UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII - PATOS
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE GRADUAÇÃO EM BACHARELADO EM COMPUTAÇÃO

ROMUALDO JUNIOR FREITAS LEITE

ANÁLISE COMPARATIVA DE MODELOS DE CLASSIFICAÇÃO EM PROCESSAMENTO DE LINGUAGEM NATURAL PARA DETECÇÃO DE SPAMS EM MENSAGENS DE TEXTO

PATOS - PB
2024

ROMUALDO JUNIOR FREITAS LEITE

ANÁLISE COMPARATIVA DE MODELOS DE CLASSIFICAÇÃO EM PROCESSAMENTO DE LINGUAGEM NATURAL PARA DETECÇÃO DE SPAMS EM MENSAGENS DE TEXTO

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Computação do Centro de Ciências Exatas e Sociais Aplicadas da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Computação.

Orientador: Dr. Jucelio Soares dos Santos

PATOS - PB
2024

L533a Leite, Romualdo Junior Freitas.

Análise comparativa de modelos de classificação em processamento de linguagem natural para detecção de spams em mensagens de texto [manuscrito] / Romualdo Junior Freitas Leite. - 2024.

50 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas, 2024.

"Orientação : Prof. Dr. Jucelio Soares dos Santos, Coordenação do Curso de Computação - CCEA. "

1. Detecção de spams. 2. Processamento de Linguagem Natural. 3. Modelos de Classificação. I. Título

21. ed. CDD 005.133

Elaborada por Camile de A. Gomes - CRB - 15/559 COBIB/UEPB

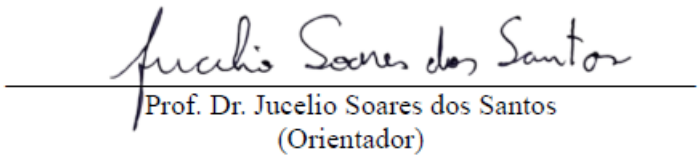
ROMUALDO JUNIOR FREITAS LEITE

ANÁLISE COMPARATIVA DE MODELOS DE CLASSIFICAÇÃO EM PROCESSAMENTO DE LINGUAGEM NATURAL PARA DETECÇÃO DE SPAMS EM MENSAGENS DE TEXTO

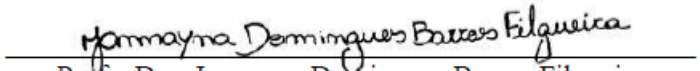
Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Computação do Centro de Ciências Exatas e Sociais Aplicadas da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Computação.

Trabalho aprovado em 27/06/2024.

BANCA EXAMINADORA



Prof. Dr. Jucelio Soares dos Santos
(Orientador)



Profa. Dra. Jannayna Domingues Barros Filgueira
(Examinadora)

Prof. Bel. Vinicius Augustus Alves Gomes
(Examinador)

Dedico este trabalho aos meus pais, Socorro e José, e à minha esposa, Mariana, cujo amor, apoio e orientação foram fundamentais. Ao meu orientador, Dr. Jucelio Soares dos Santos, pelo suporte essencial. A vocês, meu profundo agradecimento por tornarem esta jornada acadêmica uma realidade.

AGRADECIMENTOS

Esta jornada acadêmica, que moldou e continuará moldando minha vida, presenteou-me com momentos verdadeiramente excepcionais. Conheci pessoas incríveis, entre amigos e professores, e vivi experiências inesquecíveis que certamente carregarei para sempre.

Primeiramente, expresso minha gratidão a Deus pela Sua misericórdia e sabedoria que me permitiram trilhar este caminho acadêmico e alcançar este capítulo final. Como diz Romanos 8:28, "Todas as coisas cooperam para o bem daqueles que amam a Deus". Agradeço aos meus pais, cujo apoio incansável e incentivo foram fundamentais em cada passo dessa jornada. Eles sempre estiveram ao meu lado, encorajando-me a persistir nos momentos desafiadores ao longo destes anos.

À minha esposa, expresso minha gratidão por sua presença constante e seu apoio incansável nos momentos difíceis. Sua força foi meu sustento e, em muitas ocasiões, o impulso que me impediu de desistir. Esta conquista não será apenas minha, mas nossa. Sou imensamente grato por tê-la ao meu lado.

Não posso deixar de agradecer ao meu orientador, Dr. Jucelio Soares dos Santos, cujo suporte tem sido fundamental desde os meus primeiros passos na Universidade Estadual da Paraíba. Sua dedicação e disposição para com seus alunos são admiráveis. Este trabalho de conclusão de curso não teria sido possível sem seu auxílio e orientação. Sua contribuição foi essencial para que eu pudesse concluir este TCC.

“Pois, que adianta ao homem ganhar o mundo inteiro e perder a sua alma?."

Jesus Cristo

RESUMO

A detecção de *spam* é crucial para a segurança e eficiência da comunicação. Com o uso crescente de dispositivos móveis e serviços de mensagens como o *e-mail*, o envio de *spam* tornou-se uma preocupação crescente. Essas mensagens indesejadas podem conter *phishing*, fraudes, *links* maliciosos e outros conteúdos prejudiciais, representando ameaças à segurança dos usuários e à integridade dos dados. Além disso, o *spam* excessivo pode prejudicar a eficácia da comunicação, sobrecarregando as caixas de entrada dos utilizadores e dificultando a identificação e resposta a mensagens legítimas. Neste contexto, nossa pesquisa explora e avalia diferentes modelos de classificação em Processamento de Linguagem Natural em um estudo de caso específico para detecção automática de *spams* em mensagens de texto. Para tanto, utilizamos uma base de dados precisa e, após tratamento e pré-processamento dos dados, identificamos uma disparidade na distribuição das mensagens, destacando a importância de abordagens equilibradas. Empregamos vários algoritmos, como *K-Nearest Neighbors*, *Naive Bayes Multinomial*, *Support Vector Classifier*, *Logistic Regression* e *Decision Tree Classifier*, avaliando seu desempenho por meio de métricas como acurárica, precisão, *recall* e *F1-Score*. Destacamos que o *Support Vector Classifier* obteve o melhor desempenho, demonstrando sua eficácia na identificação precisa do *spam*, com um equilíbrio satisfatório entre precisão e *recall*.

Palavras-chave: Detecção de *Spams*. Processamento de Linguagem Natural. Modelos de Classificação. Eficácia.

ABSTRACT

Spam detection is crucial for the security and efficiency of communication. With the increasing use of mobile devices and messaging services such as e-mail, sending spam has become a growing concern. These unwanted messages may contain phishing, scams, malicious links, and other harmful content, threatening users' security and data integrity. Furthermore, excessive spam can harm communication effectiveness, overloading users' inboxes and making it difficult to identify and respond to legitimate messages. In this context, our research explores and evaluates different classification models in Natural Language Processing in a specific case study for automatically detecting spam in text messages. To do so, we used an accurate database, and after treatment and pre-processing the data, we identified a disparity in the distribution of messages, highlighting the importance of balanced approaches. We employ several algorithms, such as K-Nearest Neighbors, Naive Bayes Multinomial, Support Vector Classifier, Logistic Regression, and Decision Tree Classifier, evaluating their performance through accuracy, precision, recall, and F1-Score metrics. We highlight that the Support Vector Classifier achieved the best performance, demonstrating its effectiveness in accurately identifying spam with a satisfactory balance between precision and recall.

Keywords: Spam detection. Natural Language Processing. Classification Models. Efficiency.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	*Área Sob a Curva*
BERT	*Bidirectional Encoder Representations from Transformers*
BOW	*Bag-of-Words*
CNN	*Convolutional Neural Network*
CSV	*Comma-Separated Values*
DT	*Decision Tree*
F1-Score	*F-Measure*
FN	*Falsos Negativos*
FP	*Falsos Positivos*
GPT	*Generative Pre-trained Transformer*
KNN	*K-Nearest Neighbors*
LR	*Logistic Regression*
NB	*Naive Bayes*
NLTK	*Natural Language Toolkit*
NSC	*NUS SMS Corpus*
NUS	*National University of Singapore*
PLN	*Processamento de Linguagem Natural*
RBF	*Radial Basis Function*
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristic*
SMS	*Short Message Service*
SVC	*Support Vector Classifier*
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency-Inverse Document Frequency*
TFP	*Taxa de Falsos Positivos*

TN	*Verdadeiros Negativos*
TP	*Verdadeiros Positivos*
TVP	*Taxa de Verdadeiros Positivos*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, oferecemos uma visão geral desta pesquisa, delineando a contextualização do problema, a proposta de solução e os objetivos da pesquisa.

1.1 Contextualização do Problema

A revolução digital transformou radicalmente a forma como nos comunicamos e interagimos (Surdak, 2018). Com o advento das comunicações digitais, testemunhamos um aumento exponencial no volume de mensagens, e-mails e interações *online* (Reddy et al., 2020). Embora essa explosão tenha seus benefícios, também trouxe desafios significativos para a eficiência e segurança das comunicações. Um desses desafios é a proliferação de *spam* (Roy; Singh; Banerjee, 2020).

O termo "*spam*"refere-se a mensagens não solicitadas, frequentemente de natureza comercial ou maliciosa, que inundam nossas caixas de entrada de *e-mail*, mensageiros e redes sociais (Karim et al., 2019). Os efeitos prejudiciais dos *spams* são abrangentes, incluindo sobrecarga de caixas de entrada, disseminação de conteúdo enganoso ou malicioso e redução da produtividade dos usuários. Além disso, os *spams* representam uma ameaça constante à segurança digital, sendo veículos potenciais para *malware*[1], *phishing*[2] e fraudes *online* (Rao; Verma; Bhatia, 2021).

No cenário atual, a necessidade de desenvolver abordagens eficazes para detectar e filtrar *spams* tornou-se crucial. A triagem manual de mensagens indesejadas revelou-se não apenas tediosa, mas também altamente ineficiente, consumindo tempo e recursos preciosos. Diante desse desafio premente, surgiu uma solução na aplicação de técnicas de Processamento de Linguagem Natural (PLN) avançadas e algoritmos de classificação (Oliveira et al., 2022; Sen; Hajra; Ghosh, 2020).

Esses métodos sofisticados permitem a automação da identificação e eliminação de *spams*, oferecendo uma alternativa ágil e precisa para lidar com o problema. O uso do PLN capacita os sistemas a compreenderem o contexto, a semântica e as nuances das mensagens, identificando padrões sutis de *spam* que escapam à detecção humana (Junnarkar et al., 2021). Além disso, os algoritmos de classificação, impulsionados por dados de treinamento, conseguem distinguir com precisão entre mensagens legítimas e indesejadas, contribuindo para a otimização do fluxo de comunicações e a proteção contra ameaças cibernéticas.

Essa abordagem avançada não apenas simplifica o gerenciamento de e-mails, mas também permite a alocação eficaz de recursos humanos e de computação, proporcionando uma resposta

[1] *Malware*, ou *software* malicioso, é qualquer programa desenvolvido com a intenção de causar danos a computadores, redes ou dados. Vírus, *worms*, trojans, *ransomware*, *spyware*, *adware* e *botnets* são exemplos de *malwares*, cada um com propósitos distintos (Aslan; Samet, 2020).

[2] *Phishing* é um tipo de ataque cibernético em que hackers enganam indivíduos por meio de mensagens falsas, visando obter informações confidenciais, como senhas. Estas mensagens, apesar de parecerem legítimas, exploram a confiança das vítimas. A conscientização e a adoção de práticas de segurança são fundamentais para prevenir esse tipo de ameaça online (Alkhalil et al., 2021).

eficiente a uma das questões mais prementes enfrentadas pelas comunicações modernas.

1.2 Problema

A complexidade da detecção de *spams* em linguagem natural se manifesta por diversos desafios intrincados. Primeiramente, destaca-se a evolução constante das técnicas empregadas por *spammers*, que estão em constante busca por estratégias mais sofisticadas para contornar sistemas de detecção. Além disso, a natureza enganosa e variável dos *spams*, que muitas vezes se adaptam para imitar comunicações legítimas, aumenta ainda mais a dificuldade de identificação (Nivedha; Raja, 2022).

As abordagens tradicionais de detecção enfrentam limitações significativas diante desses desafios dinâmicos, exigindo uma constante atualização e aprimoramento dos métodos de análise de mensagens. A corrida armamentista digital entre desenvolvedores de soluções anti-spam e criadores de conteúdo malicioso adiciona uma camada adicional de complexidade, pois cada avanço em uma frente muitas vezes resulta em contra-ataques e adaptações por parte dos *spammers*. Esse cenário dinâmico ressalta a necessidade contínua de inovação e adaptação nas estratégias de detecção de *spams* em linguagem natural (Ora, 2020).

Os *spammers* operam em um ambiente dinâmico, constantemente inovando para contornar sistemas de detecção. Essa evolução demanda vigilância e aprimoramento contínuo dos métodos de detecção. A natureza enganosa e variável das mensagens de *spam* dificulta estabelecer regras fixas de identificação, exigindo abordagens flexíveis. As técnicas tradicionais, como regras heurísticas baseadas em padrões conhecidos, podem enfrentar dificuldades frente à crescente diversidade e sofisticação dos *spams*, tornando-se menos eficazes diante de estratégias mais avançadas (Rao; Verma; Bhatia, 2021).

Diante do desafio na detecção automática de *spams*, a exploração de diferentes modelos de classificação em PLN torna-se imperativa. Os algoritmos utilizados na detecção de *spams* desempenham um papel crucial diante da constante evolução das estratégias empregadas pelos *spammers*. Entre eles, o *Support Vector Classifier* opera com base em hiperplanos de separação, demonstrando eficácia na classificação de dados complexos. O *K-Nearest Neighbors* atribui rótulos com base na proximidade em um espaço multidimensional, sendo sensível a padrões locais. O *Naive Bayes Multinomial* emprega o teorema de Bayes para calcular probabilidades condicionais, sendo eficiente em cenários com múltiplas categorias. O *Decision Tree Classifier* utiliza estruturas de árvores de decisão, facilitando interpretação e adaptabilidade. Por fim, a *Logistic Regression* modela a probabilidade de pertencer a uma classe, sendo útil em problemas de classificação binária. Esses algoritmos, cada um com suas peculiaridades, contribuem para a efetividade na identificação de *spams* em diferentes contextos de PLN (Saidani; Adi; Allili, 2020).

Nesse contexto desafiador, surge a problemática central que norteia esta pesquisa: como realizar uma análise comparativa entre diferentes modelos de classificação em PLN, a fim de identificar qual abordagem demonstra maior eficácia na detecção de *spams*, considerando a

evolução constante das estratégias dos *spammers* e as limitações das técnicas tradicionais?

1.3 Proposta de Solução

Para resolver essa problemática complexa, propomos uma abordagem que explore e avalie diferentes modelos de classificação em PLN em um estudo de caso específico para a detecção automática de *spams* em mensagens de texto. A ênfase será na identificação e comparação de diferentes algoritmos de classificação para determinar o mais eficaz na tarefa de detectar *spams*.

A proposta central envolve a exploração de modelos de classificação em PLN, aproveitando as vantagens que essas abordagens avançadas oferecem. Utilizaremos algoritmos de aprendizado de máquina, como *Support Vector Classifier*, *K-Nearest Neighbors*, *Naive Bayes Multinomial*, *Decision Tree Classifier* e *Logistic Regression* para treinar modelos capazes de distinguir entre mensagens legítimas e *spams* (Saidani; Adi; Allili, 2020).

A avaliação da efetividade da detecção será realizada por meio de métricas apropriadas, como acurácia, precisão, recall, F1-score e a construção de matrizes de confusão (Yacouby; Axman, 2020). Dessa forma, buscamos não apenas identificar um modelo que alcance resultados promissores em termos de detecção, mas também compreender como esses modelos lidam com diferentes nuances e desafios inerentes à detecção de *spams* em PLN.

Após a execução de experimentos sistemáticos, faremos uma análise comparativa dos resultados obtidos pelos diferentes modelos. Essa análise proporcionará percepções valiosas sobre a eficácia e as limitações de cada abordagem, permitindo uma compreensão mais aprofundada das nuances da detecção de *spams* em contextos específicos.

Esperamos que esta pesquisa contribua para o avanço do campo de detecção de *spams* em PLN, oferecendo percepções sobre abordagens eficazes e fornecendo orientações para a implementação prática desses modelos em ambientes reais. Além disso, a comparação entre diferentes algoritmos de classificação em PLN pode destacar as melhores práticas e estratégias para enfrentar os desafios em constante evolução impostos pelos *spammers*.

1.4 Objetivos

Esta pesquisa visa explorar e avaliar diferentes modelos de classificação em PLN em um estudo de caso específico para detecção automática de *spams* em mensagens de texto. Para alcançar esse objetivo, desdobramos a pesquisa nos seguintes objetivos específicos:

- Realizar uma revisão bibliográfica sobre os principais conceitos, técnicas e algoritmos de classificação em PLN;
- Utilizar um conjunto de dados representativo contendo mensagens de texto rotuladas como *spam* ou não *spam* para treinamento e teste dos modelos;
- Explorar e aplicar diferentes abordagens de pré-processamento e representação de texto para a preparação dos dados antes da classificação;

- Implementar os seguintes algoritmos de classificação: *Support Vector Classifier*, *K-Nearest Neighbors*, *Naive Bayes Multinomial*, *Decision Tree Classifier* e *Logistic Regression*, adaptando-os para a tarefa de detecção de *spams*;
- Realizar experimentos sistemáticos com os modelos de classificação utilizando métricas adequadas, como acurácia, precisão, *recall*, *F1-score* e matriz de confusão, para avaliar o desempenho de cada algoritmo. Além disso, realizar uma análise comparativa e discutir os resultados obtidos pelos diferentes modelos, identificando o mais eficiente na detecção de *spams* em relação às métricas de avaliação.

1.5 Estrutura do Trabalho

Este trabalho apresenta cinco capítulos e está organizado da seguinte maneira: no capítulo 1, apresentamos uma visão geral desta investigação com relação ao cenário técnico-científico, problema, proposta de solução, objetivos, metodologia operacional e questão de pesquisa; no capítulo 2, fornecemos um embasamento teórico abrangente, explorando conceitos e pesquisas relacionadas às áreas de estudo relevantes para este estudo; no capítulo 3, relatamos a metodologia para investigar diferentes modelos de classificação em PLN em um estudo de caso específico para detecção automática de *spams* em mensagens de texto; no capítulo 4, apresentamos os resultados e discussões desta investigação; por fim, no capítulo 5, apresentamos as considerações finais e sugestões para trabalhos futuros; e ao final, encontram-se as referências utilizadas no decorrer desta pesquisa.

2 REFERENCIAL TEÓRICO

Neste capítulo, apresenta-se o embasamento teórico a partir de várias áreas e trabalhos que se relacionam e que permitem caracterizar esta pesquisa.

2.1 *Spams*

O termo *spam* tem origens peculiares que remontam à década de 1970, quando a marca de carnes enlatadas SPAM® o popularizou. A empresa, conhecida por produzir produtos de grande consumo, foi a inspiração para um esquete humorístico do grupo britânico Monty Python, no qual o termo era repetido várias vezes, criando uma relação duradoura com algo irritante e intrusivo (Cranor; LaMacchia, 1998).

Contudo, a verdadeira revolução do termo ocorreu com o advento da era digital. Com a internet, o *spam* se tornou uma preocupação significativa. O *spam* digital refere-se a mensagens não solicitadas enviadas por *e-mail*, *feeds* de mídias sociais ou mensagens de texto (Koggalahewa; Xu; Foo, 2020).

Essa mudança transformou o *spam* em uma estratégia de marketing invasiva e, em alguns casos, uma ameaça à segurança online. O *spam* eletrônico é caracterizado pelo excesso de mensagens, onde remetentes anônimos enviam mensagens em grande escala na esperança de atingir muitos destinatários. Essas mensagens, que promovem produtos e serviços ou realizam ações maliciosas, tornaram-se uma verdadeira praga no mundo digital (Koggalahewa; Xu; Foo, 2020; Jindal; Liu, 2007).

A disseminação de *spam* não se limita apenas a e-mails; redes sociais, fóruns online e até mesmo comentários em blogs são frequentemente alvos de *spam*. A utilização invasiva de *spam* interfere negativamente na experiência do usuário, interrompendo a comunicação legítima e apresentando riscos de segurança, como a disseminação de *phishing* e *malware* (Kawintiranon; Singh; Budak, 2022; Jindal; Liu, 2007). O desafio é encontrar métodos para mitigar *spam*, seja via filtros anti-*spam* avançados, educação dos usuários ou medidas regulamentares (Kawintiranon; Singh; Budak, 2022).

2.1.1 Características dos Spams

Para identificar *spams*, é fundamental reconhecer algumas características críticas. Frequentemente exploradas por sistemas de detecção, estas características ajudam a diferenciar entre mensagens legítimas e indesejadas. Algumas características comuns do *spam* envolvem conteúdo enganoso, links suspeitos, uso excessivo de maiúsculas e pontuações, solicitação de informações pessoais e ofertas não solicitadas (Frommholz et al., 2016).

Ao reconhecer estas características comuns, os sistemas de detecção de *spam* podem estabelecer padrões e identificar possíveis ameaças, protegendo eficazmente contra comunicações indesejadas. Quando avaliadas em conjunto, essas características contribuem para a criação de

modelos eficientes de detecção de *spam* (Karim et al., 2019; Asdaghi; Soleimani, 2019).

Muitos *spams* contêm conteúdo enganoso, oferecendo produtos fraudulentos, esquemas financeiros duvidosos ou informações enganosas. Promessas exageradas, linguagem sensacionalista e ofertas "irresistíveis" podem indicar *spam* (Xue et al., 2019; Pera; Ng, 2008; Frommholz et al., 2016).

Links suspeitos ou encurtados em *spams* podem atrair pessoas para sites maliciosos e ajudá-las a cometer fraudes. Incluir *Uniform Resource Locators* (URLs) desconhecidas ou suspeitas é um recurso que se destaca como fator de alerta durante processos de detecção de *spam*. Além disso, os *spammers* usam muitas letras maiúsculas e pontuação para chamar a atenção do destinatário. Essa prática foge do padrão convencional de comunicação, facilitando sua identificação (Frommholz et al., 2016).

Além disso, solicitar informações pessoais, como senhas, dados bancários e cartões de crédito, é comum em *spam*, o que pode significar tentativa de fraude. Nestes casos, os *spammers* procuram obter informações confidenciais via engano, explorando a confiança do destinatário. A identificação destes pedidos torna-se crucial para a detecção precoce de atividades fraudulentas, reforçando a importância da adoção de medidas preventivas contra estas práticas nocivas (Stringhini; Kruegel; Vigna, 2010).

2.1.2 Riscos Associados ao Spam

O *spam* é um problema sério para os usuários e pode representar um grande perigo. Esse conjunto de ações indesejadas pode causar danos à segurança, privacidade e eficiência das comunicações digitais (Paavolainen; Elo; Nikander, 2018).

Em geral, os *spams* contêm links suspeitos que direcionam os usuários para sites maliciosos. Ao clicar nesses links, os usuários podem, inadvertidamente, expor seus dispositivos a *malwares*, ataques de *phishing* e outras formas de exploração digital (Doshi et al., 2023). As mensagens de *spam* podem conter *malware*, como vírus, cavalos de troia ou *ransomware*. O *download* involuntário desses elementos maliciosos pode causar prejuízos significativos aos sistemas e informações do usuário (Trivedi; Broadhurst, 2020).

Muitos *spammers* procuram obter dados pessoais e confidenciais por meio de técnicas de *phishing* (Doshi et al., 2023). Ao solicitar dados bancários ou detalhes de cartões de crédito, os *spammers* visam enganar os usuários, comprometendo sua segurança financeira e privacidade. O *spam* pode resultar em roubo de identidade e fraude (Ferrara, 2019).

As empresas que enviam *spam* podem sofrer danos à sua reputação (Buil-Gil; Barrett et al., 2022). A percepção negativa decorrente do envio excessivo de mensagens indesejadas pode afetar negativamente a confiança do público e prejudicar a imagem da organização. Essas mensagens podem inundar as caixas de entrada, reduzindo a produtividade, pois os usuários devem dedicar um tempo significativo para filtrar e lidar com mensagens indesejadas (Silva; Al-Khatib; Tsigaris, 2020; Trivedi; Broadhurst, 2020). Ao compreender os riscos do *spam*, torna-se evidente a necessidade de estratégias de detecção e prevenção para mitigar essas ameaças e

garantir um ambiente digital mais seguro.

2.1.3 Estratégias de Combate ao Spam

Para combater eficazmente o *spam*, é necessário utilizar diversas técnicas para identificar, filtrar e prevenir a disseminação de mensagens indesejadas. Uma dessas táticas é a filtragem heurística, que utiliza regras baseadas em padrões conhecidos para identificar *spams*. Essas regras podem ajudar a identificar palavras-chave, endereços suspeitos e mensagens indesejáveis (Mehrotra et al., 2021).

Outra técnica eficaz é a análise de conteúdo, que examina o conteúdo das mensagens para identificar características típicas, como o uso excessivo de maiúsculas, pontuações e solicitações de dados pessoais. Em conjunto, a estratégia de análise comportamental observa como os usuários interagem com as mensagens recebidas, identificando padrões, como a rápida exclusão de mensagens, que podem indicar a presença de *spams*. Essas técnicas combinadas aumentam a eficiência na identificação e filtragem de mensagens indesejadas (Alsulami; Al-Aama, 2020; Sahoo et al., 2020).

A utilização de listas negras e brancas pode ser aprimorada por meio de registros atualizados de remetentes conhecidos de *spam* (listas negras) e remetentes confiáveis (listas brancas). Essas listas ajudam a classificar as mensagens automaticamente e facilitam a identificação das mensagens (Cao; Lai, 2020).

Ademais, a utilização de protocolos como *Sender Policy Framework* (SPF) e *DomainKeys Identified Mail* (DKIM) pode ser considerada uma medida efetiva. Esses protocolos foram criados para verificar a legitimidade dos remetentes, reduzindo significativamente o envio de *spams*. Assim, a combinação dessas táticas reforça a proteção contra *spams* ao abordar tanto remetentes suspeitos quanto a autenticidade das mensagens (Marzuki; Hanif; Hariyadi, 2022; Kahraman, 2020).

A conscientização dos usuários é crucial para prevenir ameaças de e-mails, incluindo *spams*. Essa estratégia ensina os usuários a utilizar e-mails de forma segura, com comportamentos cautelosos e aperfeiçoados. Os usuários devem evitar clicar em links suspeitos nas mensagens, uma vez que *spams* frequentemente usam URLs maliciosas para direcionar as vítimas para sites fraudulentos (Carpenter et al., 2020).

Além disso, a conscientização enfatiza a relevância de não compartilhar dados pessoais em resposta a mensagens suspeitas, uma vez que muitos *spams* buscam obter informações confidenciais por meio de solicitações enganosas. Incentivar os usuários a identificar e relatar mensagens suspeitas contribui para uma comunidade virtual mais segura. A compreensão e a aplicação desses princípios tornam a postura mais resistente contra tentativas de *spam* e outros ataques cibernéticos (Althobaiti, 2021; Alsubhi et al., 2021).

Por fim, modelos de aprendizado de máquina podem ser treinados para identificar padrões complexos em grandes quantidades de dados. Esses modelos possuem a habilidade de se adaptar constantemente às táticas dos *spammers*, acompanhando as mudanças nas estratégias de envio

de *spams*. A combinação de técnicas avançadas de Processamento de Linguagem Natural (PLN) e algoritmos de aprendizado de máquina permite uma resposta rápida e precisa na identificação de mensagens indesejadas, aumentando significativamente a eficácia das estratégias de combate ao *spam* (Elbagir; Yang, 2019; Schütze, 2008).

2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) emergiu recentemente como uma das tecnologias mais promissoras. Com a capacidade de compreender, interpretar e produzir linguagem humana de forma inteligente, o PLN está transformando significativamente nossa interação com a tecnologia e a forma como as máquinas entendem e respondem às nossas necessidades (Bahja, 2020).

Tradicionalmente, o PLN se concentrava na compreensão e geração de texto. No entanto, uma tendência crescente é a compreensão multimodal, que envolve a interpretação de diferentes tipos de dados, como texto, imagens, áudio e vídeo. Isso leva ao desenvolvimento de sistemas mais abrangentes e contextuais, capazes de compreender a informação de forma holística (O'Halloran; Pal; Jin, 2021).

Um dos principais desafios em PLN é a compreensão da linguagem natural, que envolve tarefas como análise sintática, análise semântica e extração de informações (Chowdhary; Chowdhary, 2020). Essas tarefas são fundamentais para a interpretação precisa do significado das frases e documentos, permitindo que os sistemas de PLN extraiam conhecimento útil e respondam de maneira eficaz às consultas dos usuários.

Outra área importante em PLN é a geração de linguagem natural, que se concentra na criação de texto coerente e significativo a partir de dados estruturados ou comandos de entrada. Isso é útil em uma variedade de aplicativos, desde assistentes virtuais e chatbots até sistemas de geração automática de relatórios e resumos (Meng et al., 2022).

A classificação em PLN é uma tarefa essencial que envolve a atribuição automática de categorias ou rótulos a documentos de texto com base em seu conteúdo (Chowdhary; Chowdhary, 2020). Para alcançar esse objetivo, são empregados algoritmos de aprendizado de máquina e técnicas de PLN, que analisam minuciosamente o texto, identificando padrões e características relevantes para a correta categorização (Schütze, 2008).

A importância da classificação em PLN é inegável, desempenhando um papel fundamental em diversos contextos (Elbagir; Yang, 2019). Primeiramente, a organização de informações é aprimorada, tornando-se mais acessível e gerenciável em bibliotecas digitais, bancos de dados e repositórios de documentos (Khurana et al., 2023). Além disso, a recuperação de informações torna-se mais precisa, beneficiando motores de busca que utilizam algoritmos de classificação para fornecer resultados mais relevantes aos usuários (Hirschberg; Manning, 2015).

A análise de sentimentos é outra aplicação crucial da classificação em PLN, permitindo que empresas e pesquisadores compreendam o sentimento do público em relação a produtos, serviços, políticas e eventos (Jiang et al., 2019). Isso é vital para tomar decisões informadas,

adaptar estratégias de negócios e compreender as tendências sociais. Além disso, a classificação desempenha um papel essencial na detecção de spam, ajudando a manter as caixas de entrada de e-mail e mensagens livres de mensagens indesejadas, melhorando a eficiência das comunicações eletrônicas.

Por fim, a categorização de documentos é uma aplicação prática da classificação em ambientes corporativos, bibliotecas e sistemas de gerenciamento de conteúdo. Ela facilita a organização e a recuperação eficiente de documentos, otimizando a gestão de informações em organizações e facilitando o acesso a dados relevantes. Portanto, a classificação em PLN desempenha um papel multifacetado e de grande relevância nos contextos modernos da informação e da comunicação.

2.3 Algoritmos de Classificação

Os algoritmos de classificação desempenham um papel fundamental em uma ampla gama de aplicações, permitindo a categorização automatizada de dados com base em suas características. Esses algoritmos são amplamente utilizados em campos como aprendizado de máquina, mineração de dados, processamento de linguagem natural e muito mais (Deepak; Ameer, 2021).

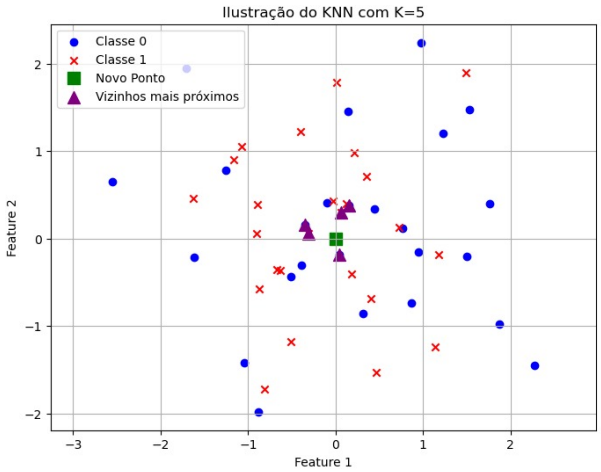
Essencialmente, os algoritmos de classificação são projetados para atribuir automaticamente rótulos ou categorias a novos conjuntos de dados com base em padrões identificados nos dados de treinamento. Eles são aplicados em uma variedade de cenários, desde a detecção de spam em e-mails até a previsão do risco de doenças em pacientes (Gasparetto et al., 2022).

Existem diversos tipos de algoritmos de classificação, cada um com suas características e aplicabilidades distintas. Alguns dos algoritmos mais comuns incluem: *K-Nearest Neighbors* (K-NN), *Naive Bayes Multinomial*, *Support Vector Classifier* (SVC), *Logistic Regression* e *Decision Tree Classifier*.

2.3.1 K-Nearest Neighbors

O algoritmo *K-Nearest Neighbors* (K-NN) é um método de classificação e regressão baseado na proximidade. Ele atribui rótulos a novos pontos de dados com base na classe da maioria dos 'K' pontos mais próximos no espaço de características. É uma técnica de aprendizado de máquina supervisionado que pressupõe que pontos de dados semelhantes estão próximos uns dos outros em um espaço de características, e, portanto, novos pontos de dados podem ser classificados com base na maioria de seus vizinhos mais próximos (Han; Pei; Tong, 2022). A Figura 1 ilustra o funcionamento do algoritmo K-NN.

Figura 1 – Exemplo do Funcionamento do Algoritmo *K-Nearest Neighbors*



Fonte: Elaborado pelo autor (2024).

Cada ponto de dados é representado por um ponto no espaço de características, e as diferentes cores indicam diferentes classes ou rótulos. Quando um novo ponto de dados (marcado como ‘quadrado verde’) precisa ser classificado, o algoritmo identifica os ’K’ pontos mais próximos a ele (definidos pela distância euclidiana, por exemplo) e atribui a classe mais comum entre esses vizinhos ao ponto de dados de teste.

O funcionamento do K-NN pode ser resumido em três etapas (Larose; Larose, 2014):

1. **Seleção de Vizinhos Mais Próximos**: Identificar os K pontos de dados mais próximos ao novo ponto de dados com base em uma medida de distância, como a distância euclidiana.

2. **Votação**: Atribuir a classe mais frequente entre os K vizinhos mais próximos ao novo ponto de dados.

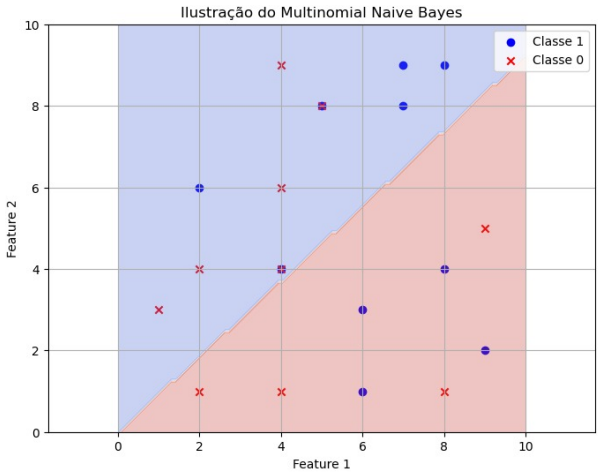
3. **Ajuste do Hiperparâmetro K**: Escolher o valor de K que equilibre a sensibilidade ao ruído e a suavidade do modelo, utilizando técnicas como validação cruzada.

O K-NN é útil para problemas de classificação, mas pode ser computacionalmente intensivo para grandes conjuntos de dados, exigindo estratégias eficientes, como árvores KD, para melhorar o desempenho (Khorshid; Abdulazeez, 2021).

2.3.2 Naive Bayes Multinomial

O *Naive Bayes* é uma abordagem de aprendizado de máquina baseada no teorema de Bayes. Ele calcula a probabilidade de um objeto pertencer a uma classe específica com base nas probabilidades condicionais das características observadas, assumindo independência entre elas (Macedo et al., 2023). A Figura 2 ilustra o funcionamento do algoritmo *Naive Bayes Multinomial*.

Figura 2 – Exemplo do Funcionamento do Algoritmo *Naive Bayes Multinomial*



Fonte: Elaborado pelo autor (2024).

Neste exemplo, o algoritmo calcula a probabilidade de um documento pertencer a uma classe específica com base na frequência das palavras no documento e na classe. O algoritmo estima as probabilidades a priori das classes a partir do conjunto de treinamento e, em seguida, calcula as probabilidades condicionais das palavras dadas essas classes. Finalmente, a classe com a maior probabilidade é atribuída ao documento.

O *Naive Bayes Multinomial*, que assume que as características são distribuídas multinomialmente, pode ser resumido em três etapas (Olanrewaju; Olanrewaju; Nafiu, 2022):

1. **Estimação das Probabilidades a Priori**: Calcular as probabilidades a priori de cada classe com base na frequência relativa das classes no conjunto de treinamento.

2. **Estimação das Probabilidades Condicionais**: Calcular as probabilidades condicionais de cada característica dada cada classe.

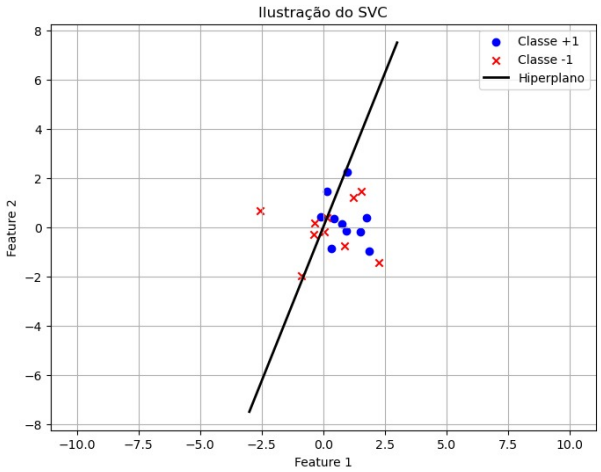
3. **Classificação**: Calcular a probabilidade de cada classe para um novo ponto de dados e selecionar a classe com a maior probabilidade.

O *Naive Bayes Multinomial* é eficiente e simples de implementar, mas a suposição de independência condicional pode não ser realista em todos os casos, afetando o desempenho do modelo (Reddy et al., 2022).

2.3.3 Support Vector Classifier

O *Support Vector Classifier* (SVC), também conhecido como *Support Vector Machine* (SVM), é um algoritmo poderoso usado tanto para classificação quanto para regressão. Ele funciona encontrando o hiperplano que melhor separa as classes no espaço de características (Pisner; Schnyer, 2020). A Figura 3 ilustra o funcionamento do *Support Vector Classifier*.

Figura 3 – Exemplo do Funcionamento do Algoritmo *Support Vector Classifier*



Fonte: Elaborado pelo autor (2024).

No exemplo mostrado, cada ponto de dados é representado em um espaço de características bidimensional, onde as diferentes cores indicam diferentes classes. O SVC encontra o hiperplano que melhor separa essas classes, maximizando a margem entre os pontos de dados das classes diferentes. A margem é a distância entre o hiperplano e os pontos de dados mais próximos de cada classe, conhecidos como vetores de suporte.

O funcionamento do SVC pode ser resumido em três etapas (Lau; Wu, 2003):

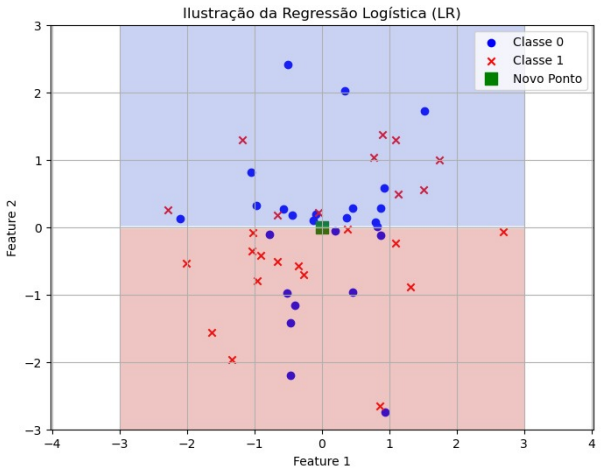
1. **Treinamento**: Encontrar o hiperplano que maximiza a margem entre as classes.
2. **Classificação**: Determinar de que lado do hiperplano o novo ponto de dados está para classificá-lo.
3. **Kernel Trick**: Usar truques de kernel para lidar com problemas de classificação não lineares, mapeando os dados em um espaço de maior dimensionalidade.

O SVC é eficaz em espaços de alta dimensão, mas pode ser sensível à escolha do parâmetro de regularização e do tipo de kernel usado (Alam et al., 2020).

2.3.4 Logistic Regression

A *Logistic Regression* é uma técnica estatística usada para prever a probabilidade de uma variável categórica binária com base em uma ou mais variáveis independentes (Gonzalez, 2018). A Figura 4 ilustra o funcionamento da *Logistic Regression*.

Figura 4 – Exemplo do Funcionamento do Algoritmo *Logistic Regression*



Fonte: Elaborado pelo autor (2024).

No exemplo mostrado, os pontos de dados são plotados em um gráfico bidimensional, onde a linha de decisão logística separa as duas classes. A linha curva representa a função logística, que modela a probabilidade de um ponto de dados pertencer a uma determinada classe com base nas variáveis independentes.

O funcionamento da *Logistic Regression* pode ser resumido em três etapas (Gonzalez, 2018):

1. **Modelagem da Probabilidade**: Modelar a relação entre os recursos e a probabilidade de pertencer a uma determinada classe usando a função logística.

2. **Estimação dos Parâmetros**: Estimar os parâmetros que maximizam a verossimilhança dos dados observados.

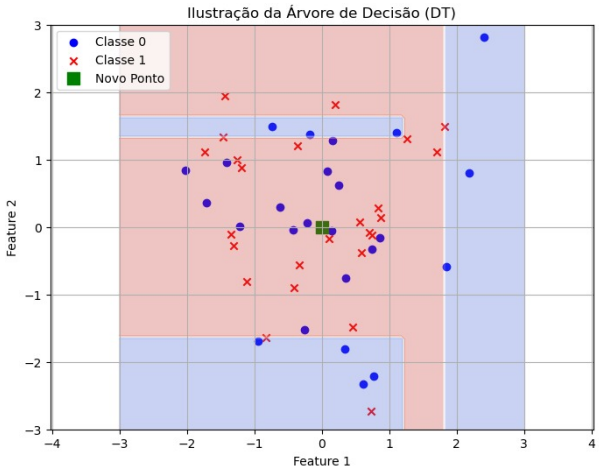
3. **Classificação**: Estimar a probabilidade de pertencer a cada classe e selecionar a classe com a maior probabilidade.

A Regressão Logística é rápida de treinar e prever, fornecendo probabilidades bem calibradas, mas pode não funcionar bem em conjuntos de dados com relações não lineares entre os recursos e as classes (Gonzalez, 2018).

2.3.5 Decision Tree Classifier

Decision Tree Classifier é um modelo de aprendizado de máquina que divide recursivamente o conjunto de dados em subconjuntos mais homogêneos com base em características específicas (Macedo et al., 2023). A Figura 5 ilustra o funcionamento de um *Decision Tree Classifier*.

Figura 5 – Exemplo do Funcionamento do Algoritmo *Decision Tree*



Fonte: Elaborado pelo autor (2024).

Os pontos de dados são representados por círculos azuis (Classe 0) e 'X' vermelhos (Classe 1), enquanto o quadrado verde representa um novo ponto a ser classificado. As áreas coloridas (azul e vermelha) representam as regiões de decisão do classificador, com as linhas delimitadoras mostrando os limites de decisão. A árvore de decisão divide o espaço de características em regiões homogêneas, classificando o novo ponto como pertencente à Classe 0, demonstrando a eficácia da árvore de decisão em segmentar dados de forma interpretável e intuitiva.

O funcionamento do *Decision Tree Classifier* pode ser resumido em três etapas:

1. **Divisão dos Dados**: Dividir recursivamente o conjunto de dados em subconjuntos mais homogêneos.
2. **Construção da Árvore**: Continuar a construção até alcançar uma condição de parada, como profundidade máxima ou pureza das folhas.
3. **Classificação**: Percorrer a árvore a partir da raiz e atribuir ao ponto a classe correspondente à folha em que ele termina.

Decision Tree Classifier são fáceis de interpretar e podem lidar com conjuntos de dados heterogêneos, mas tendem a ser propensos a *overfitting*, especialmente em conjuntos de dados com muitas características (Macedo et al., 2023).

2.4 Métricas de Avaliação de Modelos

A avaliação de modelos é uma parte crucial do processo de construção de algoritmos de aprendizado de máquina. As métricas de avaliação são ferramentas essenciais que nos permitem quantificar o desempenho e a eficácia de um modelo em relação aos dados de teste ou validação. Vamos explorar mais detalhadamente algumas das métricas mais comuns usadas para avaliar modelos de classificação e regressão (Vujović et al., 2021).

Para problemas de classificação, onde o objetivo é prever a classe de um conjunto de dados, várias métricas são comumente utilizadas para avaliar o desempenho do modelo.

2.4.1 Matriz de Confusão

A matriz de confusão é uma tabela que mostra a contagem de verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN) produzidos pelo modelo. Ela fornece uma visão detalhada do desempenho do modelo e pode ajudar a identificar áreas de melhoria. Cada linha representa as instâncias observadas em uma classe, enquanto cada coluna representa as instâncias previstas pelo modelo (Liang, 2022).

Uma representação geral da matriz de confusão é a seguinte:

Tabela 1 – Matriz de Confusão

	Classe Positiva	**Classe Negativa**
Classe Positiva	TP (Verdadeiros Positivos)	FN (Falsos Negativos)
Classe Negativa	FP (Falsos Positivos)	TN (Verdadeiros Negativos)

Fonte: Elaborado pelo autor (2024).

Onde:

- TP (Verdadeiros Positivos): Instâncias positivas que foram corretamente classificadas como positivas.
- FN (Falsos Negativos): Instâncias positivas que foram erroneamente classificadas como negativas.
- FP (Falsos Positivos): Instâncias negativas que foram erroneamente classificadas como positivas.
- TN (Verdadeiros Negativos): Instâncias negativas que foram corretamente classificadas como negativas.

A matriz de confusão é uma ferramenta poderosa para avaliar o desempenho de um modelo de classificação e entender onde ele está acertando ou errando.

2.4.2 Acurácia

A acurácia é uma métrica fundamental que mede a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões realizadas. É calculada como o número de previsões corretas dividido pelo total de previsões. A acurácia é uma métrica útil quando todas as classes têm uma importância semelhante e estão balanceadas no conjunto de dados (Stallings; Gillmore, 1971).

A fórmula da acurácia é dada por:

$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

Vamos considerar um exemplo para ilustrar como calcular a acurácia. Suponha que temos um conjunto de dados de teste com 100 exemplos, e nosso modelo de classificação fez previsões para cada exemplo. Das 100 previsões, 85 estão corretas. Portanto, a acurácia do modelo seria: Acurácia $= \frac{85}{100} = 0.85$, ou seja, o modelo classificou corretamente 85% dos exemplos no conjunto de dados de teste.

2.4.3 Precisão

A precisão mede a proporção de instâncias positivas classificadas corretamente em relação ao total de instâncias classificadas como positivas pelo modelo. Em outras palavras, é a habilidade do modelo de evitar classificar erroneamente uma instância negativa como positiva. A precisão é calculada como o número de verdadeiros positivos dividido pelo número total de instâncias classificadas como positivas pelo modelo (Stallings; Gillmore, 1971).

A fórmula da precisão é dada por:

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

Vamos considerar um exemplo para ilustrar como calcular a precisão. Suponha que temos um modelo de classificação que previu 100 instâncias como positivas. Das 100 instâncias classificadas como positivas, 80 são verdadeiros positivos e 20 são falsos positivos. Portanto, a precisão do modelo seria: Precisão $= \frac{80}{80+20} = 0.8$, ou seja, o modelo tem uma precisão de 80%, ou seja, 80% das instâncias classificadas como positivas são realmente positivas.

2.4.4 Recall

O *recall*, também conhecido como sensibilidade, mede a capacidade do modelo de identificar corretamente todas as instâncias da classe positiva. Ele é calculado como o número de verdadeiros positivos dividido pela soma dos verdadeiros positivos e falsos negativos. Em outras palavras, é a habilidade do modelo de encontrar todas as instâncias positivas (Yacouby; Axman, 2020).

A fórmula do recall é dada por:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

Vamos considerar um exemplo para ilustrar como calcular o *recall*. Suponha que temos um modelo de classificação que identificou corretamente 90 das 100 instâncias positivas. No entanto, ele falhou em identificar corretamente as outras 10 instâncias positivas, classificando-as como negativas. Portanto, o *recall* do modelo seria: Recall $= \frac{90}{90+10} = 0.9$, ou seja, o modelo tem um *recall* de 90%, o que indica que ele conseguiu encontrar corretamente 90% das instâncias positivas.

2.4.5 F1-Score

O *F1-Score* é a média harmônica da precisão e do *recall*. Ele fornece uma maneira de equilibrar a precisão e o *recall* em uma única medida. O *F1-Score* é especialmente útil quando há um desequilíbrio entre as classes ou quando as consequências dos falsos positivos e falsos negativos são desiguais. É calculado como 2 vezes o produto da precisão e do *recall*, dividido pela soma da precisão e do *recall* (Yacouby; Axman, 2020).

A fórmula do *F1-Score* é dada por:

$$\text{F1-Score} = \frac{2 \times \text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{2.4}$$

Vamos considerar um exemplo para ilustrar como calcular o *F1-Score*. Suponha que temos um modelo com uma precisão de 0.8 e um recall de 0.9. O *F1-Score* seria: F1-Score $= \frac{2 \times 0.8 \times 0.9}{0.8+0.9} = 0.84$, ou seja, o modelo tem um *F1-Score* de 0.84, indicando um bom equilíbrio entre precisão e *recall*.

3 METODOLOGIA

Neste capítulo, descrevemos detalhadamente a metodologia adotada para conduzir a pesquisa, incluindo a seleção dos dados, o tratamento e a limpeza dos dados, a execução da pesquisa, a análise dos dados e a identificação das ameaças à validade.

3.1 Seleção dos Dados

A etapa de seleção dos dados desempenha um papel fundamental na qualidade e na validade dos resultados de qualquer pesquisa. Para esta investigação, em particular, a seleção cuidadosa de um conjunto de dados representativo foi crucial para garantir a confiabilidade e a relevância dos resultados obtidos.

Nossa escolha recaiu sobre um conjunto de dados publicamente disponível, concebido especificamente para abordar o fenômeno de mensagens de texto classificadas como *spam* e não *spam*. A justificativa para essa escolha reside na necessidade de acessar um corpus abrangente e diversificado de mensagens que refletisse fielmente a realidade das comunicações digitais contemporâneas.

Ao optarmos por utilizar um conjunto de dados rotulado, onde cada mensagem já está categorizada como *spam* ou não *spam*, visamos facilitar a tarefa de treinamento e avaliação de modelos de classificação de texto. Essa abordagem permite que nos concentremos diretamente na análise das características distintivas entre os dois tipos de mensagens, sem a necessidade de rotulação manual. Além disso, a disponibilidade pública desse conjunto de dados promove a transparência e a reprodutibilidade da pesquisa, possibilitando que outros pesquisadores validem nossos resultados e comparem-nos com estudos anteriores.

O conjunto de dados foi obtido a partir do repositório <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>, que é amplamente utilizado na comunidade de aprendizado de máquina para fins de *benchmarking*[1]. Este repositório contém mensagens de texto rotuladas como *spam* ou não *spam*, o que o torna adequado para nosso estudo.

Antes de avançarmos para a análise propriamente dita, conduzimos uma análise preliminar visando aprofundar nossa compreensão das características do conjunto de dados. Essa etapa envolveu a investigação do volume de entradas, a distribuição dos rótulos (*spam* vs. não *spam*) e a natureza das mensagens contidas no conjunto de dados. Somente após essa análise inicial, procedemos com o tratamento e a limpeza dos dados, preparando-os para as etapas subsequentes da pesquisa.

[1] *Benchmarking* é o processo de comparar e medir as práticas, produtos ou serviços de uma organização em relação aos concorrentes líderes do setor ou às empresas reconhecidas como líderes em determinada área. O objetivo do *benchmarking* é identificar as melhores práticas e melhorias potenciais, além de entender como uma empresa se compara em relação aos seus concorrentes ou à indústria em geral. Pode envolver a análise de métricas de desempenho, processos operacionais, estratégias de *marketing*, satisfação do cliente e muito mais. O *benchmarking* pode ser interno, quando se compara com unidades ou departamentos dentro da mesma organização, ou externo, quando se compara com outras empresas ou organizações externas (Kiela et al., 2021).

3.2 Tratamento e Limpeza dos Dados

O pré-processamento de texto desempenha um papel crucial no tratamento de dados textuais, especialmente quando lidamos com mensagens de *e-mail*, *tweets* ou outras formas de texto não estruturado. Durante essa fase, utilizamos uma série de técnicas para preparar o texto para análise, removendo elementos irrelevantes e simplificando sua estrutura para extrair informações significativas de forma mais eficaz.

Uma técnica comum é a remoção de *stopwords*, que são palavras muito frequentes que ocorrem em praticamente todos os textos, como artigos, preposições e pronomes (Sarica; Luo, 2021). Embora essas palavras desempenhem um papel gramatical importante na linguagem, elas geralmente não carregam informações distintivas sobre o conteúdo do texto. Portanto, removemos todas as *stopwords* a fim de reduzir a dimensionalidade do conjunto de dados, eliminando termos redundantes e facilitando a identificação de padrões relevantes.

Além disso, durante o pré-processamento de texto, aplicamos outras técnicas, como a tokenização e a *stemming* (Dash; Dash, 2021). A tokenização do texto para dividir as mensagens em unidades menores, como palavras individuais ou termos. Esse processo envolve a separação do texto em tokens ou tokens, que servem como as unidades básicas de análise. Cada token representa uma palavra única ou um termo distinto no texto (Song et al., 2020). Além disso, aplicamos a técnica de *stemming* para reduzir as palavras à sua forma raiz ou base. O *stemming* ajuda a normalizar o texto, eliminando variações morfológicas e consolidando diferentes formas de uma palavra em uma única representação. Por exemplo, as palavras "correr", "corria"e "corrida"podem ser reduzidas à forma raiz "corr". Isso simplifica o texto e reduz a redundância, facilitando a identificação de padrões e insights durante a análise de dados.

3.3 Execução da Pesquisa

Após concluir as etapas de tratamento e limpeza dos dados, avançamos para a fase de execução da pesquisa, conforme planejado previamente. Nesta etapa, o foco principal foi a aplicação de algoritmos de classificação de texto, os quais são essenciais para categorizar as mensagens em diferentes grupos ou classes, como spam e não spam, por exemplo.

Para realizar essa tarefa, empregamos uma variedade de algoritmos de aprendizado de máquina e técnicas de PLN, adaptadas às características específicas do conjunto de dados e aos objetivos da pesquisa. Esses algoritmos foram treinados utilizando os dados pré-processados e limpos, com o intuito de aprender padrões e relações entre os diferentes elementos do texto.

Após a fase de treinamento, os modelos foram avaliados quanto ao seu desempenho usando métricas específicas, tais como precisão, recall, F1-score e matriz de confusão. Essas métricas fornecem uma avaliação quantitativa do quão bem o modelo é capaz de classificar as mensagens corretamente em suas respectivas categorias.

A avaliação do desempenho dos modelos foi crucial para determinar sua eficácia e identificar possíveis áreas de melhoria. Com base nos resultados obtidos, podemos ajustar os

parâmetros do modelo, experimentar diferentes algoritmos ou técnicas de pré-processamento e otimizar a configuração geral do sistema para melhorar o desempenho da classificação.

3.4 Análise dos Dados

Após a execução da pesquisa, foi conduzida uma análise minuciosa dos resultados obtidos. Essa etapa envolveu a interpretação das métricas de desempenho dos modelos de classificação de texto, bem como a exploração dos padrões e tendências presentes nos dados.

Inicialmente, foram examinadas as métricas de desempenho dos modelos, incluindo precisão, recall, F1-score e outras métricas relevantes. Essas métricas forneceram informações sobre a eficácia dos modelos na classificação das mensagens, destacando suas capacidades de discriminação entre diferentes categorias. Além disso, a análise da matriz de confusão permitiu identificar os tipos de erros cometidos pelos modelos, ajudando a entender melhor suas limitações e áreas de melhoria.

Além da avaliação do desempenho dos modelos, a análise dos dados também envolveu a identificação de padrões e tendências presentes nas mensagens classificadas. Isso incluiu a análise de palavras-chave mais frequentes em cada categoria, a detecção de tópicos dominantes nas mensagens e a identificação de padrões de comportamento dos usuários. Essa análise exploratória dos dados ajudou a compreender melhor a natureza das mensagens e os contextos em que são geradas.

3.5 Ameaças à Validade

Ao concluir a pesquisa, é crucial identificar e discutir as potenciais ameaças à validade que podem comprometer a confiabilidade e a generalização dos resultados obtidos. Esta seção aborda as principais ameaças identificadas durante o processo de pesquisa e discute suas implicações para a validade dos resultados.

Uma das principais ameaças à validade é a questão da generalização dos resultados. Os dados utilizados na pesquisa podem não representar adequadamente a população de interesse, o que limita a capacidade de generalizar os resultados para contextos mais amplos. Além disso, a utilização de um único conjunto de dados pode não capturar toda a variabilidade do fenômeno estudado, o que pode afetar a validade externa dos resultados.

Outra ameaça importante é o viés nos dados. Os conjuntos de dados podem conter viéses sistemáticos devido à forma como foram coletados ou rotulados, o que pode distorcer as conclusões da pesquisa. É crucial estar ciente desses viéses e tomar medidas para mitigá-los durante a análise dos dados.

Além disso, as limitações dos modelos utilizados na pesquisa podem representar uma ameaça à validade dos resultados. Os modelos de *machine learning* e técnicas estatísticas podem ter limitações inerentes que afetam sua capacidade de capturar a complexidade do fenômeno estudado. É importante reconhecer essas limitações e interpretar os resultados com cautela.

Outras ameaças à validade incluem questões relacionadas à confiabilidade dos dados, como erros de medição ou ruído nos dados, e a presença de variáveis não controladas que podem influenciar os resultados da pesquisa de forma inesperada.

4 RESULTADOS E DISCUSSÕES

Neste capítulo, apresentamos os resultados e discussões relacionados ao tratamento dos dados, análise exploratória dos dados, pré-processamento do texto, construção e avaliação dos modelos em PLN na detecção de *spam*.

4.1 Tratamento dos Dados

Nesta seção, abordamos as etapas cruciais realizadas no tratamento dos dados antes da análise propriamente dita. Essas etapas são essenciais para garantir a qualidade e a eficácia das análises subsequentes.

4.1.1 Leitura e Preparação dos Dados

Adquirimos o conjunto de dados a partir de um arquivo CSV denominado 'spam.csv', disponível em https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset. Para efetuar a leitura e manipulação dos dados, utilizamos a biblioteca Pandas. Inicialmente, inspecionamos 5 mensagens aleatórias, conforme mostrado na Figura 6, para compreender a estrutura do conjunto de dados, que é composto por 5572 entradas e 5 colunas distintas.

Figura 6 – Exemplo de Mensagens Aleatórias do Conjunto de Dados

```
      v1                                                 v2 Unnamed: 2  \
3944  ham  I will be gentle princess! We will make sweet …        NaN
5136  ham  There are some nice pubs near here or there is…        NaN
1233  ham                Lol ok. I'll snatch her purse too.         NaN
2092  ham  Oh, my love, it's soooo good to hear from you…        NaN
162   ham  I'm so in love with you. I'm excited each day …        NaN

     Unnamed: 3 Unnamed: 4
3944        NaN        NaN
5136        NaN        NaN
1233        NaN        NaN
2092        NaN        NaN
162         NaN        NaN
```

Fonte: Elaborado pelo autor (2024).

4.1.2 Limpeza dos Dados

Em seguida, procedemos à limpeza dos dados, uma etapa fundamental para assegurar a qualidade das análises subsequentes. Eliminamos as colunas vazias identificadas como *'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'*, pois não contribuíam significativamente para o processo de detecção de *spam*. Também removemos mensagens duplicadas, resultando em um conjunto de dados mais coeso e confiável, totalizando 5169 entradas.

Figura 7 – Exemplo de Mensagens Aleatórias do Conjunto de Dados após a Limpeza

	v1	v2
1659	ham	Yeah, where's your class at?
2988	spam	No 1 POLYPHONIC tone 4 ur mob every week! Just…
3121	spam	Free entry in 2 a weekly comp for a chance to …
1111	ham	Awesome, think we can get an 8th at usf some t…
4012	ham	Ok.

Fonte: Elaborado pelo autor (2024).

4.1.3 Pré-processamento dos Dados

Na fase de pré-processamento de texto, renomeamos as colunas remanescentes para aprimorar a compreensão, conforme apresentado na Figura 8. Especificamente, alteramos *'v1'* para *'Rótulo'* e *'v2'* para '*Mensagem'*. Essa medida tornou mais claro e interpretável o significado de cada coluna no conjunto de dados.

Figura 8 – Exemplo de Mensagens Aleatórias do Conjunto de Dados após a Nomeação das Colunas

	Rotulo	Mensagem
1020	ham	Good afternoon on this glorious anniversary da…
3829	ham	I agree. So i can stop thinkin about ipad. Can…
1904	ham	Wah… Okie okie… Muz make use of e unlimite…
187	spam	Please call our customer service representativ…
3654	ham	Senthil group company Apnt 5pm.

Fonte: Elaborado pelo autor (2024).

Adicionalmente, codificamos os rótulos categóricos em valores numéricos utilizando a classe *LabelEncoder* da biblioteca *scikit-learn*. Associamos o valor 1 para mensagens classificadas como *spam* e 0 para aquelas rotuladas como *ham*. Essa etapa viabilizou a manipulação eficiente dos rótulos durante o treinamento e avaliação dos modelos de detecção de *spam*.

Figura 9 – Exemplo de Mensagens Aleatórias do Conjunto de Dados após a Codificação dos Rótulos

	Rotulo	Mensagem
0	0	Go until jurong point, crazy.. Available only …
1	0	Ok lar… Joking wif u oni…
2	1	Free entry in 2 a wkly comp to win FA Cup fina…
3	0	U dun say so early hor… U c already then say…
4	0	Nah I don't think he goes to usf, he lives aro…

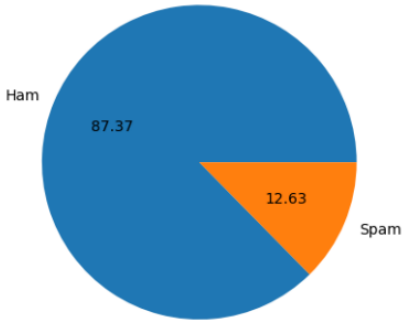
Fonte: Elaborado pelo autor (2024).

4.2 Análise Exploratória dos Dados

Realizamos a análise exploratória dos dados para fornecer percepções valiosas sobre a distribuição das mensagens rotuladas como *ham* (não *spam*) e *spam*. A Figura 10 exibe o gráfico de pizza, que evidenciou uma notável discrepância na distribuição dos rótulos. Notamos que

4516 mensagens foram classificadas como *ham*, indicando conteúdo não *spam*, enquanto 653 mensagens foram identificadas como *spam*.

Figura 10 – Distribuição de Mensagens *ham* e *spam* no Conjunto de Dados

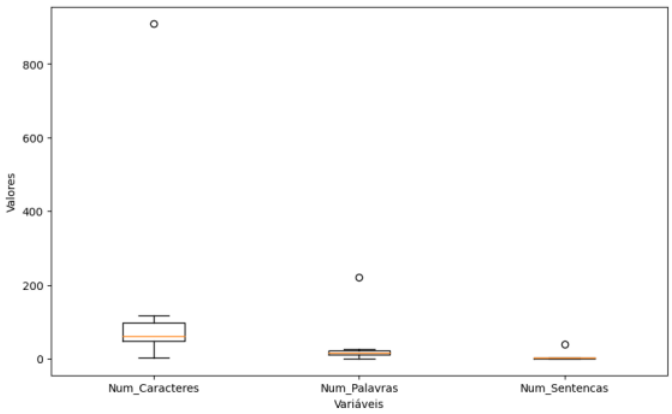


Fonte: Elaborado pelo autor (2024).

Essa disparidade na distribuição dos rótulos é importante para o desenvolvimento e avaliação dos modelos de detecção de *spam*. Compreender essa assimetria é fundamental para garantir que os modelos sejam treinados e avaliados de maneira equitativa e eficiente.

Utilizamos a biblioteca NLTK para a tokenização das mensagens, facilitando a contagem de palavras e sentenças. A tokenização é uma etapa do pré-processamento de texto que permite uma representação mais eficiente das informações contidas nas mensagens. Na Figura 11, apresentamos as características textuais das mensagens, analisando o número de caracteres, palavras e sentenças presentes em cada uma delas.

Figura 11 – Estatísticas das Características Textuais no Conjunto de Dados



Fonte: Elaborado pelo autor (2024).

As estatísticas descritivas das características textuais revelaram informações significativas sobre sua natureza e distribuição. Em média, as mensagens apresentam cerca de 79 caracteres e 18 palavras, com variação considerável evidenciada pelos desvios padrão de aproximadamente 58 e 13, respectivamente. Além disso, a média de sentenças por mensagem é de aproximadamente 2, indicando que a maioria das mensagens consiste em uma ou duas sentenças. A presença de valores extremos destaca a diversidade de comprimentos de mensagens no conjunto de dados.

4.3 Pré-processamento do Texto

O pré-processamento de dados prepara os textos para análises subsequentes. Aplicamos diversas técnicas para transformar e aprimorar as mensagens presentes no conjunto de dados. A função de pré-processamento de texto é apresentada na Figura 12.

Figura 12 – Função de Pré-processamento e Normalização de Texto

```
def Texto_Transformado(text):
    text = text.lower()
    text = nltk.word_tokenize(text)
    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in
           string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
```

Fonte: Elaborado pelo autor (2024).

Inicialmente, transformamos todas as letras nas mensagens em minúsculas usando a função *lower()*. Em seguida, tokenizamos as mensagens utilizando a função *word_tokenize()* da biblioteca NLTK, conforme ilustrado na Figura 13.

Figura 13 – Trecho da Função de Pré-processamento e Normalização de Texto - Tokenização

```
def Texto_Transformado(text):
    text = text.lower()
    text = nltk.word_tokenize(text)
```

Fonte: Elaborado pelo autor (2024).

Limpamos o texto de caracteres não alfabéticos, utilizamos um *loop* para verificar se as palavras contêm apenas caracteres alfanuméricos, e removemos *stopwords* e pontuações. Aplicamos a técnica de *stemming* para reduzir as palavras ao seu radical, utilizando a biblioteca NLTK e o módulo *nltk.stem.porter*, conforme o trecho de código ilustrado na Figura 14.

Figura 14 – Trecho do Código - *Stemming*

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')

'love'
```

Fonte: Elaborado pelo autor (2024).

A implementação da função *Texto_Transformado* consolidou essas etapas, resultando em uma representação textual mais uniforme e adequada para análises subsequentes.

4.4 Construção dos Modelos

Durante a elaboração dos modelos, empregamos algoritmos de classificação e técnicas de processamento de texto para criar modelos capazes de distinguir entre mensagens *ham* (não *spam*) e *spam*.

4.4.1 Term Frequency-Inverse Document Frequency

A representação das mensagens de maneira numérica permite a sua utilização em algoritmos de aprendizado de máquina. Utilizamos a técnica *Term Frequency-Inverse Document Frequency* (TF-IDF), que dá peso às palavras conforme a frequência com que aparecem em um documento específico e a frequência inversa em um conjunto de documentos.

Configuramos o *TfidfVectorizer* para considerar no máximo 3000 *features* e aplicamos o método *fit_transform* para ajustar o modelo aos dados e transformar o texto pré-processado em uma matriz numérica, conforme ilustrado na Figura 15.

Figura 15 – Vetorização de Texto com TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['Texto_Transformado']).toarray()
```

Fonte: Elaborado pelo autor (2024).

A matriz resultante possui dimensões (5169, 3000), indicando 5169 mensagens e 3000 *features*. Essa representação numérica é fundamental para a construção de modelos de aprendizado de máquina, permitindo que os algoritmos processem e aprendam com os dados textuais.

4.4.2 Hold-Out

Utilizamos a técnica *Hold-Out* para a preparação dos dados para o treinamento e teste dos modelos. Ela consiste na divisão do conjunto de dados em dois subconjuntos: um conjunto

de treinamento e um conjunto de teste. Utilizamos a função *train_test_split* do pacote *sklearn.model_selection* para dividir os dados em 70% para treinamento e 30% para teste, conforme ilustrado na Figura 16.

Figura 16 – Técnica *Hold-Out*

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test =
    train_test_split(X,y,test_size=0.7,random_state=2)
```

Fonte: Elaborado pelo autor (2024).

Utilizamos o parâmetro *random_state* para garantir que a divisão seja reproduzível. Essa abordagem permite avaliar o desempenho dos modelos em um conjunto independente de dados, fornecendo uma estimativa mais confiável de sua capacidade de generalização.

4.4.3 Construção dos Modelos

Para a construção do modelo *K-Nearest Neighbors*, utilizamos a classe *KNeighborsClassifier* do *sklearn*. Criamos uma instância dessa classe para o modelo *K-Nearest Neighbors*, conforme ilustrado na Figura 17.

Figura 17 – *K-Nearest Neighbors*

```
from sklearn.neighbors import KNeighborsClassifier

knc = KNeighborsClassifier()
```

Fonte: Elaborado pelo autor (2024).

Definimos uma função chamada *train_classifier* para treinar o classificador e avaliar seu desempenho utilizando métricas como acurácia, precisão, *recall* e *F1-Score*, conforme ilustrado na Figura 18. Utilizamos essa função para treinar o modelo e avaliar seu desempenho com base nos dados de treinamento e teste. Para os algoritmos *Naive Bayes Multinomial*, *Support Vector Classifier*, *Logistic Regression* e *Decision Tree Classifier*, utilizamos diferentes pacotes do *Scikit-Learn*, conforme ilustrado na Figura 19.

Figura 18 – Treinamento e Avaliação de Classificadores

```
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    return accuracy,precision,recall, f1

train_classifier(knc,X_train,y_train,X_test,y_test)
```

Fonte: Elaborado pelo autor (2024).

Figura 19 – Inicialização de Modelos de Classificação

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

mnb = MultinomialNB()
svc = SVC(kernel='sigmoid', gamma=1.0)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
dtc = DecisionTreeClassifier(max_depth=5)
```

Fonte: Elaborado pelo autor (2024).

4.5 Avaliação dos Modelos

Nesta seção, apresentamos uma avaliação comparativa do desempenho de diferentes algoritmos de classificação na detecção de *spam* em mensagens de e-mail. Utilizamos métricas como acurácia, precisão, *recall* e *F1-Score*, além de criar matrizes de confusão para uma análise mais aprofundada, conforme mostrado na Tabela 2.

Tabela 2 – Resultados do Desempenho dos Algoritmos de Classificação de *Spam*

Algoritmo	**Acurácia**	**Precisão**	**Recall**	**F1-Score**
K-Nearest Neighbors	0.8944	1.0000	0.1157	0.2075
Naive Bayes Multinomial	0.9539	1.0000	0.6134	0.7604
Support Vector Classifier	0.9699	0.9549	0.7847	0.8615
Logistic Regression	0.9356	0.8902	0.5255	0.6608
Decision Tree Classifier	0.9367	0.7557	0.6944	0.7238

Fonte: Elaborado pelo autor (2024).

As Tabelas 3, 4, 5, 6 e 7 exibem as matrizes de confusão para cada algoritmo de classificação avaliado. As matrizes de confusão destacam o número de verdadeiros positivos, verdadeiros

negativos, falsos positivos e falsos negativos para cada algoritmo, permitindo uma avaliação visual do desempenho de cada modelo na classificação de mensagens como *spam* ou não *spam*.

Tabela 3 – Matriz de Confusão *K-Nearest Neighbors*

		Previsão	
		Spam	***Ham***
Real	***Spam***	50	382
	Ham	0	3187

Fonte: Elaborado pelo autor (2024).

Tabela 4 – Matriz de Confusão *Naive Bayes Multinomial*

		Previsão	
		Spam	***Ham***
Real	***Spam***	264	167
	Ham	0	3187

Fonte: Elaborado pelo autor (2024).

Tabela 5 – Matriz de Confusão *Support Vector Classifier*

		Previsão	
		Spam	***Ham***
Real	***Spam***	339	93
	Ham	16	3171

Fonte: Elaborado pelo autor (2024).

Tabela 6 – Matriz de Confusão *Logistic Regression*

		Previsão	
		Spam	***Ham***
Real	***Spam***	227	205
	Ham	28	3159

Fonte: Elaborado pelo autor (2024).

Tabela 7 – Matriz de Confusão *Decision Tree Classifier*

		Previsão	
		Spam	***Ham***
Real	***Spam***	298	134
	Ham	97	3090

Fonte: Elaborado pelo autor (2024).

Os resultados indicam o desempenho de diferentes modelos na tarefa de classificação. A coluna “Acurácia” representa a proporção de predições corretas em relação ao total de predições

realizadas pelo modelo. Observa-se que o *Support Vector Classifier* obteve a maior acurácia, com um valor de 0.9699, seguido pelo *Naive Bayes* com 0.9539.

A coluna Precisão mostra a proporção de verdadeiros positivos (mensagens classificadas corretamente como *spam*) em relação ao total de mensagens classificadas como *spam* pelo modelo. Todos os modelos apresentaram alta precisão, com valores próximos ou iguais a 1.

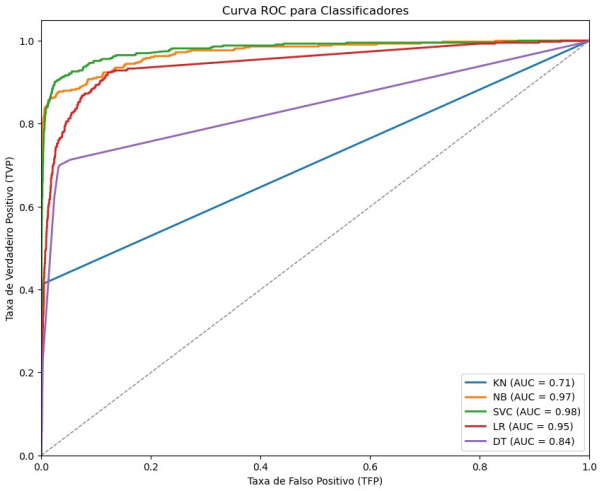
O *Recall*, também conhecido como taxa de verdadeiros positivos, indica a proporção de mensagens de *spam* que foram corretamente identificadas pelo modelo em relação ao total de mensagens de *spam* no conjunto de dados. O *Support Vector Classifier* teve o maior recall (0.7847). O *F1-Score* é uma medida que combina precisão e recall. Novamente, o *Support Vector Classifier* obteve o maior F1-Score (0.8615), seguido pelo *Naive Bayes* (0.7604).

Com base na análise dos dados apresentados, pode-se concluir que o *Support Vector Classifier* obteve o melhor desempenho em relação aos outros algoritmos avaliados, sendo a escolha mais eficaz para a tarefa específica de detecção de *spam*.

Além das métricas de acurácia, precisão, *recall* e *F1-Score*, também utilizamos a curva Receiver Operating Characteristic (ROC) para avaliar o desempenho dos modelos. A curva ROC é uma representação gráfica que ilustra a performance de um modelo de classificação em diferentes pontos de corte, traçando a taxa de verdadeiro positivo (TVP) contra a taxa de falso positivo (TFP).

A Figura 20 apresenta as Curvas ROC para os classificadores *K-Nearest Neighbors* (KN), *Naive Bayes* (NB), *Support Vector Classifier* (SVC), *Logistic Regression* (LR) e *Decision Tree* (DT).

Figura 20 – Curva ROC para Classificadores



Fonte: Elaborado pelo autor (2024).

A curva ROC é uma ferramenta eficaz para comparar os modelos de classificação, e a Área Sob a Curva (AUC) é usada como uma métrica de desempenho. A AUC varia de 0 a 1, onde 1 indica um classificador perfeito e 0.5 indica um classificador aleatório.

Os resultados mostrados na Figura 20 indicam que o *Support Vector Classifier* (SVC) tem a maior AUC (0.98), seguido pelo *Naive Bayes* (0.97) e pela *Logistic Regression* (0.95). O *Decision Tree* (DT) e o *K-Nearest Neighbors* (KN) têm AUCs menores, 0.84 e 0.71, respectivamente.

Esses resultados corroboram as métricas anteriores, indicando que o *Support Vector Classifier* é o modelo mais eficaz para a detecção de *spam* nas mensagens de e-mail. A AUC alta demonstra que o SVC tem uma capacidade discriminativa superior, conseguindo separar as classes de *spam* e não *spam* com alta precisão. A análise da Curva ROC e da AUC é fundamental para comparar a eficácia dos diferentes modelos de classificação, permitindo uma escolha informada do modelo mais adequado para o problema em questão.

5 CONSIDERAÇÕES FINAIS E SUGESTÕES PARA TRABALHOS FUTUROS

A detecção de *spam* em mensagens de *e-mail* é uma etapa para a proteção contra conteúdos indesejados e para a preservação da integridade e segurança dos usuários. Neste trabalho, exploramos técnicas de processamento de texto e algoritmos de classificação para identificar *spams* em mensagens de *e-mail*.

Tratamos e pré-processamos os dados, incluindo limpeza, tokenização e codificação dos rótulos. Essas etapas foram essenciais para garantir a qualidade e consistência dos dados nos modelos. Em seguida, analisamos os dados e notamos uma grande diferença na distribuição das mensagens classificadas como *spams* e não *spams*, reforçando a importância de abordagens equilibradas durante o treinamento e avaliação dos modelos.

Para a criação dos modelos, empregamos diversos métodos de classificação, tais como *K-Nearest Neighbors*, *Naive Bayes Multinomial*, *Support Vector Classifier*, *Logistic Regression* e *Decision Tree Classifier*. Avaliamos o desempenho de cada modelo utilizando medidas como acurácia, precisão, *recall* e *F1-Score*. Os resultados obtidos demonstraram que o *Support Vector Classifier* apresentou o melhor desempenho, alcançando a maior acurácia, *recall* e *F1-Score* entre todos os modelos avaliados. Destacando a sua eficiência na identificação precisa de mensagens de *spam*, mantendo um bom equilíbrio entre precisão e *recall*.

Este estudo destaca a importância do uso de técnicas avançadas de processamento de texto e algoritmos de classificação na detecção de *spam* em mensagens de *e-mail*. Os modelos desenvolvidos têm o potencial de auxiliar na proteção dos usuários contra conteúdo indesejado, contribuindo para uma experiência mais segura e eficiente no uso de serviços de *e-mail*. No entanto, ressalta-se a necessidade de contínuos aprimoramentos e atualizações para lidar com a constante evolução das estratégias de *spamming*. Diante dessa análise, surgem diversas oportunidades promissoras para pesquisas futuras.

Uma recomendação valiosa é a exploração de algoritmos adicionais de classificação, como Redes Neurais, *Random Forests* e *Gradient Boosting*, para aprimorar ainda mais a detecção de *spam*. Além disso, sugerimos diversificar a representação de texto além do uso exclusivo da técnica TF-IDF, considerando a aplicação de *embeddings* de palavras, como *Word2Vec* ou *GloVe*. Essa abordagem visa obter uma representação textual mais rica e semântica dos documentos, contribuindo significativamente para a eficácia na identificação de *spam*.

Outra sugestão importante é substituir a técnica de *Hold Out* por validação cruzada para uma avaliação mais robusta dos algoritmos, permitindo estimativas mais confiáveis ao treinar e testar modelos em conjuntos de dados diferentes. Além disso, propõe-se a realização de análises mais detalhadas das características influentes na detecção de *spam*, incluindo a identificação de palavras-chave específicas e a extração de características avançadas. Explorar abordagens de aprendizado de conjunto e considerar a detecção de *spam* em diferentes modalidades, como imagens ou áudio, são caminhos promissores para enriquecer a compreensão dessa problemática em constante evolução na pesquisa de aprendizado de máquina e PLN.

REFERÊNCIAS

ALAM, S. et al. One-class support vector classifiers: A survey. *Knowledge-Based Systems*, Elsevier, v. 196, p. 105754, 2020. Citado na página 25.

ALKHALIL, Z. et al. Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science*, Frontiers Media SA, v. 3, p. 563060, 2021. Citado na página 14.

ALSUBHI, A. et al. Awareness of security threats in social media. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, v. 12, n. 10, p. 3093–3100, 2021. Citado na página 20.

ALSULAMI, M. M.; AL-AAMA, A. Y. Sentifilter: A personalized filtering model for arabic semi-spam content based on sentimental and behavioral analysis. *Int. J. Adv. Comput. Sci. Appl*, v. 11, n. 2, 2020. Citado na página 20.

ALTHOBAITI, M. M. Assessing user's susceptibility and awareness of cybersecurity threats. *Intelligent Automation & Soft Computing*, v. 28, n. 1, 2021. Citado na página 20.

ASDAGHI, F.; SOLEIMANI, A. An effective feature selection method for web spam detection. *Knowledge-Based Systems*, Elsevier, v. 166, p. 198–206, 2019. Citado na página 19.

ASLAN, Ö. A.; SAMET, R. A comprehensive review on malware detection approaches. *IEEE access*, IEEE, v. 8, p. 6249–6271, 2020. Citado na página 14.

BAHJA, M. Natural language processing applications in business. *E-Business-higher education and intelligence applications*, London, United Kingdom: IntechOpen, 2020. Citado na página 21.

BUIL-GIL, D.; BARRETT, E. et al. The dynamics of business, cybersecurity and cyber-victimization: Foregrounding the internal guardian in prevention. In: *The New Technology of Financial Crime*. [S.l.]: Routledge, 2022. p. 5–34. Citado na página 19.

CAO, J.; LAI, C. A bilingual multi-type spam detection model based on m-bert. In: IEEE. *GLOBECOM 2020-2020 IEEE Global Communications Conference*. [S.l.], 2020. p. 1–6. Citado na página 20.

CARPENTER, J. P. et al. Spam and educators' twitter use: Methodological challenges and considerations. *TechTrends*, Springer, v. 64, p. 460–469, 2020. Citado na página 20.

CHOWDHARY, K.; CHOWDHARY, K. Natural language processing. *Fundamentals of artificial intelligence*, Springer, p. 603–649, 2020. Citado na página 21.

CRANOR, L. F.; LAMACCHIA, B. A. Spam! *Communications of the ACM*, ACM New York, NY, USA, v. 41, n. 8, p. 74–83, 1998. Citado na página 18.

DASH, N. S.; DASH, N. S. Lemmatization of inflected nouns. *Language Corpora Annotation and Processing*, Springer, p. 165–194, 2021. Citado na página 32.

DEEPAK, S.; AMEER, P. Automated categorization of brain tumor from mri using cnn features and svm. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 12, n. 8, p. 8357–8369, 2021. Citado na página 22.

DOSHI, J. et al. A comprehensive dual-layer architecture for phishing and spam email detection. *Computers & Security*, Elsevier, v. 133, p. 103378, 2023. Citado na página 19.

ELBAGIR, S.; YANG, J. Twitter sentiment analysis using natural language toolkit and vader sentiment. In: *Proceedings of the international multiconference of engineers and computer scientists*. [S.l.: s.n.], 2019. v. 122, p. 16. Citado na página 21.

FERRARA, E. The history of digital spam. *Communications of the ACM*, ACM New York, NY, USA, v. 62, n. 8, p. 82–91, 2019. Citado na página 19.

FROMMHOLZ, I. et al. On textual analysis and machine learning for cyberstalking detection. *Datenbank-Spektrum*, Springer, v. 16, p. 127–135, 2016. Citado 2 vezes nas páginas 18 e 19.

GASPARETTO, A. et al. A survey on text classification algorithms: From text to predictions. *Information*, MDPI, v. 13, n. 2, p. 83, 2022. Citado na página 22.

GONZALEZ, L. d. A. Regressão logística e suas aplicações. Universidade Federal do Maranhão, 2018. Citado 2 vezes nas páginas 25 e 26.

HAN, J.; PEI, J.; TONG, H. *Data mining: concepts and techniques*. [S.l.]: Morgan kaufmann, 2022. Citado na página 22.

HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. *Science*, American Association for the Advancement of Science, v. 349, n. 6245, p. 261–266, 2015. Citado na página 21.

JIANG, Q. et al. A challenge dataset and effective models for aspect-based sentiment analysis. In: *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. [S.l.: s.n.], 2019. p. 6280–6285. Citado na página 21.

JINDAL, N.; LIU, B. Review spam detection. In: *Proceedings of the 16th international conference on World Wide Web*. [S.l.: s.n.], 2007. p. 1189–1190. Citado na página 18.

JUNNARKAR, A. et al. E-mail spam classification via machine learning and natural language processing. In: IEEE. *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. [S.l.], 2021. p. 693–699. Citado na página 14.

KAHRAMAN, G. M. *Characterizing Sender Policy Framework configurations at scale*. Dissertação (Mestrado) — University of Twente, 2020. Citado na página 20.

KARIM, A. et al. A comprehensive survey for intelligent spam email detection. *IEEE Access*, IEEE, v. 7, p. 168261–168295, 2019. Citado 2 vezes nas páginas 14 e 19.

KAWINTIRANON, K.; SINGH, L.; BUDAK, C. Traditional and context-specific spam detection in low resource settings. *Machine Learning*, Springer, v. 111, n. 7, p. 2515–2536, 2022. Citado na página 18.

KHORSHID, S. F.; ABDULAZEEZ, A. M. Breast cancer diagnosis based on k-nearest neighbors: a review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, v. 18, n. 4, p. 1927–1951, 2021. Citado na página 23.

KHURANA, D. et al. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, Springer, v. 82, n. 3, p. 3713–3744, 2023. Citado na página 21.

KIELA, D. et al. Dynabench: Rethinking benchmarking in nlp. *arXiv preprint arXiv:2104.14337*, 2021. Citado na página 31.

KOGGALAHEWA, D. N.; XU, Y.; FOO, E. Spam detection in social networks based on peer acceptance. In: *Proceedings of the Australasian Computer Science Week Multiconference*. [S.l.: s.n.], 2020. p. 1–7. Citado na página 18.

LAROSE, D. T.; LAROSE, C. D. k-nearest neighbor algorithm. Wiley Data and Cybersecurity, 2014. Citado na página 23.

LAU, K.; WU, Q. Online training of support vector classifier. *Pattern Recognition*, Elsevier, v. 36, n. 8, p. 1913–1920, 2003. Citado na página 25.

LIANG, J. Confusion matrix: Machine learning. *POGIL Activity Clearinghouse*, v. 3, n. 4, 2022. Citado na página 28.

MACEDO, L. d. N. et al. Comparação de algoritmos de aprendizado de máquina para predizer futuras cepas do vírus da influenza. Universidade Federal de Uberlândia, 2023. Citado 3 vezes nas páginas 23, 26 e 27.

MARZUKI, K.; HANIF, N.; HARIYADI, I. P. Application of domain keys identified mail, sender policy framework, anti-spam, and anti-virus: The analysis on mail servers. *International Journal of Electronics and Communications Systems*, v. 2, n. 2, p. 65–73, 2022. Citado na página 20.

MEHROTRA, T. et al. Email spam filtering technique from various perspectives using machine learning algorithms. In: SPRINGER. *Data Driven Approach Towards Disruptive Technologies: Proceedings of MIDAS 2020*. [S.l.], 2021. p. 423–432. Citado na página 20.

MENG, Y. et al. Generating training data with language models: Towards zero-shot language understanding. *Advances in Neural Information Processing Systems*, v. 35, p. 462–477, 2022. Citado na página 21.

NIVEDHA, M.; RAJA, S. Detection of email spam using natural language processing based random forest approach. *International Journal of Computer Science and Mobile Computing*, v. 11, n. 2, p. 7–22, 2022. Citado na página 15.

O'HALLORAN, K. L.; PAL, G.; JIN, M. Multimodal approach to analysing big social and news media data. *Discourse, Context & Media*, Elsevier, v. 40, p. 100467, 2021. Citado na página 21.

OLANREWAJU, R. O.; OLANREWAJU, S. A.; NAFIU, L. A. Multinomial naïve bayes classifier: Bayesian versus nonparametric classifier approach. *European Journal of Statistics*, v. 2, p. 8–8, 2022. Citado na página 24.

OLIVEIRA, B. S. N. et al. Processamento de linguagem natural via aprendizagem profunda. *Sociedade Brasileira de Computação*, 2022. Citado na página 14.

ORA, A. *Spam detection in short message service using natural language processing and machine learning techniques*. Tese (Doutorado) — Dublin, National College of Ireland, 2020. Citado na página 15.

PAAVOLAINEN, S.; ELO, T.; NIKANDER, P. Risks from spam attacks on blockchains for intemet-of-things devices. In: IEEE. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. [S.l.], 2018. p. 314–320. Citado na página 19.

PERA, M. S.; NG, Y.-K. Identifying spam web pages based on content similarity. In: SPRINGER. *International Conference on Computational Science and Its Applications*. [S.l.], 2008. p. 204–219. Citado na página 19.

PISNER, D. A.; SCHNYER, D. M. Support vector machine. In: *Machine learning*. [S.l.]: Elsevier, 2020. p. 101–121. Citado na página 24.

RAO, S.; VERMA, A. K.; BHATIA, T. A review on social spam detection: challenges, open issues, and future directions. *Expert Systems with Applications*, Elsevier, v. 186, p. 115742, 2021. Citado 2 vezes nas páginas 14 e 15.

REDDY, E. M. K. et al. Introduction to naive bayes and a review on its subtypes with applications. *Bayesian reasoning and gaussian processes for machine learning applications*, Chapman and Hall/CRC, p. 1–14, 2022. Citado na página 24.

REDDY, G. T. et al. Analysis of dimensionality reduction techniques on big data. *Ieee Access*, IEEE, v. 8, p. 54776–54788, 2020. Citado na página 14.

ROY, P. K.; SINGH, J. P.; BANERJEE, S. Deep learning to filter sms spam. *Future Generation Computer Systems*, Elsevier, v. 102, p. 524–533, 2020. Citado na página 14.

SAHOO, S. R. et al. Behavioral analysis to detect social spammer in online social networks (osns). In: SPRINGER. *Computational Data and Social Networks: 9th International Conference, CSoNet 2020, Dallas, TX, USA, December 11–13, 2020, Proceedings 9*. [S.l.], 2020. p. 321–332. Citado na página 20.

SAIDANI, N.; ADI, K.; ALLILI, M. S. A semantic-based classification approach for an enhanced spam detection. *Computers & Security*, Elsevier, v. 94, p. 101716, 2020. Citado 2 vezes nas páginas 15 e 16.

SARICA, S.; LUO, J. Stopwords in technical language processing. *Plos one*, Public Library of Science San Francisco, CA USA, v. 16, n. 8, p. e0254937, 2021. Citado na página 32.

SCHÜTZE. *Introduction to information retrieval*. [S.l.]: Cambridge University Press Cambridge, 2008. v. 39. Citado na página 21.

SEN, P. C.; HAJRA, M.; GHOSH, M. Supervised classification algorithms in machine learning: A survey and review. In: SPRINGER. *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*. [S.l.], 2020. p. 99–111. Citado na página 14.

SILVA, J. A. Teixeira da; AL-KHATIB, A.; TSIGARIS, P. Spam emails in academia: issues and costs. *Scientometrics*, Springer, v. 122, p. 1171–1188, 2020. Citado na página 19.

SONG, X. et al. Fast wordpiece tokenization. *arXiv preprint arXiv:2012.15524*, 2020. Citado na página 32.

STALLINGS, W. M.; GILLMORE, G. M. A note on “accuracy” and “precision”. *Journal of Educational Measurement*, Wiley Online Library, v. 8, n. 2, p. 127–129, 1971. Citado 2 vezes nas páginas 28 e 29.

STRINGHINI, G.; KRUEGEL, C.; VIGNA, G. Detecting spammers on social networks. In: *Proceedings of the 26th annual computer security applications conference*. [S.l.: s.n.], 2010. p. 1–9. Citado na página 19.

SURDAK, C. *A Revolução Digital: Os 12 segredos para prosperar na era da tecnologia. O que é o pensamento Jerk e como ele está moldando o futuro*. [S.l.]: DVS Editora, 2018. Citado na página 14.

TRIVEDI, H.; BROADHURST, R. Malware in spam email: Risks and trends in the australian spam intelligence database. *Trends and Issues in Crime and Criminal Justice [electronic resource]*, n. 603, p. 1–18, 2020. Citado na página 19.

VUJOVIĆ, Ž. et al. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, v. 12, n. 6, p. 599–606, 2021. Citado na página 27.

XUE, H. et al. Content-aware trust propagation toward online review spam detection. *Journal of Data and Information Quality (JDIQ)*, ACM New York, NY, USA, v. 11, n. 3, p. 1–31, 2019. Citado na página 19.

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. Citado 3 vezes nas páginas 16, 29 e 30.