Detecção de Sites Fraudulentos em Tempo Real: uma Extensão Chrome com Machine Learning

Arthur Henrique Secundino Ferreira
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação (DCC)
Belo Horizonte, Brasil
arthur.ferreira@dcc.ufmg.br

Prof. Antônio Alfredo Ferreira Loureiro
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação (DCC)
Belo Horizonte, Brasil
loureiro@dcc.ufmg.br

***Resumo*—Sites fraudulentos, páginas de phishing, lojas falsas e portais de impersonação, representam uma ameaça crescente à segurança de usuários comuns da Internet. Este trabalho apresenta uma extensão para o navegador Google Chrome capaz de detectar esses sites em tempo real, a partir de um modelo de Machine Learning treinado sobre 45 features extraídas da URL, do conteúdo HTML e da estrutura da página visitada. Foi construído um dataset próprio de 2.204 URLs reais rotuladas (1.312 fraudulentas, 892 legítimas), e o modelo de Regressão Logística implantado na extensão obteve, em um conjunto de teste held-out de 440 URLs nunca vistas no treino, F1-score de 97,7%, precisão de 97,0% e recall de 98,5%. Algoritmos adicionais (Random Forest, SVM e kNN) foram treinados sobre o mesmo dataset e o mesmo split para comparação direta, seguindo a metodologia de trabalhos relacionados, com o Random Forest atingindo o melhor desempenho (F1 = 99,6%). Os quatro algoritmos foram também avaliados com validação cruzada estratificada (k=5), que confirma a estabilidade dessas métricas entre diferentes divisões dos dados (desvio-padrão de F1 entre 0,4 e 0,8 ponto percentual para três dos quatro algoritmos) e revela que o número de split único do Random Forest era moderadamente otimista. Discutimos, no entanto, limitações relevantes desses resultados: o dataset utilizado é cerca de cinco vezes menor que o de bases de referência da literatura (como a UCI Phishing Websites, com aproximadamente 11.055 instâncias), o que sugere cautela na interpretação das métricas elevadas obtidas como medida definitiva de desempenho em produção.**

***Index Terms*—detecção de phishing, machine learning, extensão de navegador, regressão logística, segurança web**

I. Introdução

O crescimento da presença digital tem sido acompanhado por um aumento expressivo no número de sites fraudulentos, páginas de phishing, lojas falsas, portais de suporte técnico fraudulento e sites que imitam instituições legítimas. No Brasil, o volume de tentativas de phishing cresce consistentemente, e a sofisticação dessas páginas torna sua identificação cada vez mais difícil para o usuário comum.

Mecanismos tradicionais de defesa, como listas negras mantidas por provedores de navegador e serviços de reputação de domínio, dependem de relatos prévios para classificar uma URL como maliciosa, o que cria uma janela de exposição entre a criação de uma página fraudulenta e sua efetiva inclusão nessas listas. Campanhas de phishing modernas exploram justamente essa janela, utilizando domínios de curta duração, registrados e descartados em poucos dias ou horas, e técnicas de ofuscação de URL para evitar a correspondência exata com entradas já catalogadas. Essa limitação motiva abordagens baseadas em aprendizado de máquina, capazes de generalizar a partir de características estruturais da página em vez de depender exclusivamente do histórico de relatos, permitindo identificar páginas fraudulentas mesmo quando o domínio específico nunca foi visto anteriormente.

Este trabalho apresenta uma extensão para o navegador Google Chrome capaz de detectar sites fraudulentos em tempo real, utilizando um modelo de Machine Learning treinado sobre 45 features extraídas de URL, conteúdo HTML e estrutura da página. Diferentemente de abordagens baseadas em listas negras estáticas, a inferência ocorre localmente no dispositivo do usuário, a partir de características observáveis da própria página, sem dependência de consultas a serviços externos em tempo de navegação.

As principais contribuições deste trabalho são: (i) a construção de um dataset próprio de 2.204 URLs reais rotuladas, combinando fontes públicas de phishing e amostras curadas de sites legítimos; (ii) a implementação completa de um pipeline de extração de 45 features e treinamento de modelo, com comparação direta entre quatro algoritmos de classificação (Regressão Logística, Random Forest, SVM e kNN) sob a mesma metodologia de avaliação; e (iii) a integração do modelo treinado a uma extensão funcional para o navegador Chrome, validada tanto quantitativamente quanto em testes manuais com sites reais.

Os três módulos planejados foram implementados e validados com dados reais: (1) construção de um dataset próprio de URLs reais rotuladas; (2) treinamento e seleção do modelo de classificação; e (3) implementação da extensão Chrome com inferência local. O restante deste artigo está organizado da seguinte forma: a Seção II apresenta conceitos fundamentais e trabalhos relacionados; a Seção III descreve a metodologia executada; a Seção IV apresenta os resultados obtidos, incluindo a importância relativa das features extraídas e uma discussão de suas limitações; e a Seção V apresenta as conclusões do trabalho e propostas de trabalhos futuros. O Apêndice A complementa o texto com a lista completa das 45 features utilizadas, categorizadas por origem.

II. Conceitos Fundamentais e Trabalhos Relacionados

A. Detecção de Fraude como Problema de Classificação

A detecção de sites fraudulentos é tratada na literatura como um problema de classificação binária supervisionada,

no qual se busca distinguir páginas legítimas de páginas maliciosas a partir de um conjunto de características observáveis. Trabalhos como os de Sahingoz *et al.* [1] e Mohammad *et al.* [2] demonstram que features extraídas da estrutura da URL e do conteúdo HTML são altamente discriminatórias para essa tarefa, atingindo acurácias superiores a 95% em datasets balanceados, resultado consistente com o observado neste trabalho (Seção IV).

B. Regressão Logística e Restrições de Implantação

Entre os algoritmos de Machine Learning aplicados ao problema, a Regressão Logística se destaca por sua interpretabilidade e viabilidade de execução em ambientes com restrições computacionais, como navegadores web: o modelo se reduz a um produto escalar seguido de uma função sigmoide, permitindo que os pesos sejam incorporados diretamente em JavaScript puro, sem dependências externas. Random Forest e SVM são alternativas frequentemente avaliadas na literatura, com maior capacidade expressiva ao custo de maior complexidade, exigindo, por exemplo, a serialização de centenas de árvores ou de vetores de suporte, o que é inviável para um modelo embutido em uma extensão de navegador. Esse foi o critério decisivo para a escolha da Regressão Logística como modelo de implantação, mesmo quando outros algoritmos avaliados (Seção IV-C) apresentam métricas superiores.

No que se refere à infraestrutura técnica, a plataforma Chrome Extensions com Manifest V3 [9] define a arquitetura adotada, separando responsabilidades entre *Content Scripts* (extração de dados da página), *Background Scripts* (inferência do modelo) e *Popup UI* (apresentação dos resultados). Essa separação garante que todo o processamento ocorra no dispositivo do usuário, sem envio de dados a servidores externos.

C. Trabalhos Relacionados

Como trabalho relacionado mais próximo, destaca-se o NoPhish [5], extensão Chrome para detecção de phishing que compara Random Forest, SVM e kNN sobre o dataset UCI Phishing Websites ($\sim$11.055 instâncias, 30 features baseadas em WHOIS e Alexa Rank), com o Random Forest vencendo (precisão = 92,1%, recall = 97,0%). Reproduzir o dataset original não é viável atualmente, pois o Alexa Rank foi descontinuado em 2022 e o acesso a dados de WHOIS está cada vez mais restrito por regulações como LGPD/GDPR. Por isso, a comparação realizada neste trabalho (Seção IV-C) treina os mesmos algoritmos sobre um dataset próprio (45 features de URL, conteúdo HTML e estrutura), o que constitui uma comparação metodológica legítima, mesma família de algoritmos, mesmo split e mesmas métricas, embora não seja uma réplica exata do experimento original. Fontes de dados públicas como PhishTank [6], OpenPhish [7] e URLhaus [8] forneceram as listas de URLs maliciosas utilizadas na construção do dataset.

D. Síntese e Posicionamento desta Pesquisa

Os trabalhos relacionados apresentados compartilham a premissa de que páginas fraudulentas exibem padrões estruturais e lexicais suficientemente distintos de páginas legítimas para viabilizar classificação automática, mas diferem na granularidade das features utilizadas e no contexto de implantação. Garera *et al.* [4] propôs uma das primeiras abordagens baseadas em features heurísticas extraídas exclusivamente da URL, antecipando a estratégia de priorizar features de baixo custo computacional adotada neste trabalho. Abdelhamid *et al.* [3] e Mohammad *et al.* [2] demonstraram, sobre datasets de características semelhantes, que diferentes famílias de algoritmos (classificação associativa, redes neurais *self-structuring*) obtêm desempenho competitivo sobre o mesmo conjunto de features, padrão também observado na comparação de algoritmos deste trabalho (Seção IV-C), na qual Regressão Logística, Random Forest, SVM e kNN atingem F1-scores na mesma faixa (96,3% a 99,6%) quando treinados sobre o mesmo conjunto de 45 features e o mesmo *split*. Em relação ao NoPhish [5], a principal diferença metodológica deste trabalho é a origem das features: enquanto o NoPhish depende de informações externas à página (WHOIS, Alexa Rank), este trabalho restringe-se a características extraíveis localmente a partir da própria URL e do HTML, sem dependência de serviços de terceiros em tempo de inferência, uma escolha de design que prioriza a viabilidade de execução em uma extensão de navegador sobre a riqueza informacional de fontes externas.

III. Metodologia Executada

O projeto manteve a estrutura de dois módulos: a extensão Chrome, responsável pela extração de features e pela inferência em tempo real, e o módulo Python, responsável pelo treinamento e validação do modelo.

A. Construção do Dataset

Foi construído um dataset próprio de URL reais rotuladas, combinando fontes públicas de phishing (PhishTank, OpenPhish, URLhaus) e URLs legítimas de sites populares verificados, com extração das 45 features definidas na arquitetura do projeto (14 de URL, 18 de conteúdo HTML e 13 de estrutura de página). O dataset inicial continha 665 URLs reais (350 phishing / 315 legítimas). Ao longo do trabalho, o dataset foi expandido para 2.204 URLs reais (1.312 phishing / 892 legítimas) por meio de coleta adicional, ampliando significativamente a base de treino e teste.

Do total de 2.204 URLs, a extração de conteúdo HTML foi bem-sucedida para 1.329 (60,3%); para as 875 URLs restantes (39,7%), das quais 609 (69,6%) são páginas de phishing, provavelmente já removidas do ar no momento da coleta, não foi possível obter o HTML, mesmo com o uso de um mecanismo de *fallback* via Wayback Machine, que tenta recuperar uma versão arquivada da página quando a URL original está inacessível. Para essas URLs, as 31 features de conteúdo e estrutura foram preenchidas com valor zero, e a classificação do modelo passa a depender exclusivamente das 14 features de URL nesses casos. Essa característica do dataset é revisitada na Seção IV-F, ao quantificar a contribuição relativa de cada categoria de features, e listada como limitação adicional na Seção IV-G.

B. Treinamento e Seleção do Modelo

O pipeline de treinamento foi implementado em Python com NumPy e scikit-learn [10], [11]. A avaliação principal reportada neste artigo (Tabelas I, III e VI) utiliza uma divisão estratificada única de treino/teste (80%/20%, seed fixa = 42), replicada de forma idêntica para todos os algoritmos comparados. Para reduzir a sensibilidade dessa avaliação a uma divisão específica dos dados, a Seção IV-D complementa esses resultados com validação cruzada estratificada (k=5) para os quatro algoritmos. A Regressão Logística foi escolhida para implantação por sua interpretabilidade e viabilidade de execução em JavaScript puro (Seção II-B); seus pesos são exportados automaticamente para a extensão.

O gerador de páginas de phishing sintéticas (usado apenas para ampliar o conjunto de treino, nunca o teste) constrói URLs e estruturas de página artificiais combinando padrões documentados na literatura de phishing: palavras associadas a engenharia social (`secure`, `verify`, `confirm`, `account`, entre outras), hifenização de marcas conhecidas em subdomínios ou caminhos (por exemplo, `secure-marca.dominio.tld/account/verify`), formulários de login ou pagamento simulados, e ausência de elementos tipicamente presentes em páginas legítimas (rodapé institucional, menu de navegação, meta tags de SEO). Esse processo gera amostras positivas adicionais com padrões plausíveis de phishing sem depender exclusivamente da disponibilidade de novas páginas reais maliciosas, que tendem a ser removidas do ar rapidamente.

Um cuidado metodológico relevante foi a separação estrita entre dados reais e dados sintéticos: páginas de phishing sintéticas, geradas programaticamente para enriquecer o treino, foram usadas apenas no conjunto de treino, nunca no conjunto de teste. Uma avaliação inicial que misturava amostras sintéticas no teste indicava F1-score de até 98,3%, mas essa estimativa foi descartada por ser metodologicamente inflada (o modelo sendo testado em padrões artificiais que ele próprio ajudou a gerar). Todas as métricas reportadas na Seção IV usam exclusivamente URLs reais no conjunto de teste.

C. Implementação da Extensão Chrome

A extensão foi implementada seguindo a arquitetura Manifest V3, com três componentes: o *Content Script* extrai as 45 features em tempo real a partir do DOM e da URL de qualquer página visitada; o *Background Script* executa a inferência com o modelo de Regressão Logística exportado para JavaScript, calculando a probabilidade de fraude; e a *Popup UI* exibe o resultado por meio de *badges* coloridos (verde/amarelo/vermelho) e alertas para sites classificados como de alto risco (*threshold* de 70% de probabilidade). Durante os testes, foi identificado e corrigido um desvio de escopo: a interface e a lógica de análise estavam restritas a páginas com aparência de e-commerce, quando o modelo treinado já generaliza para qualquer tipo de site, a extensão foi ajustada para analisar qualquer página visitada.

D. Validação em Cenários Reais

A validação quantitativa foi realizada com o conjunto de teste *held-out* (440 URLs reais, nunca usadas no treino), reportada na Seção IV. A validação qualitativa consistiu em testes manuais da extensão instalada em sites reais, legítimos e suspeitos, verificando a consistência dos alertas e a usabilidade da interface, processo que revelou e levou à correção de dois defeitos: a restrição indevida a páginas de e-commerce e uma falha na comunicação entre componentes da extensão que fazia o *popup* exibir incorretamente "Página não analisada" mesmo após uma análise bem-sucedida.

IV. Resultados e Discussão

A. Evolução do Dataset e do Modelo

A expansão do dataset de 665 para 2.204 URLs reais produziu uma melhora honesta e mensurável no desempenho do modelo implantado, mantendo a mesma metodologia de avaliação (teste exclusivamente em dados reais, nunca vistos no treino), como ilustra a Figura 1.

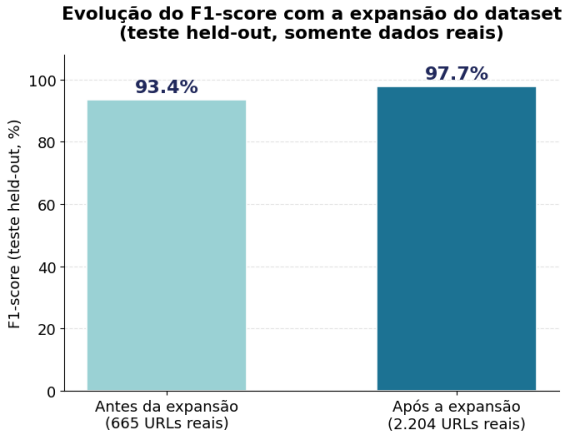

Fig. 1. Evolução do F1-score com a expansão do dataset (teste *held-out*, somente dados reais).

O ganho de aproximadamente 4,3 pontos percentuais de F1-score (93,4% $\rightarrow$ 97,7%) é consistente com a expectativa da literatura de que mais dados reais de treino tendem a melhorar a generalização do modelo, e reforça a decisão metodológica de priorizar a coleta de dados reais sobre o aumento artificial do dataset com páginas sintéticas. Ainda assim, esse ganho real e mensurável reforça, e não contradiz, a cautela da Seção IV-G: o salto expressivo de desempenho com um aumento relativamente modesto no volume de dados (665 $\rightarrow$ 2.204 URLs) sugere que o modelo ainda é sensível ao tamanho e à composição específica do dataset, e que novas expansões futuras provavelmente continuariam a alterar as métricas observadas.

B. Desempenho do Modelo Implantado

O modelo de Regressão Logística implantado na extensão (45 features, treinado e testado em 2.204 URLs reais, *split* estratificado 80/20) obteve os resultados da Tabela I no conjunto de teste *held-out* (440 URLs).

TABLE I
DESEMPENHO DO MODELO IMPLANTADO (REGRESSÃO LOGÍSTICA)

Acurácia	Precisão	Recall	F1-score
97,3%	97,0%	98,5%	**97,7%**

TABLE II
MATRIZ DE CONFUSÃO — CONJUNTO DE TESTE (N=440)

	Pred.: Fraude	Pred.: Legítimo
Real: Fraude	TP = 258	FN = 4
Real: Legítimo	FP = 8	TN = 170

A matriz de confusão (Tabela II) detalha os erros do modelo no conjunto de teste. Dos 440 casos, o modelo classificou corretamente 428 (97,3%), com 8 falsos positivos (sites legítimos classificados como fraude) e apenas 4 falsos negativos (sites fraudulentos não detectados), um resultado satisfatório considerando que o custo de um falso negativo (usuário exposto a fraude) é geralmente mais sensível do que o de um falso positivo (alerta desnecessário).

A curva ROC do modelo, com AUC = 0,996, confirma a forte separação entre as classes (Figura 2).

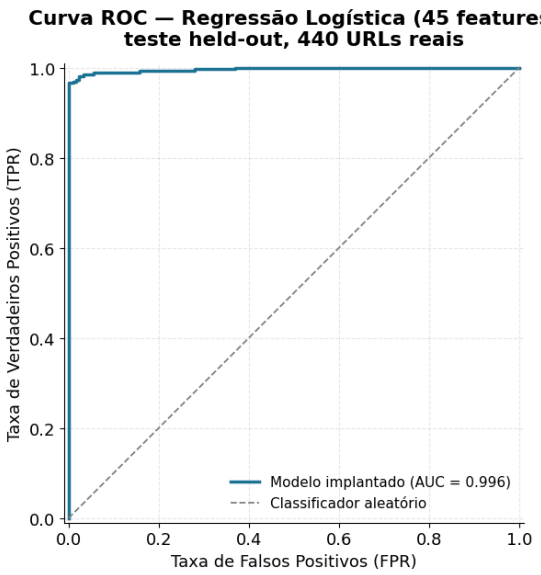

Fig. 2. Curva ROC do modelo implantado (Regressão Logística, 45 features).

C. Comparação com Algoritmos Relacionados

Seguindo a metodologia de comparação com a literatura (NoPhish, Seção II-C), Random Forest, SVM e kNN foram treinados e avaliados sobre o mesmo dataset final (2.204 URLs reais) e o mesmo *split* estratificado usado para a Regressão Logística, garantindo uma comparação direta e justa (Figura 3 e Tabela III).

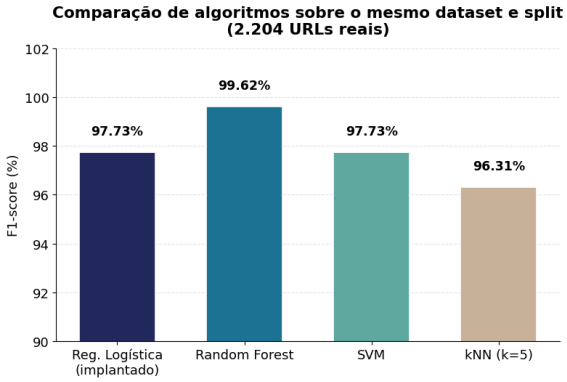

Fig. 3. Comparação de algoritmos sobre o mesmo dataset e *split* (2.204 URLs reais).

O Random Forest obteve o melhor desempenho entre os quatro algoritmos (F1 = 99,6%, zero falsos positivos), superando a Regressão Logística por quase 2 pontos percentuais. Ainda assim, a Regressão Logística foi mantida como modelo de implantação: o Random Forest treinado exigiria serializar centenas de árvores de decisão para execução em JavaScript, o que é inviável em termos de tamanho de pacote e tempo de inferência dentro de uma extensão de navegador (Seção II-B). O resultado confirma a literatura, o Random Forest tende a superar a Regressão Logística em métricas brutas, mas valida a escolha de implantação feita por critérios de engenharia, não apenas estatísticos. Essa comparação de split único é complementada, na próxima seção, por validação cruzada, que avalia a estabilidade dessas métricas entre diferentes divisões dos dados.

D. Validação Cruzada (k=5)

Para reduzir a sensibilidade da avaliação anterior a uma única divisão treino/teste, os quatro algoritmos comparados na Seção IV-C foram reavaliados com validação cruzada estratificada (k=5) sobre o mesmo dataset (2.204 URLs reais). Para a Regressão Logística, foi reutilizada exatamente a mesma implementação (gradiente descendente em NumPy puro) do modelo implantado; para Random Forest, SVM e kNN, foi utilizada a implementação do scikit-learn [10], [11], com os mesmos hiperparâmetros da Seção IV-C. Em cada fold, o escalonamento de features (z-score) foi ajustado exclusivamente nos dados de treino daquele fold, evitando vazamento de informação do conjunto de teste para o de treino. A Tabela IV reporta a média e o desvio-padrão (entre os 5 folds) de cada métrica.

Os resultados de validação cruzada são altamente consistentes com os da Tabela III (avaliação por split único) para três dos quatro algoritmos: a Regressão Logística (F1 = 97,8% $\pm$ 0,6 p.p. na validação cruzada vs. 97,7% no split único) e o SVM (97,7% $\pm$ 0,8 p.p. vs. 97,7%) apresentam médias praticamente idênticas ao valor do split único, com desvio-padrão entre folds inferior a 1 ponto percentual, e evidência de que essas estimativas são estáveis e não dependem de uma divisão de teste particularmente favorável. O kNN também

TABLE III
COMPARAÇÃO DE ALGORITMOS (MESMO DATASET E *split*)

Algoritmo	**Precisão**	**Recall**	**F1**	**ROC-AUC**
Reg. Logística (impl.)	97,0%	98,5%	97,7%	99,6%
Random Forest	100,0%	99,2%	99,6%	99,98%
SVM	97,0%	98,5%	97,7%	99,7%
kNN (k=5)	98,0%	94,7%	96,3%	98,3%

TABLE IV
VALIDAÇÃO CRUZADA ESTRATIFICADA (K=5) — MÉDIA ± DESVIO-PADRÃO ENTRE FOLDS

Algoritmo	**Precisão**	**Recall**	**F1**	**ROC-AUC**
Reg. Logística (impl.)	98,1% ± 0,7 p.p.	97,5% ± 1,0 p.p.	97,8% ± 0,6 p.p.	99,5% ± 0,2 p.p.
Random Forest	98,6% ± 1,0 p.p.	98,3% ± 0,8 p.p.	98,4% ± 0,4 p.p.	99,8% ± 0,1 p.p.
SVM	97,3% ± 0,6 p.p.	98,2% ± 1,2 p.p.	97,7% ± 0,8 p.p.	99,6% ± 0,3 p.p.
kNN (k=5)	96,4% ± 1,0 p.p.	94,6% ± 0,8 p.p.	95,5% ± 0,7 p.p.	97,8% ± 0,6 p.p.

se mantém relativamente próximo (95,5% ± 0,7 p.p. vs. 96,3%). Já o Random Forest apresenta a maior discrepância entre os dois métodos de avaliação: F1 médio de 98,4% ± 0,4 p.p. na validação cruzada, contra 99,6% no split único, uma diferença de 1,2 ponto percentual que sugere que o split original de teste, no qual o Random Forest não comete nenhum falso positivo, foi particularmente favorável a esse algoritmo; em média, na validação cruzada, ele comete entre 0 e 8 falsos positivos por fold. Esse resultado reforça a importância metodológica da validação cruzada: a métrica de split único, embora corretamente calculada em dados nunca vistos no treino, pode mascarar variação real de desempenho conforme a amostra específica usada no teste, um risco que a média entre folds mitiga, sem eliminar por completo, já que o dataset usado em cada fold ainda compartilha as mesmas limitações de tamanho e diversidade discutidas na Seção IV-G.

E. Importância das Features

Como a Regressão Logística opera sobre features padronizadas (z-score, usando média e desvio-padrão calculados no conjunto de treino), a magnitude do coeficiente de cada feature é uma medida comparável de sua contribuição relativa para a decisão do modelo. A Tabela V lista as 12 features com maior magnitude de coeficiente, dentre as 45 utilizadas.

TABLE V
AS 12 FEATURES COM MAIOR PESO (VALOR ABSOLUTO) NA REGRESSÃO LOGÍSTICA

Feature	**Peso**
`hostname_length`	+1,923
`tld`	–1,774
`is_https`	–1,690
`url_length`	+1,312
`has_numbers`	+0,820
`hyphen_count`	+0,804
`path_length`	+0,660
`subdomain_count`	–0,655
`dot_count`	–0,649
`has_social`	–0,623
`has_products`	–0,606
`internal_links`	–0,476

As três features de maior peso pertencem à categoria de URL (`hostname_length`, `tld`, `is_https`), reforçando a decisão de design de priorizar features de baixo custo computacional (Seção II-B): mesmo sem qualquer acesso ao conteúdo HTML da página, essas três features sozinhas concentram boa parte do sinal discriminativo do modelo. O peso negativo de `is_https` é intuitivo, e páginas servidas sem HTTPS estão mais associadas a fraude. Já o peso de `tld` exige uma ressalva metodológica: essa feature é codificada como um inteiro ordinal (0 para domínios não mapeados, 1 para `.com`, 2 para `.br`, 3 para `.net`, 4 para `.org`), uma codificação que impõe implicitamente uma relação de ordem entre categorias que não existe semanticamente, um problema conhecido de codificação categórica em modelos lineares, e uma limitação de implementação a ser corrigida em trabalho futuro (idealmente via *one-hot encoding*). Os pesos negativos de `subdomain_count` e `dot_count` também merecem cautela interpretativa: as duas features são estruturalmente correlacionadas (ambas derivam da segmentação do hostname por pontos), e multicolinearidade entre features correlacionadas pode produzir coeficientes individuais com sinais pouco intuitivos, mesmo quando o efeito conjunto das features é coerente, fenômeno bem documentado em regressão linear/logística, que não invalida o modelo, mas limita a interpretação isolada de cada coeficiente.

F. Contribuição das Categorias de Features (URL vs. Conteúdo/Estrutura)

Para quantificar a contribuição das features de conteúdo HTML e estrutura (31 das 45 features) além das features de URL (14 features), foi treinado e avaliado um segundo modelo de Regressão Logística utilizando exclusivamente as 14 features de URL, sob o mesmo *split* estratificado (Tabela VI).

O ganho do conjunto completo sobre o modelo URL-only é modesto em F1 (+0,22 ponto percentual: 97,51% $\rightarrow$ 97,73%), mas mais expressivo em ROC-AUC (99,33% $\rightarrow$ 99,63%) e, principalmente, em recall (97,33% $\rightarrow$ 98,47%): o modelo completo comete três falsos negativos a menos (7 $\rightarrow$ 4 em 440 casos de teste), ao custo de dois falsos positivos adicionais (6 $\rightarrow$ 8). Ou seja, as features de conteúdo e estrutura contribuem principalmente para reduzir a taxa de fraudes não detectadas, métrica mais sensível neste contexto de aplicação (Seção IV-B), em troca de uma pequena queda de precisão. O ganho relativamente contido também é explicado por uma limitação de cobertura do dataset: como discutido na Seção III-A, para 875 das 2.204 URLs (39,7%) não foi possível obter o HTML da página, e as 31 features de conteúdo/estrutura foram preenchidas com zero para esses casos, diluindo, para uma fração não desprezível do dataset, o sinal informativo que essas features poderiam oferecer caso o HTML estivesse disponível para todas as amostras.

G. Discussão e Limitações

Os resultados obtidos (F1 entre 96,3% e 99,6% no split único; 95,5% a 98,4% na validação cruzada) são elevados. No entanto, é importante destacar limitações metodológicas que pedem cautela na interpretação desses números, especialmente em comparação direta com a literatura:

- **Tamanho do dataset:** 2.204 URLs reais é uma base ainda consideravelmente menor que a usada em trabalhos relacionados, por exemplo, o dataset UCI Phishing Websites usado pelo NoPhish tem aproximadamente 11.055 instâncias, cerca de 5 vezes mais. Datasets menores tendem a produzir estimativas de desempenho mais otimistas e menos estáveis, com maior risco de o modelo capturar padrões específicos das fontes de coleta em vez de características genuinamente discriminativas de fraude em geral. A validação cruzada da Seção IV-D mitiga, mas não elimina esse risco: todos os folds são extraídos do mesmo dataset de 2.204 URLs, compartilhando as mesmas fontes de coleta e, portanto, os mesmos eventuais viéses sistemáticos dessas fontes.
- **Diversidade das fontes:** o dataset reúne URLs de um número limitado de fontes públicas; campanhas de phishing futuras, ou categorias de sites legítimos não representadas na coleta atual, podem não ser classificadas com a mesma precisão observada no conjunto de teste *held-out*.
- **Colinearidade entre features:** conforme discutido na Seção IV-E, algumas features de URL são estruturalmente correlacionadas entre si (por exemplo, `subdomain_count` e `dot_count`), o que pode produzir coeficientes individuais com sinais pouco intuitivos na Regressão Logística, mesmo sem comprometer a capacidade discriminativa agregada do modelo.

Em conjunto, esses fatores indicam que as métricas elevadas obtidas (97,7% de F1 para o modelo implantado em split único, 97,8% $\pm$ 0,6 p.p. em validação cruzada) devem ser interpretadas como evidência de que o modelo aprendeu padrões úteis nos dados disponíveis, e não como uma medida definitiva do desempenho esperado em produção, contra qualquer site da Internet. O conjunto de teste é *held-out* (nunca visto no treino), e a validação cruzada (Seção IV-D) confirma que essa estimativa não depende criticamente de uma divisão de teste específica, o que mitiga o risco de *overfitting* no sentido estrito do termo, mas não elimina o risco de viés otimista por dataset pequeno e pouco diverso, já discutido acima. Como trabalho futuro, recomenda-se: (i) ampliar continuamente o dataset com novas fontes; (ii) estender a validação cruzada para um número maior de folds (k=10) e reportar testes de significância estatística entre algoritmos; e (iii) testar o modelo contra URLs coletadas de fontes não usadas no treino, como forma de validação fora da distribuição (*out-of-distribution*).

V. Conclusão

Este trabalho desenvolveu uma extensão Chrome funcional, capaz de detectar sites fraudulentos de diferentes naturezas (phishing, lojas falsas, páginas de impersonação) em tempo real, com inferência totalmente local. Foi implementado e treinado um modelo de Machine Learning sobre um dataset próprio de 2.204 URLs reais, com avaliação honesta em dados nunca vistos no treino, e os resultados foram comparados com algoritmos da literatura (Random Forest, SVM, kNN), seguindo a metodologia de trabalhos relacionados como o NoPhish.

O resultado mais relevante do ponto de vista metodológico não foi apenas o F1-score final (97,7% para o modelo implantado), mas o processo rigoroso que levou a ele: a identificação e correção de uma avaliação inicial inflada por dados sintéticos no conjunto de teste; a expansão do dataset com dados reais adicionais, produzindo uma melhora genuína e mensurável; a validação cruzada (k=5, Seção IV-D), confirmando a estabilidade dessa melhora entre diferentes divisões dos dados para três dos quatro algoritmos comparados; e o reconhecimento explícito das limitações de escala que ainda acompanham esses números (Seção IV-G).

A análise de importância de features (Seção IV-E) revelou que a maior parte do sinal discriminativo do modelo implantado provém de um pequeno número de features de URL de baixo custo computacional, e o estudo de ablação (Seção IV-F) quantificou que as features de conteúdo HTML e estrutura, embora contribuam de forma mensurável, sobretudo para reduzir falsos negativos, têm seu impacto parcialmente diluído pela cobertura incompleta de HTML no dataset atual. Esse achado tem implicação prática direta para trabalhos futuros: ampliar a taxa de sucesso na obtenção de HTML (por exemplo, com mecanismos de *fallback* mais robustos que o Wayback Machine) é uma direção de melhoria potencialmente

TABLE VI
COMPARAÇÃO: APENAS FEATURES DE URL (14) VS. CONJUNTO COMPLETO (45)

Modelo	**Precisão**	**Recall**	**F1**	**ROC-AUC**
URL apenas (14)	97,7%	97,3%	97,5%	99,3%
Completo (45)	97,0%	98,5%	97,7%	99,6%

TABLE VII
FEATURES DE URL (14)

`url_length`	`dot_count`
`subdomain_count`	`has_at`
`has_numbers`	`double_slash`
`numbers_count`	`path_length`
`is_https`	`params_count`
`hostname_length`	`tld`
`hyphen_count`	`is_ip`

TABLE VIII
FEATURES DE CONTEÚDO HTML (18)

`text_length`	`no_alt_imgs`
`has_cart`	`external_links`
`has_checkout`	`internal_links`
`has_products`	`has_social`
`has_search`	`has_contact`
`form_count`	`has_phone`
`has_login`	`has_email`
`has_payment`	`has_lorem`
`image_count`	`text_ratio`

TABLE IX
FEATURES DE ESTRUTURA DE PÁGINA (13)

`has_meta_desc`	`iframe_count`
`has_meta_keywords`	`has_header`
`has_meta_author`	`has_footer`
`has_og`	`has_nav`
`has_favicon`	`total_elements`
`is_secure`	`link_density`
`external_scripts`	

mais custo-efetiva do que simplesmente aumentar o volume total de URLs coletadas.

Como continuidade natural deste trabalho, destacam-se a incorporação de features adicionais (certificado SSL, dados de WHOIS, quando disponíveis), a implementação de um sistema de feedback do usuário, a ampliação contínua do dataset, a correção da codificação ordinal da feature `tld` e a extensão da validação cruzada para k=10 com relato de testes de significância estatística entre os algoritmos comparados.

AGRADECIMENTOS

O autor agradece ao orientador, Prof. Antonio Alfredo Ferreira Loureiro, pela orientação ao longo do desenvolvimento deste trabalho.

APPENDIX

APÊNDICE A: LISTA COMPLETA DAS 45 FEATURES

Esta seção lista as 45 features utilizadas pelo modelo implantado, organizadas pelas três categorias definidas na arquitetura do projeto (Seção III-A), com os nomes exatamente como aparecem no pipeline de exportação para JavaScript (`weights_v3_real.json`).

Vale notar que `is_secure` é distinta de `is_https`: a correlação entre ambas no dataset final é de apenas 0,495, indicando que capturam aspectos parcialmente distintos da segurança da conexão. Diferentemente das demais features de estrutura, `is_secure` não é gerada diretamente pela função `extract_structure_features` do módulo de extração, tendo sido adicionada em uma etapa posterior do pipeline de construção do dataset, um ponto de dívida técnica a ser consolidado e documentado em trabalho futuro.

REFERENCES

[1] O. K. Sahingoz *et al.*, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.
[2] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443–458, 2014.
[3] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based on associative classification data mining," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948–5959, 2014.
[4] S. Garera *et al.*, "A framework for detection and measurement of phishing attacks," in *Proc. 2007 ACM Workshop on Recurring Malcode*, 2007.
[5] L. Thaçi, A. Halili, K. Vishi, and B. Rexha, "NoPhish: Efficient Chrome Extension for Phishing Detection Using Machine Learning Techniques," *arXiv:2409.10547*, 2024.
[6] PhishTank, "PhishTank Developer Information," 2024. [Online]. Available: https://www.phishtank.com/developer_info.php
[7] OpenPhish, "OpenPhish - Phishing Intelligence," 2024. [Online]. Available: https://openphish.com
[8] URLhaus, "URLhaus - Malware URL Exchange," 2024. [Online]. Available: https://urlhaus.abuse.ch
[9] Google, "Chrome Extensions - Manifest V3," 2024. [Online]. Available: https://developer.chrome.com/docs/extensions/mv3/intro/
[10] scikit-learn, "scikit-learn: Machine Learning in Python," 2024. [Online]. Available: https://scikit-learn.org
[11] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[12] FFW-282 Dataset, "Fake/Fraudulent Websites Dataset," 2023. [Online]. Available: https://github.com/ffwdataset/ffw-282