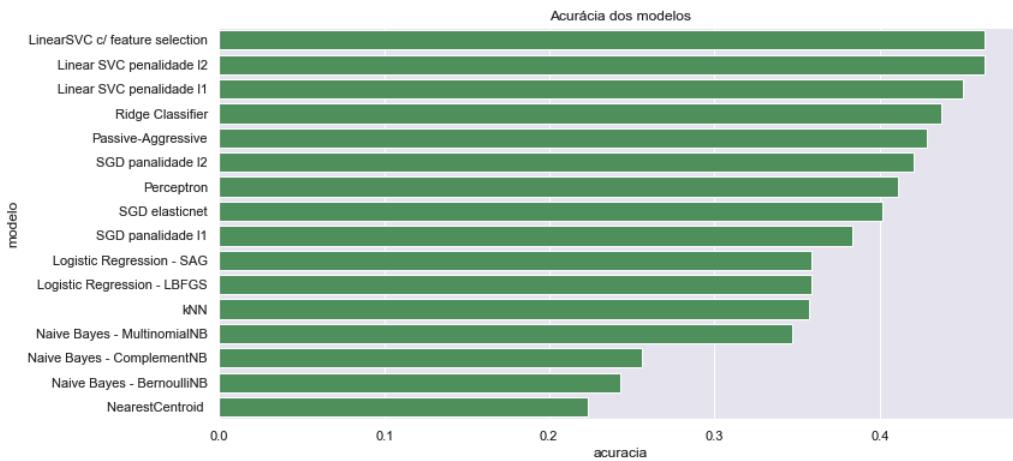
```
1 dfBenchmark_1.describe()
```

	acuracia	tempoTreino	tempoTeste
count	15.000000	15.000000	15.000000
mean	0.385353	21.164258	0.280587
std	0.076077	24.055007	0.217652
min	0.241843	1.355876	0.138109
25%	0.376127	2.841897	0.141301
50%	0.407608	9.359465	0.149341
75%	0.433842	38.822462	0.411188
max	0.467618	81.386913	0.835257

Fonte: Elaboração própria

Figura 064 - Gráfico de acurácia abordagem multi-label

```
1 f, ax = plt.subplots(figsize=(12, 6))
2 ax = sns.barplot(x="acuracia", y="modelo", data=dfBenchmark_1.sort_values(by='acuracia', ascending=False),
3 color="g")
4
5 ax.set(title='Acurácia dos modelos')
6
7 f.show()
```



Fonte: Elaboração própria

A primeira abordagem para resolver o problema de classificação multi-label, fazendo uso da classe “MultioutputClassifier” juntamente com os modelos avaliados apresentou uma acurácia média de 38,53% e uma acurácia máxima de 46,76%. Diante destes valores, foi necessário buscar uma nova abordagem para resolução do problema de classificação *multi-label* de forma a obter uma acurácia melhor na classificação dos dados.

O modelo consiste em transformar o problema de classificação *multi-label* em múltiplos problemas de classificação binária. Nesta abordagem, assume-se que as classes em que os dados são classificados não possuem correlação entre si, por exemplo um texto classificado como “Administração Pública” não impede que o mesmo seja classificado como “Defesa e Segurança” e a questão respondida é se a proposição é do tema “Administração Pública” ou não. Uma preocupação desta abordagem é o overfitting (super ajustamento do modelo aos dados de teste) em casos em que se tem dados não rotulados, contudo este cenário já foi tratado quando foram mantidas apenas proposições que estão classificadas em algum tema no *datatrain* e *datatest*.

Do ponto de vista prático foi feita uma alteração na rotina que faz o treinamento e teste dos classificadores de forma que cada modelo seja treinado uma vez para cada classe, usando como dados para treinamento apenas os valores do rótulo. Foi necessário alterar também a apuração da acurácia, passando esta a ser uma média da acurácia apurada em cada rótulo.

Figura 065 - Código da abordagem multiple single-label

```
-- 22     for i in range(len(rotulos)):
-- 23         inicio = time()
-- 24         classificador.fit(X_treino,y_treino[:,i])
-- 25         tempoTreinoTotal += time() - inicio
-- 26
-- 27         inicio=time()
-- 28         predicao = classificador.predict(X_teste)
-- 29         tempoTesteTotal += time() - inicio
-- 30
-- 31         acuraciaAcumulada += metrics.accuracy_score(y_teste[:,i], predicao)
-- 32         estatisticas['acuraciaDetalhada'].append((rotulos[i], metrics.accuracy_score(y_teste[:,i], predicao)))
-- 33
-- 34
-- 35         estatisticas['acuracia'] = acuraciaAcumulada / len(rotulos)
-- 36         estatisticas['tempoTeste'] = tempoTesteTotal
-- 37         estatisticas['tempoTreino'] = tempoTreinoTotal
-- 38
-- 39         resultadoSingleLabel.append(estatisticas)
-- 40
-- 41         print(f'\n[INFO ] Acurácia ===== {estatisticas["acuracia"]*100:.2f}')
-- 42         print(f'[INFO ] Tempo treino = {estatisticas["tempoTreino"]:.2f} segundos')
-- 43         print(f'[INFO ] Tempo teste == {estatisticas["tempoTeste"]:.2f} segundos')
-- 44         print('_'* 80)
-- 45
```

Fonte: Elaboração própria

Figura 066 - Outputs do treinamento e teste da abordagem multiple single-label

```
=====
[PROCESSANDO MODELO]: SGD penalidade 12

[INFO ] Acurácia ===== 97.18
[INFO ] Tempo treino = 4.22 segundos
[INFO ] Tempo teste == 0.14 segundos

=====
[PROCESSANDO MODELO]: SGD penalidade 11

[INFO ] Acurácia ===== 96.99
[INFO ] Tempo treino = 7.32 segundos
[INFO ] Tempo teste == 0.15 segundos

=====
[PROCESSANDO MODELO]: SGD elasticnet

[INFO ] Acurácia ===== 97.07
[INFO ] Tempo treino = 10.10 segundos
[INFO ] Tempo teste == 0.15 segundos

=====
[PROCESSANDO MODELO]: Linear SVC penalidade 12
```

Fonte: Elaboração própria

Após a execução dos modelos utilizando a segunda abordagem, o modelo mais eficiente foi o “Linear SVC - c/ feature selection” com uma acurácia de 97,38% seguido pelo classificador “Linear SVC - penalidade l2” porém com um tempo de teste e treino inferior ao primeiro colocado.

Figura 067 - Benchmark dos modelos da abordagem multiple single-label

	modelo	acuracia	tempoTreino	tempoTeste	acuraciaDetalhada
12	LinearSVC c/ feature selection	0.973867	32.137632	0.659487	[(Administração Pública, 0.8927693064436792), ...]
3	Linear SVC penalidade l2	0.973824	22.515423	0.153848	[(Administração Pública, 0.8917855386128873), ...]
4	Linear SVC penalidade l1	0.973438	38.720280	0.221502	[(Administração Pública, 0.8906378094769635), ...]
5	Ridge Classifier	0.972608	36.111493	0.149195	[(Administração Pública, 0.8922774225282833), ...]
0	SGD penalidade l2	0.971793	4.224610	0.142107	[(Administração Pública, 0.8883423512051156), ...]
2	SGD elasticnet	0.970656	10.098004	0.148421	[(Administração Pública, 0.8848991637973438), ...]
7	Passive-Aggressive	0.970233	8.278213	0.149265	[(Administração Pública, 0.8717822593867847), ...]
1	SGD penalidade l1	0.969858	7.323093	0.148147	[(Administração Pública, 0.8794884407279882), ...]
13	Logistic Regression - SAG	0.968958	50.560746	0.146873	[(Administração Pública, 0.8862108542383997), ...]
14	Logistic Regression - LBFGS	0.968953	79.975526	0.187741	[(Administração Pública, 0.8862108542383997), ...]
6	Perceptron	0.968668	3.684615	0.144672	[(Administração Pública, 0.8678471880636169), ...]
9	Naive Bayes - MultinomialNB	0.965367	1.042728	0.358642	[(Administração Pública, 0.8767010985407444), ...]
8	NearestCentroid	0.949140	2.512640	0.589609	[(Administração Pública, 0.8025905886210855), ...]
11	Naive Bayes - ComplementNB	0.942725	1.051218	0.345027	[(Administração Pública, 0.8414494179373668), ...]
10	Naive Bayes - BernoulliNB	0.927090	1.742436	0.905052	[(Administração Pública, 0.8563698967043778), ...]

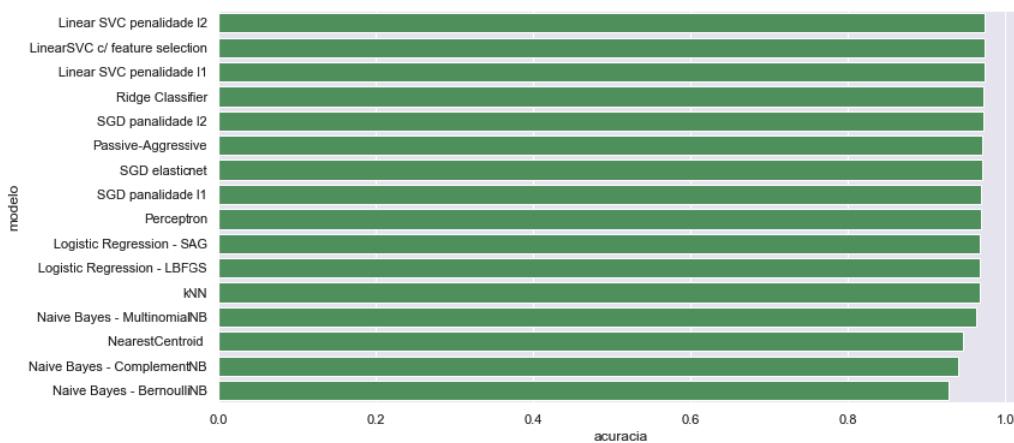
Fonte: Elaboração própria

Figura 068 - Benchmark dos modelos da abordagem multiple single-label

```

1 f, ax = plt.subplots(figsize=(12, 6))
2 ax = sns.barplot(x="acuracia", y="modelo", data=dfBenchmark_2.sort_values(by='acuracia', ascending=False), color="g")
3
4 f.show()
5

```



Fonte: Elaboração própria

O benchmark realizado também evidenciou que a acurácia média foi de 96,44%, com uma concentração em torno da média corroborada pelo desvio padrão de 0,0113. Neste comparativo também o modelo “KNN” demonstrou elevado tempo de teste chegando a 390 segundos, sendo que a média geral foi de 29 segundos.

Figura 068 - Benchmark dos modelos da abordagem multiple single-label

```

1 dfBenchmark_2.describe()

```

	acuracia	tempoTreino	tempoTeste
count	15.000000	15.000000	15.000000
mean	0.964479	19.998577	0.296639
std	0.013733	23.243265	0.237339
min	0.927090	1.042728	0.142107
25%	0.967017	3.098628	0.148284
50%	0.969858	8.278213	0.153848
75%	0.972201	34.124563	0.351835
max	0.973867	79.975526	0.905052

Fonte: Elaboração própria

Considerando a análise comparativa dos modelos de machine learning avaliados e as abordagens adotadas, ficou evidente que o classificador mais adequado para classificação temática das proposições é o classificador “Linear SVC - penalidade L2” com uma abordagem de classificação *multiple single-label*, ou seja, classificação binária para cada tema.

Uma vez definido o melhor modelo de machine learning para resolver o problema de classificação proposto, foi necessário realizar testes de uso deste para classificar proposições

novas, não utilizadas no processo de treinamento e teste do classificador. Neste sentido foi implementada uma função que recebe o conteúdo da proposição e que basicamente, 1) faz o processamento do texto removendo stopwords, 2) faz tokenização do texto usando TF-IDF, 3) faz a predição dos temas.

Com o propósito de testar a predição dos temas, foi utilizado o texto abaixo para teste:

*"Na data em que apresentamos este Projeto de Lei para apreciação da Câmara dos Deputados, o Brasil concentra um total de 21,8 milhões de casos confirmados de **Covid-19** e 607 mil mortes. À este cenário de catástrofe e **emergência de saúde** pública, gerenciado de maneira irresponsável e potencialmente criminosa pelo governo brasileiro - conforme reforça o recém publicado relatório da "CPI da Covid" do Senado Federal - soma-se uma profunda deterioração da economia e da **vida** do povo brasileiro, que enfrenta a carestia, o **desemprego**, a inflação, a insegurança alimentar e outras inúmeras dificuldades.*

*Partindo da percepção de que a atuação do governo federal frente à **pandemia** no Brasil foi desastrosa, vez que marcada pelo negacionismo, pela busca de uma **imunidade** coletiva por **contágio** e pela promoção de **medicamentos** comprovadamente ineficazes para tratamento da **Covid**, entendemos ser de responsabilidade do Estado a garantia de uma **pensão** de caráter **indenizatório/ reparatório** aos **sobreviventes** que, em razão do **vírus** e/ou dos tratamentos **médicos** ineficazes promovidos pelo governo federal, apresentem **sequelas** temporárias ou permanentes que reduzem sua capacidade **laborativa** e/ou sua **qualidade de vida**."*

O modelo de machine learning apresentou como resultado os temas “Direito Constitucional”, “Direito e Defesa do Consumidor”, “Previdência e Assistência Social” e “Saúde”. O resultado não estabelece um peso de importância para os temas e a validação precisou ser feita por meio de análise do texto para saber se a classificação foi satisfatória. Analisando o conteúdo da proposição, é possível identificar algumas palavras chaves que tem relação com a classificação obtida (palavras destacadas em negrito).

Figura 069 - Resultado de teste da abordagem multiple single-label

```
[INFO ] Tempo de predicao:0.35 segundos
=====
Direito Constitucional
Direito e Defesa do Consumidor
Previdência e Assistência Social
Saúde
```

Fonte: Elaboração própria

6. Interpretação dos Resultados

Conforme demonstrado na etapa anterior com o exemplo e com outros testes realizados, o modelo de machine learning escolhido para fazer a classificação temática das proposições demonstrou uma performance satisfatória na tarefa de classificação temática dos textos de exemplo, então a próxima etapa consistiu em utilizar o modelo para fazer a classificação temática das proposições que não há possuíam.

No período considerado para o projeto, foram identificadas 94.516 proposições que não possuem nenhum tema atribuído. Para cada uma destas proposições foi feita a previsão do tema utilizando o modelo de classificação construído e o resultado desta execução foi um dataset contendo 388.396 registros de classificação, indicando o “id” da proposição e o tema atribuído à ela. Este resultado indica que cada proposição foi classificada, em média, em quatro temas. Foi possível identificar também que 3.787 proposições não puderam ser classificadas pelo classificador.

Figura 070 - Amostra data frame classificação final

```
1 dfClassificacao = pd.read_csv(f'{DIR_DATASET}classificacaoFinal.csv', sep=';', low_memory=False)
2 dfClassificacao
```

	Id	tema	predicao
0	2190174	Processo Legislativo e Atuação Parlamentar	True
1	2190174	Trabalho e Emprego	True
2	2190289	Direito Constitucional	True
3	2190289	Homenagens e Datas Comemorativas	True
4	2190289	Processo Legislativo e Atuação Parlamentar	True
...
441660	2313739	Finanças Públicas e Orçamento	False
441661	2313739	Viação, Transporte e Mobilidade	False
441662	2313740	Energia, Recursos Hídricos e Minerais	False
441663	2313740	Finanças Públicas e Orçamento	False
441664	2314519	Viação, Transporte e Mobilidade	False

441665 rows × 3 columns

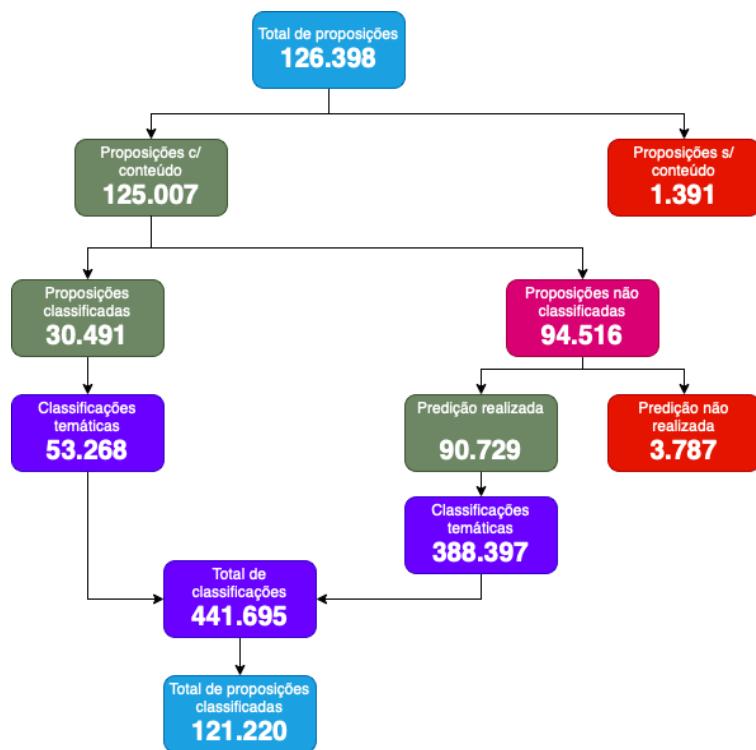
Fonte: Elaboração própria

Para realização das análises do resultado da classificação realizada, foi criado um dataset que reúne as classificações que já existiam e as classificações feitas pelo modelo de machine learning. Esta base resultou em 441.665 classificações temáticas para 121.220 proposições. O trabalho utilizou uma base inicial de proposições com 126.398 proposições, sendo que destas 1.391 não apresentavam nem ementa e nem inteiro teor, logo foram desconsideradas do projeto, restando 125.007 proposições com conteúdo que foram utilizadas seja para treinamento do modelo de machine learning (30.491 proposições) ou para fazer a predição da classificação temática (94.516 proposições), em outras palavras na base inicial o percentual de proposições classificadas era de 24%.

Considerando que as proposições podem apresentar mais de um tema atribuído, a base inicial apresentava para as proposições classificadas 53.268 classificações (1,75 temas por proposição). Após realizar a predição das classificações temáticas para as proposições não classificadas utilizando o modelo de classificação, obteve-se 388.397 classificações temáticas para 90.729 proposições (não foi possível classificar 3.787). Ao final, a base de classificações temáticas apresentou 3,64 temas por proposição.

A quantidade de temas por proposição duplicou de 2 para 4 após utilização do modelo de classificação e o percentual de proposições classificadas saltou de 24% para 96%, permitindo assim trabalhar fazer uma avaliação melhor da atividade parlamentar no que diz respeito aos temas que estão sendo debatidos na Câmara Legislativa Federal. O diagrama abaixo detalha a evolução da classificação temática de proposições antes e depois da aplicação do modelo.

Figura 071 - Diagrama de evolução da classificação



Fonte: Elaboração própria

7. Apresentação dos Resultados

Antes de detalharmos mais os resultados do trabalho, é importante ter uma visão geral das etapas do projeto de machine learning que foi desenvolvido. Para isto foi utilizado o modelo Canvas baseado no consagrado *Business Model Canvas*, desenvolvido por Jasmine Vasandani para auxiliar na construção de projetos de Ciência de Dados e *Machine Learning*.

Figura 072 - Canvas do projeto

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

Title: Classificação temática da atividade parlamentar com machine learning					
1 Problem Statement What problem are you trying to solve? What larger issues does the problem address? Contribuir para melhor o acompanhamento da atividade parlamentar da Câmara dos Deputados brasileira, por meio da avaliação dos documentos produzidos pelos deputados, podendo até contribuir para uma melhora na eficiência do processo de classificações temáticas realizado pelo Centro de Documentação e Informação da Câmara.	2 Outcomes/Predictions What prediction(s) are you trying to make? Identify applicable predictor (X) and/or target (y) variables. Determinar as classificações temáticas das propostas legislativas com base nos texto da ementa e no texto do documento de inteiro teor das propostas.	3 Data Acquisition Where are you sourcing your data from? Is there enough data? Can you work with it? A fonte dos dados é o portal de dados abertos da câmara legislativa, base de proposições, base de classificação temática, dados estruturados em documentos CSV. Documentos contendo todo o conteúdo de uma proposição. Dados desestruturados e não consolidados.			
4 Modeling What models are appropriate to use given your outcomes? Modelos de classificação de dados que possam predizer uma classe de um dado com base em um conjunto de exemplos. Algoritmos de machine learning de aprendizagem supervisionada.	5 Model Evaluation How can you evaluate your model's performance? Através da acurácia, uma medida percentual que aponta quanto o modelo acerta em suas previsões. O tempo de teste e treino do modelo para garantir a viabilidade de execução com os recursos computacionais disponíveis.	6 Data Preparation What do you need to do to your data in order to run your model and achieve your outcomes? Identificar quais atributos são relevantes. Unificar as bases de proposições e temas das proposições. Separar as proposições classificadas para treino e teste de modelos. Tokenizar os textos eliminando stopwords Vetorizar os textos a serem analisados.			

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

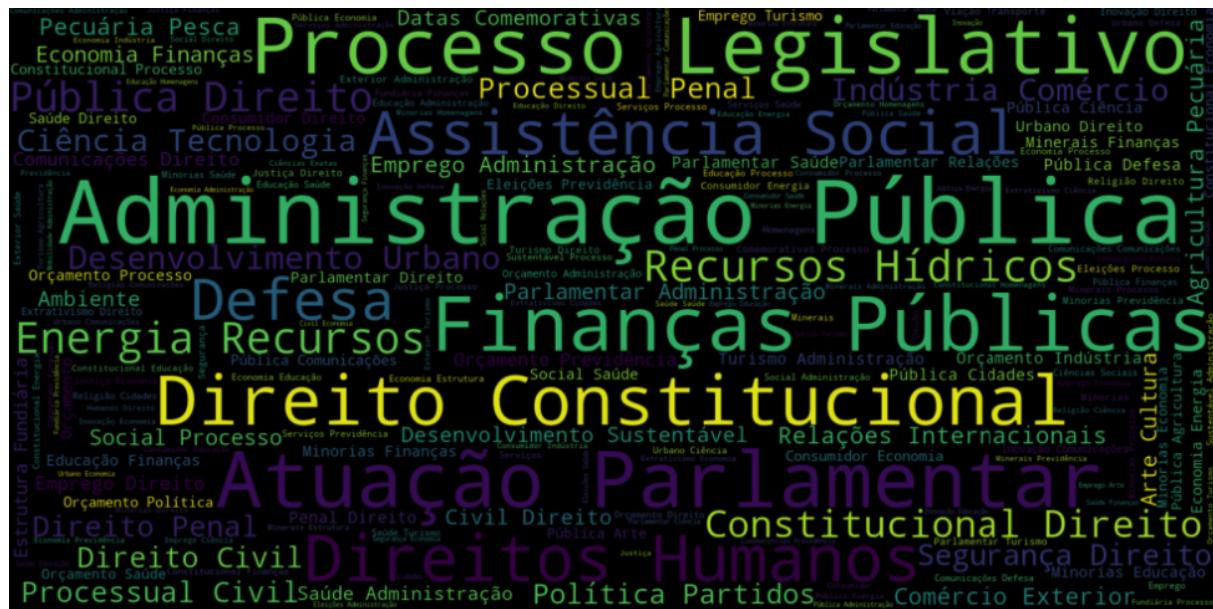
* Note: This canvas is intended to be used as a starting point for your data science projects. Data science workflows are typically nonlinear.

Fonte:

<https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0>

Com a maioria das proposições classificadas foi possível elaborar algumas análises. Foi utilizada uma nuvem de palavras com base no nome dos temas para se ter uma percepção visual dos temas que tem mais se destacado no período estudado em relação aos outros temas.

Figura 073 - Nuvem de palavras dos temas

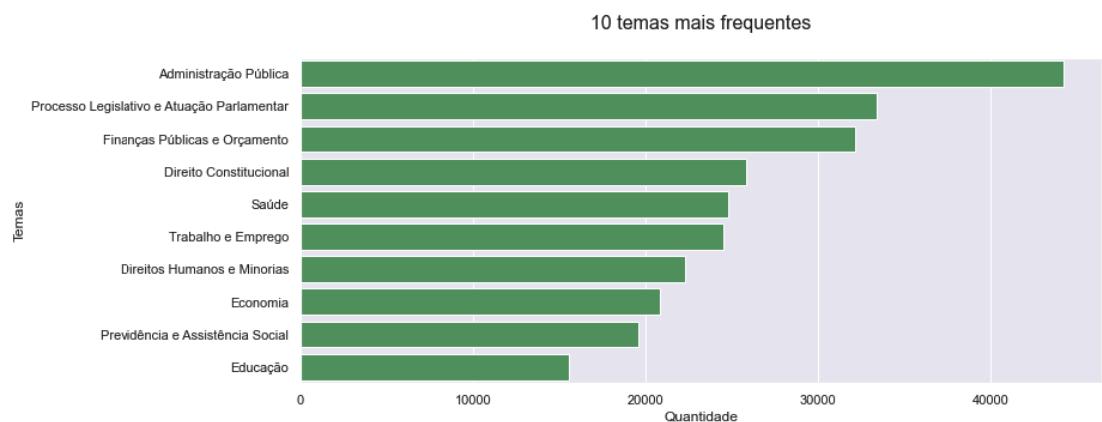


Fonte: Elaboração própria

Também foi possível determinar quais os temas mais e menos preponderantes na atividade parlamentar dentro do período de estudo, bem como uma segregação ano a ano.

Figura 074 - Gráfico temas mais frequentes

```
1 fig, ax = plt.subplots(figsize=(12,5))
2 sns.barplot(x='qtd',y='tema',data=dfResumoPorTema.head(10),color='g')
3
4 ax.set_title('10 temas mais frequentes\n',fontsize=16)
5 ax.set_xlabel('Quantidade')
6 ax.set_ylabel('Temas')
7
8 fig.show()
9
```



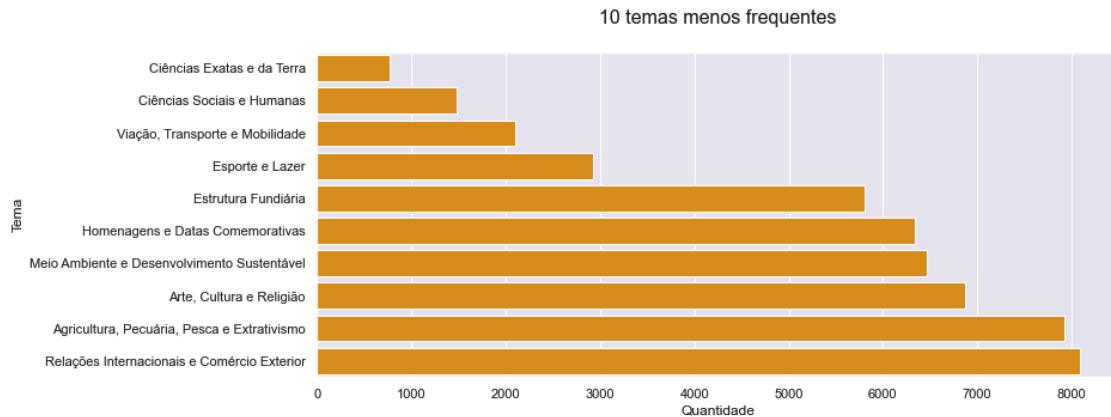
Fonte: Elaboração própria

Figura 075 -Gráfico temas menos frecuentes

```

1 fig, ax = plt.subplots(figsize=(12,5))
2 sns.barplot(x='qtd',y='tema',data=dfResumoPorTema.tail(10),
3             color='orange',
4             order=dfResumoPorTema.tail(10).sort_values(by='qtd')[['tema']])
5
6 ax.set_title('10 temas menos frequentes\n', fontsize=16)
7 ax.set_xlabel('Quantidade')
8 ax.set_ylabel('Tema')
9
10 fig.show()

```



Fonte: Elaboração própria

Por meio da análise de correlação, foi possível determinar quais temas possuem correlação entre si e qual a força desta correlação. Pode-se observar que alguns temas como "*Processo Legislativo e Atuação Parlamentar*" e "*Direito Constitucional*", "*Previdência e Assistência Social*" e "*Direitos Humanos e Minorias*" apresentam uma forte correlação entre si, por outro lado temas como "Esporte e Lazer" e "Direito Civil e Processual Civil" não tem praticamente nenhuma correlação.

Figura 076 - Código para gerar gráfico de correlação

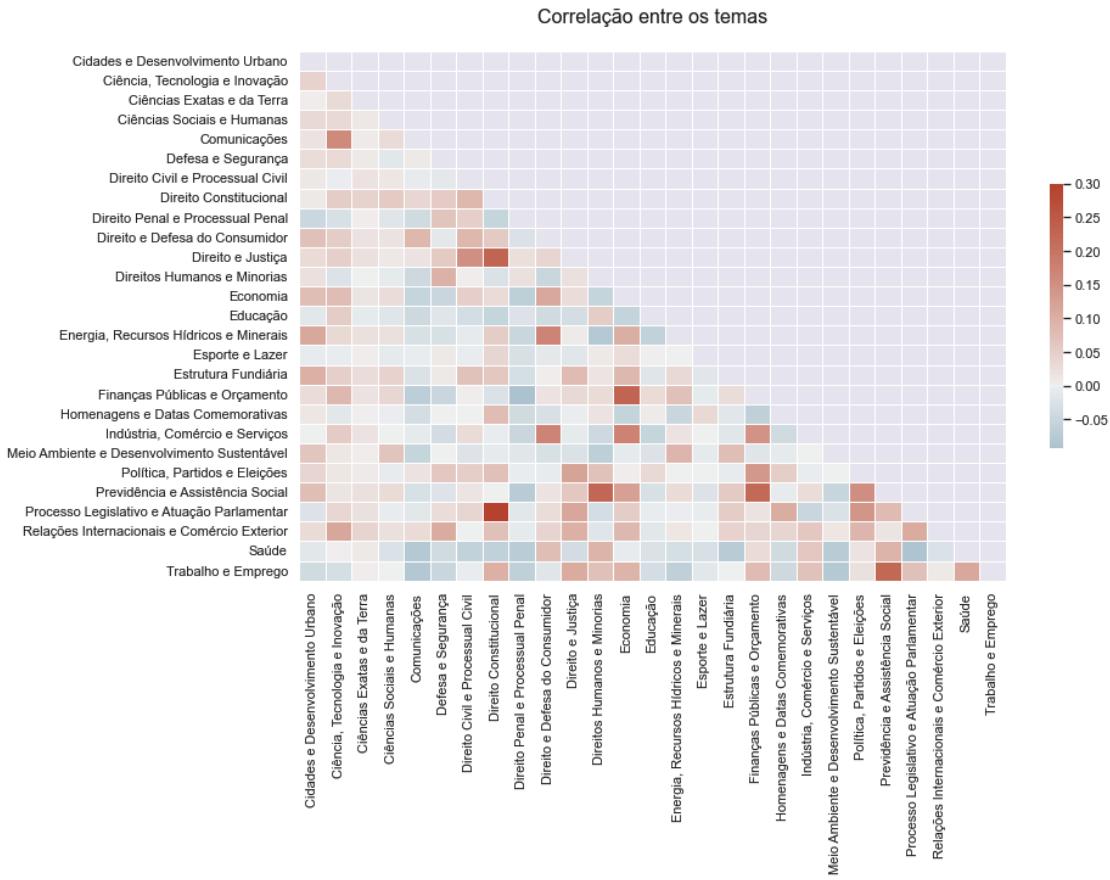
```

1
2 fig, ax = plt.subplots(figsize=(13,8))
3 mask = np.triu(np.ones_like(correlacao, dtype=bool))
4 cmap = sns.diverging_palette(230, 20, as_cmap=True)
5 sns.heatmap(correlacao,mask=mask,cmap=cmap,annot=False,
6             fmt='.0f',vmax=.3, center=0,
7             square=False, linewidths=.5, cbar_kws={"shrink": .5})
8
9 ax.set_title('Correlação entre os temas\n', fontsize=16)
10
11 plt.show()

```

Fonte: Elaboração própria

Figura 077 - Código para gerar gráfico de correlação



Fonte: Elaboração própria

Outra análise que pôde ser realizada com base na avaliação dos temas por ano da proposição, revelou como a quantidade de um tema alterou de um ano para outro, como por exemplo o tema "*Administração Pública*" que praticamente dobrou do ano de 2018 para o ano de 2019 e se manteve desta forma nos anos seguintes. Por outro lado, temas como "*Ciências Sociais e Humanas*" se mantiveram com pouca variação de um ano para o outro.

Figura 078 - Código para geração do mapa de calor temas x ano

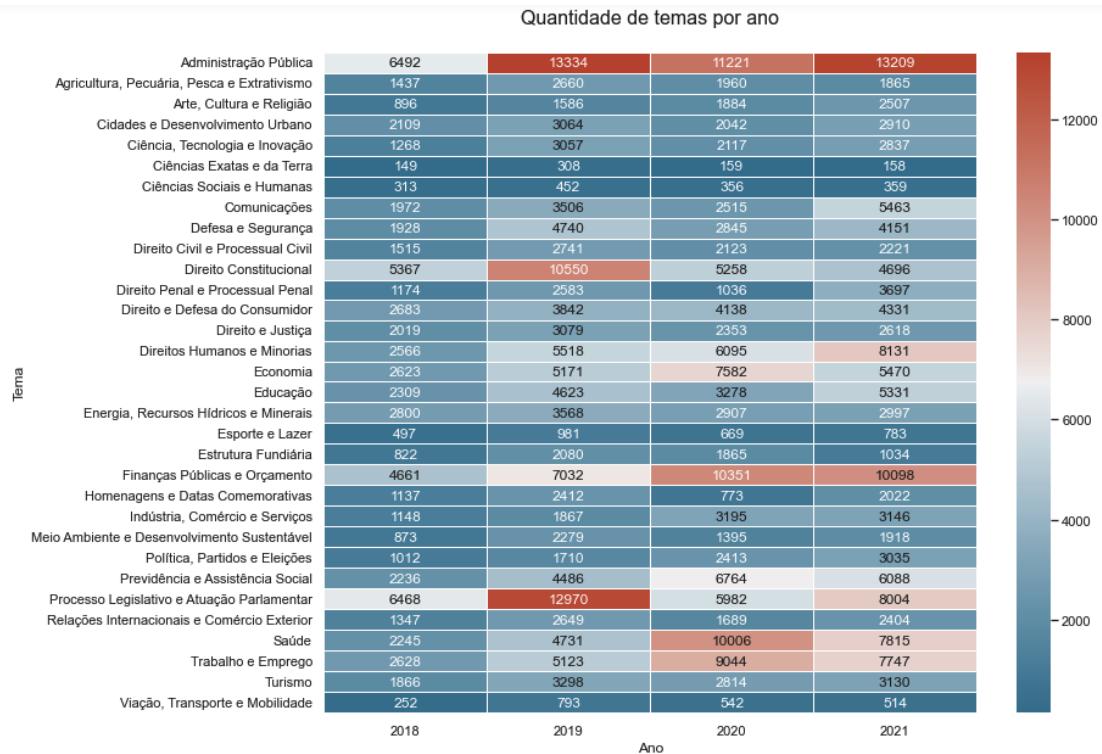
```

1 fig, ax = plt.subplots(figsize=(12,10))
2 sns.heatmap(pd.crosstab(index=[df['tema']],
3                           columns=df['anoDataset']),
4                           annot=True,
5                           fmt='d',cmap=cmap,
6                           linewidths=.5)
7
8 ax.set_title('Quantidade de temas por ano\n', fontsize=16)
9 ax.set_xlabel('Ano')
10 ax.set_ylabel('Tema')
11 plt.show()

```

Fonte: Elaboração própria

Figura 079 - Mapa de calor tema x ano



Fonte: Elaboração própria

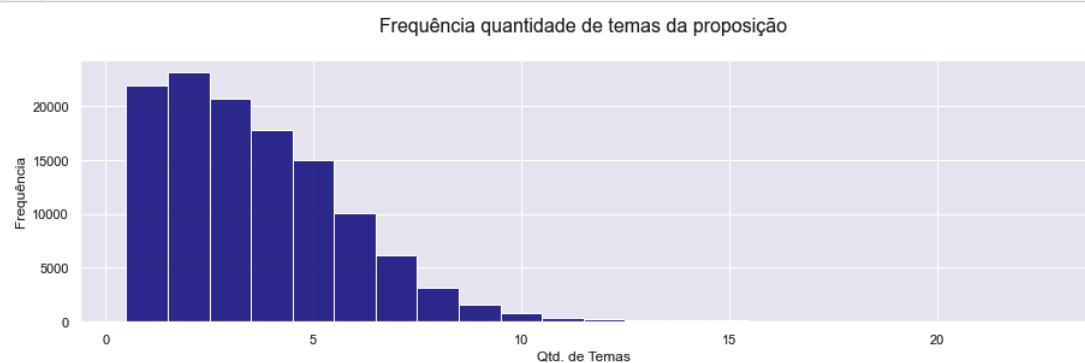
Observando a quantidade de temas que uma proposição foi atribuída, foi possível determinar que o mais comum é que uma proposição seja classificada em 2 temas, contudo existem casos extremos em que a classificação foi feita para 22 temas, sendo estes casos identificados como falha da decodificação do arquivo de inteiro teor. Considerando que 99% das proposições foram classificadas em no máximo 10 temas, decidiu-se por não tomar nenhuma ação em relação aos demais casos.

Figura 080 - Histograma de frequência quantidade de temas

```

1 fig, ax = plt.subplots(figsize=(15,4))
2 sns.histplot(data=crosstab.sum(axis=1), discrete=True,color='navy')
3
4 ax.set_title('Frequência quantidade de temas da proposição\n', fontsize=16)
5 ax.set_xlabel('Qty. de Temas')
6 ax.set_ylabel('Frequência')
7
8 plt.show()

```



Fonte: Elaboração própria

Figura 081 - Código para elaboração da tabela de frequência quantidade temas

```

1 frecuenciaTemas = crosstab.sum(axis=1)
2 dfTabelaFrequencia = pd.DataFrame(pd.Series(frecuenciaTemas.values).value_counts())
3 dfTabelaFrequencia.set_axis(['frequencia'], axis='columns')
4 dfTabelaFrequencia['%'] = dfTabelaFrequencia['frequencia']/len(frecuenciaTemas)*100
5 dfTabelaFrequencia[['freqAcumulada','%Acumulado']] = dfTabelaFrequencia.cumsum()
6

```

Fonte: Elaboração própria

Figura 082 - Tabela de frequência quantidade de temas

```
1 dfTabelaFrequencia.head(11)
```

	frequencia	%	freqAcumulada	%Acumulado
2	23087	19.045537	23087	19.045537
1	21842	18.018479	44929	37.064016
3	20639	17.026068	65568	54.090084
4	17742	14.636199	83310	68.726283
5	14951	12.333773	98261	81.060056
6	10107	8.337733	108368	89.397789
7	6140	5.065171	114508	94.462960
8	3178	2.621680	117686	97.084639
9	1574	1.298466	119260	98.383105
10	793	0.654182	120053	99.037288
11	395	0.325854	120448	99.363141

Fonte: Elaboração própria

Figura 083 - Tabela de frequência quantidade de temas (continuação)

```
1 dfTabelaFrequencia.tail(10)
```

	frequencia	%	freqAcumulada	%Acumulado
12	270	0.222736	120718	99.585877
13	160	0.131991	120878	99.717868
14	122	0.100643	121000	99.818512
15	113	0.093219	121113	99.911731
16	59	0.048672	121172	99.960403
17	29	0.023923	121201	99.984326
18	9	0.007425	121210	99.991751
20	7	0.005775	121217	99.997525
19	2	0.001650	121219	99.999175
22	1	0.000825	121220	100.000000

Fonte: Elaboração própria

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://youtu.be/rL5QDPOhcbw>

Link para o repositório:

<https://github.com/DanielCirino/tcc-datasience-bigdata>

Link para os datasets:

<https://drive.google.com/drive/folders/1KERqatQ-1xxpI9gKeGiazumO0hOuC522?usp=sharing>

Link para apresentação:

<https://docs.google.com/presentation/d/1iYeVyrZx7A7GNxRiolWcLIEbWk4xtxOJK-dx1OQjCo/edit?usp=sharing>

REFERÊNCIAS

RAMALHO, Luciano. **Python Fluente. Programação clara, concisa e eficaz.** São Paulo: Novatec Editora Ltda, 2021.

GÉRON, Aurélien. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow.** Rio de Janeiro: Alta Books, 2021.

MCKINNEY, Wes. **Python para Análise de Dados.** São Paulo: Novatec Editora Ltda, 2019.

KNAFLIC, Cole Nussbaumer. **Storytelling com dados: Um guia sobre visualização de dados para profissionais de negócios.** Rio de Janeiro: Alta Books, 2019.

FACELI, Katti et.al. **Inteligência Artificial: Uma abordagem de Aprendizado de Máquina.** 2.ed. Cidade: Rio de Janeiro, 2021.

Portal da Câmara dos Deputados. Disponível em: <https://www.camara.leg.br/> Acesso em: 15/10/2021

Dados abertos da Câmara dos deputados. Disponível em:
<https://dadosabertos.camara.leg.br/swagger/api.html#staticfile> Acesso em 15/10/2021

A separação dos três poderes: Executivo, Legislativo e Judiciário. Disponível em:
<https://www.politize.com.br/separacao-dos-tres-poderes-executivo-legislativo-e-judiciario/>

Scikit-Learn. Disponível em: <https://scikit-learn.org/> Acessado em: 01/12/2021

Multi-Label Classification in Python. Disponível em: <http://scikit.ml/> Acessado em: 01/03/2022

Matplotlib. Disponível em: <https://matplotlib.org/> Acessado em 03/02/2022

A Data Science Workflow Canvas to Kickstart your projects. Disponível em:
<https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0> Acessado em: 20/04/2022

Aprenda a estratificar dados multi-label com scikit-multilearn. Disponível em:
<https://insightlab.ufc.br/aprenda-a-estratificar-dados-multi-label-com-scikit-multilearn/>
Acessado em: 1504/2022

Uma introdução à classificação multi-label. Disponível em:
<https://acervolima.com/uma-introducao-a-classificacao-multilabel/> Acessado em 01/04/2022

Deep dive into multi-label classification. Disponível em:
<https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff> Acessado em: 01/04/2022

Flowchart Maker. Disponível em: <https://app.diagrams.net> Acessado em 01/04/2022