UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO DO NASCIMENTO OLIVEIRA

CONSTRUÇÃO E CLASSIFICAÇÃO DE UMA BASE TEXTUAL EM PORTUGUÊS

RIO DE JANEIRO
2023

THIAGO DO NASCIMENTO OLIVEIRA

CONSTRUÇÃO E CLASSIFICAÇÃO DE UMA BASE TEXTUAL EM PORTUGUÊS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Giseli Rabello Lopes
Coorientador: Prof. João Carlos P. da Silva

RIO DE JANEIRO
2023

CIP - Catalogação na Publicação

O48c

Oliveira, Thiago do Nascimento
Construção e classificação de uma base textual em português. / Thiago do Nascimento Oliveira. -- Rio de Janeiro, 2023.
81 f.

Orientadora: Giseli Rabello Lopes.
Coorientador: João Carlos P. da Silva.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2023.

1. Mineração de Textos. 2. Web Scraping. 3. Processamento de Linguagem Natural. 4. Classificação de Textos. 5. Aprendizado de Máquina. I. Lopes, Giseli Rabello, orient. II. Silva, João Carlos P. da, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

THIAGO DO NASCIMENTO OLIVEIRA

CONSTRUÇÃO E CLASSIFICAÇÃO DE UMA BASE TEXTUAL EM PORTUGUÊS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 26 de abril de 2023

BANCA EXAMINADORA:

Profa. Giseli Rabello Lopes
D.Sc. (UFRGS)

Prof. João Carlos P. da Silva
D.Sc. (UFRJ)

Profa. Carolina Gil Marcelino
D.Sc. (CEFET-MG)

Prof. João Antônio Recio da Paixão
D.Sc. (PUC-RJ)

Este trabalho e todos os frutos que dele produzir são dedicados ao meu Senhor, meu Deus, que me acompanha e me dá forças em todos os momentos da minha jornada.

AGRADECIMENTOS

Agradeço aos meus pais (Rogério e Verônica), ao meu padrasto (Joaquim Marcelo), à minha amada esposa (Nadinne), à minha querida sogra (Sandra), e às minhas irmãs (Nicole e Sthefany), pelo suporte, incentivo, paciência e apoio. Em todos os momentos eles oram, partilham a tensão, a preocupação, comemoram comigo as vitórias e seguram minha mão nas derrotas. Sem vocês, eu não teria propósito.

Agradeço a UFRJ e a todos os seus funcionários, pelo esforço que é manter uma estrutura pública de ensino funcionando em nosso país. Em especial, gostaria de agradecer à professora Giseli Lopes, pelo carinho, cuidado, paciência e pela boa vontade que sempre teve ao me receber. Especialmente também ao professor João Carlos, que muito mais que me fazer apaixonar pela Inteligência Artificial, mostrou que eu alcançava mais alto do que imaginava.

Nunca esquecerei do sorriso aberto e do carinho materno da professora Valéria Bastos, que tornou a minha caminhada acadêmica mais doce. Imensa gratidão à senhora.

Agradeço também a todos os professores e amigos que tive contato na graduação.

Todos vocês foram fundamentais, de alguma forma, para eu chegar até aqui.

Muito obrigado!

"É fazendo que se aprende a fazer aquilo que se deve aprender a fazer."

Aristóteles

RESUMO

A Web tornou-se um importante meio para disponibilização de informações. Entre as principais dificuldades, nesse contexto dinâmico, estão a busca por informações específicas e a categorização das mesmas. Com a facilidade de acesso à Internet e a possibilidade de qualquer pessoa publicar ou replicar conteúdo online, é preciso ter cuidado ao selecionar as fontes dessas informações. No domínio do setor elétrico não é diferente. Um importante ator, nesse cenário, é o IFE - Informativo Eletrônico do Setor Elétrico - que sintetiza resumos de notícias, obtidas a partir de fontes confiáveis, para profissionais do setor. A aspiração deste trabalho é propor uma metodologia a fim de se criar um modelo de classificação automática de notícias, para oferecer aos seus editores a possibilidade de uma análise rápida, completa e precisa do conteúdo do texto e atribuir de forma mais ágil e eficiente as categorias dos resumos de notícias. Uma análise das implementações clássicas de aprendizado supervisionado de máquina empregando os algoritmos k-Vizinhos-Mais-Próximos, Regressão Logística, Naïve Bayes, Máquinas de Vetores de Suporte, Floresta Randômica, e um comitê com esses classificadores foi realizada. Alguns valores candidatos para hiperparâmetros foram comparados e a melhor combinação deles para cada uma das implementações foi configurada em seu treinamento. Este trabalho conclui com a avaliação dos desempenhos alcançados por cada algoritmo na tarefa de classificação de texto no contexto de resumos de notícias do IFE.

Palavras-chave: inteligência artificial; mineração de dados; mineração de textos; web scraping; processamento de linguagem natural; classificação de textos; aprendizado de máquina.

ABSTRACT

The Web has become an important means of making information available. Among the main difficulties in this dynamic context are the search for specific information and its categorization. With the ease of access to the Internet and the ability for anyone to publish or replicate content online, care must be taken when selecting sources for this information. In the field of the electrical sector it is no different. An important player in this scenario is the IFE - Electronic Newsletter for the Electricity sector - which synthesizes summaries of news, obtained from reliable sources, for professionals in the sector. The aspiration of this work is to propose a methodology in order to create a model of automatic classification of news, to offer its editors the possibility of a fast, complete and precise analysis of the text content and to assign, in a more agile and efficient way, the categories of news summaries. An analysis of classical supervised machine learning implementations employing k-Nearest-Neighbors, Logistic Regression, Naïve Bayes, Support Vector Machine, Random Forest, and an ensemble with these classifiers was carried out. Some candidate values for hyperparameters were compared and the best combination of them for each of the implementations was configured in their training. This work concludes with the evaluation of the performances achieved by each algorithm in the text classification task in the context of IFE news summaries.

Keywords: artificial intelligence; data mining; text mining; web scraping; natural language processing; text classification; machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

UFRJ	Universidade Federal do Rio de Janeiro
IFE	Informativo Eletrônico do Setor Elétrico
GESEL	Grupo de Estudos do Setor Elétrico
SVM	Support Vector Machine
k-NN	k-Nearest Neighbors
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
URL	Uniform Resource Locator
TF	Term Frequency
DF	Document Frequency
IDF	Inverse Document Frequency
IG	Information Gain
CPU	Central Processing Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Units
NLTK	Natural Language Toolkit
DOM	Document Object Model
BoW	Bag of Words

SUMÁRIO

1 INTRODUÇÃO

A tarefa de classificar textos remonta ao trabalho seminal de M. Maron nos anos 60 (MARON, 1961). Os classificadores de textos inicialmente eram baseados na construção manual de um conjunto de regras lógicas. Uma abordagem demorada, que demandava mão de obra especializada e que dependia do domínio, onde especialistas deviam prever as ocorrências das palavras e, baseando-se em regras, indicavam a categoria (SEBASTIANI, 2002).

Em um cenário de superabundância de documentos de textos, considerando ainda o tamanho de cada documento e a variedade de temas, torna-se dispendiosa a classificação de textos baseada em engenharia do conhecimento (consistindo na definição manual de um classificador por especialistas do domínio). Essa era a abordagem mais popular até o início dos anos 90, quando começou a perder espaço em favor do aprendizado de máquina, que foi definido por Sebastiani (2002, p. 2) como "um processo indutivo geral que cria, automaticamente, um classificador de texto automático que aprende, a partir de um conjunto de documentos pré-classificados, as características das categorias de interesse" (traduzido do original em inglês).

Com a evolução da capacidade de processamento das máquinas, a tecnologia de armazenamento e o rápido desenvolvimento da Web - fonte de grande quantidade de documentos em formato digital - a classificação automática de textos emergiu novamente no cenário de aplicações computacionais motivada pelo avanço das pesquisas em aprendizado de máquina. Também chamada de categorização de textos de forma intercambiável nos círculos acadêmicos, quando associada ao aprendizado de máquina é aplicada em diversos contextos, como indexação de documentos, geração automática de textos, desambiguação do sentido das palavras, e em geral, qualquer aplicação que exija organização documental.

A Web torna-se cada vez mais uma relevante fonte de informação, em detrimento dos tradicionais jornais e revistas impressas, de telejornais, etc[1]. Entre diversas vantagens, proporciona alcance a uma infinidade de notícias, a qualquer tempo, lugar e idioma. Com essa comodidade, uma notícia pode se espalhar rapidamente e ter grande influência na sociedade, mudando opiniões, causando comoção e até mesmo influenciando o resultado de eleições. Contudo, a democratização da Internet também ampliou o alcance de fenômenos prejudiciais, como por exemplo, *spams* e notícias falsas (*fake news*).

Comparando técnicas de aprendizado de máquina, Alameri e Mohd (2021) aplicaram diferentes abordagens, desde básicas como Naïve Bayes e Máquinas de Vetores de Suporte (SVMs), até abordagens de aprendizado profundo como Rede Neurais, em diferentes conjuntos de dados, para o apontamento automatizado de notícias falsas no contexto da pandemia de COVID-19. Para isso, torna-se particularmente importante analisar e

[1] https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2021, acessado em 05/10/2022

processar textos jornalísticos, e a classificação automática é uma tecnologia chave para organizar, distinguir e selecionar rapidamente as informações de acordo com a necessidade dos interessados ou com o conteúdo da informação.

A categorização de notícias também é uma tecnologia extremamente útil para editores, que podem utilizar esses modelos para organizar notícias em diferentes seções em seus *newsletters*, sites e aplicativos, facilitando o acesso dos usuários às informações de sua conveniência. Kumar et al. (2021) lançaram mão dos algoritmos SVM, Naïve Bayes (com modelo multinomial) e da Regressão Logística para organizar textos de notícias em 20 categorias. Além disso, também pode ser útil para outras organizações, por exemplo, governamentais, instituições financeiras e empresas, que precisam monitorar informações relevantes de diferentes áreas para tomar decisões mais assertivas. Patton e Potok (2008) enfatizam, de mercados financeiros à gestão de desastres e à epidemiologia, a importância de entender os impactos que eventos podem gerar. Para tanto, eles propõem uma abordagem para fundir informações sobre eventos de desastres naturais, obtidas de diversas fontes de notícias, analisando os ciclos dessas informações para identificar padrões temporais anteriores relacionados ao impacto desses eventos.

No contexto do setor de energia isso não é diferente. Ele é extremamente dinâmico e está em constante evolução, com novas tecnologias sendo desenvolvidas constantemente, além de mudanças regulatórias e políticas que afetam diretamente as empresas e órgãos que atuam nessa área. É nesse cenário que está inserido o Informativo Eletrônico do Setor Elétrico (IFE)[2]. Uma *newsletter* editada periodicamente que é importante fonte de informações para os profissionais que atuam nesse setor, apresentando notícias, pesquisas e informações relevantes para os segmentos de geração, transmissão, distribuição e comercialização de energia elétrica, além de abranger temas como regulamentação, meio ambiente, inovação e tecnologia.

O processo de categorização dos resumos de notícias do IFE pode demandar uma grande quantidade de trabalho manual. Para otimizar esse processo, pode-se utilizar algoritmos de aprendizado de máquina para simular a inferência dos editores e automatizar o processo. Nessa pesquisa, utilizaremos técnicas de *Web Scraping* para coletar dados da página Web do IFE e aplicamos algoritmos de aprendizado supervisionado para categorizar automaticamente os resumos de notícias. A proposta é que essa técnica possa ajudar os editores do IFE a classificar futuros resumos de forma mais rápida e eficiente, além de oferecer uma análise mais completa e precisa do conteúdo do informativo.

Ao utilizar um *corpus* específico de notícias do setor elétrico, é possível treinar algoritmos de classificação de texto para compreender a linguagem técnica usada na indústria elétrica, incluindo termos específicos, jargões e abreviações. Esse conhecimento especializado pode ajudar a criar algoritmos de classificação mais precisos e eficientes para esse setor. Ademais, o IFE é uma fonte de notícias que está sempre atualizada e disponível, o

[2] http://www.gesel.ie.ufrj.br/index.php/Ifes, acessado em 12/05/2022

que pode ser vantajoso para investidores, empresas, reguladores e outras partes interessadas no setor elétrico. Com uma classificação automática mais acurada, é possível realizar uma análise mais ampla e eficiente do mercado elétrico, auxiliando na tomada de decisões informadas.

Para realizar a categorização dos resumos de notícias, os algoritmos de aprendizado de máquina podem utilizar diversos recursos do texto, como palavras-chave, frequência de termos, padrões sintáticos e/ou semânticos para identificar as características mais relevantes de cada resumo. Para tanto, podem ser aplicadas diversas técnicas de mineração de dados, mineração na Web, recuperação da informação, processamento de linguagem natural e aprendizado de máquina e é complicado traçar fronteiras bem definidas entre essas áreas de pesquisa. Cada uma dessas técnicas - algumas bem sedimentadas, outras em evolução - desempenha um papel importante no processo de classificação de texto, ajudando a extrair informações relevantes do texto, identificar padrões, tendências e assim, classificar os dados de maneira mais precisa.

Este trabalho tem como objetivo apresentar os resultados da comparação de seis diferentes algoritmos de aprendizado de máquina: k-Vizinhos-Mais-Próximos (k-NN), Regressão Logística, Naïve Bayes, Máquinas de Vetores de Suporte (SVM), Floresta Aleatória e *Ensemble*. Esses algoritmos foram selecionados por serem amplamente utilizados em problemas de classificação de texto e por serem considerados abordagens clássicas de aprendizado de máquina. A escolha desses seis algoritmos possibilita comparar diferentes tipos de algoritmos para identificar qual deles é o mais adequado para o problema em questão e fornecer informações úteis para futuros trabalhos de classificação de texto. Os algoritmos foram treinados em um conjunto de resumos de notícias que foram previamente categorizados manualmente pelos editores do IFE.

Além disso, buscamos responder às seguintes questões de pesquisa:

- Como os hiperparâmetros dos métodos de aprendizado de máquina afetam os seus desempenhos?
- Quão precisos são os métodos de aprendizado supervisionado de máquina, no âmbito do processamento de linguagem natural, aplicados a esta coleção de resumos de notícias?

O restante deste trabalho está organizado com a seguinte estrutura: no capítulo 2, serão apresentados ao leitor os conceitos fundamentais; no capítulo 3, serão discutidas as técnicas em relação à classificação de textos; no capítulo 4, serão detalhadas as metodologias empregadas nos experimentos realizados. Por fim, no capítulo 5, serão apresentadas as conclusões sobre as questões de pesquisa levantadas, bem como sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Como o capítulo anterior introduziu, a classificação de texto é um processo complexo que é influenciado por várias técnicas de diferentes áreas de pesquisa em Ciência da Computação. Neste capítulo, serão apresentados os conceitos e técnicas fundamentais desde a coleta de dados até a seleção e extração de *features*, com foco na aplicação de técnicas de aprendizado supervisionado de máquina para classificação automática de textos.

2.1 MINERAÇÃO DE TEXTO

Uma enorme quantidade de novos dados vêm sendo disponibilizados na Web continuamente. Por isso, pode ser vista como um enorme banco de dados, dinâmico e interconectado. É possível encontrar os dados, quando em formato digital, em qualquer uma das três formas (SUMATHY; CHIDAMBARAM, 2013):

- Estruturada: os dados possuem um padrão predefinido com uma estrutura rígida, bem definida e planejada;
- Não estruturada: são dados flexíveis, dinâmicos, sem uma estrutura bem definida;
- Semiestruturada: os dados possuem uma estrutura heterogênea, parcialmente estruturada e possuem algumas características definidas, mas não se limitam a uma estrutura rígida. Uma mistura dos dois tipos anteriores.

Dados em formato textual é uma das formas mais simples de informação não estruturada que podem ser geradas na maioria dos cenários (ALLAHYARI et al., 2017). A quantidade de dados nesse formato segue em crescimento devido ao volume de documentos disponíveis em formato eletrônico, como e-mails, relatórios, notícias, etc. (KHAN et al., 2010). O objetivo da mineração de textos é identificar padrões, tendências e correlações em grandes conjuntos de dados (CASTRO; FERRARI, 2017).

A mineração de textos pode ser definida como um processo de descoberta de padrões ocultos, úteis e interessantes a partir de documentos de texto não estruturados (SUMATHY; CHIDAMBARAM, 2013). Dessa forma, podemos dizer que a classificação de textos é uma técnica de mineração de textos (ALLAHYARI et al., 2017).

2.2 MINERAÇÃO NA WEB

Citada e estudada desde os anos 90, a mineração na Web possui a mesma metodologia da mineração de texto. Pode ser vista como uma das etapas fundamentais para obtenção do conjunto de dados a ser utilizado pelo modelo de classificação, que é a coleta de

Figura 1 – Estrutura do Web Scraping.

Fonte: adaptado de (SIRISURIYA, 2015, p. 136).

documentos, sendo que o primeiro passo é definir quais serão as fontes de dados. Porém, recuperar, utilizar e compreender os dados disponíveis na Web é uma tarefa singular, uma vez que os dados, em geral, são mais complexos, heterogêneos e dinâmicos que os armazenados em bancos de dados tradicionais.

Com o avanço das pesquisas e das tecnologias da computação, aliadas às técnicas de análise de dados, o vasto volume de material não estruturado e não rotulado da Web ganhou valor significativo. Isso porque a tecnologia moderna já nos permite processar grandes volumes de dados rapidamente e com pouco esforço computacional. Estruturados ou não, em diferentes formatos e de diferentes fontes, esses dados podem ser considerados como um valioso recurso para uma variedade de aplicações, como pesquisas científicas, acadêmicas e de mercado (PERSSON, 2019).

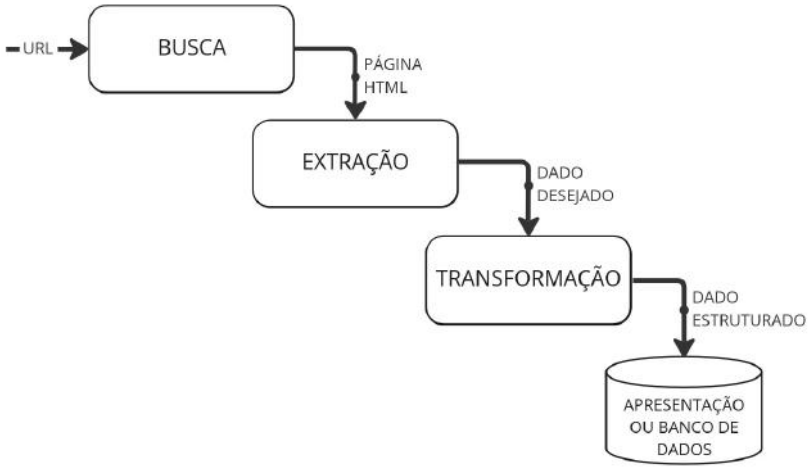
2.3 WEB SCRAPING

O *Web Scraping* (Figura 1) é uma tecnologia de mineração aplicada em páginas Web (SAURKAR; PATHARE; GODE, 2018). Uma das ferramentas mais valiosas para os cientistas de dados, pois permite, a um baixo custo, a extração de uma enorme quantidade de dados (ZHAO, 2017). Com ela, podemos extrair informações e dados de uma página Web de forma automatizada. Esses dados podem incluir textos, imagens, *links*, tabelas, entre outros elementos presentes na página (SIRISURIYA, 2015).

Como os sites podem ser construídos usando linguagens, *frameworks*, e formas variadas, é importante considerar as diferentes opções para fazer a coleta de dados em um projeto específico, que podem ser definidas ponderando a fonte dos dados, o contexto da pesquisa ou do desenvolvimento do produto, o objetivo final, e o *output* (SIRISURIYA, 2015).

A fonte dos dados pode ser privada, ou seja, documentos que estão localmente armazenados na mesma máquina, desenvolvidos pelo pesquisador ou cedidos pelo proprietário, ou pode ser de fontes públicas, como documentos disponibilizados na Internet (CHAU-LAGAIN et al., 2017). Os documentos podem ser seções de páginas Web como *tweets* do Twitter, currículos, notícias jornalísticas, transcrições de entrevistas, relatórios, etc.

Figura 2 – Processo de Web Scraping.



Fonte: adaptado de (PERSSON, 2019, p. 5).

Quanto ao contexto da pesquisa, podemos estar interessados em observar os resultados de desempenho ou as reações dos alunos, analisar sentimentos, desejos e opiniões de clientes, etc. No que tange o objetivo final, podemos considerar a análise contextual, modelagem de tópicos ou a classificação. Por fim, o *output* pode ser arquivos em formato *txt*, *csv*, ou um banco de dados. Todos os fatores devem ser levados em consideração para que a melhor forma de extrair os dados seja escolhida (MUNZERT et al., 2015).

A raspagem de dados na Web (*Web Scraping*) envolve basicamente a utilização de dois *softwares*: o *crawler* e o *scraper*. No primeiro momento, o *crawler* baixa os dados na Web de maneira sistemática. Em seguida o *scraper* extrai, codifica e armazena as informações na sua forma bruta em um banco de dados ou arquivo de acordo com uma estrutura e formato definidos pelo usuário. Este novo arquivo é então avaliado de forma que a apresentação inicial de dados na Web não permitia diretamente (FARLEY; PIEROTTE, 2017).

Como ilustrado na Figura 2, a operação de um *Web Scraper* é dividida em 3 etapas (PERSSON, 2019):

- Busca: etapa em que o *crawler* acessa, identifica a fonte de dados e obtém acesso a ela;
- Extração: nesta etapa, o *scraper* extrai os dados relevantes das páginas Web e os armazena em uma estrutura de dados. Isso pode incluir informações como títulos de página, preços de produtos, avaliações de clientes e muito mais;
- Transformação: os dados coletados são estruturados e organizados de uma maneira que possa ser utilizada pelo usuário final. Isso pode incluir a limpeza dos dados,

remoção de informações desnecessárias e a organização dos dados em um formato que possa ser facilmente lido ou usado em outras aplicações.

2.4 PREPARAÇÃO DOS DADOS

O pré-processamento do *corpus* obtido na etapa de *Web Scraping* é uma etapa fundamental na preparação dos dados para a classificação. Cada técnica de pré-processamento tem como objetivo abordar um problema específico, visando preparar adequadamente os dados para o treinamento dos métodos de classificação e reduzir as chances de erros ou resultados imprecisos.

A aplicação das técnicas de pré-processamento busca eliminar, tanto quanto possível, do documento os fatores dependentes da linguagem. Podem ser aplicadas as seguintes técnicas (KORD; MAHENDER, 2012):

1. *Tokenização*: a primeira e mais simples técnica é dividir o documento completo em frases, ou mais comumente em palavras, dependendo da aplicação. Nesta etapa, sinais de pontuação podem auxiliar a identificar a separação dos *tokens* e podem ser descartados. O resultado desta etapa é um conjunto de *tokens* e é neste conjunto, que as técnicas seguintes serão aplicadas.

2. Remoção de *Stopwords*: de uma maneira geral, são eliminadas os *tokens* que representam palavras comuns que podem ser consideradas pouco relevantes, neste caso, para a classificação mais precisa dos documentos. Normalmente, trata-se de remover alguns *tokens* que são conhecidos e já estão listados, uma vez que pertencem a um conjunto pequeno e limitado de palavras, e.g, preposições, artigos, conjunções, pronomes, etc.

3. *Stemmização*: é um processo que envolve a redução de palavras a um *stem* (raiz). A ideia por trás da *stemização* é simplificar a análise de textos, agrupando diferentes variações da mesma palavra em uma única forma (*stem*), para que possam ser tratadas de forma igual. Aplicando esta técnica reduzimos de forma significativa a dimensão do vetor que representará cada documento de nosso *corpus* (conforme descrito na seção 2.5).

2.5 EXTRAÇÃO DE *FEATURES*

O classificador não interpreta diretamente o texto. Por isso, a representação do texto é um fator crucial para denotar as características mapeadas e facilitar a análise profunda do texto, sendo importante que esse mapeamento seja aplicado uniformemente em todo o *corpus*.

Um documento geralmente é dividido em unidades menores (*tokens*, ou *n-grams*) e cada documento pode ser representado pelo conjunto destas unidades (*Bag of Words* (BoW)) (ALLAHYARI et al., 2017).

O conjunto de todos os *tokens* distintos encontrados em um *corpus* é chamado de vocabulário, e serão as *features* utilizadas na construção da representação vetorial de cada documento de nosso *corpus*. Formalmente, dada uma coleção de documentos $D = \{d_1, d_2, \cdots, d_{|D|}\}$ e um vocabulário $V = \{t_1, t_2, \cdots, t_{|V|}\}$, a frequência do *token* $t \in V$ no documento $d \in D$ é denotado como $f_d(t)$ e o número de documentos que possuem o *token* t é representado por $f_D(t)$.

Desta forma, os valores que cada uma destas *features* recebe indica a importância de um determinado *token* com relação a um dado documento. Não é difícil perceber que quanto maior o vocabulário, maior será o espaço vetorial necessário para representar um documento. Em um cenário com grande quantidade de documentos possuindo grande variedade de *tokens*, o classificador receberá uma representação do *dataset* que terá alta dimensionalidade.

Os modelos vetoriais que são mais comumente utilizados são (ALLAHYARI et al., 2017):

a) Modelo Booleano: um peso $w_d(t) > 0$, fixo e predefinido, é atribuído a cada *token* $t \in V$ que aparece no documento $d \in D$. Se o *token* t não aparece no documento d, $w_d(t) = 0$. É comum a utilização de pesos binários, $w_d(t) = 1$ denotando a presença do *token* t no documento d.

b) Modelo TF-IDF (em inglês *term frequency–inverse document frequency*): seja $w(t)$ a função que computa o peso do termo $t \in d$, então

$$w_d(t) = f_d(t) \times \log(\frac{|D|}{f_D(t)})$$

A frequência do *token* ($f_d(t)$) é normalizada pela fração inversa em escala logarítmica dos documentos que contém o *token*. Portanto, a frequência inversa do documento ($\log(\frac{|D|}{f_D(t)})$) que dá uma interpretação estatística da especificidade do *token* t. A função $w(t)$ é o resultado do produto de duas estatísticas e valora duas intuições:

(i) quanto mais frequentemente um *token* ocorre em um documento, mais representativo é para aquele documento; e

(ii) quanto mais documentos um *token* ocorrer, menos especificador, ou seja, o *token* pouco colabora para a definição da categoria do documento.

A matriz termo-documento é criada dispondo cada documento $d_j \in D$ em uma coluna e um *token* (termo de indexação) $t_i \in V$ para cada linha. Os pesos $w_d(t)$ são calculados

e inseridos na matriz, em cada interseção entre documento e *token*, como mostrado no Quadro 1. Fica claro notar que se trata de uma matriz esparsa, uma vez que em coleções com razoável número de documentos, a maioria dos *tokens* não vão aparecer em um documento, com isso $w_{d_j}(t_i) = 0$ já que $f_{d_j}(t_i) = 0$.

Quadro 1 – Exemplo da matriz termo-documento.

	d_1	d_2	...	$d_{\|D\|}$
t_1	$w_{d_1}(t_1)$	$w_{d_1}(t_1)$	...	$w_{d_1}(t_1)$
t_2	$w_{d_2}(t_2)$	$w_{d_2}(t_2)$	...	$w_{d_2}(t_2)$
...	...	...	...	...
$t_{\|V\|}$	$w_{d_{\|D\|}}(t_{\|V\|})$	$w_{d_{\|D\|}}(t_{\|V\|})$	...	$w_{d_{\|D\|}}(t_{\|V\|})$

Um vetor W_j que representará o documento d_j no espaço-vetorial será composto pelos pesos calculados para cada termo do vocabulário (BAEZA-YATES; RIBEIRO-NETO, 2013),

$$W_j = (w_{d_j}(t_1), w_{d_j}(t_2), \cdots, w_{d_j}(t_{|V|}))^T$$

2.6 SELEÇÃO DE *FEATURES*

Em muitos casos, o vocabulário pode ser muito grande e redundante. Portanto, a seleção de *features* é um passo importante para reduzir a dimensionalidade dos vetores que representam os documentos e, assim, melhorar a eficiência do modelo. Existem três categorias de métodos de seleção de *features* (SHAH; PATEL, 2016):

- *Wrappers methods*: envolvem o treinamento de um modelo de classificação e o uso desse modelo para avaliar a importância de cada *token*. Essa abordagem é computacionalmente intensiva, uma vez que envolve a avaliação de cada *token* individualmente.

- *Filters methods*: envolvem a seleção com base em uma medida estatística que é calculada independentemente do modelo de classificação. Essa abordagem é mais rápida do que a abordagem de *wrappers*, uma vez que não requer o treinamento de um modelo de classificação, mas pode ocasionar em perda de informações importantes. Alguns exemplos de técnicas de filtro incluem as seleções baseada em frequência, em informação mútua e em correlação.

- *Embedded methods*: envolvem a seleção de *features* durante o treinamento do modelo de classificação. Isso é feito ajustando os pesos dos *tokens* durante o treinamento do modelo. São mais eficientes em termos de tempo, mas podem ser menos precisos do que os *wrappers*.

Neste ponto, é importante ressaltar que os *tokens* possuem a tendência de serem dependentes. Isoladamente, um *token* pode não indicar a categoria de um documento, mas um grupo de *tokens* pode. Portanto, não se pode remover um determinado *token* porque ele aparece em muitas categorias e não é discriminador. Em conjunto com outros *tokens* ele pode determinar, até exclusivamente, a categoria correta do documento (HAN; KARYPIS; KUMAR, 2001).

Sobre as métricas para estimar esta capacidade dos *tokens* em diferenciar a categoria do documento, Khan et al. (2010, p. 6) consideram que "uma boa métrica de seleção de *features* deve considerar o domínio do problema e as características do algoritmo" (traduzido do original em inglês). Uma das métricas de avaliação que se destacaram na pesquisa deles foi o teste estatístico qui-quadrado (*chi-square*), um dos *filters methods*. O teste qui-quadrado envolve comparar as frequências observadas dos *tokens* em cada categoria com as frequências esperadas, com base em uma distribuição aleatória dos *tokens* nas categorias (SHAH; PATEL, 2016). O resultado desse teste é um valor que indica o grau de dependência entre um termo e uma categoria (BAEZA-YATES; RIBEIRO-NETO, 2013). Se o valor de qui-quadrado for alto o suficiente, pode-se concluir que há uma associação significativa entre os *tokens* e as categorias (CHEN; CHEN, 2011).

No contexto da classificação de textos, as duas variáveis em questão são a presença ou ausência de um *token* em um documento e a categoria atribuída a esse documento. O teste qui-quadrado é aplicado para cada *token* em relação a cada categoria, resultando em um valor de qui-quadrado para cada par *token*-documento. O valor de qui-quadrado é calculado utilizando a equação a seguir:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Onde:

- χ^2 representa o teste do qui-quadrado de um *token*;
- n é o número total de categorias;
- O_i é a frequência do *token* observada na i-ésima categoria;
- E_i é a frequência do *token* esperada na i-ésima categoria.

O valor do qui-quadrado mede a diferença entre a frequência observada e a frequência esperada de um *token* em uma categoria. Quanto maior o valor, maior é a relação (ou dependência) entre o *token* e a categoria, ou seja, maior é a relevância de um *token* para a classificação. Após calcular os valores do qui-quadrado para todas os *tokens* em relação a cada categoria, é possível selecionar os *tokens* mais relevantes utilizando um limite de frequência ou um limite de pontuação.

3 CLASSIFICAÇÃO DE TEXTO

A classificação de textos é a tarefa de encontrar uma função $\hat{\phi}$ que deve atribuir um valor booleano para cada par $(d_i, c_j) \in D \times C$, isto é,

$$\hat{\phi} : D \times C \rightarrow \{verdadeiro, falso\}$$

onde $D = \{d_1, d_2, \cdots, d_{|D|}\}$ é o conjunto de todos os documentos, e $C = \{c_1, c_2, \cdots, c_{|C|}\}$ é o conjunto de todas as categorias (SEBASTIANI, 2002).

Duas hipóteses são necessárias nesse contexto:

1. As categorias são rótulos simbólicos. Não há conteúdo, conhecimento ou significado por trás da forma como são construídas. Não é relevante para a construção do classificador se os rótulos são descritos por texto, números, cores ou símbolos.

2. Apenas o conhecimento endógeno deve ser levado em consideração na construção do classificador, isto é, nenhum conhecimento externo ao documento deve ser fornecido para fins de classificação. Por exemplo, data de publicação, fonte, tipo de documento, etc., não são considerados no processo. O classificador deve ser baseado apenas no texto referente ao conteúdo do documento.

Como o classificador irá atribuir um valor booleano a cada par (d_i, c_j), restrições podem ser impostas ao modelo com a finalidade do documento ser associado com:

- exatamente k categorias do conjunto C;
- com pelo menos k categorias do conjunto C, isto é, limitado inferiormente;
- com no máximo k categorias do conjunto C, isto é, limitado superiormente.

Com isso, podemos dizer que há mais dois tipos de classificação (ZHANG; ZHOU, 2014):

1. Multirrótulo (*multilabel*): quando um documento pode ser vinculado a mais de uma categoria;

2. Rótulo simples (*single-label*): quando precisamos classificar o documento em exatamente uma categoria.

Um caso ainda mais específico da classificação de rótulo simples é a classificação binária. Neste, o documento deve ser classificado levando em consideração o conjunto de categorias $C = \{c_j, \overline{c_j}\}$ caracterizado por conter apenas duas categorias, complementares.

Quando um classificador está sendo desenvolvido para um conjunto de categorias estocasticamente independentes, isto é, para quaisquer duas categorias c_x e $c_y \in C$, o valor de $\hat{\phi}(d_i, c_x)$ não depende do valor de $\hat{\phi}(d_i, c_y)$ e vice-versa, podemos transformar a classificação em $|C|$ problemas independentes de classificação binária. Neste caso, para $j \in \{1, 2, \cdots, |C|\}$, o classificador deverá atribuir o documento à categoria c_j ou à $\overline{c_j}$, complementar à c_j.

Podemos também realizar a categorização de um documento sob duas outras óticas. A primeira, dada uma categoria $c_j \in C$, encontramos todos os documentos $d_i \in D$ que podem ser rotulados com c_j. Essa forma de classificação é chamada baseada em categorias pivô (*category-pivoted*). Alternativamente, dado um documento $d_i \in D$, encontramos todas as categorias $c_j \in C$ em que o documento se enquadre. Essa segunda maneira é chamada de classificação baseada em documentos pivô (*document-pivoted*) (SEBASTIANI, 2002).

É importante entender as duas formas, pois a categorização pode ocorrer em um cenário onde os conjuntos C e D, podem não estar completamente disponíveis no momento da classificação. Além disso, é importante para ajudar a definir o método de construção do classificador.

Por fim, podemos ainda dividir a classificação de textos em mais dois tipos (SEBASTIANI, 2002):

1. Classificação Determinística (*hard classification*): quando o classificador deve classificar o documento sem a ajuda ou interferência de especialistas;

2. Classificação por Ranqueamento (*ranking classification*): ao invés de definir a classificação do documento, poderia apenas ranquear todo o conjunto C de categorias, de forma a auxiliar um especialista humano, sem tomar qualquer decisão quanto à atribuição.

Este trabalho tratará da classificação supervisionada (que usa um conjunto de documentos que já foram classificados manualmente), rótulo simples, baseada em documentos pivô e determinística. Alguns classificadores usarão a estratégia *one-vs-rest*, isto é, resolverão a classificação através da classificação binária. A estratégia de classificação *one-vs-rest* (também conhecida como *one-vs-all*) é um método usado para treinar modelos de classificação multi-classe. O objetivo é transformar o problema em vários problemas binários. Dado um conjunto de categorias $C = \{c_1, c_2, \cdots, c_{|C|}\}$, a estratégia *one-vs-rest* criará $|C|$ modelos binários. Para cada modelo, uma categoria é considerada positiva, e as demais são agrupadas em uma negativa. Para prever a categoria de uma nova amostra, cada um dos modelos é aplicado à amostra, sendo que a categoria positiva do modelo que obtiver a maior pontuação (ou probabilidade) será indicada para o documento (RIFKIN; KLAUTAU, 2004).

3.1 CLASSIFICADORES

Nesta seção, os classificadores utilizados neste trabalho: k-Vizinhos-Mais-Próximos (k-NN), Regressão Logística, *Naïve Bayes*, Máquinas de Vetores de Suporte (SVM), Floresta Aleatória e Comitê de Classificadores (*Ensemble*), são discutidos. De acordo com as suas estruturas é possível tipificar os classificadores em (CASTRO; FERRARI, 2017):

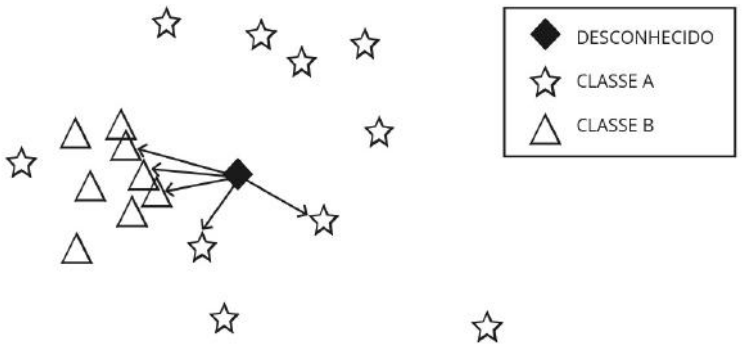
- Baseados em Conhecimento: esse tipo de classificador opera por meio de um conjunto de regras usadas para atribuir determinada categoria a um documento, caso ele satisfaça condições predefinidas;
- Baseados em Árvores: árvores são construídas durante o treinamento do classificador e o nó raiz e os nós intermediários das árvores representam testes sobre um atributo. Já os ramos representam os resultados desses testes e os nós folhas representam as categorias;
- Baseados em Distância: a classificação se dá ponderando a distância entre o documentos cuja categoria se deseja conhecer e um ou mais documentos rotulados;
- Baseados em Função: modelos paramétricos baseados em funções predefinidas e cujos parâmetros são ajustados durante o processo de treinamento.
- Probabilísticos: permitem atribuir uma probabilidade de um documento pertencer a uma ou mais categorias possíveis;
- Conexionistas: os classificadores do tipo conexionistas são aqueles modelos baseados em redes de unidades (nós) interconectadas. O melhor exemplo para essa classificação são as redes neurais artificiais.

Essas classificações atribuída aos algoritmos não são mutuamente exclusivas, e cada tipo de classificador apresenta vantagens e desvantagens, tornando-o mais ou menos adequado para diferentes tipos de problemas e conjuntos de dados. A escolha do classificador ideal dependerá da natureza dos dados, dos requisitos da tarefa de classificação e das características do modelo, como eficiência computacional, precisão e interpretabilidade. Outras etapas já citadas no capítulo anterior, como pré-processamento, extração e seleção de *features*, também podem afetar o desempenho final do modelo.

3.1.1 *k-Vizinhos-Mais-Próximos* (k-NN)

O k-Vizinhos-Mais-Próximos (*k-Nearest Neighbors*) é um algoritmo baseado em distância, e é um modelo chamado de não generalizante, uma vez que, ao invés de tentar criar um modelo geral, ele simplesmente guarda as instâncias dos dados de treinamento

Figura 3 – Classificação pelo algoritmo k-NN.



Fonte: adaptado de (KOWSARI et al., 2019, p. 27).

como pontos dispostos no espaço de $|V|$ dimensões o vetor W_i para o documento $d_i \in D$ (BAEZA-YATES; RIBEIRO-NETO, 2013).

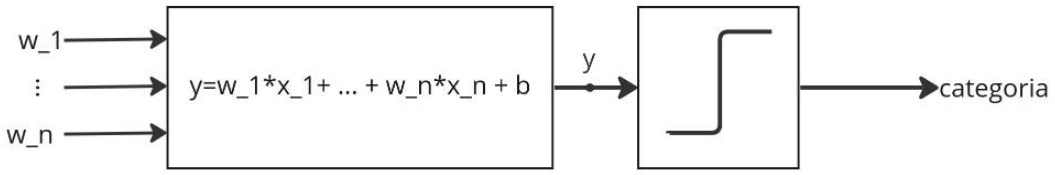
Através da distância (Euclidiana, Ponderada, entre outras) entre vetores, o classificador k-NN vai inferir a categoria c_j de uma nova instância através da maioria simples da contagem dos votos das k amostras vizinhas mais próximas que foram mapeadas no treinamento (CASTRO; FERRARI, 2017). O valor de k é um hiperparâmetro que é escolhido pelo usuário, e quanto maior for o seu valor, teremos uma classificação menos sensível a *outliers*. O inverso é observado quando valores menores de k são selecionados, uma vez que diminui a distinção entres limites das categorias.

Em geral, o k-NN é considerado um algoritmo simples, mas eficaz, especialmente quando se trata de conjuntos de dados com poucas dimensões. O tempo de treinamento é geralmente curto, mas o tempo de classificação pode ser relativamente longo, especialmente para conjuntos de dados de grande escala.

Os votos de cada exemplo vizinho vão contribuir para eleger a categoria que mais pontuou na vizinhança. Esses votos podem, por exemplo, ser uniformes, e valer um mesmo valor para qualquer vetor, ou contribuir o inverso da distância para a categoria do vizinho. A ideia é que quanto mais próximos - em relação às medidas de similaridade - mais semelhantes são os documentos e portanto, pertencem a mesma categoria (ALLAHYARI et al., 2017).

A Figura 3 ilustra o processo de classificação calculando a distância euclidiana. Começando do ponto que possui categoria desconhecida, os vizinhos mais próximos vão sendo identificados. Quando o número de k vizinhos é alcançado, começa a contagem dos votos de cada vizinho atribuindo a soma para as suas respectivas categorias. Entre as categorias desses vizinhos, a que conseguir mais pontos, é a que será inferida para a amostra a qual se pretende classificar.

Figura 4 – Classificação pelo algoritmo Regressão Logística.



3.1.2 Regressão Logística

A Regressão Logística é um modelo discriminativo, baseado em função (ZHU, 2007). Suponha um vetor $W_i = (w_1, \cdots, w_{|V|})^T$ represente as *features* de um documento $d_i \in D$ e o vetor $X_j = (x_1, \cdots, x_{|V|})^T$ que representa as características de uma categoria $c_j \in C$, estimado pelo método da máxima verossimilhança, em que encontra uma combinação de coeficientes que maximiza a probabilidade da amostra ter sido observada. Intuitivamente precisamos mapear W para um valor real $y(W)$, de modo que um valor muito grande seja mapeado para $y(W) = 1$ e um valor muito pequeno seja mapeado para $y(W) = 0$. Esse processo, como a Figura 4 mostra, ser feito por meio de um produto interno

$$y(W) = w_1 \times x_1 + w_2 \times x_2 + \cdots + w_{|V|} \times x_{|V|} + b$$

e uma função degrau.

Para o classificador linear, uma maneira de aprender a probabilidade condicional é substituir a função degrau pela função logística. A função logística é uma função que converte "chances" (*log-odds*) de um evento ocorrer para uma probabilidade - um valor entre 0 e 1 (KLEINBAUM et al., 2002)

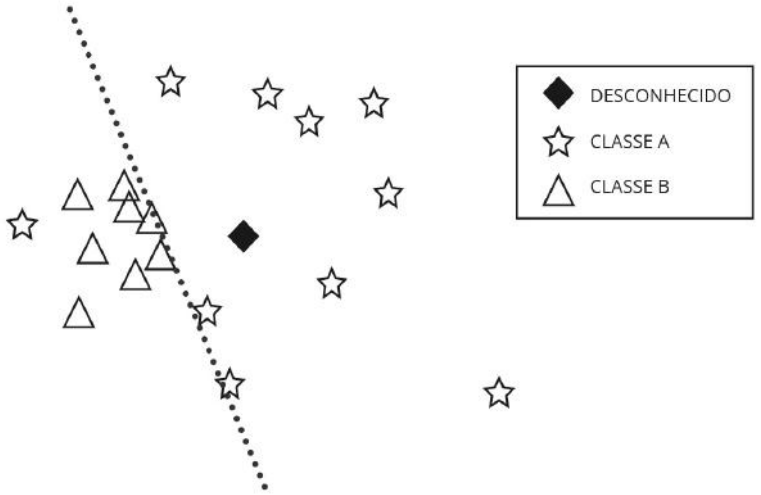
$$S(y(W)) = \frac{1}{1 + e^{-y(W)}}$$

A probabilidade dada pela regressão logística indica o quão distante está o documento do hiperplano separador das categorias (Figura 5). O modelo é um regressor pois a sua saída é um valor real indicando uma probabilidade. Porém podemos atribuir um limiar para o valor da probabilidade para fazer a separação de categorias e dessa forma usá-lo como um classificador.

A regressão logística pode ser generalizada para múltiplas categorias. Dado um conjunto de categorias $C = \{c_1, \cdots, c_{|C|}\}$, a probabilidade do documento pertencer a categoria c_j é dado por

$$p(c_j|w_1, \cdots, w_{|V|}) = \frac{e^{-y(X_i)}}{\sum_{j=1}^{|V|} e^{-y(W_i)}}$$

Figura 5 – Classificação pelo algoritmo Regressão Logística - hiperplano.



onde $y(W_i)$ é o produto interno entre o vetor de características W_i do documento $d_i \in D$ e o vetor X_j de coeficientes estimados no treinamento do modelo para a categoria c_j.

Durante o treinamento do modelo, uma função de custo é aplicada para ajustar seus parâmetros e melhorar a precisão das previsões. Uma função de custo (ou função de perda) é uma medida da discrepância entre as previsões do modelo e as classes reais dos dados de treinamento. Essa função pode ser minimizada usando diversas técnicas de otimização, entre elas (YUAN et al., 2010):

- *Newton-Conjugate Gradient*: usa a matriz Hessiana (segunda derivada) da função de custo para encontrar o mínimo. É adequado para conjuntos de dados de tamanho pequeno a médio. É um dos mais precisos, mas pode ser mais lento em grandes conjuntos de dados;

- *Limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS): é uma aproximação limitada da Broyden-Fletcher-Goldfarb-Shanno (BFGS). O BFGS é um algoritmo de otimização iterativo que usa uma aproximação da matriz Hessiana inversa para calcular a direção de busca mais adequada para minimizar uma função objetivo;

- *Stochastic Average Gradient* (SAG): é um algoritmo de gradiente descendente estocástico. Ele usa amostras aleatórias do conjunto de dados para calcular os gradientes, o que o torna adequado para grandes conjuntos de dados.

Além disso, uma penalidade pode ser adicionada ao custo da função de perda para a suavização dos coeficientes do modelo como, por exemplo, as normas $l1$ e $l2$ do vetor X_j. A norma $l1$ é uma medida de magnitude dos elementos do vetor de coeficientes do modelo. Ela é calculada como a soma dos valores absolutos dos elementos do vetor.

Formalmente, se tivermos um vetor $X = (x_1, x_2, \cdots, x_{|V|})^T$, a sua norma $l1$ é dada por:

$$||X||_1 = |x_1| + |x_2| + \cdots + |x_{|V|}|$$

Já a norma $l2$ dos coeficientes do modelo é definida como a raiz quadrada da soma dos quadrados dos coeficientes, isto é,

$$||X||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_{|V|}^2}$$

As normas $l1$ e $l2$ são amplamente usada em aprendizado de máquina para regularizar modelos de regressão e classificação, onde é comum adicioná-las ao custo da função de perda como uma forma de prevenir *overfitting*. A escolha da técnica mais adequada dependerá das características dos dados e dos objetivos específicos do modelo.

3.1.3 Naïve Bayes

Um dos mais simples e mais utilizados, o classificador probabilístico Naïve Bayes utiliza a regra de Bayes para selecionar a categoria que mais provavelmente gerou o conjunto de *tokens* que recebeu como exemplo (ALLAHYARI et al., 2017).

Dado um conjunto de categorias $C = \{c_1, \cdots, c_{|C|}\}$, como um modelo de probabilidade condicional, Naïve Bayes vai atribuir a cada categoria $c_j \in C$ uma probabilidade

$$p(c_j|w_1, \cdots, w_{|V|})$$

onde $W = (w_1, \cdots, w_{|V|})^T$ é o vetor com as importâncias dos *tokens* que representam a instância que precisa ser classificada e V é o vocabulário.

Se W possuir alta dimensão - cenário que é verificado na classificação de textos - a probabilidade condicional precisa ser decomposta usando o teorema de Bayes para torná-la viável de calcular. Dessa forma ficamos com a seguinte relação

$$p(c_j|w_1, \cdots, w_{|V|}) = \frac{p(c_j) \times p(w_1, \cdots, w_{|V|}|c_j)}{p(w_1, \cdots, w_{|V|})}.$$

Usando a suposição ingênua de independência condicional, isto é, os *tokens* representados por W são independentes uns dos outros, então,

$$p(w_k|c_j, w_1, \cdots, w_{k-1}, w_{k+1}, \cdots, w_{|V|}) = p(w_k|c_j)$$

para todo $w_k \in W$ (CASTRO; FERRARI, 2017). Com isso, podemos simplificar a relação para

$$p(c_j|w_1, \cdots w_{|V|}) = \frac{p(c_j) \times \prod_{k=1}^{|V|} p(w_k|c_j)}{p(w_1, \cdots, w_{|V|})}.$$

Como $p(w_1, \cdots, w_{|V|})$ é constante, o algoritmo faz uso da seguinte regra de classificação

$$p(c_j|w_1, \cdots, w_{|V|}) \propto p(c_j) \times \prod_{k=1}^{|V|} p(w_k|c_j).$$

Algumas implementações utilizam uma técnica chamada suavização, que é usada para evitar que a probabilidade de uma importância numérica de um *token* dada uma categoria seja igual a zero quando não há exemplos de treinamento que apresentem esse *token* nessa categoria, ou seja,

$$p(w_k|c_j) = 0, \text{ para algum } k \in \{1, 2, \cdots, |V|\} \text{ e } j \in \{1, 2, \cdots, |C|\}.$$

Isso é importante porque se $p(w_k|c_j)$ for zero, o classificador atribuirá uma probabilidade zero para toda a categoria, o que pode levar a problemas de classificação.

O hiperparâmetro de suavização do classificador Naïve Bayes é usado para ajustar a suavização das probabilidades. Quanto maior o valor do hiperparâmetro, mais suave será a probabilidade estimada e menor será a influência da frequência da importância numérica do *token* nos cálculos. Um valor baixo para esse hiperparâmetro, por outro lado, pode levar a uma superestimação da frequência da *feature* nos cálculos e pode levar a *overfitting* nos dados de treinamento. Rennie et al. (2003) discutiram vários problemas com os pressupostos do classificador Naïve Bayes, tratando tanto de problemas sistêmicos, quanto de problemas que surgem porque o texto não é realmente gerado de acordo com um modelo multinomial, incluindo a necessidade de suavização de probabilidades e o impacto nos resultados da classificação.

Duas variações são comumente utilizadas do classificador Naïve Bayes para a classificação de textos:

(i) Modelo de Bernoulli Multivariado: um documento é representado por um vetor denotando a presença ou a ausência das palavras no documento.

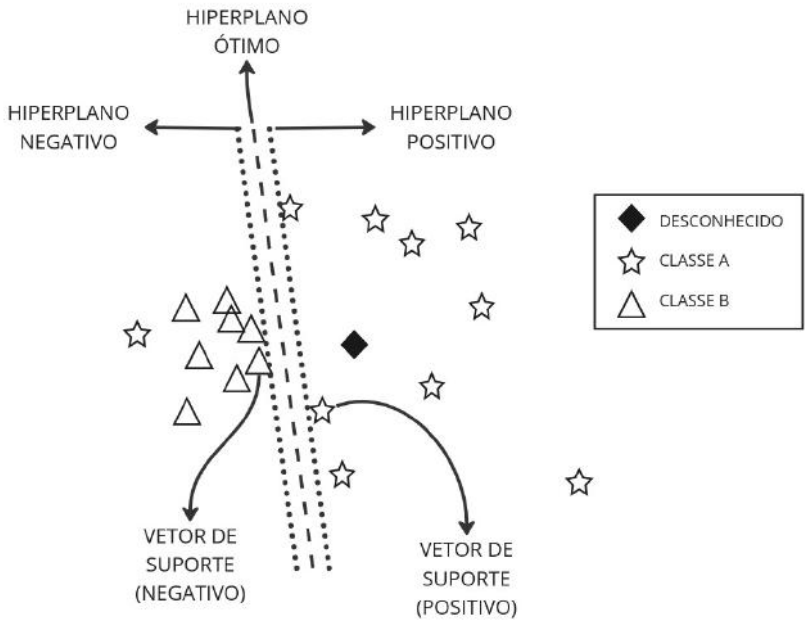
(ii) Modelo Multinomial: entre as diferentes variações do modelo multinomial, uma característica comum é a utilização da frequência dos termos.

Diversas pesquisas foram realizadas comparando os dois modelos. McCallum (1996) concluiu que para vocabulários grandes, quase sempre o modelo Multinomial supera a precisão do modelo de Bernoulli.

3.1.4 Máquinas de Vetores de Suporte - SVM

O classificador baseado no algoritmo Máquinas de Vetores de Suporte (*Support Vector Machine* - SVM), é um algoritmo do tipo baseado em distância que pode ser utilizado para resolver problemas de classificação e regressão. Ele funciona ao encontrar uma fronteira

Figura 6 – Classificação pelo algoritmo SVM.



Fonte: adaptado de (SEBASTIANI, 2002, p. 30).

de decisão que separa as diferentes categorias de dados de maneira o mais precisa possível (CORTES; VAPNIK, 1995). A ideia por trás do SVM é encontrar o hiperplano que maximiza a margem, isto é, a distância entre os pontos mais próximos entre diferentes categorias (BAEZA-YATES; RIBEIRO-NETO, 2013).

Como ilustrado na Figura 6, esses pontos são chamados de vetores de suporte e são responsáveis por determinar a fronteira de decisão. Para isso, o SVM divide o conjunto de treinamento em dois conjuntos distintos, chamado de positivo e negativo. Dado um conjunto de categorias $C = \{c_1, \cdots, c_{|C|}\}$, isto é, uma classificação com $|C|$ categorias, seria o mesmo que separar o conjunto de treinamento $|C|$ vezes em documentos que pertençam a uma determinada categoria (positivos) e os que não pertencem (negativos).

Seja V o vocabulário do *corpus*. Então, os dois conjuntos são mapeados para pontos no espaço com $|V|$ dimensões e os vizinhos de classes diferentes mais próximos neste espaço são identificados. Esses vizinhos caracterizarão dois hiperplanos aos quais pertencem. Ao final, o hiperplano ótimo que está equidistante dos dois hiperplanos dos pontos é definido. Dessa forma, quando novos documentos forem mapeados para o mesmo espaço, estará em um dos lados do hiperplano ótimo, recebendo a sua classificação.

Além disso, o SVM pode ser utilizado para lidar com dados não linearmente separáveis, aplicando técnicas de *Kernel*, que transforma os dados em uma representação em que eles possam ser separados linearmente (KHAN et al., 2010). Uma formalização detalhada de *Kernel Methods* pode ser encontrada em (SHALEV-SHWARTZ; BEN-DAVID, 2014, cap. 16).

O SVM é eficiente em lidar com grandes conjuntos de dados e é uma boa opção para problemas de classificação com muitas dimensões, como é o caso da classificação de textos. No entanto, o processo de treinamento pode ser computacionalmente intensivo, especialmente para grandes conjuntos de dados. Além disso, é sensível a valores atípicos ou *outliers*, o que pode afetar negativamente a precisão do classificador. Assim como o classificador baseado na Regressão Logística, é possível encontrar implementações do classificador SVM que utilizam a penalização em seu algoritmo através das normas $l1$ e $l2$ do vetor de coeficientes do modelo (para detalhes, ver seção 3.1.2).

3.1.5 Floresta Aleatória

Floresta Aleatória (*Random Forests*) é um algoritmo de aprendizado de máquina que se baseia em árvores de decisão (BREIMAN, 2001). Em vez de construir apenas uma única árvore de decisão, o algoritmo constrói várias árvores, cada uma delas treinada em uma amostra diferente do conjunto de treinamento. Ao combinar as previsões de várias árvores, esse algoritmo tem a vantagem de lidar com a variabilidade e a incerteza presentes nos dados, o que pode resultar em previsões mais precisas e robustas do que as obtidas por apenas uma árvore.

As árvores são construídas de maneira recursiva, começando com o nó raiz, que representa todo o conjunto de dados. Em cada nó, o algoritmo escolhe o recurso que melhor divide o conjunto de dados em subconjuntos mais homogêneos, ou seja, subconjuntos onde as observações compartilham características semelhantes. O processo de divisão é repetido recursivamente para cada subconjunto resultante, até que alguma condição de parada seja atingida, como o número máximo de níveis na árvore ou o número mínimo de observações em um nó.

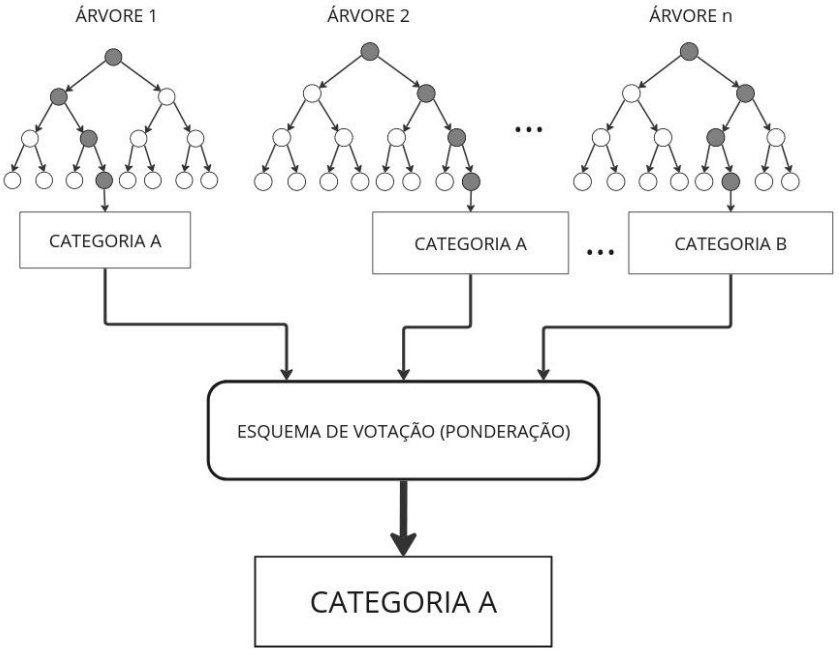
Diversas funções de medida de qualidade (ou critério) podem ser utilizadas para avaliar uma divisão (*split*) em um determinado nó da árvore de decisão. Os critérios são usados para medir a pureza dos nós da árvore de decisão, de forma que uma divisão em um nó seja feita em cima da característica que melhor separa as classes.

Com esse conjunto de árvores, esse algoritmo faz previsões para novos dados combinando as previsões de todas as árvores, como ilustra a Figura 7. A combinação das previsões pode ser feita de várias maneiras, como a votação majoritária, a média das previsões ou com a votação de veto (SHEYKHMOUSA et al., 2020).

Random Forests é muito eficiente e é amplamente utilizado em várias aplicações, incluindo a classificação, a regressão e a detecção de *outliers*. O algoritmo é conhecido por sua capacidade de lidar com dados com alta dimensionalidade e ruído, bem como sua capacidade de evitar o *overfitting*, ou seja, a capacidade de generalizar bem para dados desconhecidos (CUTLER; CUTLER; STEVENS, 2012).

Formalmente, suponha um vetor de importâncias dos *tokens* $W = (w_1, \cdots, w_{|V|})^T$ que representa um documento $d_i \in D$, de $|V|$ dimensões, onde V é o vocabulário da coleção

Figura 7 – Classificação pelo algoritmo Floresta Aleatória.



Fonte: adaptado de (KOWSARI et al., 2019, p. 33).

de documentos. E seja uma variável aleatória Y que representa a probabilidade de d_i pertencer à classe $c_k \in \{c_1, \cdots, c_{|C|}\}$. Assumimos uma distribuição conjunta desconhecida $P_{WY}(W, Y)$. Precisamos encontrar uma função de previsão $f(W)$ para prever Y, definida por uma função de perda $L(Y, f(W))$ que minimiza o valor esperado de L em relação a distribuição conjunta de W e Y

$$E_{WY}(L(Y, f(W)))$$

$L(Y, f(W))$ é uma medida de quão próximo $f(W)$ está de Y e é um hiperparâmetro do classificador, isto é, é uma escolha de quem está treinando o classificador. O propósito disso é identificar que minimizar

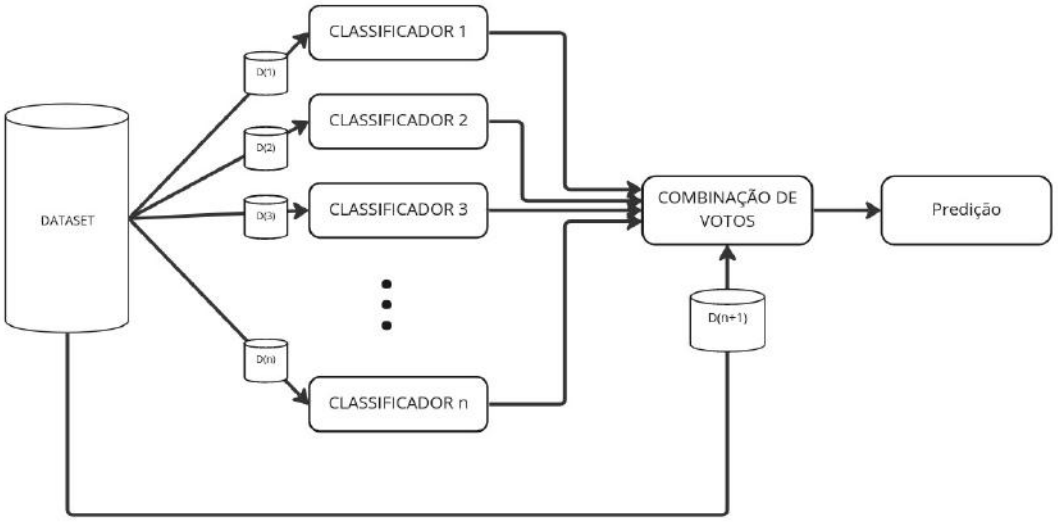
$$E_{WY}(L(Y, f(W)))$$

para a função de perda *zero-one loss* (erro de classificação) é uma função de classificação (CUTLER; CUTLER; STEVENS, 2012)

$$L(Y, f(W)) = 1(Y \neq f(W)) \begin{cases} 0, & \text{se } Y = f(W) \\ 1, & \text{caso contrário} \end{cases}$$

A desvantagem é que o *Random Forest* pode ser computacionalmente caro em grandes conjuntos de dados e pode levar mais tempo para ser treinado do que outros algoritmos de aprendizado de máquina.

Figura 8 – Fase de treinamento do classificador do tipo *Ensemble*.



Fonte: adaptado (HAN; KAMBER; PEI, 2012, p. 378).

3.1.6 Comitê de Classificadores

De forma simples e eficaz, o Comitê de Classificadores (*Ensemble*) permite que você agregue as previsões de vários estimadores com algoritmos diferentes (por exemplo, Árvores de Decisão, Regressão Logística, SVM, etc.) e os combine em uma única previsão, como apresentado na Figura 8. Esse modelo composto que consiste na combinação de classificadores baseia-se na ideia de que, embora cada estimador possa ter sua própria fraqueza, a combinação de vários estimadores pode reduzir a variância geral e melhorar a precisão das previsões reduzindo a variância e o erro de generalização de um modelo (HAN; KAMBER; PEI, 2012).

Existem várias técnicas de *ensemble*, mas as mais comuns são:

- *Bagging (Bootstrap Aggregating)*: é uma técnica que usa amostragem com reposição para criar vários subconjuntos dos dados de treinamento, treina um modelo em cada subconjunto e, em seguida, combina suas previsões;
- *Boosting*: é uma técnica que ajusta sucessivamente modelos fracos (por exemplo, árvores de decisão rasas) aos dados de treinamento de forma iterativa, dando mais peso às amostras que foram classificadas incorretamente pelo modelo anterior;
- *Stacking*: é uma técnica que usa um modelo de nível superior para combinar as previsões de vários modelos de nível inferior, em vez de simplesmente tomar uma média ponderada de suas previsões.

Han, Kamber e Pei (2012) também mencionam a diversidade como fator relevante. Os conjuntos de classificadores base com pouca correlação, aliados a um treinamento com boa

divisão dos dados de treinamento, produzem melhores resultados. A escolha da técnica de *ensemble* correta depende do problema específico e dos dados envolvidos. No geral, tem sido usado com sucesso em muitas aplicações práticas, incluindo detecção de fraude, diagnóstico médico e previsão de séries temporais, tornando-se uma ferramenta valiosa para cientistas de dados e engenheiros de aprendizado de máquina.

3.2 TÉCNICAS DE VALIDAÇÃO

Uma característica comum dos classificadores é demostrar uma previsão precisa para os dados de treinamento, enquanto não consegue bons resultados com novos dados. De forma mais frequentemente vista, o conjunto de dados é dividido em duas partes: uma para o treinamento e outra para teste. Quando se faz necessário avaliar variações de hiperparâmetros, um terceiro conjunto, exclusivo para esta tarefa, é indicado.

Portanto, os conjuntos de dados em aprendizado de máquina podem ser divididos em três tipos principais (HASTIE et al., 2009):

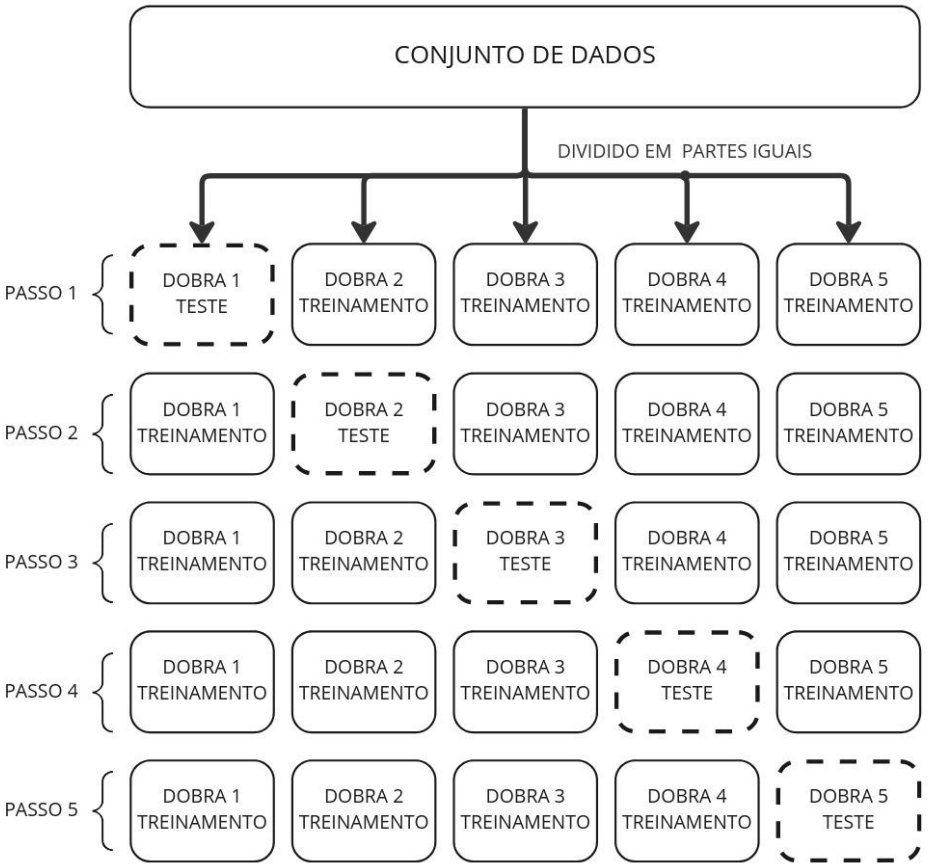
1. Conjunto de ajuste de hiperparâmetros: uma porção do conjunto de dados que é usada para ajustar os hiperparâmetros do modelo, avaliando o desempenho em diferentes configurações e selecionar a melhor combinação. Os hiperparâmetros são as configurações do modelo que não são aprendidas durante o treinamento, mas que afetam seu desempenho;

2. Conjunto de treinamento: é a porção utilizada para treinar o modelo, isto é, ajustar os parâmetros. É um subconjunto do *dataset* que contém exemplos de documentos e as categorias correspondentes que o modelo tentará aprender a partir dos dados. O modelo é ajustado minimizando os erros de previsão com base nos exemplos de entrada e saída no conjunto de treinamento;

3. Conjunto de teste: é a porção do conjunto de dados que é usada para avaliar o desempenho do modelo após o treinamento. O modelo treinado é usado para prever as saídas correspondentes às entradas do conjunto de teste e o desempenho é avaliado comparando as previsões com as saídas reais do conjunto de teste. O conjunto de teste é importante para avaliar a capacidade de generalização do modelo, ou seja, como o modelo se comporta em dados não vistos durante o treinamento.

É importante manter essas divisões distintas e separadas para garantir que o desempenho do modelo seja avaliado de forma justa e imparcial. Se o mesmo conjunto de dados for usado para ajuste de hiperparâmetros, treinamento e avaliação final, pode haver um risco de *overfitting*, em que o modelo se ajusta demais aos dados de treinamento e não consegue generalizar bem para novos dados.

Figura 9 – Validação cruzada com 5 dobras.



Fonte: adaptado de (CASTRO; FERRARI, 2017, p. 322)

.

Uma das técnicas mais aceitas para estimar o desempenho da generalização do classificador é a validação cruzada, que fornece uma solução geral para essa escolha. Os dois objetivos da validação cruzada são (YADAV; SHUKLA, 2016):

- visualizar a generalização de um classificador; e
- encontrar o algoritmo mais adequado ao conjunto de dados comparando os seus desempenhos.

Um dos métodos de validação cruzada é o *K-fold*, onde o conjunto de dados é dividido em k partes iguais ou quase iguais. Esse conjunto de partes, também chamadas de dobras (*folds*), serão utilizadas para os treinamentos e testes que ocorrerão da seguinte forma:

(i) uma parte é separada para teste; e

(ii) as $k - 1$ partes restantes são utilizadas para o treinamento do classificador.

Esse processo é repetido k vezes, em cada uma, selecionando uma dobra que ainda não foi utilizada para a etapa de teste, como demonstrado na Figura 9 para $k = 5$. É

importante observar que os dados são comumente estratificados antes de serem divididos, ou seja, são organizados de tal forma que uma dobra é um bom representante de todo o conjunto.

3.3 AVALIAÇÃO DE PERFORMANCE

As métricas de avaliação de performance de um classificador são usadas para medir o desempenho de modelos de aprendizado de máquina em problemas de classificação, regressão ou clusterização. Elas ajudam a avaliar a eficácia do modelo e a escolher o mais adequado para o problema específico.

Algumas das métricas mais comuns incluem (CASTRO; FERRARI, 2017):

- Acurácia (*Accuracy*): proporção de previsões corretas em relação ao total de previsões, isto é, a capacidade do classificador de fazer uma classificação correta. É a métrica mais comumente usada, mas pode ser enganosa em casos de desequilíbrio de classes.

$$\text{Acurácia} = \frac{\text{previsões corretas}}{\text{total de previsões}}$$

  onde *previsões corretas* é o número de previsões feitas pelo modelo que são iguais aos valores reais, e *total de previsões* é o número total de previsões feitas pelo modelo.

- Precisão (*Precision*): proporção de previsões corretas para uma categoria em relação ao total de previsões para uma categoria.

$$\text{Precisão} = \frac{\text{verdadeiros positivos}}{\text{verdadeiros positivos} + \text{falsos positivos}}$$

  onde *verdadeiros positivos* é o número de instâncias positivas classificadas corretamente pelo modelo, e *falsos positivos* é o número de instâncias negativas que o modelo classificou como positivas.

- Revocação (*Recall*): proporção de previsões corretas para uma categoria em relação ao total de elementos de uma categoria.

$$\text{Recall} = \frac{\text{verdadeiros positivos}}{\text{verdadeiros positivos} + \text{falsos negativos}}$$

  onde *verdadeiros positivos* é o número de instâncias positivas classificadas corretamente pelo modelo, e *falsos negativos* é o número de instâncias positivas que o modelo classificou como negativas.

- F1-*score* (F1-*measure*): é a média harmônica entre a precisão e a revocação.

$$F_1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}}$$

  onde Precisão e *Recall* são definidas nas equações anteriores.

A avaliação da performance para um problema de classificação multi-classe pode ser calculada das seguintes maneiras (SOKOLOVA; LAPALME, 2009):

- Micro-média: a medida é calculada para cada uma das categorias no conjunto $C = \{c_1, c_2, \cdots, c_{|C|}\}$ e então o valor cumulativo é utilizado para o cálculo da métrica para o classificador;

- Macro-média: é uma média aritmética das métricas de desempenho calculadas para cada categoria, dando peso igual a todas as categorias, independentemente do número de amostras em cada categoria. Isso significa que cada classe tem o mesmo peso na média, independentemente de quão grande ou pequena ela seja.

- Média ponderada (*weighted*): as métricas de desempenho são calculadas para cada categoria e, ao final, a média final é calculada ponderando os valores pelo suporte, isto é, do número de amostras de cada categoria.

Para a classificação de rótulo único, precisão e *recall* calculadas utilizando a micro-média, possuem o mesmo valor, que é igual ao da acurácia pois as somas dos *falsos negativos* e *falsos positivos* de todas as classes resultarão no mesmo valor e portanto, as duas métricas serão calculadas com o mesmo denominador. Logo, podemos concluir que o valor de *F1-score* calculado pela micro-média também é igual ao da acurácia para qualquer categoria.

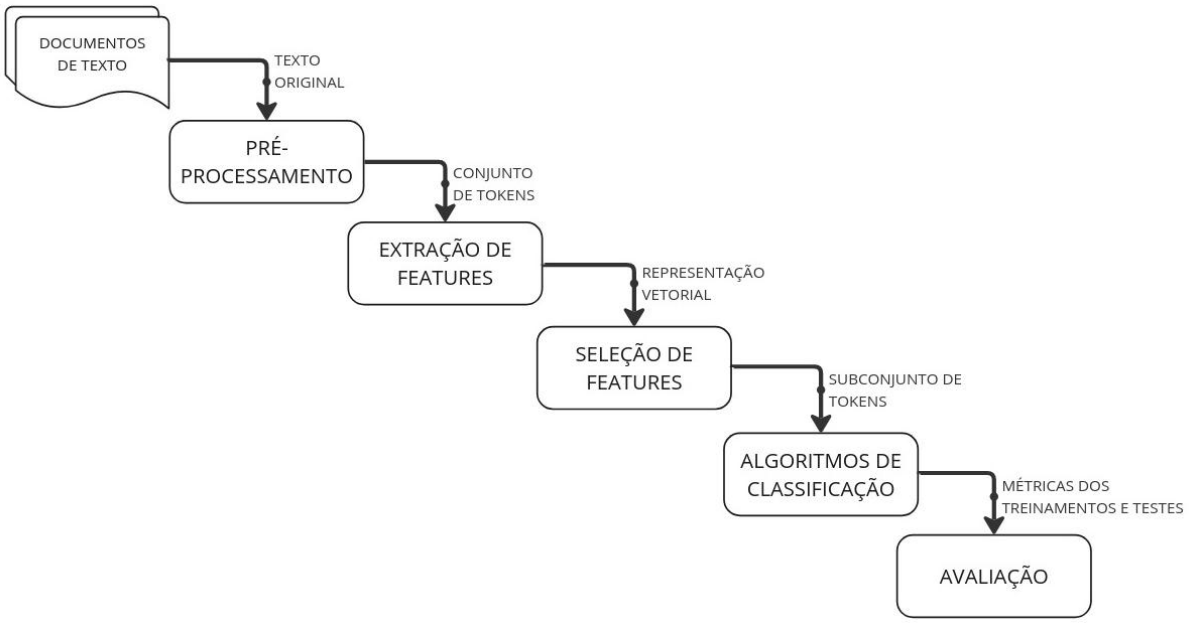
4 METODOLOGIA

Para o desenvolvimento prático deste trabalho, foi utilizada a plataforma *Google Colaboratory*[1], que disponibiliza um ambiente de código aberto, baseada na nuvem, para desenvolvimento e execução de código em linguagem *Python*. Ela é hospedada pelo Google e oferece acesso a recursos computacionais, como CPUs, GPUs e TPUs, sem a necessidade de configurar um ambiente de desenvolvimento local, incorporando uma maneira fácil de colaborar em projetos, compartilhar códigos, dados e resultados com outras pessoas.

Além de oferecer uma interface para escrever e executar códigos, armazenar e acessar arquivos no Google Drive, importar bibliotecas e módulos a partir de repositórios do *Github* e do *PyPI*, já inclui as bibliotecas mais populares de ciência de dados. Como se trata de um ambiente de colaboração, com recursos alocados por demanda, não é possível obter um ambiente estável para avaliar tempos de execução nos procedimentos de treinamento e teste dos métodos de classificação. Os *scripts* desenvolvidos para essa pesquisa podem ser visualizados em 3 *notebooks* do *Google Colaboratory*, disponíveis no seguinte diretório do Google Drive: https://bit.ly/PF-Thiago.

A metodologia aplicada nesta pesquisa segue as etapas ilustradas na Figura 10. As etapas incluem: Coleta de dados (seção 4.1); Pré-processamento (seção 4.2); Extração de *features*; Seleção de *features* (seção 4.3); Treinamento dos classificadores; e Avaliação dos resultados (seções 4.4, 4.5, 4.6 e 4.7); e serão discutidas nas seções a seguir.

Figura 10 – Etapas da Classificação de Textos.



[1] https://colab.research.google.com/, acessado em 18/02/2022

4.1 COLETA DE DADOS

O principal instrumento para a construção de um modelo de categorização automática de texto utilizando aprendizado supervisionado de máquina é uma coleção de documentos previamente classificados em categorias por especialistas. Sendo assim, a fonte dos dados usada para este trabalho é o Informativo Eletrônico do Setor Elétrico (IFE), elaborado pela equipe do Grupo de Estudos do Setor Elétrico (GESEL) do Instituto de Economia da Universidade Federal do Rio de Janeiro (UFRJ).

Criado em 1997, com o objetivo de desenvolver pesquisas e análises econômicas sobre o setor elétrico do Brasil e do mundo, o GESEL[2] é composto por especialistas e pesquisadores multidisciplinares, de diversas instituições nacionais e internacionais. Presente nos principais eventos, o grupo atua aproximando, buscando soluções e auxiliando as principais organizações do setor, governamentais ou privadas. Dessa forma, o GESEL é um importante ator no cenário nacional quando se trata de decisões vinculadas ao planejamento, regulação e modelação do setor elétrico.

O grupo edita periodicamente o IFE, com as principais notícias do setor elétrico nacional e internacional. Os editores sintetizam as principais notícias relacionadas ao setor, além de artigos e opiniões de especialistas em breves resumos que informam de maneira rápida e eficiente seus leitores. O informativo editado e distribuído para interessados pré-cadastrados e disponibilizado publicamente, a qualquer interessado, na página do grupo na Internet.

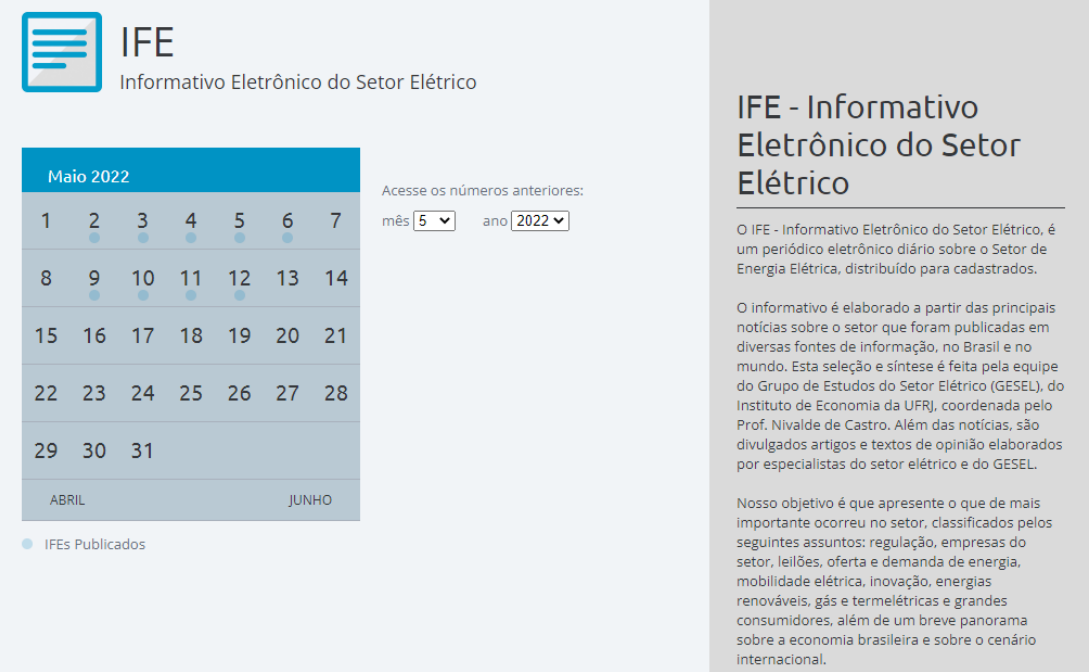
Atualmente o informativo é apresentado em categorias, sendo que cada resumo de notícia está relacionado a exatamente uma categoria indicada pelos editores. O IFE é, portanto, um repositório com resumos de notícias de diversas fontes, classificados, um a um, manualmente por especialistas do grupo.

O contexto desse trabalho é avaliar as métricas de desempenho de métodos de aprendizado de máquina aplicados no *dataset* formado pelo conteúdo dos informativos para concluir quais são mais eficazes na classificação de novos resumos de notícias. O objetivo final é encontrar um método automatizado de classificação que retrate o mais próximo possível a inferência do editor do informativo.

A coleta dos dados ocorreu diretamente no *website* do GESEL. Para essa tarefa, foi desenvolvido um software *Web Scraping* específico para coletar os dados dos informativos. Foram utilizados as ferramentas e recursos disponibilizados pela linguagem de programação *Python* (versão 3.7.13), os métodos disponibilizados pela biblioteca *BeautifulSoup* (versão 4.6.3) e pela biblioteca *Requests* (versão 2.25.1) que abstraem com elegância as requisições HTTP, permitem o acesso do *script* à *Web* e a interpretação dos arquivos HTML.

[2] http://www.gesel.ie.ufrj.br/index.php/Pages/who, acessado em 01/06/2022

Figura 11 – Recorte do *website* do IFE.



Fonte: Site GESEL (acesso 12/05/2022).

4.1.1 Etapa de Busca

O primeiro *script* desenvolvido foi um *crawler* que acessava o *website* do IFE onde, através de um calendário, é possível selecionar e acessar determinado informativo através da sua data de publicação. O calendário (Figura 11) destaca os dias que houve uma publicação e redireciona para a *webpage* em que a edição do informativo está disponibilizada.

No primeiro momento, o *crawler* navega pelo calendário coletando as *URLs* das publicações do informativo e as armazena em uma estrutura de dados chamada *DataFrame*, implementada pela biblioteca *Pandas* (versão 1.5.3). Um *DataFrame*, entre muitas aplicabilidades, organiza de forma otimizada os dados em forma de tabela e neste *script*, cada linha armazenará, conforme ilustrado na Figura 12:

(i) a data apresentada no calendário; e

(ii) o endereço eletrônico (uma *URL*) da *webpage* onde está publicado o informativo.

Ao final dessa operação, realizada no dia 01/05/2022, foram coletadas 4.897 *URLs*. A primeira publicação data 10 de novembro de 2000 e a coleta obteve os endereços até a edição do dia 29 de abril de 2022. Analisando a integridade dos dados, para cada data no calendário do *website* do IFE, apenas uma *URL* foi capturada, isto é, não há datas repetidas. Todavia, foi observado que haviam *URLs* duplicadas no *DataFrame*. Isso indica que há inconsistências entre o dia selecionado no calendário e o informativo

Figura 12 – Resultado da execução do *crawler* no calendário da página do IFE.

	date_ref	url
0	10/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...
1	22/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...
2	23/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...
3	24/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...
4	27/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...
...	...	...
4892	25/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...
4893	26/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...
4894	27/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...
4895	28/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...
4896	29/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...

4897 rows × 2 columns

correto, referente aquele dia. Além disso, algumas informações inusuais chamavam a atenção, como publicações aos domingos.

Em um segundo momento, o *crawler* acessa cada um dos endereços, identifica a estrutura das *tags* HTML e extrai todo o conteúdo em texto da página. Durante a sua execução, das 4.897 *URLs* que foram acessadas, quatro apresentaram problemas e não indicavam o endereço correto para a publicação. Todas foram alteradas manualmente encontrando padrões nas *URLs*, nos subdiretórios, inserindo o protocolo na *URL*, etc. Apesar disso, a publicação do dia 08/09/2014 (edição 3.707) não pode ser extraída do site, pois o servidor não encontrava o conteúdo da página.

Em seguida, um *scraper*, que também foi desenvolvido especificamente para este projeto, separava os dados relevantes da saída do *crawler*. Ao final da execução do *scraper*, o conteúdo dos 4.896 informativos acessados e extraídos com sucesso foram armazenados em um *DataFrame* que relacionava, conforme ilustrado na Figura 13:

(i) a data da publicação, novamente como apresentado no calendário da página do IFE;

(ii) a *URL* que foi acessada e;

(iii) o respectivo conteúdo textual do informativo.

4.1.2 Transformação

Antes da criação do *script* que realizava a transformação, foi notado que o informativo, no decorrer do tempo, possuiu apresentações diferentes, com formatações diferentes. Isso

Figura 13 – Resultado da execução do *scraper*.

	date_ref	url	content
0	10/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...	['Informe Eletrônico Sobre Empresas de Energia...
1	22/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...	['Informe Eletrônico Sobre Empresas de Energia...
2	23/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...	['Informe Eletrônico Sobre Empresas de Energia...
3	24/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...	['Informe Eletrônico Sobre Empresas de Energia...
4	27/11/2000	http://gesel.ie.ufrj.br/app/webroot/files/IFES...	['Informe Eletrônico Sobre Empresas de Energia...
...	...	...	...
4891	25/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...	['IFE: Informativo Eletrônico do Setor Elétric...
4892	26/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...	['IFE: Informativo Eletrônico do Setor Elétric...
4893	27/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...	['IFE: Informativo Eletrônico do Setor Elétric...
4894	28/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...	['IFE: Informativo Eletrônico do Setor Elétric...
4895	29/04/2022	http://www.gesel.ie.ufrj.br/app/webroot/files/...	['IFE: Informativo Eletrônico do Setor Elétric...

4896 rows × 3 columns

Tabela 1 – Alterações na apresentação do IFE.

APRESENTAÇÃO	INÍCIO	FIM	Nº DE PUBLICAÇÕES
#1	10/11/2000	01/02/2001	46 (0.93%)
#2	02/02/2001	29/03/2001	34 (0.69%)
#3	30/03/2001	23/05/2001	34 (0.69%)
#4	24/05/2001	08/05/2003	450 (9.19%)
#5	09/05/2003	09/02/2018	3352 (68.46%)
#6	19/02/2018	14/04/2020	503 (10.27%)
#7	15/04/2020	01/05/2022	242 (4.94%)

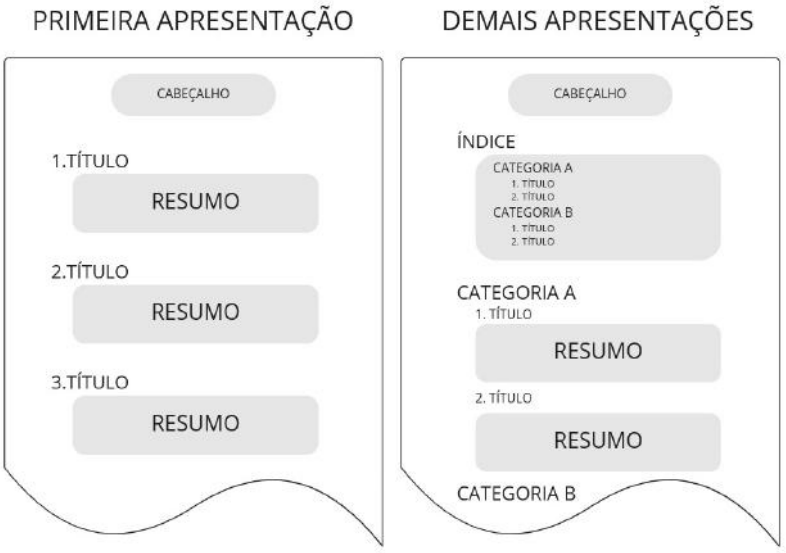
Fonte: Site GESEL (acesso 12/05/2022).

gerou a necessidade de um software adaptável, que pudesse organizar a informação em todas as formatações do informativo. A Tabela 1 mostra as apresentações que foram identificadas observando as publicações.

A primeira apresentação disponibilizada no site, possuía seu conteúdo iniciado com um cabeçalho - contendo a sua edição, data da publicação, editorial e contato - seguido dos títulos da notícias e seus respectivos resumos, isto é, o informativo não possuía categorias. Por conta disso, os resumos dos primeiros 46 informativos, que não possuíam categorias, são armazenadas com a categoria *"uncategorized"* para possível utilização futura e autorizada, porém não são empregados na tarefa de classificação.

Somente a partir da segunda apresentação, o informativo começou a estruturar suas notícias de forma categorizada. Agora, o cabeçalho era seguido de um índice com as categorias criadas por seus editores, seguidas dos títulos das notícias. Após o índice, o informativo seguia exibindo o início da categoria e, os títulos das notícias e os subsequentes resumos. A Figura 14 ilustra os formatos das duas apresentação dos IFEs e a Figura 16

Figura 14 – As diferentes apresentações do IFE.



mostra o conteúdo real de um informativo que é apresentado ao leitor.

O objetivo da etapa de transformação é reestabelecer a estrutura do informativo, mapeando a sequência de caracteres recebidos do *scraper* para um *DataFrame*, onde cada linha armazenará, respectivamente (ver Figura 15):

(i) a data de publicação - pela primeira vez seguindo o apresentado no corpo do informativo e não mais o que era apresentado no calendário da página principal do IFE;

(ii) a *URL* que foi acessada (ocultada na Figura 15);

(iii) a sua edição;

(iv) a categoria da notícia;

(v) o título da notícia; e

(vi) o resumo da notícia.

Durante a execução desse *script*, inúmeras dificuldades precisaram ser superadas. Podemos citar brevemente alguns problemas como:

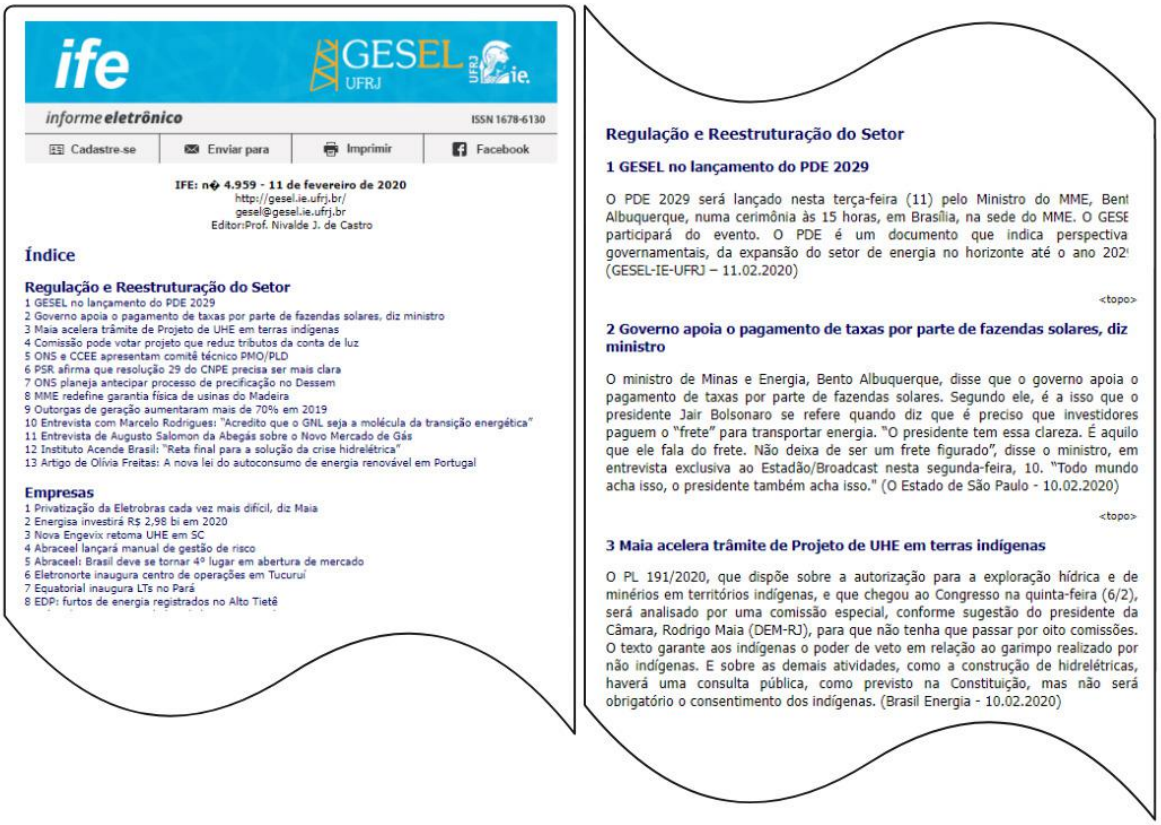
- título da notícia no corpo do informativo diferente do título da mesma notícia no índice, seja por erro de digitação, por abreviações, por paráfrase, etc.;
- sumário com notícia que não era apresentada no corpo do informativo;
- notícia apresentada no corpo do informativo, porém não sumarizada;

Figura 15 – Resultado da etapa de transformação.

	date_ref	edition	category	order	news_title	news_abstract
0	10/11/2000	525	uncategorized	1	Governo irá analisar Angra 2 antes de decidir ...	O governo vai analisar detalhadamente o desemp...
1	10/11/2000	525	uncategorized	2	Tourinho descarta vender Cepisa em 2000	O ministro das Minas e Energia, Rodolpho Touri...
2	10/11/2000	525	uncategorized	3	26 grupos apresentam proposta de pré-qualifica...	A Aneel divulgou, dia 10.11.2000, a lista dos ...
3	10/11/2000	525	uncategorized	4	BNDES ainda não tem viabilização para investim...	O presidente do BNDES, Francisco Gros, disse q...
4	10/11/2000	525	uncategorized	5	SP produzirá 300MW de energia com co-geração	O ministro de Minas e Energia, Rodolpho Tourin...
...	...	...	...	...	...	...
209184	29/04/2022	5476	Energias Renováveis	6	WEG: Mercado de geração solar deverá seguir aq...	A WEG destacou nesta quinta-feira (28) em sua ...
209185	29/04/2022	5476	Energias Renováveis	7	Whirpool: Energia renovável em fábricas no Brasil	A fabricante de eletrodomésticos Whirpool, don...
209186	29/04/2022	5476	Gás e Termelétricas	1	Petrobras: Produção termelétrica recua com mel...	A Petrobras divulgou nesta quinta-feira, 28 de...
209187	29/04/2022	5476	Gás e Termelétricas	2	Quais são as opções da Europa se a Rússia fech...	A russa Gazprom interrompeu o fornecimento de ...
209188	29/04/2022	5476	Gás e Termelétricas	3	Alemanha retira oposição a embargo ao petróleo...	A Alemanha está agora pronta para parar de com...

209189 rows × 6 columns

Figura 16 – Apresentação atual do IFE.



ife

GESEL UFRJ

informe eletrônico

ISSN 1678-6130

Cadastre-se | Enviar para | Imprimir | Facebook

IFE: n° 4.959 - 11 de fevereiro de 2020

http://gesel.ie.ufrj.br/

gesel@gesel.ie.ufrj.br

Editor:Prof. Nivalde J. de Castro

Índice

Regulação e Reestruturação do Setor

1 GESEL no lançamento do PDE 2029
2 Governo apoia o pagamento de taxas por parte de fazendas solares, diz ministro
3 Maia acelera trâmite de Projeto de UHE em terras indígenas
4 Comissão pode votar projeto que reduz tributos da conta de luz
5 ONS e CCEE apresentam comitê técnico PMO/PLD
6 PSR afirma que resolução 29 do CNPE precisa ser mais clara
7 ONS planeja antecipar processo de precificação no Dessem
8 MME redefine garantia física de usinas do Madeira
9 Outorgas de geração aumentaram mais de 70% em 2019
10 Entrevista com Marcelo Rodrigues: "Acredito que o GNL seja a molécula da transição energética"
11 Entrevista de Augusto Salomon da Abegás sobre o Novo Mercado de Gás
12 Instituto Acende Brasil: "Reta final para a solução da crise hidrelétrica"
13 Artigo de Olívia Freitas: A nova lei do autoconsumo de energia renovável em Portugal

Empresas

1 Privatização da Eletrobras cada vez mais difícil, diz Maia
2 Energisa investirá R$ 2,98 bi em 2020
3 Nova Engevix retoma UHE em SC
4 Abraceel lançará manual de gestão de risco
5 Abraceel: Brasil deve se tornar 4º lugar em abertura de mercado
6 Eletronorte inaugura centro de operações em Tucuruí
7 Equatorial inaugura LTs no Pará
8 EDP: furtos de energia registrados no Alto Tietê

Regulação e Reestruturação do Setor

1 GESEL no lançamento do PDE 2029

O PDE 2029 será lançado nesta terça-feira (11) pelo Ministro do MME, Bent Albuquerque, numa cerimônia às 15 horas, em Brasília, na sede do MME. O GESE participará do evento. O PDE é um documento que indica perspectiva governamentais, da expansão do setor de energia no horizonte até o ano 202 (GESEL-IE-UFRJ – 11.02.2020)

<topo>

2 Governo apoia o pagamento de taxas por parte de fazendas solares, diz ministro

O ministro de Minas e Energia, Bento Albuquerque, disse que o governo apoia o pagamento de taxas por parte de fazendas solares. Segundo ele, é a isso que o presidente Jair Bolsonaro se refere quando diz que é preciso que investidores paguem o "frete" para transportar energia. "O presidente tem essa clareza. É aquilo que ele fala do frete. Não deixa de ser um frete figurado", disse o ministro, em entrevista exclusiva ao Estadão/Broadcast nesta segunda-feira, 10. "Todo mundo acha isso, o presidente também acha isso." (O Estado de São Paulo - 10.02.2020)

<topo>

3 Maia acelera trâmite de Projeto de UHE em terras indígenas

O PL 191/2020, que dispõe sobre a autorização para a exploração hídrica e de minérios em territórios indígenas, e que chegou ao Congresso na quinta-feira (6/2), será analisado por uma comissão especial, conforme sugestão do presidente da Câmara, Rodrigo Maia (DEM-RJ), para que não tenha que passar por oito comissões. O texto garante aos indígenas o poder de veto em relação ao garimpo realizado por não indígenas. E sobre as demais atividades, como a construção de hidrelétricas, haverá uma consulta pública, como previsto na Constituição, mas não será obrigatório o consentimento dos indígenas. (Brasil Energia - 10.02.2020)

Fonte: Site GESEL (acesso 12/05/2022).

- resumos de notícias que eram apresentadas com mesmo título, mas com conteúdos diferentes;
- edição com conteúdo parcialmente corrompido;
- data da publicação no informativo não condizia com a do calendário do *website* do IFE que encaminhava para o informativo;
- datas e edições em duplicidade, porém em informativos com conteúdos diferentes;
- edições especiais - de final ou início de ano, com avisos especiais, dias com acontecimentos importantes para o setor, etc;

Além de muitos testes, observações e cuidado, foi necessário lançar mão da biblioteca *NLTK - Natural Language Toolkit* (versão 3.7), com destaque para a função que calculava a distância entre duas *strings*. Com ela foi possível associar os títulos às notícias corretas, vencendo pequenos erros de digitação.

Ao final da sua execução, o *script* conseguiu analisar todos os 4.896 informativos que foram inseridos, e retornou com sucesso o mapeamento de 209.189 resumos de notícias, não conseguindo associar 126 títulos de notícias aos seus respectivos resumos, aproximadamente 0,06% do total analisado.

Neste terceiro processo, a integridade dos dados também foi analisada. Como esperado, as 7 *URLs* que apareceram duas vezes no resultado da etapa de busca, retornaram exatamente o mesmo conteúdo. Além disso, não haviam valores nulos em nenhuma entrada do *DataFrame* e os mais de 209 mil resumos foram classificados em 41 categorias diferentes, salvo os resumos sem categoria foram classificados à parte em outra categoria (*uncategorized*). Alguns detalhes são apresentados no Apêndice A.

4.1.3 Questões éticas

Não foram encontrados os termos de serviços do site durante o processo de coleta, nem o arquivo *robots.txt* na pasta raiz do diretório no servidor do site. Apesar disso, foi usada uma taxa de requisição conservadora de uma requisição a cada cinco segundos, de forma a preservar o acesso e disponibilidade do servidor do site. Os dados coletados não foram em nenhum momento disponibilizados em nenhuma plataforma ou replicados a nenhuma outra pesquisa, site ou editora. Toda informação coletada serviu exclusivamente para treinamento dos modelos de aprendizado de máquina deste trabalho.

4.2 PREPARAÇÃO DOS DADOS

Esta etapa recebe os dados estruturados dos informativos, resultante da etapa anterior - a coleta de dados - e visa reduzir a grande dimensionalidade que um *dataset* de docu-

mentos texto pode gerar. Além disso, visa prepará-los para as etapas que vão permitir as suas interpretações pelos métodos de classificação.

Inicialmente, a base de dados foi reduzida antes do processo de seleção de *features*, a fim de evitar o pré-processamento em dados que não foram considerados relevantes. Uma vez que já foi verificado que o *dataset* não possui nenhum dado vazio ou nulo, a primeira tarefa é retirar os dados replicados. Então, os 251 resumos de notícia dos 7 informativos que estão duplicados são removidos do *dataset*. Com isso, a quantidade de resumos de notícia cai para 208.938.

Sob o prisma das categorias, foi realizada uma análise empírica, onde foi notado que algumas equivaliam a uma única, seja por falha na digitação, seja visando melhorar o nome da categoria no transcorrer do tempo ou pela omissão de alguns termos. Alguns exemplos que ilustram esse cenário:

(i) *Gás e Termelétricas* que possui 13.057 e *Gás e Termoelétricas* com 1.984 resumos de notícias, que se diferenciam pelo vocábulo com mais de uma grafia correta;

(ii) *Bioeletricidade e Eólica* com 976 resumos, *Biomassa e Eólica* com 4 resumos e *Biomassa e eólica* também com 4 resumos de notícias;

(iii) *Biblioteca Virtual do SEE* com 4.056 resumos e *Biblioteca Virtual* com 656 resumos de notícias;

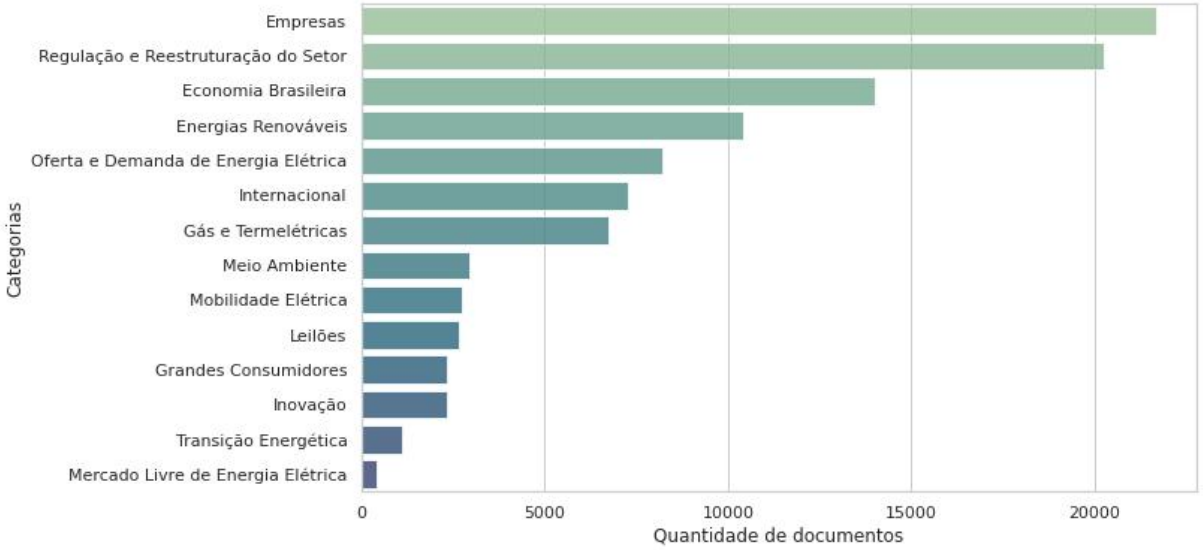
(iv) *Economia Brasileira* com 29.024 resumos e *Economia* com apenas 7 resumos de notícias;

(v) *Regulação e Reestruturação do Setor* com 32.515 resumos, *Regulação e Novo Modelo* com 3.575, *Regulação* com 3.141, *Reestruturação e Regulação* com 1.275 e *Regulação e Reestruturação* com apenas 4 resumos de notícias.

Nota-se que, em todos os casos, há semelhanças nos termos que descrevem a categoria. Além disso, elas são disjuntas no que tange as datas das publicações, ou seja, não ocorrem em uma mesma publicação e seguem uma linha temporal. Isso significa que quando uma deixa de ser utilizada, imediatamente é continuada pela outra. Esses são os argumentos usados para inferir que elas representam uma única categoria. Portanto, categorias que se enquadravam nesta situação receberam o mesmo nome que a mais recente de cada grupo.

Com o propósito de reduzirmos o tamanho do *dataset* para permitir uma execução dos algoritmos em tempo hábil, apenas os resumos publicados nos dez últimos anos foram utilizados. Essa amostragem temporal busca remover as categorias antigas, já não mais utilizadas, com o objetivo de evitar ruído no treinamento do classificador. Essa técnica de amostragem é útil para evitar o sobreajuste (*overfitting*) do modelo a dados antigos e para melhorar a capacidade de generalização do classificador a novos dados. É importante lembrar que ao mesmo tempo que essa operação de amostragem no *dataset* é realizada, uma

Figura 17 – Distribuição dos resumos de notícia por categoria.



operação de filtragem torna-se desnecessária, pois com ela, os primeiros resumos de notícias, atribuídos a categoria "*uncategorized*", também são desprezados. Essa amostragem reduziu o tamanho do *dataset* para 107.607 resumos.

Ainda observando as publicações, foi notado que a categoria *Biblioteca Virtual*, neste ponto já unida com a categoria *Biblioteca Virtual do SEE*, indicava, em grande parte, *links* e/ou descrições para artigos, publicações originais, teses, etc., como uma referência bibliográfica para artigos de interesses. Dessa forma, não consistiam em um resumo de notícia propriamente dito, além de se constituírem por palavras repetidamente usadas, o que poderia acarretar em ruídos e diminuir a precisão dos classificadores. Dessa forma elas não foram consideradas neste trabalho, desprezando então, mais 2.651 resumos que pertenciam a essa categoria. Como ilustra a Figura 17, após essas operações no *dataset*, os resumos passaram a ser distribuídos em 14 categorias.

Para aumentar a proteção a ruídos, os resumos que possuíam menos de 40 palavras também foram removidos do *dataset*. Esse valor, foi obtido empiricamente, observando o conteúdo textual dos resumos que não alcançavam essa marca, geralmente textos repetitivos, frequentemente usados. Essa filtragem por tamanho tem como objetivo melhorar a qualidade do conjunto de dados que será usado para treinar o classificador, pois textos muito pequenos podem não conter informações suficientes para o modelo aprender com precisão ou, em alguns casos, podem introduzir ruído no conjunto de dados. Essa salvaguarda nos permite afirmar que não existe resumo com menos de 13 *tokens* após o pré-processamento e resultou em desconsiderar mais 1.895 resumos de notícias.

Cada um dos 103.061 resumos resultantes são apresentados ao algoritmo de pré-processamento. O primeiro processo desse algoritmo é uniformizar tipograficamente as palavras deixando todas as letras minúsculas. Em seguida, todos os números e pala-

Figura 18 – Resultado do pré-processamento: em cada linha a categoria do resumo e seu conjunto de *tokens*.

	category	tokens
0	Economia Brasileira	setor públic financeir registr marc superávit ...
1	Economia Brasileira	duas seman mal-est govern febraban mead dest m...
2	Economia Brasileira	tax inadimplent empres aument marc dest ano co...
3	Economia Brasileira	ipp alta marc apresent deflaçã fevereir revis ...
4	Economia Brasileira	primeir boletim focus divulg banc central após...
...	...	...
103056	Transição Energética	abril govern biden restaur princip regul lei n...
103057	Transição Energética	tod país projet lider indígen aument econom lo...
103058	Transição Energética	enel brasil delloit farã estud caminh transiçã...
103059	Transição Energética	consumers energy lanc nov program client resid...
103060	Transição Energética	iberdrol acert empréstim verd banc santand apo...

103061 rows × 2 columns

vras com até duas letras, na maioria siglas, são removidos do documento. Logo após, as *stopwords* (preposições, artigos, numerais, etc.) são removidas junto com todos os sinais de pontuação. Neste processo, buscamos obter somente *tokens* com alguma relevância semântica para caracterizar o resumo.

O processo de formação de palavras da língua portuguesa permite uma enorme variedade de flexões, gerando uma infinidade de palavras derivadas. Para moderar esse fator do idioma, lançamos mão do *SnowballStemmer*, mais um recurso disponibilizado pela biblioteca NLTK extremamente eficiente e com suporte à língua portuguesa, que nos permite obter uma forma comum chamada *stem*, removendo caracteres do início e/ou fim do *token* baseando-se nas regras de composição das palavras.

Apesar de parecer complexo, a centena de milhar dos resumos de notícia é processada em aproximadamente cinco minutos no *notebook Colab*. O resultado desse algoritmo é um *DataFrame* com 2 colunas e 103.061 linhas, uma para cada resumo, contendo (ver Figura 18):

(i) a categoria da notícia; e

(ii) os *tokens* do resumo da notícia.

4.3 EXTRAÇÃO E SELEÇÃO DE *FEATURES*

Uma vez que o texto foi pré-processado, o próximo passo é extrair as características relevantes do texto. É a etapa responsável por transformar os dados de texto brutos em um formato que permita ser explorado por um algoritmo de aprendizado de máquina. Utilizando um vetorizador de textos implementado pela biblioteca *Scikit-learn* (versão

1.2.2), os resumos são mapeados para uma matriz termo-documento, onde cada elemento indica a importância de cada *token* em relação ao resumo e ao *corpus* como um todo. O cálculo dessa importância é obtido aplicando a formulação do modelo TF-IDF. O retorno dessa operação é uma matriz esparsa de 103.061 colunas - uma para cada resumo - e 54.364 linhas - uma para cada *token* no vocabulário do *dataset*.

Em seguida, outra implementação disponível na *Scikit-learn*, a *SelectKBest*, separa os melhores 30 mil *tokens* ranqueando-os através do teste estatístico qui-quadrado. O valor específico de 30 mil *tokens* foi definido, empiricamente, através da relação entre as métricas de interesse e do tempo de treinamento dos classificadores. Esse valor foi sendo reduzido gradativamente e as métricas foram observadas.

Dependendo do algoritmo de classificação, as métricas variaram na terceira casa decimal e o tempo de treinamento enquadrou-se num intervalo viável. Com isso, a matriz termo-documento passou a representar um subconjunto dos *tokens* do vocabulário, apresentando 103.061 colunas e 30 mil linhas.

4.4 IMPLEMENTAÇÕES DOS CLASSIFICADORES

Existem vários métodos de classificação automática de textos disponíveis, implementados em diversas bibliotecas e em diferentes linguagens de programação. Cada método possui suas próprias vantagens e desvantagens e seus resultados variam para diferentes *datasets*. Essa pesquisa busca comparar os resultados de seis métodos clássicos, ora chamados de básicos pela comunidade de pesquisa, de aprendizado supervisionado de máquina implementados pela biblioteca *Scikit-learn* ao *corpus* com os resumos de notícias dos IFEs.

Para um dos algoritmos mais presentes em pesquisas, o *k-Vizinhos-Mais-Próximos* (k-NN), foi utilizada a implementação *KNeighborsClassifier*. O classificador baseado na Regressão Logística, também chamado de classificador de máxima entropia, ou classificador *log-linear*, foi empregado através do método *LogisticRegression*.

Uma das abordagens mais simples e amplamente utilizada é o algoritmo de Naïve Bayes, que foi analisado através da aplicação do método *MultinomialNB*. Já o método que implementa o algoritmo SVM que foi utilizado, o *LinearSVC*, utiliza a função de *Kernel* linear e é otimizado para flexibilidade na escolha de penalidades e funções de perda e é melhor dimensionado para grandes números de amostras[3].

Para obter os resultados do classificador baseado na floresta aleatória, lançamos mão do método *RandomForestClassifier*. Por fim, um comitê de classificadores é disponibilizado através do método *VotingClassifier* e utiliza os cinco estimadores descritos anteriormente para inferir a categoria de uma nova amostra.

[3] https://Scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, acessado em 22/01/2023

4.5 DIVISÕES DO *DATASET*

Como todas as implementações dos classificadores possuem hiperparâmetros, se faz necessário encontrar a melhor combinação de valores desses hiperparâmetros para o *corpus* dessa pesquisa. Por conta disso, os dados foram extraídos de maneiras diferentes em dois momentos.

No primeiro momento, precisamos treinar os classificadores para ajustar seus hiperparâmetros. Por utilizar o método *K-fold* ($k = 5$) de validação para cada uma das combinações de valores, o número de treinamentos é alto e todo o processo é demorado e computacionalmente intenso. Portanto, torna-se imprescindível a utilização de um subconjunto do *dataset* para esta etapa ser praticável.

Então, uma subamostra aleatória utilizada exclusivamente para o ajuste dos hiperparâmetros, que represente bem todo o conjunto de dados, é extraída do conjunto de resumos inicial observando a sua distribuição estatística entre categorias. Para isso, foi realizado uma amostragem estratificada, com 25 mil resumos, com o objetivo de encurtar o tempo de busca dos melhores hiperparâmetros do algoritmo e torná-lo mais gerenciável para análise.

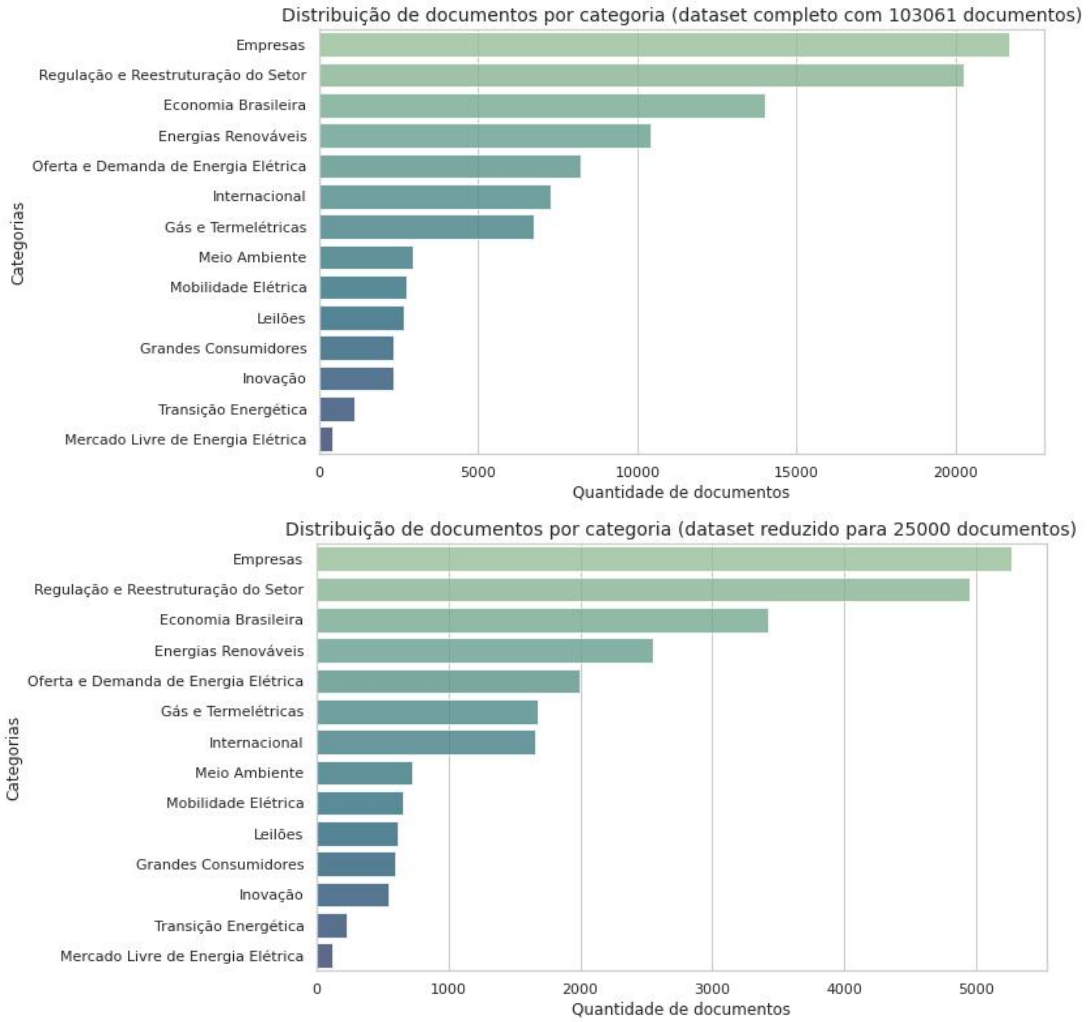
A Figura 19 ilustra que, nesse conjunto de ajuste dos hiperparâmetros, a distribuição estatística foi preservada e ainda se manteve com tamanho grande o suficiente para representar adequadamente o conjunto de dados original. Dentro desse conjunto, o método de validação cruzada *K-fold* ($k = 5$) irá selecionar, a cada rodada de treinamento, 20% para teste e os outros 80% serão utilizados para o treinamento. Esse processo se repetirá outras 4 vezes, sempre selecionando amostras diferentes para teste. O fluxo mais a esquerda da Figura 20 ilustra o procedimento.

No segundo momento, já com a otimização de hiperparâmetros realizada, 80% da outra parte do *dataset* original (cerca de 62 mil resumos de notícias), é aplicada ao método de classificação, que no processo de validação cruzada *K-fold*, também será dividida em cinco partes ($k = 5$). Novamente, quatro dessas dobras são utilizadas para o treinamento, e a restante é separada para o teste do classificador. Assim como no momento de busca dos hiperparâmetros, o processo de treinamento e teste repetido por mais quatro vezes, selecionando uma dobra diferente de teste para cada rodada e treinando o classificador com as demais, analisando a capacidade do modelo de se ajustar aos dados.

Esse processo irá gerar cinco classificadores com diferentes parâmetros ajustados, uma vez que eles foram treinados com conjuntos diferentes, uma característica do *K-fold*. Com isso, o que obteve melhor *F1-score*, isto é, o que melhor se ajustou, será utilizado para prever os 20% restantes do *dataset* original (cerca de 15 mil resumos de notícias). A Figura 20 é uma adaptação da que está na documentação da *Scikit-learn*[4] e ilustra esse procedimento.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Figura 19 – *Dataset* completo e conjunto de ajuste de hiperparâmetros.



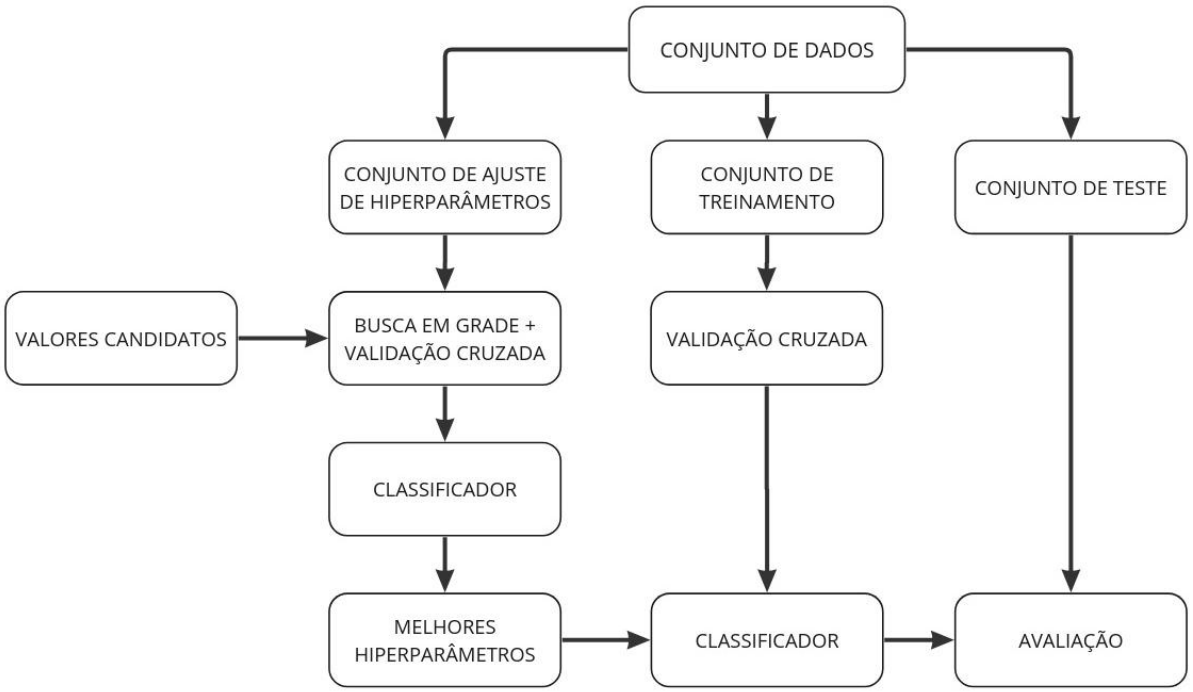
4.6 AVALIAÇÃO DOS HIPERPARÂMETROS

Hiperparâmetros são parâmetros da implementação do modelo que definem como o algoritmo de aprendizado de máquina será configurado e são definidos antes do treinamento do classificador. Para encontrar as melhores combinações para o *dataset* dessa pesquisa, utilizamos a busca em grade (*grid search*).

A técnica de *grid search* é uma técnica de ajuste de hiperparâmetros amplamente utilizada em aprendizado de máquina e consiste em criar uma grade de valores para cada hiperparâmetro que se deseja ajustar, definindo um conjunto de valores possíveis para cada um deles. Então, o modelo é treinado e validado, utilizando a estratégia gulosa, com cada combinação de valores possíveis de hiperparâmetros, e a combinação que apresentar o melhor desempenho na validação é selecionada.

Durante a avaliação dos hiperparâmetros, a macro-média do *F1-score* será levada em consideração como métrica de desempenho. Isso porque o *dataset* é desbalanceado e o objetivo é avaliar a performance geral do modelo, e não avaliar a performance do modelo

Figura 20 – Processo de busca em grade, treinamento e teste.



Fonte: adaptado da documentação *Scikit-learn* (acesso 10/01/2023).

em cada classe de forma individual. Como o *F1-score* é a média harmônica da precisão e do *recall*, também foram observadas a macro-média dessas métricas e a acurácia, para obter informações sobre o impacto do desbalanceamento na eficácia dos modelos.

4.6.1 *k-Vizinhos-Mais-Próximos* (k-NN)

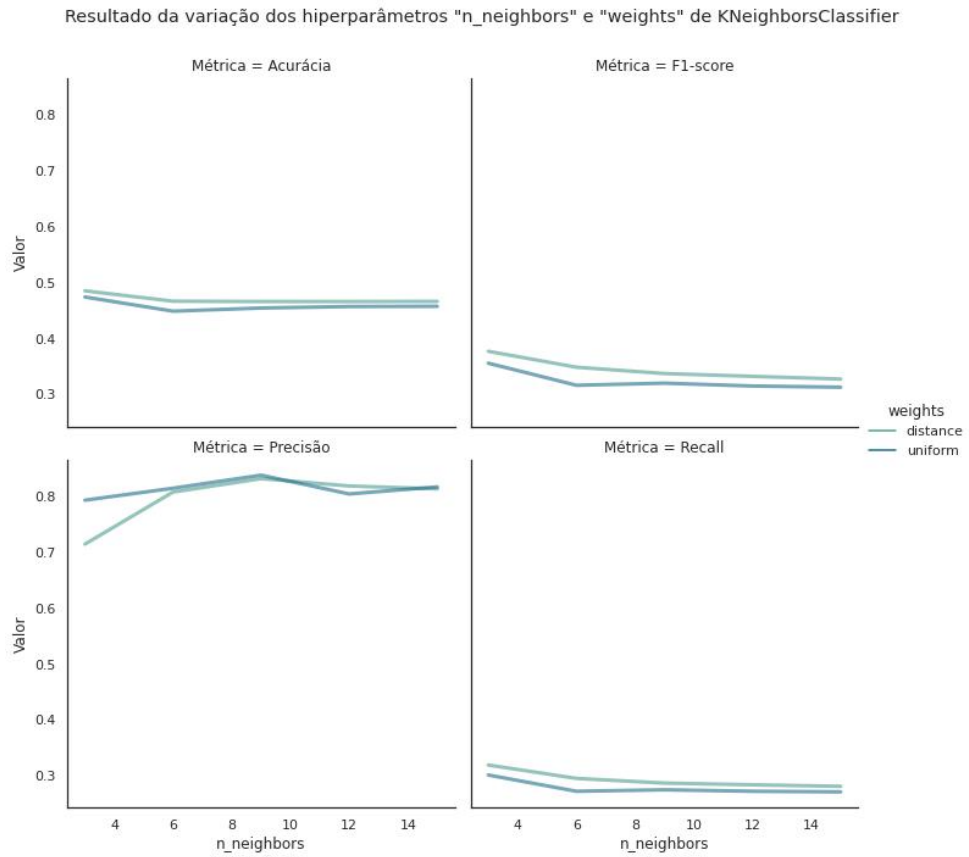
O modelo baseado no algoritmo *k-Nearest Neighbors* (k-NN), pode ser computacionalmente custoso para grandes conjuntos de dados, especialmente quando a dimensão da representação dos dados é alta. Além disso, o modelo pode ser sensível à escolha do valor de k e à métrica de distância utilizada, o que pode afetar significativamente a precisão das previsões.

Por conta disso, dois hiperparâmetros de *KNeighborsClassifier* são variados durante a busca em grade:

(i) $n\_neighbors \in [1, \infty)$ (número de vizinhos mais próximos a serem considerados) - Foram testados os valores candidatos 3, 6, 9, 12 e 15.

(ii) $weights$ (forma de ponderação dos votos) - Existem duas opções para este parâmetro $weights = uniform$, onde todas os vizinhos têm o mesmo peso na previsão da categoria de uma nova amostra e, $weights = distance$, onde os votos de vizinhos mais próximos têm mais influência do que vizinhos mais distantes.

Figura 21 – Métricas obtidas na variação dos hiperparâmetros *n_ neighbors* e *weights* de *KNeighborsClassifier*.

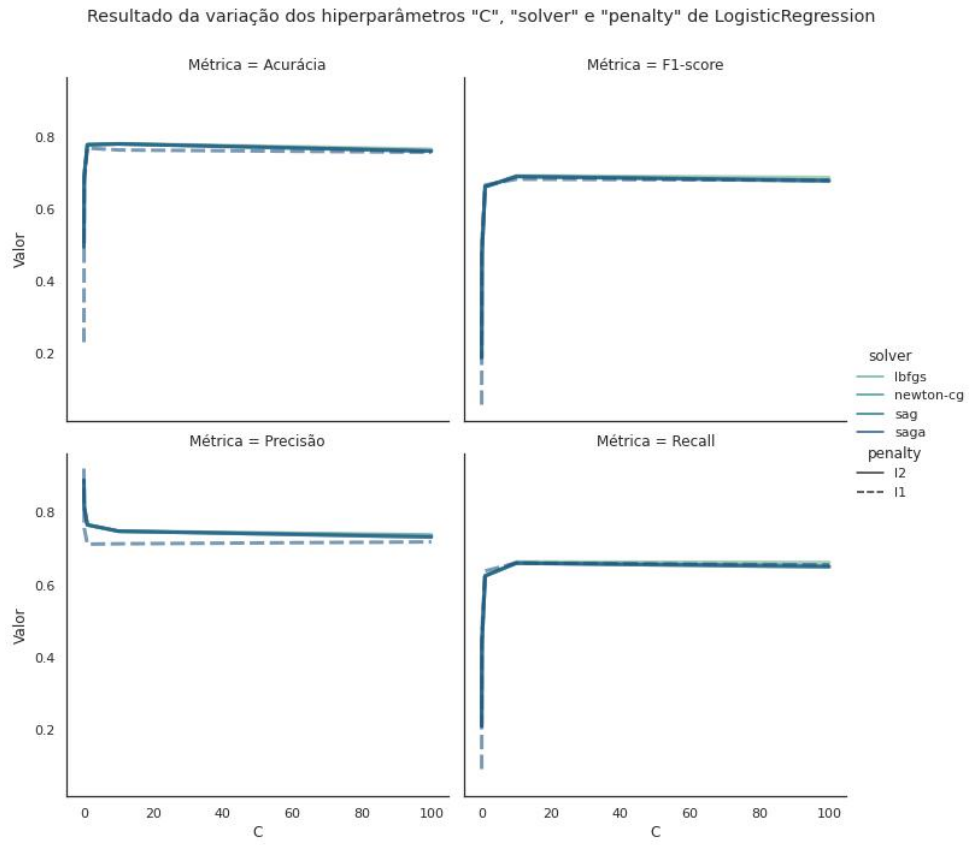


A Figura 21 mostra que apesar do *KNeighborsClassifier* demonstrar altíssima precisão, o que significa que está classificando a maioria dos exemplos positivos corretamente, demostrou valores apáticos para *recall*, ou seja, também está classificando de maneira equivocada exemplos negativos. Isso afetou diretamente o *F1-score* aferido. Como esse comportamento foi verificado independentemente dos hiperparâmetros, pode-se concluir que o *dataset* desbalanceado está afetando a eficácia do modelo, e/ou há muito ruído, isto é, não há fronteiras bem definidas entre as amostras de categorias diferentes, principalmente para categorias menores. O maior valor para *F1-score* na *grid search* foi 0.377 quando a combinação $n\_neighbors = 3$ e $weights = distance$ foi verificada. Para isso 10 combinações de valores candidatos para os hiperparâmetros foram testadas. Esses dados estão listados na Tabela 2.

4.6.2 Regressão Logística

Durante o treinamento, o *LogisticRegression* busca os coeficientes da função de custo que melhor se ajustam aos conjunto de treinamento. O objetivo é encontrar os valores dos pesos que minimizam a função de custo, que é definida como a diferença entre as previsões do modelo e os valores reais dos dados de treinamento.

Figura 22 – Métricas obtidas na variação dos hiperparâmetro *solver*, *penalty* e C de *LogisticRegression*.



Por isso, três hiperparâmetros foram escolhidos para observar as variações no desempenho do modelo e o impacto nas métricas:

(i) *solver* (tipo de algoritmo usado para otimizar a função de custo) - Os valores possíveis são $newton-cg$, $lbfgs$, $liblinear$, sag e $saga$. Porém, $liblinear$ não foi levado em consideração, pois é recomendado para *datasets* pequenos, o que não é o caso. $saga$ é uma versão melhorada do sag, que suporta a regularização com a norma $l1$. É adequado para grandes conjuntos de dados e é geralmente mais rápido e preciso do que o original;

(ii) *penalty* (tipo de penalidade a ser usada na função de custo) - Os valores possíveis são $l1$, $l2$, $elasticnet$ e $None$ ($None$ significa que nenhuma penalidade é aplicada). $elasticnet$ é uma combinação linear de $l1$ e $l2$ e somente pode ser combinado com $solver = saga$. Por conta disso, a opção com $penalty = elasticnet$ não foi testada. Com exceção de $solver = saga$, os demais algoritmos de otimização da função de custo aceitam somente $penalty \in \{l2, None\}$;

(iii) $C \in (0, \infty)$ (inverso da força de regularização) - Cinco valores candidatos do domínio foram testados, 1×10^{-2}, 1×10^{-1}, 1×10^{0}, 1×10^{1} e 1×10^{2}.

Os resultados podem ser observados na Figura 22. O hiperparâmetro *solver* é o que configura uma das técnicas de otimização da função de custo (citadas previamente na seção 3.1.2). C é chamado inverso da força de regularização. Quanto maior o valor de C, menor será a regularização. Por outro lado, um valor menor de C significa mais regularização, o que pode ajudar a evitar *overfitting*, mas pode resultar em um modelo menos preciso. A regularização aqui descrita acontece através das penalidades aplicando as normas $l1$ e $l2$ (também descritas previamente na seção 3.1.2).

Os valores candidatos desses hiperparâmetros geraram 45 combinações. Essa busca em grade encontrou o melhor valor de *F1-score* igual a 0.692 quando $solver = lbfgs$, $penalty = l2$ e $C = 10$. Esta configuração é listada na Tabela 2.

4.6.3 Naïve Bayes

O classificador Naïve Bayes utilizado, o *MultinomialNB*, possui o hiperparâmetro de suavização $\alpha \in (0, 1]$ (descrito previamente na seção 3.1.3). Portanto, neste trabalho, a busca em grade ocorreu com:

(i) α (parâmetro de suavização) - Foi variado no conjunto que divide o domínio em 11 pontos quase equidistantes, $\alpha \in \{1 \times 10^{-10}, 0.1, 0.2, \cdots, 1.0\}$.

Dessa forma, 11 rodadas de treinamento e teste foram executadas, uma para cada valor candidato de α. Para o *dataset* utilizado, este classificador mostrou excelente precisão, porém valores baixos de *recall*, como mostra a Figura 23. Isso indica que o modelo não está identificando os exemplos positivos e/ou está classificando erroneamente exemplos negativos como positivos. A melhor configuração (α próximo de zero) e o maior valor para *F1-score* (0.576) estão explicitados na Tabela 2.

4.6.4 Máquinas de Vetores de Suporte - SVM

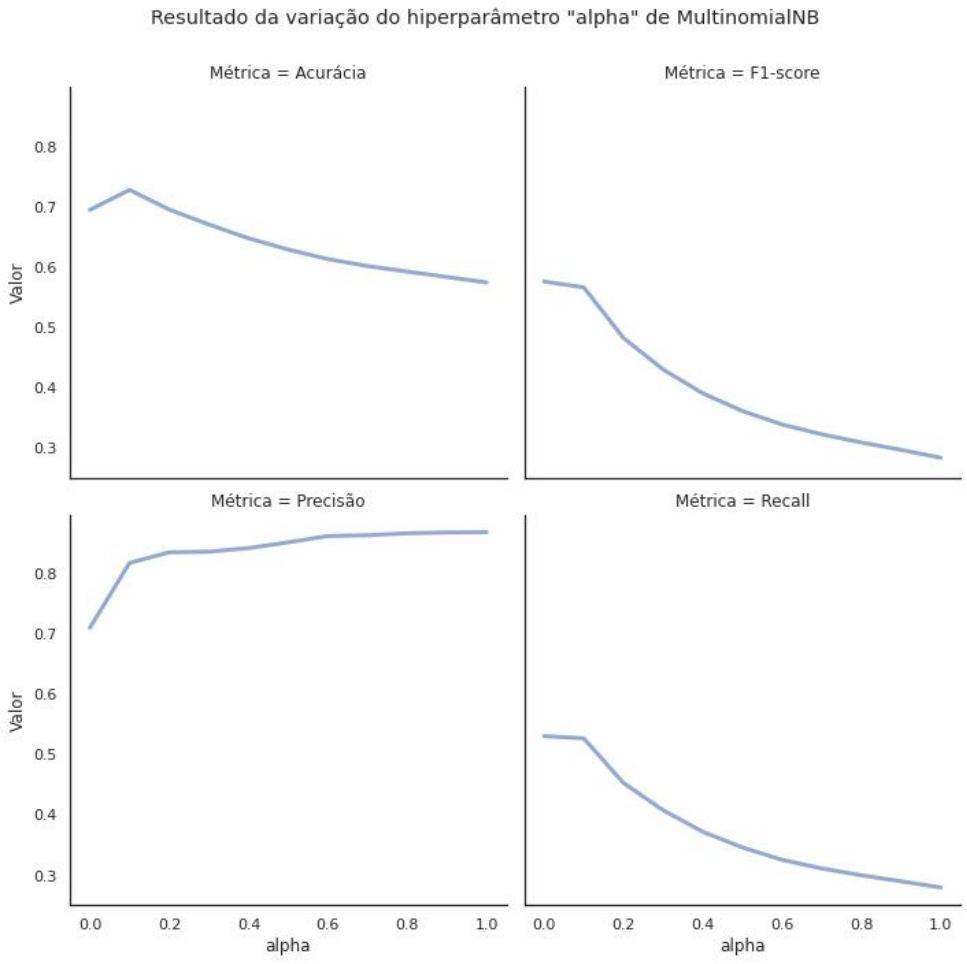
A implementação utilizada do classificador SVM, o *LinearSVC*, possui diversos hiperparâmetros. É implementado com o *kernel* linear e para problemas multiclasse, como deste trabalho, utiliza a estratégia *one-vs-rest* (descrita no capítulo 3). Os hiperparâmetros escolhidos para busca em grade foram:

(i) $C \in (0, \infty)$ (inverso da força de regularização) - Assim como no *LogisticRegression*, foi variado entre potências de 10 no intervalo $[1 \times 10^{-2}, 1 \times 10^{2}]$;

(ii) *dual* (seleciona o algoritmo para resolver o problema de otimização) - Por ser uma *flag*, dois valores são permitidos, $True$ e $False$;

(iii) *penalty* (tipo de penalidade a ser usada na função de custo) - Dois valores são possíveis nesta implementação, $l1$ e $l2$.

Figura 23 – Métricas obtidas na variação do hiperparâmetro α de *MultinomialNB*.

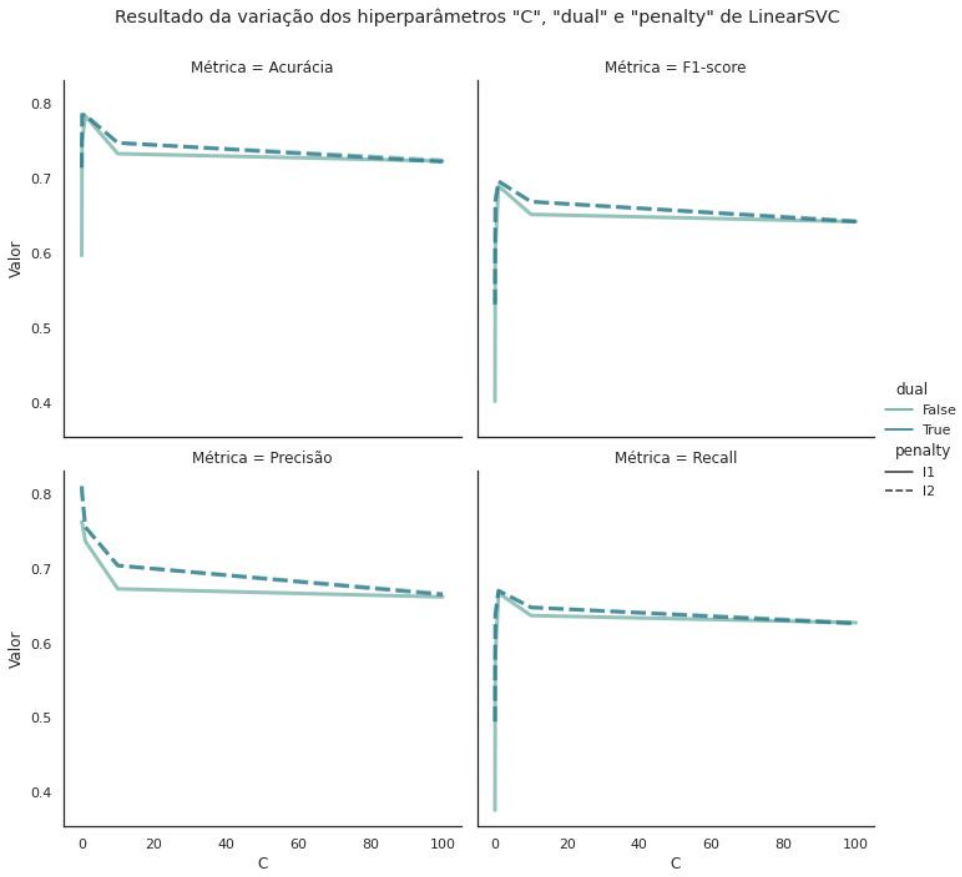


O hiperparâmetro *dual* precisou ser inserido na busca em grade pois o hiperparâmetro que seleciona a função de perda ($loss$), em sua configuração *default* ($loss = squared\_hinge$), é o único suportado com $penalty = l1$ e não suporta a configuração *default* de *dual* ($True$). Por conta disso, foi preciso utilizar $dual = False$ quando $penalty = l1$ foi avaliado. O hiperparâmetro *dual* seleciona o algoritmo de resolução do problema, se *dual* ou *primal*. A documentação do *LinearSVC*[5] orienta a preferir $dual = False$ quando o número de amostras for maior que o número de *tokens*. Os hiperparâmetros C e *penalty* são os mesmos que são apresentados na Regressão Logística (descritos na seção 3.1.2).

Com isso, 15 combinações de valores candidatos para hiperparâmetros foram testadas e o melhor resultado foi encontrado com $C = 1.0$, $penalty = l2$ e $dual = True$ (*default*). O resultado desses testes estão ilustrados na Figura 24 e a Tabela 2 também lista essa combinação de hiperparâmetros, e os associa com o maior valor de *F1-score* que foi obtido (0.696).

[5] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, acessado em 10/01/2023

Figura 24 – Métricas obtidas na variação dos hiperparâmetros C, *dual* e *penalty* de *LinearSVC*.



4.6.5 Floresta Aleatória

A ideia básica por trás do algoritmo de Floresta Aleatória é construir determinada quantidade de árvores de decisão independentes a partir do conjunto de dados de treinamento. Para fazer uma previsão, o modelo *RandomForestClassifier* combina as previsões de todas as árvores na floresta, utilizando um esquema de votação por maioria. A classificação final é dada pela classe com mais votos.

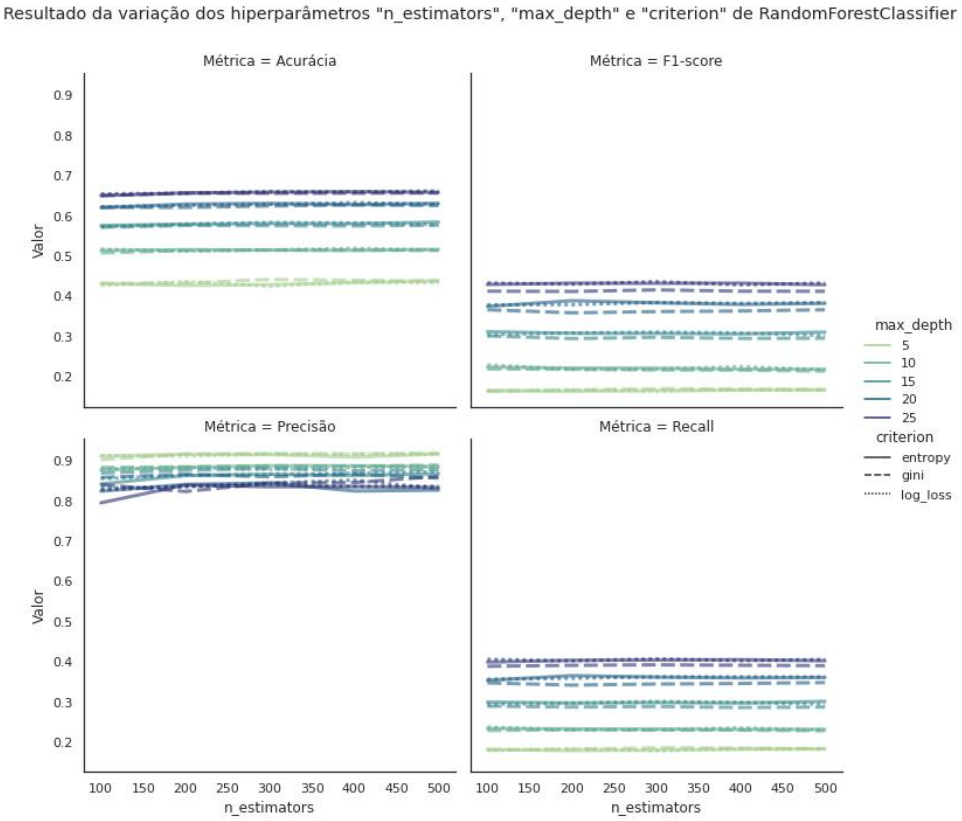
Logo, podemos concluir que dentre os hiperparâmetros de *RandomForestClassifier*, três são de extrema relevância:

(i) $n\_estimators \in [1, \infty)$ (controla o número de árvores na floresta aleatória) - Os valores testados na busca em grade foram 100, 200, 300, 400 e 500;

(ii) $max\_depth \in [1, \infty)$ (controla a profundidade máxima das árvores na floresta aleatória) - Foram candidatos os valores 5, 10, 15, 20, 25;

(iii) *criterion* (controla o critério utilizado para dividir os nós durante a construção das árvores na floresta aleatória) - configura a função de medida de qualidade (que foi brevemente citada na seção 3.1.5), e determina a qualidade da divisão de um nó em subnós. Três valores são aceitos: $criterion = gini$ é uma medida de impureza

Figura 25 – Métricas obtidas na variação dos hiperparâmetros *n_estimators*, *max_depth* e *criterion* de *RandomForestClassifier*.



que mede a probabilidade de um elemento escolhido aleatoriamente em um nó ser classificado incorretamente, com base na distribuição das classes nos subnós. Já *criterion* = *entropy* e *criterion* = *log_loss* (também conhecidos como ganho de informação de Shannon) calcula a redução na entropia (ou incerteza) dos dados após a divisão de um nó. Esses três valores possíveis foram testados.

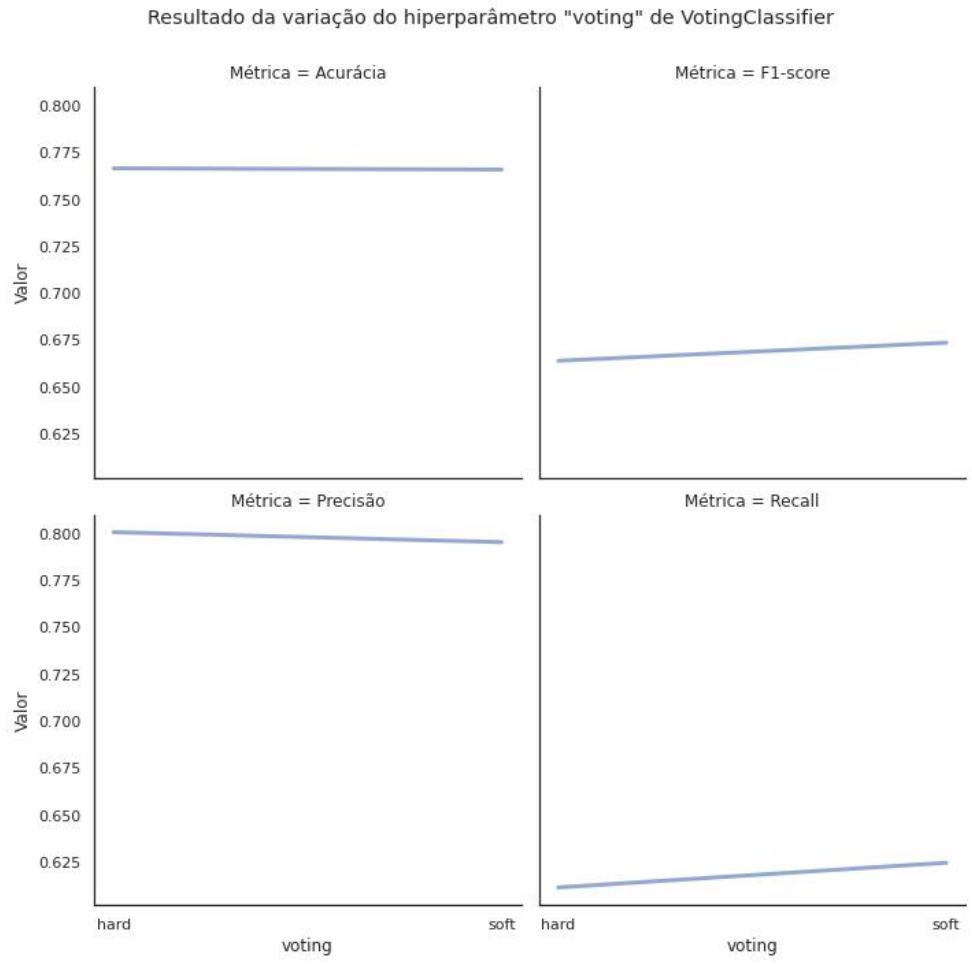
Para encontrar a melhor combinação desses hiperparâmetros para o *dataset* do IFE, 75 rodadas de treinamento foram executadas e o maior valor de *F1-score* obtido foi 0.437. Isso quando $n\_estimators = 300$, $max\_depth = 25$ e $criterion = log\_loss$. Essa combinação de hiperparâmetros e esse melhor valor de *F1-score* também estão apresentados na Tabela 2.

4.6.6 Comitê de Classificadores

O *VotingClassifier* é um modelo de aprendizado de máquina *ensembling* disponível na biblioteca *Scikit-learn*. Um único hiperparâmetro desse método de classificação foi observado na busca em grade:

(i) *voting* (forma de ponderação dos votos) - Quando *voting* = *hard*, ele usa as previsões de categorias de cada modelo, por maioria simples. Porém, quando

Figura 26 – Métricas obtidas na variação do hiperparâmetro *voting* de *VotingClassifier*.



$voting = soft$, ele usa as probabilidades de categorias previstas por cada modelo.

A Figura 26 mostra a comparação das métrica durante o processo de busca em grade e $voting = soft$ se saiu melhor alcançando *F1-score* igual a 0.674. A Tabela 2 também lista essa configuração.

Podemos perceber, visualizando os resultados sumarizados na Tabela 2, que já nesse conjunto de resumos utilizados para a busca de valores dos hiperparâmetros, *LinearSVC* se saiu melhor, seguido de perto por *LogisticRegression*, cujo resultado se diferenciou na terceira casa decimal. Ainda próximo a eles se pôs o *VotingClassifier*. Por fim, os outros três modelos, *MultinomialNB*, *RandomForestClassifier* e *KNeighborsClassifier*, nesta ordem. De fato, esse procedimento não traz nenhuma conclusão quanto à definição da melhor abordagem para a classificação, mas dá uma prévia e traça uma referência do que se esperar dos modelos.

Tabela 2 – Melhores hiperparâmetros encontrados na busca em grade.

CLASSIFICADOR	#COMBINAÇÕES	HIPERPARÂMETRO	MELHOR CONFIGURAÇÃO	F1-SCORE
KNeighborsClassifier	10	*n_neighbors*	3	0.377
		weights	distance	
LogisticRegression	45	solver	lbfgs	0.692
		penalty	l2	
		C	10	
MultinomialNB	11	α	1×10^{-10}	0.576
LinearSVC	15	C	1.0	0.696
		penalty	l2	
		dual	True	
RandomForestClassifier	75	criterion	log_loss	0.437
		max_depth	25	
		n_estimators	300	
VotingClassifier	2	voting	soft	0.674

4.7 TREINAMENTO E AVALIAÇÃO DOS RESULTADOS EXPERIMENTAIS

Nessa etapa, instâncias não treinadas dos seis classificadores foram configuradas com os hiperparâmetros que apresentaram melhores resultados na busca em grade, dentre os que foram testados. Essas instâncias receberam dados antes nunca aplicados: o conjunto de treinamento, com 62.448 resumos de notícias (representando 80% dos 78.061 resumos que foram separados do conjunto de ajuste de hiperparâmetros).

Para garantir a imparcialidade na seleção dos dados, foi utilizada a técnica de validação *K-fold* com $K = 5$, que divide o conjunto de treinamento em cinco partes iguais. É considerada uma boa prática para evitar vieses de seleção de dados e para garantir que o modelo seja capaz de generalizar para novos dados.

O resultado da validação cruzada, executada com o método *cross_validate* do *Scikit-learn*, forneceu as métricas e os classificadores ajustados para as cinco rodadas de treinamento. Entre os cinco classificadores ajustados, o que obteve o melhor *F1-score* foi selecionado para prever as categorias do conjunto de teste, que consistia de 15.613 resumos de notícias ainda não utilizados nesta pesquisa. A Figura 20 ilustra como o processo descrito nesta seção foi realizado.

É importante ressaltar que o mesmo critério adotado na seção anterior foi mantido. A performance dos modelos foi avaliada com base na macro-média do *F1-score*, mas também foram observadas a acurácia, e as macro-médias da precisão e do *recall*. A Figura 27 ilustra o alcance entre as funções de classificação utilizada pelos métodos ($\hat{\phi}$) em relação à função de classificação real dos documentos (ϕ).

A Tabela 3 lista detalhadamente os valores de *F1-score* que cada classificador alcançou. Observamos que o *LinearSVC* e o *LogisticRegression* alcançaram valores próximos, 0.74 e 0.73 respectivamente, seguidos de perto pelo *VotingClassifier*, com 0.72. Vale destacar que o *LinearSVC* e a *LogisticRegression* também foram os menos afetados pela desbalanceamento do conjunto de dados, apresentando valores próximos para precisão e *recall*. Além disso, eles são mais rápidos e consomem menos recursos computacionais do que o

Figura 27 – Resultado do teste dos classificadores.

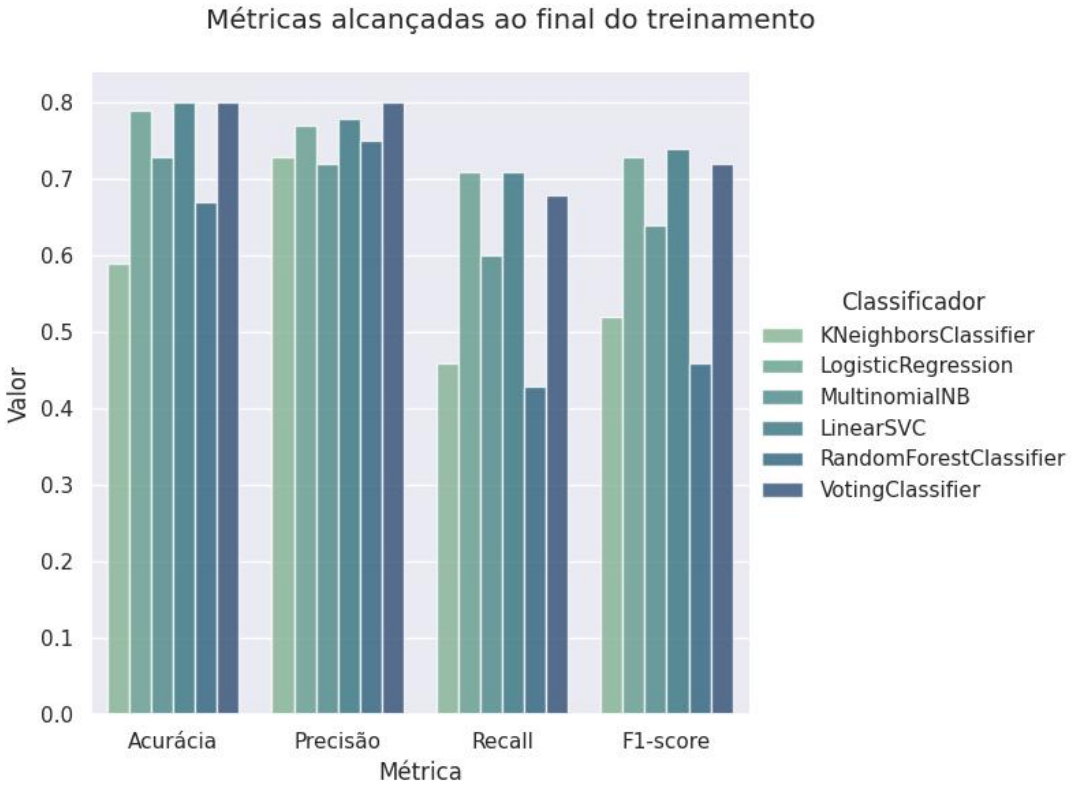


Tabela 3 – *F1-score* obtido no processo de teste dos classificadores.

CLASSIFICADOR	F1-SCORE
KNeighborsClassifier	0.52
LogisticRegression	0.73
MultinomialNB	0.64
LinearSVC	0.74
RandomForestClassifier	0.46
VotingClassifier	0.72

VotingClassifier, o que pode minimizar o resultado quase semelhante obtido por ele.

Logo em seguida, *MultinomialNB* alcançou a marca de 0.64. Antagônicos aos classificadores que obtiveram melhores resultados, *KNeighborsClassifier* e *RandomForestClassifier* foram afetados pelo desbalanceamento das categorias e apresentaram *F1-score* de 0.52 e 0.46, respectivamente. Como é possível observar na Figura 27, o resultado de *F1-score* para eles foi afetado por valores ruins de *recall*, apesar de terem obtido boa precisão. Os resultados detalhados por classe, obtidos por cada um dos classificadores, podem ser visualizados no Apêndice B.

Os resultados obtidos na avaliação indicam que os modelos *LinearSVC* e *LogisticRegression* apresentaram desempenhos superiores aos demais, o que sugere que modelos lineares podem ser uma escolha mais adequada para pesquisas futuras com esse *dataset* e podem ser úteis para a classificação de textos em português. Além disso, o modelo

VotingClassifier, que combina diferentes algoritmos de classificação, por fim, se beneficiou das técnicas lineares.

É importante lembrar que a implementação dos algoritmos de classificação foi conduzida no ambiente colaborativo *Google Colaboratory*. Devido à natureza da alocação de recursos sob demanda, os tempos de treinamento dos classificadores podem variar substancialmente. No entanto, ainda é possível obter uma estimativa aproximada do tempo de execução para cada um dos modelos empregados. Os tempos médios de treinamento (por dobra, em segundos) na etapa de busca em grade de hiperparâmetros e as respectivas métricas alcançadas pelo classificador para cada combinação de hiperparâmetros estão no Apêndice C, assim como a Tabela 5 com os tempos médios (por dobra, em segundos) de treinamento dos classificadores já configurados.

5 CONCLUSÃO E TRABALHOS FUTUROS

Foi realizada uma pesquisa cujo resultado é um modelo de aprendizado de máquina capaz de classificar novos resumos de notícias no contexto do setor elétrico em diferentes categorias, levando em consideração o conteúdo textual. Para isso, foi lançado mão de seis implementações de algoritmos de aprendizado de máquina e avaliamos seus desempenhos na classificação desses resumos de notícias.

A partir dos experimentos realizados, foi possível observar que as abordagens baseadas nos algoritmos de Máquinas de Vetores de Suporte e Regressão Logística, implementadas pelos métodos *LinearSVC* e *LogisticRegression*, respectivamente, apresentaram os melhores resultados em relação às outras técnicas avaliadas. Com a técnica proposta, as melhores abordagens foram capaz de categorizar de forma satisfatória resumos de notícias do setor elétrico em 14 categorias.

No entanto, é importante destacar que o presente estudo apresenta algumas limitações, como o fato de estar restrito a um domínio específico. Apesar disso, os resultados obtidos evidenciam a relevância e eficácia de cada técnica na tarefa de classificação automática de textos. Esses resultados destacam a importância de se avaliar diferentes técnicas e algoritmos para a tarefa de classificação automática de textos e aprimorar a escolha do modelo mais adequado para cada caso específico.

Dispor de um extenso *dataset* em língua portuguesa é uma contribuição significativa para a comunidade de pesquisa, uma vez que grande parte dos estudos nesta área emprega dados em inglês. Ao utilizar um conjunto de dados em português, é possível desenvolver e aprimorar modelos de classificação de texto que melhor atendam às necessidades desse idioma, incluindo sua sintaxe e estrutura gramatical e léxica. Esses dados podem fomentar o progresso da pesquisa em processamento de linguagem natural em língua portuguesa, bem como contribuir para o desenvolvimento e aprimoramento de algoritmos de classificação de texto específicos, gerando aplicações mais eficazes em diversas áreas.

Existem diversas oportunidades de ampliação e reanálise desta pesquisa em vários contextos distintos, por exemplo:

- Quanto aos algoritmos de classificação: os resultados desta pesquisa podem estimular uma busca mais minuciosa por combinações de hiperparâmetros mais efetivas. Além disso, outras técnicas de classificação de texto podem ser exploradas nesse mesmo contexto, e é possível que soluções mais modernas, como redes neurais, obtenham resultados ainda mais satisfatórios. Ademais, os resultados obtidos nesta pesquisa apontam que os classificadores lineares são uma abordagem promissora nesse cenário. Seria interessante aprofundar o estudo das técnicas de regularização, como a $l1$ e a $l2$, para averiguar se elas podem contribuir ainda mais para o de-

sempenho dos classificadores lineares. Outra possibilidade seria a utilização mais abrangente de técnicas de *ensemble*, para combinar os resultados de diferentes classificadores lineares e gerar um modelo mais robusto. Tais investigações poderiam aprimorar a compreensão da eficácia dos classificadores lineares na classificação de textos;

- Quanto à seleção de *features*: uma alternativa ao teste qui-quadrado que poderia ser explorada seria a análise de correlação, que pode identificar atributos fortemente correlacionados com a variável de saída. Outra opção seria a utilização da importância das *features* em modelos de aprendizado de máquina ou a análise de componentes principais para identificar as combinações lineares de atributos mais importantes. Além disso, a seleção de *features* baseada em modelos e em redes neurais também poderia ser investigada como possível alternativa para o teste qui-quadrado;

- Quanto à extração de *features*: embora o modelo TF-IDF seja uma técnica amplamente utilizada para a representação de características em problemas de classificação de texto, existem outras alternativas que podem ser exploradas em futuros trabalhos. Uma opção seria o uso de modelos baseados em *word embeddings* (INCITTI; URLI; SNIDARO, 2023), como *Word2Vec* ou *GloVe*, que têm mostrado resultados promissores em tarefas de classificação de texto. Além disso, também poderia ser investigada a utilização de redes neurais, que podem aprender representações de características mais complexas a partir dos dados brutos;

- Quanto ao pré-processamento: uma opção seria o uso de outras bibliotecas de *stemmers*, com abordagens diferentes para realizar a redução de palavras a uma forma comum. Além disso, também poderia ser explorado o uso de técnicas de lematização, que podem produzir resultados mais precisos na redução de palavras a suas formas básicas, levando em conta o contexto em que elas aparecem. Por fim, técnicas de tokenização baseadas em modelos de linguagem, como os modelos de transformadores, podem ser utilizadas para gerar *tokens* mais informativos, considerando o contexto em que as palavras aparecem;

- Quanto ao desbalanceamento do *dataset*: um dos pontos críticos deste projeto, pois impactou o desempenho de alguns classificadores devido ao fato de que algumas classes eram recentes e apresentavam poucas amostras. Existem diversas soluções implementadas para lidar com esse problema, incluindo técnicas de sobreamostragem (*oversampling*) e subamostragem (*undersampling*), cada uma com suas vantagens e desvantagens a serem analisadas;

- Quanto à paralelização: uma possibilidade interessante para trabalhos futuros seria explorar técnicas de paralelização para acelerar o processo de pré-processamento e

classificação de grandes volumes de texto. Uma abordagem que poderia ser adotada seria a utilização de *pipelines*, que permitem dividir o fluxo de trabalho em tarefas independentes que podem ser executadas em paralelo, reduzindo o tempo de processamento. Essas técnicas poderiam ser aplicadas tanto na fase de pré-processamento, como na limpeza, normalização e redução de dimensionalidade, quanto na fase de classificação propriamente dita, como no treinamento de modelos de aprendizado de máquina.

- Quanto a expansão do trabalho: um *script* de *Web Scraping* pode ser desenvolvido para buscar novas notícias do setor elétrico ou outras notícias em português de maneira geral. Pode-se ainda, analisar diversas *newsletters* automaticamente com o objetivo de encontrar notícias atuais e relevantes. Em seguida, técnicas de sumarização de texto (EL-KASSAS et al., 2021) podem ser empregadas para resumir automaticamente novas notícias. Por fim, uma plataforma pode ser desenvolvida para agrupar todas essas implementações e auxiliar na geração semi-automática de um informativo.

REFERÊNCIAS

ALAMERI, S. A.; MOHD, M. Comparison of fake news detection using machine learning and deep learning techniques. In: IEEE. **2021 3rd International Cyber Resilience Conference (CRC)**. [S.l.], 2021. p. 1–6.

ALLAHYARI, M. et al. A brief survey of text mining: Classification, clustering and extraction techniques. **ArXiv**, abs/1707.02919, 2017.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca**. 2. ed. [S.l.]: Bookman, 2013.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001.

CASTRO, L. N. de; FERRARI, D. G. **Introdução à mineração de dados**. [S.l.]: Saraiva Educação SA, 2017.

CHAULAGAIN, R. S. et al. Cloud based web scraping for big data applications. **IEEE International Conference on Smart Cloud**, p. 138–143, 2017.

CHEN, Y.-T.; CHEN, M. C. Using chi-square statistics to measure similarities for text categorization. **Expert systems with applications**, Elsevier, v. 38, n. 4, p. 3085–3090, 2011.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, p. 273–297, 1995.

CUTLER, A.; CUTLER, D. R.; STEVENS, J. R. Random forests. In: **Ensemble machine learning**. [S.l.]: Springer, 2012. p. 157–175.

EL-KASSAS, W. S. et al. Automatic text summarization: A comprehensive survey. **Expert Systems with Applications**, v. 165, p. 113679, 2021. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417420305030.

FARLEY, E.; PIEROTTE, L. Web scraping: An emerging data collection method for criminal justice researchers. Justice Research and Statistics Association, 2017.

HAN, E.-H. S.; KARYPIS, G.; KUMAR, V. Text categorization using weight adjusted k-nearest neighbor classification. In: SPRINGER. **Pacific-asia conference on knowledge discovery and data mining**. [S.l.], 2001. p. 53–65.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3. ed. Massachusetts: Bookman, 2012.

HASTIE, T. et al. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. v. 2.

INCITTI, F.; URLI, F.; SNIDARO, L. Beyond word embeddings: A survey. **Information Fusion**, v. 89, p. 418–436, 2023. ISSN 1566-2535. Disponível em: https://www.sciencedirect.com/science/article/pii/S1566253522001233.

KHAN, A. et al. A review of machine learning algorithms for text-documents classification. **JOURNAL OF ADVANCES IN INFORMATION TECHNOLOGY**, v. 1, n. 1, p. 04–20, 2010.

KLEINBAUM, D. G. et al. **Logistic regression**. [S.l.]: Springer, 2002.

KORD, V.; MAHENDER, C. N. Text classification and classifiers: A survey. **International Journal of Artificial Intelligence Applications (IJAIA)**, v. 3, n. 2, p. 85–99, 2012.

KOWSARI, K. et al. Text classification algorithms: A survey. **Information**, v. 10, n. 4, 2019. ISSN 2078-2489. Disponível em: https://www.mdpi.com/2078-2489/10/4/150.

KUMAR, S. et al. Categorizing text documents using naïve bayes, svm and logistic regression. In: SPRINGER. **Data Management, Analytics and Innovation: Proceedings of ICDMAI 2020, Volume 2**. [S.l.], 2021. p. 225–235.

MARON, M. E. Automatic indexing: An experimental inquiry. **J. ACM**, v. 8, p. 404–417, 1961.

MCCALLUM, A. K. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. **http://www.cs.cmu.edu/mccallum/bow/**, 1996.

MUNZERT, S. et al. **Automated Data Collection with R - A Practical Guide to Web Scraping and Text Mining**. 1. ed. West Sussex, United Kingdom: John Wiley Sons, Ltd, 2015.

PATTON, R. M.; POTOK, T. E. Identifying event impacts by monitoring the news media. In: **2008 12th International Conference Information Visualisation**. [S.l.: s.n.], 2008. p. 333–337.

PERSSON, E. **Evaluating tools and techniques for web scraping**. 2019.

RENNIE, J. D. et al. Tackling the poor assumptions of naive bayes text classifiers. In: **Proceedings of the 20th international conference on machine learning (ICML-03)**. [S.l.: s.n.], 2003. p. 616–623.

RIFKIN, R.; KLAUTAU, A. In defense of one-vs-all classification. **The Journal of Machine Learning Research**, JMLR. org, v. 5, p. 101–141, 2004.

SAURKAR, A. V.; PATHARE, K. G.; GODE, S. A. An overview on web scraping techniques and tools. **International Journal on Future Revolution in Computer Science Communication Engineering**, v. 4, n. 4, p. 363 – 367, 2018.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM Computing Surveys**, v. 34, n. 1, p. 1–47, 2002.

SHAH, F. P.; PATEL, V. A review on feature selection and feature extraction for text classification. In: **2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)**. [S.l.: s.n.], 2016. p. 2264–2268.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: From theory to algorithms**. [S.l.]: Cambridge university press, 2014.

SHEYKHMOUSA, M. et al. Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 13, p. 6308–6325, 2020.

SIRISURIYA, S. C. M. d. S. A comparative study on web scraping. **Proceedings of 8th International Research Conference, KDU**, p. 135–140, 2015.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information Processing Management**, v. 45, n. 4, p. 427–437, 2009. ISSN 0306-4573. Disponível em: https://www.sciencedirect.com/science/article/pii/S0306457309000259.

SUMATHY, K.; CHIDAMBARAM, M. Text mining: Concepts, applications, tools and issues – an overview. **International Journal of Computer Applications**, v. 80, n. 4, p. 29–32, 2013.

YADAV, S.; SHUKLA, S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: **2016 IEEE 6th International Conference on Advanced Computing (IACC)**. [S.l.: s.n.], 2016. p. 78–83.

YUAN, G.-X. et al. A comparison of optimization methods and software for large-scale l1-regularized linear classification. **The Journal of Machine Learning Research**, JMLR. org, v. 11, p. 3183–3234, 2010.

ZHANG, M.-L.; ZHOU, Z.-H. A review on multi-label learning algorithms. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 8, p. 1819–1837, 2014.

ZHAO, B. Web scraping. **Encyclopedia of Big Data**, 2017.

ZHU, X. Cs838-1 advanced nlp?: Text categorization with logistic regression. **no**, v. 3, p. 1–3, 2007.

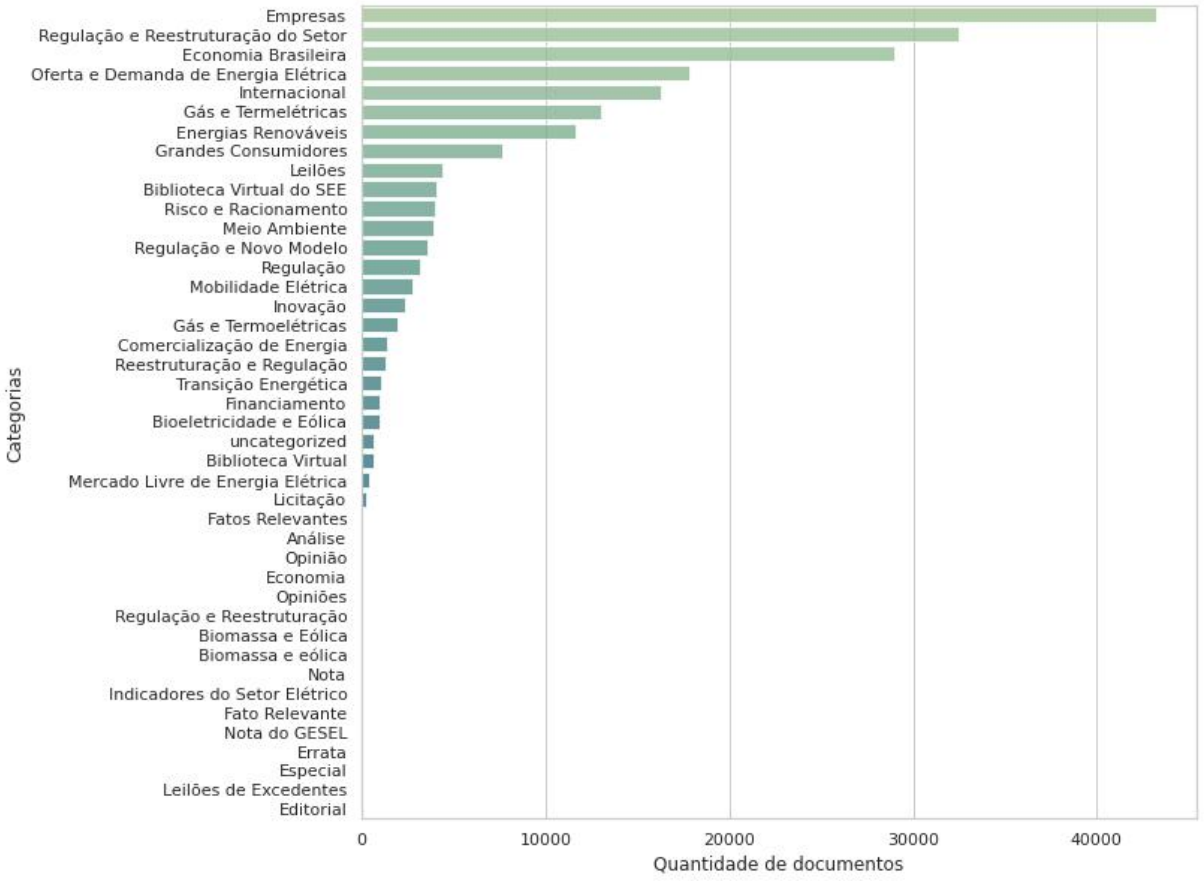
APÊNDICES

APÊNDICE A – RESULTADOS DA EXECUÇÃO DO WEB SCRAPER

Tabela 4 – URLs duplicadas no *dataset*.

EDIÇÃO	URL	#RESUMOS
796	gesel.ie.ufrj.br/app/webroot/files/IFES/IFE796.html	49
642	gesel.ie.ufrj.br/app/webroot/files/IFES/IFE642.html	41
1.484	gesel.ie.ufrj.br/app/webroot/files/IFES/IFE1484.html	23
3.424	www.provedor.nuca.ie.ufrj.br/eletrobras/arquivos/ifes/IFE3424.html	43
3.809	www.provedor.nuca.ie.ufrj.br/eletrobras/arquivos/ifes/IFE3809.html	34
3.816	www.provedor.nuca.ie.ufrj.br/eletrobras/arquivos/ifes/IFE3816.html	30
4.949	www.gesel.ie.ufrj.br/app/webroot/files/publications/01_IFE4949.html	31

Figura 28 – Distribuição dos documentos por categorias - *dataset* sem pré-processamento.



APÊNDICE B – RESULTADOS DA PREDIÇÃO DOS CLASSIFICADORES AJUSTADOS

Figura 29 – Resultado da predição de *KNeighborsClassifier*.

---------- KNeighborsClassifier TEST RESULT ----------

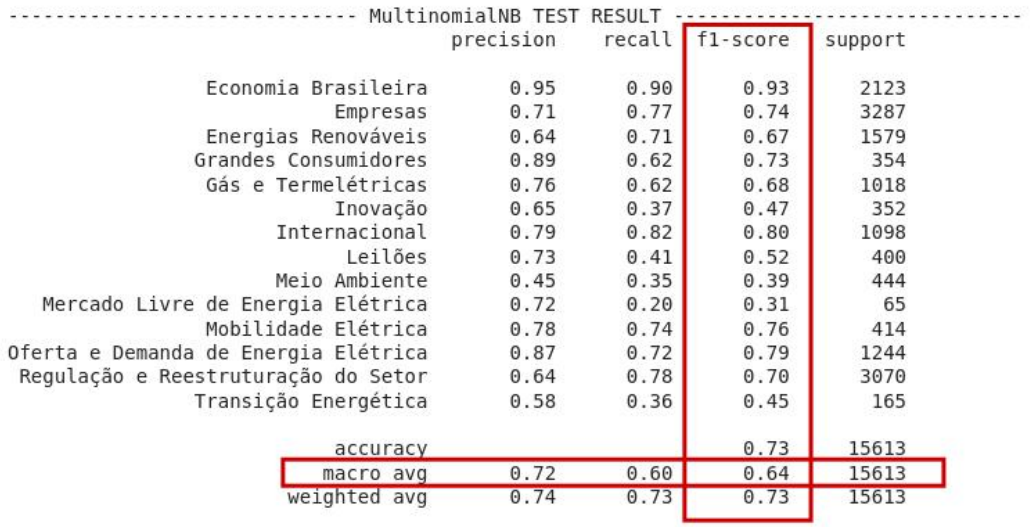
	precision	recall	f1-score	support
Economia Brasileira	0.97	0.79	0.88	2123
Empresas	0.59	0.74	0.65	3287
Energias Renováveis	0.25	0.77	0.37	1579
Grandes Consumidores	0.91	0.50	0.65	354
Gás e Termelétricas	0.82	0.49	0.61	1018
Inovação	0.80	0.23	0.36	352
Internacional	0.87	0.32	0.47	1098
Leilões	0.76	0.42	0.55	400
Meio Ambiente	0.61	0.18	0.28	444
Mercado Livre de Energia Elétrica	0.61	0.26	0.37	65
Mobilidade Elétrica	0.83	0.36	0.51	414
Oferta e Demanda de Energia Elétrica	0.87	0.69	0.77	1244
Regulação e Reestruturação do Setor	0.74	0.46	0.57	3070
Transição Energética	0.54	0.16	0.24	165
accuracy			0.59	15613
macro avg	0.73	0.46	0.52	15613
weighted avg	0.72	0.59	0.60	15613

Figura 30 – Resultado da predição de *LogisticRegression*.

---------- LogisticRegression TEST RESULT ----------

	precision	recall	f1-score	support
Economia Brasileira	0.96	0.97	0.97	2123
Empresas	0.77	0.80	0.79	3287
Energias Renováveis	0.70	0.73	0.71	1579
Grandes Consumidores	0.83	0.80	0.81	354
Gás e Termelétricas	0.79	0.77	0.78	1018
Inovação	0.72	0.53	0.61	352
Internacional	0.86	0.86	0.86	1098
Leilões	0.74	0.66	0.70	400
Meio Ambiente	0.61	0.53	0.57	444
Mercado Livre de Energia Elétrica	0.61	0.34	0.44	65
Mobilidade Elétrica	0.85	0.86	0.86	414
Oferta e Demanda de Energia Elétrica	0.83	0.79	0.81	1244
Regulação e Reestruturação do Setor	0.74	0.77	0.75	3070
Transição Energética	0.71	0.53	0.61	165
accuracy			0.79	15613
macro avg	0.77	0.71	0.73	15613
weighted avg	0.79	0.79	0.79	15613

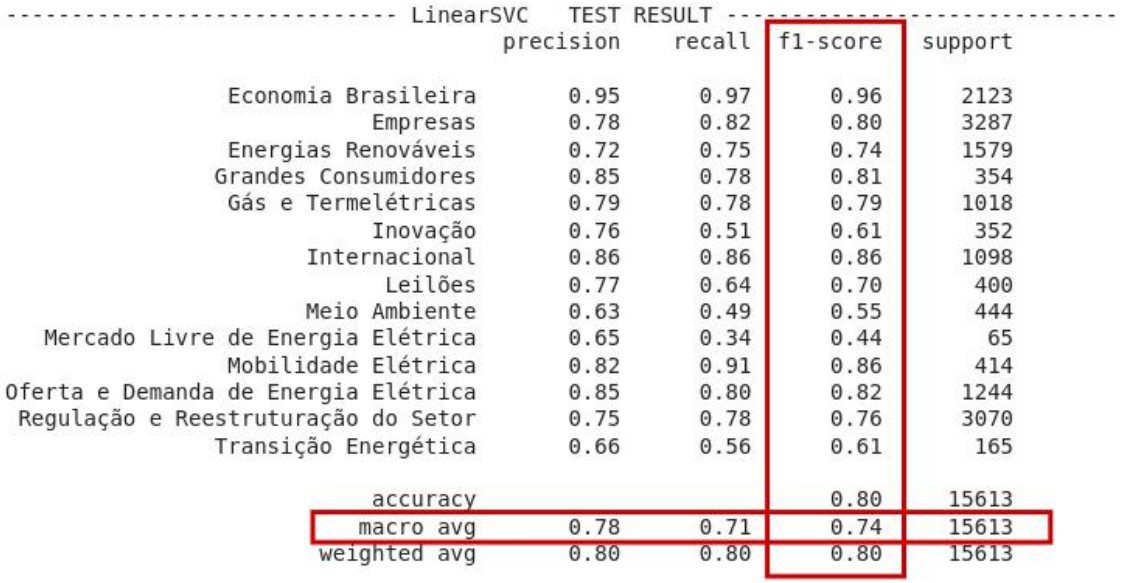
Figura 31 – Resultado da predição de *MultinomialNB*.



---------- MultinomialNB TEST RESULT ----------

	precision	recall	f1-score	support
Economia Brasileira	0.95	0.90	0.93	2123
Empresas	0.71	0.77	0.74	3287
Energias Renováveis	0.64	0.71	0.67	1579
Grandes Consumidores	0.89	0.62	0.73	354
Gás e Termelétricas	0.76	0.62	0.68	1018
Inovação	0.65	0.37	0.47	352
Internacional	0.79	0.82	0.80	1098
Leilões	0.73	0.41	0.52	400
Meio Ambiente	0.45	0.35	0.39	444
Mercado Livre de Energia Elétrica	0.72	0.20	0.31	65
Mobilidade Elétrica	0.78	0.74	0.76	414
Oferta e Demanda de Energia Elétrica	0.87	0.72	0.79	1244
Regulação e Reestruturação do Setor	0.64	0.78	0.70	3070
Transição Energética	0.58	0.36	0.45	165
accuracy			0.73	15613
macro avg	0.72	0.60	0.64	15613
weighted avg	0.74	0.73	0.73	15613

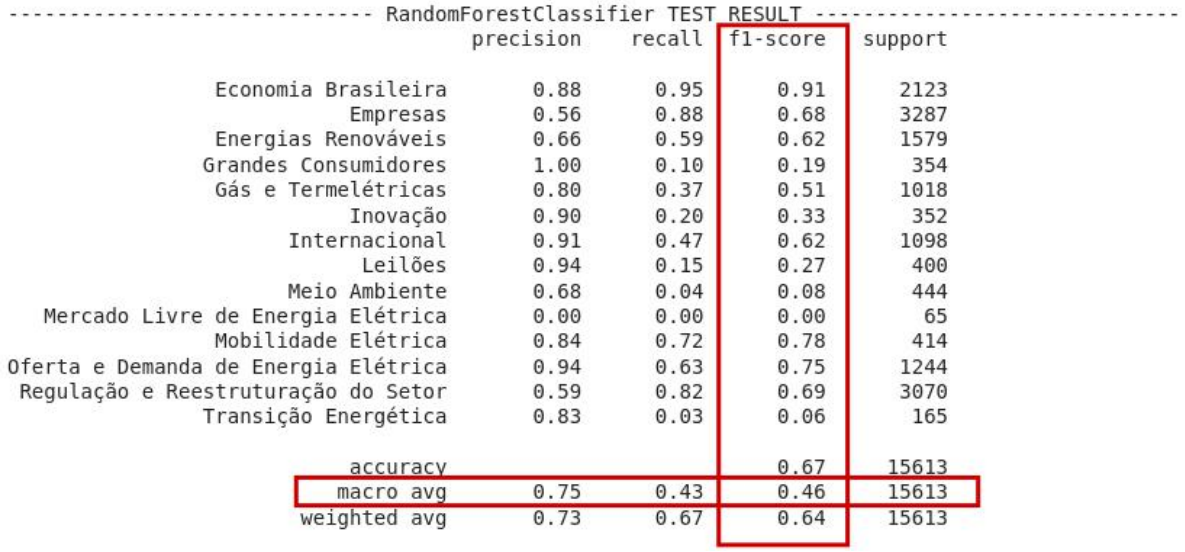
Figura 32 – Resultado da predição de *LinearSVC*.



---------- LinearSVC TEST RESULT ----------

	precision	recall	f1-score	support
Economia Brasileira	0.95	0.97	0.96	2123
Empresas	0.78	0.82	0.80	3287
Energias Renováveis	0.72	0.75	0.74	1579
Grandes Consumidores	0.85	0.78	0.81	354
Gás e Termelétricas	0.79	0.78	0.79	1018
Inovação	0.76	0.51	0.61	352
Internacional	0.86	0.86	0.86	1098
Leilões	0.77	0.64	0.70	400
Meio Ambiente	0.63	0.49	0.55	444
Mercado Livre de Energia Elétrica	0.65	0.34	0.44	65
Mobilidade Elétrica	0.82	0.91	0.86	414
Oferta e Demanda de Energia Elétrica	0.85	0.80	0.82	1244
Regulação e Reestruturação do Setor	0.75	0.78	0.76	3070
Transição Energética	0.66	0.56	0.61	165
accuracy			0.80	15613
macro avg	0.78	0.71	0.74	15613
weighted avg	0.80	0.80	0.80	15613

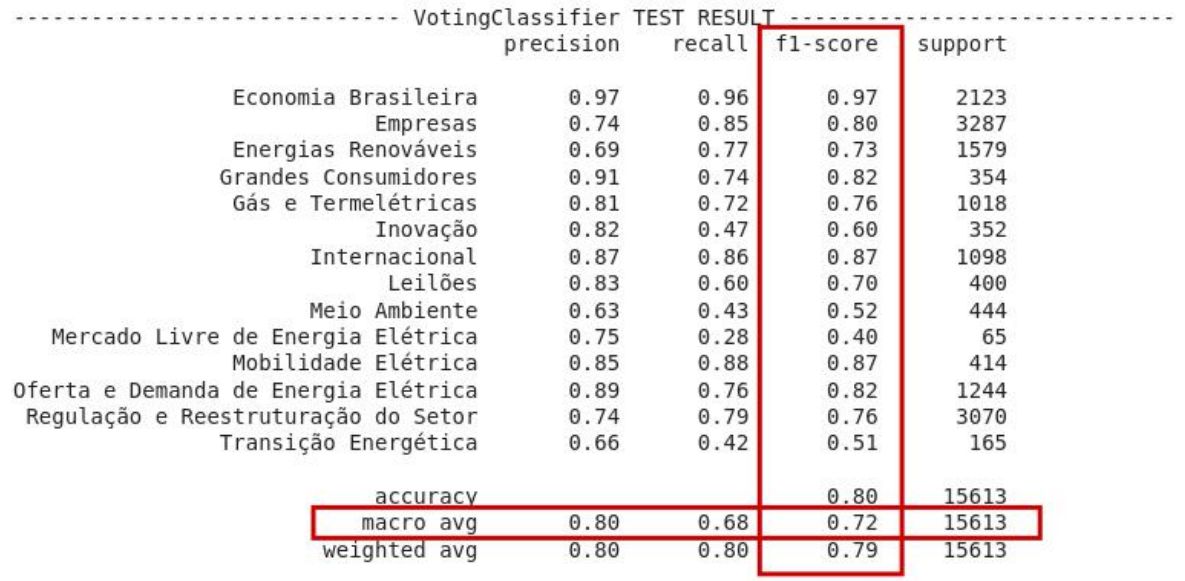
Figura 33 – Resultado da predição de *RandomForestClassifier*.



---------- RandomForestClassifier TEST RESULT ----------

	precision	recall	f1-score	support
Economia Brasileira	0.88	0.95	0.91	2123
Empresas	0.56	0.88	0.68	3287
Energias Renováveis	0.66	0.59	0.62	1579
Grandes Consumidores	1.00	0.10	0.19	354
Gás e Termelétricas	0.80	0.37	0.51	1018
Inovação	0.90	0.20	0.33	352
Internacional	0.91	0.47	0.62	1098
Leilões	0.94	0.15	0.27	400
Meio Ambiente	0.68	0.04	0.08	444
Mercado Livre de Energia Elétrica	0.00	0.00	0.00	65
Mobilidade Elétrica	0.84	0.72	0.78	414
Oferta e Demanda de Energia Elétrica	0.94	0.63	0.75	1244
Regulação e Reestruturação do Setor	0.59	0.82	0.69	3070
Transição Energética	0.83	0.03	0.06	165
accuracy			0.67	15613
macro avg	0.75	0.43	0.46	15613
weighted avg	0.73	0.67	0.64	15613

Figura 34 – Resultado da predição de *VotingClassifier*.



---------- VotingClassifier TEST RESULT ----------

	precision	recall	f1-score	support
Economia Brasileira	0.97	0.96	0.97	2123
Empresas	0.74	0.85	0.80	3287
Energias Renováveis	0.69	0.77	0.73	1579
Grandes Consumidores	0.91	0.74	0.82	354
Gás e Termelétricas	0.81	0.72	0.76	1018
Inovação	0.82	0.47	0.60	352
Internacional	0.87	0.86	0.87	1098
Leilões	0.83	0.60	0.70	400
Meio Ambiente	0.63	0.43	0.52	444
Mercado Livre de Energia Elétrica	0.75	0.28	0.40	65
Mobilidade Elétrica	0.85	0.88	0.87	414
Oferta e Demanda de Energia Elétrica	0.89	0.76	0.82	1244
Regulação e Reestruturação do Setor	0.74	0.79	0.76	3070
Transição Energética	0.66	0.42	0.51	165
accuracy			0.80	15613
macro avg	0.80	0.68	0.72	15613
weighted avg	0.80	0.80	0.79	15613

APÊNDICE C – TEMPOS MÉDIOS AFERIDOS E MÉTRICAS DA BUSCA EM GRADE DE HIPERPARÂMETROS

Tabela 5 – Tempos médios (por dobra, em segundos) aferidos no processo de treinamento dos classificadores.

CLASSIFICADOR	TEMPO MÉDIO
KNeighborsClassifier	0.072
LogisticRegression	26.335
MultinomialNB	0.249
LinearSVC	5.038
RandomForestClassifier	134.291
VotingClassifier	188.910

Figura 35 – Tempos médios aferidos e métricas de *KNeighborsClassifier* na busca em grade de hiperparâmetros.

	n_neighbors	weights	mean_fit_time	accuracy	precision	recall	f1-score
0	3	distance	0:00:00.033	0.48552	0.714704	0.319330	0.377326
1	6	distance	0:00:00.034	0.46696	0.807937	0.295426	0.348796
2	9	distance	0:00:00.035	0.46640	0.831970	0.287013	0.337465
3	12	distance	0:00:00.032	0.46640	0.818838	0.283951	0.332477
4	15	distance	0:00:00.032	0.46668	0.813641	0.281238	0.327637
5	3	uniform	0:00:00.034	0.47448	0.793388	0.301468	0.356008
6	6	uniform	0:00:00.030	0.44912	0.814783	0.272368	0.316506
7	9	uniform	0:00:00.035	0.45476	0.838214	0.274991	0.320258
8	12	uniform	0:00:00.030	0.45740	0.804545	0.272377	0.315141
9	15	uniform	0:00:00.031	0.45788	0.817179	0.271280	0.313019

Figura 36 – Tempos médios aferidos e métricas de *LogisticRegression* na busca em grade de hiperparâmetros.

	C	penalty	solver	mean_fit_time	accuracy	precision	recall	f1-score
0	0.01	l1	saga	0:00:04.688	0.23192	0.922776	0.091478	0.058978
1	0.01	l2	lbfgs	0:00:07.910	0.49512	0.890097	0.210487	0.189440
2	0.01	l2	newton-cg	0:00:12.019	0.49512	0.890097	0.210487	0.189440
3	0.01	l2	sag	0:00:02.432	0.49512	0.890097	0.210487	0.189440
4	0.01	l2	saga	0:00:04.292	0.49512	0.890097	0.210487	0.189440
5	0.01	None	lbfgs	0:00:28.771	0.75248	0.716213	0.654506	0.673649
6	0.01	None	newton-cg	0:01:54.781	0.73712	0.694837	0.640230	0.661231
7	0.01	None	sag	0:00:15.683	0.75368	0.712813	0.649180	0.672590
8	0.01	None	saga	0:00:19.440	0.75700	0.723236	0.652389	0.677507
9	0.10	l1	saga	0:00:06.474	0.67352	0.758107	0.474131	0.510828
10	0.10	l2	lbfgs	0:00:17.794	0.69212	0.817414	0.441591	0.478037
11	0.10	l2	newton-cg	0:00:20.013	0.69204	0.817382	0.441562	0.478008
12	0.10	l2	sag	0:00:02.378	0.69204	0.817382	0.441562	0.478008
13	0.10	l2	saga	0:00:03.691	0.69212	0.817444	0.441644	0.478091
14	0.10	None	lbfgs	0:00:25.396	0.75248	0.716213	0.654506	0.673649
15	0.10	None	newton-cg	0:01:52.672	0.73712	0.694837	0.640230	0.661231
16	0.10	None	sag	0:00:15.793	0.75476	0.724351	0.654055	0.679107
17	0.10	None	saga	0:00:19.527	0.75788	0.723046	0.654838	0.679723
18	1.00	l1	saga	0:00:40.902	0.76900	0.713626	0.639512	0.666817
19	1.00	l2	lbfgs	0:00:25.674	0.77860	0.766994	0.626243	0.662342
20	1.00	l2	newton-cg	0:00:31.795	0.77828	0.766906	0.625678	0.661749
21	1.00	l2	sag	0:00:03.207	0.77824	0.766866	0.625664	0.661726
22	1.00	l2	saga	0:00:04.142	0.77832	0.766919	0.626050	0.662177
23	1.00	None	lbfgs	0:00:25.834	0.75248	0.716213	0.654506	0.673649
24	1.00	None	newton-cg	0:01:54.372	0.73712	0.694837	0.640230	0.661231
25	1.00	None	sag	0:00:15.944	0.75420	0.719503	0.652901	0.676711
26	1.00	None	saga	0:00:19.756	0.75764	0.720866	0.652043	0.676509
27	10.00	l1	saga	0:03:01.213	0.76368	0.714638	0.661653	0.682884
28	10.00	l2	lbfgs	0:00:25.265	0.78136	0.748723	0.664397	0.692330
29	10.00	l2	newton-cg	0:00:40.100	0.78040	0.749554	0.661026	0.690343
30	10.00	l2	sag	0:00:09.775	0.78040	0.749554	0.661026	0.690343
31	10.00	l2	saga	0:00:19.200	0.78048	0.749544	0.661311	0.690592
32	10.00	None	lbfgs	0:00:25.393	0.75248	0.716213	0.654506	0.673649
33	10.00	None	newton-cg	0:01:54.461	0.73712	0.694837	0.640230	0.661231
34	10.00	None	sag	0:00:15.459	0.75352	0.718196	0.652153	0.675855
35	10.00	None	saga	0:00:19.296	0.75836	0.726360	0.655976	0.681214
36	100.00	l1	saga	0:09:35.076	0.75732	0.719829	0.656370	0.679630
37	100.00	l2	lbfgs	0:00:25.538	0.76592	0.737009	0.662799	0.687798
38	100.00	l2	newton-cg	0:01:06.527	0.76032	0.739389	0.650738	0.678423
39	100.00	l2	sag	0:00:15.577	0.76000	0.732872	0.650260	0.677265
40	100.00	l2	saga	0:00:19.545	0.76176	0.733275	0.653432	0.680475
41	100.00	None	lbfgs	0:00:25.518	0.75248	0.716213	0.654506	0.673649
42	100.00	None	newton-cg	0:01:54.676	0.73712	0.694837	0.640230	0.661231
43	100.00	None	sag	0:00:15.793	0.75412	0.718819	0.653145	0.677397
44	100.00	None	saga	0:00:19.520	0.75784	0.722600	0.654190	0.678645

Figura 37 – Tempos médios aferidos e métricas de *MultinomialNB* na busca em grade de hiperparâmetros.

	alpha	mean_fit_time	accuracy	precision	recall	f1-score
0	0.0	0:00:00.191	0.69452	0.710820	0.531177	0.575690
1	0.1	0:00:00.179	0.72776	0.818639	0.527534	0.565942
2	0.2	0:00:00.156	0.69516	0.836276	0.453471	0.481961
3	0.3	0:00:00.101	0.67036	0.837306	0.408132	0.429654
4	0.4	0:00:00.102	0.64732	0.843212	0.372414	0.390046
5	0.5	0:00:00.105	0.62872	0.852849	0.346282	0.360718
6	0.6	0:00:00.102	0.61288	0.863326	0.325854	0.338273
7	0.7	0:00:00.100	0.60124	0.864599	0.311563	0.322041
8	0.8	0:00:00.101	0.59216	0.867876	0.300176	0.308484
9	0.9	0:00:00.100	0.58316	0.869217	0.290153	0.296212
10	1.0	0:00:00.101	0.57408	0.869742	0.280000	0.283083

Figura 38 – Tempos médios aferidos e métricas de *LinearSVC* na busca em grade de hiperparâmetros.

	C	dual	penalty	mean_fit_time	accuracy	precision	recall	f1-score
0	0.01	False	l1	0:00:01.607	0.59676	0.762371	0.376664	0.401771
1	0.10	False	l1	0:00:04.732	0.74956	0.762210	0.581623	0.605531
2	1.00	False	l1	0:00:38.364	0.78372	0.736642	0.667255	0.689899
3	10.00	False	l1	0:01:25.359	0.73312	0.672922	0.637217	0.651861
4	100.00	False	l1	0:01:38.767	0.72324	0.662129	0.627763	0.642125
5	0.01	False	l2	0:00:05.690	0.71436	0.810279	0.495523	0.531313
6	0.10	False	l2	0:00:06.334	0.78528	0.801253	0.638535	0.669955
7	1.00	False	l2	0:00:08.860	0.78480	0.755672	0.670532	0.696081
8	10.00	False	l2	0:00:16.666	0.74756	0.704455	0.648108	0.668938
9	100.00	False	l2	0:00:34.027	0.72432	0.666387	0.627663	0.643156
10	0.01	True	l2	0:00:02.215	0.71436	0.810321	0.495523	0.531321
11	0.10	True	l2	0:00:01.324	0.78532	0.801266	0.638548	0.669968
12	1.00	True	l2	0:00:02.535	0.78480	0.755675	0.670533	0.696082
13	10.00	True	l2	0:00:06.942	0.74760	0.703981	0.648134	0.668835
14	100.00	True	l2	0:00:14.644	0.72212	0.664993	0.626268	0.641989

Figura 39 – Tempos médios aferidos e métricas de *RandomForestClassifier* na busca em grade de hiperparâmetros (parte 1).

	criterion	max_depth	n_estimators	mean_fit_time	accuracy	precision	recall	f1-score
0	entropy	5	100	0:00:02.948	0.43272	0.912066	0.182693	0.166112
1	entropy	10	100	0:00:06.324	0.51496	0.877746	0.234610	0.224710
2	entropy	15	100	0:00:10.872	0.57672	0.842922	0.300336	0.312579
3	entropy	20	100	0:00:16.501	0.62120	0.824933	0.354410	0.375490
4	entropy	25	100	0:00:23.347	0.65120	0.795937	0.399731	0.430194
5	entropy	5	200	0:00:05.810	0.42756	0.916780	0.179362	0.164435
6	entropy	10	200	0:00:12.233	0.51668	0.884069	0.233741	0.222429
7	entropy	15	200	0:00:20.905	0.57996	0.863953	0.298010	0.308807
8	entropy	20	200	0:00:33.178	0.62996	0.841739	0.366161	0.389802
9	entropy	25	200	0:00:46.696	0.65780	0.840121	0.404242	0.433386
10	entropy	5	300	0:00:08.494	0.42980	0.916126	0.181183	0.166220
11	entropy	10	300	0:00:18.277	0.51572	0.887923	0.233007	0.221808
12	entropy	15	300	0:00:31.492	0.58184	0.866736	0.299032	0.308535
13	entropy	20	300	0:00:49.579	0.63176	0.845303	0.362480	0.384582
14	entropy	25	300	0:01:10.675	0.65948	0.835892	0.404959	0.433050
15	entropy	5	400	0:00:11.907	0.43444	0.909578	0.183045	0.167492
16	entropy	10	400	0:00:24.215	0.51488	0.888131	0.232356	0.219545
17	entropy	15	400	0:00:42.005	0.58132	0.863820	0.298070	0.307300
18	entropy	20	400	0:01:06.143	0.62880	0.825282	0.360447	0.380590
19	entropy	25	400	0:01:32.973	0.66088	0.836316	0.405753	0.433411
20	entropy	5	500	0:00:14.633	0.43976	0.917501	0.184362	0.168155
21	entropy	10	500	0:00:30.155	0.51744	0.885804	0.232513	0.219074
22	entropy	15	500	0:00:52.654	0.58540	0.868403	0.301996	0.311648
23	entropy	20	500	0:01:22.548	0.63192	0.827571	0.361589	0.383199
24	entropy	25	500	0:01:57.723	0.65936	0.834023	0.403191	0.429510
25	gini	5	100	0:00:03.316	0.43284	0.903289	0.181560	0.165342
26	gini	10	100	0:00:04.479	0.50864	0.884102	0.229524	0.219857
27	gini	15	100	0:00:07.012	0.57236	0.870034	0.290188	0.302208
28	gini	20	100	0:00:10.396	0.62200	0.859001	0.348215	0.367279
29	gini	25	100	0:00:14.631	0.65092	0.841034	0.389347	0.413498
30	gini	5	200	0:00:06.526	0.43344	0.917305	0.183099	0.167800
31	gini	10	200	0:00:13.323	0.51448	0.885571	0.231249	0.219056
32	gini	15	200	0:00:14.473	0.57692	0.876976	0.288019	0.295619
33	gini	20	200	0:00:21.231	0.62176	0.866993	0.342485	0.359602
34	gini	25	200	0:00:30.225	0.65712	0.824218	0.391713	0.412749
35	gini	5	300	0:00:09.682	0.44260	0.918441	0.186649	0.170686
36	gini	10	300	0:00:19.889	0.51556	0.888053	0.230518	0.217381
37	gini	15	300	0:00:21.088	0.57652	0.881580	0.289294	0.298710
38	gini	20	300	0:00:31.562	0.62540	0.860694	0.345277	0.362660
39	gini	25	300	0:00:44.431	0.65700	0.844974	0.393063	0.416678

Figura 40 – Tempos médios aferidos e métricas de *RandomForestClassifier* na busca em grade de hiperparâmetros (parte 2).

	criterion	max_depth	n_estimators	mean_fit_time	accuracy	precision	recall	f1-score
40	gini	5	400	0:00:13.066	0.44060	0.917821	0.185781	0.169733
41	gini	10	400	0:00:21.195	0.51576	0.887192	0.230316	0.217040
42	gini	15	400	0:00:27.607	0.57580	0.875984	0.286869	0.296007
43	gini	20	400	0:00:42.270	0.62788	0.867448	0.346648	0.364193
44	gini	25	400	0:00:59.299	0.65720	0.846408	0.391965	0.413463
45	gini	5	500	0:00:16.410	0.43848	0.919240	0.184098	0.168657
46	gini	10	500	0:00:20.671	0.51440	0.889857	0.228997	0.214318
47	gini	15	500	0:00:34.345	0.57752	0.878728	0.288061	0.296518
48	gini	20	500	0:00:52.520	0.62648	0.857256	0.348803	0.367234
49	gini	25	500	0:01:14.872	0.65776	0.861694	0.391004	0.413194
50	log_loss	5	100	0:00:02.891	0.42728	0.913532	0.179806	0.164317
51	log_loss	10	100	0:00:06.090	0.51816	0.879574	0.238401	0.230304
52	log_loss	15	100	0:00:10.436	0.57368	0.856242	0.294561	0.304848
53	log_loss	20	100	0:00:16.619	0.62336	0.833242	0.357180	0.379416
54	log_loss	25	100	0:00:23.581	0.65544	0.827842	0.406695	0.433796
55	log_loss	5	200	0:00:05.810	0.43740	0.911791	0.183787	0.167735
56	log_loss	10	200	0:00:12.121	0.51300	0.883877	0.230920	0.218884
57	log_loss	15	200	0:00:21.365	0.58056	0.863917	0.298013	0.309658
58	log_loss	20	200	0:00:33.621	0.62708	0.836165	0.359517	0.380004
59	log_loss	25	200	0:00:47.768	0.65732	0.837105	0.402875	0.430799
60	log_loss	5	300	0:00:08.508	0.42548	0.916898	0.178925	0.164454
61	log_loss	10	300	0:00:18.089	0.51612	0.878451	0.232406	0.219988
62	log_loss	15	300	0:00:32.006	0.58428	0.866918	0.301807	0.311703
63	log_loss	20	300	0:00:50.111	0.62948	0.845137	0.363038	0.385247
64	log_loss	25	300	0:01:11.810	0.66128	0.841744	0.408110	0.437148
65	log_loss	5	400	0:00:11.378	0.43640	0.916337	0.183701	0.167592
66	log_loss	10	400	0:00:24.314	0.52020	0.883342	0.236251	0.225959
67	log_loss	15	400	0:00:44.554	0.58336	0.869943	0.299649	0.309673
68	log_loss	20	400	0:01:06.611	0.63352	0.852399	0.363127	0.383148
69	log_loss	25	400	0:01:34.817	0.66040	0.836660	0.403379	0.429125
70	log_loss	5	500	0:00:14.405	0.43396	0.917358	0.182756	0.167465
71	log_loss	10	500	0:00:31.280	0.51604	0.886167	0.231500	0.218296
72	log_loss	15	500	0:00:53.467	0.57868	0.873312	0.295229	0.303489
73	log_loss	20	500	0:01:24.154	0.62960	0.830736	0.362720	0.385031
74	log_loss	25	500	0:01:58.929	0.66268	0.836999	0.406348	0.433823

Figura 41 – Tempos médios aferidos e métricas de *VotingClassifier* na busca em grade de hiperparâmetros.

	voting	mean_fit_time	accuracy	precision	recall	f1-score
0	hard	0:01:07.405	0.76680	0.800924	0.611454	0.664177
1	soft	0:01:11.405	0.76616	0.795596	0.624632	0.673834