INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO ESPÍRITO SANTO
MESTRADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MARCIEL MARIO DEGASPERI

CLASSIFICAÇÃO DE RELEVÂNCIA DE ATENDIMENTOS DE UMA BASE DE *HELP DESK* POR MEIO DE TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL

Serra - ES
2023

MARCIEL MARIO DEGASPERI

CLASSIFICAÇÃO DE RELEVÂNCIA DE ATENDIMENTOS DE UMA BASE DE *HELP DESK* POR MEIO DE TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Engenharia de Controle de Automação.

Orientador: Prof. Dr. Daniel Cruz Cavalieri

Coorientador: Prof. Dr. Fidelis Zanetti de Castro

Serra - ES
2023

Dados Internacionais de Catalogação na Publicação (CIP)

D317c 2023

Degasperi, Marciel Mário

Classificação de relevância de atendimentos de uma base de help desk por meio de técnicas de processamento de linguagem natural / Marciel Mário Degasperi. - 2023.

56 f.; il.; 30 cm

Orientador: Prof. Dr. Daniel Cruz Cavalier.

Coorientador: Prof. Dr. Fidelis Zanetti de Castro.

Dissertação (mestrado) - Instituto Federal do Espírito Santo, Mestrado Profissional em Engenharia de Controle de Automação, 2023.

1. Automação. 2. Redes neurais (Computadores). 3. Processamento de linguagem natural (Computadores). I. Cavalier, Daniel Cruz. II. Castro, Fidelis Zanetti de. III. Instituto Federal do Espírito Santo. IV. Título.

CDD 629.895

Bibliotecário: Valmir Oliveira de Aguiar - CRB6/ES 566

MINISTÉRIO DA EDUCAÇÃO

INSTITUTO FEDERAL DO ESPÍRITO SANTO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MARCIEL MARIO DEGASPERI

"CLASSIFICAÇÃO DE RELEVÂNCIA DE ATENDIMENTOS DE UMA BASE DE HELP DESK POR MEIO DE TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL"

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo, como requisito parcial para obtenção de título de Mestre em Engenharia de Controle e Automação.

Aprovado em 12 de abril de 2023

COMISSÃO EXAMINADORA

Documento assinado digitalmente
gov.br DANIEL CRUZ CAVALIERI
Data: 20/06/2023 15:33:10-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Daniel Cruz Cavalieri
Instituto Federal do Espírito Santo
Orientador

Documento assinado digitalmente
gov.br FIDELIS ZANETTI DE CASTRO
Data: 21/06/2023 16:22:33-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fidelis Zanetti de Castro
Instituto Federal do Espírito Santo
Coorientador

Documento assinado digitalmente
gov.br GABRIEL TOZATTO ZAGO
Data: 21/06/2023 15:54:09-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gabriel Tozatto Zago
Instituto Federal do Espírito Santo
Membro Interno

Documento assinado digitalmente
gov.br HILARIO TOMAZ ALVES DE OLIVEIRA
Data: 20/06/2023 17:49:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Hilário Tomaz Alves de Oliveira
Instituto Federal do Espírito Santo
Membro Externo

AGRADECIMENTOS

A Deus, por me mostrar o caminho e me dar resiliência para segui-lo.

À minha esposa Juliette, que me incentivou e apoiou em todos os momentos, me dando forças e ânimo para seguir em frente. Seu amor, apoio e paciência foram fundamentais para o sucesso deste estudo. Você foi minha rocha durante os momentos difíceis e minha inspiração durante os momentos de triunfo.

Aos meus pais Eunice e Mario, que sempre foram uma fonte de incentivo e apoio em minha vida acadêmica. Sua dedicação, amor e suporte ao longo dos anos têm sido inestimáveis e eu sou muito grato por tudo o que fizeram por mim. Seu exemplo de perseverança e dedicação me inspirou a buscar meus sonhos e alcançar meus objetivos.

Aos meus professores, em especial meus orientadores Daniel e Fidelis, por todo o conhecimento, orientação e apoio que me deram durante o meu mestrado. Seus conselhos e *insights* foram inestimáveis e me ajudaram a crescer como pesquisador.

Aos diretores da empresa Conexos Consultoria e Sistemas LTDA, que me forneceram acesso à base de dados objeto deste estudo.

E a todos os familiares e amigos que direta ou indiretamente colaboraram em meu mestrado, seja com sugestões, críticas, elogios ou simplesmente com sua presença.

Não adianta olhar para o céu com muita fé e pouca luta.
(Gabriel O Pensador)

RESUMO

Sistemas de *Help Desk* possuem uma vasta e rica base de informações, composta pelo histórico de atendimentos realizados, que pode e deve ser utilizada como base de consulta para atendimentos seguintes. Ferramentas comuns de busca, como buscas por palavras-chave, mostram-se inviáveis em grandes *datasets*, além de poderem trazer resultados não necessariamente relacionados ao problema. Técnicas estado-da-arte existem, mas exigem grande custo computacional e operacional para seu treinamento e utilização. Nesse sentido, a proposta deste trabalho é investigar a capacidade de algoritmos de aprendizado de máquina em encontrar a característica aqui definida como "relevância": a característica de textos com conhecimento que possa ser reutilizado. A motivação é que os textos não relevantes possam ser removidos antecipadamente da base de dados, permitindo que algoritmos complexos possam ser empregados em uma base de dados mais condensada, reduzindo-se os custos computacionais. Foram realizados testes com diversas combinações entre o vetorizador TF-IDF e o *word embedding Doc2Vec* e os classificadores clássicos *Naive-Bayes*, *Adaptive Boosting*, Floresta Aleatória, *Stochastic Gradient Descent*, Regressão Logística, *Support Vector Machine* e *Light Gradient Boosting Machine*, e o classificador *TextConvoNet*, uma arquitetura baseada em Redes Neurais Convolucionais. O classificador *TextConvoNet* apresentou os melhores resultados, com acurária e F1-score próximas a 0,93, mostrando que o conceito é detectável e que a técnica é viável para a remoção de textos não relevantes de uma base de dados.

Palavras-chave: Redes Neurais. Processamento de Linguagem Natural. *Help Desk*. Classificação.

ABSTRACT

Service Desk systems have a vast and rich information base, consisting of the history of calls made, which can and should be used as a reference base for subsequent calls. Common search tools, such as keyword searches, prove to be unfeasible in large datasets, in addition to being able to bring results not necessarily related to the problem. There are several "state-of-the-art" approaches; however, these approaches generally require a high computational and operational costs, both for their training and for their use. In this context, this work aims to investigate the ability of machine learning algorithms in finding the characteristic defined here as "relevance": the characteristic of texts with knowledge that can be reused. The motivation is that non-relevant texts can be removed in advance from the database, allowing complex algorithms to be employed in a more condensed database, reducing computational costs. Tests were performed with several combinations between the TF-IDF vectorizer and the word embedding Doc2Vec and the classic classifiers Naive-Bayes, Adaptive Boosting, Random Forest, Stochastic Gradient Descent, Logistic Regression, Support Vector Machine and Light Gradient Boosting Machine, and the classifier TextConvoNet, an architecture based on Convolutional Neural Networks. The TextConvoNet classifier presented the best results, with accuracy and f1-score close to 0.93, showing that the concept is detectable and that the technique is viable for removing non-relevant texts from a database.

Keywords: Neural Network. Natural Language Processing. Service Desk. Classification.

LISTA DE FIGURAS

Figura 1 – Fluxo de atendimento em um sistema *Help Desk* típico. 10
Figura 2 – Exemplo de atendimentos relevante e não relevante 13
Figura 3 – Vetores no *Word2Vec* 18
Figura 4 – Representação gráfica dos algoritmos CBOW e *Skip-gram* 18
Figura 5 – Arquitetura da Floresta Aleatória 22
Figura 6 – Algoritmo *Adaptive Boosting* criando um classificador forte a partir de classificadores fracos 24
Figura 7 – Gradiente Descendente 24
Figura 8 – Representação do algoritmo SVM e seus vetores de suporte na classificação. 26
Figura 9 – Arquitetura do algoritmo *Gradient Boosting Decision Tree* 27
Figura 10 – Representação da arquitetura de uma CNN 29
Figura 11 – Arquitetura da Rede *TextConvoNet* 30
Figura 12 – Validação Cruzada, com k = 5 31
Figura 13 – Fluxograma do Trabalho. 32
Figura 14 – Classificação do *dataset*. 34
Figura 15 – Nuvem de Palavras que compõem o *dataset*. 36
Figura 16 – Nuvem de Palavras que compõem o *dataset* após remoção de saudações. 37
Figura 17 – Nuvem de Palavras que compõem o *dataset* por categoria. 39
Figura 18 – Curvas ROC com desvio-padrão - TF-IDF x Classificadores Clássicos - Dataset com 1084 atendimentos 42
Figura 19 – Curvas ROC com desvio-padrão - TF-IDF x Classificadores Clássicos - Dataset com 1868 atendimentos 44
Figura 20 – Curvas ROC com desvio-padrão - Doc2Vec x Classificadores Clássicos - Dataset com 1868 atendimentos 46
Figura 21 – Curva ROC com desvio-padrão - *TextConvoNet* 47
Figura 22 – Curva ROC com desvio-padrão - *TextConvoNet* modificado 48
Figura 23 – Curvas ROC com desvio-padrão - Melhores classificadores 49

LISTA DE TABELAS

Tabela 1 – Informações descritivas do *dataset*. 35
Tabela 2 – Informações descritivas do *dataset* de treinamento. 38
Tabela 3 – Palavras mais recorrentes no *dataset* integral após remoção de saudações. 38
Tabela 4 – Palavras mais recorrentes no *dataset* relevante após remoção de saudações. 38
Tabela 5 – Palavras mais recorrentes no *dataset* não relevante após remoção de saudações. 39
Tabela 6 – Implementação dos algoritmos de classificação. 40
Tabela 7 – Métricas de alguns classificadores clássicos quando aplicados ao conjunto de dados. 41
Tabela 8 – Métricas de classificadores clássicos em conjunto com o vetorizador TF-IDF. 43
Tabela 9 – Métricas de classificadores clássicos em conjunto com o vetorizador Doc2Vec. 45
Tabela 10 – Métricas do classificador TextConvoNet. 47
Tabela 11 – Métricas do classificador TextConvoNet modificado em conjunto com o vetorizador Doc2Vec. 48
Tabela 12 – Métricas dos melhores classificadores utilizados. 48

LISTA DE SIGLAS

AUC	– Area Under Curve
CBOW	– Continuous Bag of Words
CNN	– Convolutional Neural Network
EFB	– Exclusive Feature Bundling
Doc2Vec	– Document to Vector
GloVe	– Global Vectors for Word Representation
GPT-3	– Generative Pre-Training Transformer - versão 3
HD	– Help Desk
LaMDA	– Language Model for Dialogue Applications
LightGBM	– Light Gradient Boost Decision Tree
PLN	– Processamento de Linguagem Natural
ROC	– Receiver Operating Characteristics Curve
SGD	– Stochastic Gradient Descent
SVM	– Support Vector Machine
TF-IDF	– Term Frequency - Inverse Document Frequency
Word2Vec	– Word to Vector

SUMÁRIO

1 INTRODUÇÃO 10
1.1 OBJETIVOS E PRESSUPOSTOS 13
1.1.1 Objetivo Geral 13
1.1.2 Objetivos Específicos 13
1.1.3 Pressupostos 14
1.2 ESTRUTURA DO TEXTO 14
2 REVISÃO BIBLIOGRÁFICA 15
2.1 PROCESSAMENTO DE LINGUAGEM NATURAL 15
2.2 ALGORITMOS DE VETORIZAÇÃO E *EMBEDDING* 15
2.2.1 TF-IDF 16
2.2.2 *Word2Vec* 17
2.2.3 *Doc2Vec* 19
2.2.4 *GloVe* 19
2.3 ALGORITMOS DE CLASSIFICAÇÃO 20
2.3.1 *Naive-Bayes* 20
2.3.2 Floresta Aleatória 21
2.3.3 *Adaptive Boosting* 23
2.3.4 *Stochastic Gradient Descent* 24
2.3.5 Regressão Logística 25
2.3.6 *Support Vector Machine* 25
2.3.7 *Light Gradient Boosting Machine* 27
2.3.8 *Redes Neurais Convolucionais* 28
2.3.9 *TextConvoNet* 29
2.4 MÉTRICAS DE AVALIAÇÃO 30
2.5 VALIDAÇÃO CRUZADA 31
3 METODOLOGIA 32
3.1 AMBIENTE DE DESENVOLVIMENTO 32
3.1.1 Extração dos dados 32
3.1.2 Processamento dos dados 32
3.2 MÉTODOS PROPOSTOS 33
3.3 BASE DE DADOS 34
3.4 ANÁLISE EXPLORATÓRIA DOS DADOS 35
3.5 PRÉ-PROCESSAMENTO DOS DADOS 36
4 RESULTADOS 40
4.1 VETORIZADOR TF-IDF X CLASSIFICADORES CLÁSSICOS 40
4.1.1 Base de dados com 1084 atendimentos 40
4.1.2 Base de dados com 1868 atendimentos 43
4.2 *EMBEDDING* DOC2VEC X CLASSIFICADORES CLÁSSICOS 45

4.3 TEXTCONVONET 47
4.4 TEXTCONVONET MODIFICADO 47
4.5 SÍNTESE DOS RESULTADOS 48
5 CONSIDERAÇÕES FINAIS 51
REFERÊNCIAS 53

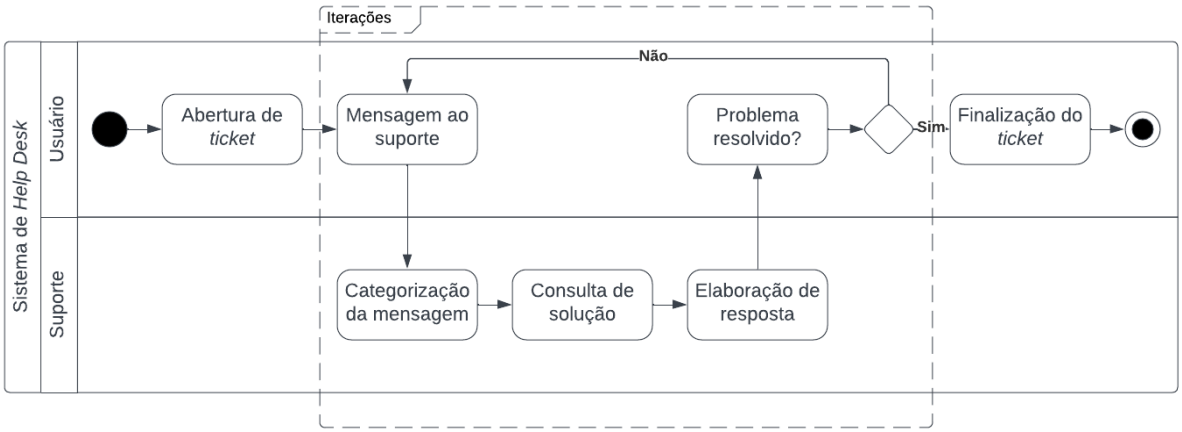
1 INTRODUÇÃO

A preocupação com a satisfação do cliente é uma constante para empresas manterem-se competitivas. Essa satisfação se dá por meio da avaliação que o consumidor faz em relação ao produto ou serviço ofertado. O não atendimento das necessidades e expectativas do cliente gera insatisfação com os serviços e consequentemente com a empresa (LAS CASAS, 1991). Para se diferenciar das demais e gerar credibilidade, é essencial que as organizações tenham a capacidade de responder prontamente às solicitações, sugestões ou reclamações (COSTA; SANTANA; TRIGO, 2015).

Help Desk é um termo que designa o serviço de apoio a clientes para suporte e resolução de problemas técnicos (MELENDEZ FILHO, 2011). Os sistemas de *Help Desk* desempenham um papel importante como intermediários entre as empresas e seus clientes. Eles oferecem um sistema eficiente que permite a coleta e a resolução ágil de problemas, além de registrar as soluções apresentadas e correlacioná-las com outras situações (BOSCOLO, 2009).

Em um sistema *Help Desk* típico, o fluxo inicia-se com um chamado de serviço (*ticket*) criado por um usuário, detalhando os problemas encontrados. Cada *ticket* é associado manualmente a uma categoria, que está vinculada a uma ou mais bases de conhecimento. Por fim, uma solução é recuperada de uma base de conhecimento relacionada ao problema. Este ciclo pode se repetir até que o usuário tenha sua solicitação atendida ou seu problema resolvido (LO et al., 2019). A Figura 1 apresenta o fluxo em um atendimento em um sistema de *Help Desk* típico.

Figura 1 – Fluxo de atendimento em um sistema *Help Desk* típico.



Fonte: Autoria própria (2023)

Esta abordagem possui problemas a médio e longo prazo, como: número de categorias crescente e com duplicidades; é difícil classificar manualmente os *tickets* com precisão,

o que pode levar a uma resposta errônea, demandando esforço e tempo extras na solução; e o número de *tickets* pode crescer rapidamente caso haja uma falha em um sistema amplamente implantado (LO et al., 2019).

A utilização de sistemas de *Help Desk* possibilita a criação de uma ampla base de dados para a empresa, a qual permite gerenciar os problemas resolvendo-o na sua raiz, diminuindo custos operacionais. O *Help Desk* pode centralizar uma diversidade de informações e áreas de atendimento, tornando-se assim um ponto chave na administração e na solução de problemas (CAVALARI; COSTA, 2005).

Com o crescimento da base de dados dos sistemas de *Help Desk*, cresce também a necessidade em recuperar informações de forma eficiente. Uma forma simples de obter um conjunto de respostas para uma consulta de usuário é determinar quais documentos em uma coleção contém um certo conjunto de palavras-chave da consulta. Todavia, isto pode não ser suficiente para atender ao usuário, pois a presença de documentos não relevantes entre os documentos retornados por uma consulta é praticamente certa. Nesse cenário, o principal objetivo de um sistema desta natureza deve ser recuperar o maior número possível de documentos relevantes e o menor número possível de documentos não relevantes (BAEZA-YATES; RIBEIRO-NETO, 2013). Caso contrário, seu acesso é inviabilizado e o conhecimento contido se torna inutilizável.

Diante do exposto, verifica-se a necessidade da utilização de ferramentas de apoio na análise dos atendimentos de forma a auxiliar a classificação e recuperação de informações no conjunto de dados. Como os dados dos atendimentos são compostos basicamente por textos descritivos, dá-se a necessidade de propor e realizar sistemas computacionais baseados em Processamento de Linguagem Natural (PLN).

Com o avanço dos atendimentos remotos via sistemas de *Help Desk* e a evolução das técnicas de aprendizado de máquina, a busca por ferramentas que auxiliem na classificação e busca de atendimentos vem ganhando importância. Lo et al. (2019) realizam alguns ensaios de viabilidade para otimizar os processos de consulta e classificação da base de dados de *Help Desk*, utilizando técnicas de *Big Data* e PLN, com resultados promissores. Já os trabalhos de Paramesh e Shreedhara (2019) e Al-Hawari e Barham (2021) utilizam aprendizado de máquina para categorização multiclasse automatizada dos atendimentos a partir do texto do problema relatado pelo usuário.

Apesar do grande volume de produções na comunidade acadêmica, o PLN continua sendo um grande desafio no campo computacional. Yao et al. (2018) destacam que para o PLN são necessárias análise gramatical, semântica e compreensão de texto, abrangendo diversas áreas de conhecimento, como mineração de dados, aprendizado

de máquina, aquisição de conhecimento, engenharia do conhecimento, pesquisa de inteligência artificial e pesquisa linguística relacionada ao processamento de linguagem, como pode ser visto pelos trabalhos de Yao e Guan (2018), Bassani e Araujo (2019) e Minaee et al. (2021).

Sistemas de *Help Desk* geram uma base de conhecimento rica, porém esparsa. Os atendimentos prestados podem não conter conhecimento que possa ser reaproveitado futuramente. Por exemplo, um relato de um problema pontual, corrigido imediatamente, um atendimento referente a uma cobrança em atraso, ou um atendimento aberto por engano.

Existem algoritmos estado-da-arte para processamento de dados textuais. Uma família de algoritmos que ganharam destaque recente são as chamadas Redes *Transformers*, como a GPT-3 (*Generative Pre-trained Transformer* - terceira geração) e a LaMDA (*Language Model for Dialogue Applications*). Apesar de apresentarem resultados surpreendentes, seu treinamento necessita de ajuste fino em um conjunto de dados de milhares a centenas de milhares de exemplos específicos para a tarefa. Além disso, exigem um grande poder computacional para seu processamento devido a este volume de informações (BROWN et al., 2020). Dessa forma, faz-se necessário pesquisas em estratégias de condensação de informações relevantes, de forma a minimizar os custos computacionais e operacionais.

A proposta deste trabalho é analisar a viabilidade de algoritmos de classificação existentes na literatura na tarefa de identificação de atendimentos que possuam conhecimento a ser reutilizado, aqui denominados “relevantes”. Com este resultado, é possível montar um novo *dataset*, mais condensado, onde outras estratégias de extração de conhecimento possam ser aplicadas de forma otimizada.

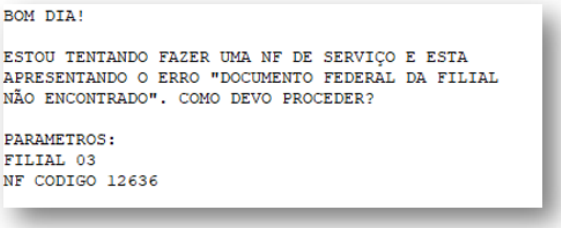
O conceito de relevância é um tema amplamente debatido na Ciência da Informação. Para Baeza-Yates e Ribeiro-Neto (2013), a relevância é definida como uma métrica de qualidade, ligada à satisfação de um documento atender a um determinado pedido. A noção de relevância é humana e não técnica, composta por uma gama de variáveis e, portanto, frágil e de difícil definição.

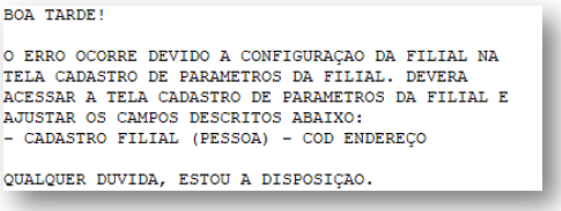
O conceito de relevância explorado neste trabalho é mais amplo, e abstrato. Neste estudo, foi definido como relevância a característica de textos que possuam qualquer conhecimento reutilizável. Em outras palavras, um texto é definido como “relevante” se um leitor puder aprender algo em sua leitura e reutilizar este conhecimento futuramente, independente da área de conhecimento correlacionada ao texto. Uma pessoa, ao ler um texto sobre qualquer assunto, pode dizer com certa facilidade “Eu aprendi algo lendo isto” ou “Não há nada de importante neste documento”, apesar de ser difícil para

uma pessoa descrever quais elementos textuais foram usados para esta afirmação. A Figura 2 demonstra dois atendimentos reais extraídos do *dataset*. Embora sejam semelhantes, apenas o primeiro atendimento possui conhecimento reutilizável em outros atendimentos e, portanto, considerado relevante.

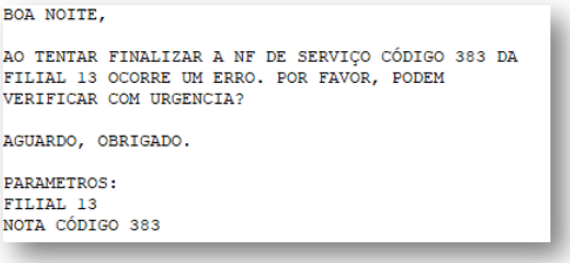
Figura 2 – Exemplo de atendimentos relevante e não relevante

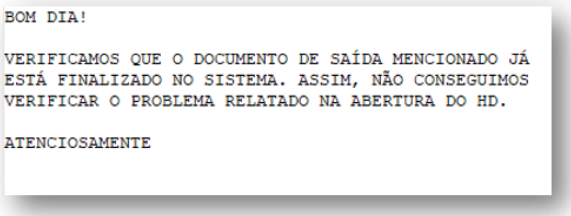
(a) Atendimento relevante





(b) Atendimento não relevante





Fonte: Autoria própria (2023)

Aparentemente, este conceito não parece diretamente ligado às palavras-chave, nem à forma como o texto foi estruturalmente redigido. Um texto curto como uma postagem de *microblog* pode conter mais relevância do que um matéria de revista com várias páginas. O desafio deste trabalho é estudar a capacidade de técnicas de classificação conhecidas para a identificação de características tão abstratas quanto a apresentada.

1.1 OBJETIVOS E PRESSUPOSTOS

Neste tópico são apresentados os objetivos e pressupostos do trabalho.

1.1.1 Objetivo Geral

Desenvolver métodos que possam auxiliar no processo de classificação de relevância dos atendimentos da base de *Help Desk*.

1.1.2 Objetivos Específicos

Para atingir o propósito deste trabalho, os seguintes objetivos específicos foram propostos:

1. Montar uma base de dados a partir de dados do sistema *Help Desk*, classificando os atendimentos em relevantes ou não relevantes;

2. Representar o texto de cada atendimento na forma vetorizada utilizando a estratégia TF-IDF e as estratégias de *word embedding* Doc2Vec e GloVe;

3. Implementar um conjunto de algoritmos clássicos para a classificação dos atendimentos em relevantes e não relevantes;

4. Implementar um conjunto de algoritmos baseados em redes neurais profundas para a classificação dos atendimentos em relevantes ou não relevantes;

5. Comparar e avaliar os resultados obtidos.

1.1.3 Pressupostos

Como resultado da aplicação da ferramenta a ser desenvolvida, são pressupostos deste trabalho:

- Melhorar o tempo de atendimento de solicitações dos clientes.

- Melhorar a assertividade das respostas às solicitações.

- Melhorar o tempo de treinamento de novos funcionários no atendimento.

- Melhorar a qualidade no atendimento e, por conseguinte, a satisfação do cliente.

1.2 ESTRUTURA DO TEXTO

O presente trabalho está dividido em cinco capítulos, conforme segue:

Capítulo 1: Apresentação geral sobre o problema, os objetivos gerais e específicos, e os pressupostos.

Capítulo 2: Revisão bibliográfica de conceitos e métodos utilizados neste trabalho.

Capítulo 3: Apresentação do *dataset* de estudo e a metodologia empregada.

Capítulo 4: Apresentação dos resultados.

Capítulo 5: Apresentação das conclusões e trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão apresentados os conceitos e ferramentas relevantes ao trabalho.

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

O Processamento de Linguagem Natural (PLN) é uma importante área de interesse em Inteligência Artificial e Ciência da Computação. A importância desse campo de estudo cresce paralelamente à quantidade de informações textuais digitalizadas sem precedente na história. Como um campo de estudo científico, o PLN engloba conceitos de ciência da computação, linguística e matemática com o objetivo principal de traduzir a linguagem humana em comandos que podem ser executados por computadores. Embora existam diversas técnicas bem estabelecidas de PLN, a linguagem natural continua sendo difícil de processar. Como exemplo, é possível que as palavras omitidas e gramática ruim não interfiram na comunicação entre duas pessoas. Para computadores, no entanto, isto é um grande obstáculo (KANG et al., 2020).

De acordo com Kang et al. (2020), o PLN pode ser dividido em quatro etapas: pré-processamento, vetorização, treinamento do modelo e avaliação do modelo. O pré-processamento de texto visa obter um texto “limpo”, removendo símbolos e outros elementos que possam gerar ruído na análise subsequente. Após o pré-processamento do texto, é selecionado um algoritmo que converta palavras em uma representação numérica, como vetores. Com base nesses vetores de palavras, é possível aplicar algoritmos para treinar um modelo que pode resolver o problema real. Por fim, após o treinamento do modelo, é necessário sua avaliação para aferir sua qualidade e sua aplicabilidade em outros cenários.

2.2 ALGORITMOS DE VETORIZAÇÃO E *EMBEDDING*

Os computadores, embora sejam capazes de executar uma ampla gama de tarefas envolvendo multimídia, ainda são máquinas executando instruções sobre números no sistema binário. Dessa forma, para possibilitar a classificação de textos com recursos computacionais, é necessário a seleção de alguma estratégia de tradução que possa representar os conjuntos de palavras do documento em forma numérica, sem que haja perda de características importantes (SINGH; SHASHI, 2019).

Existem duas categorias principais de estratégias de conversão de textos em valores numéricos: a vetorização e o *embedding*. A vetorização utiliza técnicas estatísticas para associar cada palavra de um texto a um número que represente sua importância no texto, utilizando-se de uma ou mais métricas para seu cálculo (RAJARAMAN; ULLMAN, 2011). Já no *embedding*, cada palavra é associada a um vetor em um

espaço multidimensional, de tal forma que seu valor semântico é mantido, permitindo que relações semânticas sejam definidas a partir de operações vetoriais nos vetores associados (LE; MIKOLOV, 2014).

Entre os algoritmos de vetorização e *embedding* mais utilizados, podemos citar *Term Frequency times Inverse Document Frequency* (TF-IDF), *Word2Vec*, *Doc2Vec* e *GloVe*. Mais detalhes destes algoritmos são apresentados nos tópicos a seguir.

2.2.1 TF-IDF

Uma forma comum de categorização de textos é identificando a ocorrência de palavras-chave que caracterizam os documentos sobre um tema em específico. Por exemplo, artigos sobre beisebol tenderiam a ter muitas ocorrências de palavras como “bola”, “bastão”, “arremesso”, “correr” e assim por diante. O algoritmo *Term Frequency times Inverse Document Frequency*, ou simplesmente TF-IDF, é um algoritmo de vetorização utilizado para calcular a importância de uma palavra em um documento em relação a um conjunto de documentos. Ele é amplamente utilizado em tarefas de processamento de linguagem natural, como recuperação de informação e análise de sentimentos (RAJARAMAN; ULLMAN, 2011).

O algoritmo utiliza duas medidas: a frequência do termo (TF) e a frequência inversa do documento (IDF). A frequência do termo (TF) é a medida de quantas vezes um termo aparece em um documento, definido pela Equação 1:

$$TF_{ij} = \frac{f_{ij}}{max_k(f_{kj})} \tag{1}$$

onde

- f_{ij}: Número de ocorrências do termo i no texto j;
- $max_k(f_{kj})$: Maior índice k das palavras do texto j, ou seja, a quantidade total de palavras no texto j.

Já a frequência inversa do documento (IDF) é a medida de quanto um termo é raro em relação a um conjunto de documentos, definido pela Equação 2:

$$IDF_i = \log_2 \frac{N}{n_i} \tag{2}$$

onde

- N: Número total de documentos;

- n_i: Número de documentos que contém o termo i.

A importância de um termo em um documento é dada pela multiplicação das medidas TF e IDF (RAJARAMAN; ULLMAN, 2011). Logo, o valor TF-IDF para cada palavra i do documento j é dado pela Equação 3:

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i \tag{3}$$

Por exemplo, suponha que temos um conjunto de documentos relacionados à tecnologia da informação, e queremos calcular a importância de uma palavra específica em cada documento. Se a palavra “algoritmo” aparecer 10 vezes no documento A, que tem um total de 1000 palavras, temos:

$$TF = \frac{10}{1000} = 0{,}01$$

Se a palavra “algoritmo” aparecer em 125 dos 1000 documentos que temos, então:

$$IDF = \log_2 \frac{1000}{125} = \log_2 8 = 3$$

O TF-IDF para a palavra “algoritmo” no documento A será:

$$TF\text{-}IDF = 0{,}01 \cdot 3 = 0{,}03$$

Dessa forma, podemos calcular o TF-IDF para cada palavra em cada documento e, em seguida, usar esses valores para identificar as palavras mais relevantes em cada documento e também comparar a relevância das palavras em diferentes documentos.

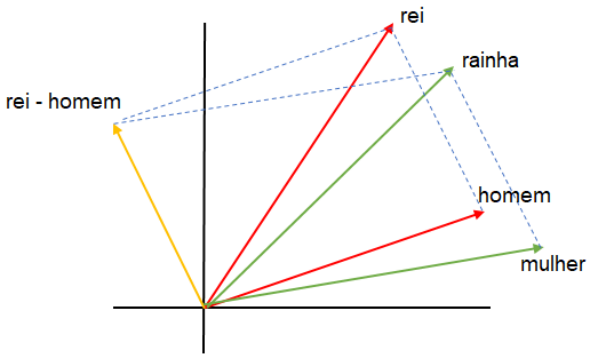
A importância de um termo para um documento aumenta se ele aparece com frequência no documento e raramente aparece nos outros documentos. Isso permite que o algoritmo TD-IDF consiga destacar palavras-chave relevantes em um documento, o que é útil em tarefas de recuperação de informação e indexação de documentos (RAJARAMAN; ULLMAN, 2011).

2.2.2 *Word2Vec*

O *Word2Vec* é um algoritmo de aprendizado não supervisionado para gerar representações vetoriais de palavras, ou *embeddings*. Ele foi desenvolvido em 2013 e é amplamente utilizado em tarefas de processamento de linguagem natural, como classificação de texto, tradução automática e análise semântica. Durante o treinamento, o algoritmo gera esses vetores de forma que palavras similares tenham vetores similares. Como os vetores são representações numéricas das palavras, é possível realizar operações matemáticas com eles, como adição e subtração. Por exemplo, se “homem” - “mulher” + “rei” = “rainha”, podemos inferir que a palavra

“rainha” é relacionada a “rei” da mesma forma que “mulher” é relacionada a “homem” (MIKOLOV et al., 2013). Na Figura 3 é representado a organização dos dados no modelo Word2Vec.

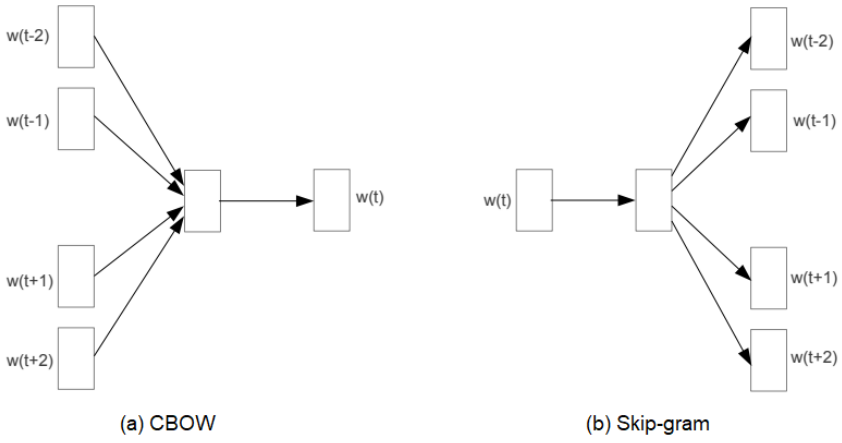
Figura 3 – Vetores no *Word2Vec*



Fonte: Autoria própria (2023)

Word2Vec é um algoritmo composto por dois modelos de aprendizagem: o *Continuous Bag of Words* (CBOW), que prevê a palavra dado seu contexto e o *Skip-gram*, que prevê o contexto dado uma palavra. Na Figura 4 é representado o funcionamento dos algoritmos CBOW e *Skip-gram*.

Figura 4 – Representação gráfica dos algoritmos CBOW e *Skip-gram*



Fonte: Adaptado de Mikolov, Le e Sutskever (2013)

Ao alimentar o corpo de texto em um modelo de aprendizado, o *Word2Vec* gera vetores de palavras. Nesse processo, o *Word2Vec* primeiro constrói um vocabulário a partir do texto de treinamento e aprende as representações vetoriais de cada palavra. Além disso, a *Word2Vec* tem a capacidade de calcular a distância entre cada palavra. Com isso, palavras semelhantes são agrupadas com base em suas distâncias. Ao agrupar as palavras semelhantes, a dimensão original do recurso é projetada para uma nova dimensão reduzida (MA; ZHANG, 2015).

2.2.3 *Doc2Vec*

Doc2vec é uma extensão do algoritmo *Word2Vec* que permite a representação vetorial de documentos inteiros e não apenas palavras individuais. O objetivo da rede *Doc2Vec* é aprender uma representação de cada documento de tal forma que documentos similares tenham vetores de representação próximos no espaço vetorial.

O treinamento da *Doc2Vec* é realizando usando-se uma grande quantidade de documentos etiquetados, onde a etiqueta indica qual é o tema principal do documento. Durante o treinamento, a rede é exposta a várias combinações de documentos e etiquetas e tenta prever a etiqueta de um documento dado a partir da sua representação vetorial. Uma vez treinada, a rede *Doc2Vec* pode ser usada para produzir vetores de representação de novos documentos de texto. Estes vetores podem então ser usados em várias tarefas, como classificação de documentos, agrupamento de documentos e recuperação de informações (LE; MIKOLOV, 2014).

2.2.4 *GloVe*

GloVe (*Global Vectors for Word Representation*) é um modelo de *embedding* que utiliza técnicas algébricas para associar cada palavra a um vetor de características numéricas. O objetivo é capturar as relações semânticas e sintáticas entre as palavras a partir do contexto em que elas aparecem em um grande *corpus* de texto. Assim como no *Word2Vec*, os vetores obtidos pelo *GloVe* têm a propriedade de possuir similaridade semântica e sintática. Por exemplo, palavras similares tendem a ter vetores similares, e operações matemáticas com os vetores podem resultar em palavras relacionadas a elas.

O processo de treinamento do *GloVe* consiste em algumas etapas:

1. Co-ocorrência de palavras: é construída uma matriz de co-ocorrência de palavras a partir do corpus, em que cada entrada na matriz representa a frequência com que duas palavras aparecem juntas em um contexto dado.

2. Decomposição de valor singular (SVD): é realizada uma decomposição de valor singular (SVD) na matriz de co-ocorrência para reduzir sua dimensionalidade e obter os vetores de características das palavras.

3. Aprendizado de *embeddings*: são obtidos os vetores de *embeddings* das palavras, que são obtidos a partir de uma função de custo que busca minimizar a diferença entre a frequência esperada de co-ocorrência de duas palavras e a frequência observada, ponderada pela similaridade entre as palavras.

A propriedade de similaridade semântica e sintática dos vetores obtidos pelo *GloVe* é obtida a partir da função de custo e a definição de frequência esperada de co-ocorrência, que leva em conta a similaridade entre as palavras.

O *GloVe* é um modelo de *embedding* de palavras muito utilizado em tarefas de processamento de linguagem natural, como análise de sentimentos, classificação e geração de texto (PENNINGTON; SOCHER; MANNING, 2014).

2.3 ALGORITMOS DE CLASSIFICAÇÃO

Os algoritmos de classificação são métodos usados para identificar novas classes de observações com base em dados de treinamento. O algoritmo aprende a partir de conjuntos de dados ou observações e então classifica novas observações em várias classes ou grupos (KOWSARI et al., 2019). A seguir são expostos os algoritmos de classificação utilizados neste trabalho.

2.3.1 *Naive-Bayes*

O Teorema de Bayes, inicialmente descrito em trabalhos do matemático inglês Thomas Bayes e formalizado pelo matemático francês Pierre-Simon Laplace no século XVIII, descreve a probabilidade de um evento ocorrer baseado em um conhecimento *a priori* que pode estar relacionado ao evento (SWINBURNE, 2005).

O teorema de Bayes é definido pela Equação 4:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{4}$$

onde:

- A e B são eventos;
- $P(A|B)$ é a probabilidade de A ocorrer dado que B ocorreu;
- $P(B|A)$ é a probabilidade de B ocorrer dado que A ocorreu;
- $P(A)$ é a probabilidade de A ocorrer;
- $P(B)$ é a probabilidade de B ocorrer;
- $P(B) \neq 0$.

O Teorema de Bayes expressa uma relação entre novos dados com dados preexistentes e conhecidos. Essa relação algorítmica entre novo e o conhecido tem apoio intuitivo no aprendizado, visto que a experiência anterior ajuda a interpretar novas experiências adquiridas. Segundo a filosofia, linguística e psicologia modernas, chamamos essa atividade de aprendizagem de "construtivista". O Teorema de Bayes oferece um modelo plausível dessa visão de mundo construtivista, o que o torna uma importante ferramenta de modelagem para grande parte da IA moderna, incluindo algoritmos para compreensão de linguagem natural, robótica e aprendizado de máquina (LUGER; CHAKRABARTI, 2017).

O classificador *Naive-Bayes* é um algoritmo de classificação probabilístico que se baseia no Teorema de Bayes para fazer previsões. Ele é chamado de "*naive*" (ingênuo) pois assume que todas as características de um dado são independentes entre si, o que nem sempre é verdade. No entanto, mesmo com essa suposição simplificadora, o classificador *Naive-Bayes* geralmente tem um bom desempenho em várias tarefas de classificação (CHEN et al., 2009).

Na classificação de textos, o classificador *Naive-Bayes* é um método popular devido à sua eficiência computacional e bom desempenho na predição (CHEN et al., 2009). Neste classificador, cada documento é visto como uma coleção de palavras, onde a posição de ocorrência de cada palavra não é considerada. Por ser simples e rápido, normalmente se sobressai a outros classificadores. Além disso, o *Naive-Bayes* só precisa de um pequeno número de dados de teste para concluir classificações com uma boa precisão (FRANK; BOUCKAERT, 2006).

Existem algumas implementações com nuances diferentes do classificador *Naive Bayes*, dentre as quais podemos citar o modelo Gaussiano, o modelo Multinomial e o modelo de Bernoulli. Destes, o modelo Multinomial apresenta os melhores resultados para a classificação de textos (SINGH et al., 2019).

2.3.2 Floresta Aleatória

Árvore de Decisão é um modelo sequencial que combina de forma lógica uma sequência de testes simples. Cada teste compara um atributo numérico com um valor limite ou um atributo nominal com um conjunto de valores possíveis (KOTSIANTIS, 2013). Com isso, o algoritmo consegue predizer a classe associada a um objeto viajando da raiz da árvore para uma folha. Em cada nó no caminho da raiz para a folha, o filho sucessor é escolhido com base na divisão do espaço de entrada. Normalmente, a divisão é baseada em uma das características ou em um conjunto predefinido de regras (SHALEV-SHWARTZ; BEN-DAVID, 2014). As Árvores de Decisão podem ser usadas para descobrir recursos e extrair padrões em grandes bancos de dados que

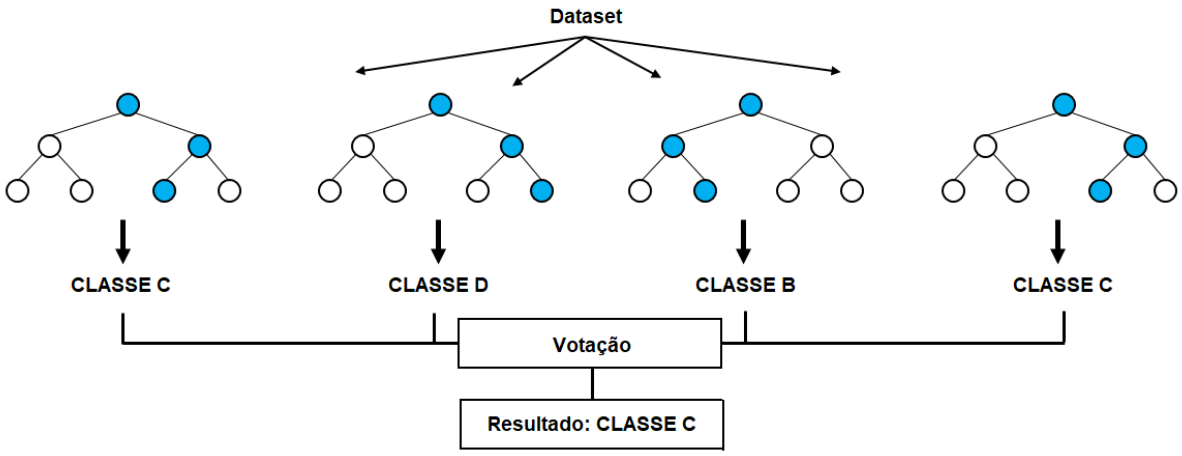
são importantes para discriminação e modelagem preditiva. Essas características, juntamente com sua interpretação intuitiva, são algumas das razões pelas quais as árvores de decisão serem usadas extensivamente para análise exploratória de dados (MYLES et al., 2004).

De acordo com Rokach e Maimon (2005), a Árvore de Decisão possui algumas limitações conhecidas. Dente elas, podemos citar:

- Instabilidade: Pequenas variações nos dados de treinamento podem gerar árvores de decisão significativamente diferentes, afetando a estabilidade do modelo;
- *Overfitting*: A árvore pode se ajustar aos dados de treinamento, mas não generalizar o problema;
- Baixa performance em grande quantidade de dados;
- Dificuldade em lidar com relações não lineares.

A fim de minimizar as limitações conhecidas na Árvore de Decisão, Ho (1995) propôs a Floresta Aleatória, ou *Random Forest*. Trata-se de uma combinação de árvores de decisão, totalmente independentes, de tal forma que cada árvore é construída em um subespaço aleatório do espaço de características. Árvores em diferentes subespaços generalizam sua classificação de maneira complementar, e sua classificação combinada apresenta melhor precisão do que a classificação individual. O resultado final é a média ou a votação das previsões de cada árvore na floresta. A Figura 5 representa o algoritmo Floresta Aleatória na tarefa de classificação.

Figura 5 – Arquitetura da Floresta Aleatória



Fonte: Adaptado de medium.com[1]

[1] Adaptado de <https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57>. Acesso em: 21 fev. 2023.

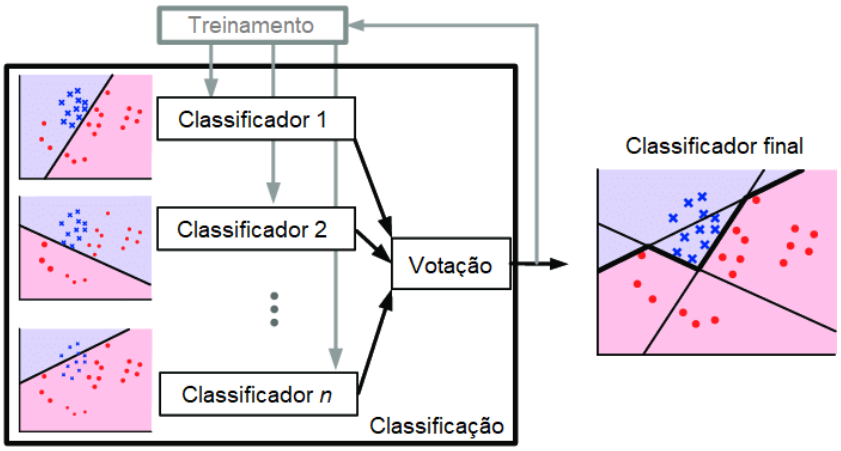
2.3.3 *Adaptive Boosting*

"*Boosting*" é uma técnica proposta na década de 90. A estratégia central do algoritmo é combinar diversos classificadores pouco precisos, de forma a produzir um novo algoritmo de maior precisão (MAYR et al., 2018).

Schapire e Freund (2013) compararam o conceito geral de *boosting* com "obter sabedoria de um conselho de tolos". Os "tolos", neste caso, são os classificadores simples, que classificam apenas um pouco melhor do que o lançamento de uma moeda. Estes classificadores simples por si só não são suficientes para fornecer uma resposta precisa, porém contém alguma informação válida sobre a estrutura do problema. A tarefa do algoritmo é, portanto, aprender com a aplicação iterativa de um classificador simples e usar essa informação para combiná-la a uma classificação precisa. No entanto, chamar o classificador simples várias vezes nos mesmos dados de treinamento não afeta seu desempenho. O conceito de reforço não é manipular o próprio classificador, mas sim manipular os dados de treinamento subjacentes reponderando iterativamente as observações. Como resultado, o classificador base encontrará uma nova solução em cada iteração (MAYR et al., 2018).

Dentre as várias implementações de *boosting* existentes, o *Adaptive Boosting*, ou *AdaBoost* é comumente mais usada (FRIEDMAN; HASTIE; TIBSHIRANI, 2000). O conceito básico por trás do *AdaBoost* é definir os pesos dos classificadores e treinar a amostra de dados em cada iteração de forma que garanta as previsões precisas de observações incomuns. Em cada iteração, ele tenta fornecer um ajuste ótimo para esses exemplos, minimizando o erro de treinamento para que a próxima iteração faça uma classificação mais precisa (KOWSARI et al., 2019). As implementações do *AdaBoost* normalmente utilizam uma Árvore de Decisão como classificador base, e normalmente resulta em um desempenho significativamente melhor quando comparado com a Árvore de Decisão sozinha (MAYR et al., 2018). A Figura 6 representa o funcionamento do *AdaBoost*.

Figura 6 – Algoritmo *Adaptive Boosting* criando um classificador forte a partir de classificadores fracos

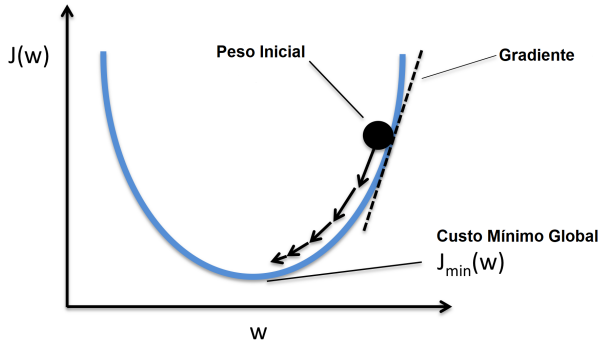


Fonte: Adaptado de Wang, Zhang e Verma (2015)

2.3.4 *Stochastic Gradient Descent*

Na matemática, "Gradiente Descendente" é um algoritmo de otimização iterativa de primeira ordem para encontrar um mínimo local de uma função diferenciável. A ideia é dar passos repetidos na direção oposta do gradiente da função no ponto atual, pois esta é a direção de descida mais acentuada (MIKOLOV; LE; SUTSKEVER, 2013). Na Figura 7 é exibido uma visualização do Gradiente Descendente.

Figura 7 – Gradiente Descendente



Fonte: Adaptado de sebastianraschka.com[2]

O *Stochastic Gradient Descent* (SGD) pode ser considerado como uma aproximação estocástica da otimização de gradiente descendente, uma vez que substitui o gradiente real, calculado a partir de todo o conjunto de dados, por uma estimativa do mesmo, calculada a partir de um subconjunto de dados selecionado aleatoriamente, reduzindo assim a carga computacional (SRA; NOWOZIN; WRIGHT, 2012). O SGD realiza atualizações em seus parâmetros de forma frequente e com uma alta variância, o que faz com que a função objetivo flutue fortemente. Essa flutuação permite que ele

[2] Adaptado de <https://sebastianraschka.com/faq/docs/gradient-optimization.html>. Acesso em: 21 fev. 2023.

eventualmente salte para novos e potencialmente melhores mínimos locais (RUDER, 2016).

O SGD tem a vantagem de ser computacionalmente eficiente, pois ele atualiza os pesos após cada amostra, e por isso não precisa armazenar todo o conjunto de dados na memória. Além disso, ele também é capaz de lidar com conjuntos de dados de grande escala e pode ser usado para treinar modelos online, ou seja, modelos que aprendem continuamente ao longo do tempo. A desvantagem é que o SGD é propenso a ficar preso em mínimos locais, o que pode ser mitigado através de técnicas como *momentum*, ou usando uma taxa de aprendizado adaptativa (DOGO et al., 2018).

2.3.5 Regressão Logística

Os modelos lineares são compostos por uma ou mais variáveis independentes que descrevem uma relação com uma variável de resposta dependente. Um dos modelos estatísticos lineares mais comumente utilizados para análise discriminante é a regressão logística (KIRASICH; SMITH; SADLER, 2018).

A regressão logística é uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de dados, um modelo que permita a predição de valores tomados por uma variável categórica a partir de uma série de variáveis explicativas (RYMARCZYK et al., 2019) Em um problema de classificação, a variável de saída pode assumir apenas valores discretos para um determinado conjunto de entradas. O modelo de regressão logística constrói um modelo de regressão para prever a probabilidade de uma determinada entrada de dados pertencer a uma categoria. Assim como a regressão linear assume que os dados seguem uma função linear, a regressão logística modela os dados usando a função sigmóide (LEVY; O'MALLEY, 2020).

A simplicidade e a interoperabilidade da regressão logística podem ocasionalmente levar à superação de outros modelos não lineares sofisticados. No entanto, no caso de a variável de resposta ser extraída de uma amostra pequena, os modelos de regressão logística tornam-se insuficientes e têm desempenho ruim para respostas binárias. Além disso, a regressão logística faz várias suposições, como independência entre as variáveis e distribuição normal e variância constante das respostas em todos os níveis da variável explicativa (KIRASICH; SMITH; SADLER, 2018).

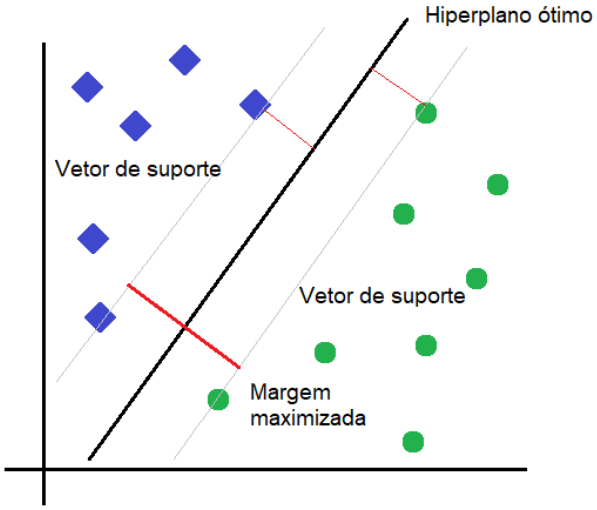
2.3.6 *Support Vector Machine*

O classificador *Support Vector Machine* (SVM) toma como entrada um conjunto de dados e prediz, para cada entrada dada, qual de duas possíveis classes a entrada faz parte. Dados um conjunto de exemplos de treinamento, cada um marcado como pertencente a uma de duas categorias, um algoritmo de treinamento do SVM constrói

um modelo que atribui novos exemplos a uma categoria ou outra (CORTES; VAPNIK, 1995).

Um modelo SVM é uma representação de exemplos como pontos no espaço, mapeados de maneira que os exemplos de cada categoria sejam divididos por um espaço claro que seja tão amplo quanto possível. Os novos exemplos são então mapeados no mesmo espaço e preditos como pertencentes a uma categoria baseados em qual o lado do espaço eles são colocados (SHALEV-SHWARTZ; BEN-DAVID, 2014). A Figura 8 representa como o algoritmo SVM consegue classificar os dados.

Figura 8 – Representação do algoritmo SVM e seus vetores de suporte na classificação.



Fonte: Adaptado de Cortes e Vapnik (1995)

O SVM pode usar diferentes funções *kernel* para mapear os dados para um espaço de dimensão superior, onde pode ser mais fácil encontrar um hiperplano de separação ótimo. Diferentes funções *kernel* também tornam o SVM capaz de lidar com dados não linearmente separáveis (BURGES, 1998). Alguns exemplos de funções *kernel* comuns incluem:

- Linear: é a função *kernel* mais simples, que pode ser utilizada quando os dados são linearmente separáveis. Essa função *kernel* realiza um produto interno entre dois vetores de entrada, transformando-os em um único valor escalar.

- Polinomial: utilizado quando os dados não são linearmente separáveis. Essa função *kernel* mapeia os dados para um espaço de dimensão superior através da elevação de cada característica a um grau específico. O grau do polinômio é um parâmetro do modelo que deve ser ajustado de acordo com o conjunto de dados.

- Gaussiano: também conhecido como RBF (*Radial Basis Function*), é uma função *kernel* amplamente utilizada. Essa função *kernel* mapeia os dados para um espaço de dimensão infinita através da utilização de uma função exponencial que mede a distância entre dois vetores de entrada. O parâmetro *sigma* controla a largura da função exponencial e deve ser ajustado de acordo com o conjunto de dados.

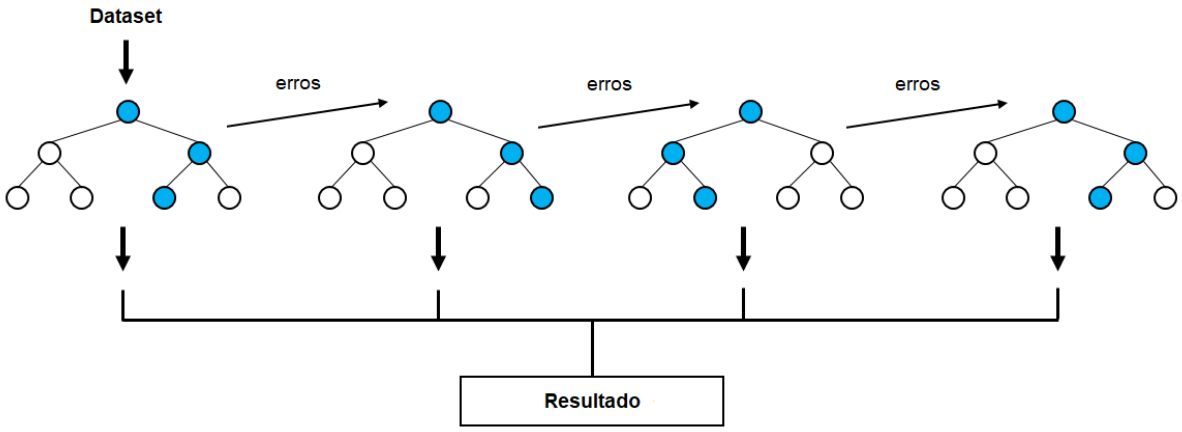
Além dessas funções *kernel*, existem outras opções disponíveis no SVM, como o *kernel* sigmóide, utilizado em tarefas de classificação binária, e o *kernel* Laplaciano, adequado para conjuntos de dados que possuem características esparsas e *outliers* (TING; ZHU; ZHOU, 2018).

O classificador SVM é um algoritmo de aprendizado de máquina altamente eficaz para problemas de classificação e regressão, especialmente quando as amostras não podem ser separadas linearmente. Ele funciona encontrando o hiperplano de separação que maximiza a margem entre as amostras de diferentes classes e usando vetores de suporte para definir o hiperplano. No entanto, ele pode ser sensível a ruído e dados desbalanceados, além de ter uma complexidade computacional elevada (BHAVSAR; PANCHAL, 2012).

2.3.7 *Light Gradient Boosting Machine*

Gradient Boosting Decision Tree é um algoritmo de *boosting* que combina sequencialmente várias árvores de decisão. Cada nova árvore é ajustada para se concentrar nos erros cometidos pelas árvores anteriores, criando um modelo final forte. Na Figura 9 é representado o funcionamento de *Gradient Boosting Decision Tree*.

Figura 9 – Arquitetura do algoritmo *Gradient Boosting Decision Tree*



Fonte: Autoria própria (2023)

Gradient Boosting Decision Tree é um algoritmo de aprendizado de máquina popular

e possui algumas implementações eficazes, contudo sua eficiência e escalabilidade ainda são insatisfatórias para *datasets* extensos (KE et al., 2017).

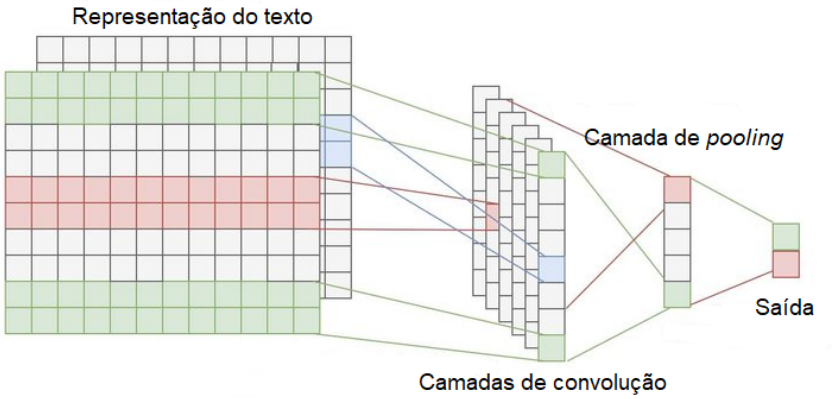
O algoritmo *Light Gradient Boosting Machine* (LightGBM) foi desenvolvido com a proposta de contornar as limitações do *Gradient Boosting Decision Tree*. O objetivo da *Light Gradient Boosting Machine* é reduzir o número de instâncias de dados e o número de recursos, acelerando assim o processo de treinamento. Para isso, duas técnicas foram criadas: *Gradient-based One-Side Sampling* (GOSS), que propõe a redução do volume de dados realizando amostragem aleatória nas instâncias com gradientes pequenos, enquanto mantém todas as instâncias com gradientes grandes, e o *Exclusive Feature Bundling* (EFB), que agrupa recursos exclusivos. Com estas abordagens, o LightGBM consegue resultados até 20 vezes mais rápido do que seu antecessor, com praticamente a mesma precisão (KE et al., 2017).

LightGBM é um algoritmo de aprendizado de máquina altamente eficiente, projetado para lidar com grandes conjuntos de dados e problemas de classificação e regressão. O LightGBM também oferece suporte para dados categóricos e é capaz de lidar com dados desbalanceados. Ele também tem suporte para distribuição de computação paralela para acelerar o treinamento em grandes conjuntos de dados (MACHADO; KARRAY; SOUSA, 2019).

2.3.8 *Redes Neurais Convolucionais*

Uma rede neural convolucional (*Convolutional Neural Networks*, ou CNN) é uma arquitetura de aprendizado profundo originalmente construída para processamento de imagens, porém efetiva para classificação de texto. As CNNs são baseadas em redes neurais tradicionais, mas incluem camadas adicionais de processamento, chamadas camadas de convolução, que são projetadas para identificar padrões e características nos dados de entrada, seguidos por camadas de *pooling*, que recebem cada saída do mapa de características da camada convolucional e preparam um mapa de características condensadas (KOWSARI et al., 2019). Na Figura 10 é representado a arquitetura de uma CNN.

Figura 10 – Representação da arquitetura de uma CNN



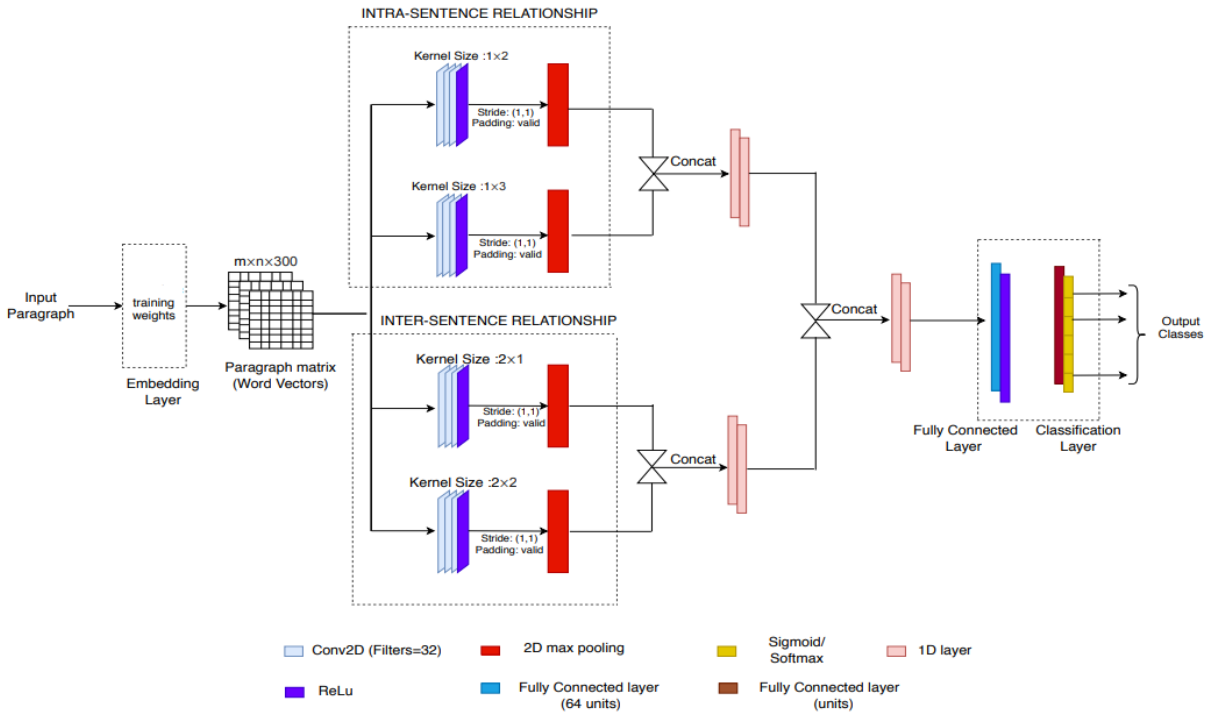
Fonte: Adaptado de Krešňáková, Sarnovsky e Butka (2019)

Um problema potencial que surge ao usar CNN para classificação de textos é o número de “canais” (tamanho do espaço de recursos). Enquanto as aplicações de classificação de imagem geralmente têm poucos canais (em geral apenas 3 canais de RGB), para aplicações de classificação de texto esse número pode ser muito grande, resultando em dimensionalidade muito alta (KOWSARI et al., 2019).

2.3.9 *TextConvoNet*

TextConvoNet é uma arquitetura proposta por Soni, Chouhan e Rathore (2022) baseada em CNN para classificação de textos. Diferentemente de outras arquiteturas geralmente utilizadas para processamento de linguagem natural, a rede *TextConvoNet* utiliza uma matriz bidirecional como entrada, gerado pelo vetorizador *GloVe*, e aplica uma operação convolucional multiescala. Na Figura 11 é representado a arquitetura da rede *TextConvoNet*.

Figura 11 – Arquitetura da Rede *TextConvoNet*



Fonte: Soni, Chouhan e Rathore (2022)

A *TextConvoNet* foi selecionada para este trabalho pois, segundo os autores, “supera os modelos de aprendizado de máquina e aprendizado profundo de última geração para fins de classificação de texto” (SONI; CHOUHAN; RATHORE, 2022).

2.4 MÉTRICAS DE AVALIAÇÃO

Classificadores geralmente são treinados para minimizar erros. Esse erro é avaliado usando uma ou várias métricas, cuja escolha tem sido um debate contínuo na pesquisa e na indústria há várias décadas (YACOUBY; AXMAN, 2020).

Dentre as métricas utilizadas na avaliação de classificadores, podemos citar a Acurária, Precisão, *Recall*, *F1-score*, *Receiver Operating Characteristics Curve* (ROC) e *Area Under Curve* (AUC). Acurácia é definida como a proporção de instâncias verdadeiras recuperadas, tanto positivas quanto negativas, entre todas as instâncias recuperadas. Precisão mede o número de instâncias corretas recuperadas dividido por todas as instâncias recuperadas. O *Recall*, também chamado de Cobertura ou Sensibilidade, mede o número de instâncias corretas recuperadas dividido por todas as instâncias corretas. O *F1-score* é a média harmônica entre precisão e *recall* (DALIANIS, 2018). Os gráficos ROC são gráficos bidimensionais que exibem a relação entre as instâncias verdadeiras positivas e falsas positivas. Por fim, a AUC trata-se da área sob a curva ROC, ou seja, uma representação numérica do mesmo indicador (ZHANG et al., 2015).

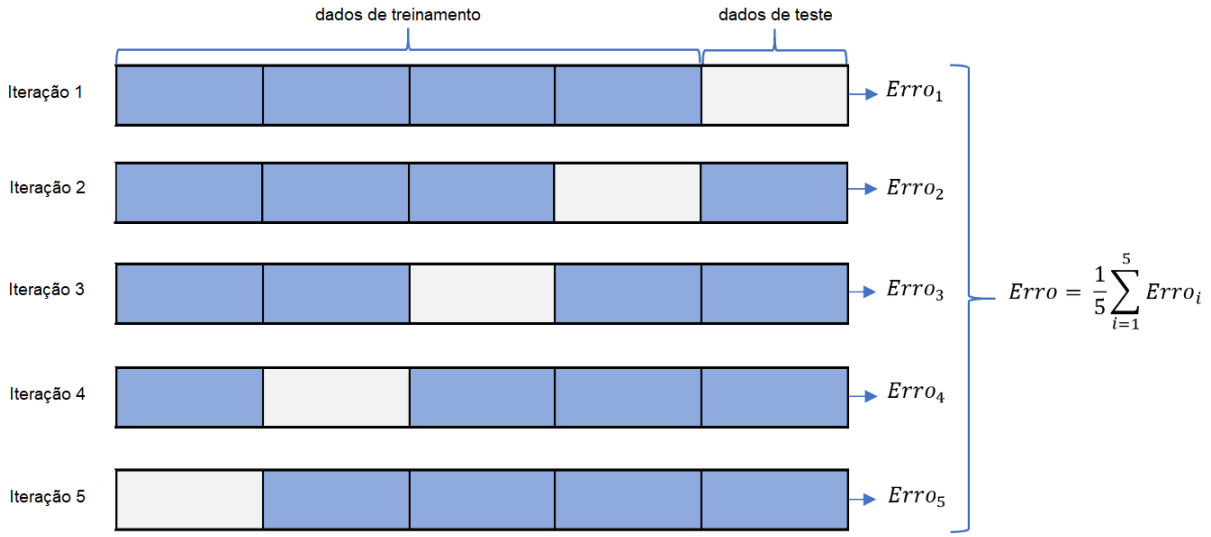
2.5 VALIDAÇÃO CRUZADA

Ao implantar um modelo preditivo, é importante entender sua precisão de previsão em testes futuros. Portanto, é essencial que os resultados reflitam boas estimativas e intervalos de confiança precisos. A validação cruzada é uma abordagem amplamente utilizada para essas duas tarefas (BATES; HASTIE; TIBSHIRANI, 2021).

A validação cruzada opera dividindo o *dataset* em dois segmentos: um usado para treinar e o outro usado para testar o modelo. Os conjuntos de treinamento e teste devem ser cruzados em rodadas sucessivas, de modo que cada ponto de dados tenha a chance de ser testado. A forma básica de validação cruzada é a validação cruzada *k-fold*. Os dados são primeiro particionados em k segmentos ou dobras de tamanho similar. Em seguida, k iterações de treinamento e teste são realizadas de tal forma que dentro de cada iteração uma dobra diferente dos dados é mantida para teste, enquanto as $k - 1$ dobras restantes são usadas para aprendizado. Após a conclusão, k amostras da métrica de desempenho estarão disponíveis para cada algoritmo. Diferentes metodologias, como a média, podem ser usadas para obter uma medida agregada dessas amostras, ou essas amostras podem ser usadas em um teste de hipótese estatística para mostrar que um algoritmo é superior a outro. (REFAEILZADEH; TANG; LIU, 2016).

O funcionamento da validação cruzada está representado pela Figura 12.

Figura 12 – Validação Cruzada, com k = 5



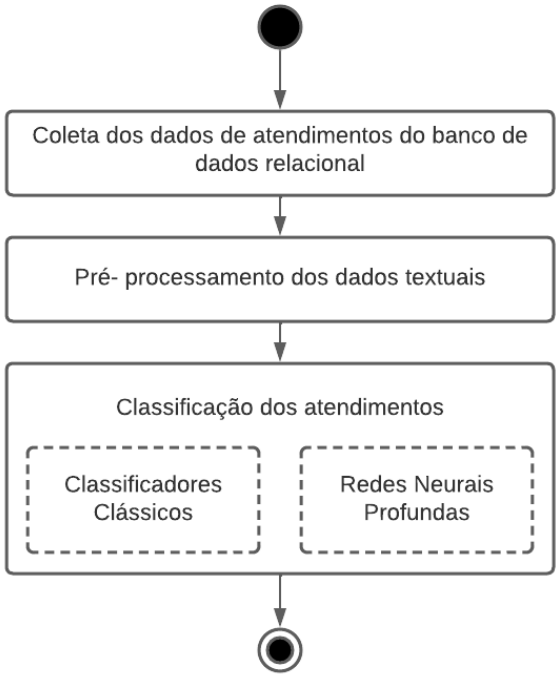
Fonte: Adaptado de towardsdatascience.com[3]

[3] Adaptado de <https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-e54df2fc179b>. Acesso em: 25 fev. 2023.

3 METODOLOGIA

Neste capítulo são apresentados os materiais e os métodos utilizados neste trabalho. Na Figura 13 é apresentado um fluxograma das etapas do desenvolvimento.

Figura 13 – Fluxograma do Trabalho.



Fonte: Autoria própria (2023)

3.1 AMBIENTE DE DESENVOLVIMENTO

3.1.1 Extração dos dados

O sistema de *Help Desk* utiliza um banco de dados relacional Oracle para a persistência dos dados. A extração dos dados do sistema foi realizada utilizando-se de comandos SQL na ferramenta *SQL Developer*, da própria Oracle. Os dados foram extraídos e convertidos em uma planilha CSV.

3.1.2 Processamento dos dados

Para o processamento dos dados foi utilizada a linguagem de programação Python no ambiente de desenvolvimento Google Colaboratory (Colab). O Colab é uma plataforma de desenvolvimento criada pelo Google, que permite a utilização de códigos em Python com acesso gratuito a GPUs e TPUs de alto desempenho.

Para a implementação dos testes foram utilizadas diversas bibliotecas, as principais listadas abaixo:

- *lightgbm*, para o classificador LightGBM,
- *sklearn*, para os demais classificadores clássicos,
- *gensim*, para os algoritmos de *word embedding*, e
- *tensorflow*, para a implementação das redes CNN.

3.2 MÉTODOS PROPOSTOS

Inicialmente, foi realizada a extração de dados do sistema de *Help Desk*. O sistema utiliza um banco de dados relacional Oracle, e a extração foi realizada utilizando-se comandos SQL. Dados sensíveis, como nomes de clientes, empresas, usuários e informações pessoais foram manualmente removidos no processo de extração. Desta forma foram coletados dados dos atendimentos realizados entre os anos de 2019 e 2020, totalizando 11449 atendimentos. Uma vez extraídos, os dados foram convertidos em uma planilha para permitir a importação no ambiente de desenvolvimento Colab.

Em seguida, foi realizada uma breve análise exploratória dos dados. Com o apoio visual de uma nuvem de palavras e de alguns contadores aplicados ao dataset, algumas características se evidenciaram e direcionaram o próximo passo de pré-processamento.

No passo seguinte, foi feito o pré-processamento dos dados. Foram removidos caracteres especiais, acentuações e o texto, originalmente todo em maiúsculo, foi convertido para minúsculo. Foram removidos os termos recorrentes sem valor semântico, como saudações e termos de encerramento (“bom dia”, “atenciosamente” etc), além do chamado “*stopwords*”, conjunto composto por preposições e artigos. Alguns sinônimos recorrentes também foram substituídos por um único termo (por exemplo, os termos “nota fiscal”, “nf-e”, “nota de venda”, “nota de mercadoria”, “nota eletrônica” foram substituídos por “nfe”). Uma nova análise do *dataset* evidenciou uma melhora significativa na distribuição das palavras.

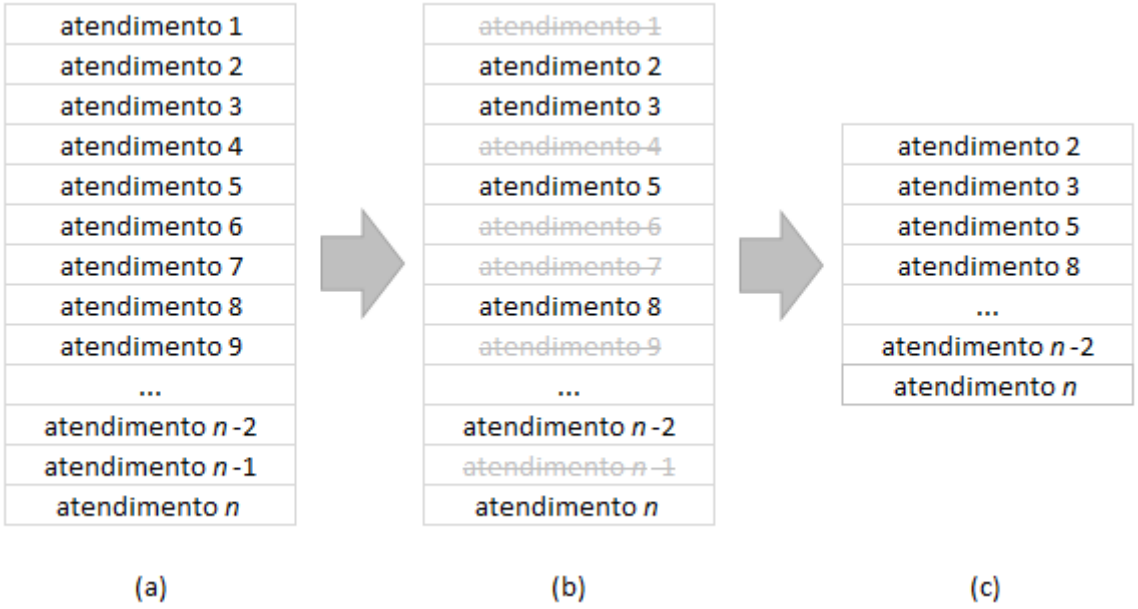
Em seguida, foi feita a conversão do conteúdo da base de dados em correspondentes numéricos. Primeiramente cada texto foi convertido em um vetor de palavras. Em seguida, os algoritmos *TF-IDF* e *Doc2Vec* foram aplicados separadamente ao *dataset* original, gerando dois conjuntos distintos de dados numéricos representando os documentos. Um banco de vetores de palavras pré processado, o *GloVe*, também foi utilizado como requisito da arquitetura do *TextConvoNet*.

Finalmente, a etapa seguinte consiste nos testes dos algoritmos de classificação. Os testes foram realizados em quatro etapas: Vetorizador TF-IDF x Classificadores Clássicos, *word embedding Doc2Vec* x Classificadores Clássicos, classificador

TextConvoNet e uma proposta de modificação no *TextConvoNet*, aplicando como entrada o resultado do *word embedding Doc2Vec*. Várias métricas de desempenho foram aplicadas e sintetizadas em gráficos e tabelas para melhor comparação dos resultados.

A Figura 14 simboliza a classificação do *dataset* original (a), a classificação e descarte dos elementos não relevantes (b) e a geração de um novo *dataset* de atendimentos relevantes (c).

Figura 14 – Classificação do *dataset*.



Fonte: Autoria própria (2023)

3.3 BASE DE DADOS

A base de dados a ser utilizada foi coletada de um banco de dados relacional do sistema *Help Desk Conexos*, *software* interno da empresa Conexos Consultoria e Sistemas LTDA. Este sistema está em funcionamento desde o ano de 2004, contendo uma considerável coleção de atendimentos prestados pela empresa a seus clientes.

O fluxo de interações dos atendimentos deste sistema é similar ao fluxo de fóruns online: a partir de uma solicitação do cliente, que pode ser um questionamento, relato de um problema ou uma solicitação de incremento no sistema, podem haver diversas respostas por parte da prestadora de serviços, bem como diversas réplicas e tréplicas até que o cliente esteja com a solicitação resolvida, momento no qual o atendimento é finalizado. Este formato permite a geração de atendimentos com fluxos complexos, onde frequentemente ocorrem oscilações entre assuntos distintos, dificultando sua classificação.

Diante do exposto foram selecionados, para os testes iniciais, somente os atendimentos onde o fluxo de atendimento é composto por uma pergunta do cliente, uma resposta do prestador de serviços e a finalização do atendimento. Atendendo às condições estabelecidas, foram coletados dados dos atendimentos realizados entre os anos de 2019 e 2020, totalizando 11449 atendimentos. Atendendo às diretrizes da Lei Geral de Proteção dos Dados (BRASIL, 2018), os dados foram anonimizados, de forma que é impossível identificar os autores ou quaisquer informações pessoais nos dados coletados.

3.4 ANÁLISE EXPLORATÓRIA DOS DADOS

A Tabela 1 traz alguns dados estatísticos das palavras que compõem o *dataset*. Entre as características notáveis destaca-se a alta ocorrência de repetição de palavras nos textos.

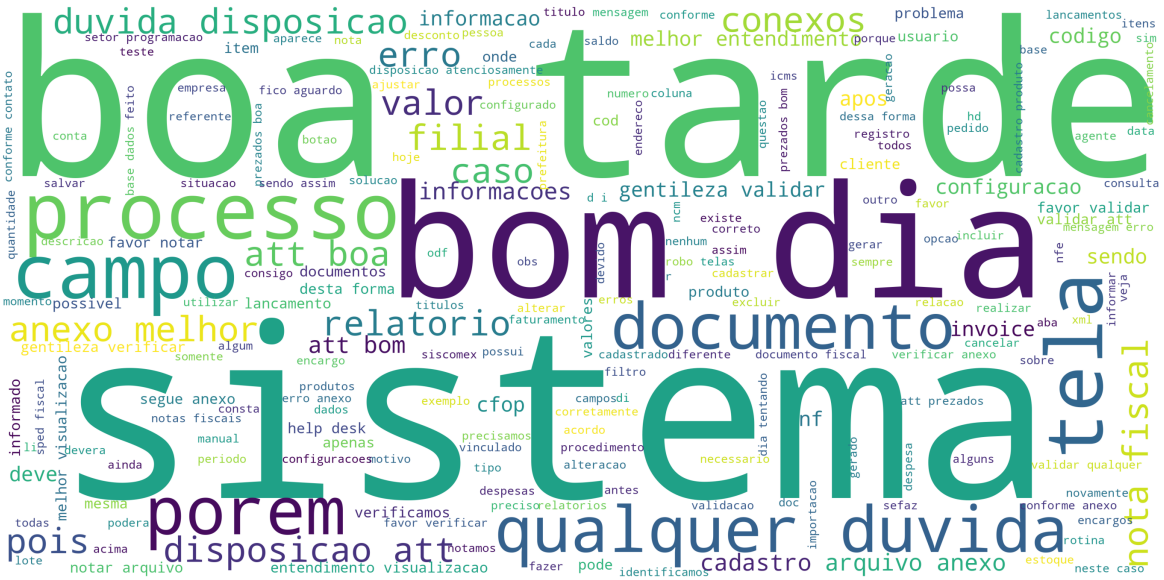
Tabela 1 – Informações descritivas do *dataset*.

Indicador	**Quantidade**
Total de Atendimentos	11449
Total de Palavras	953728
Média de Palavras por Atendimento	83,30
Palavras Distintas	29408
Média de Palavras Distintas por Atendimento	28,90

Fonte: Autoria própria (2023)

Uma ferramenta intuitiva que permite rápida análise de ocorrências de palavras em um documento é a chamada Nuvem de Palavras. Uma Nuvem de Palavras é uma representação visual da frequência das palavras, de forma que quanto mais comumente o termo aparecer dentro do texto que está sendo analisado, maior a palavra aparecerá na imagem gerada (ATENSTAEDT, 2012). Na Figura 15 é exibido a Nuvem de Palavras para o *dataset*.

Figura 15 – Nuvem de Palavras que compõem o *dataset*.



Fonte: Autoria própria (2023)

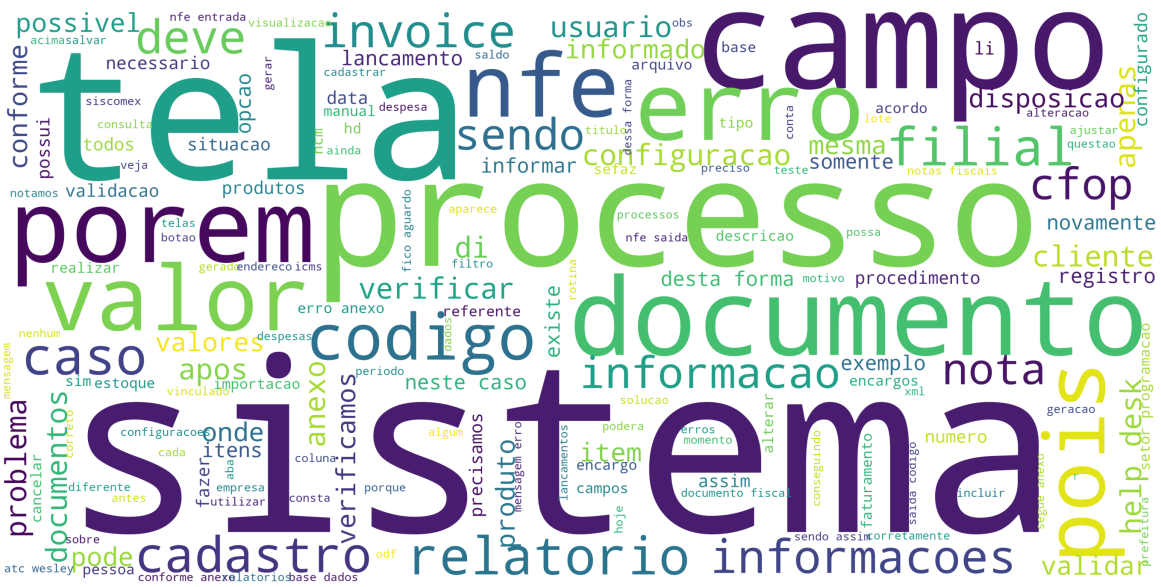
A Nuvem de Palavras evidencia uma importante característica do *dataset* em questão: a presença expressiva de saudações e termos comuns de apresentação e encerramento, tais como “bom dia”, “prezados” e “gentileza validar”. Para Hajrizi e Nuçi (2020) estes termos, apesar de representarem um gesto respeitoso na comunicação humana, se tornam ruído para os algoritmos de processamento de texto, afetando seu desempenho. Portanto, parece razoável afirmar que tais termos devam ser removidos do *dataset* antes da classificação, visando concentrar as informações e evitar possíveis vieses no treinamento dos classificadores. Essa remoção será detalhada no tópico seguinte.

3.5 PRÉ-PROCESSAMENTO DOS DADOS

O pré-processamento de texto é uma etapa essencial em qualquer Sistema de Processamento de Linguagem Natural. Esta etapa envolve sua transformação ante sua análise, e engloba estratégias como: remover conteúdo irrelevante para a tarefa, aglomerar termos semanticamente relacionados, e aumentar a quantidade de informações semânticas capturadas (HICKMAN et al., 2022).

Na Figura 16 é representado a Nuvem de Palavras após a remoção de saudações e termos comuns de apresentação e encerramento. Pode-se notar uma melhor distribuição de termos com sentido contextual, como as palavras “processo”, “sistema”, “relatório” entre outras.

Figura 16 – Nuvem de Palavras que compõem o *dataset* após remoção de saudações.



Fonte: Autoria própria (2023)

Para o treinamento dos algoritmos de classificação, é necessário que os dados estejam previamente classificados, para que os algoritmos possam se embasar durante treinamento e validação dos resultados obtidos. Para tal, um subconjunto atendimentos foi classificado, de forma a compor o *dataset* de treinamento. Devido a limitações de mão de obra qualificada disponível para esta tarefa, a classificação manual foi realizada por um único profissional da empresa. O profissional designado para a tarefa possui 15 anos de experiência na empresa, em trabalho direto com o sistema de *Help Desk*.

A distribuição não balanceada entre as classes no *dataset* faz com que os modelos do classificador tendam para classes com muitas amostras de treinamento (OKKALIOGLU; OKKALIOGLU, 2022). Desta forma, os atendimentos foram selecionados de forma a compor um *dataset* perfeitamente balanceado. Esta decisão foi tomada para uniformização dos testes, pois alguns dos algoritmos do estudo não apresentam bons resultados em conjuntos de dados desbalanceados, como pode ser visto nos trabalhos de Chawla e Sylvester (2007), Pozzolo et al. (2015) e Sundarkumar e Ravi (2015). Contudo, é importante frisar que essa situação de balanceamento não ocorre no *dataset* original, onde foi constatado que cerca de 30% dos textos são relevantes.

Na Tabela 2 verifica-se informações estatísticas do subconjunto para treinamento. Uma característica notável é que, em média, atendimentos relevantes possuem uma quantidade de palavras maior do que os atendimentos não relevantes.

Tabela 2 – Informações descritivas do *dataset* de treinamento.

Indicador	**Total**	**Positivos**	**Negativos**
Número de Atendimentos	1084	542	542
Total de Palavras	88244	52347	35897
Palavras por Atendimento	81,41	96,58	66,23
Palavras Distintas	8355	6167	5194
Palavras Distintas por Atendimento	28,92	29,88	27,97

Fonte: Autoria própria (2023)

Na Tabela 3 é exibida a lista das palavras mais usadas nos *dataset* em sua íntegra, enquanto nas Tabelas 4 e 5 são exibidas as listas das palavras mais usadas no subconjunto de atendimentos relevantes e não relevantes, respectivamente, do mesmo *dataset*. Pode-se verificar que há uniformidade na distribuição de frequência das palavras. Nota-se também que os *datasets* relevante e não relevante apresentam muitas palavras em comum.

Tabela 3 – Palavras mais recorrentes no *dataset* integral após remoção de saudações.

Palavra	**Ocorrências**	**% do Total**
sistema	1099	1,32
tela	983	1,18
erro	820	0,98
documento	764	0,92
nfe	715	0,86
processo	556	0,67
caso	544	0,65
codigo	526	0,63
anexo	494	0,59
campo	460	0,55

Fonte: Autoria própria (2023)

Tabela 4 – Palavras mais recorrentes no *dataset* relevante após remoção de saudações.

Palavra	**Ocorrências**	**% do Total**
sistema	677	1,35
tela	670	1,33
documento	605	1,20
nfe	551	1,09
erro	457	0,91
processo	391	0,78
codigo	349	0,69
caso	342	0,68
campo	327	0,65
cadastro	323	0,64

Fonte: Autoria própria (2023)

Tabela 5 – Palavras mais recorrentes no *dataset* não relevante após remoção de saudações.

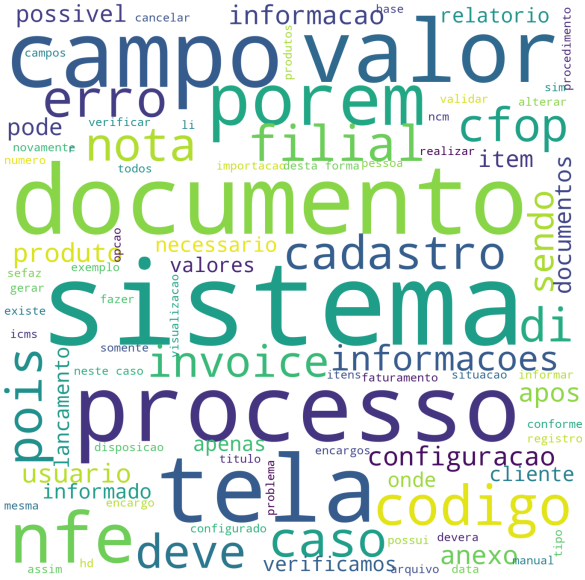
Palavra	**Ocorrências**	**% do Total**
sistema	422	1,28
erro	363	1,10
tela	313	0,95
anexo	228	0,69
base	212	0,64
disposicao	212	0,64
relatorio	208	0,63
caso	202	0,61
campo	190	0,57
verificar	185	0,56

Fonte: Autoria própria (2023)

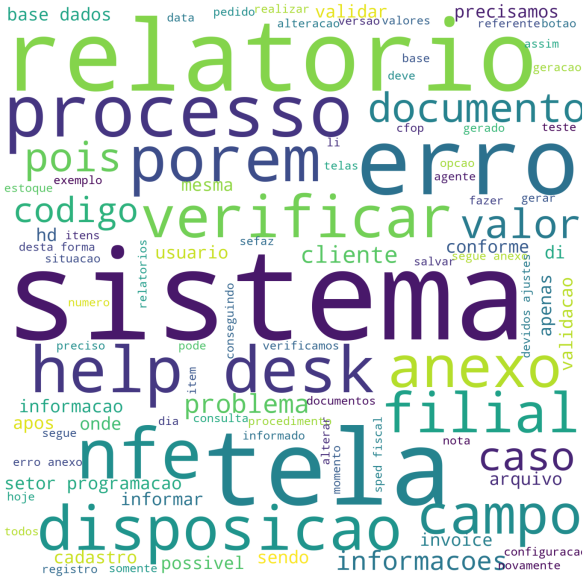
A Figura 17 representa a nuvem de palavras individuais para o *dataset* de atendimentos relevantes (a) e para o *dataset* de atendimentos não relevantes (b).

Figura 17 – Nuvem de Palavras que compõem o *dataset* por categoria.

(a) *Dataset* de atendimentos relevantes.



(b) *Dataset* de atendimentos não relevantes.



Fonte: Autoria própria (2023)

4 RESULTADOS

Para os testes foram utilizados dois subconjuntos do *dataset* original. O primeiro composto por 1084 atendimentos, e o segundo composto por 1868, sendo este uma expansão do *dataset* anterior. Os *datasets* foram manualmente classificados e balanceados.

4.1 VETORIZADOR TF-IDF X CLASSIFICADORES CLÁSSICOS

A seguir serão apresentados os resultados utilizando-se para vetorização o algoritmo TF-IDF e para classificação os algoritmos *Adaptive Boosting*, *Light Gradient Boosting Machine*, Regressão Logística, Naive-Bayes, Floresta Aleatória, *Stochastic Gradient Descent* e *Support Vector Machine*.

Alguns classificadores possuem múltiplas implementações e diversos parâmetros de entrada. A Tabela 6 exibe as classes python e os parâmetros adicionais utilizados em todos os testes deste trabalho.

Tabela 6 – Implementação dos algoritmos de classificação.

Classificador	**Classe de implementação**	**Parâmetros Adicionais**
AdaBoost	AdaBoostClassifier()	padrão
LightGBM	LGBMClassifier()	padrão
Regressão Logística	LogisticRegression()	multi_class="ovr", solver="newton-cg"
Naive-Bayes	MultinomialNB()	padrão
Floresta Aleatória	RandomForestClassifier()	criterion="gini"
SGD	SGDClassifier()	loss='log'
SVM	SVC()	kernel="linear"

Fonte: Autoria própria (2023)

Os testes foram realizados com validação cruzada, com k = 10.

4.1.1 Base de dados com 1084 atendimentos

Na Tabela 7 são exibidos os valores das principais métricas dos classificadores, obtidos através de experimentos com validação cruzada realizados com o conjunto de dados.

Tabela 7 – Métricas de alguns classificadores clássicos quando aplicados ao conjunto de dados.

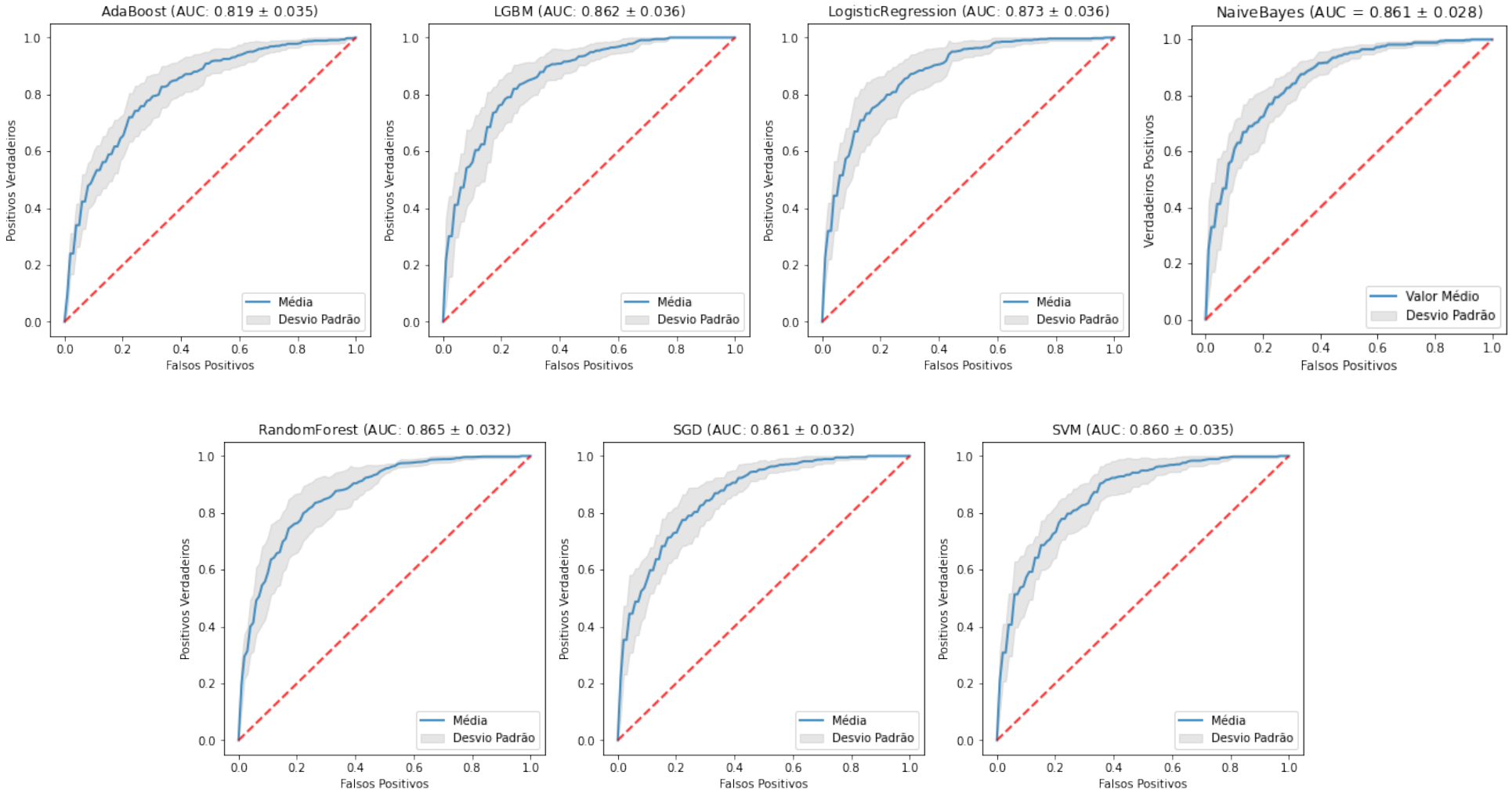
Classificador	**Acurácia**	**Precisão**	***Recall***	***F1-Score***
AdaBoost	0,736 ± 0,044	0,731 ± 0,043	0,749 ± 0,062	0,739 ± 0,045
LightGBM	0,786 ± 0,042	**0,782 ± 0,067**	0,806 ± 0,058	0,791 ± 0,037
Regressão Logística	0,781 ± 0,053	0,772 ± 0,061	0,804 ± 0,077	0,785 ± 0,054
Naive-Bayes	0,763 ± 0,051	0,739 ± 0,051	0,819 ± 0,068	0,775 ± 0,048
Floresta Aleatória	**0,791 ± 0,036**	0,771 ± 0,052	**0,836 ± 0,052**	**0,800 ± 0,034**
SGD	0,744 ± 0,037	0,739 ± 0,059	0,775 ± 0,075	0,752 ± 0,032
SVM	0,778 ± 0,048	0,759 ± 0,046	0,817 ± 0,087	0,785 ± 0,052

Fonte: Autoria própria (2023)

Verifica-se que os classificadores do experimento apresentaram métricas similares, com destaque para o classificador Floresta Aleatória que obteve os melhores resultados.

Na Figura 18 é exibida a curva ROC de cada algoritmo testado. A área em cinza corresponde ao desvio padrão. Nota-se que apesar de um valor de AUC acima de 0,8, os resultados possuem um desvio padrão significativo.

Figura 18 – Curvas ROC com desvio-padrão - TF-IDF x Classificadores Clássicos - Dataset com 1084 atendimentos



Fonte: Autoria própria (2023)

4.1.2 Base de dados com 1868 atendimentos

Na Tabela 8 são exibidos os valores das principais métricas dos classificadores, obtidos através de experimentos com validação cruzada realizados com o conjunto de dados.

Tabela 8 – Métricas de classificadores clássicos em conjunto com o vetorizador TF-IDF.

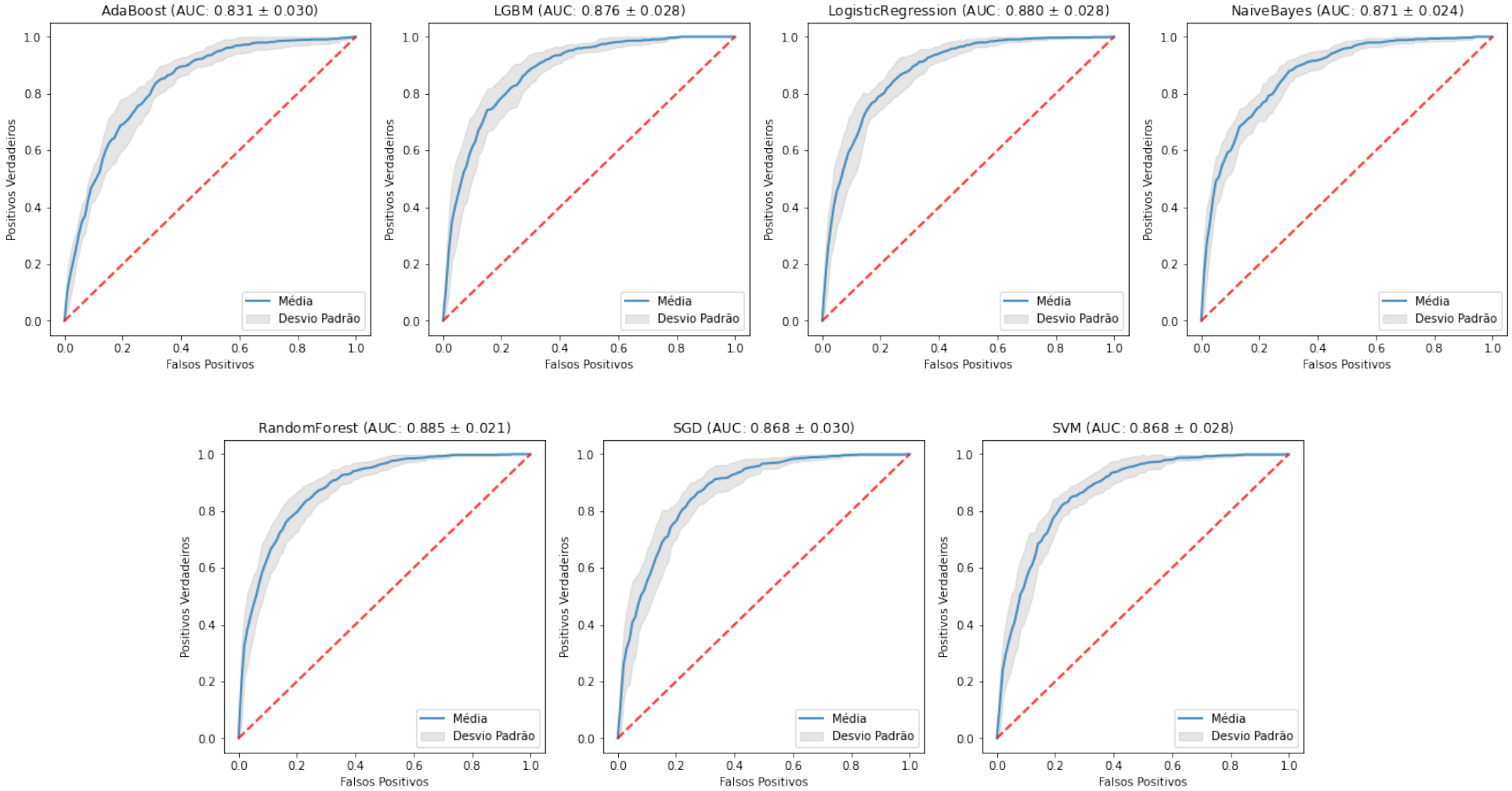
Classificador	**Acurácia**	**Precisão**	***Recall***	***F1-Score***
AdaBoost	0,750 ± 0,028	0,743 ± 0,039	0,770 ± 0,049	0,755 ± 0,029
LightGBM	0,795 ± 0,031	**0,781 ± 0,037**	0,823 ± 0,031	0,801 ± 0,029
Regressão Logística	0,795 ± 0,029	0,780 ± 0,027	0,824 ± 0,045	0,801 ± 0,030
Naive-Bayes	0,792 ± 0,030	0,764 ± 0,029	0,846 ± 0,041	0,803 ± 0,030
Floresta Aleatória	**0,804 ± 0,028**	0,778 ± 0,032	**0,851 ± 0,027**	**0,813 ± 0,026**
SGD	0,761 ± 0,029	0,752 ± 0,027	0,779 ± 0,045	0,765 ± 0,031
SVM	0,798 ± 0,024	0,779 ± 0,029	0,835 ± 0,038	0,805 ± 0,024

Fonte: Autoria própria (2023)

Verifica-se melhora em praticamente todos os indicadores, bem como no desvio padrão das métricas da validação cruzada. O classificador Floresta Aleatória manteve-se com os melhores resultados. Na Figura 19 é exibida a curva ROC de cada algoritmo testado para o novo *dataset*. Em uma comparação visual com a Figura 18 é notável a redução do desvio padrão representado pela área em cinza dos gráficos.

Dado estas características, conclue-se que o aumento do *dataset* de treinamento melhorou sua representatividade do problema. Portanto, nos próximos testes somente a base maior será utilizada.

Figura 19 – Curvas ROC com desvio-padrão - TF-IDF x Classificadores Clássicos - Dataset com 1868 atendimentos



Fonte: Autoria própria (2023)

4.2 *EMBEDDING* DOC2VEC X CLASSIFICADORES CLÁSSICOS

Na Tabela 9 são apresentados os resultados utilizando-se como *embedding* o algoritmo *Doc2Vec* e para classificação os algoritmos *Adaptive Boosting*, *Light Gradient Boosting Machine*, Regressão Logística, Naive-Bayes, Floresta Aleatória, *Stochastic Gradient Descent* e *Support Vector Machine*. Os testes foram realizados com validação cruzada, com k = 10.

Tabela 9 – Métricas de classificadores clássicos em conjunto com o vetorizador Doc2Vec.

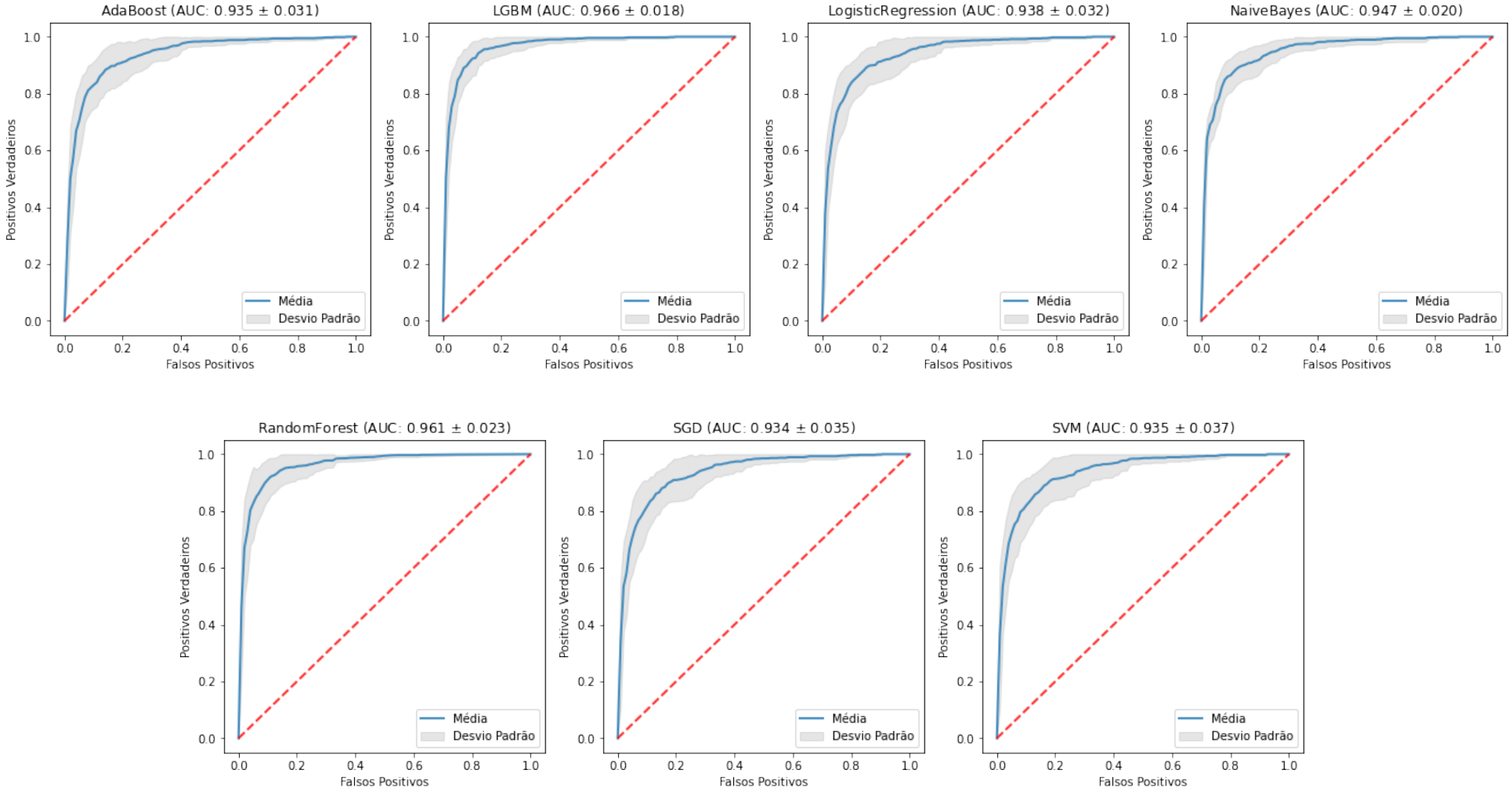
Classificador	**Acurácia**	**Precisão**	***Recall***	***F1-Score***
AdaBoost	0,862 ± 0,040	0,870 ± 0,043	0,852 ± 0,070	0,859 ± 0,045
LightGBM	**0,922 ± 0,039**	**0,928 ± 0,027**	0,914 ± 0,056	**0,921 ± 0,042**
Regressão Logística	0,858 ± 0,042	0,865 ± 0,037	0,848 ± 0,065	0,855 ± 0,045
Naive-Bayes	0,860 ± 0,040	0,873 ± 0,065	0,850 ± 0,031	0,860 ± 0,035
Floresta Aleatória	0,920 ± 0,027	0,916 ± 0,028	**0,925 ± 0,033**	0,920 ± 0,027
SGD	0,822 ± 0,065	0,830 ± 0,086	0,839 ± 0,156	0,819 ± 0,087
SVM	0,862 ± 0,046	0,869 ± 0,048	0,852 ± 0,057	0,860 ± 0,048

Fonte: Autoria própria (2023)

Percebe-se uma melhora significativa na maioria dos indicadores de todos os classificadores, onde o classificador *LightGBM* e Floresta Aleatória alcançaram métricas em torno de 0,92. Utilizando a vetorização com o *Doc2Vec*, o classificador *LightGBM* apresentou os melhores resultados dentre os algoritmos de classificação utilizados.

Na Figura 20 é exibida a curva ROC de cada algoritmo testado para o vetorizador *Word2Vec*. Em uma comparação visual com a Figuras 18 e 19 nota-se uma melhora significativa do classificador, bem como uma redução do desvio padrão representado pela área em cinza dos gráficos.

Figura 20 – Curvas ROC com desvio-padrão - Doc2Vec x Classificadores Clássicos - Dataset com 1868 atendimentos



Fonte: Autoria própria (2023)

4.3 TEXTCONVONET

Na Tabela 10 são apresentados os resultados do algoritmo *TextConvoNet*.

Tabela 10 – Métricas do classificador TextConvoNet.

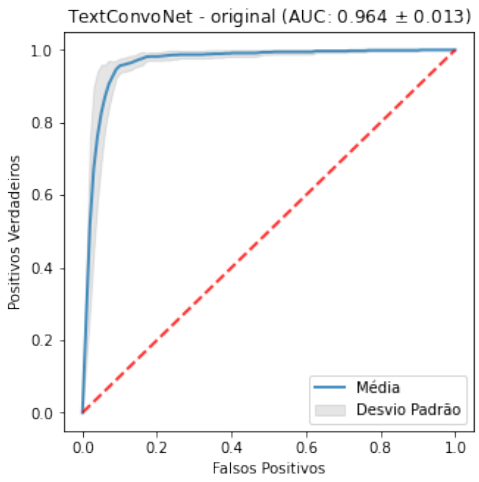
Classificador	**Acurácia**	**Precisão**	***Recall***	***F1-Score***
TextConvoNet	0,928 ± 0,017	0,925 ± 0,026	0,931 ± 0,020	0,928 ± 0,018

Fonte: Autoria própria (2023)

O *TextConvoNet* teve métricas próximas a 0,93, superiores às métricas dos classificadores clássicos.

Na Figura 21 é exibida a curva ROC do algoritmo *TextConvoNet*. Em uma comparação visual com a Figuras 19 e 20 nota-se uma melhora significativa do classificador, bem como uma redução do desvio padrão representado pela área em cinza dos gráficos.

Figura 21 – Curva ROC com desvio-padrão - *TextConvoNet*



Fonte: Autoria própria (2023)

4.4 TEXTCONVONET MODIFICADO

Na Tabela 11 são apresentados os resultados do algoritmo *TextConvoNet* modificado. O algoritmo *TextConvoNet* original espera como entrada um vetor bidimensional gerado pelo vetorizador *GloVe*. As alterações realizadas foram somente para que o algoritmo receba como entrada um vetor unidimensional, resultado do *embedding Doc2Vec*. Para tal, as camadas de convolução e *pooling*, antes operando em duas dimensões, foram substituídas por camadas de convolução e *pooling* para operar em dimensão. A motivação por trás desta modificação é em comparar os resultados deste classificador de forma mais fiel aos classificadores clássicos, utilizando-se do mesmo algoritmo de vetorização.

Tabela 11 – Métricas do classificador TextConvoNet modificado em conjunto com o vetorizador Doc2Vec.

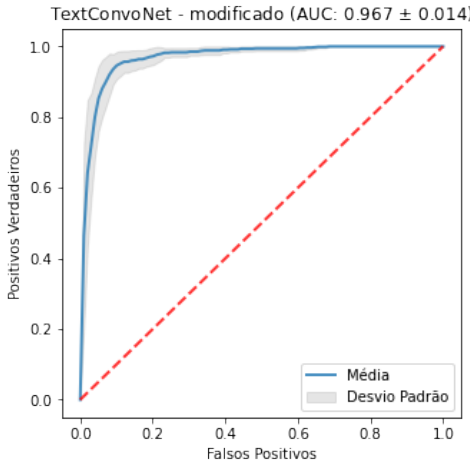
Classificador	**Acurácia**	**Precisão**	***Recall***	***F1-Score***
TextConvoNet modificado	0,925 ± 0,024	0,931 ± 0,036	0,922 ± 0,037	0,925 ± 0,024

Fonte: Autoria própria (2023)

O *TextConvoNet* modificado apresentou métricas superiores a 0,92, melhores do que as métricas dos classificadores clássicos e similares à implementação original do mesmo algoritmo.

Na Figura 22 é exibida a curva ROC do algoritmo *TextConvoNet* modificado para o vetorizador *Word2Vec*. Em uma comparação visual com a Figura 21 verifica-se que há muita similaridade entre os resultados do classificador modificado e do classificador original.

Figura 22 – Curva ROC com desvio-padrão - *TextConvoNet* modificado



Fonte: Autoria própria (2023)

4.5 SÍNTESE DOS RESULTADOS

Neste tópico são exibidos os principais resultados dos testes, a fim de ter uma melhor visualização do progresso nos resultados. A Tabela 12 apresenta as métricas dos algoritmos com melhores resultados.

Tabela 12 – Métricas dos melhores classificadores utilizados.

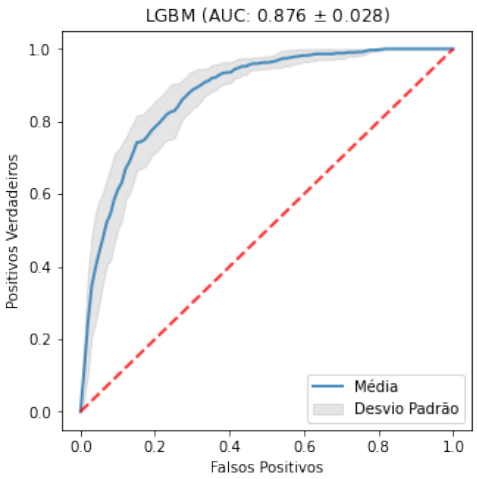
Classificador	**Acurácia**	**Precisão**	***Recall***	***F1-Score***
TD-IDF x *LightGBM*	0,795 ± 0,031	0,781 ± 0,037	0,823 ± 0,031	0,801 ± 0,029
TD-IDF x *Flor. Aleatória*	0,804 ± 0,028	0,778 ± 0,032	0,851 ± 0,027	0,813 ± 0,026
Doc2Vec x *LightGBM*	0,922 ± 0,039	0,928 ± 0,027	0,914 ± 0,056	0,921 ± 0,042
Doc2Vec x *Flor. Aleatória*	0,920 ± 0,027	0,916 ± 0,028	0,925 ± 0,033	0,920 ± 0,027
TextConvoNet	**0,928 ± 0,017**	0,925 ± 0,026	**0,931 ± 0,020**	**0,928 ± 0,018**
TextConvoNet modificado	0,925 ± 0,024	**0,931 ± 0,036**	0,922 ± 0,037	0,925 ± 0,024

Fonte: Autoria própria (2023)

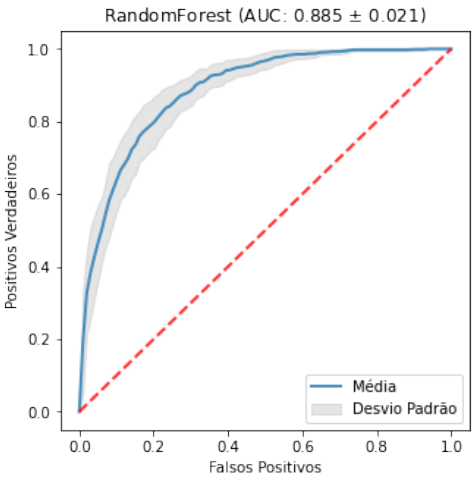
Na Figura 22 é exibida a curva ROC dos algoritmos com melhores resultados.

Figura 23 – Curvas ROC com desvio-padrão - Melhores classificadores

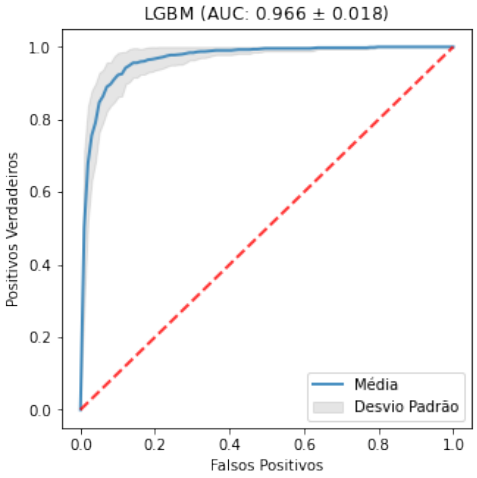
(a) TF-IDF + LightGBM



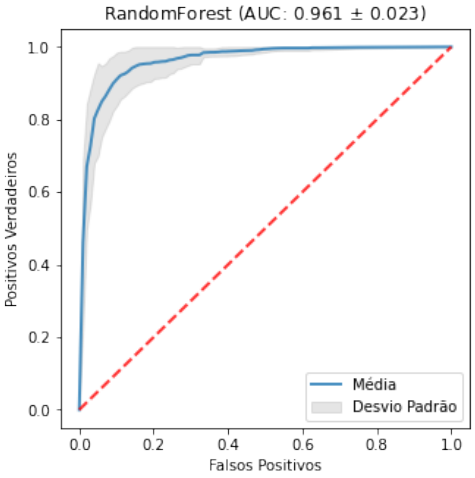
(b) TF-IDF + Floresta Aleatória



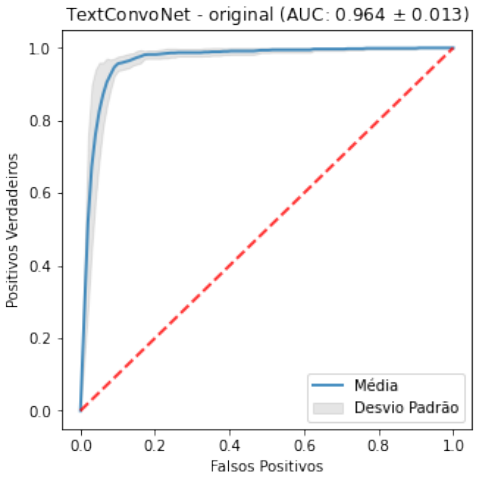
(c) Doc2Vec + LightGBM



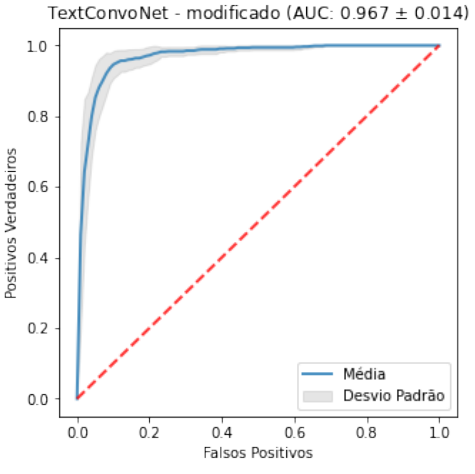
(d) Doc2Vec + Floresta Aleatória



(e) TextConvoNet - original



(f) TextConvoNet - modificado



Fonte: Autoria própria (2023)

Com estes dados, verifica-se que:

1. Os algoritmos de *word embedding* produziram resultados superiores ao algoritmo de vetorização TF-IDF. Em outras palavras, as estratégias de *word embedding* conseguiram manter as características de relevância contidas no texto durante sua conversão para a representação numérica de forma superior à estratégia do algoritmo TF-IDF. De fato, a Análise Exploratória dos Dados sugeriu que a frequência de palavras não era um critério significativo para a identificação de relevância, característica esta que é explorada pelo algoritmo TF-IDF.

2. Os classificadores clássicos com melhores resultados, independente da estratégia de representação numérica dos textos utilizada, foram o *LightGBM* e a Floresta Aleatória, ambos algoritmos de *boosting* baseados em árvores de decisão. Árvores de decisão são conhecidas por apresentarem bons resultados em modelos não-lineares e/ou de classificação binária.

3. Os melhores classificadores clássicos apresentaram bons resultados, com valores inferiores, mas próximos, à rede *TextConvoNet*, uma arquitetura baseada em Redes Convolucionais.

4. Os algoritmos de *word embedding GloVe* e *Word2Vec* apresentaram resultados muito próximos aplicados a algoritmos baseados em Redes Convolucionais, apesar do algoritmo *TextConvoNet* utilizar originalmente como entrada um vetor bidimensional do *GloVe* para análise multiescala. Isso se deve ao fato de que o *Doc2Vec* sintetiza a multidimensionalidade do *Word2Vec* em um resultado unidimensional, sem grandes perdas semânticas.

5 CONSIDERAÇÕES FINAIS

A definição de “relevância” do conteúdo, apesar da aparente simplicidade por trás da classificação binária, é um conceito abstrato. A análise exploratória inicial do *dataset* mostrou que as características que identificam a relevância são sutis, e envolvem mais do que somente a presença ou não de palavras no texto.

Por limitações operacionais, o *dataset* de treinamento foi classificado somente por um humano. A classificação de relevância do *dataset* de treinamento pode apresentar vieses pois representa a opinião pessoal de quem realizou a classificação. Apesar disso, é importante destacar que esses vieses do *dataset* não invalidam os resultados de classificação, que se mostraram eficazes na detecção da opinião. Embora uma classificação mais diversa possa ser mais desejável para futuras análises, os resultados alcançados neste estudo ainda fornecem *insights* valiosos sobre a detecção de opiniões em *datasets* limitados.

Diversos testes foram realizados na tentativa de identificar o conceito de “relevância” dos textos do *dataset*, divididos em quatro cenários:

- No primeiro cenário, foi utilizado como vetorizador o algoritmo TF-IDF e classificadores *Adaptive Boosting*, *LightGBM*, Regressão Logística, Naive-Bayes, Floresta Aleatória, SGD e SVM. Todos os classificadores apresentaram métricas similares, próximas a 0,8, com destaque para o classificador *Random Forest*.

- No segundo cenário, foi utilizado o vetorizador *Doc2Vec* e classificadores *Adaptive Boosting*, *LightGBM*, Regressão Logística, Naive-Bayes, Floresta Aleatória, SGD e SVM. Houve melhora em todas as métricas, com destaque para o classificador *LightGBM*, que apresentou métricas acima de 0,91.

- No terceiro cenário, foi utilizado o algoritmo *TextConvoNet*, que utiliza como vetorizador o algoritmo *GloVe*, e como classificador uma arquitetura própria baseada em CNN. As métricas neste cenário oscilaram próximas ao valor de 0,93.

- Por fim, no quarto cenário foi proposto uma modificação no algoritmo *TextConvoNet*, fornecendo como entrada os vetores gerados pela *Doc2Vec*. A motivação desta modificação foi testar somente seu núcleo de classificação com os mesmos dados vetorizados que foram utilizados em testes anteriores, para uma melhor comparação dos resultados. Foram necessários alguns ajustes na estrutura do algoritmo para que recebesse como entrada o formato de dados fornecido pela *Doc2Vec*. Os resultados desta modificação ficaram muito próximos da implementação original, próximo a 0,93. Vale ressaltar que o objetivo deste trabalho não é validar o desempenho

da *TextConvoNet*, e sim sua aplicabilidade no contexto de detecção de relevância proposto.

Conclue-se com estes resultados que o conceito de “relevância”, apesar de abstrato, é passível de detecção tanto por classificadores quanto por redes neurais complexas.

Os classificadores clássicos mostraram resultados discretos quando utilizados em conjunto com o vetorizador TF-IDF, porém apresentaram melhores resultados utilizando-se do *embedding Doc2Vec*. Ou seja, o algoritmo *Doc2Vec* conseguiu preservar a característica de relevância dos textos de forma mais assertiva do que o vetorizador TF-IDF. Conclue-se que a frequência de palavras específicas, estratégia abordada pelo vetorizador TF-IDF, não é um critério adequado para a identificação de relevância, característica esta já conjecturada durante a etapa de Análise Exploratória dos Dados. Adicionalmente, estes resultados evidenciam que a escolha de um vetorizador ou *word embedding* adequado ao problema é fundamental para a obtenção de bons resultados nas tarefas de classificação.

Por fim, a técnica apresentada se mostrou viável na classificação de atendimentos em relevantes e não relevantes no *dataset* de *Help Desk*.

Como trabalhos futuros, faz-se necessário um esforço maior na criação do *dataset*, utilizando-se mais classificadores humanos e alguma técnica de validação das marcações, de modo a reforçar a representatividade do conceito de “relevância” pelo *dataset* de treinamento.

REFERÊNCIAS

AL-HAWARI, F.; BARHAM, H. A machine learning based help desk system for it service management. *Journal of King Saud University - Computer and Information Sciences*, v. 33, n. 6, p. 702–718, 2021. ISSN 1319-1578.

ATENSTAEDT, R. Word cloud analysis of the BJGP. *British Journal of General Practice*, Royal College of General Practitioners, v. 62, n. 596, p. 148–148, 2012. ISSN 0960-1643.

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

BASSANI, H. F.; ARAUJO, A. F. A neural network architecture for learning word–referent associations in multiple contexts. *Neural Networks*, v. 117, p. 249–267, 2019. ISSN 0893-6080.

BATES, S.; HASTIE, T.; TIBSHIRANI, R. Cross-validation: what does it estimate and how well does it do it? *arXiv preprint arXiv:2104.00673*, 2021.

BHAVSAR, H.; PANCHAL, M. H. A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, v. 1, n. 10, p. 185–189, 2012.

BOSCOLO, V. G. *Sistema de Gerenciamento de Help-Desk*. Monografia (TCC) — Universidade Federal do Rio de Janeiro, set. 2009.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). *Diário Oficial da República Federativa do Brasil*, Brasília, DF, 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm>.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. [S.l.]: arXiv, 2020.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, v. 2, n. 2, p. 121–167, Jun 1998. ISSN 1573-756X. Disponível em: <https://doi.org/10.1023/A:1009715923555>.

CAVALARI, G. O. T.; COSTA, H. A. X. Modelagem e Desenvolvimento de um Sistema Help-Desk para a Prefeitura Municipal de Lavras. *Revista Eletrônica de Sistemas de Informação*, v. 4, n. 2, 2005.

CHAWLA, N. V.; SYLVESTER, J. Exploiting diversity in ensembles: Improving the performance on unbalanced datasets. In: HAINDL, M.; KITTLER, J.; ROLI, F. (Ed.). *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 397–406. ISBN 978-3-540-72523-7.

CHEN, J. et al. Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications*, v. 36, n. 3, Part 1, p. 5432–5435, 2009. ISSN 0957-4174.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. ISSN 1573-0565.

COSTA, A. d. S. C.; SANTANA, L. C. D.; TRIGO, A. C. Qualidade no atendimento ao cliente: Um grande diferencial competitivo para as organizações. *ACM Computing Surveys (CSUR)*, Cairu, BA: Revista de Iniciação Científica – RIC, v. 02, n. 02, p. 155–172, 2015.

DALIANIS, H. Evaluation metrics and evaluation. In: *Clinical text mining*. [S.l.]: Springer, 2018. p. 45–53.

DOGO, E. M. et al. A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. [S.l.: s.n.], 2018. p. 92–99.

FRANK, E.; BOUCKAERT, R. R. Naive Bayes for text classification with unbalanced classes. In: FÜRNKRANZ, J.; SCHEFFER, T.; SPILIOPOULOU, M. (Ed.). *Knowledge Discovery in Databases: PKDD 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 503–510.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics*, Institute of Mathematical Statistics, v. 28, n. 2, p. 337 – 407, 2000.

HAJRIZI, R.; NUÇI, K. P. Aspect-based sentiment analysis in education domain. *CoRR*, abs/2010.01429, 2020.

HICKMAN, L. et al. Text preprocessing for text mining in organizational research: Review and recommendations. *Organizational Research Methods*, v. 25, n. 1, p. 114–146, 2022.

HO, T. K. Random Decision Forests. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. [S.l.: s.n.], 1995. v. 1, p. 278–282 vol.1.

KANG, Y. et al. Natural Language Processing (NLP) in Management Research: A literature review. *Journal of Management Analytics*, Taylor & Francis, v. 7, n. 2, p. 139–172, 2020.

KE, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2017. v. 30.

KIRASICH, K.; SMITH, T.; SADLER, B. *Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets*. Southern Methodist University, 2018. 9 p. Disponível em: <https://core.ac.uk/download/pdf/216913541.pdf>.

KOTSIANTIS, S. B. Decision trees: a recent overview. *Artificial Intelligence Review*, v. 39, n. 4, p. 261–283, Apr 2013. ISSN 1573-7462.

KOWSARI, K. et al. Text classification algorithms: A survey. *Information*, v. 10, n. 4, 2019. ISSN 2078-2489.

KREšňáKOVá, V. M.; SARNOVSKY, M.; BUTKA, P. Deep learning methods for fake news detection. In: . [S.l.: s.n.], 2019.

LAS CASAS, A. L. *Marketing de serviços*. [S.l.]: Atlas, 1991.

LE, Q. V.; MIKOLOV, T. *Distributed Representations of Sentences and Documents*. arXiv, 2014. Disponível em: <https://arxiv.org/abs/1405.4053>.

LEVY, J. J.; O'MALLEY, A. J. Don't dismiss logistic regression: the case for sensible extraction of interactions in the era of machine learning. *BMC Medical Research Methodology*, v. 20, n. 1, p. 171, 2020. ISSN 1471-2288.

LO, D. et al. An emperical study on application of big data analytics to automate service desk business process. In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. [S.l.: s.n.], 2019. v. 2, p. 670–675.

LUGER, G. F.; CHAKRABARTI, C. From alan turing to modern ai: practical solutions and an implicit epistemic stance. *AI & SOCIETY*, v. 32, n. 3, p. 321–338, Aug 2017. ISSN 1435-5655.

MA, L.; ZHANG, Y. Using word2vec to process big text data. In: *2015 IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2015. p. 2895–2897.

MACHADO, M. R.; KARRAY, S.; SOUSA, I. T. de. Lightgbm: an effective decision tree gradient boosting method to predict customer loyalty in the finance industry. In: *2019 14th International Conference on Computer Science & Education (ICCSE)*. [S.l.: s.n.], 2019. p. 1111–1116.

MAYR, A. et al. The evolution of boosting algorithms. *Methods Inf Med*, Schattauer GmbH, v. 53, n. 06, p. 419–427, Jan 2018. ISSN 0026-1270.

MELENDEZ FILHO, R. *Service Desk Corporativo - Solução com Base na Itil V3*. [S.l.]: São Paulo: Novatec, 2011.

MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. [S.l.]: arXiv, 2013.

MIKOLOV, T.; LE, Q. V.; SUTSKEVER, I. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

MINAEE, S. et al. Deep learning–based text classification: A comprehensive review. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 54, n. 3, p. 1–40, 2021.

MYLES, A. J. et al. An introduction to decision tree modeling. *Journal of Chemometrics*, v. 18, n. 6, p. 275–285, 2004.

OKKALIOGLU, M.; OKKALIOGLU, B. D. Afe-mert: Imbalanced text classification with abstract feature extraction. *Applied Intelligence*, Springer, p. 1–17, 2022.

PARAMESH, S. P.; SHREEDHARA, K. S. Automated it service desk systems using machine learning techniques. In: NAGABHUSHAN, P. et al. (Ed.). *Data Analytics and Learning*. Singapore: Springer Singapore, 2019. p. 331–346. ISBN 978-981-13-2514-4.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543.

POZZOLO, A. D. et al. Calibrating probability with undersampling for unbalanced classification. In: *2015 IEEE Symposium Series on Computational Intelligence*. [S.l.: s.n.], 2015. p. 159–166.

RAJARAMAN, A.; ULLMAN, J. D. Data mining. In: ____. *Mining of Massive Datasets*. [S.l.]: Cambridge University Press, 2011. p. 1–17.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: ____. *Encyclopedia of Database Systems*. New York, NY: Springer New York, 2016. p. 1–7. ISBN 978-1-4899-7993-3.

ROKACH, L.; MAIMON, O. Decision trees. In: ____. *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2005. p. 165–192. ISBN 978-0-387-25465-4.

RUDER, S. *An overview of gradient descent optimization algorithms*. [S.l.]: arXiv, 2016.

RYMARCZYK, T. et al. Logistic regression for machine learning in process tomography. *Sensors*, MDPI AG, v. 19, n. 15, p. 3400, Ago 2019. ISSN 1424-8220.

SCHAPIRE, R. E.; FREUND, Y. Boosting: Foundations and algorithms. *Kybernetes*, Emerald Group Publishing Limited, v. 42, n. 1, p. 164–166, Jan 2013. ISSN 0368-492X.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning - from Theory to Algoritms*. [S.l.]: Cambridge University Press, 2014.

SINGH, A. K.; SHASHI, M. Vectorization of text documents for identifying unifiable news articles. *Int. J. Adv. Comput. Sci. Appl*, v. 10, n. 7, 2019.

SINGH, G. et al. Comparison between multinomial and bernoulli naïve bayes for text classification. In: *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*. [S.l.: s.n.], 2019. p. 593–596.

SONI, S.; CHOUHAN, S. S.; RATHORE, S. S. *TextConvoNet: A Convolutional Neural Network based Architecture for Text Classification*. [S.l.]: arXiv, 2022.

SRA, S.; NOWOZIN, S.; WRIGHT, S. *Optimization for Machine Learning*. [S.l.]: MIT Press, 2012. (Neural information processing series). ISBN 9780262016469.

SUNDARKUMAR, G. G.; RAVI, V. A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance. *Engineering Applications of Artificial Intelligence*, v. 37, p. 368–377, 2015. ISSN 0952-1976.

SWINBURNE, R. Bayes's Theorem. British Academy, 2005.

TING, K. M.; ZHU, Y.; ZHOU, Z.-H. Isolation kernel and its effect on svm. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2018. (KDD '18), p. 2329–2337. ISBN 9781450355520. Disponível em: <https://doi.org/10.1145/3219819.3219990>.

WANG, Z.; ZHANG, J.; VERMA, N. Realizing low-energy classification systems by implementing matrix multiplication directly within an adc. *IEEE Transactions on Biomedical Circuits and Systems*, v. 9, p. 1–1, 12 2015.

YACOUBY, R.; AXMAN, D. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In: *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*. [S.l.: s.n.], 2020. p. 79–91.

YAO, L.; GUAN, Y. An improved LSTM structure for natural language processing. In: *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. [S.l.: s.n.], 2018. p. 565–569.

YAO, Y. et al. Learning to activate logic rules for textual reasoning. *Neural Networks*, v. 106, p. 42–49, 2018. ISSN 0893-6080.

ZHANG, X. et al. The use of roc and auc in the validation of objective image fusion evaluation metrics. *Signal processing*, Elsevier, v. 115, p. 38–48, 2015.