Phishing Guardian: Detecção de sites de *phishing* com *Machine Learning*

Bianca Domingos Guarizi[1], **Dalbert Matos Mascarenhas**[1], **Igor Monteiro Moraes**[2]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ)
Petrópolis, RJ, Brasil
[2] Laboratório MídiaCom, Universidade Federal Fluminense (UFF)
Niterói, RJ, Brasil

bianca.guarizi@aluno.cefet-rj.br, dalbert.mascarenhas@cefet-rj.br,
igor@ic.uff.br

Abstract. *Phishing remains one of the cyber threats with the greatest financial and social impact. This paper investigates the effectiveness of Machine Learning techniques in detecting malicious URLs, addressing shortcomings related to incomplete databases and systematic comparisons between algorithms. It uses a database of 50,261 URLs (55.5% malicious) collected from public sources and active scanning. The Random Forest, XGBoost and SVM algorithms are trained with cross-validation, with XGBoost achieving 99.51% accuracy. A tool was developed that contains the classifier and a browser extension that displays non-intrusive alerts to the user, in order to guarantee a good user experience.*

Resumo. *O phishing permanece como uma das ameaças cibernéticas de maior impacto financeiro e social. Este trabalho investiga a eficácia de técnicas de Machine Learning na detecção de URLs maliciosas, abordando lacunas relacionadas a bases de dados incompletas e comparações sistemáticas entre algoritmos. Utiliza-se uma base de dados de 50.261 URLs (55,5% maliciosas) coletadas de fontes públicas e varredura ativa. Os algoritmos Random Forest, XGBoost e SVM são treinados com validação cruzada, com o XGBoost alcançando 99,51% de acurácia. Foi desenvolvida uma ferramenta que contém o classificador e uma extensão de navegador que exibe alertas não intrusivos ao usuário, objetivando garantir uma boa experiência de utilização.*

1. Introdução

O *phishing* permanece como uma das ameaças cibernéticas de maior impacto financeiro e social. Relatórios de ameaças recentes apontam perdas globais de US$ 14,8 bilhões em 2024, um crescimento de 290% em relação a 2019 [Sullivan]. A rápida digitalização de serviços, acelerada pela pandemia de COVID-19, expôs uma nova massa de usuários pouco experientes e tornou o ecossistema ainda mais atrativo para agentes mal-intencionados [Alawida et al. 2022, Pranggono and Arabo 2021, Kamal et al. 2020].

Apesar de avanços em mecanismos de defesa, soluções baseadas em *blacklists* continuam dependentes de atualizações frequentes, tornando-se ineficazes contra ataques *zero-day* [Ahmad et al. 2023, Guo 2023]. Já abordagens puramente heurísticas, embora

detectem tentativas conhecidas, apresentam taxas de falsos positivos que afetam a experiência do usuário e oneram equipes de suporte. Nesse cenário, técnicas de *machine learning* (ML) emergem como alternativas promissoras por sua capacidade de aprender padrões complexos e adaptar-se rapidamente a táticas inéditas [Abu-Nimeh et al. 2007].

Embora a literatura recente apresente progressos importantes na detecção automática de *phishing*, foram observadas duas lacunas recorrentes: (i) dependência de bases que possuem grandes referências de páginas fraudulentas e poucas páginas legítimas, o que causa desbalanceamento da base e a criação de bases pequenas e (ii) falta de análise comparativa entre algoritmos leves e de alto desempenho sob as mesmas condições experimentais [Souza and Mascarenhas 2023, Correia and Pedrini 2020]. Diante disso, formulam-se as seguintes perguntas de pesquisa: **P1** – Qual algoritmo supervisionado oferece o melhor compromisso entre acurácia e tempo de resposta em execução local? **P2** – Quais grupos de atributos mais contribuem para reduzir falsos positivos na detecção de páginas fraudulentas? **P3** – Como garantir uma base de dados robusta e equilibrada?

Para responder a essas questões, conduziu-se um estudo quantitativo baseado em uma base de dados composta de 50.261 Uniform Resource Locator (URLs), com um total de 55,5% maliciosas e 44,5% legítimas, coletadas de fontes públicas e varredura ativa através de estratégias que garantem a coleta ágil de páginas maliciosas e um grande número de páginas web verdadeiramente legítimas. Incluindo o treinamento de 3 algoritmos diferentes: Random Forest, XGBoost e SVM, a fim de realizar um estudo comparativo entre estes. Além de um estudo abrangente para seleção de atributos específicos para otimizar a classificação.

Todos os datasets, código-fonte e a extensão de navegador são disponibilizados, podendo ser acessados em [Guarizi and Mascarenhas 2025], fomentando a reprodutibilidade e a adoção pela comunidade de pesquisa em segurança. Além de aferir desempenho preditivo, a ferramenta *Phishing Guardian* propõe uma integração prática com o usuário final através da extensão de navegador, capaz de identificar páginas em tempo real, mediando informações entre o classificador e o navegador do usuário, exibindo alertas não intrusivos.

As contribuições principais deste artigo são:

- Construção de uma base híbrida de grande porte, combinando fontes públicas (PhishTank, OpenPhish) e técnicas de enumeração para URLs legítimas.
- Avaliação sistemática de três algoritmos de ML com métricas de acurácia, precisão, recall e F1-score, identificando o XGBoost como a melhor opção (99,51% de acurácia).
- Implementação de um protótipo completo: modelo preditivo, servidor Flask e extensão para navegadores Google Chrome.

Por fim, o artigo está organizado da seguinte forma: a Seção 2 discute a ambientação tecnológica sobre *phishing*; a Seção 3 trata dos trabalhos relacionados; a Seção 4 detalha a ferramenta desenvolvida; a Seção 5 apresenta e analisa os resultados; a Seção 6 descreve a evolução do projeto; e a Seção 7 traz as conclusões e direções futuras.

2. Ambientação Tecnológica sobre *Phishing*

Ataques de *phishing* evoluíram além do tradicional e-mail e hoje incluem SMS (*smishing*), chamadas de voz (*vishing*), redes sociais, códigos QR e notificações

push [Al Saidat et al. 2024, Sharevski et al. 2022, Salloum et al. 2022, Kim et al. 2022]. Esse alargamento de superfície torna os filtros baseados em *blacklists* cada vez menos eficazes, pois dependem de atualização reativa e não cobrem URLs inéditas (*zero-day*), limitação já destacada na Introdução do artigo.

A popularização de *phishing kits* prontos, hospedagem *bullet-proof* e certificados *TLS* gratuitos reduziu as barreiras de entrada para agentes maliciosos [Castaño et al. 2023]. Em poucos minutos, um atacante clona uma página legítima, configura a coleta de credenciais e inicia redirecionamentos a partir de domínios recentemente registrados, explorando o intervalo entre registro e detecção [Bhattacharya et al. 2023].

No Brasil, a Lei Geral de Proteção de Dados (LGPD) impõe deveres de notificação de incidentes, enquanto a adoção massiva de meios de pagamento instantâneo (*PIX*) e mensageria móvel cria cenários férteis para engenharia social [Kyriazoglou 2024]. Portais de serviços públicos, sistemas universitários e aplicativos bancários transformaram-se em alvos frequentes, exigindo defesas que considerem idioma, gírias e marcas locais [Singh et al. 2024].

Os controles tradicionais incluem: (i) filtros heurísticos de código-fonte e (ii) bloqueio por *blacklists* de domínios. Entretanto, tais mecanismos apresentam limitações como altas taxas de falsos positivos e incapacidade de lidar com homógrafos ou encurtadores de URL [Alanezi 2021, Sheng et al. 2009]. Nos últimos anos, técnicas de ML ganharam destaque por capturar padrões léxicos e de rede de forma dinâmica, motivo pelo qual o presente trabalho compara três algoritmos (Random Forest, XGBoost e SVM) sobre um conjunto representativo de URLs.

Persistem desafios em: (a) melhoria das métricas garantindo a diminuição de falsos positivos, (b) adaptar as bases de dados para que sejam mais amplas e equilibradas entre páginas de fraude e legítimas, (c) elaborar uma ferramenta funcional e fluída para garantir uma boa usabilidade pelo usuário e (d) identificar os melhores algoritmos de ML para detecção de páginas fraudulentas. A solução proposta neste artigo direciona-se a esses desafios. Isso porque o presente trabalho compara diferentes modelos, selecionando o melhor para incorporá-lo em uma ferramenta que gera alertas diretamente no navegador do usuário, além de elaborar uma base de dados através de técnicas que garantem um melhor equilíbrio entre sites maliciosos e legítimos, conteúdo descrito na Seção 4.

3. Trabalhos Relacionados

Em sua pesquisa, [Safi and Singh 2023] apresenta uma análise abrangente sobre métodos de primeira geração baseados em *blacklists*, que dominaram as soluções de segurança na década de 2010. Os autores analisaram 80 trabalhos nesta área, concluindo que essas soluções apresentam eficácia limitada (60-75%) contra ataques recentes devido ao seu caráter reativo e à rápida obsolescência das listas de URLs maliciosas.

Em [Correia and Pedrini 2020] é abordada a terceira geração de soluções que utilizam técnicas de machine learning, com foco em algoritmos baseados em árvores de decisão. Os autores demonstraram a superioridade do Random Forest (95% de acurácia) sobre SVM e KNN em conjuntos de dados balanceados, utilizando uma base de dados de 7120 URLs, além de utilizarem 24 atributos, porém sem integração com sistemas de navegação reais.

No trabalho [Sadaf 2023], é apresentada uma comparação entre os classificadores XGBoost e CatBoost para detecção de *phishing*, utilizando dois conjuntos de dados distintos da University of California, Irvine (UCI). Os resultados mostraram que ambos os modelos superam classificadores tradicionais, com XGBoost apresentando um desempenho ligeiramente superior (96,79% contra 96,69% de acurácia) em um dataset contendo 2456 URLs e 30 atributos.

A análise realizada por [Fajar et al. 2024] investiga a detecção de URLs maliciosas através de análise de importância de features e técnicas de Explainable AI (XAI). Os autores compararam CatBoost, XGBoost e Explainable Boosting Machine (EBM), concluindo que XGBoost é mais eficiente em tempo de execução, enquanto CatBoost mantém alta acurácia mesmo com redução de features.

O trabalho realizado por [Souza and Mascarenhas 2023] executou a comparação entre 5 classificadores, sendo eles MLP, SVM, KNN, Árvore de decisão e *Random Forest*. Foi utilizada uma base de dados de 990 URLs, além de 25 atributos no total. Como resultado inicial, utilizando todos os atributos coletados pelos autores, o melhor desempenho foi obtido através do classificador SVM, com acurácia de 99,19%; além disso, testes foram realizados removendo atributos que proporcionavam mais tempo de resposta do algoritmo, desse modo, houve um melhor desempenho através do algoritmo *Random Forest* com 94,24%.

Este trabalho avança o estado da arte em três frentes principais: A utilização de uma base de dados significativamente maior comparada com os trabalhos relacionados apresentados (50.261 URLs); o refinamento da seleção de atributos, mantendo apenas as 13 features mais discriminativas (uma diminuição significativa comparada ao quantitativo visto nos trabalhos apresentados); e a implementação da ferramenta *Phishing Guardian*, uma solução completa, desde o modelo preditivo até a interface com o usuário final, resolvendo desafios práticos de latência e usabilidade não abordados em trabalhos anteriores.

Tabela 1. Comparação com trabalhos relacionados.

Autor	Base de dados	Atributos	Ferramenta proposta
Correia e Pedrini	7120 URLs	24	Nenhuma
Sadaf	2456 URLs	30	Nenhuma
Souza e Mascarenhas	990 URLs	25	Nenhuma
Presente trabalho	50261 URLs	13	*Phishing Guardian*

4. A Ferramenta Proposta *Phishing Guardian*

O *Phishing Guardian* consiste em uma ferramenta integrada que combina um modelo de aprendizado de máquina com uma extensão de navegador, destinada à identificação de páginas web fraudulentas utilizadas em ataques de *phishing*. A ferramenta foi projetada para analisar o tráfego do usuário em tempo real, coletando atributos específicos das URLs acessadas e classificando-as como legítimas ou maliciosas, com base em padrões previamente aprendidos. A abordagem adotada busca superar as limitações de métodos tradicionais, como *blacklists*, ao utilizar técnicas de aprendizado de máquina para detectar tanto ataques conhecidos quanto ataques *zero-day*.

4.1. Tecnologias Implementadas

A implementação da ferramenta proposta demandou a utilização de tecnologias específicas para cada componente, sendo detalhadas a seguir:

- Linguagem de Programação Python: Foi escolhida para o desenvolvimento dos classificadores. A biblioteca Scikit-learn foi utilizada para implementar os algoritmos escolhidos: Random Forest, XGBoost e SVM;
- Linguagem de Programação JavaScript: Empregada no desenvolvimento da extensão para navegador. A extensão tem a função de monitorar as URLs acessadas pelo usuário em tempo real e enviá-las para análise;
- Framework Flask: Para estabelecer a comunicação entre a extensão de navegador e o classificador em Python, foi utilizado o Flask, um microframework web leve e flexível. Um servidor local foi configurado na rota http://localhost:5000/receive_url, responsável por receber as URLs coletadas pela extensão, encaminhá-las ao classificador e retornar o resultado da análise. O Flask foi escolhido por sua simplicidade e eficiência no tratamento de requisições HTTP e manipulação de dados no formato JSON, facilitando a troca de informações entre os módulos da ferramenta.
- Integração via JSON: A comunicação entre a extensão e o servidor Flask foi implementada utilizando o formato JSON (JavaScript Object Notation), que permite a estruturação clara e eficiente dos dados transmitidos. Essa abordagem garante a interoperabilidade entre as tecnologias envolvidas, além de facilitar a escalabilidade e manutenção da ferramenta.

A Figura 1 ilustra o fluxo de comunicação entre os componentes, destacando o papel de cada um no processo de classificação e alerta ao usuário.

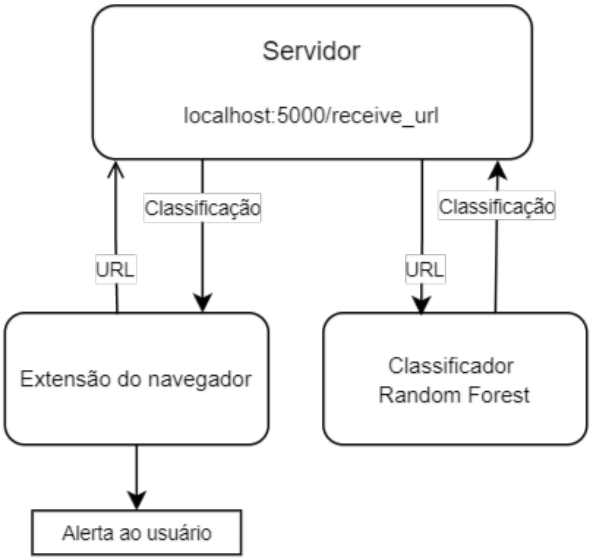

Figura 1. Fluxograma de funcionamento da ferramenta *Phishing Guardian*.

4.2. Ferramenta Desenvolvida

O *Phishing Guardian* consiste em uma ferramenta integrada composta por uma extensão de navegador e um módulo de classificação em Python, que trabalham em conjunto para identificar páginas de *phishing* em tempo real. O fluxo de funcionamento da ferramenta é ilustrado na Figura 2, que detalha a interação entre os componentes e as etapas do processo de análise.

4.2.1. Extensão para Navegador

A extensão, desenvolvida para o Google Chrome, monitora continuamente o tráfego do usuário durante a navegação, realizando três funções principais: coleta automática das URLs acessadas em tempo real, comunicação com o servidor Flask através de requisições HTTP para envio dos dados coletados e exibição de alertas ao usuário.

O processo de coleta das URLs se dá de forma que todo o caminho acessado pelo usuário é identificado e analisado, garantindo uma análise assertiva dos sites visitados. Assim que a URL é identificada, a mesma é enviada através do servidor Flask para que o código em Python inicie a análise e a coleta dos atributos.

A comunicação com o servidor Flask ocorre através de requisições POST assíncronas, utilizando o formato JSON para transmitir a URL que será analisada. A implementação foi projetada para operar de forma leve, garantindo que as requisições sejam processadas rapidamente sem afetar o desempenho do navegador ou a experiência de navegação do usuário.

Já a exibição de alertas ao usuário é apresentada em formato de pop-up, informando claramente se a página analisada é classificada como segura (”Navegação Segura”) ou potencialmente maliciosa (”Site Suspeito”), proporcionando um aviso em tempo real sem bloqueio da navegação do usuário, uma vez que o alerta desaparece após 3 segundos de tempo na tela.

A ferramenta opera de forma integrada, garantindo que toda URL acessada seja imediatamente capturada e submetida ao processo de análise. A comunicação eficiente com o servidor Flask permite a transmissão rápida dos dados, enquanto a interface de alerta proporciona feedback instantâneo ao usuário, mantendo-o informado sobre a segurança dos sites visitados sem interromper significativamente sua experiência de navegação.

4.2.2. Módulo de Classificação em Python

O classificador, implementado em Python, é responsável pela análise das URLs, realizando inicialmente o pré-processamento dos dados. Ao receber uma URL, o módulo extrai e normaliza os 13 atributos predefinidos, incluindo características como tamanho da URL, uso de HTTPS e validade do certificado SSL, preparando-os para a etapa de classificação.

Em seguida, a ferramenta utiliza o modelo de aprendizado de máquina previamente treinado para classificar a URL como legítima (0) ou maliciosa (1). Uma vez

concluída a análise, o resultado é enviado de volta ao servidor Flask, que atua como intermediário, repassando a informação à extensão do navegador para que seja exibido o alerta correspondente ao usuário final.

4.2.3. Implementação dos modelos

Os hiperparâmetros de cada algoritmo foram cuidadosamente ajustados para otimizar o desempenho na tarefa de detecção de *phishing*. A seleção final baseou-se em testes empíricos que avaliaram diversas combinações de parâmetros, mantendo sempre o equilíbrio entre capacidade preditiva e eficiência computacional. Para o classificador *Random Forest* os hiperparâmetros determinados foram:

- n_estimators = 100;
- criterion = 'gini';
- max_features = 'sqrt';
- Bootstrap = True;
- min_impurity_decrease = 0.00001;
- max_depth = None.

O classificador XGBoost foi desenvolvido utilizando os hiperparâmetros a seguir:

- n_estimators = 100;
- booster='gbtree';
- max_depth=6;
- learning_rate=0.3;

Enquanto o SVM foi modelado da seguinte forma:

- kernel='rbf';
- degree=3;
- gamma='scale';
- decision_function_shape='ovr';

Para cada hiperparâmetro descrito, foram feitos testes modificando-os a fim de identificar o impacto de cada um na classificação final. Além dos detalhados acima, os outros hiperparâmetros foram mantidos em seus valores padrões, sem nenhuma modificação.

4.2.4. Fluxo de Comunicação

A integração entre os componentes ocorre da seguinte forma:

1. A extensão envia a URL acessada para o servidor Flask (http://localhost:5000/receive_url);
2. O servidor encaminha a URL ao módulo Python, que realiza a classificação;
3. O resultado é retornado ao servidor e, posteriormente, à extensão, que exibe o alerta correspondente.

A Figura 2 destaca esse processo, evidenciando a sequência de ações desde a coleta da URL até a exibição do alerta.

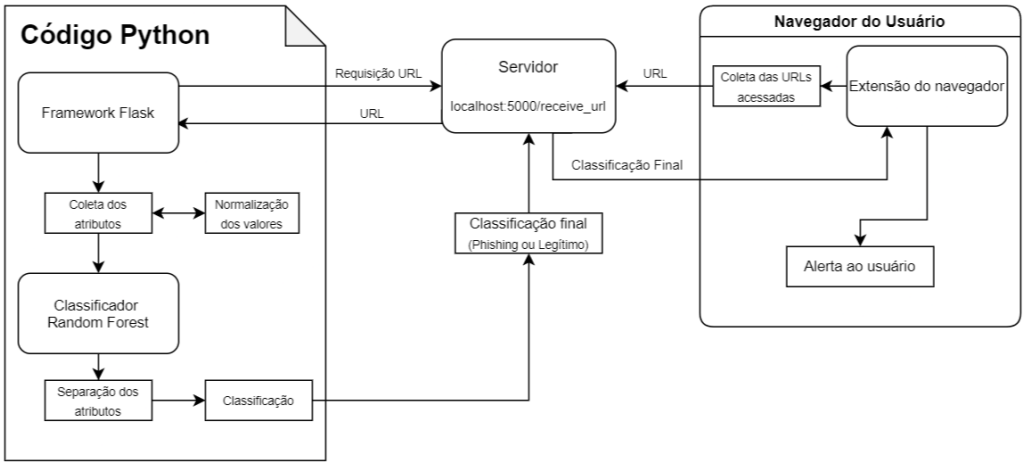

Figura 2. Fluxograma sobre as funcionalidades dos componentes da ferramenta *Phishing Guardian*.

4.3. Base de Dados

A ferramenta desenvolvida utiliza uma base de dados híbrida, composta por URLs legítimas e de *phishing* cuidadosamente selecionadas e processadas. Esta abordagem garantiu a criação de um conjunto de dados robusto e representativo das condições reais de navegação na web.

Foram utilizadas duas fontes principais para coleta de URLs de *phishing*: PhishTank, uma base de dados pública mantida pela Cisco Systems, contendo mais de 40 mil URLs de *phishing* verificadas e reportadas pela comunidade [Cisco Systems, Inc.] e OpenPhish, uma plataforma que disponibiliza gratuitamente 500 URLs de *phishing* ativas, atualizadas regularmente [OpenPhish].

A combinação dessas fontes permitiu a obtenção de uma variedade significativa de padrões de URLs maliciosas, aumentando a capacidade do modelo de identificar diferentes técnicas utilizadas por atacantes.

Por outro lado, para as URLs legítimas, adotou-se uma abordagem em duas etapas. Inicialmente, foi realizada a seleção de URLs conhecidas, utilizando-se da lista dos 500 websites mais populares do mundo disponibilizada pelo Moz, classificados por Autoridade de Domínio (esta métrica considera fatores como número de links externos e relevância nos mecanismos de busca).

A segunda etapa consistiu em executar a técnica de enumeração de domínios. Aplicou-se a ferramenta WayBack Urls para identificar subdomínios e páginas associadas aos domínios principais; esta técnica permitiu ampliar significativamente a base de URLs legítimas, mantendo sua confiabilidade.

A base de dados resultante se encontra contendo um total de 50.261 URLs, dentre elas sendo 55,5% (27.891) de *phishing* e 44,5% (22.370) legítimas.

Esta composição balanceada, aliada à variedade de fontes e técnicas de coleta, proporcionou um conjunto de dados robusto para treinamento e validação do modelo, superando as limitações de bases públicas tradicionais que frequentemente são pequenas ou não refletem adequadamente a diversidade da web atual.

4.4. Atributos Utilizados

Para o treinamento do classificador, foram selecionados 13 atributos estratégicos que capturam características distintivas entre URLs legítimas e maliciosas. A seleção baseou-se em análises de trabalhos relacionados e observações empíricas sobre padrões comuns em ataques de *phishing*. Os atributos escolhidos foram os seguintes:

- **Tamanho do URL (A1):** Quantidade total de caracteres na URL. URLs excessivamente longas podem indicar tentativas de ocultar informações suspeitas [Mohammad et al. 2012];
- **Utiliza HTTPS (A2):** Indicador booleano (0/1) da presença de criptografia. Embora muitos sites *phishing* já utilizem HTTPS, sua ausência ainda é um forte indicativo de risco;
- **Possui formato IP (A3):** Flag (0/1) para URLs que utilizam endereços IP diretamente, prática incomum em sites legítimos;
- **Quantidade de pontos '.' (A4):** Número de pontos na URL, onde valores elevados podem sugerir subdomínios artificiais;
- **Contém '@' (A5):** Indicador (0/1) da presença deste caractere, frequentemente usado para mascarar domínios maliciosos;
- **Total de dias ativos (A6):** Idade do domínio em dias. Domínios muito novos têm maior probabilidade de serem maliciosos;
- **PageRank (A7):** Pontuação de relevância do domínio. Sites *phishing* geralmente possuem scores significativamente mais baixos;
- **Possui formulário HTML (A8):** Flag (0/1) para detecção de campos de input, comuns em páginas que coletam dados sensíveis;
- **Validade do certificado SSL (A9):** Duração em dias da certificação. Certificados de curta duração (ex: 3 meses) são mais comuns em *phishing*;
- **Contém '//' (A10):** Indicador (0/1) de redirecionamentos potencialmente maliciosos no corpo da URL;
- **Possui 'https' no corpo (A11):** Flag (0/1) para tentativas de enganar usuários com falsos indicativos de segurança;
- **Contém hífen '-' (A12):** Marcador (0/1) de domínios suspeitos que usam hifens para imitar marcas legítimas;
- **Possui iframe (A13):** Indicador (0/1) da presença desta tag HTML, potencialmente usada para embutir conteúdo malicioso.

5. Resultados

Nesta seção, são apresentados os resultados obtidos na avaliação dos modelos de aprendizado de máquina selecionados. Foram comparados três algoritmos amplamente utilizados na literatura: *Random Forest*, *Support Vector Machine* (SVM) e XGBoost. A escolha desses modelos se justifica por suas características distintas: enquanto o SVM é conhecido por sua capacidade de generalização em espaços de alta dimensionalidade, o *Random Forest* se destaca pela robustez e facilidade de interpretação, e o XGBoost tem ganhado notoriedade por seu desempenho superior em tarefas de classificação, especialmente em conjuntos de dados complexos.

5.1. Metodologia de Avaliação

Foram avaliados três algoritmos de aprendizado de máquina: Random Forest, Support Vector Machine (SVM) e XGBoost. Todos os modelos foram treinados com a mesma base de dados contendo 50.261 URLs, utilizando uma divisão de 85% para treinamento e 15% para teste. As métricas de avaliação incluíram acurácia, precisão, recall e F1-score.

A acurácia foi incluída como métrica base por fornecer uma medida geral da eficácia do classificador, representando a proporção total de previsões corretas. Contudo, considerando a natureza sensível do problema e a possível distribuição não uniforme das classes, complementamos esta análise com métricas mais específicas.

A precisão assume particular importância por quantificar a confiabilidade dos alertas gerados, sendo essencial para minimizar falsos positivos que poderiam comprometer a usabilidade da ferramenta ao classificar erroneamente sites legítimos como maliciosos.

O recall, por sua vez, mede a capacidade da ferramenta em identificar corretamente as ameaças reais, aspecto crítico em aplicações de cibersegurança, nas quais falhas na detecção podem ter consequências significativas.

Por fim, o F1-Score foi incorporado como métrica síntese, proporcionando um equilíbrio entre precisão e recall através de sua média harmônica.

5.2. Descrição dos resultados

Para cada algoritmo, a acurácia final demonstrada foi quantificada a partir de validação cruzada, com a divisão dos dados em 10 partes e a realização da média final. Este processo foi feito com o intuito de gerar um resultado mais real e com menor possibilidade de ocorrência de overfitting nos modelos.

Em relação ao algoritmo SVM, este foi o que demonstrou o pior desempenho durante os testes, retornando 96,78% de acurácia, 96,86% de precisão, 95,62% de recall e 96,24% de F1 Score.

Na sequência, o Random Forest teve o desempenho que ficou em segundo lugar, com 99,40% de acurácia, 99,64% de precisão, 99,02% de recall e 99,33% de F1 Score.

Em primeiro lugar, com o melhor desempenho, o algoritmo XGBoost retornou 99,51% de acurácia, 99,58% de precisão, 99,31% de recall e 99,45% de F1 Score.

A Tabela 2 apresenta os resultados comparativos entre os modelos.

Tabela 2. Comparação de desempenho entre os modelos.

Modelo	Acurácia	Precisão	Recall	F1-Score
XGBoost	99,51%	99,58%	99,31%	99,45%
Random Forest	99,40%	99,64%	99,02%	99,33%
SVM	96,78%	96,86%	95,62%	96,24%

O XGBoost demonstrou superioridade geral, mesmo tendo o Random Forest tido um desempenho bastante próximo. Essa diferença pode ser atribuída à melhor capacidade do XGBoost em lidar com relações não lineares e seu mecanismo de regularização. O

desempenho inferior do SVM, pode ser atribuído possivelmente à dificuldade em lidar com conjuntos de dados grandes e altamente dimensionais.

Além disso, uma análise sobre o coeficiente de correlação (demonstrado na Figura 3) foi conduzida a fim de identificar possíveis atributos que pudessem não estar contribuindo tanto para a classificação dos modelos, demonstrando que os atributos A3, A5 e A10 apresentavam correlação quase nula com os resultados, sendo esses selecionados para remoção.

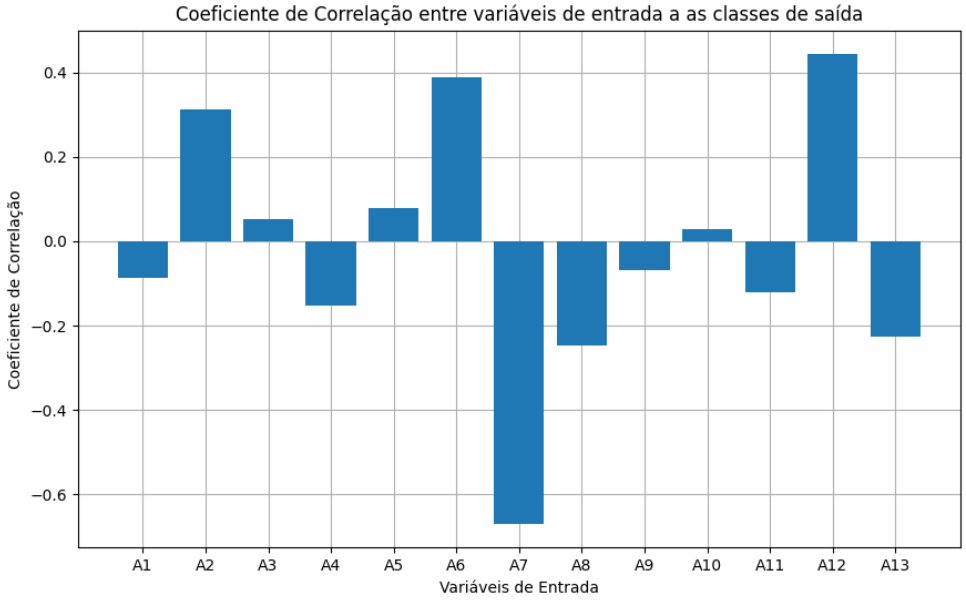

Figura 3. Coeficiente de correlação dos atributos.

Sua remoção levou a alterações no desempenho dos modelos, conforme mostrado na Tabela 3:

Tabela 3. Comparação dos resultados sem os atributos A3, A5 e A10.

Modelo	Acurácia	Precisão	Recall	F1-Score
XGBoost	99,50%	99,55%	99,28%	99,42%
Random Forest	99,39%	99,58%	99,20%	99,39%

Embora o Random Forest tenha apresentado melhora nas métricas após a remoção dos atributos, o XGBoost manteve-se superior, mesmo com uma pequena redução de desempenho. Portanto, optou-se por utilizar o XGBoost com todos os 13 atributos originais, que proporcionou os melhores resultados globais para a tarefa de detecção de *phishing*.

Apesar da superioridade do XGBoost, o Random Forest manteve-se como uma excelente alternativa, com diferenças mínimas nas principais métricas, demonstrando sua robustez para o problema em questão. A escolha final pelo XGBoost considerou não apenas seu desempenho marginalmente superior, mas também sua eficiência computacional e capacidade de generalização, características essenciais para uma implementação em ambiente real de detecção de *phishing*.

5.3. Discussão Detalhada dos Resultados

A Tabela 2 evidencia a superioridade do XGBoost, que alcançou 99,51% de acurácia e F1-Score de 99,45%, ultrapassando o Random Forest em 0.11 p.p. de acurácia e 0.12 p.p. de F1-Score. Essa vantagem está ligada à capacidade do XGBoost de explorar relações não lineares, com regularização que atenua sobreajuste em conjuntos de dados de alta dimensionalidade.

Em contraste, o SVM apresentou 96,78% de acurácia, confirmando limitações já discutidas na literatura para bases volumosas e heterogêneas. Apesar de valores absolutos elevados, a diferença de três pontos percentuais em relação aos algoritmos baseados em árvores reforça a necessidade de balancear custo computacional e desempenho preditivo quando se planeja uma implantação em tempo real.

5.4. Impacto da Remoção de Atributos pouco Relevantes

A remoção dos atributos *A3, A5* e *A10*, identificados como de correlação quase nula (Figura 3), levou aos resultados sumarizados na Tabela 2. Observa-se que o XGBoost manteve a liderança, agora com 99,50% de acurácia, enquanto o Random Forest ficou em 99,39%. A variação máxima observada foi de 0.02 p.p. nos quatro indicadores, indicando que os três atributos removidos possuem efeito marginal sobre o poder discriminatório do classificador.

Este achado confirma duas hipóteses implícitas: (a) parte dos atributos originais apresenta redundância, e (b) o XGBoost é menos sensível a essa redundância graças ao seu mecanismo de poda de árvores e regularização, tornando-o a escolha recomendada para a versão de produção da ferramenta.

6. Evolução da Pesquisa e Melhorias Implementadas

O presente trabalho passou por três estágios distintos de desenvolvimento, cada um representando avanços significativos em relação ao anterior. O trabalho [Guarizi and Mascarenhas 2024] estabeleceu as bases da ferramenta com um protótipo funcional utilizando Random Forest, alcançando métricas iniciais promissoras, contendo uma base de dados com mais de 50 mil URLs, porém com limitações na experiência do usuário e na abrangência da avaliação técnica.

Na continuidade do desenvolvimento, a ferramenta foi substancialmente aprimorada com o refinamento dos atributos utilizados, resultando em um aumento considerável na acurácia do modelo. Entretanto, mantinha-se a limitação sobre os alertas retornados ao usuário que exigiam interação manual, interrompendo o fluxo de navegação a cada classificação, além de se manter limitado à visualização do desempenho de apenas 1 modelo, sendo o Random Forest.

A versão atual incorpora duas melhorias fundamentais que elevam a ferramenta proposta ainda mais. Primeiramente, o sistema de alerta foi completamente redesenhado - agora as notificações desaparecem automaticamente após um tempo pré-definido, preservando a atenção do usuário sobre possíveis riscos sem comprometer a fluidez da navegação. Além disso, o alerta realizado possui uma coloração de fundo que alerta o usuário sobre a classificação, sendo a coloração verde quando a detecção é de uma página legítima (demonstrado na Figura 4) e vermelha quando é uma página de possível fraude

(demonstrado na Figura 5). Esta mudança representa um avanço significativo na usabilidade prática da ferramenta.

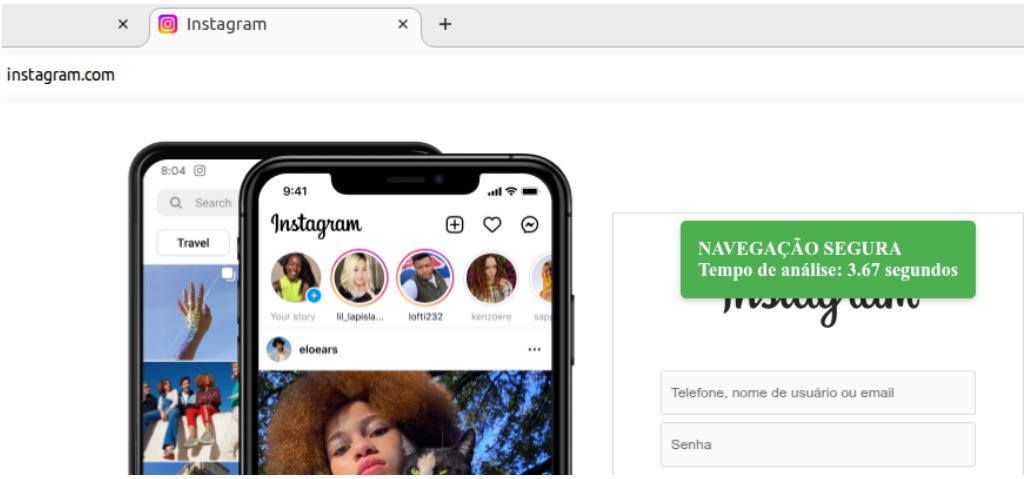

Figura 4. Alerta - Sites Legítimos.

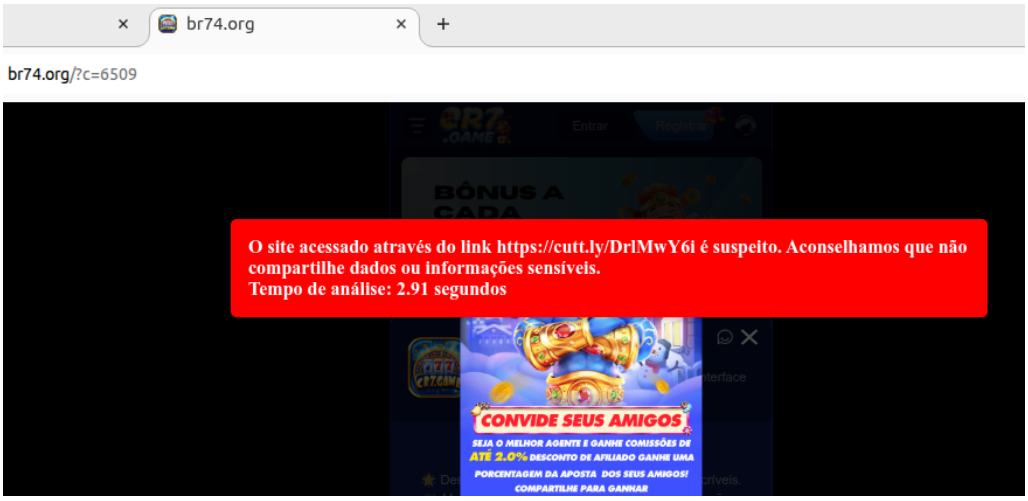

Figura 5. Alerta - Sites *Phishing*.

Em segundo lugar, a metodologia de avaliação foi ampliada com a inclusão de dois modelos adicionais (SVM e XGBoost) em uma análise comparativa abrangente. Os resultados demonstraram que o XGBoost superou o desempenho do Random Forest original, alcançando 99,51% de acurácia contra 99,40% do modelo Random Forest, além de melhorias consistentes nas demais métricas (F1-Score de 99,45% contra 99,33%). O SVM, embora tenha apresentado desempenho inferior (96,78% de acurácia), forneceu insights valiosos sobre as limitações de certas abordagens para este problema específico.

Estas evoluções cumulativas transformaram a solução de um protótipo acadêmico em uma ferramenta robusta e prática, com potencial real de aplicação no cotidiano dos usuários, ao mesmo tempo que estabelecem novas referências em termos de desempenho para sistemas de detecção de *phishing* baseados em aprendizado de máquina.

Lição de usabilidade e foco no usuário. A reformulação do sistema de alerta, que agora desaparece automaticamente após poucos segundos, resultou de testes exploratórios descritos na versão preliminar do trabalho. Dessa forma, o sistema de alerta agora evita a "fadiga de cliques" quando usuários precisariam confirmar manualmente avisos sucessivos, especialmente durante campanhas de validação que envolviam diversas URLs em sequência. A solução adotada preserva o caráter informativo da notificação, mas respeita o princípio de mínima interrupção da navegação, fortalecendo a aceitabilidade prática do protótipo.

Impacto do estudo de correlação nos atributos. A análise de correlação (Figura 3) evidenciou que os atributos *A3*, *A5* e *A10* apresentavam contribuição estatisticamente irrelevante para os classificadores, o que garantiria uma latência de retorno menor. Entretanto, este processo de remoção apenas favoreceu as métricas adquiridas através do algoritmo Random Forest, tendo sido selecionado e implementado o XGBoost utilizando todos os 13 atributos.

Aprendizado a partir da comparação multialgorítmica. A inclusão de SVM e XGBoost ao conjunto de experimentos forneceu um olhar abrangente sobre diferentes paradigmas de aprendizagem. Os testes confirmaram a vantagem consistente do XGBoost em relação ao Random Forest, enquanto expuseram limitações do SVM para o domínio de URLs. Essa evidência objetiva tornou-se decisiva para a escolha do XGBoost na versão operacional, mantendo o Random Forest como alternativa robusta em cenários onde dependências externas ou políticas de licenciamento inviabilizem o primeiro.

Roteiro de continuidade da pesquisa. Com as melhorias consolidadas, o protótipo deixa de ser apenas um artefato acadêmico e passa a atender requisitos de uso cotidiano. Como próximos passos, foi identificada a possibilidade de ampliar ainda mais a base de URLs, e investigar técnicas de otimização incremental que reduzam ainda mais o tempo de resposta do classificador. Essas iniciativas pretendem fortalecer a detecção de ataques *zero-day* e garantir que a ferramenta acompanhe a rápida evolução do cenário de *phishing*.

7. Conclusão

Este trabalho desenvolveu uma ferramenta para detecção de *phishing*, partindo da construção de uma base de dados robusta e dando prosseguimento com o processo de desenvolvimento, que envolveu uma criteriosa seleção e validação de 13 atributos distintos, sendo estes desde características básicas como tamanho da URL até métricas mais complexas como PageRank e validade de certificados SSL, todos meticulosamente extraídos e normalizados para alimentar os modelos de ML. O *Phishing Guardian* pode ser acessado através do GitHub em [Guarizi and Mascarenhas 2025].

A implementação técnica combinou uma extensão para Chrome desenvolvida em JavaScript com um backend em Python, utilizando o framework Flask para estabelecer uma comunicação eficiente entre os componentes. Foram conduzidos testes comparativos entre três algoritmos distintos (XGBoost, Random Forest e SVM), onde o XGBoost se

destacou com 99,51% de acurácia e 99,45% de F1-Score, superando ligeiramente o Random Forest (99,40% de acurácia) e apresentando vantagem significativa sobre o SVM (96,78% de acurácia).

Como perspectivas para trabalhos futuros, destaca-se a importância de expandir continuamente a base de dados para acompanhar as novas táticas de *phishing* que surgem diariamente, além de explorar técnicas de otimização para reduzir o tempo de resposta da ferramenta. Testes em larga escala com usuários reais em diferentes cenários de navegação seriam valiosos para validar a eficácia prática da ferramenta, assim como a incorporação de técnicas avançadas de processamento de linguagem natural poderia elevar ainda mais a precisão da detecção.

Outra perspectiva para evoluções futuras deste trabalho seria a utilização de redes neurais para detecção de páginas de *phishing*, entretanto, é fato que redes neurais são modelos muito custosos computacionalmente, que consomem grande quantidade de memória e processamento. Levando em consideração que o objetivo inicial do presente trabalho foi o desenvolvimento de uma ferramenta com foco no usuário, podendo ser executada diretamente em sua máquina, optou-se por escolher modelos mais leves e ainda precisos para este tipo de detecção. Ainda assim, a implementação da ferramenta em nuvem viabilizaria essa possibilidade.

Os resultados obtidos nesta pesquisa não apenas comprovam a eficácia da abordagem proposta, mas também estabelecem uma base sólida para futuros desenvolvimentos no combate ao *phishing*, uma ameaça cada vez mais sofisticada no cenário de cibersegurança atual.

Agradecimentos

Este capítulo foi realizado com recursos do CNPq (processos 405940/2022-0 e 408255/2023-4), CAPES (processos 88887.954253/2024-00 e 88887.123038/2025-00), FAPERJ e RNP/SENAI-SP/Softex/MCTI.

Referências

Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S. (2007). A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-Phishing Working Groups 2nd Annual ECrime Researchers Summit*, pages 60–69.

Ahmad, R., Alsmadi, I., Alhamdani, W., and Tawalbeh, L. (2023). Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, 56(10):10733–10811.

Al Saidat, M. R., Yerima, S. Y., and Shaalan, K. (2024). Advancements of SMS spam detection: A comprehensive survey of NLP and ML techniques. *Procedia Computer Science*, 244:248–259.

Alanezi, M. (2021). Phishing detection methods: A review. Technium.

Alawida, M., Omolara, A. E., Abiodun, O. I., and Al-Rajab, M. (2022). A deeper look into cybersecurity issues in the wake of Covid-19: A survey. *Journal of King Saud University-Computer and Information Sciences*, 34(10):8176–8206.

Bhattacharya, T., Veeramalla, S., and Tanniru, V. (2023). A survey on retrieving confidential data using phishing attack. In *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, pages 2528–2535. IEEE.

Castaño, F., Fernañdez, E. F., Alaiz-Rodríguez, R., and Alegre, E. (2023). Phikita: Phishing kit attacks dataset for phishing websites identification. *IEEE Access*, 11:40779–40789.

Cisco Systems, Inc. Phishtank. Disponível em `https://phishtank.org/`. Acessado em maio de 2024.

Correia, P. H. B. and Pedrini, H. (2020). Detecção de domínios maliciosos baseada em técnicas de aprendizado de máquina. Trabalho de conclusão de curso (Bacharelado em Ciência da Computação). Universidade Estadual de Campinas.

Fajar, A., Yazid, S., and Budi, I. (2024). Enhancing phishing detection through feature importance analysis and explainable AI: A comparative study of CatBoost, XGBoost, and EBM models. arXiv.

Guarizi, B. D. and Mascarenhas, D. M. (2024). Identificação de ataques de phishing através de machine learning. In *Anais Estendidos do XXIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. SBC.

Guarizi, B. D. and Mascarenhas, D. M. (2025). Phishing guardian. Disponível em `https://github.com/bguarizi/phishing-guardian`. Acessado em julho de 2025.

Guo, Y. (2023). A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer communications*, 198:175–185.

Kamal, A. H. A., Yen, C. C. Y., Ping, M. H., and Zahra., F. (2020). Cybersecurity issues and challenges during Covid-19 pandemic. *Preprints.org*.

Kim, J., Kim, J., Wi, S., Kim, Y., and Son, S. (2022). Hearmeout: detecting voice phishing activities in Android. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 422–435.

Kyriazoglou, J. (2024). Information security and breach definitions and obligations. In *Information Security Incident and Data Breach Management: A Step-by-Step Approach*, pages 1–14. Springer.

Mohammad, R. M., Thabtah, F., and McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. In *2012 international conference for internet technology and secured transactions*, pages 492–497. IEEE.

OpenPhish. Openphish. Disponível em `https://openphish.com/`. Acessado em maio de 2024.

Pranggono, B. and Arabo, A. (2021). Covid-19 pandemic cybersecurity issues. *Internet Technology Letters*, 4(2):e247.

Sadaf, K. (2023). Phishing website detection using XGBoost and Catboost classifiers. In *2023 International Conference on Smart Computing and Application (ICSCA)*. IEEE.

Safi, A. and Singh, S. (2023). A systematic literature review on phishing website detection techniques. In *Journal of King Saud University-Computer and Information Sciences*, pages 590–611.

Salloum, S., Gaber, T., Vadera, S., and Shaalan, K. (2022). A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access*, 10:65703–65727.

Sharevski, F., Devine, A., Pieroni, E., and Jachim, P. (2022). Phishing with malicious qr codes. In *Proceedings of the 2022 European Symposium on Usable Security*, pages 160–171.

Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., and Zhang, C. (2009). An empirical analysis of phishing blacklists.

Singh, T., Kumar, M., and Kumar, S. (2024). Walkthrough phishing detection techniques. *Computers and Electrical Engineering*, 118:109374.

Souza, J. A. and Mascarenhas, D. M. (2023). Detecção de ataques de phishing em tempo real utilizando algoritmos de aprendizado de máquina. In *Anais Estendidos do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. SBC.

Sullivan, B. Phishing costs have tripled since 2015. Disponível em `https://ponemonsullivanreport.com/2021/08/`. Acessado em janeiro de 2025.