Universidade de Brasília – UnB

Faculdade de Direito

LEONARDO CAPPELLESSO BIGOLIN

CLASSIFICAÇÃO DE CERTIDÕES DE JULGAMENTO DE MANDADOS DE SEGURANÇA DO STF UTILIZANDO APRENDIZADO DE MÁQUINA E ANÁLISE DE SENTIMENTOS

Classification of Brazilian's Supreme Court injunctions trial certificates with machine learning and sentiment analysis

Brasília

2023

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE DIREITO

CLASSIFICAÇÃO DE CERTIDÕES DE JULGAMENTO DE MANDADOS DE SEGURANÇA DO STF UTILIZANDO APRENDIZADO DE MÁQUINA E ANÁLISE DE SENTIMENTOS

Autor: Leonardo Cappellesso Bigolin

Orientador: Prof. Dr. Henrique Araujo Costa

Monografia apresentada como requisito parcial à obtenção do grau de Bacharel, no Programa de Graduação da Faculdade de Direito da Universidade de Brasília.

Brasília, 14 de fevereiro de 2023.

FOLHA DE APROVAÇÃO

Leonardo Cappellesso Bigolin

Classificação de certidões de julgamentos de Mandados de Segurança do STF utilizando aprendizado de máquina e análise de sentimentos.

Monografia apresentada como requisito parcial à obtenção do grau de Bacharel, no Programa de Graduação da Faculdade de Direito da Universidade de Brasília.

Aprovada em: 14 de fevereiro de 2023.

BANCA EXAMINADORA

_____

Prof. Dr. Henrique Araujo Costa

(Orientador – Presidente)

_____

Dr. Luan Carlos de Sena Monteiro Ozelim

_____

Dr. Darym Junior Ferrari de Campos

FICHA CATALOGRÁFICA

Cappellesso Bigolin, Leonardo

CB594c Classificação de certidões de julgamentos de Mandados de Segurança do STF utilizando aprendizado de máquina e análise de sentimentos. / Leonardo Cappellesso Bigolin; orientador Henrique Araujo Costa. -- Brasília, 2023.

55 p.

Monografia (Graduação - Direito) -- Universidade de Brasília, 2023.

1. Mandados de Segurança do STF. 2. Inteligência Artificial aplicada ao Direito. 3. Machine Learning. 4. Análise de Sentimentos. 5. Certidões de Julgamento. I. Araujo Costa, Henrique, orient. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

BIGOLIN, L. B. (2023). Classificação de certidões de julgamentos de Mandados de Segurança do STF utilizando aprendizado de máquina e análise de sentimentos. Monografia Final de Curso de Direito, Faculdade de Direito, Universidade de Brasília, Brasília, DF, 55 p.

RESUMO

Esta monografia de graduação trata de uma descrição do procedimento para o desenvolvimento de uma aplicação que utiliza aprendizado de máquina para classificar o resultado de decisões de mandados de seguranças impetrados no Supremo Tribunal Federal. Para isso, são utilizados como dados de treinamento a informação textual de certidões de julgamento, disponibilizadas pelo STF, e um rótulo, previamente atribuído para cada certidão, que pode representar a procedência do pedido do mandado de segurança ou a sua improcedência. Uma vez que aplicação seja desenvolvida, esta será capaz de gerar uma classificação para novas certidões de julgamento a ela apresentadas. Este trabalho fornece subsídios para a construção dessa aplicação e para que novas tecnologias sejam desenvolvidas utilizando dados textuais eletrônicos, uma vez que os processos judiciais estão predominantemente na forma eletrônica e, em boa parte, disponíveis para utilização.

ABSTACT

This final paper addresses the procedure description for the development of an application that uses machine learning to classify the result of decisions of injunctions filed in the Brazilian Supreme Court. For this, textual data from judgment certificates, made available by the STF, and a label, previously assigned to each certificate, are used as training data, which may represent the admission of the request of the injunction or its rejection. Once the application is developed, it will be able to generate a classification to new judgment certificates presented to it. This work provides subsidies for the construction of this application and for new technologies to be developed using electronic textual data, since judicial processes in Brazil are predominantly computerized and, in large part, available for use.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREAVIATURAS E SIGLAS

AM – Aprendizado de Máquina

CF – Constituição Federal

CNJ – Conselho Nacional de Justiça

IA – Inteligência Artificial

FN – Falso Negativo

FP – Falso Positivo

MS – Mandado de Segurança

OCR – *Optical Character Recognition*

PLN – Processamento de Linguagem Natural

STF – Supremo Tribunal Federal

VP – Verdadeiro Negativo

VP – Verdadeiro Positivo

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações Iniciais

O objetivo deste capítulo inicial é contextualizar o tema de classificação de certidões de julgamento utilizando análise de sentimentos e aprendizado de máquina demonstrando quais são as motivações para o desenvolvimento da presente monografia (seção 1.2) e, em seguida , evidenciar para o leitor quais sãos as questões de pesquisa que tentarão ser respondidas no trabalho e os seus objetivos (seção 1.3).

1.2 Contextualização e motivação

O congestionamento do judiciário é tema de discussão frequente no Brasil. Conforme aponta o relatório Justiça em Números de 2022 do Conselho Nacional de Justiça (CNJ), até dia 30 de abril de 2022 havia 76.600.070 processos pendentes no Poder Judiciário brasileiro. Esse mesmo relatório indica que para a manutenção do funcionamento deste poder, dada a grande quantidade de processos, o gasto público necessário é de R$ 83,7 bilhões, descontadas as despesas referentes a gastos com inativos. Esse valor representa 1% de todo o PIB nacional (CNJ, 2022).

Como forma de mitigação desse cenário litigioso, em conjunto com uma tendência de desjudicialização dos meios de resoluções de controvérsias (como a mediação e a arbitragem), surgem propostas do uso de tecnologias que se utilizam da engenharia de *software* direcionadas ao mercado jurídico.

Essas propostas utilizam a inteligência artificial (IA) com a finalidade de automatizar procedimentos repetitivos. Como a maior parte dos processos existentes atualmente se encontra na forma eletrônica, a disponibilidade de dados para alimentar esses sistemas é abundante. Como exemplo de potencial aplicação para esse tipo de tecnologia é possível citar o auxílio no julgamento de processos repetitivos, o que reduziria custos e permitiria o melhor aproveitamento do tempo dos agentes do Poder Judiciário.

Embora exista a necessidade de mais celeridade no julgamento de processos do Poder Judiciário e, consequentemente, um grande interesse na adoção da inteligência artificial para essa finalidade, muito há que ser desenvolvido sobre o tema, tanto âmbito de pesquisas em

aplicações e testes, como na parte da regulamentação adequada para o uso da inteligência artificial para essa finalidade.

Como contribuição para o desenvolvimento do tema, Guilherme Ramos de Morais, orientado pelo Dr. Henrique Araújo Costa, desenvolveu o trabalho "Direito e Inteligência Artificial: Análise de Sentimento Aplicada em Certidões de Julgamento de Mandados de Segurança Impetrados no Supremo Tribunal Federal" (MORAIS, 2019) em que foram classificadas, com o uso de análise de sentimentos, 2.158 certidões de julgamento de mandados de segurança do STF em duas categorias, que basicamente descrevem a procedência ou improcedência do pedido.

Aplicações que utilizam sistema de inteligência artificial, necessitam de dados rotulados para realizar o aprendizado de máquina e o trabalho de Morais (2019) contribui ao fornecer uma base de dados passível de ser utilizada nessas aplicações. Por esse motivo, a presente monografia tratará de descrever como os dados disponibilizados por aquele trabalho podem ser utilizados em uma aplicação de inteligência artificial capaz de predizer de forma autônoma o resultado de julgamentos de mandados de segurança impetrados no STF, utilizando o texto contido nas certidões de julgamento desses processos como dados de aprendizado de máquina. Além disso, o presente trabalho descreverá como avaliar os resultados que se espera obter após o desenvolvimento e utilização desta aplicação.

1.3 Questões de Pesquisa e Objetivos

Conforme mencionado anteriormente, a presente monografia utiliza o trabalho de Guilherme Ramos de Morais (MORAIS, 2019) para descrever como os dados fornecidos podem ser utilizados para o treinamento de um algoritmo de aprendizado de máquina para classificação da procedência do pedido utilizando certidões de julgamento de mandados de segurança interpostos no Supremo Tribunal Federal (STF). Dessa forma formulou-se a seguinte questão para nortear este projeto de pesquisa:

"*De que forma dados de classificação binária do resultado do julgamento de mandados de segurança impetrados no STF podem ser usados para o desenvolvimento de uma aplicação de aprendizado de máquina que prevê de forma autônoma o resultado de outros processos da mesma natureza?*".

Diante dessa questão de pesquisa, pode-se extrair os seguintes objetivos para o desenvolvimento deste trabalho:

- Descrever como os dados disponibilizados pelo trabalho de Morais (2019) podem ser utilizados em uma aplicação de aprendizado de máquina para prever o resultado de processos de mandado de segurança impetrados no STF;
- Descrever a etapa de tratamentos dos dados disponibilizados pelo trabalho de Morais (2019);
- Descrever como avaliar os resultados após o desenvolvimento da aplicação.

2 REVISÃO BIBLIOGRÁFICA

2.1 Considerações Iniciais

A descrição de uma aplicação que classifique de forma autônoma certidões de julgamento coletadas no STF exige conhecimento em diversas áreas. Dessa forma, para embasar o conteúdo deste trabalho, este capítulo abordará os principais tópicos necessários para o entendimento da descrição e desenvolvimento dessa aplicação.

Ao formalizar uma manifestação a respeito de uma decisão, o STF o faz por meio da linguagem textual. Por isso, para que seja possível a interpretação dessa linguagem pelas máquinas, deve-se realizar a sua transposição para uma forma compreensível para elas. Isso é feito com ajuda do Processamento de Linguagem Natural (PLN), abordado na seção 2.2.

Uma vez transportados os dados para uma linguagem computacional, é possível fazer com que a máquina absorva conhecimento utilizando esses dados e o aplique em outras atividades, por esse motivo, na seção 2.3 será abordado o tópico de aprendizado de máquina.

Por fim, conforme mencionado, a presente monografia se utiliza de dados previamente coletados para o alcance de seus objetivos. Assim, a seção 2.4 trata de descrever a base de dados construída no trabalho de Morais (2019).

2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é um dos ramos da Inteligência Artificial, e que também se comunica com a Linguística, que busca processar e analisar a linguagem natural, que é a linguagem que o ser humano utiliza para se comunicar, seja na forma escrita ou na falada. Em outras palavras, o PLN transforma as palavras e textos, que utilizamos para transmitir informações, para uma representação na qual as máquinas são capazes de compreender para que, através dessa compreensão, seja possível a realização de tarefas autônomas (VASILIEV, 2020; VAJJALA, MAJUMDER, *et al.*, 2020).

Diversas aplicações que utilizamos em nosso cotidiano utilizam PLN. Alguns exemplos são os assistentes de voz (como Apple Siri e Amazon Alexa), tradutores, classificação e resumo de textos, recuperação e extração de informações textuais contidas em documentos, filtros de *spam* em plataformas de *e-mail*, robôs que simulam uma conversa com um ser humano

(*chatbots*), corretores gramaticais, construção de bases de conhecimento e diversos outros (VAJJALA, MAJUMDER, *et al.*, 2020).

Pela definição de VAJJALA *et al*. (2020), a linguagem natural "é um sistema estruturado de comunicação que envolve combinações complexas de seus componentes constituintes, como caracteres, palavras, sentenças, etc.", pode-se supor que, para viabilizar o uso de uma aplicação em PLN que possa ser utilizada para utilidades práticas e retornando um resultado confiável, são necessárias grandes quantidades de dados textuais ou de fala. Esse banco de dados textual utilizado para essas aplicações é comumente chamado de *corpus*.

Dessa forma, o *corpus* é utilizado como dados de treinamento para aplicações que utilizam o aprendizado de máquina para realizar determinada tarefa. Um exemplo de uma aplicação do uso de aprendizado supervisionado seria a utilização de um conjunto de artigos de notícias, classificadas manualmente em diversas categorias para desenvolver uma aplicação que classifica automaticamente, conforme o seu assunto, uma notícia fornecida como *input*. Por outro lado, é possível a utilização do PLN em aplicações de aprendizado não supervisionado, como exemplo para agrupar dados textuais com características semelhantes (VAJJALA, MAJUMDER, *et al.*, 2020).

2.2.1 Pré-processamento

Uma etapa de pré-processamento pode ser realizada nos dados de treinamento, antes de alimentar um sistema que se utiliza de algoritmos de aprendizado de máquina para remover informações irrelevantes sobre esses dados. Exemplos de pré-processamento pode ser a remoção de *links*, *emoticons*, pontuação, *hashtags*, caracteres especiais, e qualquer informação que não seja útil, a depender do caso. Em algumas situações pode-se remover determinadas palavras não relevantes para a realização da tarefa em questão (denominadas *stopwords*), como por exemplo artigos, conjunções, pronomes ou até mesmo palavras que se repetem com frequência e não acrescentam à semântica do texto. Além disso, pode ser irrelevante a presença de caracteres maiúsculos no texto e a etapa de pré-processamento pode consistir na passagem de todos os caracteres existentes para minúsculos ou vice-versa (VAJJALA, MAJUMDER, *et al.*, 2020; GÉRON, 2019).

Além dos mencionados no parágrafo anterior, podem ser utilizados pré-processamentos que atuam a nível de palavras do texto, como a "stemização" e a "lematização". O nome "stemização" é uma derivação aportuguesada da palavra "*stem"* que, na língua inglesa, se refere

ao radical de uma palavra. O processo de “stemização” consiste na remoção de sufixos e, consequentemente, na redução de cada palavra a seu radical, que gera todas as demais variantes dessa palavra. Já o processo de “lematização”, consiste no mapeamento de todos os diferentes formatos de uma palavra a uma palavra base ou lema, ou seja, se refere à derivação linguística das palavras. Apesar de “stemização” e “lematização” aparentarem similares, estas não o são. Por exemplo, ao aplicar a “stemização” à palavra “amiga” se obteria “amig”, mas ao se aplicar a lematização à mesma palavra o resultado seria a palavra “amizade” (VAJJALA, MAJUMDER, *et al.*, 2020; GANEGEDARA, 2022).

Ressalta-se que nem sempre todos os pré-processamentos mencionados são necessários e que não necessariamente são executados na ordem que foram mencionados. Por exemplo, se for necessária a remoção de dígitos e pontuação, essa etapa pode ser realizada após a realização da “stemização” ou “lematização”, porém comumente passa-se todos os caracteres para minúsculo antes de realizar esses procedimentos. Além disso, as etapas mencionadas não são as únicas existentes, podendo outros tipos de processamento serem necessários a depender da natureza dos dados (VAJJALA, MAJUMDER, *et al.*, 2020; GANEGEDARA, 2022).

2.2.2 Tokenização

A etapa após o pré-processamento dos dados consiste em dividir o texto em pedaços para que o sistema de PLN seja capaz de analisar a informação textual e essa divisão pode ser, por exemplo, a nível de palavras ou de caracteres. Como cada palavra deve ser analisada no contexto em que é inserida, primeiramente divide-se o texto em sentenças, procedimento que é denominado segmentação de sentenças. Posteriormente, divide-se cada sentença em conjunto de palavras ou de caracteres, que são chamadas de *tokens*, motivo pelo qual essa etapa é comumente chamada de “tokenização” (VAJJALA, MAJUMDER, *et al.*, 2020; GANEGEDARA, 2022).

2.2.3 Transformando o texto em vetores

Como mencionado anteriormente, uma aplicação em PLN deve ser capaz de converter a linguagem natural para uma representação em que seja possível o seu processamento pelos computadores. Essa conversão, no caso de dados textuais, é chamada pela literatura de representação de texto (*text representation*) e consiste na transformação do texto bruto para

uma linguagem numérica, mais especificamente na transformação para um vetor ou matriz numérica (VAJJALA, MAJUMDER, *et al.*, 2020).

Uma das abordagens básicas de vetorização é a Codificação *one-hot* (*one-hot encoding*) em que é atribuída para cada palavra diferente do *corpus* um número inteiro, em seguida esse número é representado em uma posição de um vetor de dimensão igual ao número de palavras diferentes presentes nesse *corpus*. Como um exemplo muito simplificado, pode-se imaginar um *corpus* que contém três palavras: "rato"; "comeu"; e "queijo". Primeiramente para cada palavra é atribuído um número inteiro, exemplo: rato = 1; comeu = 2; e queijo = 3. Em seguida, como há apenas três palavras únicas nesse *corpus*, a cada número inteiro é atribuído um vetor de dimensão 3: rato = 1 = [0 0 1]; comeu = 2 = [0 1 0]; e queijo = 3 = [0 0 1] (VAJJALA, MAJUMDER, *et al.*, 2020; GANEGEDARA, 2022).

Outra abordagem comumente utilizada para vetorização de textos é a chamada Saco de Palavras (*bag of words*). Essa técnica define um vetor de dimensão igual ao número de palavras únicas do *corpus* para cada documento integrante desse conjunto de dados. Em seguida o vetor de cada documento é preenchido de acordo com o número de ocorrências de cada palavra, que é armazenada na posição referente à palavra em questão. Como exemplo pode-se considerar um caso em que o *corpus* possui seis palavras diferentes e a palavra "rato" foi observada três vezes em um dos documentos desse *corpus*, a palavra "comeu" uma vez e as palavras "queijo", "comida", "peixe" e "gato" não apareceram nenhuma vez no documento em questão. Supondo que as palavras foram armazenas na ordem em que foram mencionadas, o vetor resultante desse documento seria [3 2 0 0 0 0]. Essa técnica é muito utilizada para tarefas de classificação de textos, uma vez que, ao se comparar vetores de diferentes textos do *corpus*, pode-se observar similaridades semânticas (GANEGEDARA, 2022).

Nas duas abordagens descritas acima, as palavras foram tratadas como unidades independentes, ou seja, ignorando-se o seu contexto e a ordem em que foram observadas no texto. Para considerar esses dois fatores, uma técnica de vetorização utilizada é o Saco de *N-grams* (*bag of N-grams*). Essa técnica quebra o texto em pedaços (*tokens*) de N caracteres ou N palavras contíguas, podendo N ser qualquer número inteiro e cada pedaço é chamado de *N-gram*. Utilizando essa abordagem é criado um vetor de dimensão N (em que N é igual ao número de *N-grams* diferentes no *corpus*), para cada documento do *corpus*, é registrado a frequência de cada *N-gram* presente nesse documento, da mesma maneira que o Saco de Palavras (VAJJALA, MAJUMDER, *et al.*, 2020; GANEGEDARA, 2022).

Por fim, uma abordagem muito utilizada para recuperação de informações em documentos é a chamada TF-IDF. Essa técnica possui a vantagem de ponderar a importância de cada palavra em relação a outras nos documentos textuais. A ideia é que se uma palavra aparece diversas vezes em um determinado documento, mas ocorre pouco ou nenhuma vez em outros documentos, consequentemente supõe-se que essa palavra é importante para o documento em questão. Matematicamente isso é representado pela combinação de duas medidas: a frequência do termo em um documento (DF) e a frequência desse termo em todos os documentos (IDF). Como em documentos muito longos a quantidade de um determinado termo relevante pode ser repetida mais vezes do que em documentos menores, a frequência de termo é normalizada pela quantidade total de termos do documento. Por fim divide-se DF por IDF para se determinar a importância de determinado termo para o documento (VAJJALA, MAJUMDER, *et al.*, 2020).

Uma vez transportado o texto para um vetor numérico, é possível induzir a máquina a extrair conhecimento desses dados vetorizados e utilizá-lo para a realização de tarefas automatizadas, assunto que será abordado na seção 2.3.

2.2.4 Análise de sentimentos

A análise de sentimentos, ou mineração de opinião, é um ramo do estudo que investiga os sentimentos, opiniões, atitudes e emoções em relação a entidades e como esses atributos são expressos. Apesar de já haver avanços em outros formatos de dados, pesquisas e aplicações têm focado principalmente em texto escrito e, por isso, a análise de sentimento têm sido um campo de pesquisa do Processamento de Linguagem Natural (PLN) e, nesse caso, utiliza as palavras para tentar identificar opiniões positivas ou negativas expressas em determinado texto assim como o alvo da opinião ou sentimento (MAJUMDAR, 2021).

Já se constata o uso de aplicações que utilizam análises de sentimento na política. Através do monitoramento de postagem de usuários em redes sociais é possível ter um entendimento da opinião geral do público a respeito de um determinado tópico, possibilitando aos governantes responderem de maneira mais célere à situação. Além da política, a análise de sentimento já é implementada na área de negócios ao minerar a opinião de consumidores a respeito de produtos, filmes e até ações do mercado financeiro (LIU, 2020).

Há três níveis de contexto (ou granularidade) nas pesquisas de análise de sentimento: nível de documento; nível de sentença; e nível de aspecto. A nível de documento, a análise é feita levando-se em consideração toda a informação contida em um documento e retornando um resultado que leva em consideração o sentimento geral dele. Em nível de sentença, é avaliado o sentimento de cada sentença separadamente e retorna-se ao final um resultado levando-se em consideração cada uma delas. Por fim, em nível de aspecto tem-se em vista o alvo das opiniões contidas nas sentenças do documento, uma vez que uma sentença pode expressar sentimentos diversos para entidades diferentes, como no exemplo a seguir "apesar de o serviço não ter sido ótimo, eu ainda amei o restaurante". Nessa sentença é atribuído um sentimento negativo ao serviço prestado, porém um sentimento positivo é atribuído ao restaurante (LIU, 2020).

2.3 Aprendizado de Máquina

O aprendizado de máquina é um ramo da inteligência artificial que se utiliza de dados para alimentar algoritmos, que extraem conhecimento desses dados com a finalidade de realizar predições. Com a utilização do aprendizado de máquina não é necessário o desenvolvimento de regras ou de modelos para a realização da tarefa, ao invés disso o algoritmo captura o conhecimento presente nos dados de maneira a aumentar a sua performance gradualmente de forma autônoma. Quanto maior for a quantidade de dados disponíveis, a capacidade da máquina de fazer predições precisas tende a melhorar (RASHKA e MIRJALILI, 2019).

Cada vez mais o aprendizado de máquina vem sendo utilizado em pesquisas e projetos da ciência da computação e já se apresenta em diversas aplicações do dia a dia tais como em filtros de *spam*, *softwares* de reconhecimento de texto, voz ou biometria, *sites* de busca na *Internet* e até em jogos de Xadrez. Porém grandes avanços têm sido feitos em projetos mais complexos como modelos que detectam câncer de pele ou que predizem a estrutura tridimensional de proteínas (RASHKA e MIRJALILI, 2019; GÉRON, 2019).

Há três tipos principais de aprendizado de máquina: aprendizado supervisionado; aprendizado não supervisionado; e aprendizado por reforço. O aprendizado supervisionado se utiliza de dados de treinamento cujo rótulo da classificação que se deseja fazer já é conhecido e, posteriormente, utiliza o conhecimento adquirido por esses dados para classificar novos dados em que esse rótulo é desconhecido. Como exemplo pode-se citar uma aplicação para

reconhecer imagens de elefantes e girafas e classificá-las nessas duas categorias. Para isso o sistema é alimentado com diversas imagens de girafas e de elefantes já rotuladas para poder fazer a classificação de novas imagens. Essa classificação do aprendizado supervisionado pode ser realizada para um número indefinido de categorias, porém para o caso em que há apenas duas categorias é chamada de classificação binária (GÉRON, 2019).

Diferentemente, no aprendizado não supervisionado o sistema é alimentado com dados não rotulados e o algoritmo detectará similaridades nesses dados para agrupá-los em diferentes categorias. Um exemplo de aplicação do aprendizado não supervisionado é o agrupamento de diversos tipos de consumidores de acordo com características semelhantes. Ressalta-se que na ausência de rótulos em todos os dados é possível a utilização de dados rotulados em conjunto com dados não rotulados para o treinamento do algoritmo, o que consiste no aprendizado semi-supervisionado (RASHKA e MIRJALILI, 2019; GÉRON, 2019).

Finalmente, no aprendizado por reforço o objetivo é desenvolver um sistema, que comumente é chamado de agente, que gradualmente tende a melhorar a sua performance através de interações com o ambiente. Caso o agente se mova em direção a seu objetivo, este recebe recompensas positivas e entende que percorreu um caminho melhor que o anterior. Diferentemente, caso se distancie de seu objetivo, recebe recompensas negativas e percebe que não deveria ter realizado seu último movimento. Após a realização de diversas simulações o melhor resultado para o ambiente em estudo é aquele que obteve o maior número de recompensas alcançando seu objetivo (FENNER, 2019; RASHKA e MIRJALILI, 2019).

Como o presente trabalho se propõe a descrever um sistema de aprendizado de máquina que classifica as certidões de julgamento de ações do STF entre certidões com sentimento positivo e certidões com sentimento negativo utilizando dados já rotulados, a técnica utilizada para essa atividade é o aprendizado de máquina supervisionado para classificação binária. Por isso, os próximos tópicos vão se ater a esse tipo de aprendizado, descrevendo o funcionamento dos dados de treinamento e teste e as principais medidas de desempenho.

2.3.1 Treinamento e teste

Antes de ser possível a utilização de uma aplicação de aprendizado de máquina para realizar determinada tarefa, o algoritmo deve ser alimentado com dados de treinamento para que ocorra o aprendizado. Posteriormente, para determinar se o desempenho do aprendizado

foi aceitável, o algoritmo deve ser testado. Por esse motivo, divide-se os dados disponíveis em um conjunto de dados de treinamento e outro conjunto de dados de teste (FENNER, 2019; CABRAL, 2021).

Essa divisão ocorre separando-se uma percentagem dos dados para o conjunto de treinamento e o resto para o conjunto de teste. Em seguida, ocorre o treinamento e, posteriormente, o teste. Este último é realizado ao compelir o algoritmo a classificar os dados de teste utilizando o conhecimento adquirido na etapa de treinamento, por isso, os dados de teste são apresentados sem os seus respectivos rótulos de classificação. Por fim, compara-se as classificações realizadas pelo algoritmo treinado com os rótulos originais dos dados de teste e obtêm-se a quantidade de erros e acertos obtidas (FENNER, 2019; RASHKA e MIRJALILI, 2019).

Apesar de ocorrer essa divisão entre dados de teste e dados de treinamento, na prática, o algoritmo é testado para ambos os conjuntos de dados. Ou seja, após a fase de treinamento, o algoritmo é compelido a classificar tanto os dados de teste como os próprios dados de treinamento. O motivo disso é para verificar o quanto o algoritmo aprendeu dos dados de treinamento e verificar também a sua capacidade de generalização deste conhecimento para novos dados (RASHKA e MIRJALILI, 2019).

Por exemplo, após o treinamento e teste do algoritmo pode-se obter um resultado em que a quantidade de acertos para os dados de treinamento é muito grande e, diferentemente, muito pequena para os dados de teste. Nesse cenário, percebe-se que o algoritmo conseguiu aprender de forma eficiente o conhecimento presente nos dados de treinamento, porém não consegue aplicar (generalizar) com a mesma eficiência esse conhecimento para a classificação de novos dados. A literatura, ao se referir a essa situação, diz que o modelo está sobre ajustado aos dados de treinamento (*overfitting*) e que possui um viés alto em relação a esses dados, como ilustra o gráfico da Figura 1 (RASHKA e MIRJALILI, 2019; GÉRON, 2019).

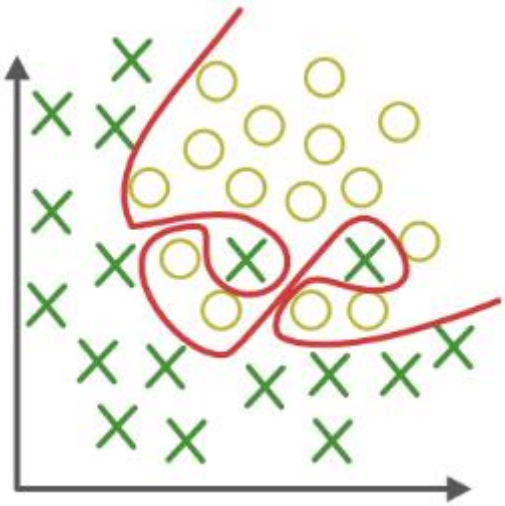

Figura 1 - Exemplo gráfico de modelo de classificação sobreajustado (*overfitting*). Disponível em https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. Acesso em: 20 nov. 2022.

Diferentemente, para o caso de o algoritmo não obter boa performance para os dados de treinamento diz-se que o modelo está sub ajustado aos dados de treinamento (*underfitting*). Neste caso, o algoritmo não foi capaz de aprender de aprender de forma a retornar resultados confiáveis sendo a sua variância muito grande como mostra o gráfico da Figura 2 (RASHKA e MIRJALILI, 2019; GÉRON, 2019).

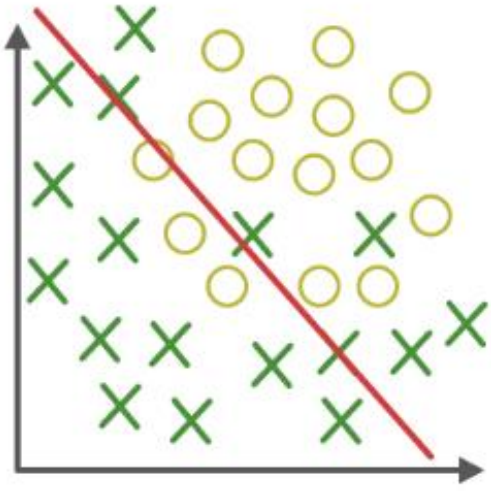

Figura 2 - Exemplo gráfico de modelo de classificação sub ajustado (*underfitting*). Disponível em https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. Acesso em: 20 nov. 2022.

Um modelo ideal deve ter boa performance tanto para os dados de treinamento como para os de teste, isso mostra que foi capaz de absorver o conhecimento contido nos dados de teste e, ao mesmo tempo, capaz de generalizar esse conhecimento para a classificação de novos dados. Um exemplo de um modelo apropriado está ilustrado na Figura 3 (RASHKA e MIRJALILI, 2019).

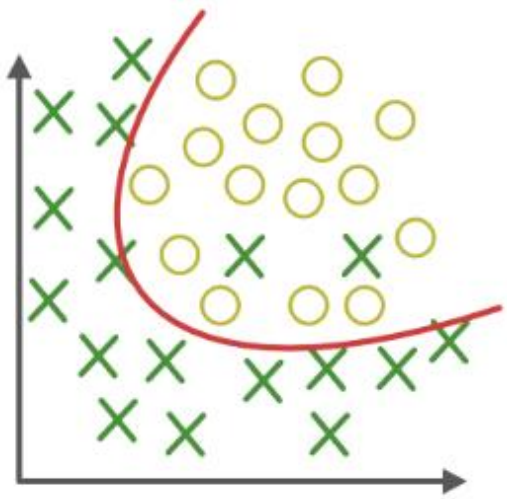

Figura 3 - Exemplo gráfico de modelo de classificação de ajuste de dados apropriada. Disponível em https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. Acesso em: 20 nov. 2022.

Há diversas formas de se melhorar a capacidade de generalização e reduzir o viés do modelo. A seleção de método de classificação ideal interfere na performance que será obtida. Além disso, espera-se que quanto mais dados disponíveis para o aprendizado menor será o viés de um modelo. E, por último, há técnicas que podem ser utilizadas para essa finalidade. Uma que merece ser mencionada é chamada de *k-fold cross validation*, que é uma técnica de reamostragem com ou sem reposição na qual se divide o conjunto de dados em k partes e realiza-se k iterações de treinamento e teste e em cada iteração altera-se o conjunto de treinamento e teste como é mostrado na Figura 4. Ao final do procedimento, calcula-se a média do desempenho obtido em cada iteração (FENNER, 2019; RASHKA e MIRJALILI, 2019).

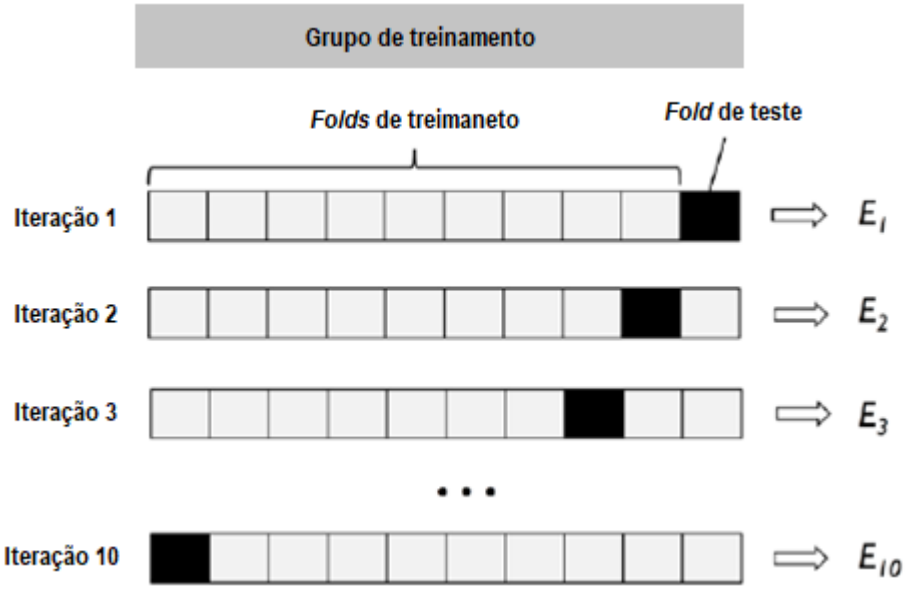

Figura 4 - Exemplo de aplicação da técnica *k-fold cross validation* em que o k é igual a 10 *(RASHKA e MIRJALILI, 2019,* traduzido*).*

Por fim, é importante ressaltar a importância de haver quantidades semelhantes de entidades rotuladas em cada categoria. Para o caso de um algoritmo de aprendizado de máquina supervisionado de classificação binária espera-se que haja aproximadamente metade dos dados rotulados de uma forma e a outra metade rotulados de outra forma. Isso deve ser verdadeiro também para classificações em que haja mais de duas classes. Quando há quantidades muito discrepantes de rótulos de classificação na base de dados diz que os dados estão desbalanceados e quando isso acontece há uma tendência de enviesamento para as categorias que possuírem a maior quantidade de entidades (RASHKA e MIRJALILI, 2019).

Para resolver esse problema, diversas técnicas podem ser aplicadas, como por exemplo a simples exclusão de dados de categorias dominantes de forma a equalizar a quantidade de dados de cada categoria. O problema dessa técnica reside no fato da perda de informação dos dados deletados o que, para bases de dados pequenas, pode resultar em perda de desempenho. Para contornar essa limitação, outra técnica de balanceamento de dados muito utilizada é a criação sintética de entidades para as classes minoritárias de forma a equalizar a quantidade de entidades em cada classe (RASHKA e MIRJALILI, 2019).

2.3.2 Principais Medidas de desempenho

Há diversas maneiras de avaliar o desempenho de aplicações que se utilizam do aprendizado de máquina, essas medidas de desempenho normalmente são utilizadas em conjunto para obtenção de um conhecimento mais amplo do resultado apresentado pelo sistema. Nessa sessão serão descritas a Matriz de Confusão, Acurácia, Precisão, Revocação, Pontução F1 e a Curva ROC e AUC.

2.3.2.1 Matriz de Confusão

A Matriz de Confusão consiste em uma matriz quadrada de duas linhas e duas colunas que contabiliza em cada uma de suas posições o número de Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN). Um VP ocorre quando o algoritmo classifica corretamente o rotulo de uma entidade como positivo, um VN ocorre quando esse algoritmo classifica corretamente uma entidade como negativo, um FP ocorre quando o algoritmo classifica incorretamente o rótulo de uma entidade como positivo e, por fim, um FN ocorre quando o algoritmo classifica incorretamente o rótulo de uma entidade como negativo (RASHKA e MIRJALILI, 2019).

Dessa forma, a matriz de confusão fornece informações para verificar o desempenho do modelo de classificação na classificação de cada rótulo separadamente (positivo e negativo, para o caso), dessa forma é possível verificar se há desequilíbrios nas falhas e acertos de classificação de cada rótulo. A Figura 5 evidencia o formato de uma Matriz de Confusão (CABRAL, 2021; RASHKA e MIRJALILI, 2019).

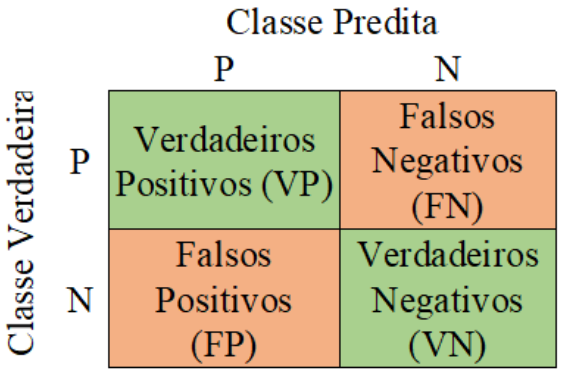

Figura 5 - Representação de uma Matriz de Confusão (imagem do autor)

2.3.2.2 Acurácia

A acurácia é a soma das previsões corretas (positivas ou negativas) dividida pelo número total de previsões realizadas (RASHKA e MIRJALILI, 2019).

A acurácia é calculada pela fórmula abaixo:

$$\text{Acurácia} = \frac{VP + VP}{VP + VN + FP + FN}$$

2.3.2.3 Precisão

A precisão mede o desempenho das classificações realizadas pelo algoritmo no âmbito das entidades que foram ou deveriam ter sido classificados como pertencentes à determinada classe. Ou seja, leva em consideração apenas os Verdadeiros Positivos ou os Falsos Positivos. Essa métrica é muito utilizada, por exemplo, para aplicações como o filtro de *spam*. Para esse tipo de aplicação, o envio equivocado de um e-mail para a caixa de *spam* é muito mais danoso do que a manutenção desse *spam* na caixa de entrada. Dessa forma, a presença de Falsos Positivos é menos desejada do que a de Falsos Negativos (CABRAL, 2021).

O cálculo da Precisão é dado pela fórmula abaixo:

$$Precisão = \frac{VP}{VP + VF}$$

2.3.2.4 Revocação

Diferentemente da Precisão, pode haver situações em que a presença de Falsos Negativos é indesejada, como é o caso de uso de algoritmos para a detecção de doenças. Para esses casos o cálculo da Revocação é muito relevante, pois considera a taxa de Verdadeiros Positivos dividida pelo somatório do número de Verdadeiros Positivos e o número de Falsos Negativos (CABRAL, 2021; RASHKA e MIRJALILI, 2019).

O cálculo da Revocação é dado pela fórmula abaixo:

$$Revocação = \frac{VP}{VP + FN}$$

2.3.2.5 Pontuação F1

Uma métrica utilizada para verificar o balanceamento dos lados negativos e positivos é a Pontuação F1, que utiliza a combinação da Revocação e da Precisão para o cálculo do seu valor. Usualmente o cálculo da Pontuação F1 é feita por meio da média harmônica da desses dois valores e é uma medida menos suscetível a distorções na presença de *outliers* (CABRAL, 2021; RASHKA e MIRJALILI, 2019).

O cálculo da Pontuação F1 pela média harmônica é dada pela fórmula abaixo:

$$F1 = 2 \times \frac{Revocação \times Precisão}{Revocação + Precisão}$$

2.3.2.6 Curva ROC e AUC

As Curvas ROC e AUC são ferramentas úteis para avaliar e comparar modelos de classificação binários em aprendizado de máquina utilizando as taxas de Falsos Positivos e de Verdadeiros Positivos obtidas. O nome ROC vem do inglês *Receiver Operating Characteristic* (ROC), que traduzindo significa "curva característica de operação" e o AUC vem da expressão *Area Under the Curve* (AUC), que significa "área abaixo da curva" (RASHKA e MIRJALILI, 2019).

Ao variar o limiar de decisão do classificador e computando as taxas de Verdadeiros Positivos e de Falsos Positivos para cada caso, traça-se a respectiva curva ROC. Em um classificador com performance perfeita não ocorreria Falsos Positivos e a taxa de Verdadeiros Positivos seria 1 (100%), por isso para esse caso hipotético a curva se situaria no canto superior esquerdo como evidencia a linha pontilhada preta da Figura 6. A diagonal do gráfico, ilustrada pela linha pontilhada cinza da Figura 6, pode ser interpretada como o desempenho de um classificador binário aleatório, e pode ser relacionada aos casos em que não houve aprendizado pelo modelo. Como a Curva ROC é muito utilizada para a comparação de diversos modelos diferentes e como a observação da curva traçada pode ser confusa e não evidenciar claramente qual foi o modelo com melhor desempenho, calcula-se a área embaixo da curva. Quanto mais próximo de 1 for este valor, melhor o desempenho do modelo. A Figura 6 ilustra a Curva ROC de três modelos distintos e a área abaixo da curva de cada uma é evidenciada na legenda (RASHKA e MIRJALILI, 2019).

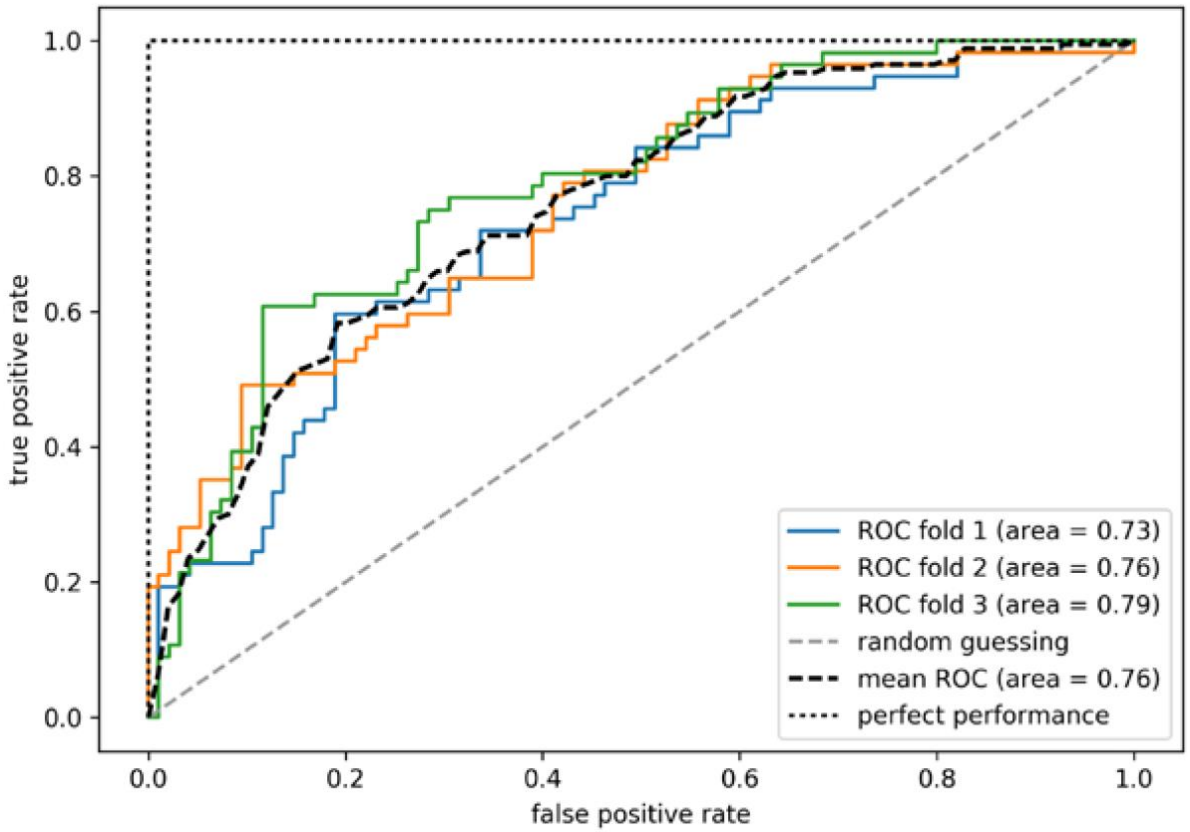

Figura 6 - Exemplo de Curva ROC e AUC. Fonte: *(RASHKA e MIRJALILI, 2019)*.

2.4 Descrição da base de dados

No trabalho "Direito e Inteligência Artificial: Análise de Sentimento Aplicada em Certidões de Julgamento de Mandados de Segurança Impetrados no Supremo Tribunal Federal" de Guilherme Ramos de Morais, com coautoria do Professor Doutor Henrique Araújo Costa, foram analisados 2.158 mandados de segurança, assim como os recursos conexos a essas ações, impetrados no Supremo Tribunal Federal. Os dados foram coletados pelo grupo de pesquisa Direito e Tecnologia da Universidade de Brasília (DireitoTec) e a base de dados contém informações de certidões de julgamento como data de julgamento, órgão responsável pelo julgamento, relator, número do processo, a Unidade da Federação de origem, a turma responsável e outras informações (MORAIS, 2019).

A próxima etapa daquele trabalho consistiu na atribuição de rótulos a cada certidão de julgamento, ou seja, atribuir uma classificação de sentimento positiva no caso de procedência do objeto da demanda ou uma classificação negativa no caso de improcedência. No entanto, podem ocorrer dois tipos de julgamentos para cada demanda apresentada ao STF: julgamento

preliminar ou julgamento de mérito. Cada tipo de julgamento gera uma sentença que deve ser atribuída uma classificação de sentimento cada uma. Por fim, os sentimentos de cada sentença são considerados conjuntamente, através de critérios lógicos, para se chegar ao sentimento final da certidão de julgamento (MORAIS, 2019).

Conforme é explicado por Morais (2019), adotou-se o método léxico para a aplicação da análise de sentimentos. Esse método identifica, nas decisões de cada juízo, as expressões e palavras relevantes para a análise de sentimento e associa a cada uma delas um sentimento. Uma vez realizada a análise de cada juízo, utilizou-se regras de classificação para se obter o sentimento final. Dessa maneira, foram abarcadas todas as granularidades em sequência. Primeiramente o sentimento das palavras principais são analisadas (granularidade de aspecto ou entidade); em seguida utiliza-se o sentimento de cada uma das palavras para se chegar a um sentimento resultante, que é vinculado a um juízo (granularidade de sentença); e finalmente as sentenças de cada juízo são analisadas em conjunto para se obter o sentimento final (granularidade de documento) da certidão de julgamento (MORAIS, 2019).

Para o juízo preliminar, que verifica os requisitos de procedência da ação e do conhecimento do recurso, foi criada uma coluna denominada "preliminar_dispositivo*"*, que registra a expressão literal utilizada para anunciar o seu resultado na certidão de julgamento preliminar. Da mesma forma, para o juízo de mérito, que trata de analisar o mérito da ação, foi criada a coluna "mérito_dispositivo*"* para registrar a expressão literal utilizada na certidão de julgamento de mérito. Para viabilizar o aprendizado de máquina é importante registrar a expressão literal utilizada uma vez que é atribuído um sentimento à cada uma das expressões registradas e a mudança de um caractere sequer pode impedir que a máquina reconheça o respectivo sentimento (MORAIS, 2019).

Em seguida, para cada uma dessas expressões, e para cada um dos juízos, foi atribuído de forma manual um sentimento. O sentimento "positivo" foi atribuído ao dispositivo preliminar das certidões em que o Mandado de Segurança ou recurso a ele conexo, fora admitido ou conhecido. De maneira diversa, um sentimento "negativo" foi atribuído aos dispositivos preliminares que não admitiam o Mandado de Segurança ou não conheciam o recurso a ele conexa. O sentimento do juízo preliminar foi armazenado na coluna "preliminar_sentimento*"* (MORAIS, 2019).

Já para o juízo de mérito foi atribuído sentimento “positivo” ao dispositivo em que houve procedência do pedido do Mandado de Segurança ou de um recurso do mesmo e sentimento “negativo” no caso de improcedência (MORAIS, 2019).

Por fim, criou-se uma coluna para registrar a relação entre os sentimentos dos dispositivos dos dois juízos, denominada “final_sentimento*”*. Para definir qual seria o sentimento final da certidão foram definidas duas regras. A primeira delas é que se o sentimento atribuído à certidão do juízo de mérito for positivo, o sentimento final é positivo, pois pode-se inferir que o juízo preliminar também é positivo. Já a segunda regra diz que se o sentimento atribuído ao juízo preliminar for negativo, esse juízo vai ser o utilizado para obter o sentimento final da decisão, que também será negativo, pois se não reconhecida a procedência em um juízo preliminar, não há que se falar em juízo de mérito (MORAIS, 2019).

De forma a aumentar a quantidade de informações que podem ser utilizadas por algoritmos de aprendizado de máquina, nessa base de dados foi também acrescentada uma coluna para descrever se a decisão tomada pelo juízo foi de forma unânime ou por maioria. Como são tomadas decisões tanto no juízo preliminar como no de mérito, duas colunas foram adicionadas para armazenar essa informação: “preliminar_modo*”* e “mérito_modo*”*. Além disso, também foi criado uma coluna denominada “mérito_abrangência*”* para registrar se a ação foi admitida ou rejeitada em sua totalidade ou apenas parcialmente (MORAIS, 2019).

A Tabela 1 ilustra o modelo de análise do trabalho de Morais (2019), que descreve os atributos criados e acrescentados à base de dados para se chegar a um sentimento final das certidões de julgamento.

Tabela 1 – Modelo de Análise retirada do trabalho de Morais (2019).

preliminar_modo	preliminar_dispositivo	mérito_modo	mérito_dispositivo	mérito_abrangência
unânime	conheceu	unânime	denegou a sentença	total
preliminar_sentimento		mérito_sentimento	final_sentimento	
positivo		negativo	negativo	

O resultado do trabalho utilizando a metodologia descrita foi a classificação de 1.644 das 2.158 certidões de julgamento, que foram coletadas pelo grupo de pesquisa Direito e Tecnologia da Universidade de Brasília (DireitoTec). A Tabela 2 evidencia o resultado da rotulação.

Tabela 2 - Resultado da rotulação recuperado de MORAIS (2019, p. 69).

Sentimento	Negativo	Positivo	Não Rotulado
Preliminar	131	128	1385
Mérito	1257	256	131
Final	1388	256	514

Conforme é explicado por Morais (2019), o campo “Não Rotulado” descreve os casos em que não há expressão para ser rotulada, como é o caso de negativa no juízo preliminar ou existência de julgamento de mérito com ausência de juízo preliminar. O autor também assegura que a ausência desses rótulos não compromete a análise final.

Dos 514 casos em que não foi atribuído um sentimento final à certidão, 492 se justificam por erro do programa responsável pelo *download* dos dados e os outros 22 tratam de classificações a serem finalizadas. Dessa forma, dos 1.666 casos, apenas 22 não puderam ser analisados (MORAIS, 2019).

Ao final do trabalho de Morais (2019), são evidenciados diagramas de Sankey, que descrevem de maneira visual a composição da base dados e o resultado da rotulação manual realizada. O primeiro deles, evidenciado pela Figura 7, situa o sentimento do juízo preliminar na primeira coluna a esquerda, o sentimento do juízo de mérito na coluna do meio e o sentimento final na coluna mais à direita, explicitando a proporção de casos de rótulos de sentimento em cada uma das colunas, ressaltando que o número “0” representa os casos em que não houve rotulação no juízo preliminar.

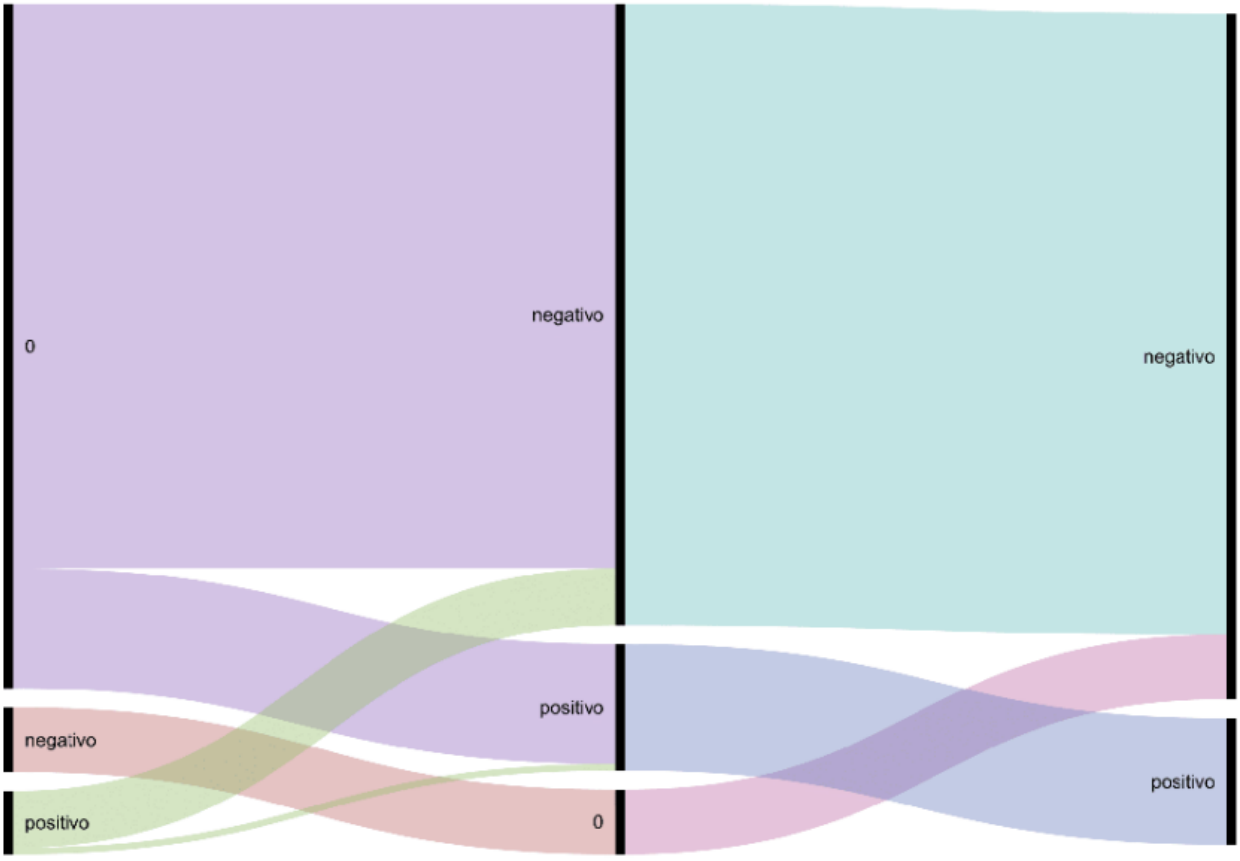

Figura 7 - Distribuição proporcional da classificação dos sentimentos preliminar, de mérito e final (MORAIS, 2019).

A Figura 8 evidencia as expressões literais retiradas das certidões de julgamento do juízo preliminar classificadas com sentimento positivo, quais expressões literais evidenciadas nessas certidões e como essas expressões foram utilizadas para definir se a decisão foi declarada por maioria ou de maneira unânime. A Figura 9 descreve a mesma situação, porém para o caso das certidões de julgamento do juízo preliminar classificadas com sentimento negativo.

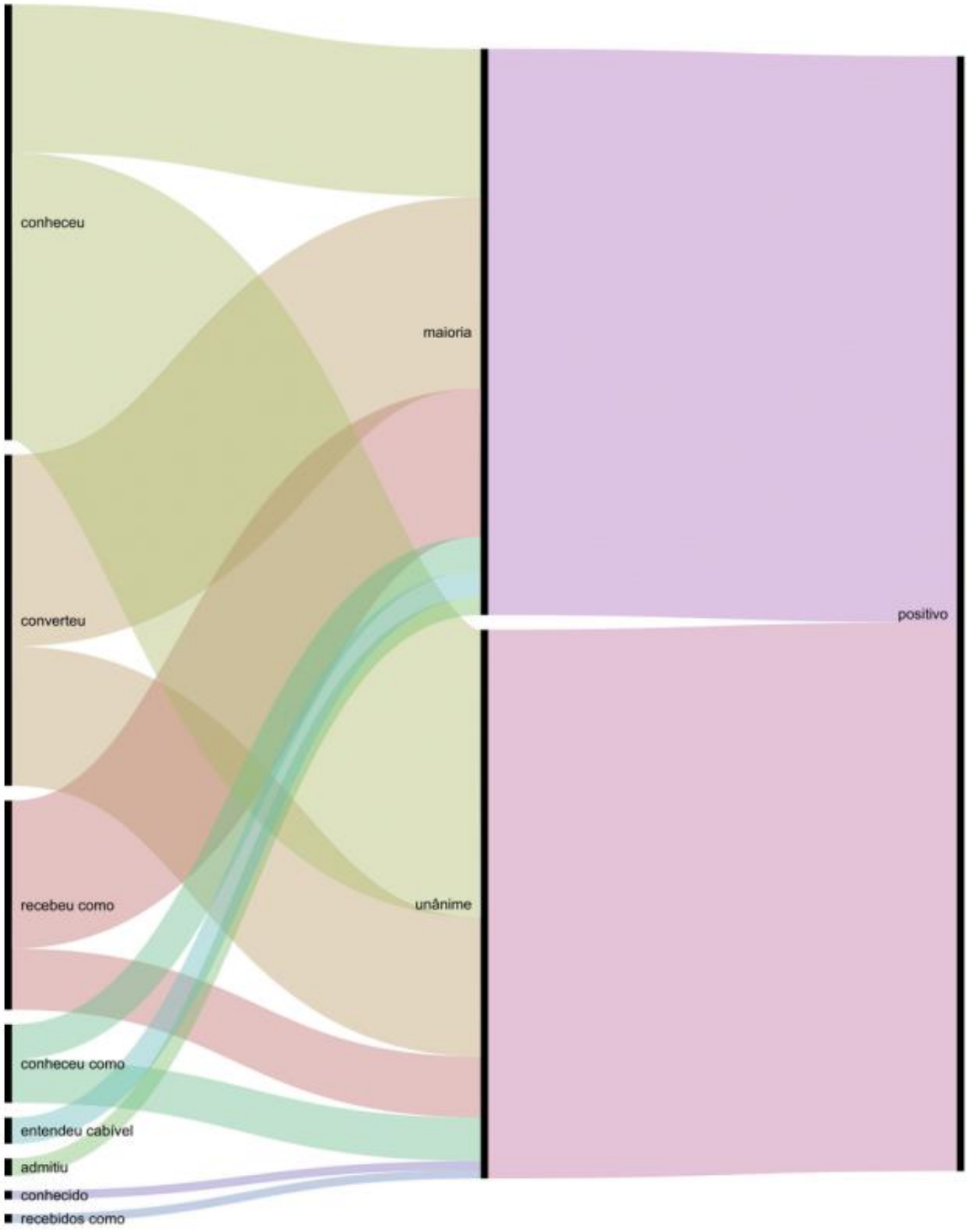

Figura 8 - Distribuição proporcional das expressões do juízo preliminar com sentimento positivo (MORAIS, 2019).

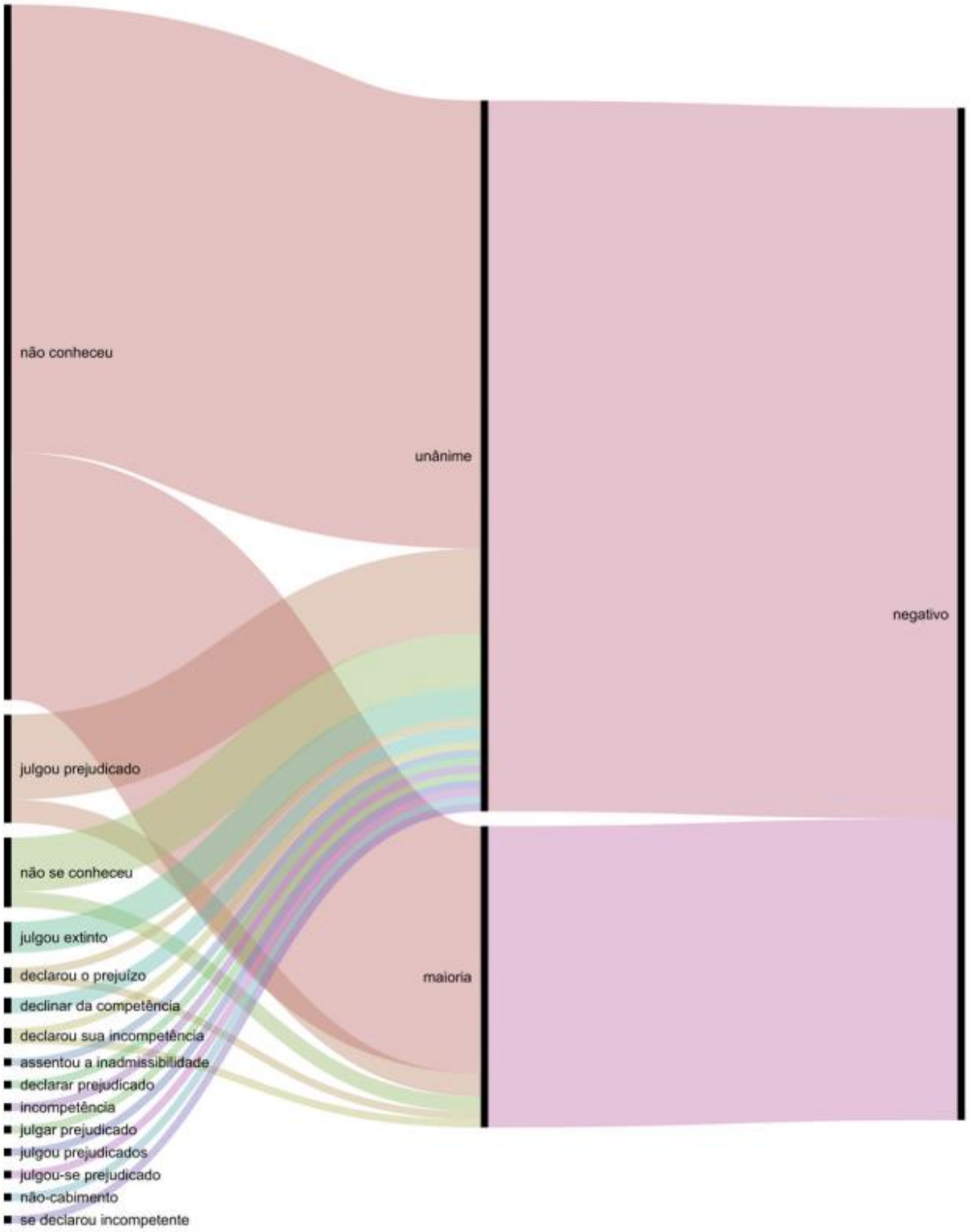

Figura 9 - Distribuição proporcional das expressões do juízo preliminar com sentimento negativo (MORAIS, 2019).

A Figura 10 evidencia as expressões literais retiradas das certidões de julgamento do juízo de mérito classificadas com sentimento positivo, quais expressões literais evidenciadas nessas certidões e como essas expressões foram utilizadas para definir se a decisão foi declarada

por maioria ou de maneira unânime. Finalmente, a Figura 11 descreve a mesma situação, porém para o caso das certidões de julgamento do juízo preliminar classificadas com sentimento negativo.

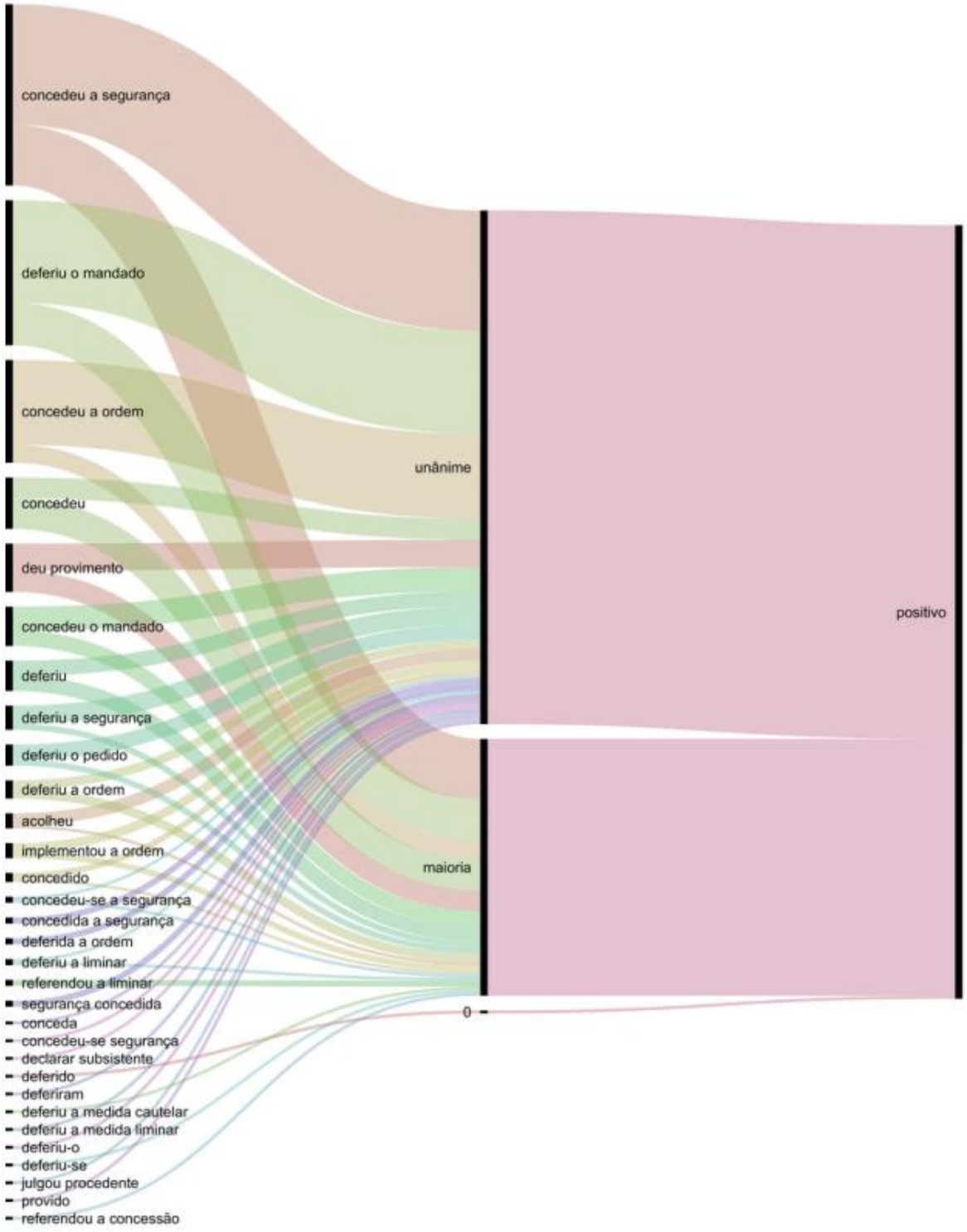

Figura 10 - Distribuição proporcional das expressões do juízo de mérito com sentimento positivo *(MORAIS, 2019)*.

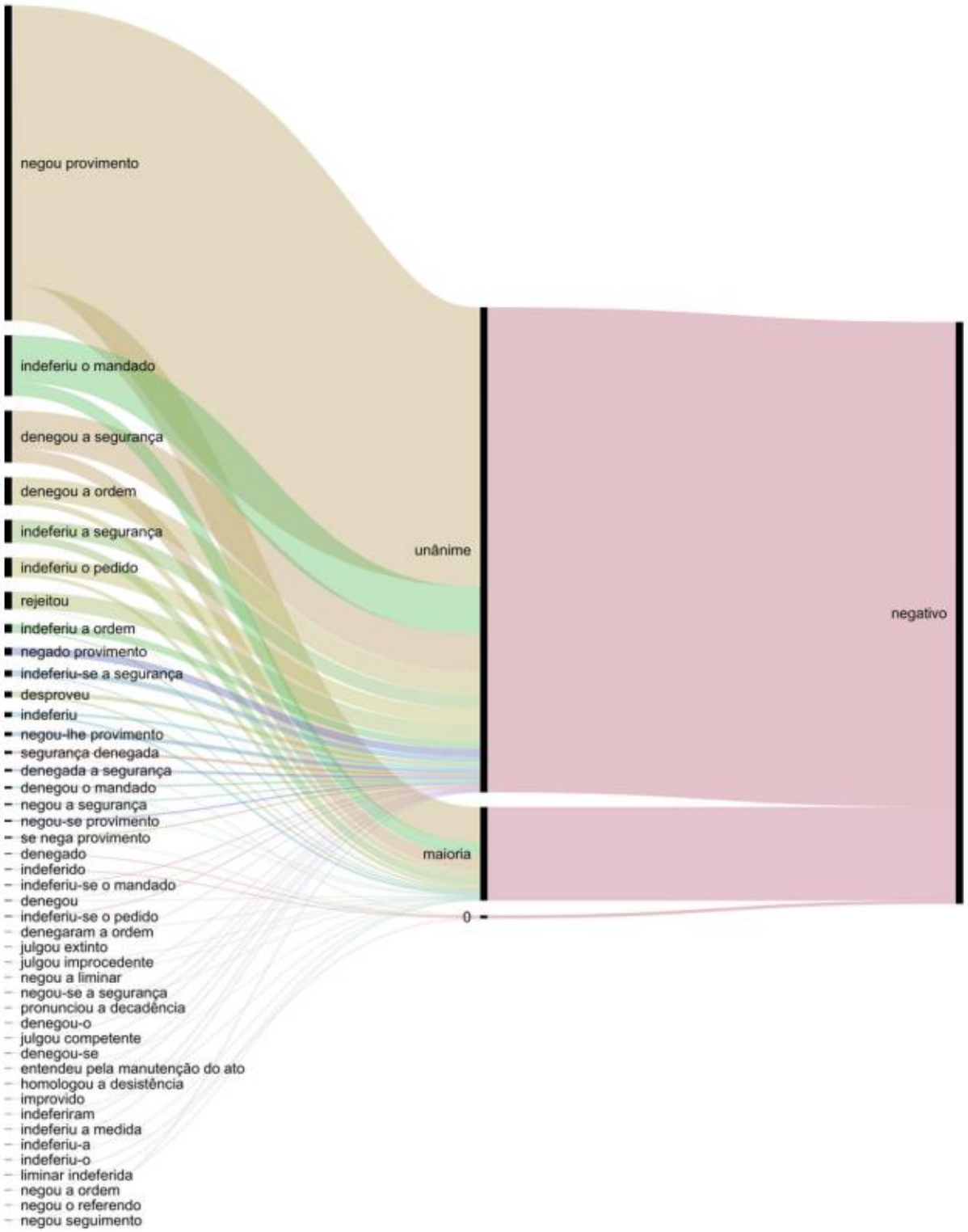

Figura 11 - Distribuição proporcional das expressões do juízo de mérito com sentimento negativo (MORAIS, 2019).

3 APLICAÇÃO PARA CLASSIFICAÇÃO AUTÔNOMA DE CERTIDÕES DE JULGAMENTO DO STF

3.1 Considerações Iniciais

Este capítulo apresenta primeiramente a metodologia utilizada para o desenvolvimento deste trabalho (seção 3.2) e em seguida descreve uma aplicação para classificar binariamente e de forma autônoma certidões de julgamento de mandados de segurança apresentados ao STF (seção 3.3).

3.2 Metodologia

Por tratar-se de uma monografia para conclusão de um curso de graduação, este trabalho se limitará em descrever as etapas para a criação de uma aplicação que, através do aprendizado de máquina, prediz de forma autônoma o resultado de um julgamento de mandado de segurança impetrado no STF, recebendo informações de certidões de julgamento como dados de alimentação (*input*).

Para alimentar o algoritmo de aprendizado de máquina que será descrito, serão utilizados os dados gerados pelo trabalho de Morais (2019), que foram descritos na Sessão 2.4. Dessa forma, antes de descrever o algoritmo da aplicação será apresentado uma proposta de como deve ser realizado o tratamento de dados preliminar para alimentar o algoritmo, levando em consideração quais são as informações que são efetivamente relevantes para o aprendizado de máquina.

Por fim serão enunciados, os códigos de programação necessários para a realização dos passos descritos. Para isso, foi escolhida a linguagem *Python*.

3.3 Descrição da aplicação

3.3.1 Considerações Iniciais

Esta seção será dividida em três etapas: seleção dos atributos relevantes para utilização na aplicação (subseção 3.3.2); de limpeza dos dados (subseção 3.3.3); pré-processamento dos dados (subseção 3.3.4); balanceamento dos dados (subseção 3.3.5); aprendizado de máquina

(subseção 3.3.6); teste e avaliação dos modelos (subseção 3.3.7); e, por fim, a aplicação dos modelos treinados para a classificação autônoma de certidões de julgamento (subseção 3.3.8).

3.3.2 Dados Relevantes para a Aplicação

Os dados coletados pelo grupo de pesquisa Direito e Tecnologia da Universidade de Brasília (DireitoTec), que foram posteriormente complementados por Morais (2019), contém 2.158 linhas e 17 colunas. A Figura 12 mostra as 10 primeiras linhas das as colunas desses dados.

#	1	2	3	4	5	6	7	8	9	10	preliminar_modo	preliminar_dispositivo	mérito_modo	mérito_dispositivo	mérito_abrangência	mérito_sentimento
1	1.0	2.9474	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
2	2.0	30.8880	DF	Mandado de Segurança	Min. Teori Zavascki	7/10/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-227 DIVULG 24-10-2016...	Ementa: AGRAVO REGIMENTAL EM MANDADO DE SEGURA...	A Turma, por unanimidade, negou provimento ao ...	NaN	NaN	unânime	negou provimento	NaN	negativo
3	3.0	29.0130	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
4	4.0	29.0600	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
5	5.0	29.0630	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
6	6.0	29.0640	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
7	7.0	29.0660	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
8	8.0	29.0700	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
9	9.0	29.0710	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	PROCESSO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo
10	14.0	28.9570	DF	Mandado de Segurança	Min. Teori Zavascki	2/12/2016	Segunda Turma	ACÓRDÃO ELETRÔNICO\nDJe-266 DIVULG 14-12-2016 ...	Ementa: PROCESSUAL CIVIL. EMBARGOS DE DECLARAÇ...	A Turma, por unanimidade, rejeitou os embargos...	NaN	NaN	unânime	rejeitou	NaN	negativo

Figura 12 - Dez primeiras linhas contidas na base de dados de Morais (2019).

Como é possível visualizar, cada coluna representa uma informação a respeito de um determinado processo de mandado de segurança. A Tabela 3 evidencia as informações que cada uma representa. Ressalta-se que a presença de duas colunas de identificador único para cada linha desses dados não é necessária e foi disponibilizada na base de dados de maneira redundante.

Tabela 3 - Informações de cada coluna da base de dados de Morais (2019).

1	Identificador único de cada linha de processo contido nesses dados.
2	Identificador único de cada linha de processo contido nesses dados (redundante).
3	Número do processo.
4	Estado da Federação.
5	Tipo de ação protocolada.
6	Relator do processo.
7	Data de Início do Processo.
8	Turma Responsável.
9	Informações sobre o Processo.
10	Ementa.
11	Dispositivo.
12	Modo que foi decidido o julgamento preliminar.
13	Vocábulo utilizado na ementa para caracterizar a decisão do julgamento preliminar.
14	Modo que foi decidido o julgamento de mérito.
15	Vocábulo utilizado na ementa para caracterizar a decisão do julgamento de mérito.
16	Abrangência da decisão.
17	Sentimento.

Como a finalidade da aplicação prever o resultado da decisão, é recomendável filtrar apenas as informações que contribuem para esse objetivo. Dessa maneira as colunas número do processo, Estado da federação, relator do processo, data de início do processo, turma responsável, modo que foi decidido o julgamento preliminar e modo que foi decidido o julgamento de mérito não serão consideradas. Em trabalhos futuros é possível verificar, por métodos estatísticos, se há correlação entre essas informações e o resultado da decisão, mas para os fins deste trabalho será considerado que essa correlação é inexistente.

Ademais, esta aplicação leva em consideração apenas processos de mandados de segurança do STF e a base de dados contém apenas esse tipo de ação, esse atributo torna-se igualmente irrelevante e serão desconsideradas. Da mesma forma, em trabalhos futuros, é possível o desenvolvimento de aplicações que analisam as informações de certidão de julgamentos de diferentes tipos de ações, ocasião na qual a informação dessa coluna se tornaria relevante.

Por fim, pelo fato de a aplicação receber como *input* apenas o texto do dispositivo da certidão de julgamento, será desconsiderado também as colunas das informações sobre o processo, da ementa, do vocábulo utilizado para caracterizar as decisões de julgamento (preliminar e de mérito) e da abrangência da decisão.

Dessa forma, as três colunas que serão utilizadas para treinar o algoritmo de classificação serão: a primeira coluna do identificador único de cada linha (apenas para fins de identificação); a do texto do dispositivo; e o sentimento atribuído por Morais (2019).

3.3.3 Limpeza de Dados

Após a retirada das colunas que não serão utilizadas para o desenvolvimento da aplicação, verifica-se as informações contidas nas linhas. Para isso, compara-se os valores presentes nas linhas com o que se espera encontrar, verificando a presença de inconsistências. Por exemplo, para a coluna do identificador único de cada linha, espera-se encontrar um número natural que identifica a posição de determinada linha em toda a tabela (para o caso da base de dados, valores variando de 0 a 2.157, uma vez que a contagem se inicia com o valor 0). Para a coluna dos textos do dispositivo da certidão de julgamento, espera-se encontrar dados no formato textual. E, finalmente para a coluna do sentimento, espera-se encontrar apenas dois valores textuais diferentes: “positivo” ou “negativo”.

Normalmente para a coluna de identificação única não são encontradas inconsistências, uma vez que os comandos responsáveis por atribuir uma numeração para a posição de cada linha realizam esse procedimento de forma automática.

Em certas situações, o programa utilizado para coletar as informações de mandados de segurança do STF, por diversos motivos, não é capaz de coletar a informação. Isso resulta de um valor nulo na base de dados. Para o caso da base de dados de Morais (2019) o valor nulo é representado pelo valor “*NaN*”. Como a informação presente em todas as colunas é de fundamental importância para o aprendizado de máquina, os valores “*NaN*” das colunas de certidão de julgamento e do sentimento devem ser removidos.

Por se tratar de uma aplicação para classificar de forma binária o texto contido no dispositivo de uma certidão de julgamento, o algoritmo de treinamento esperará apenas dois valores diferentes na coluna do sentimento no momento que estiver processando os dados. Dessa forma, deve-se verificar se há apenas esses dois valores nessa coluna. Após a execução do comando *Python* “*value_counts*” é possível verificar a quantidade de ocorrências de todos os valores únicos da coluna, obtendo-se o resultado evidenciado pela Figura 13.

negativo	1257
xxx ERRO xxx	492
positivo	256
NaN	131
xxx PENDENTE xxx	22

Figura 13 - Valores únicos e suas quantidades da coluna "*merito_sentimento*" na base de dados de Morais (2019).

Dessa forma, devem ser excluídas todas as linhas que contenham, na coluna "mérito_sentimento" valores diferentes de "negativo" ou "positivo", que no caso da base de dados fornecida são os valores "xxx ERRO xxx", "NaN" e "xxx PENDENTE xxx".

Após a realização do processo de limpeza de dados descrita acima, a quantidade de linhas presentes nos dados reduziu de 2.158 para 1.513.

3.3.4 Pré-processamento

Após a remoção de colunas que não contribuem com informações relevantes, é possível remover informações irrelevantes contidas nas colunas que restaram. A única coluna possível de ser submetida a esse procedimento é a coluna do texto do dispositivo da certidão uma vez que a demais contém apenas um vocábulo, essencial para caracterizar a decisão do processo em questão.

Para essa aplicação será gerada uma nova coluna de texto de certidão submetida a cada técnica de pré-processamento para, posteriormente, comparar os resultados obtidos por cada um juntamente com os resultados obtidos com a utilização do texto bruto do dispositivo da certidão de julgamento.

Todos os caracteres do texto de todas as certidões de julgamento serão convertidos em minúsculas (caso já não estejam nesse formato) e será gerada uma coluna com o texto de cada certidão de julgamento. Esse resultado é obtido utilizando a função "*lower()*" no texto de cada dispositivo da certidão de julgamento.

Da mesma forma, será gerada uma terceira coluna apenas com o radical de cada palavra do texto dos dispositivos das certidões de julgamento utilizando a técnica de *stemização*. O

comando "*stemmer.stem(texto)"* pode realizar essa conversão. Trata-se de função da biblioteca *NLTK*, que disponibiliza diversas ferramentas para o PLN.

O último pré-processamento a ser utilizado nessa aplicação é a remoção das *stopwords*. Para isso, deve-se primeiramente definir quais são essas palavras. Para essa aplicação, optou-se por definir artigos, conjunções e preposições como *stopwords*. O comando para realizar a remoção dessas palavras é evidenciado na etapa de aprendizado de máquina, uma vez que essa opção está embutida em um dos comandos utilizados nessa etapa. Por esse motivo, também, não será necessário a geração de uma nova coluna para armazenamento do texto sem as *stopwords*.

Dessa maneira, nesta etapa serão geradas duas novas colunas, além do texto sem modificações: texto em minúsculas; e texto "estemizado". As três colunas serão igualmente submetidas aos procedimentos descritos a seguir, dessa forma será possível verificar se algum dos pré-processamentos impacta de maneira positiva nas métricas de resultado a serem obtidas.

3.3.5 Balanceamento de dados

Conforme verificado no Capítulo 3.3.3, há uma quantidade de entidades classificadas com sentimento negativo (1.257 linhas) muito superior em comparação aos casos rotulados como positivos (256 linhas). Conforme mencionado no Capítulo 2.3.1, esse desbalanceamento dos dados poderá causar um enviesamento do algoritmo, tendendo-o a atribuir um rótulo "negativo" com muito mais frequência.

Para resolver esse problema existem diversas alternativas, tais como a criação de novas linhas com rótulo "positivo" ou a exclusão de linhas com rótulo "negativo". Como há um número razoável de linhas classificadas positivamente, é possível optar por excluir 1.001 linhas rotuladas negativamente para igualar a quantidade de entidades existentes em cada classificação o que resultaria em uma base de dados com 512 linhas.

Outra solução seria a ampliação de casos classificados positivamente, pela criação de novas entidades classificadas. No entanto, como trata-se de dados textuais relativamente extensos, a criação manual de novas certidões de julgamento seria um processo trabalhoso, além de enviesar os dados com a percepção do indivíduo que os criou. É possível também realizar a cópia de algumas certidões de julgamento classificadas como "positivas", porém esse

procedimento não acrescenta mais informações para o aprendizado de máquina, além de possibilitar, também, um enviesamento direcionado para essas cópias.

Dessa forma, para ampliar a quantidade de entidades classificadas como “positiva” o processo mais recomendável é a coleta e classificação de novos dados. Como coleta de dados não é o escopo deste trabalho, e 256 entidades presentes em cada classe (positiva e negativa) é um número aceitável para o desenvolvimento da aplicação, optou-se por selecionar por meio de amostragem 256 linhas classificadas com sentimento “negativo” e desconsiderar as demais. Para isso, filtra-se as entidades com sentimento negativo, e colhe-se uma amostra de 256 linhas através do comando “*sample(256)*” da biblioteca Pandas. Por fim, junta-se as 256 linhas classificadas com sentimento “positivo” com essa amostra, totalizando uma base de dados com 512 linhas (256 classificações “positivas” e 256 “negativas”).

3.3.6 Aprendizado de máquina

Uma vez que os dados foram limpos, pré-processados e balanceados, é possível utilizá-los para treinar um algoritmo para classificar de forma autônoma uma certidão de julgamento utilizando um classificador. O primeiro passo é dividir os dados disponíveis entre dados de teste e dados de treinamento. É possível realizar essa tarefa de maneira simples e direta ao se definir qual será a porcentagem de dados que será inserida em cada grupo, porém para melhorar a capacidade de generalização do algoritmo pode-se utilizar a técnica *k-fold cross validation* descrita no Capítulo 2.3.1, com o K representando o número de grupos a serem criados. Optou-se em realizar a divisão em 10 grupos, com isso 90% dos dados de cada um dos grupos são direcionados para o treinamento e os 10% restantes para teste.

Na linguagem *Python* técnica do *k-fold cross validation* é executada pelo comando “*Kfold*” da biblioteca *Sklearn*, que é muito utilizada para aplicações de Aprendizado de Máquina e modelagem estatística. A Figura 14 evidencia o código na linguagem *Python* para execução dessa técnica.

```
from sklearn.model_selection import KFold

# Divisão dos dados em 10 "folds"
kf = KFold(n_splits=10,shuffle=True,random_state=0)

for train_index, test_index in kf.split(dados):
    ########### Divisão Treinamento/Teste ##############
    X_train, X_test = P[train_index], P[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Figura 14 - Comando para aplicação do *k-fold* para divisão dos dados em 10 grupos para treinamento e teste na linguagem *Python*.

Após a divisão dos dados para treinamento e teste e aplicação da validação cruzada, aplica-se um vetorizador, que transformará os dados textuais das certidões de julgamento em dados numéricos (vetores ou matrizes) para que sejam processados pela máquina. Para essa aplicação será descrito o uso do vetorizador *TD-IDF* da biblioteca *Sklearn* (*TfidfVectorizer*), no entanto em outros trabalhos é possível a utilização de outros vetorizadores como o "*CountVectorizer*" e o "*HashingVectorizer*" para comparação de desempenho obtido por cada um.

Uma vez selecionado o vetorizador, pode-se definir o âmbito de análise para geração de *tokens* e formação dos *N-grams*. Na biblioteca *Sklearn* há três possíveis opções: em nível de palavras (padrão); em nível de caracteres; e sem levar em consideração os espaços em branco para a formação dos *N-grams*. Além disso, pode-se definir o âmbito de variação dos tamanhos dos *tokens* a serem considerados, uma vez que no valor padrão do comando utilizado para realizar a vetorização é definido que um *token* é formado apenas com *N-grams* com N igual a 1. Para essa aplicação utilizar-se-á dois tipos de formatos: *tokens* formados apenas por *1-grams*; e *tokens* formados por *1-grams, 2,-grams* e *3-grams*.

Além disso, conforme já mencionado na etapa de pré-processamento, no próprio comando para aplicar a vetorização ao dado textual, é possível definir se serão removidas as *stopwords*, previamente definidas. Para fins de comparação de desempenho será evidenciado o uso do texto em dois formatos: sem alteração; e com remoção das *stopwords*.

A Figura 15 evidencia os comandos em *Python* para a definição de cada uma das 12 formas acima descritas para vetorizar os textos da certidão de julgamento. Posteriormente, esses comandos devem ser inseridos em uma estrutura de repetição para serem executados de forma sucessiva.

```
## Diferentes modelos de aplicação do vetorizador ##
# Sem Stopword e N-grams de 1 token
vetorizador = TfidfVectorizer(analyzer='char_wb')
vetorizador = TfidfVectorizer(analyzer='char')
vetorizador = TfidfVectorizer(analyzer='word')
# Com Stopword e N-grams de 1 token
vetorizador = TfidfVectorizer(analyzer='char_wb',stop_words=stop)
vetorizador = TfidfVectorizer(analyzer='char',stop_words=stop)
vetorizador = TfidfVectorizer(analyzer='word',stop_words=stop)
# Sem Stopword e N-grams de 1 a 3 tokens
vetorizador = TfidfVectorizer(analyzer='char_wb',ngram_range=(1,3))
vetorizador = TfidfVectorizer(analyzer='char',ngram_range=(1,3))
vetorizador = TfidfVectorizer(analyzer='word',ngram_range=(1,3))
# Com Stopword e N-grams de 1 a 3 tokens
vetorizador = TfidfVectorizer(analyzer='char_wb', stop_words=stop,ngram_range=(1,3))
vetorizador = TfidfVectorizer(analyzer='char', stop_words=stop,ngram_range=(1,3))
vetorizador = TfidfVectorizer(analyzer='word', stop_words=stop,ngram_range=(1,3))
```

Figura 15 - Modelos de vetorização dos textos das certidões de julgamento.

Cabe ressaltar, que não é necessário aplicar um vetorizador na coluna que descreve o sentimento de cada certidão de julgamento, uma vez que, por se tratar de apenas duas classes, será atribuído o valor 0 ou 1 a depender de qual categoria a certidão estiver inserida.

Para executar a transformação nos dados de treinamento e teste divididos na etapa anterior, executa-se os comandos evidenciados na Figura 16. As variáveis "*treinamento_vetor*" e "*teste_vetor*" foram criadas para posteriormente serem utilizadas na aplicação do classificador.

```
##### Transformando Texto em Vetor
treinamento_vetor = vetorizador.fit_transform(X_train)
teste_vetor = transformador.transform(X_test)
```

Figura 16 - Comandos em Python para aplicação do vetorizador nos dados textuais das certidões de julgamento.

Uma vez vetorizados os dados, pode-se prosseguir para o treinamento dos dados, que é realizado baseado em um classificador. Há diversos classificadores disponíveis para efetuar predições binárias e, a depender da aplicação, alguns podem ser mais adequados que outros. Uma vez que a execução do algoritmo é automatizada, foram selecionados 17 classificadores binários a serem utilizados para verificar o desempenho de cada um. A Tabela 4 evidencia esses classificadores, as suas respectivas bibliotecas e os comandos em Python para executá-los. Ressalta-se que cada um desses classificadores, possuem parâmetros que podem ser ajustados para melhor se adequarem à situação em questão.

Tabela 4 - Classificadores binários, suas bibliotecas e o comando em Python para executá-lo.

Classificadores	Biblioteca	Comando em *Python*
NuSVC	Sklearn	NuSVC()
SVC		SVC()
MultinomialNB		MultinomialNB()
BernoulliNB		BernoulliNB()
ComplementNB		ComplementNB()
RandomForestClassifier		RandomForestClassifier()
ExtraTreesClassifier		ExtraTreesClassifier()
BaggingClassifier		BaggingClassifier()
GradientBoostingClassifier		GradientBoostingClassifier()
AdaBoostClassifier		AdaBoostClassifier()
ExtraTreeClassifier		ExtraTreeClassifier()
DecisionTreeClassifier		DecisionTreeClassifier()
MLPClassifier		MLPClassifier()
KNeighborsClassifier		KNeighborsClassifier()
LogisticRegression		LogisticRegression()
SGDClassifier		SGDClassifier()
XGBClassifier	Xgboost	XGBClassifier()

A execução do treinamento é feita pelo comando evidenciado na Figura 17, sendo que a palavra “classificador” se refere aos comandos da Tabela 4. Uma vez executado esse comando, a fase de treinamento estará encerrada e o modelo estará pronto para ser testado.

```
##### Treinando o classificador
classificador.fit(treinamento_vetor,y_train)
```

Figura 17 - Comando em Python para executar o treinamento do algoritmo.

Esse procedimento é repetido para cada coluna gerada na etapa de pré-processamento para o fim de verificação de qual modificação no texto produziu melhores resultados. Ao total poderão ser comparados 612 modelos diferentes ao se combinar os 3 tipos de pré-processamentos, as 12 modalidades de vetorização e os 17 classificadores.

Vale acrescentar que há a possibilidade de utilizar mais de um classificador para retornar apenas uma classificação, com o fim de buscar melhor desempenho. Nesse caso, seria possível, por exemplo, selecionar os classificadores que obtiveram o melhor resultado, armazenar as classificações geradas por esses classificadores para cada dado textual de certidão de julgamento e verificar o maior número de ocorrências das classificações para retornar um resultado final de classificação.

3.3.7 Teste e Avaliação dos modelos

A avaliação dos modelos neste trabalho consistiu em calcular a acurácia, a precisão, a revocação, a pontuação F1, o traçado da curva ROC e a área abaixo da curva (AUC). A Figura ***18*** mostra os códigos para o cálculo dessas medidas de desempenho e a Figura 19 os códigos para o traçado da curva ROC. Essas medidas de desempenho são calculadas para cada um dos 612 modelos e através delas é possível verificar qual apresentou o melhor desempenho.

```
##### Avaliação do Modelo
y_score = classificador.fit(treinamento_vetor, y_train).predict_proba(teste_vetor)
# Testando o modelo aprendido nos dados de treinamento
predicao = classificador.predict(teste_vetor)
#Acurácia
acuracia = accuracy_score(y_test,predicao)
## Revocação, Pontuação F1, Precisão
relatorio = classification_report(y_test,predicao)
# Área abaixo da curva ROC
auc = roc_auc_score(y_test, y_score[:,1])
```

Figura 18 - Códigos na linguagem Python para cálculo das medidas de desempenho.

```
fpr, tpr, thresholds = roc_curve(y_test, y_score[:, 1], pos_label=1)
plt.plot(fpr, tpr, color=lista_cores[i], lw=lw, label= "Classificador" + ' (area = %0.4f)' % roc_auc)
lt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--', label='Aleatório')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taxa Falsos Positivos')
plt.ylabel('Taxa Verdadeiros Positivos')
plt.title('Curva ROC')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Figura 19 - Códigos em Python para o traçado da curva ROC.

3.3.8 Aplicação do modelo treinado em novos dados

Uma vez definido o melhor modelo, é possível aplicá-lo a novos dados. Ressalta-se que o uso do modelo não necessita mais que os dados estejam rotulados com sentimento negativo ou positivo, bastando fornecer como *input* uma certidão de julgamento de um mandado de segurança para que o algoritmo retorne uma predição de classificação de sentimento positiva ou negativa. Evidentemente que a depender dos resultados obtidos durante a etapa de treinamento do modelo, a confiabilidade da classificação retornada pelo algoritmo pode não ser suficiente para utilizá-lo indiscriminadamente, mas posteriormente alterações no algoritmo podem ser implementadas a fim de tornar os seus resultados mais confiáveis.

A Figura 20 evidencia o código utilizado para aplicar o algoritmo já treinado em novos dados, sendo

```
# Realiza a classificação de novos dados
predicao = classificador.predict(novos_dados)
```

Figura 20 - Código em Python para compelir o algoritmo a a predizer a classificação de novos dados.

4 CONCLUSÃO

4.1 Considerações Iniciais

Extrair o resultado de forma automatizada de mandados de segurança apresentados ao STF - fornecendo, para isso, apenas informações das certidões de julgamento -, pode ser muito útil para diversas finalidades, entre elas a contabilização total de cada tipo de resultado, o que fornece visão mais generalizada das decisões declaradas por esse tribunal. Uma vez descrito o passo a passo até a construção dessa aplicação, esta pode ser desenvolvida e seus resultados podem ser medidos. Nas seguintes seções são discutidas as contribuições deste trabalho para o tema (seção 4.2) e possíveis desenvolvimentos que podem ser feitos nessa linha de pesquisa em trabalhos futuros (seção 4.3).

4.2 Contribuições

Dando prosseguimento ao trabalho de Morais (2019), que disponibilizou uma base de dados passível de ser utilizada para uma aplicação de aprendizado de máquina, este trabalho descreveu a construção de uma aplicação de classificação de processos de mandados de seguranças do STF utilizando a informação textual de certidões de julgamento disponibilizadas pelo próprio tribunal em meio eletrônico. Uma vez construída a aplicação, espera-se que esta seja capaz de prever, com aceitável grau de acurácia, se certidões de julgamento fornecidas para aplicação culminaram em pleito favorável ao pedido inicial ou não. Esse procedimento é feito sem a necessidade de intervenção humana e de forma muito mais rápida podendo fornecer informações estatísticas - que demorariam meses, caso fossem buscadas de forma manual -, em questão de minutos.

Além das contribuições acima citadas, esta monografia avançou o tema nos seguintes aspectos:

- Descrição pormenorizada dos procedimentos básicos de construção de uma aplicação de classificação binária utilizando dados textuais;
- Descrição das limpezas e balanceamento de dados a que esses devem ser submetidos antes de serem utilizados na aplicação, o que pode guiar futuros trabalhos que exigem coletas de dados para a mesma finalidade;

- Enumeração dos principais códigos na linguagem *Python* para o desenvolvimento de aplicações que utilizam o aprendizado de máquina para classificações binárias em dados de texto;
- Sugestão de classificadores aptos a serem utilizados para classificações binárias e o código *Python* para executá-los;
- Descrição das principais métricas de avaliação de classificadores binários e o código *Python* para executá-las.

4.3 Trabalhos Futuros

Apesar de a presente monografia ter se limitado apenas a descrever a construção e o provável funcionamento de uma aplicação para que, em futuros trabalhos, esta seja de fato implementada, é possível ir mais além e mencionar algumas melhorias ou ampliações que podem dar prosseguimento ao tema.

De início pode-se mencionar que esta aplicação pode ser adaptada para a previsão de resultados de, não apenas de certificados de julgamento de mandados de segurança do STF, mas como também de qualquer certidão de julgamento publicada por qualquer tribunal. Nesse caso, seria possível, e até recomendável, que o tipo de ação interposta - assim como o tribunal responsável pela decisão -, seja informado ao algoritmo para que este capture as particularidades do texto de cada certidão de julgamento e saiba diferenciar os termos comumente utilizados em cada uma delas, resultando, dessa forma, em maior acurácia em seus resultados.

Ademais, é possível a verificar se a utilização de outros classificadores, assim como a de outras estratégias de vetorização, retornam resultados mais confiáveis nas classificações realizadas pela aplicação. Uma possibilidade seria o treinamento de uma rede neural artificial para essa finalidade.

Uma vez que a aplicação a ser desenvolvida já é capaz de processar dados textuais a ela submetidos, isso abre a possibilidade para o desenvolvimento de um algoritmo que processe não apenas certidões de julgamento, mas também peças processuais ou processos inteiros, podendo retornar uma classificação do resultado ou outros tipos de *output*, como resumos ou outras informações desejadas.

Por fim, essa aplicação se limitou a retornar apenas dois tipos de resultados: procedente, classificada como sentimento “positivo” nesta monografia; ou improcedente, classificada como sentimento “negativo”. No entanto, os resultados possíveis não são apenas dois, já que diversos resultados são considerados parcialmente procedentes ou são encaminhados para outros tribunais ou instâncias. Dessa forma, a evolução natural dessa aplicação seria a previsão desses outros resultados possíveis.

5 REFERÊNCIAS BIBLIOGRÁFICAS

CABRAL, M. O. **Detecção de Postagens com Informações Falsas Sobre a Pandemia do COVID-19 na Rede Social Instagram**. São Carlos: [s.n.], 2021.

CNJ. **Justiça em Números**. Conselho Nacional de Justiça. Brasília, p. 331. 2022.

FENNER, M. E. **Machine Learning With Python For Everyone**. [S.l.]: Addison-Wesley, 2019.

GANEGEDARA, T. **Natural Language Processing. The definitive NLP book to implement the most sought-after machine learning models and tasks.** Segunda Edição. ed. Birmingham: Packt Publishing Ltd., 2022.

GÉRON, A. **Mãos à Obra:** Aprendizado de Máquina com Scikit-Learn & TensorFlow. Conceitos, Ferramentas e Técnicas para a Construção de Sistemas Inteligentes. Rio de Janeiro: Starlin Alta Editora, 2019.

LIU, B. **Sentiment analysis:** mining opinions, sentiments, and emotions. Second Edition. ed. Illinois: Cambridge University Press, 2020.

MAJUMDAR, P. **Learn Emotion Analysis With R. Perform Sentiment Assessments, Extract Emotions, and Learn NLP Techniques Using R and Shiny.** New Delhi: BPB PUBLICATIONS, 2021.

MORAIS, G. R. D. **Direito e Inteligência Artificial:** Análise de Sentimento Aplicada em Certidões de Julgamento de Mandados de Segurança Impetrados no Supremo Tribunal Federal. Brasília: [s.n.], 2019.

RASHKA, S.; MIRJALILI, V. **Python Machine Learning:** Machine Learning and Deep Learning With Python, Schikit-Learn and TensorFlow 2. Third Edition. ed. Birmingham - Mumbai: Packt Publishing, 2019.

VAJJALA, S. *et al*. **Pratical Natural Language Processing:** Comprehensive Guide to Building Real-World NLP Systems. [S.l.]: O'Reilly Media, 2020.

VASILIEV, Y. **Natural Language Processing With Python and SpaCy:** A Pratical Introduction. San Francisco: [s.n.], 2020.