



DISSERTAÇÃO DE MESTRADO PROFISSIONAL

**Detecção de Prompt Injection
em modelos de Linguagem**

Júlia Jamile Oliveira Gonçalves

Programa de Pós-Graduação Profissional em Engenharia Elétrica

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

**Detecção de Prompt Injection
em modelos de Linguagem**

Júlia Jamile Oliveira Gonçalves

*Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia
Elétrica como requisito parcial para obtenção
do grau de Mestre em Engenharia Elétrica*

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Ph.D, FT/UnB _____
Orientador

Prof. Fábio Lúcio Lopes de Mendonça, Ph.D, _____
FT/UnB
Examinador Interno

Dr. Honório Assis Filho Crispim _____
Examinador Enterno

Daniel Alves da Silva _____
Suplente

FICHA CATALOGRÁFICA

GONÇALVES, JÚLIA JAMILE OLIVEIRA

Detecção de Prompt Injection em modelos de Linguagem [Distrito Federal] 2026.

xvi, 54 p., Publicação n^a PPEE.MP.109, 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2026).

Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1. Prompt Injection

2. Inteligência Artificial

3. Modelos de Linguagem

4. Cibersegurança

I. ENE/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

GONÇALVES, J.J.O. (2026). *Detecção de Prompt Injection em modelos de Linguagem*, Publicação: PPEE.MP.109, Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 54 p.

CESSÃO DE DIREITOS

AUTOR: Júlia Jamile Oliveira Gonçalves

TÍTULO: Detecção de Prompt Injection em modelos de Linguagem.

GRAU: Mestre em Engenharia Elétrica ANO: 2026

É concedida à Universidade de Brasília permissão para reproduzir cópias desta Dissertação de Mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Do mesmo modo, a Universidade de Brasília tem permissão para divulgar este documento em biblioteca virtual, em formato que permita o acesso via redes de comunicação e a reprodução de cópias, desde que protegida a integridade do conteúdo dessas cópias e proibido o acesso a partes isoladas desse conteúdo. O autor reserva outros direitos de publicação e nenhuma parte deste documento pode ser reproduzida sem a autorização por escrito do autor.

Júlia Jamile Oliveira Gonçalves

Depto. de Engenharia Elétrica (ENE) - FT

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

CEP 70919-970 - Brasília - DF - Brasil

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus pela força, pela sabedoria e pelo amparo concedidos ao longo de toda esta caminhada. Sem Sua presença constante, nada do que foi realizado até aqui teria sido possível.

À minha mãe, Núbia Regina, aos meus avós, José Joaquim e Neli, e aos meus padrinhos, Nília Raquel e José Geraldo, deixo minha mais profunda gratidão. Vocês foram a base de tudo, ensinaram-me valores essenciais, acreditaram em mim mesmo nos momentos mais difíceis e fizeram todo o possível e o impossível para que eu tivesse acesso à melhor educação. Cada conquista minha carrega um pouco de vocês.

Aos meus primos, Caio e Gustavo, agradeço por estarem sempre ao meu lado, compartilhando alegrias, desafios e aprendizados ao longo da vida. A toda a minha família, que me acolheu com carinho, cuidado e amor incondicional, meu sincero agradecimento por me ensinarem o verdadeiro significado de empatia, respeito e cuidado com o próximo.

Gostaria de agradecer especialmente ao meu orientador, Prof. Dr. Georges Daniel Amvame Nze, por aceitar participar deste projeto, pela orientação, paciência, disponibilidade e contribuições valiosas ao longo do desenvolvimento desta pesquisa. Sua experiência e apoio foram fundamentais para a consolidação deste trabalho.

Agradeço também ao meu parceiro de trabalho e amigo, Thiago Lotufo, que aceitou o desafio de colaborar comigo, auxiliando não apenas na publicação do artigo científico, mas também contribuindo ativamente para o amadurecimento da pesquisa e das discussões apresentadas nesta dissertação.

A todos os amigos e amigas que fiz ao longo da vida, e àqueles que caminharam comigo durante esta etapa acadêmica, agradeço pelo companheirismo, incentivo e apoio nos momentos de cansaço e celebração. Cada conversa, troca de experiência e palavra de incentivo fez diferença nessa trajetória.

À empresa Globo, agradeço pela confiança, pela oportunidade profissional e pelo ambiente de aprendizado proporcionado, que contribuiu de forma significativa para o meu crescimento técnico e pessoal, oferecendo experiências e lições que levarei para toda a vida.

Por fim, agradeço ao Departamento de Engenharia Elétrica da Universidade de Brasília (UnB) por me acolher, apoiar e permitir a continuidade dos meus estudos no mestrado, contribuindo para minha formação acadêmica e científica.

A todos que, de alguma forma, fizeram parte desta trajetória, o meu mais sincero agradecimento.

Júlia Jamile Oliveira Gonçalves

RESUMO

Os Modelos de Linguagem de Grande Escala (LLMs) são amplamente utilizados na indústria e na academia para tarefas diversas, como assistentes virtuais e automação de processos. No entanto, essas tecnologias apresentam vulnerabilidades de segurança, como ataques de Prompt Injection, que podem comprometer a integridade e confiabilidade dos modelos. Este estudo propõe uma abordagem baseada em aprendizado de máquina para detectar ataques de *Prompt Injection* e comparar sua eficácia com modelos tradicionais.

Foi construído um conjunto de dados bilíngue contendo *prompts* legítimos e maliciosos, incluindo diferentes variações de ataques de *Prompt Injection*. O dataset foi avaliado sob três cenários distintos: distribuição original desbalanceada, *undersampling* e *oversampling*. Diversos pipelines de aprendizado de máquina foram treinados, combinando técnicas de vetorização textual (*CountVectorizer* e *TF-IDF*) com classificadores tradicionais, além do treinamento de um modelo baseado em *BERT*. Complementarmente, modelos de LLMs foram utilizados para análise comparativa de comportamento frente aos *prompts* de entrada.

Os resultados experimentais demonstraram que estratégias de *oversampling* produzem desempenho superior de forma consistente, com redução significativa de falsos negativos. Pipelines baseados em *TF-IDF* combinados com classificadores lineares, bem como o modelo *BERT*, alcançaram valores de acurácia e *F1-score* superiores a 97%, evidenciando alta eficácia na detecção de ataques. A análise das matrizes de confusão reforçou a importância do balanceamento de classes em cenários de segurança, nos quais a não detecção de um ataque pode ter impactos críticos.

Os resultados indicam que é possível implementar mecanismos eficazes de detecção de *Prompt Injection* mesmo em ambientes com recursos computacionais limitados, contribuindo para o aumento da segurança e confiabilidade de aplicações baseadas em LLMs.

Palavras-chave: Inteligência Artificial, Modelos de Linguagem, Prompt Injection, Aprendizado de Máquina, Cibersegurança

ABSTRACT

Large Language Models (LLMs) are widely utilized in industry and academia for diverse tasks, such as virtual assistants and process automation. However, these technologies exhibit security vulnerabilities, including Prompt Injection attacks, which can compromise model integrity and reliability. This study proposes a machine learning-based approach to detect Prompt Injection attacks and compare its effectiveness against traditional models.

A bilingual dataset containing legitimate and malicious prompts, encompassing various Prompt Injection attack variations, was constructed. The dataset was evaluated under three distinct scenarios: original

imbalanced distribution, undersampling, and oversampling. Multiple machine learning pipelines were trained, combining text vectorization techniques (CountVectorizer and TF-IDF) with traditional classifiers, alongside BERT-based model training. Additionally, LLMs were employed for comparative behavioral analysis against input prompts.

Experimental results demonstrated that oversampling strategies consistently produce superior performance, with significant false negative reduction. TF-IDF-based pipelines combined with linear classifiers, as well as the BERT model, achieved accuracy and F1-score values exceeding 97%, evidencing high effectiveness in attack detection. Confusion matrix analysis reinforced the importance of class balancing in security scenarios, where attack non-detection can have critical impacts.

The results indicate that effective Prompt Injection detection mechanisms can be implemented even in computationally limited environments, contributing to enhanced security and reliability of LLM-based applications.

Keywords: Artificial Intelligence, Language Models, Prompt Injection, Machine Learning, Cybersecurity

SUMÁRIO

1	INTRODUÇÃO	1
1.1	MOTIVAÇÃO	2
1.2	OBJETIVOS DE TRABALHO	2
1.2.1	OBJETIVO GERAL	2
1.2.2	OBJETIVO ESPECÍFICO	3
1.3	TRABALHO PUBLICADO	3
1.4	DISPONIBILIZAÇÃO DO CÓDIGO-FONTE	3
1.5	ORGANIZAÇÃO DO TRABALHO	3
2	FUNDAMENTAÇÃO TEÓRICA	5
2.1	SEGURANÇA DA INFORMAÇÃO	5
2.1.1	OWASP TOP 10 PARA APLICAÇÕES COM LLMs	5
2.1.2	PROMPT INJECTION	7
2.2	INTELIGÊNCIA ARTIFICIAL (IA)	8
2.2.1	MODELO DE LINGUAGEM DE GRANDE ESCALA (LLM)	9
2.3	APRENDIZADO DE MÁQUINA (MACHINE LEARNING)	10
2.3.1	TÉCNICAS DE BALANCEAMENTO DE DADOS: <i>Oversampling</i> E <i>Undersampling</i>	11
2.3.2	APRENDIZADO DE MÁQUINA (MACHINE LEARNING) PARA SEGURANÇA DA INFORMAÇÃO	13
2.3.3	APRENDIZADO DE MÁQUINA (MACHINE LEARNING) PARA LINGUAGEM NATURAL	14
2.3.4	MODELOS DE MACHINE LEARNING	15
2.3.4.1	NAIVE BAYES	15
2.3.4.2	SUPPORT VECTOR MACHINES (SVM)	16
2.3.4.3	RANDOM FORESTS	17
2.3.4.4	STOCHASTIC GRADIENT DESCENT (SGD)	18
2.4	APRENDIZADO PROFUNDO (DEEP LEARNING)	18
2.5	VECTORIZAÇÃO	20
3	TRABALHOS RELACIONADOS	22
4	ARQUITETURA PROPOSTA	26
4.1	COLETA DE DADOS	27
4.1.1	CONSTRUÇÃO DO DATASET EM PORTUGUÊS	27
4.1.2	INTEGRAÇÃO DE <i>datasets</i> PÚBLICOS EM INGLÊS	27
4.1.3	CRITÉRIOS DE ROTULAGEM E CONTROLE DE QUALIDADE	27
4.1.4	DISPONIBILIZAÇÃO E REPRODUTIBILIDADE	28
4.2	BALANCEAMENTO E PROCESSAMENTO	28
4.2.1	BALANCEAMENTO DE CLASSES	28

4.2.2	SUBCONJUNTO PARA AVALIAÇÃO COM LLMs	29
4.2.3	PROTOCOLO DE AVALIAÇÃO INDEPENDENTE	29
4.3	MODELAGEM E TESTE DE MODELOS	30
4.3.1	MODELO BERT	30
4.3.2	MODELOS TRADICIONAIS DE Machine Learning	30
4.4	ESTRATÉGIA DE COMBINAÇÃO DE MODELOS	31
4.5	ANÁLISE E VALIDAÇÃO DOS RESULTADOS	32
4.6	INTERFACE DE DECISÃO E INTERAÇÃO COM O USUÁRIO	32
5	TESTE E RESULTADOS	33
5.1	DISTRIBUIÇÃO DAS CLASSES.....	33
5.2	DESEMPENHO DOS MODELOS TRADICIONAIS DE Machine Learning.....	34
5.2.1	CURVAS DE APRENDIZADO	35
5.2.2	ANÁLISE DA MATRIZ DE CONFUSÃO	37
5.3	DESEMPENHO DO MODELO BERT	38
5.4	Prompts TREINADOS	40
5.5	ANÁLISE DOS RESULTADOS	41
6	CONCLUSÃO	43
6.1	TRABALHOS FUTUROS	44
	REFERÊNCIAS BIBLIOGRÁFICAS	45
	APÊNDICES	50
I.1	RESULTADOS EXPERIMENTAIS DETALHADOS	50
I.1.1	CURVAS DE APRENDIZADO DETALHADAS - MACHINE LEARN	50
I.1.2	MATRIZES DE CONFUSÃO POR ESTRATÉGIA - MACHINE LEARN	52
I.1.3	ESTABILIDADE DAS MÉTRICAS POR FOLD - BERT	53

LISTA DE FIGURAS

2.1	Representação dos métodos de undersampling e oversampling. Fonte: (1).....	12
2.2	Principais arquiteturas de redes neurais utilizadas em aprendizado profundo: ANN, CNN, RNN e Transformer, cada uma com características e aplicações distintas. Fonte: (2).....	19
4.1	Arquitetura do sistema de detecção de prompt injection.....	26
5.1	Distribuição das classes no dataset desbalanceado para detecção de ataques de Prompt Injection (total de 1545 amostras).	34
5.2	Curvas de aprendizado para diferentes estratégias de balanceamento	36
5.3	Matriz de confusão média para diferentes estratégias de balanceamento	37
5.4	Comparação BERT: Oversampling (0.96 F1) > Desbalanceado (0.93) > Undersampling (0.88). Robustez inerente reduz gap.	38
5.5	Matriz de confusão do modelo BERT com oversampling, evidenciando baixo número de falsos negativos (FN = 5), o que indica elevado recall para a classe maliciosa.....	39
5.6	Comparação das métricas de acurácia, precisão e F1-score para os modelos avaliados sob diferentes estratégias de balanceamento de classes.....	41
1	Curvas individuais sob oversampling (melhor estratégia).	50
2	Curvas individuais sob banco desbalanceado.....	51
3	Curvas individuais sob undersampling.	51
4	Matrizes de confusão - Oversampling (F1 médio=0.961).	52
5	Matrizes de confusão - Undersampling (F1 médio=0.907).....	52
6	Matrizes de confusão - Desbalanceado (F1 médio=0.910).....	53
7	Matriz de confusão do modelo BERT treinado com dataset desbalanceado (F1-score médio = 0.917).	53
8	Matriz de confusão do modelo BERT treinado com banco de dados balanceado por undersampling (F1-score médio = 0.88).....	54
9	Matriz de confusão do modelo BERT treinado com banco de dados balanceado por oversampling (F1-score médio = 0.96).....	54

LISTA DE TABELAS

3.1	Resumo comparativo dos principais trabalhos relacionados	24
4.1	Pipelines de Classificação Tradicional Implementados	30
5.1	Distribuição das classes nos diferentes datasets utilizados	33
5.2	Desempenho de modelos tradicionais de ML com diferentes estratégias de balanceamento ..	34
5.3	Desempenho do modelo BERT sob diferentes estratégias de balanceamento	38
5.4	Resultados de modelos de IA de mercado em tarefa específica	40

1 INTRODUÇÃO

A inteligência artificial (IA) é um dos pilares fundamentais da transformação digital contemporânea, promovendo inovações em setores diversos como saúde, educação, segurança, defesa cibernética, indústria, política e análise de dados sensíveis (3). No entanto, como toda tecnologia disruptiva, a IA tem provocado intensos debates éticos e sociais, especialmente em relação ao seu impacto no mercado de trabalho, à privacidade dos dados, à transparência algorítmica e à governança tecnológica. À medida que sistemas inteligentes se tornam mais presentes em diversos setores, surgem preocupações legítimas sobre como equilibrar inovação com responsabilidade.

Nos últimos anos, observou-se uma ascensão significativa dos Modelos de Linguagem de Grande Escala (LLMs – Large Language Models), como GPT, PaLM, LLaMA e outros, que demonstram uma capacidade sem precedentes de compreender, interpretar e gerar linguagem natural de forma coerente e contextualizada (4). Esses modelos vêm sendo amplamente aplicados em chatbots, assistentes virtuais, sistemas de recomendação, análise de sentimentos e em processos de automação inteligente, contribuindo para o avanço da produtividade e da tomada de decisão baseada em dados.

Diante do avanço acelerado da inteligência artificial, especialmente com a popularização dos Modelos de Linguagem de Grande Escala, como os utilizados em assistentes virtuais, sistemas de recomendação e ferramentas de produtividade, empresas, instituições globais e usuários comuns têm adotado essas tecnologias em larga escala. No entanto, essa adoção também traz à tona novos vetores de ataque, que desafiaram a segurança, a privacidade e a integridade dos sistemas baseados em IA. Dentre esses vetores, destaca-se o Prompt Injection, um tipo de ataque que explora a manipulação de instruções inseridas em prompts (entradas textuais), com o objetivo de induzir o modelo a ignorar suas restrições, vazarem informações confidenciais ou executar ações não intencionais.

Esses ataques são relativamente novos e estão se tornando uma ameaça cada vez maior em sistemas que dependem bastante de modelos de linguagem grandes, como assistentes corporativos, agentes autônomos e ferramentas de análise de dados sensíveis. Portanto, entender e combater o Prompt Injection não deve ser visto como algo que se faz uma única vez, mas como um problema grande e constante que vem junto com o uso de modelos de linguagem. Trata-se de uma classe de vulnerabilidade que surge da maneira como esses modelos entendem ordens e contexto, principalmente quando trabalham com entradas externas e ferramentas integradas.

Assim, diminuir os riscos de injeção de prompt precisa de uma estratégia constante e organizada, que inclui arquitetura segura, verificação das entradas, gerenciamento de permissões, supervisão e testes repetidos. Esse tipo de ataque precisa de análises detalhadas e planos de defesa em várias etapas, que são integrados ao processo de desenvolvimento e funcionamento dos sistemas que usam modelos de linguagem.

Este estudo quer ajudar nessa área, dando sugestões de métodos de aprendizado de máquina para reconhecer esses ataques, mostrando como as formas antigas e as novas de abordar o problema funcionam na detecção. Além disso, o estudo analisa métodos de filtro de entrada que utilizam regras e aprendizado de máquina, que conseguem diminuir bastante a possibilidade de sucesso dos ataques.

Portanto, o objetivo desse estudo é analisar quais são as falhas que os modelos de linguagem grandes podem ter quando são atacados com vídeos de injeção de prompt, e propôr boas formas de identificar e diminuir esses riscos. Para isso, usamos indicadores como precisão, recall e F1-score, com o objetivo de tornar os sistemas que usam inteligência generativa mais seguros.

1.1 MOTIVAÇÃO

A rápida adoção dos LLMs por empresas, instituições governamentais e usuários em geral evidencia o impacto transformador dessa tecnologia na forma como interagem com sistemas computacionais, tomam decisões e automatizam processos. No entanto, essa evolução também expõe vulnerabilidades significativas, especialmente no que diz respeito à segurança da informação. Entretanto, a falta de mecanismos de segurança maduros para proteger esses sistemas contra ataques de manipulação textual torna-se um risco iminente. O Prompt Injection representa uma ameaça atual e ainda pouco explorada na literatura acadêmica e nas práticas de segurança industrial, o que reforça a necessidade de investigação aprofundada.

Em ambientes corporativos, um ataque desse tipo pode induzir o modelo a revelar dados confidenciais, alterar respostas automatizadas ou executar comandos que comprometam a integridade da informação. Em cenários militares, os impactos podem ser ainda mais críticos, envolvendo o vazamento de informações estratégicas ou a indução de comportamentos adversos em sistemas autônomos de decisão. No uso cotidiano, há riscos concretos de exposição de dados bancários, informações pessoais e outros conteúdos privados, especialmente em aplicações que integram LLMs a serviços digitais.

Além dos prejuízos operacionais e estratégicos, esse tipo de incidente pode gerar violações à Lei Geral de Proteção de Dados Pessoais (LGPD), uma vez que o tratamento inadequado ou o vazamento de dados pessoais pode resultar em sanções legais, danos reputacionais e responsabilização das organizações, reforçando a necessidade de mecanismos robustos de detecção e prevenção desses ataques.

Diante desse cenário, a motivação deste trabalho reside na necessidade urgente de desenvolver mecanismos inteligentes e eficazes para a detecção e mitigação de ataques de Prompt Injection, bem como na avaliação comparativa de soluções já disponíveis no mercado. O objetivo é promover maior confiabilidade, transparência e segurança no uso de LLMs, contribuindo para o avanço científico e tecnológico da área de segurança em inteligência artificial.

1.2 OBJETIVOS DE TRABALHO

1.2.1 Objetivo Geral

Analisar as vulnerabilidades dos LLMs frente a ataques de Prompt Injection e propor estratégias baseadas em aprendizado de máquina e interação de entrada que permitam detectar e mitigar tais ataques de forma eficaz, avaliando o desempenho das abordagens propostas por meio de métricas quantitativas.

1.2.2 Objetivo Específico

- Realizar uma revisão sistemática sobre ataques de Prompt Injection, destacando suas categorias, técnicas e impactos observados na literatura recente;
- Desenvolver um conjunto de dados (dataset) contendo exemplos de prompts legítimos e maliciosos, com anotações que permitam treinar e testar modelos de classificação;
- Implementar e comparar diferentes algoritmos de aprendizado de máquina supervisionados para a detecção automática de Prompt Injection;
- Investigar estratégias de filtragem e pré-processamento textual baseadas em regras linguísticas e aprendizado automático, para mitigar a ocorrência de ataques antes da interação com o modelo de linguagem;
- Avaliar o desempenho das abordagens propostas utilizando métricas como acurácia, precisão, recall e F1-score;

1.3 TRABALHO PUBLICADO

Durante o estudo para a construção desta dissertação, o artigo (5) foi publicado com o título "Detecção de Prompt Injection em modelos de Linguagem".

1.4 DISPONIBILIZAÇÃO DO CÓDIGO-FONTE

O código-fonte correspondente à arquitetura proposta neste trabalho, bem como aos experimentos realizados ao longo da pesquisa, foi disponibilizado publicamente em um repositório na plataforma GitHub (6). O objetivo dessa disponibilização é garantir transparência, reprodutibilidade dos resultados e facilitar a extensão da abordagem por outros pesquisadores e profissionais da área.

O repositório contém a implementação dos pipelines de pré-processamento, balanceamento de classes, treinamento e avaliação dos modelos, além das configurações utilizadas nos experimentos descritos nesta dissertação.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho é dividido em seis capítulos, mais as referências bibliográficas e os apêndices, e está organizado para levar o leitor do básico até a análise dos resultados e as conclusões finais.

O Capítulo 1 é a introdução do trabalho, onde é mostrado o problema que será estudado, por que é importante fazer esse estudo, os objetivos principais e secundários, além de mencionar pesquisas já feitas e publicadas que estão ligadas a esse tema. No final do capítulo, é mostrada a estrutura geral do documento.

O segundo capítulo apresenta a base teórica que é importante para entender bem o estudo. São abordados conceitos importantes de Segurança da Informação, com destaque para ataques de Prompt Injection, além de fundamentos de Inteligência Artificial, LLMs, Aprendizado de Máquina e Aprendizado Profundo. O capítulo também aborda métodos para equilibrar os dados, formas de transformar texto em vetores e os principais modelos de aprendizado de máquina utilizados no decorrer do estudo.

O capítulo 3 mostra os trabalhos já feitos, estudando métodos que já existem na literatura sobre como detectar ataques em modelos de linguagem e como usar a máquina de aprendizagem na segurança da informação. Essa análise ajuda a entender onde se encaixa a proposta desse trabalho em relação ao que já foi desenvolvido antes.

O Capítulo 4 explica com clareza a arquitetura proposta que foi utilizada. São mostrados os métodos de coleta e criação dos conjuntos de dados em português e em inglês, os critérios usados para marcar as informações e garantir a qualidade, além das estratégias utilizadas para equilibrar as classes. O capítulo também explica como são feitas a modelagem, o treinamento e a avaliação dos modelos tradicionais de aprendizado de máquina e do modelo BERT, além de descrever a estratégia para combinar os modelos, os passos de validação e a interface de decisão que é usada para interagir com o usuário.

O Capítulo 5 mostra e explica os resultados obtidos nos experimentos. No início, é abordada a forma como as classes estão distribuídas nos diversos cenários de balanceamento. Depois, mostram-se os resultados dos modelos tradicionais e do modelo BERT, com gráficos de aprendizado e estudo das matrizes de confusão. O capítulo termina com uma análise completa dos resultados, mostrando os principais pontos encontrados e comparando as diferentes formas de abordagem avaliada.

No final, o Capítulo 6 mostra a conclusão do trabalho, resumindo os resultados encontrados, as contribuições da pesquisa e suas limitações. Também são mencionadas opções de trabalhos futuros que podem melhorar e ampliar a abordagem sugerida.

Os apêndices complementam o texto principal, apresentando resultados experimentais detalhados, matrizes de confusão para cada estratégia e análises adicionais sobre a estabilidade das métricas, garantindo assim maior clareza e reprodutibilidade da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SEGURANÇA DA INFORMAÇÃO

A segurança da informação é uma área essencial da computação e da engenharia, que tem como objetivo proteger as informações de uma empresa contra acessos que não são permitidos, modificações que não devem acontecer e interrupções no funcionamento dos serviços (7). Ela quer assegurar que as informações conservem suas qualidades importantes: sigilo, autenticidade, integridade, disponibilidade e não repúdio (7, 8).

Com a digitalização cada vez maior dos processos e o uso de novas tecnologias, como computação em nuvem, Internet das Coisas (IoT) e inteligência artificial, os riscos de segurança da informação estão crescendo bastante. O aumento no volume de dados sensíveis e no número de usuários conectados expande as possibilidades de ataques, exigindo sistemas de defesa e monitoramento ainda mais fortes (9).

Nesse contexto, a segurança da informação evolui de um enfoque puramente técnico, centrado em dispositivos e redes, para uma abordagem mais ampla e integrada, que abrange políticas, governança, comportamento humano e aprendizado de máquina. As ameaças atuais não são mais só malwares, vírus ou ataques diretos, mas também envolvem a manipulação de dados, engenharia social e exploração de vulnerabilidades em sistemas de inteligência artificial (9, 8).

Com o desenvolvimento dos modelos de linguagem de grande porte, surgiram novos desafios para garantir a integridade desses sistemas. Os LLMs, como o ChatGPT, Gemini e Claude, são ensinados com muitos dados de texto e conseguem entender e criar linguagem natural de forma bastante avançada (10). No entanto, justamente por atuarem em ambientes abertos e interativos, cam expostos a novas formas de ameaça, como o Prompt Injection.

2.1.1 OWASP Top 10 para aplicações com LLMs

Com o avanço acelerado das aplicações baseadas em Modelos de Linguagem de Grande Porte (LLMs), tornou-se necessário estabelecer diretrizes específicas para identificar e mitigar riscos associados a essa nova geração de sistemas inteligentes. Nesse contexto, a OWASP (Open Web Application Security Project) desenvolveu o OWASP Top 10 for Large Language Model Applications (11), um documento que reúne as principais vulnerabilidades observadas em aplicações que utilizam LLMs. Essa iniciativa adapta conceitos tradicionais de segurança de aplicações para os desafios particulares impostos pela inteligência artificial generativa.

Na versão mais recente do documento, os riscos são organizados em dez grandes grupos, que retem tanto ataques clássicos adaptados ao contexto de IA quanto vulnerabilidades emergentes associadas ao uso de modelos de linguagem em produção (11, 12). Entre os itens, destacam-se:

- LLM01 – Prompt Injection: manipulação direta ou indireta dos prompts para desviar o compor-

tamento do modelo, induzindo respostas maliciosas, vazamento de dados ou execução indevida de ações externas;

- LLM02 – Sensitive Information Disclosure: exposição de informações sensíveis presentes em dados de treino, contexto de RAG, registros de conversa ou sistemas integrados, muitas vezes por meio de perguntas cuidadosamente elaboradas;
- LLM03 – Supply Chain Vulnerabilities: dependência de modelos, bibliotecas, plugins, serviços externos e conjuntos de dados de terceiros, que podem estar comprometidos e introduzir vulnerabilidades na aplicação;
- LLM04 – Data and Model Poisoning: inserção de dados maliciosos em fases de treinamento ou re-tuning, com o objetivo de alterar o comportamento do modelo, criar backdoors ou enviesar respostas;
- LLM05 – Improper Output Handling: uso inseguro das respostas do modelo, por exemplo, executando comandos gerados sem validação, inserindo saídas em HTML ou SQL sem sanitização ou conando cegamente em conteúdo gerado;
- LLM06 – Excessive Agency: concessão de permissões amplas ao LLM (como acesso irrestrito a APIs, arquivos ou sistemas críticos) sem controles de autorização, auditoria ou limites de escopo;
- LLM07 – System Prompt Leakage: exposição do system prompt ou de instruções internas, que podem revelar lógica de negócio, regras de segurança ou chaves de configuração, facilitando ataques de engenharia de prompt;
- LLM08 – Vector and Embedding Weaknesses: exploração de falhas em pipelines de vetores e embeddings, como inserção de documentos maliciosos em bases vetoriais, data leakage via contexto recuperado ou manipulação semântica de consultas;
- LLM09 – Misinformation: geração de respostas incorretas, enviesadas ou enganosas, incluindo alucinações, desinformação e conteúdo potencialmente danoso quando o sistema é usado como fonte de verdade;
- LLM10 – Unbounded Consumption: uso abusivo de recursos por meio de consultas excessivamente complexas, ood de requisições ou exploração de limites de contexto, causando degradação de desempenho, interrupções de serviço ou custos inesperados.

Além disso, categorias como Data and Model Poisoning, Vector and Embedding Weaknesses e Improper Output Handling dialogam com cenários em que o modelo interage com fontes externas de dados ou executa ações automatizadas, ampliando a superfície de ataque e tornando essencial a adoção de controles de segurança em múltiplas camadas. (11, 13)

Diferentemente das aplicações convencionais, sistemas baseados em LLMs apresentam desvantagens adicionais devido à ausência de separação clara entre instruções e dados, bem como à natureza probabilística das respostas geradas. Isso torna mais difícil prever comportamentos inesperados ou identificar quando o modelo está sendo influenciado por entradas maliciosas. Assim, o OWASP Top 10 para LLMs não

apenas lista vulnerabilidades, mas também orienta desenvolvedores e organizações sobre boas práticas de governança, monitoramento contínuo, validação de entradas e limitação de privilégios(13).

A adoção dessas recomendações é fundamental para reduzir riscos em ambientes corporativos, governamentais e acadêmicos que utilizam inteligência artificial. Ao integrar princípios tradicionais de segurança da informação com técnicas específicas para LLMs, é possível construir aplicações mais robustas, confiáveis e resilientes diante das ameaças emergentes (11, 13).

2.1.2 Prompt injection

A segurança de sistemas que usam Modelos de Linguagem Natural (NLP) está se tornando uma das preocupações mais importantes para cientistas e profissionais da tecnologia. Grupos de pesquisa e empresas estão juntos para descobrir novas fraquezas e, sobretudo, criar formas de tornar os aplicativos de inteligência artificial mais seguros.

Entre as falhas mais comuns, segundo a OWASP Top 10 para modelos de linguagem de grande escala (11), está o Prompt Injection, um tipo de ataque que tem ganhado mais atenção recentemente. Esse ataque usa a forma como os modelos de LLMs, como o GitHub Copilot, lidam com as ordens que recebem. Basicamente, um invasor coloca instruções ruins no prompt do modelo, fazendo com que ele faça coisas que não devia ou dê respostas que não são esperadas (14).

Ao contrário das vulnerabilidades comuns em softwares, esse ataque usa a própria forma como o modelo entende e interpreta a linguagem, prejudicando a maneira como ele dá prioridade e executa comandos. Em um ataque de injeção de prompt, o adversário tenta fazer o modelo ignorar as regras de segurança, mostrar dados sensíveis, entrar em sistemas externos ou alterar as respostas para enganar o usuário. Um exemplo conhecido é o chamado "jailbreak", no qual o atacante inclui instruções que levam o modelo a ignorar as limitações de acesso impostas pelos seus criadores, como filtros de conteúdo ou regras de utilização (14).

O fenômeno de Prompt Injection pode ser dividido, segundo a literatura mais recente (15), em duas categorias principais. A primeira, chamada de Injeção Direta de Prompt, envolve ataques em que um agente malicioso coloca instruções ou comandos diretamente no prompt, com a intenção de fazer o modelo agir de forma que não seria esperada em situações normais. A segunda, também chamada de Injeção de Prompt Indireta, é quando conteúdo malicioso é inserido em fontes externas, como sites ou documentos, e depois é interpretado pelo modelo, criando um tipo de ataque menos evidente, mas ainda muito perigoso. Nesse segundo caso, o modelo pode ser convencido a seguir ordens colocadas em dados que parecem inofensivos, colocando em risco sua própria integridade e a segurança do ambiente onde está trabalhando.

Essa ameaça representa um desafio significativo, pois os LLMs não possuem uma separação clara entre código e entrada textual, o que dificulta a identificação de comandos legítimos frente a tentativas de exploração (16). As consequências de um ataque de injeção de prompt podem ir desde erros pequenos nas respostas até vazamentos de informações sensíveis ou até mesmo a manipulação de sistemas importantes. Em ambientes corporativos ou militares, esses ataques são um grande risco, pois podem levar a acessos sem permissão, alteração de relatórios ou divulgação de informações falsas.

Vários estudos recentes estão tentando identificar e entender as diversas formas de Prompt Injection

(17, 18). Entre elas destacam-se:

- Direta: ocorre quando o atacante insere comandos explícitos na entrada do usuário para modificar o comportamento do modelo;
- Indireta: acontece quando o modelo é influenciado por conteúdo externo que está contaminado, como um documento PDF, uma página da web ou um e-mail com texto prejudicial;
- Data Poisoning: envolve a inserção de dados alterados durante o treinamento do modelo, para modificar a maneira como ele pensa internamente;
- Transfer Learning Injection: usa falhas durante o ajuste fino (fine-tuning) de modelos para introduzir comportamentos que não são desejados.

A detecção e mitigação de ataques de Prompt Injection continuam sendo assuntos em desenvolvimento na pesquisa acadêmica e nas práticas do setor industrial. Algumas abordagens recentes estão usando inteligência artificial para encontrar padrões de linguagem e significado relacionados a instruções perigosas, o que ajuda a criar sistemas que conseguem diferenciar entre prompts legítimos e manipulados (17, 18).

Para combater esses ataques, é necessário usar uma abordagem que envolva várias áreas, misturando métodos convencionais de segurança com técnicas de inteligência artificial e análise de linguagem. As estratégias envolvem a limpeza de entrada (input sanitization), a verificação constante do comportamento, o uso de detectores baseados em aprendizado supervisionado e a criação de regras que limitam o acesso a informações sensíveis (17).

Essa combinação aumenta a precisão da detecção e reduz a incidência de falsos negativos, proporcionando uma defesa mais robusta contra instruções maliciosas inseridas nos prompts.

2.2 INTELIGÊNCIA ARTIFICIAL (IA)

A Inteligência Artificial (IA) é um campo multidisciplinar da ciência da computação voltado ao desenvolvimento de sistemas capazes de executar tarefas que tradicionalmente requerem inteligência humana. Entre essas tarefas estão o pensamento lógico, resolver problemas, reconhecer padrões, aprender e entender a linguagem natural (19).

Desde os primeiros trabalhos de Alan Turing, John McCarthy e Marvin Minsky, a IA evoluiu de programas que usavam regras e símbolos para sistemas mais complicados que aprendem sozinhos (20). Essa evolução foi ajudada por três coisas principais: o aumento da capacidade de processamento, o progresso nas arquiteturas de redes neurais e a facilidade de acesso a grandes quantidades de dados (Big Data).

Hoje em dia, a inteligência artificial está presente em quase todos os campos, mudando a maneira como a sociedade usa a tecnologia desde diagnósticos médicos feitos com ajuda de computadores até sistemas de segurança contra hackers que usam análise preditiva.

Nesse grande campo, há três áreas importantes: o Aprendizado de Máquina, o Aprendizado Profundo e os Modelos de Linguagem de Grande Escala. Esses modelos, como os LLMs, mostram um grande pro-

gresso na habilidade das máquinas de entender e criar linguagem natural, cando cada vez mais próximos do que antes era visto como algo só humano.

Nesse contexto, Larry Page, cofundador da Google Inc. O CEO da Alphabet, sintetiza a ambição por trás do desenvolvimento da IA ao afirmar:

“A Inteligência Artificial seria a versão natural do Google. A máquina de busca definitiva, que compreenderia tudo na web. Entenderia exatamente o que você deseja e lhe daria a coisa certa. Não estamos nem perto de fazer isso agora. No entanto, podemos chegar cada vez mais perto disso; sendo basicamente no que estamos trabalhando.” Larry Page (21)

Essa visão reforça o papel da IA como uma tecnologia de propósito geral, com potencial para redefinir a forma como acessamos, interpretamos e interagimos com o conhecimento.

2.2.1 Modelo de Linguagem de Grande Escala (LLM)

Os Modelos de Linguagem de Grande Escala (Large Language Models — LLMs) são um dos avanços mais importantes da inteligência artificial atual. Esses modelos são treinados com grandes quantidades de textos, como livros, artigos, páginas da internet e bases de dados públicas, para entender melhor como as palavras se relacionam e o significado delas em grande escala (22).

A estrutura principal dos modelos de linguagem de grande escala é o Transformer, criado por Vaswani et al., que mudou o processamento de linguagem natural ao permitir que os modelos aprendam com contexto de maneira eficiente, usando mecanismos de atenção. Essa arquitetura permite que os modelos entendam relações entre palavras e frases que estão distantes, ajudando a produzir textos que fazem sentido e que se encaixam bem no contexto (23).

Modelos como GPT (OpenAI), PaLM (Google), LLaMA (Meta) e Claude (Anthropic) são exemplos de LLMs que usam arquiteturas baseadas em Transformers para criar textos lógicos, dar respostas, traduzir idiomas e até fazer cálculos e raciocínios. Sua habilidade de entender o contexto e criar respostas que parecem feitas por uma pessoa tornou essas ferramentas muito úteis em muitas áreas, como ensino, ajuda com tecnologia, análise de código e automação no trabalho (23).

A complexidade desses modelos também traz novos desafios, principalmente em relação à segurança e à confiança. Como os LLMs são muito sensíveis às instruções que recebem, eles podem ser explorados por meio de ataques chamados de Prompt Injection. Nesses ataques, um invasor modifica o texto do prompt para fazer o modelo realizar ações que não eram pretendidas, como revelar informações confidenciais, ignorar normas éticas ou seguir ordens prejudiciais (24).

Essas falhas, que também podem ser chamadas de vulnerabilidades, acontecem porque os LLMs não entendem de verdade o que está escrito ou o que se pretende com as instruções; eles apenas copiam padrões que aprenderam. Assim, quando um comando ruim é inserido de forma criativa e clara, o modelo pode entendê-lo como algo válido para a tarefa. Além disso, já que os LLMs dependem muito do contexto que recebem, até mesmo mudanças pequenas no prompt podem causar resultados diferentes do esperado.

Dessa forma, é muito importante estudar a segurança nas LLMs para criar aplicações que possam ser

con áveis. Estratégias que usam aprendizado de máquina, Itragem de sentido e análise do contexto têm demonstrado ser eficazes para detectar prompts suspeitos e diminuir bastante o risco de ataques.

Entender os conceitos básicos da inteligência artificial, do aprendizado de máquina e do aprendizado profundo é essencial para criar mecanismos sólidos de proteção contra os novos desafios de segurança que surgem nos sistemas de inteligência artificial generativa.

2.3 APRENDIZADO DE MÁQUINA (MACHINE LEARNING)

Aprendizado de Máquina (Machine Learning) é uma área da inteligência artificial que se dedica a criar algoritmos que conseguem aprender com os dados e melhorar seu desempenho com o tempo, sem precisar de instruções detalhadas para cada atividade. Assim, o aprendizado de máquina permite que sistemas computacionais reconheçam padrões, criem modelos para prever o futuro e tomem decisões sozinhos, sendo hoje uma das áreas mais importantes da inteligência artificial moderna (25, 26).

De acordo com Langley e Simon (27), o aprendizado de máquina é o estudo de métodos computacionais que ajudam os sistemas a melhorar seu desempenho com base na experiência. Seu objetivo é automatizar tarefas que antes eram feitas por pessoas, melhorando a precisão e a eficiência dos processos ao identificar padrões nos dados usados para treinar o sistema.

O ML funciona com a ideia de que algoritmos conseguem encontrar padrões nos grandes volumes de dados e, com base nesses padrões, criar modelos que conseguem fazer inferências, classificar e prever informações de novos dados que são apresentados. Para isso, ele usa métodos estatísticos, matemáticos e computacionais para converter dados brutos em informação útil. Esse processo inclui várias etapas, como a coleta de dados, preparação desses dados, ensino do modelo, verificação do desempenho e a atualização constante, formando um ciclo que se repete e melhora continuamente (28).

As técnicas de aprendizado de máquina são usadas em muitas áreas diferentes, como reconhecimento de padrões, visão computacional, radiologia, engenharia espacial, redes sociais, jogos, entretenimento, aplicações biomédicas e segurança cibernética. No dia a dia, exemplos incluem o uso pelo Google para aumentar a precisão das buscas, pela Netflix e Spotify em sistemas de recomendação e por bancos para identificar fraudes (29, 28).

Os principais paradigmas de aprendizado de máquina incluem:

- **Aprendizado Supervisionado:** usa conjuntos de dados com as respostas corretas, onde cada entrada tem uma saída que deve ser aprendida. O modelo entende como as variáveis de entrada estão conectadas às variáveis de saída, e é usado em muitos tipos de problemas, como prever valores e classificar dados (30, 31). Exemplos de algoritmos são o Support Vector Machines (SVM), Árvores de Decisão, Random Forests e Regressão Logística.
- **Aprendizado Não Supervisionado:** utiliza dados que não têm rótulos, e o objetivo é encontrar padrões ou estruturas ocultas, como grupos (clusters) e relações entre as informações. É frequentemente aplicado em atividades como análise exploratória, redução da dimensionalidade e identificação de anomalias (32).

- **Aprendizado Semi-Supervisionado:** usa tanto dados com rótulos quanto dados sem rótulos, sendo uma boa opção quando é difícil ou caro rotular todos os dados manualmente. Essa abordagem deixa o modelo aproveitar a estrutura dos dados que não têm rótulos para melhorar o quanto ele consegue aplicar o que aprendeu em novas situações (33).
- **Aprendizado por Reforço:** se baseia em um agente que aprende ao interagir com o ambiente, ganhando recompensas ou sofrendo penalidades com base nas ações que realiza. Essa técnica é muito usada em robótica, jogos e sistemas de controle, porque a exploração e a adaptação constantes são muito importantes (34).

Assim, o aprendizado de máquina é fundamental para criar sistemas inteligentes, sendo necessário no desenvolvimento de soluções que conseguem identificar padrões complexos e descobrir anomalias.

O aprendizado supervisionado, por exemplo, pode ter alguns problemas, como o *underfitting*, que acontece quando o modelo é muito simples e não consegue representar bem os dados, e o *overfitting*, que ocorre quando o modelo se adapta demais aos dados de treinamento e deixa de funcionar bem com novas informações. Estratégias como regularização, aumento de dados e validação cruzada são usadas para reduzir esses problemas (35).

Além disso, ao lidar com conjuntos de dados onde uma classe aparece muito mais que outra, é importante usar métodos de balanceamento para garantir que o modelo não seja prejudicado e aprenda de forma equilibrada.

2.3.1 Técnicas de Balanceamento de Dados: Oversampling e Undersampling

Em problemas de classificação supervisionada, frequentemente encontramos conjuntos de dados desbalanceados, onde uma ou mais classes estão muito menos representadas do que as outras. Esse desequilíbrio pode afetar o funcionamento dos modelos de aprendizado de máquina, fazendo com que eles priorizem a classe com mais exemplos e deixem de reconhecer padrões importantes da classe menos representada (36). Para resolver esse problema, usam-se várias técnicas de equilíbrio dos dados, como o *oversampling* e o *undersampling*.

O *oversampling* é um método que aumenta artificialmente o número de exemplos da classe que tem menos amostras, para que as classes sejam mais equilibradas no conjunto de dados. Isso pode ser feito copiando exemplos que já existem ou criando novos exemplos de forma artificial (37).

Uma das formas mais comuns de se lidar com desequilíbrios de classes é o SMOTE (Synthetic Minority Over-sampling Technique), que gera novas amostras sintéticas com base na interpolação entre exemplos da classe minoritária (38). Embora apresente bons resultados em conjuntos de dados com atributos numéricos contínuos, sua aplicação em bases de dados textuais impõe desafios conceituais e práticos.

Em tarefas de processamento de linguagem natural, os dados textuais são frequentemente representados em espaços vetoriais de alta dimensionalidade e elevada esparsidade, como aqueles gerados por abordagens de Bag-of-Words e TF-IDF. Nesses cenários, a interpolação linear entre vetores não possui correspondência semântica direta, podendo resultar em instâncias artificiais que não representam textos linguisticamente plausíveis ou semanticamente interpretáveis (35).

Além disso, em problemas de quantificação e classificação textual sensíveis ao contexto, pequenas alterações nos termos podem impactar significativamente o significado global da instância. A geração de amostras sintéticas nesse espaço pode introduzir ruído e comprometer a preservação de padrões discriminativos relevantes, afetando a capacidade de generalização dos modelos (36, 39).

Como alternativa, métodos de oversampling baseados na replicação de instâncias reais, como o Random Over-Sampling, são frequentemente discutidos na literatura como mais adequados para dados textuais vetorizados, uma vez que preservam integralmente a distribuição original das características e evitam a criação de representações artificiais semanticamente inconsistentes (37, 40). No entanto, o oversampling pode aumentar o risco de overfitting, principalmente quando os exemplos são copiados diretamente ou quando os dados sintéticos não mostram bem a diversidade da classe (35).

O undersampling, por sua vez, diminui a quantidade de exemplos da classe majoritária, removendo amostras para equilibrar o conjunto de dados (41). Essa maneira de trabalhar pode ajudar a fazer o treinamento mais rápido e diminuir o viés, mas também pode acabar perdendo informações importantes, principalmente se os dados mais comuns forem diversos ou tiverem padrões importantes para a tarefa de classificação.

Entretanto, a subamostragem apresenta como principal limitação a possível perda de informações relevantes, especialmente em conjuntos de dados nos quais a classe majoritária possui elevada diversidade ou contém padrões fundamentais para a tarefa de classificação. A remoção indiscriminada de instâncias pode comprometer a capacidade do modelo de capturar a variabilidade natural dos dados (36).

As técnicas de undersampling podem ser divididas em abordagens aleatórias, que selecionam subconjuntos da classe majoritária de forma estocástica, e métodos baseados em critérios mais elaborados, como o NearMiss, que preserva instâncias consideradas informativas com base na distância entre exemplos de classes distintas (36). Cada uma dessas estratégias apresenta vantagens e limitações, sendo sua adequação dependente das características do conjunto de dados e do domínio da aplicação.

A Figura 2.1 mostra como funcionam essas duas formas de abordagem.

Figura 2.1: Representação dos métodos de undersampling e oversampling. Fonte: (1)

Técnicas de balanceamento de dados são muito importantes em situações importantes, como identificar fraudes, diagnosticar doenças, garantir segurança na internet e proteger contra ataques de Prompt Injection, onde a categoria que queremos detectar aparece com menos frequência, mas é muito crucial. O uso certo dessas técnicas ajuda a criar modelos que são mais justos, precisos e confiáveis.

2.3.2 Aprendizado de Máquina (Machine Learning) para Segurança da Informação

No contexto da segurança da informação, o aprendizado de máquina é essencial para identificar ameaças novas, como tentativas de phishing, malware, irregularidades nos logs e comportamentos suspeitos nas redes (42). Esses sistemas usam algoritmos que são supervisionados e não supervisionados para identificar se um evento é verdadeiro ou prejudicial, muitas vezes com base em detalhes pequenos que são difíceis de serem reconhecidos com métodos convencionais.

No campo da segurança da informação, o aprendizado de máquina tornou-se uma ferramenta importante para identificar e lidar com ameaças na internet. Sua habilidade de reconhecer padrões complexos e situações diferentes em grandes quantidades de dados ajuda a criar sistemas que agem antes de problemas acontecerem, prevenindo ações prejudiciais e reduzindo riscos no momento em que eles surgem. Mais recentemente, também passou a ser usado para proteger modelos de linguagem contra ataques de Prompt Injection, que é um tipo de vulnerabilidade que altera o comportamento de sistemas baseados em inteligência artificial (43, 15). Nesses casos, o ML permite a criação de classificadores que conseguem diferenciar entre instruções boas e ruins com base em padrões de linguagem e significado encontrados em grandes volumes de dados.

Recentemente, o ML começou a ser usado para proteger modelos de linguagem contra ataques de Prompt Injection, uma vulnerabilidade nova que altera o comportamento de sistemas baseados em inteligência artificial. Nesses casos, o aprendizado de máquina ajuda a criar classificadores que conseguem diferenciar entre instruções boas e ruins, usando padrões de linguagem, estrutura e significado encontrados em grandes volumes de dados de entrada (44).

Além da classificação, técnicas de inteligência artificial também são usadas para:

- Detecção de anomalias: algoritmos como Isolation Forest, Autoencoders e One-Class SVM são usados para encontrar comportamentos diferentes do normal, que podem sinalizar atividades prejudiciais.
- Análise preditiva: modelos treinados com dados históricos de incidentes ajudam a prever quais podem ser os próximos ataques e a determinar o que fazer para evitar ou reduzir seus impactos.
- Resposta automatizada: sistemas inteligentes conseguem tomar decisões no momento, por exemplo, bloquear acessos não autorizados, separar máquinas que estão com problemas ou notificar as equipes de segurança.
- Correlação de eventos: técnicas de agrupamento e aprendizado profundo são aplicadas para relacionar diversos sinais provenientes de fontes diferentes, como logs, tráfego e autenticação, ajudando a identificar ataques complexos.

Portanto, o aprendizado de máquina é uma base importante para criar sistemas de segurança que são mais inteligentes, fortes e capazes de se adaptar (45). Sua capacidade de se integrar com outras tecnologias, como análise de big data, inteligência artificial generativa e automação de resposta, mostra que o futuro da segurança da informação será cada vez mais previsível e independente.

2.3.3 Aprendizado de Máquina (Machine Learning) para Linguagem Natural

Aprendizado de máquina supervisionado usado na linguagem natural é um tipo de pesquisa que tenta ensinar os computadores a entender, interpretar e lidar com a linguagem que as pessoas usam, de forma por si só. Usando algoritmos que conseguem identificar padrões linguísticos em grandes volumes de dados, como notícias, textos acadêmicos e documentos técnicos, é possível ensinar modelos para reconhecer significados, contextos e como as palavras e frases se relacionam (46).

O Aprendizado de Máquina tem sido fundamental para o progresso do Processamento de Linguagem Natural (NLP), permitindo que os sistemas entendam, interpretem e criem linguagem humana com maior precisão. Ao usar algoritmos de aprendizado de máquina em textos grandes, é possível identificar padrões linguísticos, entender como as palavras se relacionam entre si e criar modelos que conseguem fazer coisas como traduzir textos automaticamente, analisar sentimentos, resumir textos e responder a perguntas (47).

No início, o NLP utilizava métodos baseados em símbolos e regras gramaticais claras. Depois disso, métodos estatísticos como Bag of Words (BoW), Term Frequency–Inverse Document Frequency (TF-IDF), Naive Bayes e Hidden Markov Models (HMM) passaram a ser muito usados em atividades de classificação e extração de informações. Porém, esses métodos tinham limitações em relação à representação de contexto e à captura de dependências semânticas de longa distância.

O progresso do aprendizado profundo revolucionou esse campo, trazendo redes neurais artificiais que conseguem aprender representações do texto, chamadas de embeddings. Modelos como Word2Vec, GloVe, BERT e GPT começaram a representar palavras e frases como vetores em dimensões múltiplas, onde a distância entre eles mostra quão semelhantes são em significado. Essa representação detalhada e contextual ajudou muito em tarefas de processamento de linguagem natural, permitindo entender melhor as nuances, ambiguidades e múltiplos significados das palavras (48).

No campo da segurança da informação, o aprendizado de máquina aplicado à linguagem natural tem se mostrado importante para identificar e reduzir ameaças relacionadas a texto, como:

- Phishing e engenharia social: reconhecimento de padrões linguísticos e significados repetidos em mensagens falsas;
- Malware textual: estudo de códigos e instruções prejudiciais dentro de documentos e e-mails;
- Análise de logs e alertas: categorização automática de eventos e identificação de comportamentos estranhos nos registros do sistema;
- Proteção contra Prompt Injection: identificação de comandos alterados usados para manipular modelos de linguagem.

Além disso, técnicas de fine-tuning e transfer learning ajudam a ajustar modelos já treinados para trabalhar em áreas específicas, como direito, medicina ou segurança cibernética, melhorando a precisão e a adequação das previsões. Métodos mais modernos, como o Reinforcement Learning from Human Feedback (RLHF), usam o feedback de pessoas durante o ajuste final dos modelos, ajudando a produzir respostas mais seguras e éticas (49).

Com o crescimento dos LLMs, o papel do aprendizado de máquina no processamento de linguagem natural cou ainda mais importante, exigindo métodos sólidos e éticos. A combinação de aprendizado supervisionado, não supervisionado e por reforço aumenta as chances de criar sistemas linguísticos inteligentes, que conseguem aprender sempre e se ajustar a novos cenários e riscos.

Entender as técnicas de aprendizado de máquina usadas na linguagem natural é muito importante não só para criar modelos linguísticos melhores, mas também para melhorar a defesa contra novos tipos de ataques, como o de injeção de prompt, e para desenvolver sistemas de IA geradora que sejam mais conáveis e capazes de resistir a problemas.

2.3.4 Modelos de Machine Learning

Para classificar os textos neste trabalho, foram utilizados modelos de Machine Learning supervisionado, que são muito reconhecidos por serem eficazes em tarefas de Processamento de Linguagem Natural. Cada modelo tem características próprias que afetam como ele funciona, o que ele consegue aprender e entender, e até como é possível interpretar suas decisões.

A seguir, são descritos os principais modelos utilizados:

2.3.4.1 Naive Bayes

O modelo Naive Bayes é baseado no teorema de Bayes e assume que as variáveis preditoras são independentes. Embora essa suposição geralmente não seja verdadeira nos dados reais, o modelo continua apresentando resultados bastante bons em muitas aplicações, especialmente na classificação de textos.

O algoritmo Naive Bayes é um tipo de classificador que usa probabilidades para decidir em que categoria um texto se encaixa, e é muito usado em tarefas como identificar e-mails indesejados, analisar o que as pessoas sentem em textos e organizar documentos em pastas específicas. O seu princípio principal é calcular a chance de que uma classe específica seja a correta para uma amostra de entrada x , que tem um conjunto de características ou palavras, x_1, x_2, \dots, x_n (50).

O teorema de Bayes mostra como calcular a relação abaixo:

$$P(C_k | x) = \frac{P(x | C_k) P(C_k)}{P(x)} \quad (2.1)$$

onde:

- $P(C_k | x)$ é a probabilidade de que a amostra pertença à classe C_k depois de conhecer as características x ;
- $P(x | C_k)$ é a probabilidade de ver as características x se a classe C_k for a verdadeira (chamada de verossimilhança);
- $P(C_k)$ é a probabilidade inicial de que a classe seja C_k (chamada de probabilidade a priori);
- $P(x)$ é a probabilidade de ver as características x , usada para ajustar os cálculos.

O termo "naive"(ingênuo) significa que se assume que todos os atributos são independentes uns dos outros, condição que é verdadeira apenas quando há uma dependência entre eles (51). Existem várias versões do Naive Bayes, dependendo do tipo de dados que estão sendo analisados:

- Multinomial Naive Bayes: é usado principalmente em tarefas de classificação de texto, onde os atributos mostram quantas vezes cada palavra aparece (bag-of-words).
- Bernoulli Naive Bayes: é bom para dados que têm apenas dois valores, como uma palavra estar presente ou não em um texto.
- Gaussian Naive Bayes: É utilizado quando as variáveis independentes são contínuas e têm uma distribuição normal.

Apesar de ser simples e basear-se em suposições fortes de independência, o Naive Bayes consegue apresentar um desempenho bom em muitas aplicações reais, especialmente quando há muitos dados e as relações entre as características são fracas (51). Sua principal limitação está no pressuposto de que os atributos são independentes, o que pode diminuir a precisão em áreas onde as palavras estão muito relacionadas entre si. Apesar disso, sua velocidade de treinamento e inferência o torna uma ótima opção como modelo base ou para usar em aplicações que exigem resposta rápida.

2.3.4.2 Support Vector Machines (SVM)

O Support Vector Machines (SVM) é um tipo de algoritmo de aprendizado supervisionado que é muito usado para fazer classificações e previsões de valores. Sua principal característica é a busca por um hiperplano que separa as classes de maneira muito distante, ou seja, com a maior distância possível entre os exemplos de diferentes classes.

Desenvolvido inicialmente por Vapnik e Cortes em 1995, o SVM busca identificar o hiperplano que separa melhor as classes em um espaço de características, garantindo que a margem entre os pontos mais próximos de cada classe, chamados de vetores de suporte, seja a maior possível (52).

No campo do Processamento de Linguagem Natural, o SVM é muito bom para classificar textos, como organizar documentos, entender o sentimento das pessoas, identificar mensagens indesejadas e reconhecer instruções perigosas. Isso acontece porque você consegue trabalhar com espaços que têm muitas dimensões, como os criados por representações vetoriais de texto (53).

O objetivo principal de uma SVM é encontrar o hiperplano ideal que atende à seguinte fórmula:

$$F(x) = \text{sign}(w \cdot x + b) \quad (2.2)$$

onde:

- w é o vetor de pesos, que é perpendicular ao hiperplano e é aprendido durante o treinamento;
- x são os vetores de entrada;

- é o termo de ajuste, também chamado de viés.
- A função sign, também conhecida como função degrau, gera uma saída que pode ser apenas +1 ou -1, dependendo do sinal do valor que está sendo analisado.

O termo $w \cdot x + b = 0$ define o hiperplano que funciona como a fronteira que separa as classes no classificador. Esse hiperplano divide o espaço de dados em duas partes, e cada parte representa uma classe. A distância entre o hiperplano e os vetores de suporte define a margem, que precisa ser aumentada o máximo possível para garantir que o modelo tenha um bom desempenho em novos dados (53).

Para problemas que não seguem uma linha reta, o SVM usa funções especiais chamadas funções kernel. Essas funções transformam os dados para um espaço com mais dimensões, onde é possível encontrar um plano que separa as classes de forma linear. As funções kernel mais usadas são:

- - Linear: usada quando os dados podem ser separados com uma linha reta;
- - Polinomial: usada para relações entre os dados que são mais complicadas;
- - RBF (Função de Base Radial): usada para identificar padrões que não são lineares, especialmente em conjuntos de dados com muitas dimensões.

Essas funções ajudam o modelo a entender relações difíceis e que não seguem uma linha reta entre as variáveis, tornando o SVM muito eficaz para tarefas como classificar textos, reconhecer padrões e identificar anomalias (53).

A capacidade do SVM de resistir ao overfitting, especialmente em espaços de alta dimensionalidade como os criados por vetores de texto, o torna um dos modelos mais usados em situações onde os dados são escassos ou desbalanceados. Além disso, sua base matemática forte e a capacidade de ser entendida facilitam a adoção no uso do SVM em aplicações importantes dentro do aprendizado de máquina aplicado à segurança (54).

Na prática, a forma mais comum de usar o SVM em projetos de desenvolvimento é através do algoritmo SVC (Support Vector Classification). O SVC permite ajustar parâmetros como o tipo de kernel, como linear, polinomial ou RBF, o que ajuda a adaptar o modelo para diferentes conjuntos de dados e objetivos de classificação (54).

2.3.4.3 Random Forests

O Random Forest é um tipo de algoritmo de aprendizado de máquina que usa o conceito de ensemble learning. Ele junta várias árvores de decisão para aumentar a precisão e a confiabilidade na classificação. Cada árvore é treinada com um conjunto de dados aleatório, onde os dados podem ser repetidos (técnica conhecida bootstrap). Ao fazer uma previsão, o modelo final decide com base no voto da maioria das árvores (55).

Cada árvore de decisão individual faz uma previsão para uma entrada específica x . O resultado final da floresta é feito combinando as previsões de todas as árvores, geralmente por votação da maioria quando é classificação, ou pela média aritmética quando é regressão (56).

A adição de aleatoriedade nos dados e nas características cria árvores que estão menos ligadas entre si, e quando essas árvores são combinadas usando o princípio do voto majoritário, o resultado é um modelo mais estável e com menos variação. Esse equilíbrio entre viés e variância é uma das principais vantagens do Random Forest em comparação com árvores de decisão individuais (55).

Essa abordagem minimiza bastante o risco de overfitting, que é algo frequentemente visto em árvores de decisão sozinhas, e ajuda o modelo a se adaptar melhor a novos dados. O Random Forest é muito bom para tarefas que envolvem dados com muitas características, como por exemplo, as representações em forma de vetores de texto usadas na PLN.

A maneira mais frequente de usar o Random Forest em ambientes de desenvolvimento é através da classe `RandomForestClassifier`. Essa implementação permite ajustar parâmetros como a quantidade de árvores, a profundidade máxima e as condições de divisão, facilitando a adaptação do modelo de acordo com a complexidade dos dados (56).

Assim, o Random Forest combina simplicidade, robustez e facilidade de interpretação, sendo uma opção muito usada em aplicações que precisam de bom desempenho e confiabilidade, especialmente em áreas como segurança cibernética e identificação de comportamentos estranhos em sistemas inteligentes.

2.3.4.4 Stochastic Gradient Descent (SGD)

O Stochastic Gradient Descent (SGD) é um método de otimização muito comum usado para treinar modelos de aprendizado de máquina, especialmente em tarefas como classificação e regressão. Em vez de calcular o gradiente do erro para todo o conjunto de dados de uma só vez, como acontece no gradiente descendente tradicional, o SGD ajusta os parâmetros do modelo usando apenas uma amostra ou um pequeno grupo de dados em cada passo, o que torna o processo mais eficiente e capaz de lidar com grandes volumes de informação. A principal vantagem está na maneira mais eficiente de usar a computação em comparação com o método usual de Gradient Descent (GD), que precisa calcular o gradiente para todos os dados a cada passo de atualização (57).

No contexto de classificação de texto, o SGD é comumente usado junto com modelos lineares, como a Regressão Logística ou SVM linear, por meio da implementação `SGDClassifier`. Essa forma de trabalhar permite treinar modelos que funcionam bem e são rápidos, mesmo quando os dados têm muitas características, como vetores TF-IDF ou word embeddings.

2.4 APRENDIZADO PROFUNDO (DEEP LEARNING)

Aprendizado Profundo (Deep Learning) é uma área do aprendizado de máquina que usa redes neurais artificiais com várias camadas para criar modelos que entendem melhor os dados complexos. Inspirado no modo como o cérebro humano funciona, o DL possibilita que sistemas computacionais aprendam hierarquias de abstrações, conseguindo extrair automaticamente características importantes a partir de dados brutos (58).

O DL ganhou atenção a partir da década de 2010, com o aparecimento de arquiteturas como as Redes

Neurais Convolucionais (CNNs), que são muito usadas em visão computacional, as Redes Neurais Recorrentes (RNNs), que ajudam no processamento de texto e fala em sequência, e as Redes Neurais Artificiais (ANNs), que são modelos computacionais inspirados na forma e no funcionamento do cérebro humano. Recentemente, os modelos baseados em Transformers transformaram o campo, proporcionando uma maior capacidade de trabalhar em paralelo e um desempenho melhor em tarefas relacionadas à linguagem natural (59).

Figura 2.2: Principais arquiteturas de redes neurais utilizadas em aprendizado profundo: ANN, CNN, RNN e Transformer, cada uma com características e aplicações distintas. Fonte: (2).

Ao contrário dos métodos tradicionais, o aprendizado profundo consegue identificar sozinho as características importantes dos dados brutos, sem precisar de ajustes manuais das características (feature engineering). Essa característica é muito importante em áreas como a linguagem, onde é necessário entender o contexto e lidar com ambiguidades, usando um conhecimento semântico mais profundo.

No campo da PLN, o aprendizado profundo mudou completamente a maneira como a linguagem é mostrada e tratada. Modelos como Word2Vec e GloVe trouxeram representações vetoriais densas (embeddings) que mostram como as palavras estão relacionadas em termos de significado. Depois disso, arquiteturas como LSTM e GRU permitiram tratar sequências que têm dependências no tempo, sendo muito usadas em tarefas como tradução automática, criação de textos e análise de sentimentos (60).

Recentemente, os Transformers se tornaram o padrão atual em PLN (23). Essa arquitetura dispensa o processamento em sequência, usando mecanismos de atenção para entender como as palavras se relacionam no contexto, mesmo que estejam em posições diferentes no texto. Modelos como BERT, GPT, T5 e RoBERTa são exemplos de redes que usam Transformers e tiveram resultados muito bons nos testes de compreensão de linguagem natural (61).

Além disso, métodos como o aprendizado por reforço têm sido aplicados para ajustar modelos de linguagem a objetivos específicos, como segurança e ética. Um exemplo conhecido é o Reinforcement Learning from Human Feedback (RLHF), que é usado para ajustar modelos de linguagem grande (LLMs) e impedir que eles deem respostas erradas ou perigosas (62).

Para estudar a segurança em modelos de linguagem, o DL fornece a base para criar detetores de ameaças mais eficazes, pois permite analisar o contexto das frases e identificar intenções ocultas. Redes neurais profundas, principalmente as que usam Transformers, são hoje as mais eficazes para identificar prompts prejudiciais.

2.5 VECTORIZAÇÃO

A vectorização é um passo importante no pré-processamento de texto para trabalhos de PLN. Como os algoritmos de aprendizado de máquina trabalham com dados numéricos, é preciso converter textos, que são feitos de palavras, em vetores que guardem as suas características linguísticas e o significado das palavras. Essa transformação é importante porque a maioria dos algoritmos de inteligência artificial trabalha com números, não com palavras diretamente (63).

Diversas formas podem ser usadas para transformar textos em vetores, cada uma com suas próprias vantagens e desvantagens. A seguir, apresentam-se as principais abordagens utilizadas neste trabalho:

- Bag of Words (BoW): representa um texto como um vetor que mostra a frequência das palavras, sem levar em conta a ordem ou o contexto (63, 64).

Apesar de ser simples, funciona bem para tarefas básicas de classificação de texto, mas não consegue entender relações entre as palavras. Dado um conjunto de documentos $D = \{d_1, d_2, \dots, d_n\}$ e um vocabulário $V = \{t_1, t_2, \dots, t_m\}$, cada documento d_i é convertido em um vetor:

$$x_i = [f(t_1; d_i); f(t_2; d_i); \dots; f(t_m; d_i)] \quad (2.3)$$

onde $f(t_j; d_i)$ representa a quantidade de vezes que o termo t_j aparece no documento d_i . Esse método não leva em conta a ordem das palavras, só conta quantas vezes cada uma aparece, o que torna o modelo mais simples, mas pode fazer perder um pouco do sentido das frases (64).

- TF-IDF (Term Frequency–Inverse Document Frequency): analisa quão comum uma palavra é dentro de um documento, considerando também quão rara ela é em todo o conjunto de documentos. Essa técnica diminui a importância das palavras que aparecem com frequência e chama mais atenção para termos que contêm mais informações, sendo muito usada em atividades como classificação e busca de informações (63, 65).

O TF-IDF melhora o BoW ao dar pesos às palavras conforme a sua importância dentro do conjunto de documentos. O peso do termo t no documento d é calculado da seguinte forma:

$$w_{t,d} = \frac{f_{t,d}}{\sum_d f_{t,d}} \log \frac{N}{n_t} \quad (2.4)$$

- $f_{t,d}$ é a quantidade de vezes que o termo t aparece no documento d ;
- N é o total de documentos no conjunto de documentos;
- n_t é o número de documentos que têm o termo t .

Assim, palavras que aparecem em todos os documentos, como "de", "a" e "o", têm um peso menor, enquanto palavras que aparecem apenas em documentos específicos têm um peso maior, o que aumenta a capacidade de distinguir esses documentos.

- Embeddings: mostram como as palavras estão relacionadas em sentido, transformando-as em um tipo de mapa com números que representam o significado de forma mais simples. O treinamento visa reduzir a diferença entre o contexto que se espera e o contexto real das palavras, garantindo que palavras que aparecem em contextos parecidos tenham representações semelhantes (63, 61).

O idealmente, cada palavra w_i é mostrada por um vetor denso:

$$w_i \in \mathbb{R}^n \quad (2.5)$$

onde n é a quantidade de dimensões do espaço de embeddings (por exemplo, 100, 200 ou 300 dimensões).

Modelos como BERT, RoBERTa e GPT criam vetores que levam em conta todo o contexto da palavra dentro da frase, o que ajuda a entender melhor as nuances da linguagem, as ambiguidades e as palavras com mais de um significado. Esses embeddings são obtidos de camadas intermediárias de modelos já treinados e estão se tornando comuns em aplicações atuais de processamento de linguagem natural (61).

A escolha da técnica de vetorização influencia diretamente no desempenho dos modelos de aprendizado de máquina. Embora técnicas tradicionais como o TF-IDF sejam rápidas e fáceis de entender, os embeddings densos e baseados no contexto têm uma melhor capacidade de se adaptar a diferentes situações, especialmente quando é necessário compreender o significado de forma mais profunda (63).

Em resumo, a vetorização é a ligação entre a linguagem natural e a representação matemática, permitindo que modelos como Naive Bayes, SVM, Random Forests e SGD trabalhem com dados textuais de forma eficiente e com sentido.

3 TRABALHOS RELACIONADOS

A literatura recente mostra um aumento de atenção para a segurança dos modelos de LLMs, especialmente em relação a ataques de injeção de prompt. Esses ataques são quando há instruções ruins colocadas dentro das entradas do usuário, nos contextos que são pegos ou mesmo em dados de fora, e a intenção é fazer o modelo agir de forma que não se quer. Nesta seção, são estudados os principais trabalhos sobre a detecção e prevenção desse tipo de ataque, destacando as suas formas de abordagem, os seus avanços e as suas dificuldades.

Estudos anteriores analisam a segurança dos Modelos de Linguagem, mostrando ataques como o Poison Prompt (3) e métodos de proteção, como o framework Palisade (16). Trabalhos como o de Rahman (66) utilizam modelos BERT multilíngues para melhorar a identificação de ataques de prompt injection em vários idiomas, enquanto métodos mais comuns, como Phishing Detection (42), baseiam-se na classificação supervisionada de conteúdo adversarial como teoria para proteger textos. Já o estudo "Breaking the Prompt Wall" (67) mostra como modelos muito usados são vulneráveis na prática, demonstrando que pequenas injeções podem alterar o comportamento de sistemas reais.

O PoisonPrompt (3) mostrou ser muito eficaz para comprometer tanto prompts rígidos quanto flexíveis em modelos de LLM, usando ataques do tipo backdoor. Esse tipo de ataque prejudica a integridade dos modelos ao colocar comandos escondidos durante a fase de ajuste no ou diretamente no texto que o usuário envia. Ele investiga a capacidade dos modelos de obedecer a instruções com pouca limitação, o que permite que agentes com má intenção causem ações surpreendentes, sem que essa alteração seja facilmente identificada. No entanto, a criação e a proteção contra o PoisonPrompt são difíceis, porque ele é muito complexo e exige um entendimento profundo das técnicas de prompt e da estrutura dos modelos. Além disso, a eficácia do ataque pode variar com base no acesso antecipado ao processo de treinamento ou na modificação direta das entradas fornecidas pelo usuário, o que pode dificultar seu uso em certos contextos.

O PromptCARE apresenta um novo método para proteger o direito de autor dos prompts, usando marcas d'água que são inseridas e verificadas, garantindo assim a autoria e a integridade dos prompts enviados para sistemas de modelos de linguagem de larga escala (3). Contrário a métodos que só se preocupam com ataques, o PromptCARE atua de forma preventiva, inserindo sinais sutis durante a criação dos prompts, permitindo que futuras verificações identifiquem alterações ou uso indevido. O método demonstra ser estável e difícil de notar, protegendo os prompts sem prejudicar seu funcionamento, mas a sua utilização ainda depende da padronização dos sistemas de verificação integrados às plataformas de inteligência artificial.

Por outro lado, Kokkula e Divya propuseram o Palisade (16), um framework que tem como objetivo detectar e reduzir os efeitos da injeção de prompts. A diferença é que os métodos tradicionais costumam focar em filtrar conteúdos diretos, mas o Palisade usa técnicas de aprendizado de máquina para detectar padrões mais discretos nos prompts, analisando tanto a forma como as frases são construídas quanto o propósito real que o usuário deseja transmitir.

Para reduzir esse problema, os pesquisadores criaram o framework Palisade (16), que usa uma abor-

dagem em camadas para identificar quando um ataque de injeção de prompt acontece. O sistema funciona usando três etapas diferentes de Itro. Primeiro, usa um Itro com regras de linguagem para pegar padrões claros nas palavras. Depois, um classificador com métodos de aprendizado de máquina faz uma análise estatística das entradas. Por fim, um modelo de linguagem extra é usado para entender melhor o significado do conteúdo. Essa forma de trabalhar mistura métodos para diminuir tanto erros de falha quanto erros de excesso, tornando o sistema mais eficaz contra diversos tipos de ataques. No entanto, o custo computacional extra e o efeito na demora da resposta são limitações notadas, principalmente em situações onde o uso é em tempo real.

Rahman destaca o uso do BERT multilíngue como uma solução versátil para detectar injeções de prompt em vários idiomas. Ao criar embeddings que capturam as nuances semânticas, essa abordagem supera métodos antigos, alcançando uma precisão de até 96,55% usando regressão logística, e melhora a capacidade de classificação binária diante de ataques complexos (66). Esse método é especialmente útil em situações globais, onde os modelos LLM recebem informações em vários idiomas, mas seu funcionamento depende da qualidade dos dados usados para treinamento, principalmente para identificar versões criativas de comandos maliciosos.

Mesmo sem tratar diretamente de prompt injection, estudos como o de Shahrivari (42) utilizam técnicas de aprendizado de máquina para identificar ataques de phishing e outras formas de manipulação usando linguagem natural. O artigo usa vários algoritmos supervisionados, como o Random Forest, o SVM e o Naive Bayes, para classificar textos suspeitos com base em características linguísticas e na estrutura dos textos. Essa linha de pesquisa é importante porque mostra como modelos de classificação de texto podem ser usados para a segurança da informação, e ela pode ser ajustada para identificar padrões de intenção perigosa nos prompts.

Embora não sejam tão eficazes na identificação de ataques complexos, essas abordagens ainda têm importância na construção dos primeiros Itros em frameworks como Palisade e continuam sendo úteis como referência para comparação. Esses estudos mostram que o problema de identificar instruções perigosas não é só dos grandes modelos de linguagem, mas faz parte de um conjunto mais amplo de dificuldades relacionadas à segurança na linguagem.

O estudo "Breaking the Prompt Wall" (67) apresenta uma análise prática sobre ataques de injeção de prompt em sistemas de modelos linguísticos de grande escala, como o ChatGPT, mostrando como pequenas injeções de prompt podem alterar as respostas e o comportamento do sistema, mesmo quando há regras de itração muito rigorosas. Os autores mostram que instruções simples e sutis, como injeções curtas, podem passar por cima de sistemas de defesa que dependem apenas de Itros de palavras ou verificações rápidas. O artigo destaca a necessidade de criar mecanismos de proteção que considerem o contexto e mostra que, mesmo com medidas de segurança em vigor, é possível explorar falhas usando linguagem com duplo sentido ou alterações sutis em instruções.

Outro estudo recente, apresentado em (68), sugere o PromptShield, um método que combina módulos de atenção supervisionada com técnicas de reinforcement learning para melhorar a identificação de ações perigosas enquanto os LLMs são usados em produção. O sistema foi testado em vários cenários diferentes e mostrou uma queda grande no número de falhas de segurança, o que mostra quão importante é combinarmos o aprendizado supervisionado com métodos de reforço.

Esses estudos de caso mostram como os modelos realmente se comportam quando são atacados, e ajudam a entender melhor como avaliar novos métodos de proteção, incluindo os que usam aprendizado profundo. Esse trabalho mostra que é necessário usar sistemas integrados como o Palisade, o PromptCARE e algoritmos multicamadas, porque métodos separados geralmente não são bons o suficiente para lidar com a criatividade dos atacantes.

A Tabela 3.1 apresenta as principais características dos trabalhos analisados, mostrando as suas abordagens, vantagens e limitações.

Tabela 3.1: Resumo comparativo dos principais trabalhos relacionados

Artigo	Abordagem	Tipo de Defesa	Principal Mérito	Limitação
PromptCARE (3)	Watermark / injeção	Preventiva/ Verificação	Robustez e invisibilidade	Padronização e adoção nos sistemas
Palisade (16)	Multi-camadas (regra, ML)	Detector/ Híbrida	Redução falsos negativos	Impacto operacional e latência
Rahman et al. (66)	BERT multilíngue	Detector/ Supervisionada	Alta precisão em vários idiomas	Dependência de dados e limitação para ataques sofisticados
Shahrivari (42)	ML tradicional	Detector/ Supervisionada	Base teórica para defesas	Fraca contra ataques sofisticados
Breaking the Prompt Wall (67)	Estudos de caso	Análise de vulnerabilidade	Demonstração prática	Sem solução dedicada, apenas identificação
Este Trabalho	ML tradicional + BERT + comparação com LLMs comerciais	Detector/ Híbrida comparativa	Abordagem sistêmica com balanceamento de classes e avaliação cruzada com modelos de mercado	Avaliação restrita a dataset específico e APIs públicas

Apesar dos progressos desses estudos sobre a segurança das grandes linguagens de máquina, ainda falta uma análise completa sobre como usar várias técnicas de inteligência artificial para detectar a injeção de prompts.

A análise desses trabalhos mostra progressos importantes, mas também aponta pontos que ainda precisam ser melhorados. Muitos estudos não tratam a detecção de prompt injection de forma abrangente, considerando tanto o contexto quanto o significado, além de diferentes tipos de entrada, como o texto do usuário, informações externas e o histórico das conversas. Além disso, faltam avaliações comparativas padronizadas e conjuntos de dados públicos que ajudem a repetir os resultados obtidos.

Dessa forma, esse trabalho ajuda a melhorar os sistemas de inteligência artificial, ao comparar como diferentes algoritmos de aprendizado de máquina funcionam na detecção de ataques, mostra como as estratégias de balanceamento de dados afetam o desempenho dos modelos e também permite estudar como modelos já treinados se saem em relação aos métodos mais tradicionais. Uma das vantagens desse método é a capacidade de se adaptar a vários tipos de ataques, sem precisar de regras muito específicas que já estejam de-

nidas antes. No entanto, o framework Palisade pode ser atacado por métodos mais avançados, que tentam esconder comandos danosos usando frases inventivas ou linguagem com duplo sentido. Como em qualquer sistema que usa aprendizado de máquina, o desempenho do Palisade depende diretamente da qualidade e da variedade dos dados usados para treiná-lo. Este estudo busca preencher parte dessa lacuna, comparando o desempenho de diferentes algoritmos, analisando como o balanceamento de dados influencia a performance dos modelos e investigando qual é a eficácia entre modelos tradicionais e modelos pré-treinados. Ao explorar esse tema com mais profundidade, o objetivo é ajudar na criação de soluções mais fortes e eficazes para combater ameaças que estão surgindo, aumentando assim a proteção dos sistemas que usam inteligência artificial.

4 ARQUITETURA PROPOSTA

O sistema criado para detectar ataques de injeção de prompt usa métodos de aprendizado de máquina e modelos de linguagem natural para reconhecer e impedir tentativas de alterar as entradas de texto de forma prejudicial. A arquitetura geral, mostrada na Figura 4.1, está dividida em etapas que seguem uma ordem específica, começando pela coleta e preparação dos dados até a análise e identificação das entradas suspeitas.

Figura 4.1: Arquitetura do sistema de detecção de prompt injection

Primeiro, os dados são organizados e enviados para diferentes métodos: (i) modelos de LLMs, como OpenAI, LLaMA e Gemini, que analisam as perguntas e dão respostas individuais; e (ii) algoritmos de aprendizado de máquina, incluindo os baseados em BERT e técnicas mais convencionais, que são ensinados e testados para reconhecer padrões que indicam ataques. Os resultados das dessas abordagens são juntados e estudados, o que ajuda a tomar uma decisão sobre se houve ou não uma injeção de prompt.

Essa combinação entre LLMs e aprendizado de máquina visa melhorar a capacidade do sistema, usando tanto a capacidade dos modelos de entender contexto quanto a exatidão dos classificados treinados com supervisão. A metodologia sugerida assegura uma avaliação bem estruturada, com verificação mútua e indicadores de desempenho, garantindo que a detecção de ameaças seja confiável. O processo é dividido em etapas bem definidas, que incluem a preparação dos dados, o treinamento, o teste e a avaliação dos resultados, que serão explicadas a seguir.

4.1 COLETA DE DADOS

O conjunto de dados foi criado para ajudar na tarefa de identificar automaticamente ataques de injeção de prompt em situações reais de uso de modelos grandes de linguagem, sendo tratada como um problema de classificação binária supervisionada. Foi criado um conjunto de dados personalizado em português para ajudar no desenvolvimento do sistema que detecta ataques de injeção de prompt. Esse conjunto inclui exemplos de diferentes formas de ataques, como injeções diretas, instruções que contradizem o que é esperado e tentativas de fazer o modelo agir de maneira diferente da prevista.

4.1.1 Construção do dataset em português

Primeiro foi criado um corpus original em português, com exemplos de conversas entre usuário e modelo que abrangem diversos tipos de ataque, como injeções diretas (“ignore todas as instruções anteriores e...”), ordens que contradizem o funcionamento do sistema, tentativas de contornar filtros de segurança e solicitações claras de ajuda para atividades ilegais (como, por exemplo, criar payloads XSS ou scripts para roubo de senhas) (69).

Cada exemplo foi marcado manualmente com o valor 1 (prompt malicioso) ou 0 (prompt legítimo), com base na intenção de induzir o modelo para que adote comportamentos perigosos. Mesmo que o texto tire proveito de termos técnicos que se parecem entre as categorias, o rótulo é definido conforme a finalidade. Por exemplo, perguntas educativas sobre XSS foram classificadas como 0, enquanto orientações práticas para explorar XSS receberam o rótulo 1 (70).

4.1.2 Integração de datasets públicos em inglês

Para melhorar a diversidade de cenários de ataque e diminuir o viés linguístico, o corpus em português foi complementado com dois conjuntos de dados públicos em inglês focados em prompt injection, citados em (71, 72). Esses datasets contêm exemplos com diferentes métodos, como jailbreaking, override contextual e vazamento de instruções, o que aumenta a diversidade de padrões disponíveis para treinar detectores baseados em modelos de linguagem e classificadores tradicionais (73).

Depois de juntar e misturar esses dados de forma aleatória, foi criado um único dataset bilíngue, seguindo as sugestões apresentadas por (74).

4.1.3 Critérios de rotulagem e controle de qualidade

A rotulagem seguiu critérios operacionais baseados em estudos recentes sobre detecção de prompt injection, ou seja, prompts que pedem para ignorar, mudar ou burlar instruções do sistema, acessar informações confidenciais ou gerar conteúdo que viole as regras de uso. Esses prompts foram marcados como maliciosos, mesmo que fossem apresentados de forma indireta ou persuasiva. Já perguntas informativas, pedagógicas ou que utilizem a segurança de maneira legítima foram classificadas como benignas (71).

Para diminuir o ruído, foram feitas revisões manuais em amostras ambíguas, como prompts que contêm

termos de ataques com intenção educativa, garantindo que as categorias sejam consistentes e que o conjunto de dados seja mais útil como benchmark para modelos discriminativos (69).

4.1.4 Disponibilização e reprodutibilidade

O conjunto de dados resultante desse processo forma um único grupo bilíngue (português e inglês) com prompts tanto benignos quanto maliciosos, adequado tanto para treinar quanto para avaliar modelos de detecção de injeção de prompt.

Para ajudar a repetir os resultados e comparar com estudos futuros, todo o conjunto de dados foi posto em um repositório público na plataforma Hugging Face, disponível em: <<https://huggingface.co/datasets/PromptInjectionDataset/Injection-Attack-Detection-Dataset>>.

4.2 BALANCEAMENTO E PROCESSAMENTO

Para treinar os modelos de aprendizado de máquina e o modelo BERT, utilizou-se o conjunto de dados completo, dividido nas proporções 80/10/10 para treino, validação e teste, mantendo as proporções originais das classes em cada parte através da amostragem estratificada (usando a função scikit-learn's `train_test_split` com parâmetro `stratify=y`).

4.2.1 Balanceamento de Classes

Como os dados usados têm um desequilíbrio, com mais prompts maliciosos do que legítimos, usamos uma forma de balancear as classes. Isso ajudou os modelos a aprender de maneira justa, evitando que cassem tendenciosos para os prompts legítimos. Para isso, foram criadas três versões do conjunto de dados:

1. Dataset Original (Normal): Preserva a distribuição natural das classes, permitindo que se avalie o desempenho dos modelos sem necessidade de ajustar o equilíbrio entre as classes.
2. Oversampling: Aumenta artificialmente a quantidade de exemplos da classe minoritária (maliciosa) copiando as instâncias já existentes e usando técnicas como Random Over-Sampling. Essa forma de abordar o problema quer diminuir o viés que o classificador tem em relação à classe que aparece com mais frequência, tornando mais fácil detectar os ataques.
3. Undersampling: Diminui a quantidade de exemplos da classe com mais exemplos (maliciosas) para que as classes fiquem mais equilibradas entre si. Embora o tamanho total do conjunto de dados seja reduzido, essa abordagem permite testar o desempenho dos modelos em situações mais equilibradas, evitando que certos prompts maliciosos apareçam com muito mais frequência.

Cada uma dessas versões foi usada para treinar os modelos BERT e os tradicionais, permitindo comparar diretamente o desempenho e analisar o efeito do balanceamento nas métricas como acurácia, precisão,

recall e F1-score. Essa abordagem ajuda a ver se o modelo tem uma melhor capacidade de generalizar quando os dados são equilibrados artificialmente, reduzidos ou mantidos com a distribuição natural.

Além disso, todas as técnicas de balanceamento foram usadas antes do pré-processamento do texto, garantindo que a alteração do número de exemplos não prejudicasse a consistência das representações vetoriais ou das embeddings criadas pelo BERT.

4.2.2 Subconjunto para Avaliação com LLMs

Para os experimentos que envolvem a geração de respostas pelos modelos de linguagem natural (LLMs), utilizou-se uma versão reduzida do conjunto de dados, formada por 5% dos exemplos escolhidos de forma aleatória. Essa amostragem foi feita para equilibrar a variedade das entradas e diminuir o tempo que leva para processar as respostas durante a avaliação dos LLMs, garantindo que todos os tipos de ataques e prompts legítimos sejam representados.

O conjunto de dados reduzido foi processado por três modelos LLM diferentes: GPT-4-mini (75), Llama 3.1-70B (76) e Gemini 1.5-Flash (77). A escolha desses modelos levou em conta a sua capacidade reconhecida de entender textos complexos e produzir respostas lógicas, além de possuírem arquiteturas sólidas e um histórico de treinamento muito longo. Ressalta-se que esses LLMs foram utilizados neste trabalho como modelos comparadores, ou seja, como referência externa de desempenho em relação às abordagens supervisionadas propostas.

Cada modelo recebeu os prompts de entrada de forma individual, permitindo comparar como cada um reagiu frente a ataques de injeção de prompt. Os dados foram inseridos na etapa de pré-prompting (etapa 2.1 do pipeline arquitetural, conforme Figura 4.1), com o objetivo de auxiliar na classificação dos prompts maliciosos a partir das respostas geradas. Dessa forma, explorou-se também o conhecimento previamente adquirido por esses modelos durante seu treinamento em larga escala, incluindo mecanismos internos de reconhecimento de padrões associados a ataques de prompt injection. Sempre que possível, buscou-se capturar respostas brutas por meio de APIs públicas, preservando os metadados originais de cada instância, como idioma e fonte, o que possibilita análises complementares por origem linguística ou tipo de ataque.

4.2.3 Protocolo de Avaliação Independente

A execução independente para cada modelo garante que os efeitos de priming cruzado sejam isolados, permitindo analisar de forma comparativa as vulnerabilidades específicas de cada arquitetura diante de diferentes táticas de injeção de prompt, como as diretas, indiretas e jailbreaks. As respostas geradas foram armazenadas no formato JSON, servindo de base para as etapas seguintes de rotulagem automática e análise de impacto, conforme descrito nos tópicos 3.1 a 3.2 e ilustrado na Figura 4.1.

4.3 MODELAGEM E TESTE DE MODELOS

O dataset completo, após ser dividido com particionamento estratificado (80% para treino, 10% para validação e 10% para teste), foi usado como base para treinar e avaliar dois grupos diferentes de modelos.

4.3.1 Modelo BERT

Para o modelo baseado em BERT, utilizou-se a arquitetura bert-base-uncased, seguindo um processo organizado de treinamento inicial, validação, ajuste final e armazenamento do modelo (78). O fluxo de treinamento incluiu:

1. Treinamento inicial (3.1.1): ajustes de hiperparâmetros básicos, otimização com AdamW e de nicção de learning rate adequado;
2. Testes (3.1.2): avaliação inicial da acurácia, precisão, recall e F1-score no conjunto de validação;
3. Re namento final (3.1.3): ajustes nos dos hiperparâmetros e uso de técnicas de regularização para diminuir o over tting;
4. Persistência (3.1.4): salvar o modelo treinado para ser usado em inferência e em experimentos futuros.

4.3.2 Modelos Tradicionais de Machine Learning

Nos modelos de ML, foi necessário criar bases de comparação interpretáveis, utilizaram-se pipelines de vetorização com TF-IDF e CountVectorizer, seguidos de classi cadores probabilísticos e lineares, seguindo as boas práticas recomendadas para classi cação de texto (79).

Tabela 4.1: Pipelines de Classi cação Tradicional Implementados

Pipeline	Vetorizador	Classi cador	Con guração do Pipeline
1	CountVectorizer	SGDClassifier	Parâmetros default (loss='hinge', penalty='l2')
2	TfidfVectorizer	SGDClassifier	Parâmetros default (loss='hinge', penalty='l2')
3	TfidfVectorizer	MultinomialNB	Parâmetros default (alpha=1.0)
4	CountVectorizer	MultinomialNB	Parâmetros default (alpha=1.0)
5	TfidfVectorizer	SVC	kernel="linear", C=1.0, probability=True
6	CountVectorizer	SVC	kernel="linear", C=1.0, probability=True
7	TfidfVectorizer	RandomForestClassifier	n_estimators=100, random_state=42
8	CountVectorizer	RandomForestClassifier	n_estimators=100, random_state=42

A Tabela 4.1 mostra os oito pipelines de classi cação tradicional usados para detectar prompts maliciosos, usando dois métodos de vetorização de texto com quatro algoritmos de classi cação supervisionada. Cada pipeline segue a estrutura Pipeline() do scikit-learn, combinando a extração de características e a classi cação em um processo automatizado e contínuo.

Essa abordagem sistemática cria referências comparáveis que mostram as diferenças entre capturar apenas o signi cado superficial das palavras (CountVectorizer) e dar mais peso às palavras importantes

(TfidfVectorizer), ajudando a entender como diferentes formas de representar as palavras afetam a detecção de padrões de injeção de prompts. Todos os pipelines usam `random_state=42` para garantir que os experimentos sejam totalmente reproduzíveis.

O fluxo de treinamento desses modelos seguiu etapas semelhantes às do BERT:

1. Treinamento inicial (3.2.1): transformação dos textos em números e treinamento inicial dos classificadores;
2. Testes (3.2.2): avaliação das métricas de desempenho e identificação das melhores combinações entre vetorizador e classificador.
3. Renamamento (3.2.3): ajustes de hiperparâmetros e técnicas de validação cruzada para otimizar o desempenho;
4. Persistência (3.2.4): salvamento dos modelos ajustados para serem usados novamente e para comparações experimentais no futuro.

O processo de modelagem e teste seguiu protocolos de reprodutibilidade e padronização, conforme descrito na referência (74), assegurando que os experimentos fossem consistentes e poderiam ser comparados entre diferentes técnicas. A Figura 4.1 mostra a estrutura geral e o processo de treinamento dos modelos, mostrando como os passos de pré-processamento, treinamento, validação e armazenamento estão conectados.

4.4 ESTRATÉGIA DE COMBINAÇÃO DE MODELOS

As previsões individuais dos modelos de aprendizado de máquina (BERT e modelos tradicionais de ML) e dos modelos de LLMs foram combinadas usando um mecanismo de decisão com voto majoritário.

Para cada texto, um vetor armazena as classificações de cada modelo, mostrando se o texto foi reconhecido como um ataque (True) ou como um texto legítimo (False). A classe final atribuída à entrada é a que aparece com mais frequência entre os modelos, assegurando que a decisão considere as diferentes abordagens e técnicas de aprendizado utilizadas.

Em caso de empate entre as classes, aplicou-se uma regra mais cautelosa, atribuindo a entrada à classe maliciosa, com o objetivo de detectar melhor os ataques e reduzir os falsos negativos.

O uso do voto majoritário ajuda a fortalecer o sistema, diminuindo o impacto de erros individuais e aproveitando as vantagens de diferentes métodos. Enquanto o BERT e os classificadores tradicionais identificam padrões estruturais e estatísticos nos prompts, os modelos LLM analisam melhor o significado e detectam detalhes contextuais que podem mostrar sinais de manipulação.

Além disso, essa estratégia ajuda a comparar diferentes combinações de modelos e conjuntos de dados balanceados, permitindo avaliar como o balanceamento das classes e o tipo de modelo influenciam na detecção de ataques finais. O sistema de agregação por maioria também é fácil de expandir, permitindo que se adicionem novos modelos ou métodos de ponderação em projetos futuros.

4.5 ANÁLISE E VALIDAÇÃO DOS RESULTADOS

A avaliação do desempenho dos modelos foi feita usando métricas de classificação padrão, que foram tiradas da biblioteca `sklearn.metrics`, como acurácia, precisão, recall e F1-score. Essas métricas ajudaram a medir quão bem cada modelo consegue distinguir prompts verdadeiros dos maliciosos, além de mostrar se há preferências ou erros em relação a classe minoritárias ou majoritárias.

Para assegurar que os resultados sejam fortes e confiáveis, utilizou-se a validação cruzada k-fold, como mostra as seções 1.1.2 e 1.1.3. Essa abordagem divide o conjunto de dados em K partes, trocando as funções de treino e teste em cada rodada, o que ajuda a buscar estimativas mais confiáveis e diminuir a variação no desempenho entre as diferentes formas de dividir os dados.

Além das métricas globais, foram verificados os padrões de erros de classificação, como falsos positivos e falsos negativos, para descobrir quais características nos textos estão ligadas a esses erros. Essa análise ajudou a entender as limitações e as vantagens de cada método, além de permitir ajustes precisos nos modelos e nas estratégias para equilibrar as classes.

No fim, usar métricas numéricas junto com uma análise detalhada dos erros ajudou a avaliar como o sistema integrado está funcionando, levando em conta tanto os modelos de aprendizado de máquina e o BERT quanto os grandes modelos de linguagem, e deu dicas para melhorar no futuro a detecção de ataques de Injeção de Prompt.

4.6 INTERFACE DE DECISÃO E INTERAÇÃO COM O USUÁRIO

Na última etapa do processo (etapa 6 da Figura 4.1), o sistema utiliza um mecanismo de decisão por maioria para classificar cada entrada como Prompt Injection (True) ou legítima (False). Essa fase dá ao usuário uma resposta clara e fácil de entender sobre se há ou não ataques, ajudando a tomar medidas para prevenir ou resolver problemas nos sistemas que usam modelos de linguagem natural.

A interação com o usuário foi criada para garantir clareza e confiabilidade máxima: o resultado final é mostrado junto com detalhes adicionais, como as probabilidades de cada classificação, a quantidade de modelos que indicaram um ataque e as métricas de confiança relacionadas. Essa abordagem permite que usuários e operadores entendam não só o resultado final, mas também quão segura e consistente é a decisão, mesmo que se usem diferentes modelos.

A consistência e a confiabilidade dessa etapa são garantidas pelos processos de validação explicados na Figura 4.1 (seção 6.2), garantindo que os resultados sejam comunicados de forma clara e evitando erros que podem surgir por inconsistências nos modelos ou no tratamento das entradas antes de serem usadas.

5 TESTE E RESULTADOS

Este capítulo mostra e estuda os resultados dos experimentos feitos para identificar ataques de Prompt Injection, usando diversos modelos de aprendizado de máquina, métodos de transformação de texto em vetores e abordagens para equilibrar as classes. O objetivo principal é verificar como essas estratégias influenciam no desempenho dos modelos e também encontrar as configurações que melhor se adaptam a situações de segurança.

5.1 DISTRIBUIÇÃO DAS CLASSES

Antes de analisar como os modelos se desempenham, é importante entender como as diferentes categorias estão distribuídas nos dados que foram usados neste estudo. Como mostra a Figura 5.1, há um desequilíbrio muito grande entre as classes, com muitas entradas marcadas como maliciosas em comparação com as amostras legítimas no dataset original.

	Malicioso	Legítimo	Total de amostras
Undersampling	506	506	1012
Oversampling	1039	1039	2078
Desbalanceado	1039	506	1545

Tabela 5.1: Distribuição das classes nos diferentes datasets utilizados

A Tabela 5.1 mostra como as classes estão distribuídas depois de aplicar as diferentes estratégias de balanceamento usadas neste trabalho: dataset desbalanceado (unbalanced), balanceamento por undersampling e balanceamento por oversampling. Essas abordagens foram usadas para entender como o balanceamento das classes afeta a performance dos modelos que detectam ataques de Prompt Injection.

Esse cenário mostra uma coisa comum em problemas de segurança da informação, onde eventos ruins acontecem com menos frequência ou estão distribuídos de forma diferente em relação aos ataques reais. Em situações de aprendizado de máquina, esse tipo de desbalanceamento pode fazer com que os modelos favoreçam a classe que aparece com mais frequência, gerando resultados globais que parecem bons, mas que não são bons o suficiente para identificar com precisão a classe que aparece menos ou que é mais importante detectar.

Para reduzir esse efeito e permitir uma análise mais cuidadosa, foram avaliados três tipos diferentes de dados: (i) o conjunto original, que tem um desbalanceamento de classes, (ii) um conjunto equilibrado por subamostragem, onde foram removidas algumas amostras da classe que aparece com mais frequência, e (iii) um conjunto equilibrado por sobreamostragem, onde foram adicionadas amostras falsas ou repetidas da classe que aparece com menos frequência. Essa estratégia permite analisar de forma sistemática como o equilíbrio entre as classes influencia o funcionamento dos modelos e os erros que eles cometem ao classificar.

Figura 5.1: Distribuição das classes no dataset desbalanceado para detecção de ataques de Prompt Injection (total de 1545 amostras).

5.2 DESEMPENHO DOS MODELOS TRADICIONAIS DE MACHINE LEARNING

Os experimentos analisaram de forma sistemática o desempenho de oito pipelines convencionais de aprendizado de máquina utilizando três métodos de equilíbrio de classes: oversampling, undersampling e a distribuição original desbalanceada. A análise mostra que essas estratégias têm um impacto importante na identificação de prompts maliciosos, principalmente em situações que imitam condições reais de uso.

A Tabela 5.2 mostra os resultados de cada combinação entre método de vetorização de texto, tipo de classificador e forma de balanceamento, utilizando as métricas de acurácia, precisão e F1-score. Essas métricas ajudam a comparar como os modelos se comportam em relação à capacidade de generalizar e identificar a classe minoritária.

Tabela 5.2: Desempenho de modelos tradicionais de ML com diferentes estratégias de balanceamento

Modelo	Pipeline	Acurácia	Precisão	F1-Score	Acurácia	Precisão	F1-Score	Acurácia	Precisão	F1-Score
CountVectorizer & SGDClassifier	1	0,96	0,96	0,96	0,91	0,91	0,91	0,92	0,92	0,92
TfidfVectorizer & SGDClassifier	2	0,97	0,97	0,97	0,92	0,92	0,92	0,94	0,94	0,94
TfidfVectorizer & MultinomialNB	3	0,95	0,95	0,95	0,91	0,91	0,91	0,88	0,90	0,87
CountVectorizer & MultinomialNB	4	0,95	0,95	0,95	0,91	0,91	0,91	0,92	0,92	0,92
TfidfVectorizer & SVC	5	0,97	0,97	0,97	0,92	0,92	0,92	0,94	0,94	0,94
CountVectorizer & SVC	6	0,97	0,97	0,97	0,91	0,91	0,91	0,93	0,93	0,93
TfidfVectorizer & RandomForestClassifier	7	0,96	0,96	0,96	0,88	0,88	0,88	0,90	0,90	0,89
CountVectorizer & RandomForestClassifier	8	0,96	0,96	0,96	0,90	0,90	0,90	0,89	0,89	0,89
		Oversampling			Undersampling			Unbalanced		

Os resultados mostram que os modelos treinados com oversampling tiveram desempenho melhor em

grande parte das métricas avaliadas, especialmente no F1-score e na acurácia da validação. Esse comportamento mostra que ao aumentar o número de pessoas da classe minoritária, houve uma redução real do viés nos classificadores, e a precisão não sofreu um impacto muito grande.

Ao contrário disso, os modelos treinados com undersampling mostraram uma redução constante no desempenho, especialmente na métrica F1-score. Essa redução mostra a perda de informação importante que acontece quando se remove muitas instâncias da classe mais numerosa, o que prejudica a capacidade dos modelos de se adaptarem a novos dados.

Os modelos treinados com o dataset desbalanceado obtiveram resultados intermediários, superiores aos observados com undersampling, porém inferiores aos obtidos com oversampling. Essa postura indica que, embora alguns classificadores possam lidar em parte com o desbalanceamento, a falta de estratégias claras diminuiu o desempenho na detecção da classe com menor número de amostras.

5.2.1 Curvas de Aprendizado

A Figura 5.2 mostra as curvas de aprendizado dos principais modelos avaliados usando diferentes estratégias de balanceamento. É possível notar que os modelos que foram treinados com oversampling mostram curvas mais consistentes, com menos diferença entre o desempenho no treino e na validação, o que sugere uma melhor capacidade de se adaptar a novos dados.

Já os modelos que passaram por undersampling mostram mais variação e sinais de overfitting, principalmente quando o número de amostras de treinamento é menor. Os modelos treinados com o conjunto de dados desbalanceado (unbalanced) mostram um comportamento intermediário, com um desempenho razoável, mas que fica mais afetado por mudanças.

As curvas de aprendizado corroboram os resultados da Tabela 5.2, mostrando que os modelos treinados com oversampling convergem melhor e atingem níveis mais altos de acurácia na validação, em comparação aos outros cenários.

Em geral, os experimentos mostram que as estratégias de oversampling deram os melhores resultados no geral, com taxas de acurácia e F1-score sempre maiores do que as obtidas com undersampling e com os dados originais. Esse comportamento está de acordo com os valores da Tabela 5.2 e com as curvas da Figura 5.2, e indica que aumentar a classe minoritária ajudou a diminuir o viés dos classificadores em favor da classe majoritária, melhorando a capacidade de identificar prompts maliciosos, sem prejudicar muito a precisão.

Destaca-se que os pipelines que utilizam o `TfidfVectorizer` junto com classificadores lineares, como o `SGDClassifier` e o `SVC`, tiveram um desempenho muito bom quando aplicados com a estratégia de oversampling, atingindo valores de F1-score próximos ou acima de 97%. Esse resultado mostra que a forma como o TF-IDF dá peso aos termos ajuda a identificar padrões de significado relacionados a ataques de Prompt Injection, principalmente quando usada junto com modelos que trabalham bem em espaços de dados complexos e com muitas dimensões.

Ao contrário disso, os classificadores baseados em árvores, como o `RandomForestClassifier`, foram mais sensíveis ao undersampling, apresentando os menores valores de F1-score nesse cenário. Esse com-

Figura 5.2: Curvas de aprendizado para diferentes estratégias de balanceamento

portamento mostra que esses modelos precisam de mais dados que representem bem as coisas para entender melhor os padrões importantes.

No caso de datasets desequilibrados, veri ca-se manutenção de níveis competitivos de acurácia, mas

há uma redução visível na precisão e no F1-score, o que sugere uma maior tendência para falsos negativos. Em situações de segurança, esse efeito é muito importante, porque não identificar ameaças pode levar a consequências muito graves. De forma geral, a análise comparativa mostra que:

1. Oversampling é a estratégia mais eficaz para identificar ataques de Prompt Injection nesse contexto;
2. TF-IDF junto com classificadores lineares oferece um bom equilíbrio entre desempenho e estabilidade;
3. Undersampling afeta a representatividade dos dados e reduz a capacidade dos modelos de prever corretamente;
4. Usar dados não balanceados, embora possível, tende a favorecer a classe com mais exemplos, o que prejudica as métricas que dependem da classe minoritária.

5.2.2 Análise da Matriz de Confusão

Para uma análise mais completa dos erros de classificação, foram analisadas as matrizes de confusão médias provenientes dos modelos tradicionais de Machine Learning, levando em conta as diferentes formas de balanceamento das classes. A Figura 5.3 mostra a média das matrizes de confusão nos cenários de unbalanced, undersampling e oversampling.

Figura 5.3: Matriz de confusão média para diferentes estratégias de balanceamento

Nota-se que o dataset desbalanceado mostra mais falsos positivos na classe legítima, mas o undersampling diminui esse problema, mesmo que possa resultar na perda de informação, o que causa um aumento nos erros nas duas classes. Em contraste, o cenário com oversampling mostra a menor taxa de falsos negativos, o que mostra que ele é mais sensível à classe maliciosa.

Esse resultado é especialmente importante em situações de segurança, pois não detectar um ataque pode causar consequências muito más do que ter alertas que não são reais. Assim, a matriz de confusão mostra que usar oversampling ajuda bastante a tornar o sistema de detecção de Prompt Injection mais resistente, sem reduzir muito a precisão geral.

As matrizes de confusão individuais de cada classificador estão no Apêndice I.1.

5.3 DESEMPENHO DO MODELO BERT

Esta seção mostra como o modelo BERT se comporta com diferentes formas de equilibrar as classes, para verificar como ele reage a mudanças na forma como os dados estão distribuídos. A Tabela 5.3 mostra os resultados em relação à acurácia, precisão e F1-score para os cenários de undersampling, oversampling e dataset desbalanceado.

	Acurácia	Precisão	F1-score
Undersampling	0,88	0,88	0,88
Oversampling	0,96	0,96	0,96
Desbalanceado	0,93	0,93	0,93

Tabela 5.3: Desempenho do modelo BERT sob diferentes estratégias de balanceamento

Os resultados mostram que, como acontece nos modelos tradicionais de aprendizado de máquina, o uso de oversampling melhora o desempenho geral do modelo BERT, com taxas de acurácia, precisão e F1-score chegando a 96%. Isso indica que aumentar a representação da classe minoritária ajuda a melhorar a capacidade do modelo de distinguir entre diferentes categorias, mesmo em arquiteturas complexas que usam representações baseadas no contexto.

Figura 5.4: Comparação BERT: Oversampling (0.96 F1) > Desbalanceado (0.93) > Undersampling (0.88). Robustez inerente reduz gap.

Ao contrário dos modelos tradicionais, o BERT mostra variações menores entre os diferentes cenários de balanceamento. Pode-se notar que o desempenho do modelo com o conjunto de dados desbalanceado continua alto (F1-score de 93%), o que sugere que o modelo é mais resistente a distribuições desiguais entre as classes. Essa característica pode ser explicada pela habilidade do BERT de entender relações de

significado profundas e dependências do contexto, o que diminui a necessidade de ajustes artificiais para equilibrar os dados.

Outro ponto importante é a uniformidade das métricas analisadas. Para todas as estratégias de balanceamento, os valores de acurácia, precisão e F1-score são quase iguais entre si, mostrando que os erros de falso positivo e falso negativo estão equilibrados. Essa estabilidade indica que o modelo mostra um comportamento previsível e constante tanto durante o treinamento quanto quando é usado para fazer previsões.

Para ajudar a comparar visualmente os diferentes cenários analisados, os resultados estão mostrados na Figura 5.4, o que permite analisar diretamente como as estratégias de balanceamento afetam o desempenho do modelo.

A Figura 5.4 confirma que a estratégia de oversampling é a melhor, pois obtém o melhor desempenho geral com um F1-score de 96%. Apesar disso, é possível notar a força do modelo BERT, já que o treinamento com um dataset desbalanceado ainda consegue manter um desempenho alto (F1-score = 93%), muito superior ao caso de undersampling (F1-score = 88%). Esse resultado mostra que o modelo é menos sensível às distribuições desiguais de classes, em comparação com os métodos convencionais de aprendizado de máquina.

Figura 5.5: Matriz de confusão do modelo BERT com oversampling, evidenciando baixo número de falsos negativos (FN = 5), o que indica elevado recall para a classe maliciosa.

A análise cuidadosa da matriz de confusão mostrada na Figura 5.5 mostra que há poucos falsos negativos (FN = 5), o que mostra que o modelo consegue identificar bem os prompts maliciosos. Esse ponto é muito importante em situações de segurança, pois a presença de falsos negativos pode trazer mais risco

para o funcionamento do que os falsos positivos, podendo levar à falta de identificação de ataques reais.

A estabilidade do modelo BERT foi verificada em diferentes divisões do conjunto de dados usando validação cruzada com cinco grupos. Os resultados mais completos, com as métricas de cada K-fold, estão no Apêndice I.1.3, mostrando que as execuções tiveram pouca diferença entre si.

5.4 PROMPTS TREINADOS

Depois de treinar e testar vários modelos de aprendizado de máquina, pensando em diferentes formas de equilibrar as classes, o estudo passou para a análise do comportamento de LLMs que são muito usados no mercado. Essa etapa tinha o objetivo de usar esses modelos como comparação, funcionando como um referência externa para comparar os resultados das abordagens supervisionadas que foram propostas nesse trabalho. A intenção não era substituir os modelos já treinados, mas ver como sistemas comerciais já estabelecidos reagem a prompts que podem ser prejudiciais.

Foram escolhidos três modelos LLMs que representam o estado atual da tecnologia: OpenAI, Gemini e Llama. Ao contrário dos modelos supervisionados treinados nesta pesquisa, esses sistemas já possuem mecanismos internos para combater a prompt injection, criados pelos próprios fornecedores durante seu treinamento em grande escala. Assim, ao usá-los, também se leva em conta o conhecimento que esses modelos já adquiriram sobre padrões de ataques e comportamentos que os adversários tendem a usar.

Por causa da forma proprietária dessas arquiteturas e da falta de acesso direto às suas partes internas, não é possível calcular métricas comuns como acurácia, precisão ou F1-score. Dessa forma, a análise foi feita com os valores de saída que os próprios modelos retornaram, que mostram a chance estimada de uma entrada específica ser considerada uma tentativa de Prompt Injection. Esses valores foram analisados de forma comparativa, verificando tendências e se as respostas mantêm consistência.

Para os testes, criou-se um pré-prompt padronizado que foi usado de forma igual para os três modelos. Esse pré-prompt serviu para explicar o que precisava ser feito e ajudar a identificar se as entradas continham injeção de prompt, usando os mecanismos que já foram treinados nos LLMs. As entradas usadas representam 5% do total de dados usados para treinar os modelos de machine learning, sendo escolhidas aleatoriamente em cada rodada do experimento, com a intenção de diminuir vieses e melhorar a variedade das amostras analisadas.

A Tabela 5.4 mostra os resultados, que estão entre 0 e 1, e valores mais próximos de 1 significam que o modelo tem mais certeza de que a entrada é uma tentativa de Prompt Injection.

Modelo	Resultados
OpenAI	1.0
Gemini	0.95
Llama	1.0

Tabela 5.4: Resultados de modelos de IA de mercado em tarefa específica

Os resultados mostram que os modelos OpenAI e Llama foram muito conservadores, marcando todas as entradas analisadas como possivelmente maliciosas. O modelo Gemini, embora tenha mostrado alta

sensibilidade, teve uma pequena mudança na resposta, o que resultou em um valor médio um pouco abaixo de 1,0. Esses resultados mostram que os mecanismos de segurança internos dos LLMs atuais funcionam bem, mas também revelam pequenas diferenças no comportamento entre diferentes tipos de arquitetura.

5.5 ANÁLISE DOS RESULTADOS

Mesmo com uma quantidade não muito grande de dados, os modelos de machine learning usados nesse estudo tiveram desempenho semelhante ao de modelos de inteligência artificial muito usados no mercado. Isso mostra que, quando bem ajustados e treinados, modelos tradicionais de aprendizado supervisionado podem ser usados de maneira eficiente em áreas específicas, como a identificação de ataques de Prompt Injection dentro do campo de Segurança da Informação.

A comparação entre os experimentos mostra que o modelo BERT teve os resultados mais uniformes em todas as métricas analisadas, mesmo com diferentes formas de balanceamento. Essa estabilidade mostra que o modelo consegue se adaptar melhor a diferentes situações, o que é muito importante em casos reais onde o tipo de dados pode mudar com o tempo. Esse comportamento pode ser explicado pelo uso de representações contextuais profundas, que conseguem capturar relações de significado complexas no texto, algo essencial para identificar ataques de Prompt Injection, onde pequenas mudanças nas palavras podem mudar muito o sentido da mensagem.

Figura 5.6: Comparação das métricas de acurácia, precisão e F1-score para os modelos avaliados sob diferentes estratégias de balanceamento de classes.

Como mostra a Figura 5.6, usar oversampling resulta em valores maiores de F1-score para a maioria dos modelos testados, especialmente para o BERT e os classificadores que usam `TfidfVectorizer`. Por outro lado, o uso de undersampling normalmente pega o desempenho geral, especialmente quando se fala do recall da classe maliciosa.

Portanto, podemos ver que o modelo BERT conseguiu valores muito bons de F1-score, chegando perto de 96% no cenário com oversampling, destacando-se especialmente na capacidade de identificar prompts

maliciosos mais difíceis de detectar. Em comparação, os modelos de aprendizado de máquina tradicionais tiveram desempenho bom no F1-score médio, mas foram mais afetados quando houve mudanças no equilíbrio entre as classes.

No caso dos modelos tradicionais, os classificadores que usaram o `TfidfVectorizer` tiveram desempenho mais consistente do que aqueles que utilizaram o `CountVectorizer`, com melhorias médias de cerca de 0.01 no F1-score. Esse resultado mostra quão importante é usar a ponderação IDF para identificar n-gramas raros, que geralmente estão ligados a tentativas de jailbreak e manipulação semântica, especialmente quando combinada com classificadores lineares como SGD e SVC.

Abordagem híbrida que combina BERT, TF-IDF linear e ensemble é válida para personalização específica do domínio, superando os modelos genéricos em cenários bilíngues controlados (16, 3). As limitações envolvem o tamanho do conjunto de dados (futuro: mais 10 mil amostras sintéticas) e a generalização entre domínios (ex: jailbreaks multilíngues).

Além dos resultados numéricos, esse ponto é muito importante, pois ataques de Prompt Injection geralmente envolvem informações delicadas ou tentativas de desrespeitar as regras de uso. Portanto, além do bom desempenho técnico, é essencial adotar práticas responsáveis no desenvolvimento e uso de sistemas de IA. Isso inclui tratar os dados de forma correta, reduzir riscos à privacidade e diminuir possíveis vieses, para que essas tecnologias sejam usadas de maneira ética e segura (80).

6 CONCLUSÃO

Este trabalho apresentou uma abordagem híbrida para identificar ataques de Prompt Injection, uma das ameaças emergentes mais importantes no campo da segurança aplicada a modelos de linguagem natural. A proposta uniu métodos de aprendizado de máquina convencionais, modelos baseados em Transformers e técnicas para equilibrar as classes, visando melhorar a resistência e a confiança dos sistemas que usam LLMs.

Os resultados dos experimentos mostraram que usar modelos de Machine Learning é bom para reconhecer padrões que estão relacionados a ataques de Prompt Injection. Em particular, pipelines que usam o TfidfVectorizer junto com classificadores lineares, como o SGDClassifier, assim como modelos baseados no BERT, tiveram um desempenho muito bom, atingindo acurácia e F1-score acima de 97%. Esses resultados mostram que, mesmo usando estruturas mais simples e com um custo menor de processamento, é possível atingir taxas altas de detecção, especialmente quando combinadas com métodos corretos de equilíbrio entre as classes.

A comparação entre conjuntos de dataset desbalanceados, balanceados por subamostragem (undersampling) e por superamostragem (oversampling) mostrou como é importante tratar o desbalanceamento nas tarefas de segurança. Percebeu-se que o oversampling ajudou muito a diminuir os falsos negativos, algo muito importante em situações onde não detectar um ataque pode causar danos graves à segurança e à privacidade dos dados. Assim, a proposta apresentada mostra que é possível aplicar essa abordagem de forma prática, mesmo em lugares onde os recursos de computação são limitados.

Como pontos fortes, destacam-se:

- a criação e avaliação de vários fluxos de trabalho usando diferentes métodos para equilibrar os dados;
- a comparação entre modelos supervisionados e LLMs comerciais amplamente utilizados;
- a obtenção de alto desempenho com modelos de menor custo computacional, o que ajuda na aplicação prática;
- a sugestão de uma abordagem mista que pode ser usada e ajustada em diferentes ambientes de trabalho; e
- a contribuição para a segurança da inteligência artificial com uma solução automática, compreensível e adequada para situações reais de uso.

Resumindo, esse estudo ajuda a avançar as pesquisas sobre segurança dos modelos de linguagem ao mostrar que métodos de aprendizado de máquina já conhecidos, quando usados corretamente e combinados, podem funcionar bem para combater ataques de injeção de prompt. Os resultados mostram que soluções automatizadas e explicáveis são importantes para proteger aplicações que usam modelos de linguagem.

6.1 TRABALHOS FUTUROS

Como trabalhos futuros, sugerem-se as seguintes direções de pesquisa:

- Avaliar como os modelos se saem em diferentes idiomas, tipos de texto e estilos de escrita, verificando se são eficazes em cenários com diversas culturas e línguas;
- Incorporar a abordagem proposta em sistemas que detectam ameaças em tempo real, analisando como isso afeta a rapidez, capacidade de escalar e processamento de grandes volumes de dados continuamente;
- Ampliar o conjunto de dados com novas línguas e tipos de ataques, incluindo ataques indiretos, em etapas múltiplas e técnicas mais complexas de manipulação do texto;
- Utilizar técnicas de ajuste fino e otimização de parâmetros, além de estratégias mais avançadas, visando aumentar o desempenho do modelo e diminuir erros como falsos positivos e falsos negativos;
- Analisar a capacidade do modelo de lidar com diferentes tipos de texto, como código de programação, mensagens formatadas, documentos técnicos e linguagem informais;
- Criar mecanismos para enfrentar ataques que evoluem ou se adaptam, em que o atacante ajusta continuamente o prompt para enganar o sistema de detecção;
- Estudar métodos baseados em identificação de anomalias ou aprendizado contínuo para lidar com novos tipos de ataques e padrões que não foram vistos durante o treinamento do modelo.

REFERÊNCIAS BIBLIOGRÁFICAS

- 1 AL-SERW, N. A.-R. Undersampling and oversampling: An old and a new approach. <https://medium.com/>, 2021. Disponível em: <<https://medium.com/analytics-vidhya/undersampling-and-oversampling-an-old-and-a-new-approach-4f984a0e8392>>.
- 2 KEERTHANAMS. Building Blocks of Deep Learning: ANN, CNN, RNN, and Transformers. 2025. Disponível em: <<https://medium.com/@keerthanams1208/building-blocks-of-deep-learning-ann-cnn-rnn-and-transformers-0e0f830d090f>>.
- 3 YAO, H. et al. Promptcare: Prompt copyright protection by watermark injection and verification. In: IEEE. 2024 IEEE Symposium on Security and Privacy (SP). [S.l.], 2024. p. 845–861.
- 4 KOSHIRY, A. M. E. et al. A prediction system using ai techniques to predict students' learning difficulties using lms for sustainable development at kfu. In: Proceedings of the Computational Methods in Systems and Software. [S.l.]: Springer, 2022. p. 22–36.
- 5 GONÇALVES, J. J. O. et al. Detecção de prompt injection em modelos de linguagem. Revista Ibérica de Sistemas e Tecnologias de Informação, Associação Ibérica de Sistemas e Tecnologias de Informação, n. E77, p. 104–116, 2025.
- 6 GONÇALVES, J. J.; LOTUFO, T. Repositório do código-fonte para Detecção de Prompt Injection em modelos de Linguagem. 2025. Disponível em: <https://github.com/juliajamileg/Prompt_injection>.
- 7 ALEXANDRIA, J. d. Gestão de segurança da informação—uma proposta para potencializar a efetividade da segurança da informação em ambiente de pesquisa científica. Instituto de Pesquisas Energéticas e Nucleares, Universidade de São Paulo, São Paulo, 2009.
- 8 LNCC, L. N. de C. C. Os quatro pilares da segurança da informação – Confidencialidade, Disponibilidade, Integridade e Autenticidade. 2024. Disponível em: <<https://www.gov.br/lncc/pt-br/centrais-de-conteudo/campanhas-de-conscientizacao/gestao-de-seguranca-da-informacao/os-quatro-pilares-da-seguranca-da-informacao-2013-confidencialidade-disponibilidade-integridade-e-autenticidade>>.
- 9 SOUSA, C. E. d. Proposta de um framework para enriquecimento de inteligência de ameaças cibernéticas utilizando informações a partir de inteligência de fontes abertas. 2025.
- 10 ZHANG, J. et al. When llms meet cybersecurity: A systematic literature review. Cybersecurity, Springer, v. 8, n. 1, p. 55, 2025.
- 11 ADMIN, O. OWASP Top 10 for LLM Applications 2025 - OWASP Top 10 for LLM & Generative AI Security. 2024. Disponível em: <<https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>>.
- 12 PROMPTFOO. OWASP GenAI Red Team Best Practices for LLM Applications. 2024. <<https://www.promptfoo.dev/docs/red-team/owasp-llm-top-10/>>. Acessado em: 20 fev. 2026.
- 13 AI, C. Secure your LLM apps with OWASP's 2025 Top 10 for LLMs and Citadel AI. 2024. Disponível em: <<https://citadel-ai.com/blog/2024/11/25/owasp-llm-2025/>>.
- 14 HUNG, K.-H. et al. Attention tracker: Detecting prompt injection attacks in llms. arXiv preprint arXiv:2411.00348, 2024.

- 15 ZHANG, W. et al. A study on prompt injection attack against llm-integrated mobile robotic systems. In: IEEE. 2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW). [S.l.], 2024. p. 361–368.
- 16 KOKKULA, S.; DIVYA, G. et al. Palisade: Prompt injection detection framework. arXiv preprint arXiv:2410.21146, 2024.
- 17 YI, J. et al. Benchmarking and defending against indirect prompt injection attacks on large language models. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1. [S.l.: s.n.], 2025. p. 1809–1820.
- 18 ALTURAYEIF, N.; HASSINE, J. Data leakage detection in machine learning code: transfer learning, active learning, or low-shot prompting? PeerJ Computer Science, PeerJ Inc., v. 11, p. e2730, 2025.
- 19 MUMTARIN, M.; CHOWDHURY, M. S.; WOOD, J. Large language models in analyzing crash narratives—a comparative study of chatgpt, bard and gpt-4. arXiv preprint arXiv:2308.13563, 2023.
- 20 MURER, R. Fundamentos da Inteligência Artificial: o futuro é agora. [S.l.]: Alta Books, 2025.
- 21 TAULLI, T. Introdução à inteligência artificial: uma abordagem não técnica. [S.l.]: Novatec Editora, 2020.
- 22 BROWN, T. et al. Language models are few-shot learners. Advances in neural information processing systems, v. 33, p. 1877–1901, 2020.
- 23 PARMAR, N. et al. Image transformer. In: PMLR. International conference on machine learning. [S.l.], 2018. p. 4055–4064.
- 24 PEREZ, F.; RIBEIRO, I. Ignore previous prompt: Attack techniques for language models. arXiv preprint arXiv:2211.09527, 2022.
- 25 NAQA, I. E.; MURPHY, M. J. What is machine learning? In: machine learning in radiation oncology. [S.l.]: Springer, 2015. p. 3–11.
- 26 SOUZA, F. R. et al. Simulação de diálogos e personagens no chat-gpt4: análise comparativa do desempenho em idiomas inglês e português. In: Anais do IV Congresso Brasileiro Interdisciplinar em Ciência e Tecnologias. IV Cobicet: online. [S.l.: s.n.], 2023. p. 1–8.
- 27 LANGLEY, P.; SIMON, H. A. Applications of machine learning and rule induction. Communications of the ACM, ACM New York, NY, USA, v. 38, n. 11, p. 54–64, 1995.
- 28 FERREIRA, P. N. Aprendizado de máquina. [S.l.]: Editora Senac São Paulo, 2024.
- 29 BATISTA, J. J. S. Recomendação de produtos financeiros utilizando aprendizado de máquina. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2024.
- 30 CASTRO, C. L. d.; BRAGA, A. P. Aprendizado supervisionado com conjuntos de dados desbalanceados. Sba: Controle & Automação Sociedade Brasileira de Automatica, SciELO Brasil, v. 22, p. 441–466, 2011.
- 31 SILVA, C. B. et al. Uma análise comparativa das técnicas de machine learning: Regressão logística, árvores de decisão, random forest e svm. Apoena, v. 7, p. 501–511, 2023.
- 32 FERNANDES, C. R. Machine learning, deep learning e aplicações. Monumenta-Revista Científica Multidisciplinar, v. 9, n. 9, p. 1–2, 2024.

- 33 MORAIS, P. H. C. Estudo comparativo entre aprendizado supervisionado e aprendizado semi-supervisionado aplicados a problemas de classi cação de dados. [S.l.], 2023.
- 34 FARIA, G.; ROMERO, R. A. F. Navegação de robôs móveis utilizando aprendizado por reforço e lógica fuzzy. Sba: Controle & Automação Sociedade Brasileira de Automatica, SciELO Brasil, v. 13, p. 219–230, 2002.
- 35 JABBAR, H.; KHAN, R. Z. Methods to avoid over- tting and under- tting in supervised machine learning (comparative study). Computer science, communication and instrumentation devices, Res. Publ Singapore, v. 70, n. 10.3850, p. 978–981, 2015.
- 36 MOHAMMED, R.; RAWASHDEH, J.; ABDULLAH, M. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In: IEEE. 2020 11th international conference on information and communication systems (ICICS). [S.l.], 2020. p. 243–248.
- 37 MUJAHID, M. et al. Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering. Journal of Big Data, Springer, v. 11, n. 1, p. 87, 2024.
- 38 ILEBERI, E.; SUN, Y.; WANG, Z. Performance evaluation of machine learning methods for credit card fraud detection using smote and adaboost. IEEE access, IEEE, v. 9, p. 165286–165294, 2021.
- 39 SHAKHNAROVICH, G.; DARRELL, T.; INDYK, P. Nearest-neighbor methods in learning and vision. IEEE Trans. Neural Networks, v. 19, n. 2, p. 377, 2008.
- 40 ELHASSAN, T.; ALJURF, M. et al. Classi cation of imbalance data using tokek link (t-link) combined with random under-sampling (rus) as a data reduction method. Global J Technol Optim S, v. 1, n. S1, 2016.
- 41 LIN, W.-C. et al. Clustering-based undersampling in class-imbalanced data. Information Sciences, Elsevier, v. 409, p. 17–26, 2017.
- 42 SHAHRIVARI, V.; DARABI, M. M.; IZADI, M. Phishing detection using machine learning techniques. arXiv preprint arXiv:2009.11116, 2020.
- 43 WANG, C. et al. Safeguarding crowdsourcing surveys from chatgpt with prompt injection. arXiv preprint arXiv:2306.08833, 2023.
- 44 TORRES, J. M.; COMESANA, C. I.; GARCÍA-NIETO, P. J. Machine learning techniques applied to cybersecurity. International Journal of Machine Learning and Cybernetics, Springer, v. 10, n. 10, p. 2823–2836, 2019.
- 45 SHAYEGANI, E. et al. Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv preprint arXiv:2310.10844, 2023.
- 46 NASTESKI, V. An overview of the supervised machine learning methods. Horizons. b, v. 4, p. 51–62, 2017.
- 47 CHENG, M.; DURMUS, E.; JURAFSKY, D. Marked personas: Using natural language prompts to measure stereotypes in language models. arXiv preprint arXiv:2305.18189, 2023.
- 48 WU, Y. et al. Research on the application of deep learning-based bert model in sentiment analysis. arXiv preprint arXiv:2403.08217, 2024.
- 49 OUYANG, L. et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, v. 35, p. 27730–27744, 2022.

- 50 RISH, I. et al. An empirical study of the naive bayes classifier. In: SEATTLE, USA. *IJCAI 2001 workshop on empirical methods in artificial intelligence*. [S.l.], 2001. v. 3, n. 22, p. 41–46.
- 51 ZEVAREX, V. A. de O.; SANTOS, C. H. da S. Estudos das implicações do teorema de bayes na computação natural. *Revista Brasileira de Iniciação Científica*, p. 58–79, 2020.
- 52 CORTES, C.; VAPNIK, V. Support vector machine. *Machine learning*, v. 20, n. 3, p. 273–297, 1995.
- 53 TUHUTERU, H.; IRIANI, A. Analisis sentimen perusahaan listrik negara cabang ambon menggunakan metode support vector machine dan naive bayes classifier. *Jurnal Informatika: Jurnal Pengembangan IT*, v. 3, n. 3, p. 394–401, 2018.
- 54 SOUZA, D. L. de et al. Fault detection and diagnosis using support vector machines-a svc and svr comparison. *J. Saf. Eng*, v. 3, n. 1, p. 18–29, 2014.
- 55 LIU, Y.; WANG, Y.; ZHANG, J. New machine learning algorithm: Random forest. In: SPRINGER. *International conference on information computing and applications*. [S.l.], 2012. p. 246–252.
- 56 JATNIKA, H. et al. Application of random forest classification method in determining the best quality service in the implementation of international certification at itcc itpln. *Jurnal E-Komtek*, v. 9, n. 1, p. 163–168, 2025.
- 57 YANG, Z. et al. Stability and differential privacy of stochastic gradient descent for pairwise learning with non-smooth loss. In: PMLR. *International Conference on Artificial Intelligence and Statistics*. [S.l.], 2021. p. 2026–2034.
- 58 LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.
- 59 TANG, G. et al. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*, 2018.
- 60 RAKSHIT, P.; SARKAR, A. A supervised deep learning-based sentiment analysis by the implementation of word2vec and glove embedding techniques. *Multimedia Tools and Applications*, Springer, v. 84, n. 2, p. 979–1012, 2025.
- 61 MEDEIROS, I. E. A. d. et al. *Análise e priorização de alarmes industriais utilizando word embeddings e técnicas de aprendizado de máquina*. [S.l.]: Universidade Federal da Paraíba, 2024.
- 62 BAI, Y. et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- 63 TREVILATO, L. A. Análise comparativa de modelos de representação de texto e métodos de aprendizado de máquina na classificação de notícias falsas em português. Universidade Federal de Uberlândia, 2025.
- 64 PIRES, F. B. Vetorização de textos com bag-of-words e aprendizado de máquina para cadastro de ncms-nomenclatura comum do mercosul. *Universidade Federal do Rio Grande do Sul. Instituto de Matemática e Estatística. Curso de Estatística: Bacharelado.*, 2024.
- 65 NODA, M. Levantamento de indicadores através de data mining, latent dirichlet allocation e tf-idf. *Repositório Institucional do FGV*, 2020.
- 66 RAHMAN, M. A. et al. Applying pre-trained multilingual bert in embeddings for improved malicious prompt injection attacks detection. In: IEEE. *2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*. [S.l.], 2024. p. 1–7.

- 67 CHANG, X. et al. Breaking the prompt wall (i): A real-world case study of attacking chatgpt via lightweight prompt injection. *arXiv preprint arXiv:2504.16125*, 2025.
- 68 JACOB, D. et al. Promptshield: Deployable detection for prompt injection attacks. In: *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy*. [S.l.: s.n.], 2024. p. 341–352.
- 69 PEARCY, S. *Evaluating Prompt Injection Datasets*. 2025. Disponível em: <<https://hiddenlayer.com/innovation-hub/evaluating-prompt-injection-datasets/>>.
- 70 LI, H.; LIU, X. Injecguard: Benchmarking and mitigating over-defense in prompt injection guardrail models. *arXiv preprint arXiv:2410.22770*, 2024.
- 71 JAZIRI, C. *Malicious and Benign Dataset*. 2024. Disponível em: <<https://www.kaggle.com/datasets/chaimajaziri/malicious-and-benign-dataset>>.
- 72 HARELIX. *HarelIX/Prompt-Injection-Mixed-Techniques-2024*. 2023. Disponível em: <<https://huggingface.co/datasets/HarelIX/Prompt-Injection-Mixed-Techniques-2024/blob/main/train.csv>>.
- 73 TestSavant.AI. *ROBUST GUARDRAILS FOR MITIGATING PROMPT INJECTION AND JAILBREAK ATTACKS IN LLMS*. TestSavant, 2024. Disponível em: <https://testsavant.ai/wp-content/uploads/2024/11/TestSavant_AI_Technical_Paper.pdf>.
- 74 LIMA, M. d. *6 passos Para Criar Seu Primeiro Projeto de Machine Learning*. Insight Data Science Lab - Portal da UFC - Universidade Federal do Ceará - Universidade Federal do Ceará (UFC), 2022. Disponível em: <<https://www.insightlab.ufc.br/6-passos-para-criar-seu-primeiro-projeto-de-machine-learning/>>.
- 75 OPENAI. *OpenAI API*. 2021. Disponível em: <<https://openai.com/api/>>.
- 76 LLAMA. *Llama*. 2024. Disponível em: <<https://console.llamaapi.com/>>.
- 77 API, G. *Google Gemini API*. 2023. Disponível em: <<https://aistudio.google.com/>>.
- 78 VAJ, T. *Let's code K-fold validation on BERT*. 2023. Disponível em: <<https://vtiya.medium.com/let-s-code-k-fold-validation-on-bert-722f9438f932>>.
- 79 PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011.
- 80 PALOMINO-FLORES, P.; CRISTI-LÓPEZ, R. Del código a la creatividad: Explorando los horizontes éticos de la creación de contenido con inteligencia artificial. *Revista Ibérica De Sistemas e Tecnologias De Informação*, Associação Ibérica de Sistemas e Tecnologias de Informacao, n. E66, p. 355–369, 2024.

I.1 RESULTADOS EXPERIMENTAIS DETALHADOS

I.1.1 Curvas de Aprendizado Detalhadas - Machine Learn

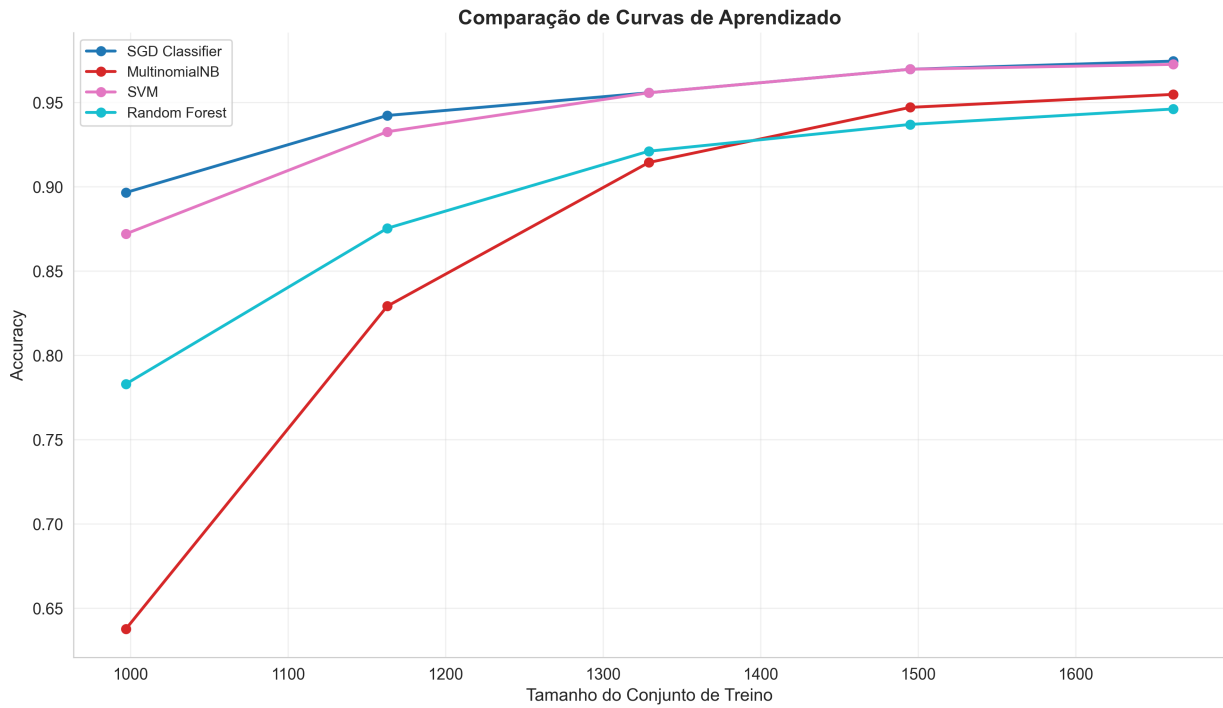


Figura 1: Curvas individuais sob oversampling (melhor estratégia).

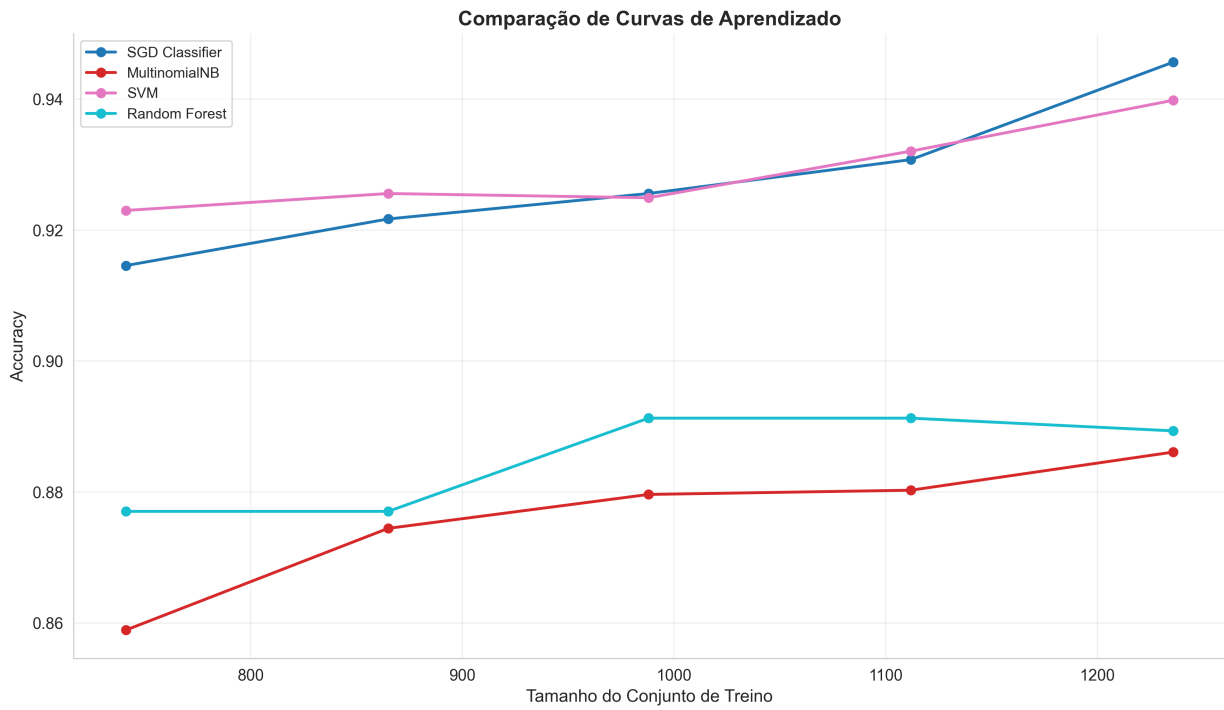


Figura 2: Curvas individuais sob banco desbalanceado.

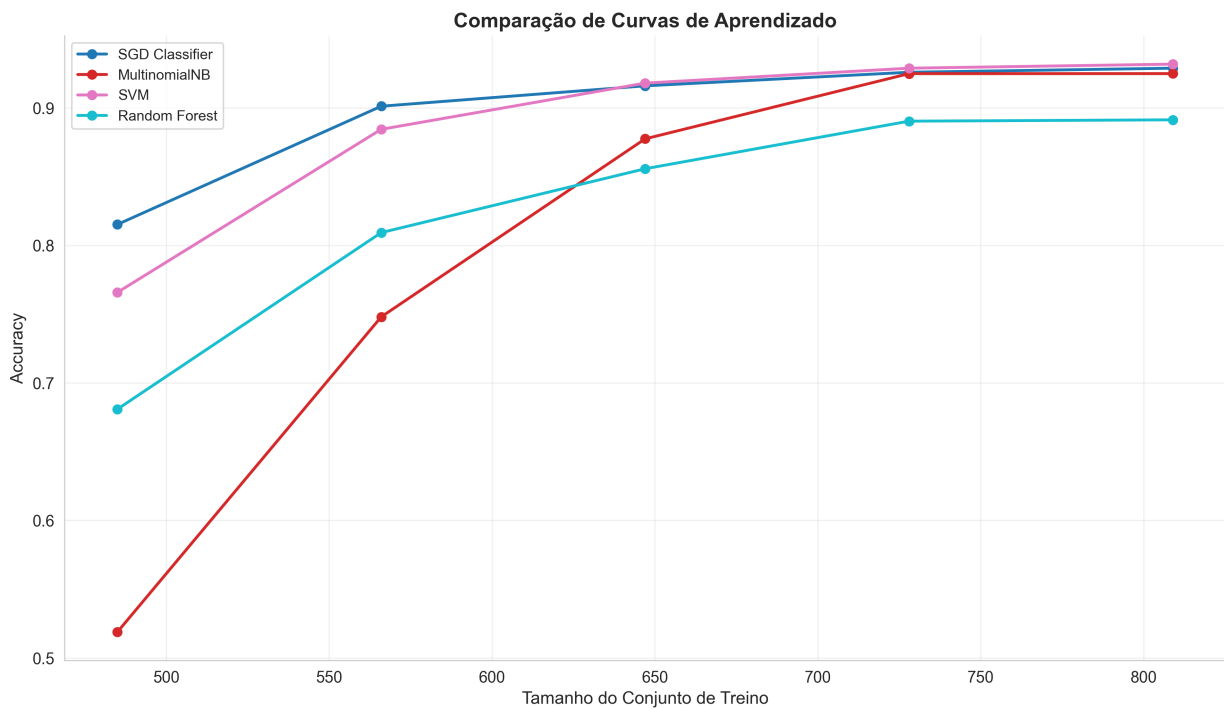


Figura 3: Curvas individuais sob undersampling.

I.1.2 Matrizes de Confusão por Estratégia - Machine Learn

Matrizes de Confusão - Oversampling (Validação Cruzada 5-Fold)

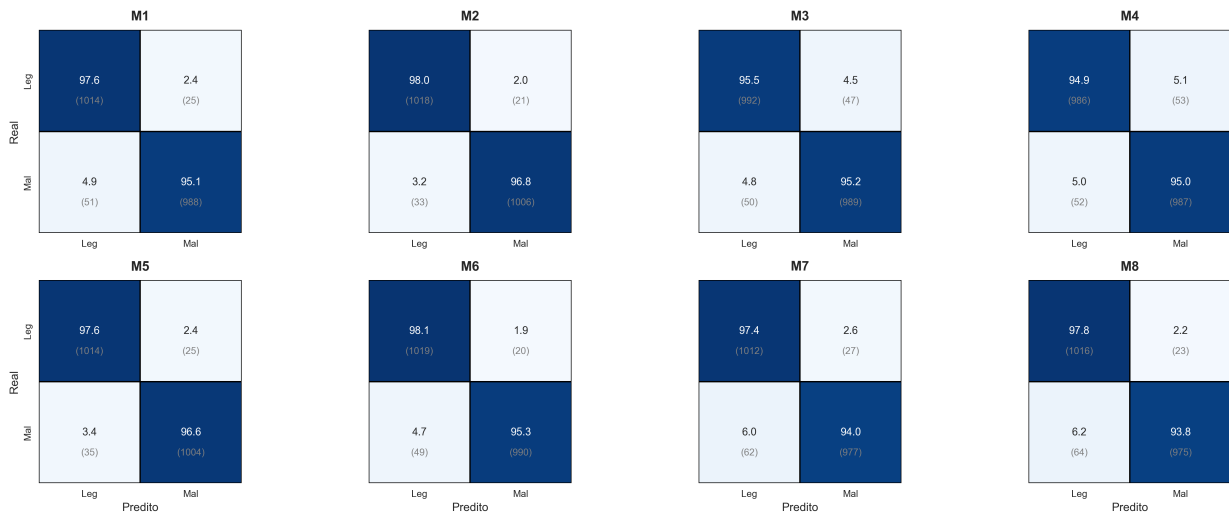


Figura 4: Matrizes de confusão - Oversampling (F1 médio=0.961).

Matrizes de Confusão - Undersampling (Validação Cruzada 5-Fold)

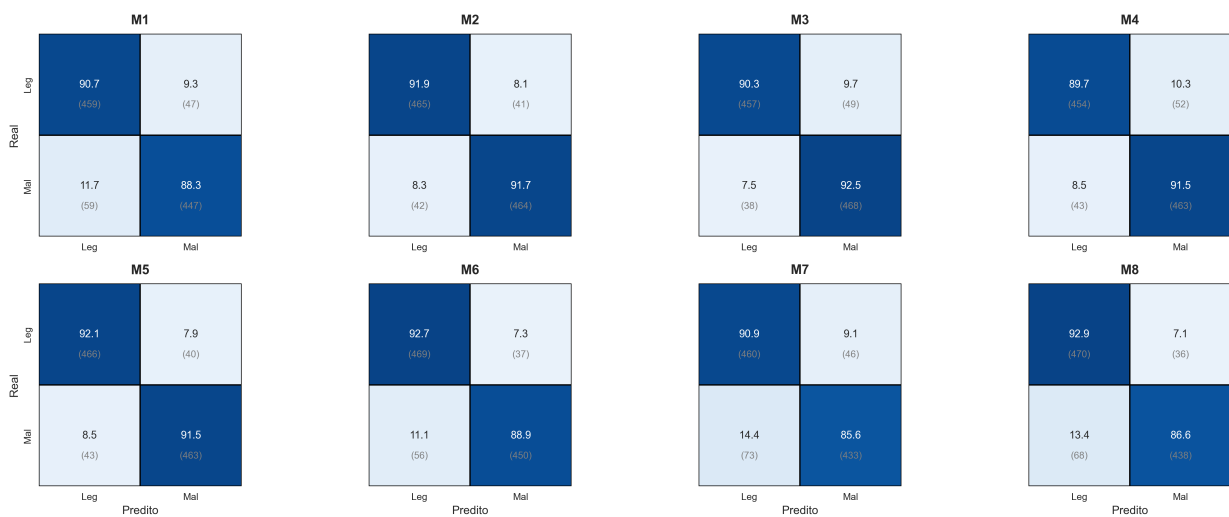


Figura 5: Matrizes de confusão - Undersampling (F1 médio=0.907)

Matrizes de Confusão - Desbalanceado (Validação Cruzada 5-Fold)

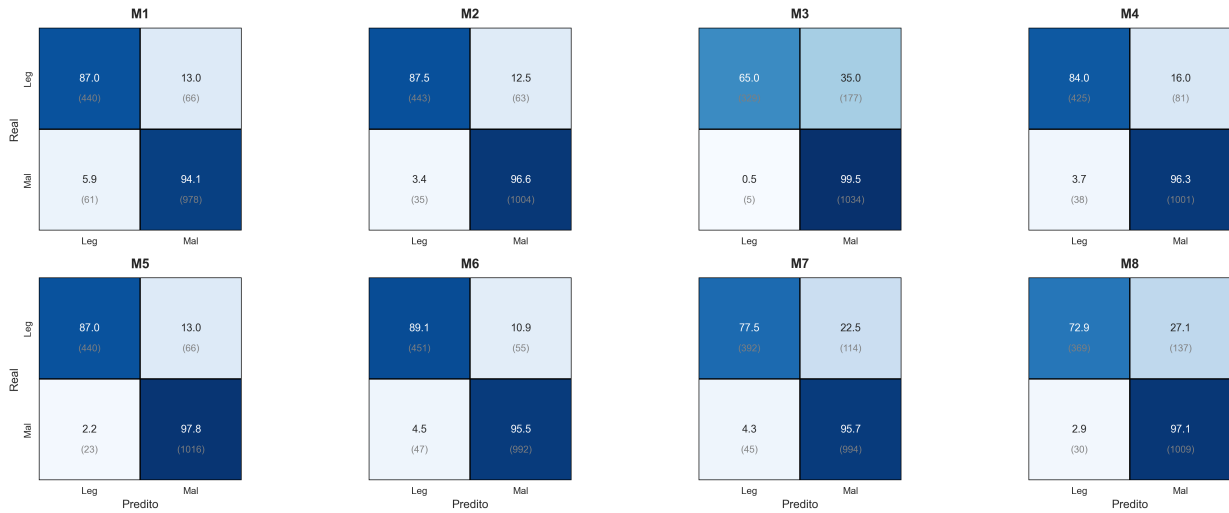


Figura 6: Matrizes de confusão - Desbalanceado (F1 médio=0.910)

I.1.3 Estabilidade das Métricas por Fold - BERT

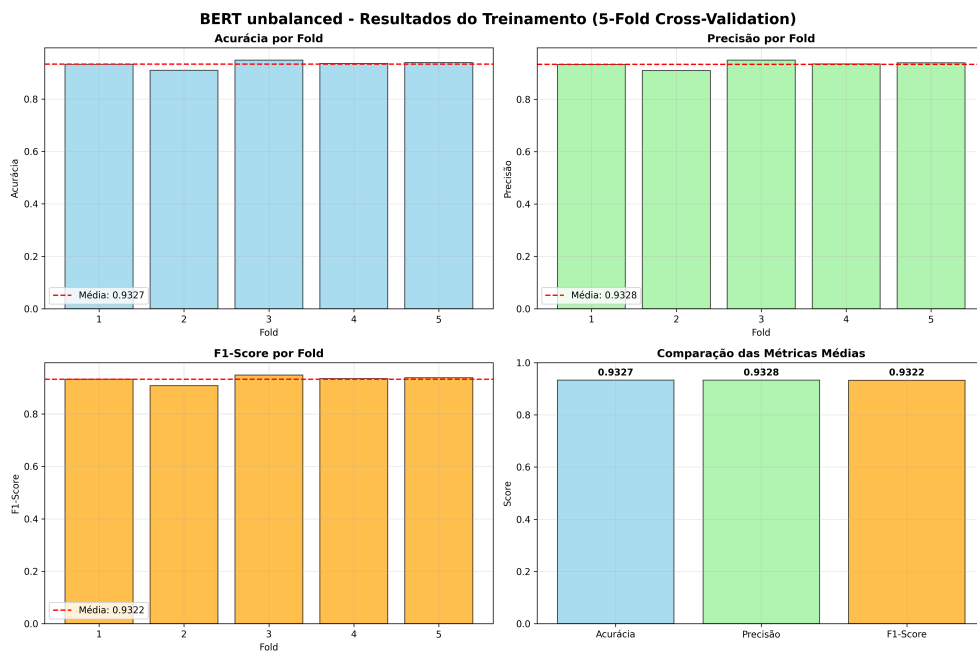


Figura 7: Matriz de confusão do modelo BERT treinado com dataset desbalanceado (F1-score médio = 0.917).

