UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Celso Soares Cassiano de Souza

Aplicação Web para Detecção Automática de URLs Maliciosas com Aprendizado de Máquina

Recife

Agosto de 2025

Celso Soares Cassiano de Souza

Aplicação Web para Detecção Automática de URLs Maliciosas com Aprendizado de Máquina

Artigo apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE
Departamento de Estatística e Informática
Curso de Bacharelado em Sistemas de Informação

Orientador: Lidiano Oliveira

Recife
Agosto de 2025

Aplicação Web para Detecção Automática de URLs Maliciosas com Aprendizado de Máquina

Celso Soares Cassiano de Souza [1], **Lidiano Oliveira** [1]

[1]Departamento de Estatística e Informática – Universidade Federal Rural de Pernambuco
Rua Dom Manuel de Medeiros, s/n, - CEP: 52171-900 – Recife – PE – Brasil

celso.soares@ufrpe.br, lidiano.oliveira@ufrpe.br

Resumo. *A segurança cibernética tem se tornado uma das principais preocupações da era digital, impulsionada pelo crescimento acelerado da internet e pela proliferação de ameaças como phishing, malware e roubo de dados. Este trabalho propõe uma abordagem baseada em aprendizado de máquina para classificar URLs como legítimas ou maliciosas, utilizando um conjunto abrangente de atributos extraídos diretamente das URLs e de fontes complementares, como registros WHOIS e informações de rede. Foram aplicados e analisados algoritmos como Random Forest, SVM e XGBoost sobre um conjunto de dados coletado de fontes confiáveis, como PhishTank e Kaggle. As características consideradas englobam aspectos léxicos, informações de rede, conexão e reputação. A avaliação dos modelos foi conduzida por meio de métricas como acurácia, precisão, recall e F1-score, evidenciando um desempenho satisfatório na detecção de sites maliciosos. Como aplicação prática, foi desenvolvida uma plataforma interativa com Streamlit, permitindo que qualquer usuário insira uma URL e receba uma análise imediata sobre sua legitimidade. A análise de importância das variáveis forneceu insights valiosos sobre os fatores mais influentes no processo de classificação, contribuindo tanto para a transparência quanto para a evolução futura do sistema.*

Abstract. *Cyber security has become one of the main concerns of the digital age, driven by the accelerated growth of the internet and the proliferation of threats such as phishing, malware and data theft. This work proposes a machine learning -based approach to classify URLs as legitimate or malicious, using a comprehensive set of attributes extracted directly from URLs and complementary sources, such as whois records and network information. Algorithms such as Random Forest, SVM and XGboost were applied and analyzed on a data set collected from reliable sources such as Phishtank and Kaggle. The characteristics considered encompass lexical aspects, network information, connection and reputation. The evaluation of the models was conducted through metrics such as accuracy, accuracy, recall and F1-Score, showing satisfactory performance in detecting malicious websites. As a practical application, an interactive streamlit platform has been developed, allowing any user to enter a URL and receive an immediate analysis of its legitimacy. The analysis of importance of variables provided valuable insights on the most influential factors in the classification process, contributing to both transparency and future system evolution.*

1. Introdução

Com o crescimento exponencial da internet, também aumentaram os ataques cibernéticos, sendo os sites maliciosos uma das formas mais comuns de disseminação de ameaças, como *phishing*, *malware* e roubo de dados [Anderson 2020]. Além de comprometer a segurança dos usuários, os ataques representam uma ameaça significativa para empresas e sistemas, que frequentemente são alvo de ações fraudulentas. Durante a pandemia de COVID-19, a digitalização acelerada da sociedade contribuiu para um aumento expressivo nos crimes cibernéticos, com destaque para fraudes online e ataques de ransomware, impulsionados pela maior exposição de usuários e sistemas [Freitas and Loiola Junior 2023]. A identificação e a prevenção desses riscos são fundamentais para garantir a integridade e a segurança no ambiente digital.

Neste contexto, métodos automatizados que auxiliem na identificação da legitimidade de sites tornam-se cada vez mais necessários. A detecção de sites maliciosos, de forma rápida e precisa, é um desafio crescente, devido a frequente sofisticação das ameaças e à facilidade com que novas URLs maliciosas são criadas e distribuídas. Soluções que permitam classificar sites como legítimos ou maliciosos de maneira eficiente podem ser determinantes na proteção de usuários e sistemas contra ataques cibernéticos [Ma et al. 2009].

Atualmente, abordagens tradicionais de segurança online, como antivírus, listas negras e filtros manuais, têm se mostrado insuficientes diante da evolução constante das técnicas de ataque. Esses mecanismos reativos, por dependerem da identificação prévia da ameaça, falham na detecção de ataques *zero-day* ou URLs recém-geradas. Isso torna imprescindível o uso de métodos mais sofisticados e proativos, como o aprendizado de máquina, que permite uma análise automática e em tempo real da legitimidade de sites. Ao utilizar algoritmos capazes de aprender padrões a partir de grandes volumes de dados e identificar características comuns em URLs maliciosas, é possível automatizar a detecção com maior precisão, agilidade e escalabilidade [Ayres et al. 2019b].

Para viabilizar o desenvolvimento da pesquisa, foram utilizadas duas fontes principais de dados: o conjunto *Malicious URLs Dataset*, disponível na plataforma Kaggle, e os registros atualizados do *PhishTank*. O primeiro reúne mais de 650 mil URLs rotuladas em categorias como benigno, *phishing*, desfiguração de páginas (*deface* ou *defacement* como é conhecido popularmente) e *malware*, oferecendo uma base ampla e diversificada em formato CSV. Por sua vez, o *PhishTank* complementa essa abordagem com amostras em tempo real, que são validadas colaborativamente por usuários da comunidade

Este trabalho visa explorar essa abordagem, desenvolvendo um sistema de classificação de sites legítimos e maliciosos por meio de técnicas de aprendizado de máquina. O modelo proposto consiste em um sistema de detecção de URLs maliciosas baseado em aprendizado de máquina supervisionado, capaz de classificar automaticamente sites como legítimos ou maliciosos com base em múltiplas características extraídas das URLs. O sistema será treinado com um conjunto diversificado de características extraídas de URLs, incluindo aspectos léxicos (como comprimento da URL, presença de símbolos etc), dados de rede (como tempo de resposta e número de redirecionamentos) e informações relacionadas à hospedagem, como dados WHOIS (idade e tempo de expiração do domínio, país de registro etc).

A combinação desses fatores proporciona uma análise abrangente, permitindo identificar comportamentos suspeitos e distinguir entre sites confiáveis e potencialmente perigosos com maior robustez.

Este trabalho utiliza a linguagem Python para a manipulação dos dados e o treinamento dos modelos de aprendizado de máquina. Por meio de bibliotecas amplamente consolidadas no ecossistema de ciência de dados será possível realizar etapas fundamentais do processo, como limpeza, transformação e análise dos dados, bem como a construção, validação e comparação dos modelos preditivos. Essa escolha visa garantir maior eficiência no desenvolvimento, além de aproveitar os recursos e a flexibilidade que a linguagem oferece para experimentação e reprodutibilidade dos resultados.

Além da construção do modelo de aprendizado de máquina, este trabalho também busca proporcionar uma aplicação prática para os usuários finais. Para isso, será desenvolvida uma plataforma interativa utilizando a biblioteca Streamlit, que permitirá que qualquer usuário insira uma URL e obtenha uma classificação instantânea sobre a segurança do site. Essa interface tem como objetivo tornar o sistema acessível mesmo a usuários sem conhecimento técnico, contribuindo para democratização da cibersegurança e promovendo uma maior conscientização sobre os riscos da navegação digital.

Por fim, o projeto também se preocupa com a interpretabilidade do modelo, promovendo a explicação dos fatores que influenciam a classificação por meio da análise de importância das variáveis. Essa etapa é essencial para tornar os modelos mais transparentes, confiáveis e auditáveis, contribuindo não apenas para detecção, mas também para compreensão dos padrões de comportamento associados a ameaças digitais. Com isso, este trabalho pretende oferecer uma contribuição técnica e prática para o combate ao crescente problema da disseminação de URLs maliciosas na web.

1.1. Objetivos

O presente trabalho tem como objetivo geral desenvolver uma solução automatizada para classificar URLs como legítimas ou maliciosas, utilizando algoritmos de aprendizado de máquina treinados a partir de atributos léxicos e de rede extraídos das próprias URLs.

Como objetivos específicos, propõe-se:

- Extrair características relevantes das URLs, incluindo informações léxicas, de domínio, rede e hospedagem, para composição de um dataset robusto;
- Construir e treinar modelos de aprendizado de máquina capazes de identificar padrões associados a URLs maliciosas;
- Avaliar o desempenho dos modelos com métricas apropriadas, como acurácia, precisão e taxa de falsos positivos;
- Investigar o impacto da inclusão de diferentes tipos de atributos (como WHOIS, DNS e redirecionamentos) na melhoria da capacidade de detecção;
- Desenvolver uma plataforma interativa com Streamlit, que permita ao usuário final inserir URLs e obter, em tempo real, uma análise sobre sua legitimidade;
- Contribuir com uma ferramenta prática que auxilie na conscientização e prevenção de ataques cibernéticos baseados em engenharia social, como phishing e malware.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve os fundamentos teóricos adotados. A

Seção 4 detalha as ferramentas e métodos utilizados. A Seção 5 discute os resultados experimentais. Por fim, a Seção 6 apresenta as conclusões e sugestões para trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção, abordam-se resumidamente alguns trabalhos relacionados ao tema proposto. O estudo apresentado por [Ayres et al. 2019a] descreve a aplicação de aprendizado de máquina para a detecção automática de URLs maliciosas no contexto brasileiro. Foram utilizadas mais de 110 características extraídas das URLs, abrangendo aspectos léxicos, de rede e reputação. Os algoritmos *Random Forest* e *Gradient Boosting* obtiveram acurácias superiores a 96%, demonstrando como a combinação de diversas fontes de dados pode elevar significativamente a eficácia dos modelos de classificação. Este trabalho, por sua vez, propõe uma solução mais generalista, voltada à detecção de URLs maliciosas em escala global. Apesar dos bons resultados, o estudo limita-se ao cenário nacional, o que pode restringir a aplicabilidade do modelo em contextos mais amplos, além de não apresentar uma interface prática que permita a utilização da solução por usuários finais ou sua integração com sistemas em tempo real.

O trabalho de [Le et al. 2018] introduz o modelo URLNet, que utiliza redes neurais convolucionais (CNNs) para representar e classificar URLs maliciosas sem a necessidade de extração manual de características. O diferencial da abordagem está na capacidade de aprender automaticamente padrões diretamente da representação textual da URL. O modelo demonstrou desempenho competitivo, abrindo caminho para soluções baseadas em deep learning. Embora o modelo tenha apresentado desempenho competitivo, sua natureza de caixa-preta dificulta a interpretação dos resultados, o que pode limitar sua adoção em cenários onde a transparência é essencial.

A pesquisa conduzida em [Ma et al. 2009] aplicou métodos supervisionados, como *Support Vector Machine* e *Naive Bayes*, para classificar URLs maliciosas com base exclusivamente em características léxicas, sem acessar o conteúdo das páginas. O estudo demonstrou que informações léxicas são eficazes para a detecção de URLs maliciosas, mesmo com modelos relativamente simples. Mas a limitação à análise textual pode comprometer a detecção de ameaças mais sofisticadas.

O estudo de [Coutinho 2022] tratou da detecção de páginas de phishing utilizando algoritmos como Árvore de Decisão, *Random Forest*, *Extra Trees* e *XGBoost* em um conjunto de dados com mais de 88 mil registros. O algoritmo *XGBoost* apresentou os melhores resultados, comprovando sua eficácia na detecção de páginas falsas. No entanto, a análise concentrou-se principalmente em atributos extraídos diretamente das páginas ou do HTML.

A proposta de [Domingos and Dalbert 2023] propõe uma extensão de navegador que utiliza o algoritmo *Random Forest* para detectar URLs maliciosas em tempo real, emitindo alertas ao usuário durante a navegação. A proposta reforça a viabilidade de embarcar modelos de inteligência artificial em soluções práticas de cibersegurança. No entanto, a abordagem está restrita ao contexto específico de navegação via browser e requer instalação local.

A pesquisa de [Shaikh et al. 2023] introduz o *AntiPhishStack*, um modelo empilhado baseado em LSTM combinado a outros classificadores para otimizar a detecção de

URLs de *phishing*. Avaliando diversas características, o estudo demonstrou que a abordagem empilhada superou os classificadores individuais. No entanto, a complexidade computacional e a baixa interpretabilidade desses modelos podem dificultar sua adoção em cenários com recursos limitados ou que exigem transparência nos resultados.

O estudo de [Sahoo et al. 2017] avaliou algoritmos como *Random Forest*, Árvore de Decisão e *Support Vector Machine* na detecção de sites de *phishing* com base em 30 características extraídas das páginas. Os resultados destacaram a robustez do *Random Forest*, que obteve excelente acurácia.No entanto, o trabalho foi conduzido com um conjunto de dados estático e relativamente limitado, o que pode comprometer a generalização do modelo a ambientes dinâmicos e ataques mais recentes, que evoluem constantemente em termos de estrutura e técnicas de evasão.

3. Referencial Teórico

Nesta seção, é apresentada uma introdução aos conceitos fundamentais, técnicas e temas discutidos ao longo deste artigo. Compreender esses elementos é essencial para alcançar uma compreensão plena da proposta deste trabalho.

3.1. Segurança da Informação

A segurança da informação é o conjunto de práticas, políticas e tecnologias que visam proteger os ativos informacionais, digitais ou físicos, contra ameaças que possam comprometer sua confidencialidade, integridade e disponibilidade (CIA - *confidentiality*, *integrity* e *availability*) [Anderson 2020]. A tríade CIA é complementada por princípios como autenticidade, não repúdio e auditoria.

Ela é essencial para garantir confiança e continuidade nos ambientes digitais, evitando perdas financeiras, danos à reputação e violações de privacidade. Com a digitalização crescente, ataques como *phishing*, *malware*, *ransomware* e roubo de credenciais se tornaram frequentes, sendo o *phishing* responsável por mais de 90% das violações bem-sucedidas [Jakobsson and Myers 2006].

A análise de URLs tornou-se uma camada crítica de defesa, pois grande parte das ameaças começa com links aparentemente inofensivos. Detectar padrões maliciosos permite barrar acessos perigosos antes que o conteúdo seja carregado, algo essencial em sistemas corporativos como filtros de *email* e *gateways web*.

Soluções tradicionais como antivírus e *firewalls*, baseadas em assinaturas, têm dificuldades em lidar com ameaças *zero-day* e variantes modificadas. Com isso, técnicas de aprendizado de máquina têm ganhado destaque. Algoritmos como *Random Forest*, *XGBoost* e *SVM* permitem prever se uma URL é legítima ou maliciosa com base em características estruturais [Sikorski and Honig 2012, Breiman 2001].

A integração da inteligência artificial à segurança cibernética representa um avanço significativo, permitindo a detecção automatizada e proativa de ameaças. Esses modelos oferecem escalabilidade e agilidade na resposta a incidentes, reforçando a segurança digital de usuários e organizações. Técnicas de aprendizado de máquina são capazes de identificar padrões anômalos em tempo real, minimizando o impacto de ataques antes que se propaguem. Essa abordagem dinâmica torna a defesa cibernética mais adaptável frente à evolução constante das ameaças digitais.

3.1.1. Phishing

Phishing é uma das técnicas mais comuns de ataque cibernético e consiste na tentativa de enganar usuários para que forneçam informações sensíveis, como senhas, dados bancários ou credenciais de acesso, simulando páginas legítimas ou enviando mensagens fraudulentas [Jakobsson and Myers 2006]. Esses ataques exploram vulnerabilidades humanas, utilizando a engenharia social como principal arma para manipular a confiança das vítimas.

Com a massificação da internet, os ataques de *phishing* tornaram-se cada vez mais sofisticados e diversificados. Inicialmente, esses ataques eram conduzidos por meio de emails genéricos e pouco elaborados, mas atualmente se estendem para redes sociais, aplicativos de mensagens instantâneas, mensagens SMS (*smishing*), ligações telefônicas (*vishing*) e até mesmo anúncios patrocinados em mecanismos de busca. Essa evolução ampliou o alcance e a eficácia dos ataques, dificultando sua detecção e aumentando o risco para os usuários [Kumaraguru et al. 2010].

Um exemplo típico de fluxo de ataque por *phishing* pode ser visualizado na Figura 1, que evidencia as principais etapas envolvidas em um ataque típico de *phishing*, representando desde o envio inicial de um email malicioso até o acesso indevido à conta da vítima. O fluxo inicia com o atacante disparando uma mensagem que contém um link para uma página fraudulenta, desenvolvida para imitar visualmente o portal legítimo. Ao clicar e inserir suas credenciais, o usuário entrega involuntariamente essas informações ao invasor, que então as utiliza para acessar o sistema verdadeiro. Essa representação gráfica reforça a lógica enganosa por trás desses ataques e como o fator humano é explorado por meio da manipulação de interfaces visuais confiáveis.

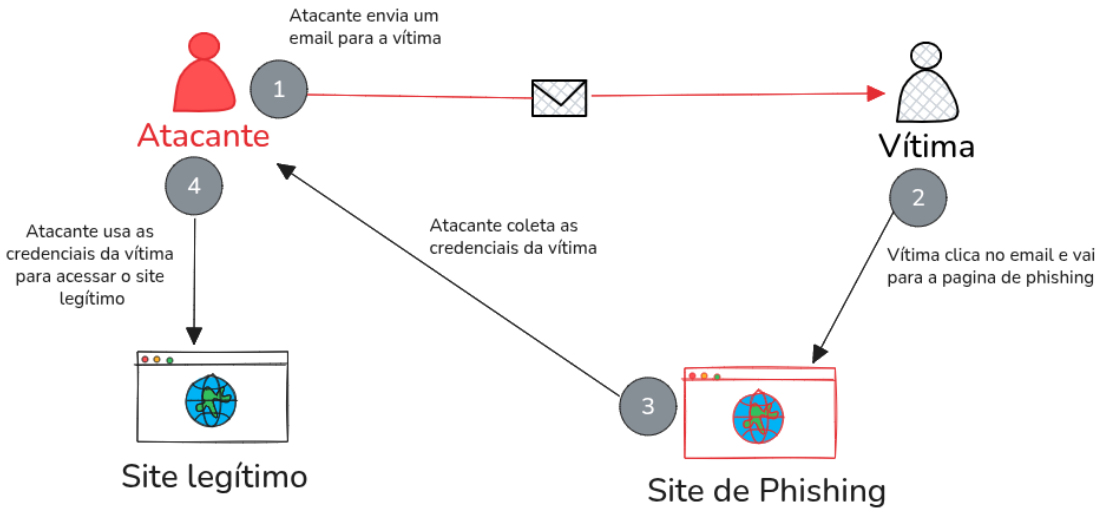

Figura 1. Exemplo de ataque de phishing por simulação visual de páginas legítimas.

Os ataques de *phishing* ocorrem em diversos canais e exploram vulnerabilidades humanas para obter acesso a informações sensíveis. A seguir, são apresentados alguns cenários representativos:

1. *Phishing* via email corporativo
   Funcionários recebem e-mails supostamente enviados por executivos solicitando ações urgentes, como transferências bancárias. O domínio do remetente é visualmente semelhante ao oficial, e os links direcionam para páginas falsas de login interno. O ataque explora engenharia social e senso de urgência.
2. *Phishing* via SMS (*smishing*)
   Mensagens SMS informam bloqueio de conta ou atualização de segurança, contendo links encurtados que redirecionam para páginas falsas de Internet *Banking*. Essas páginas solicitam credenciais pessoais e simulam perfeitamente o visual dos sites legítimos.
3. *Phishing* em redes sociais
   Perfis em plataformas como Instagram recebem mensagens diretas alertando sobre violação de direitos autorais, com link para uma página falsa de verificação. A aparência visual da página simula o portal oficial e coleta credenciais do usuário, permitindo a invasão da conta.
4. *Phishing* via URLs falsas em anúncios patrocinados
   Durante buscas em mecanismos de pesquisa, usuários clicam em anúncios que imitam serviços legítimos. As URLs possuem pequenas variações e levam a páginas clonadas que solicitam dados ou pagamentos indevidos.

Esses casos ilustram como o phishing se adapta a diferentes contextos, aumentando seu alcance e complexidade. A similaridade visual e a simulação de legitimidade tornam esses ataques difíceis de identificar, destacando a importância de soluções automatizadas para sua detecção e bloqueio. Além disso, muitos sites falsos utilizados em ataques de *phishing* são visualmente idênticos aos originais, com *layouts*, logotipos e textos semelhantes, tornando difícil sua identificação por usuários não treinados. Os criminosos utilizam técnicas avançadas para construir páginas fraudulentas que exploram a confiança e a distração dos usuários, o que evidencia a importância de soluções automatizadas para sua detecção.

Uma das formas mais recorrentes de *phishing* envolve o uso de URLs maliciosas, endereços criados para parecer legítimos, mas que redirecionam o usuário para páginas falsas com o objetivo de roubar dados pessoais e financeiros. Esses domínios frequentemente empregam subdomínios confusos, caracteres semelhantes aos de sites confiáveis, além de redirecionamentos automáticos e uso de portas não convencionais para mascarar sua real natureza [Ma et al. 2009].

O impacto do *phishing* é significativo, não apenas pela perda direta de dados e dinheiro, mas também pelos danos à reputação das organizações envolvidas e pela erosão da confiança do público em serviços digitais. Em um cenário onde a transformação digital é acelerada, o combate efetivo ao phishing é essencial para garantir a segurança e a privacidade dos usuários na internet.

Dada a variedade e a complexidade dessas técnicas, a análise automatizada de URLs tem se mostrado uma ferramenta fundamental no combate ao phishing [Ma et al. 2009]. Ao extrair características relevantes de uma URL, como comprimento, uso de símbolos, quantidade de subdomínios e presença de parâmetros suspeitos, sistemas baseados em aprendizado de máquina podem classificar links com alta precisão, permitindo o bloqueio antecipado dessas ameaças antes que alcancem o usuário final.

3.2. Machine Learning

O aprendizado de Máquina (*Machine Learning - ML*) é um campo da inteligência artificial que se baseia em desenvolver algoritmos capazes de aprender padrões e realizar previsões a partir de dados, sem que precisem ser explicitamente programados para cada tarefa específica [Mitchell 1997]. Diferentemente de regras fixas, os modelos de aprendizado de máquina aprendem a partir de exemplos. Após o treinamento, esses modelos conseguem generalizar o conhecimento adquirido, aplicando-o a novos dados para realizar tarefas como classificação, regressão, agrupamento e detecção de anomalias.

O aprendizado de máquina pode ser dividido em três grandes categorias:

- Aprendizado supervisionado: utiliza um conjunto de dados rotulado para treinar o modelo, associando entradas a saídas esperadas. É amplamente aplicado em sistemas de detecção de spam, diagnósticos médicos, reconhecimento de voz e, mais recentemente, em sistemas de detecção de URLs maliciosas [Sarker 2021].
- Aprendizado não supervisionado: explora padrões ocultos em dados sem rótulos, comumente usado para segmentação de clientes, agrupamento e redução de dimensionalidade.
- Aprendizado por reforço: baseado em recompensas e penalidades, onde o algoritmo aprende por meio de interações com o ambiente, como no caso de agentes autônomos e jogos.

No contexto da segurança cibernética, o ML tem se mostrado especialmente promissor, pois possibilita analisar grandes volumes de dados em tempo real, detectando comportamentos maliciosos difíceis de detectar por abordagens tradicionais que muitas vezes escapam de soluções baseadas em regras. Essa capacidade é essencial em um cenário onde novas ameaças surgem constantemente e atacantes ajustam seus métodos para evitar a detecção. Enquanto regras tradicionais e listas negras são eficazes contra ameaças conhecidas, elas falham frente a ataques inovadores ou variantes levemente modificadas.

Modelos de ML oferecem uma abordagem mais adaptativa, permitindo detectar padrões anômalos e características incomuns em URLs, cabeçalhos HTTP, pacotes de rede e outros elementos críticos. Por exemplo, ao extrair características como o comprimento da URL, número de subdomínios, presença de símbolos especiais e padrões nos parâmetros da requisição, os algoritmos conseguem identificar comportamentos típicos de *phishing*, *malware* ou tentativas de exploração.

Algoritmos como *Random Forest*, *Support Vector Machine (SVM)* e *XGBoost* são amplamente utilizados em tarefas de classificação no domínio da segurança. Esses métodos são valorizados por sua capacidade de lidar com dados complexos e de alta dimensionalidade, bem como por sua robustez frente a ruídos e desequilíbrios no conjunto de dados [Buczak and Guven 2016]. Quando combinados com boas práticas de engenharia de atributos (*feature engineering*), esses algoritmos alcançam altos níveis de acurácia, possibilitando a construção de sistemas de defesa cibernética mais eficientes.

A aplicação de *Machine Learning* em segurança da informação representa uma mudança de paradigma: em vez de reagir às ameaças após sua ocorrência, é possível adotar uma postura mais proativa e preventiva, identificando indícios de ataques em estágio inicial e tomando decisões automatizadas para mitigar riscos.

A evolução do aprendizado de máquina tem permitido o desenvolvimento de sistemas de detecção inteligente, capazes de evoluir continuamente à medida que novos dados são coletados. Isso é especialmente relevante em ambientes dinâmicos como a *web*, onde novas URLs maliciosas surgem diariamente com variações sutis e altamente enganosas. O uso dessas tecnologias contribui diretamente para o fortalecimento da segurança digital, tornando-se uma peça central na arquitetura moderna de defesa cibernética. A seguir, serão apresentados três algoritmos supervisionados amplamente utilizados para essa finalidade: *Random Forest*, *Support Vector Machine (SVM)* e *XGBoost*, destacando suas características, vantagens e aplicações na identificação de padrões suspeitos em URLs.

3.2.1. Random Forest

Random Forest é um algoritmo de aprendizado de máquina baseado em ensemble que combina múltiplas árvores de decisão para melhorar a capacidade preditiva e controlar o problema de *overfitting*, comum em modelos baseados em árvores individuais. Esse método constrói várias árvores durante o treinamento, utilizando o método de *bootstrap*, no qual cada árvore é treinada com uma amostra aleatória dos dados, e, em cada nó de decisão, seleciona aleatoriamente um subconjunto de atributos para determinar a melhor divisão [Breiman 2001].

Essa técnica baseia-se no princípio do *bagging*, que propõe que a combinação de múltiplos modelos fracos, como árvores de decisão simples, pode resultar em um modelo forte e mais robusto. No caso de problemas de classificação, o *Random Forest* realiza uma votação majoritária entre as árvores. Para tarefas de regressão, calcula-se a média das previsões de todas as árvores. Essa abordagem reduz a variância do modelo, melhora sua capacidade de generalização e o torna eficaz para lidar com dados de alta dimensionalidade, ruidosos ou com padrões complexos [Liaw and Wiener 2002], conforme ilustrado na Figura 2.

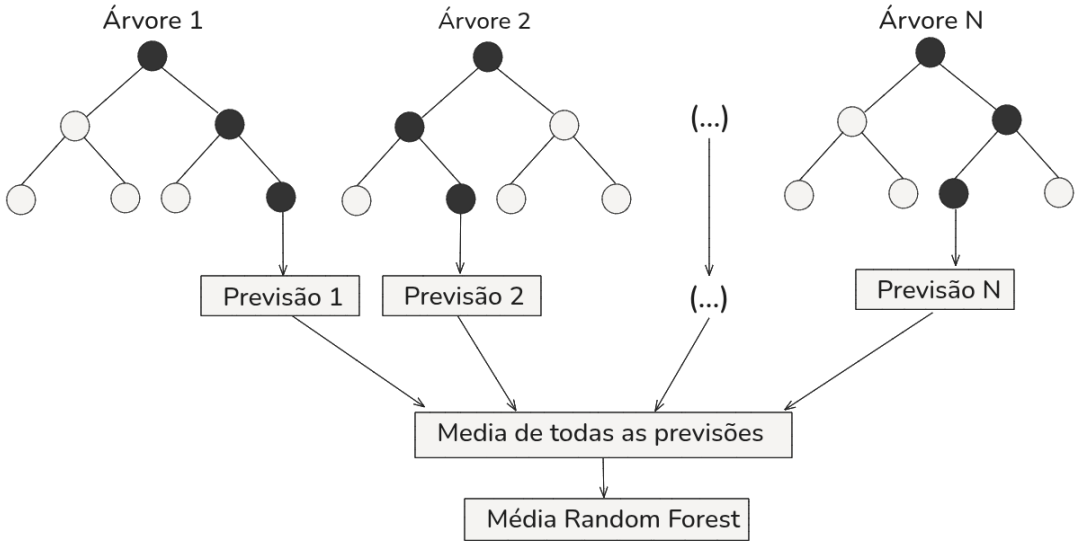

Figura 2. Arquitetura do algoritmo Random Forest.

Um dos principais diferenciais do *Random Forest* é sua resistência ao *overfitting*, fenômeno em que o modelo aprende demasiadamente os dados de treino que passa a memorizar ruídos e exceções, perdendo a capacidade de generalizar para dados novos. Ao limitar o número de atributos avaliados em cada divisão e ao treinar cada árvore com subconjuntos distintos dos dados, o algoritmo consegue capturar relações relevantes sem cair nesse excesso de ajuste.

O *Random Forest* oferece métricas internas de importância das variáveis, o que facilita a interpretação do modelo e a identificação das características mais relevantes para a tomada de decisão. Essa funcionalidade é extremamente útil em aplicações como a análise de URLs maliciosas, permitindo aos pesquisadores e profissionais de segurança entender quais atributos, como número de subdomínios, extensão da URL ou presença de parâmetros, mais contribuem para a identificação de ameaças.

No contexto da segurança da informação, o *Random Forest* é amplamente utilizado em tarefas de detecção de anomalias, identificação de padrões maliciosos, classificação de tráfego de rede e prevenção de ataques baseados em URLs ou payloads maliciosos. Sua capacidade de lidar bem com dados desbalanceados e heterogêneos o torna adequado para cenários nos quais há poucos exemplos de ataques reais e muitos exemplos de tráfego benigno, situação comum em aplicações de detecção de *phishing*, *malware* e desfiguração de páginas.

Outra vantagem prática do algoritmo é sua facilidade de implementação e paralelização. Como as árvores podem ser construídas de forma independente, o treinamento e a inferência podem ser otimizados em ambientes distribuídos ou paralelos, o que reduz o tempo de processamento em aplicações em tempo real.

Por outro lado, o *Random Forest* pode apresentar desempenho inferior quando comparado a algoritmos de aprendizado mais sofisticados, especialmente em contextos sensíveis a pequenas variações nos dados ou em cenários com grandes volumes de informações altamente desbalanceadas. Ainda assim, sua robustez, simplicidade e desempenho consistente fazem dele uma escolha confiável e eficaz para sistemas preditivos voltados à segurança cibernética.

3.2.2. Support Vector Machine (SVM)

Support Vector Machine (SVM) é um algoritmo de aprendizado supervisionado utilizado para tarefas de classificação e regressão, cuja principal função é encontrar o hiperplano ótimo que separa as classes maximizando a margem entre os pontos mais próximos de cada classe, conhecidos como vetores de suporte [Cortes and Vapnik 1995]. Nos cenários aplicados, a compreensão adequada dessa separação é fundamental para garantir a eficácia do modelo em diferentes domínios, como detecção de anomalias e reconhecimento de padrões.

O SVM pode ser implementado em ambientes com alta dimensionalidade, inclusive quando o número de amostras é limitado, o que reforça sua aplicabilidade em áreas com restrições de dados. A margem maximizada contribui para que o modelo tenha uma melhor capacidade de generalização, ou seja, um desempenho mais estável em dados que não foram vistos durante o treinamento, como ilustrado na Figura 3.

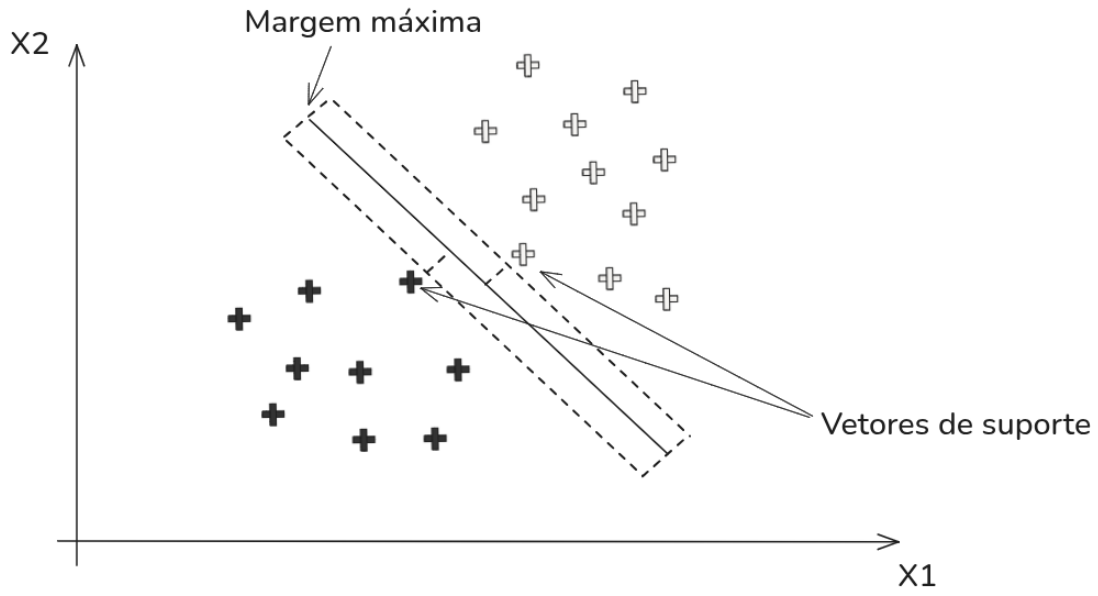

Figura 3. Arquitetura do algoritmo Support Vector Machine (SVM).

Nos casos em que os dados não são linearmente separáveis, o SVM recorre ao uso de funções *kernel*, que realizam uma transformação não linear dos dados para um espaço de maior dimensionalidade, permitindo que a separação entre classes se torne possível. Entre os *kernels* mais utilizados estão o linear, o polinomial, o sigmoide e o radial basis function (RBF), sendo este último amplamente aplicado quando não há conhecimento prévio sobre a distribuição dos dados [Sarker 2021].

Uma das grandes vantagens do SVM é sua eficiência em contextos de alta dimensionalidade, mesmo quando a quantidade de amostras disponíveis é limitada. O algoritmo é especialmente eficaz em cenários onde existe uma separação clara entre as classes. Além disso, o SVM é menos suscetível ao problema da maldição da dimensionalidade e pode ser ajustado com regularização para evitar *overfitting*, controlando o equilíbrio entre maximizar a margem e minimizar o erro de classificação. Por essas razões, o SVM tem sido amplamente aplicado em áreas como reconhecimento de padrões, classificação de texto, diagnóstico médico e detecção de fraudes. No campo da segurança da informação, destaca-se como uma ferramenta eficaz na detecção de ameaças cibernéticas, como *phishing*, *malware* e URLs maliciosas, devido à sua capacidade de identificar fronteiras complexas entre comportamentos benignos e maliciosos, mesmo com conjuntos de dados desbalanceados ou com padrões sutis [Zhao et al. 2021a].

A aplicação do *Support Vector Machine* em sistemas de análise de URLs envolve a extração de diversas características dos endereços, como comprimento da URL, presença de IPs ou caracteres suspeitos, e a sua utilização como variáveis de entrada para o modelo. Essas variáveis refletem comportamentos recorrentes de ataques, auxiliando na identificação de padrões maliciosos com maior precisão. A partir disso, o SVM é treinado para aprender os padrões associados a URLs legítimas e maliciosas, sendo capaz de classificar novas instâncias com elevada precisão, desde que os parâmetros do modelo (como o tipo de *kernel*e o coeficiente *gamma*) estejam bem ajustados.

Apesar de seu bom desempenho, o SVM pode apresentar dificuldade de escalabilidade quando aplicado a conjuntos de dados muito grandes, já que seu custo computacional aumenta significativamente com o número de amostras. Ainda assim, para aplicações em que a precisão e a robustez são prioritárias, como no seu caso, com foco na detecção de sites maliciosos, o SVM continua sendo uma opção bastante relevante [Zhao et al. 2021a].

3.2.3. XGBoost

XGBoost (*Extreme Gradient Boosting*) é um algoritmo de aprendizado de máquina baseado em árvores de decisão que utiliza o método de *boosting* para melhorar a precisão dos modelos. Diferentemente de abordagens como o *Random Forest*, que treinam múltiplas árvores de forma independente, o *XGBoost* constrói árvores de maneira sequencial, onde cada nova árvore é treinada para corrigir os erros cometidos pelas anteriores, com maior foco nas instâncias mais difíceis de classificar [Chen and Guestrin 2016].

A técnica de *boosting* tem como objetivo transformar modelos fracos em modelos fortes, agregando o conhecimento de várias árvores simples para criar um preditor mais robusto. Em cada etapa, o *XGBoost* minimiza uma função de perda diferenciável com base em gradientes, construindo novas árvores que corrigem os erros cometidos pelas anteriores. Dessa forma, o modelo passa a focar progressivamente nas amostras mais difíceis de classificar, resultando em um preditor final mais preciso e adaptativo. Esse processo é representado na Figura 4, que ilustra como cada árvore contribui para melhorar o desempenho do modelo: as árvores iniciais geram previsões básicas, enquanto as posteriores refinam os resultados ao corrigir erros específicos, culminando em uma saída final mais precisa e confiável.

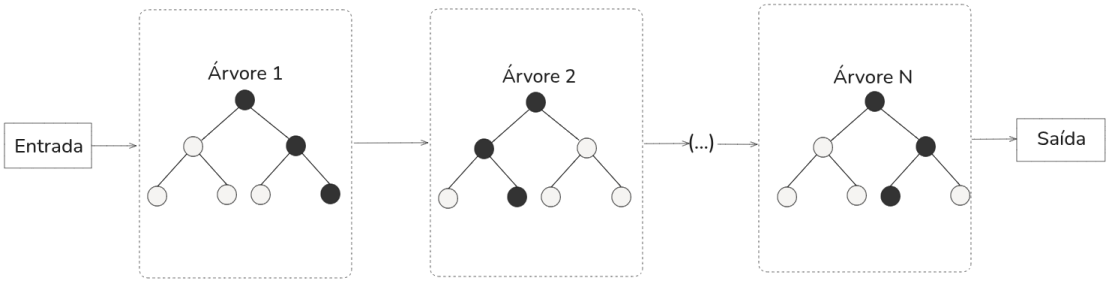

Figura 4. Arquitetura do algoritmo XGBoost.

Além do alto desempenho preditivo, o *XGBoost* foi projetado com foco em eficiência computacional, possibilitando o treinamento rápido mesmo com grandes volumes de dados. Isso é possível graças à implementação de recursos como paralelização das árvores, poda em profundidade inteligente, e tratamento eficiente de dados esparsos. Essas características o tornaram amplamente utilizado em aplicações práticas.

Outro diferencial do *XGBoost* é o uso de regularização L1 e L2 para controlar a complexidade do modelo, o que ajuda a evitar *overfitting*, principalmente em bases de dados ruidosas ou desbalanceadas [Natekin and Knoll 2013]. Essa capacidade de controlar o sobreajuste, aliada à flexibilidade na escolha de funções de perda e métricas de avaliação, torna o algoritmo altamente ajustável a diferentes tipos de problemas.

O *XGBoost* também fornece métricas de importância de atributos, permitindo a análise de quais variáveis têm maior impacto na decisão do modelo. Isso é particularmente útil em projetos de segurança da informação, onde entender os fatores que indicam uma ameaça é essencial para a construção de sistemas mais transparentes e explicáveis.

No contexto da detecção de URLs maliciosas, o *XGBoost* tem se destacado por sua capacidade de capturar padrões complexos e não lineares entre os atributos extraídos das URLs, como presença de IPs, número de diretórios, comprimento total, uso de HTTPS, entre outros. Devido à sua robustez, é especialmente eficaz em cenários com dados desbalanceados, como é o caso da segurança cibernética, onde a quantidade de URLs legítimas costuma ser muito superior à de URLs maliciosas [Natekin and Knoll 2013].

Estudos mostram que o *XGBoost* frequentemente supera outros algoritmos, como *SVM* ou *Random Forest*, em termos de acurácia e sensibilidade, especialmente quando bem ajustado com técnicas de validação cruzada e tuning de hiperparâmetros [Zhao et al. 2021b]. Além disso, sua flexibilidade para personalização, aliada à escalabilidade, o torna uma excelente escolha para sistemas preditivos em tempo real, como filtros automáticos de phishing ou motores de recomendação seguros.

Apesar de suas inúmeras vantagens, o *XGBoost* pode ser sensível à escolha dos hiperparâmetros, e seu processo de treinamento tende a ser mais complexo, exigindo maior atenção à validação dos resultados para evitar viés de seleção ou overfitting sutil. No entanto, com práticas adequadas de tuning e validação, esse algoritmo oferece um excelente equilíbrio entre performance, robustez e interpretabilidade.

O *XGBoost* consolida-se como uma ferramenta robusta e eficaz no conjunto de algoritmos de aprendizado de máquina, especialmente em aplicações voltadas à segurança digital. Sua utilização na detecção automatizada de URLs maliciosas evidencia o potencial das soluções orientadas por dados na mitigação de ameaças cibernéticas, contribuindo significativamente para a proteção de usuários e sistemas frente a ataques cada vez mais sofisticados.

3.2.4. Python

A linguagem *Python* tem se consolidado, nas últimas décadas, como uma das mais populares e versáteis no desenvolvimento de soluções computacionais, especialmente nas áreas de ciência de dados, aprendizado de máquina e automação. *Python* é uma linguagem de alto nível, interpretada e de tipagem dinâmica, cujo principal diferencial está na simplicidade de sua sintaxe, que favorece a legibilidade do código e acelera o processo de desenvolvimento [van Rossum and Drake 2009]. Esse conjunto de características também contribui para sua ampla adoção tanto por iniciantes quanto por profissionais experientes.

No contexto de projetos baseados em dados, *Python* possui um ecossistema robusto de bibliotecas que facilitam desde a aquisição e pré-processamento até a modelagem e visualização dos dados. Dentre as bibliotecas mais utilizadas, destacam-se o `Pandas` para manipulação e análise de dados estruturados, `NumPy` para operações numéricas de alto desempenho, `Matplotlib` e `Seaborn` para visualização gráfica, além do `Scikit-learn`, que fornece uma ampla gama de algoritmos de aprendizado supervisionado e não supervisionado [Pedregosa et al. 2011].

Para técnicas mais avançadas, como *boosting* e *deep learning*, o *Python* também oferece bibliotecas especializadas como `XGBoost`, `TensorFlow` e `PyTorch`, que permitem a construção de modelos altamente performáticos com grande flexibilidade. Essas ferramentas tornam *Python* uma linguagem adequada para o desenvolvimento de sistemas inteligentes, como o proposto neste trabalho, que envolve a detecção automática de URLs maliciosas com base em características extraídas dos endereços.

Dessa forma, a escolha do *Python* neste projeto não se dá apenas por sua popularidade, mas principalmente por sua eficiência, simplicidade e ampla adoção na comunidade de ciência de dados, sendo um componente essencial para a execução das etapas de manipulação, modelagem e avaliação dos modelos de aprendizado de máquina utilizados.

3.2.5. Streamlit

O *Streamlit* é uma biblioteca de código aberto desenvolvida em *Python*, projetada para facilitar a criação rápida de aplicações *web* interativas, especialmente em contextos voltados à ciência de dados, aprendizado de máquina e análise exploratória. Sua principal característica é a possibilidade de construir interfaces gráficas de maneira declarativa e intuitiva, utilizando exclusivamente comandos em *Python*, eliminando a necessidade de conhecimentos específicos em tecnologias de *front-end*, como HTML, CSS ou *JavaScript*.

A estrutura do *Streamlit* permite que cada elemento da interface, como gráficos, tabelas, botões, caixas de seleção e campos de entrada, seja definido diretamente no código, promovendo uma experiência de desenvolvimento fluida. Além disso, os scripts são executados de forma sequencial e a interface é atualizada automaticamente sempre que o código é modificado, o que favorece um processo iterativo e ágil de prototipação.

Entre suas funcionalidades mais relevantes destacam-se o suporte a dados tabulares, a renderização de gráficos interativos, o carregamento de arquivos, e a integração nativa com bibliotecas amplamente utilizadas no ecossistema *Python*, como *Pandas*, *NumPy*, *Matplotlib*, *Plotly* e *Scikit-learn*. A biblioteca também disponibiliza recursos para implantação em nuvem, permitindo que aplicações desenvolvidas sejam facilmente compartilhadas por meio de um link acessível, sem necessidade de configuração de servidores complexos.

4. Abordagem da Proposta

Nesta seção, descrevemos a abordagem adotada para o desenvolvimento da solução proposta. São detalhadas as etapas de seleção e preparação dos dados, transformação das variáveis, escolha das features e demais procedimentos utilizados para construção do modelo de classificação de URLs maliciosas.

4.1. Seleção de Dados

Para o desenvolvimento do modelo de detecção de URLs maliciosas, foram utilizadas duas fontes principais: o *Malicious URLs Dataset*, disponível na plataforma Kaggle, e a base em tempo real do *PhishTank*. O primeiro fornece mais de 650 mil registros rotulados como benigno, *phishing*, desfiguração de páginas ou *malware*, em formato CSV, com colunas url e *type*. O outro contribuiu com amostras recentes de phishing validadas colaborativamente por usuários da comunidade.

A partir da combinação dessas duas bases, foi gerada uma terceira fonte de dados, que constitui o principal conjunto utilizado neste projeto. Esse novo *dataset*, construído com o objetivo de otimizar o desempenho da tarefa de classificação binária, apresenta URLs rotuladas como *good* (benignas) e *bad* (maliciosas). A simplificação na estrutura das classes favorece uma abordagem mais eficiente e robusta na etapa de treinamento do modelo. No total, o conjunto final contém aproximadamente 2.000 URLs válidas, garantindo uma base significativa para o treinamento e avaliação dos algoritmos.

Como critério, foram mantidas apenas URLs únicas, válidas e ativas. Após o processo de limpeza e balanceamento, os dados foram preparados para extração de atributos como: estrutura léxica, informações de domínio, DNS, tempo de resposta e redirecionamentos. Essa preparação gerou a base final de entrada para os algoritmos de aprendizado de máquina.

Para garantir a validação consistente e confiável do desempenho dos modelos, os dados foram divididos em 80% para treinamento e 20% para teste, utilizando a biblioteca *Scikit-learn*. Essa divisão assegura uma avaliação justa e realista dos resultados obtidos durante a fase de experimentação.

4.2. Pré-processamento

Após a coleta das URLs, foi conduzido um processo criterioso de limpeza e padronização dos dados para garantir a consistência e qualidade do conjunto. Inicialmente, URLs duplicadas foram removidas e registros com valores ausentes, inválidos ou com erros de formatação foram descartados. Em seguida, aplicou-se uma verificação de acessibilidade utilizando uma rotina automatizada que testa a conexão e o tempo de resposta dos sites, descartando URLs que apresentem erros de rede, SSL ou tempo limite, com o objetivo de preservar apenas aquelas efetivamente funcionais e ativas.

Durante a etapa de extração de características, atributos inconsistentes ou não coletados por falhas de rede ou DNS foram tratados por meio de codificação padrão, como o valor -1. Para aprimorar a qualidade do conjunto de dados, foi implementada uma validação estruturada de *features*, baseada em um conjunto mínimo de atributos considerados essenciais para a análise, como tempo para expiração do domínio, país de origem do IP, número de redirecionamentos, presença em listas RBL, entre outros. URLs que não apresentassem ao menos 70% desses atributos válidos foram descartadas, garantindo robustez na representação de cada amostra.

Para otimizar o processo de coleta e extração das *features*, foi utilizado o sistema de *threads* da linguagem *Python*, permitindo a execução paralela de múltiplas requisições e consultas, reduzindo significativamente o tempo total necessário para processar o conjunto de dados.

Além disso, com o objetivo de garantir a segurança e a reprodutibilidade dos testes com URLs potencialmente maliciosas, toda a aplicação foi encapsulada em contêineres utilizando Docker. A configuração composta por um `Dockerfile` e um `docker-compose.yml` define um ambiente controlado, assegurando o isolamento da aplicação em relação ao sistema operacional do host. Esse isolamento é especialmente relevante ao lidar com páginas maliciosas, pois previne que comportamentos inesperados ou acessos arriscados afetem o ambiente local. O uso de Docker também facilita a

replicação do ambiente, além de garantir que os testes e as coletas ocorram sob as mesmas condições, favorecendo a integridade das validações.

Variáveis categóricas, como o tipo da URL (por exemplo, *phishing*, benigna), foram convertidas em valores binários (0 ou 1), permitindo a aplicação direta de algoritmos de aprendizado de máquina. Os dados numéricos extraídos, como tempo de resposta, idade do domínio e número de redirecionamentos, foram normalizados por meio de técnicas estatísticas para reduzir a variância entre escalas e assegurar maior estabilidade durante o treinamento dos modelos.

4.3. Transformação

Nesta etapa, as URLs passaram por um processo sistemático de transformação com o objetivo de convertê-las em vetores de atributos numéricos e categóricos compatíveis com algoritmos supervisionados de aprendizado de máquina. Para isso, foram empregadas técnicas de *parsing*, extração de metadados e análise sintática e semântica, utilizando bibliotecas especializadas e chamadas externas.

Inicialmente, aplicou-se a biblioteca tldextract para decompor a URL em suas partes constituintes: subdomínio, domínio e sufixo. Essa separação permitiu a extração de atributos como a quantidade de subdomínios, a presença de hífens e avaliação do comprimento do domínio, sendo este último útil na identificação de domínios muito curtos, considerados suspeitos por estarem associados a práticas fraudulentas. Também foi realizada a verificação da presença de palavras suspeitas diretamente na string da URL, utilizando listas de vocabulário especializado.

No que se refere à análise estrutural da URL, foram aplicadas rotinas de inspeção da *query string* por meio do módulo urllib.parse, permitindo a identificação de parâmetros suspeitos, contagem de parâmetros de consulta e verificação de portas não convencionais que fogem aos padrões HTTP e HTTPS.

Para análise da conectividade e comportamento da URL em tempo real, foi utilizada a biblioteca requests, em conjunto com funções de tratamento, que resolve redirecionamentos e retorna o destino final da URL. A partir dela, foi possível calcular o número de redirecionamentos e o tempo de resposta do servidor, medido em milissegundos. URLs com alto tempo de resposta ou múltiplos redirecionamentos foram marcadas como potencialmente problemáticas, por refletirem instabilidade ou tentativas de ocultar o destino real.

Recursos de rede como *socket* e *dns.resolver* foram empregados para obtenção de IPs associados aos domínios, permitindo a consulta em listas negras (RBLs) e análise da reputação do IP. Por meio de requisições à API pública *ipinfo.io*, identificou-se também a procedência geográfica do IP, destacando países com recorrência de fraudes como indicadores de risco adicional.

A biblioteca *whois* foi utilizada para consultar dados do registro de domínio, extraindo atributos como idade do domínio e tempo restante até sua expiração. Domínios muito recentes ou com prazo de validade curto foram considerados menos confiáveis, por apresentarem perfil típico de uso em golpes temporários.

Adicionalmente, foram aplicadas verificações quanto à presença de protocolo HTTPS, à indexação do site nos mecanismos de busca, ao uso de encurtadores de URL e

ao formato do domínio, especialmente quando representado diretamente por um endereço IP. Também foi integrada uma função que consulta a base do *Google Safe Browsing*, identificando URLs potencialmente perigosas conforme registros públicos de ameaças.

A partir das etapas descritas, cada URL foi convertida em um vetor de atributos composto por informações léxicas, estruturais, contextuais e reputacionais. Esses atributos foram extraídos automaticamente por meio das diversas técnicas aplicadas, como análise da estrutura da URL, comportamento de conexão, reputação do IP e dados de registro do domínio. O resultado final é um conjunto de representações que fornece uma base sólida e confiável para os modelos de aprendizado supervisionado, permitindo a classificação binária entre URLs legítimas e maliciosas com maior precisão e eficiência.

4.4. Escolha das Features

A seleção das *features* utilizadas neste trabalho foi guiada por estudos prévios relevantes [Ayres et al. 2019b, Ma et al. 2009], além de diretrizes técnicas e referências de especialistas em segurança cibernética. As variáveis foram agrupadas em três grandes categorias: léxicas, de conectividade, e contextuais. Cada verificação implementada foi incorporada ao *pipeline* de pré-processamento, com limiares definidos (*thresholds*) e validações específicas para garantir a robustez e eficiência do modelo.

4.4.1. Características Léxicas

As características léxicas analisam os elementos visuais e textuais presentes diretamente na estrutura da URL, como número de caracteres, tipo de símbolos utilizados, presença de palavras-chave suspeitas ou padrões de codificação. Essas métricas são relevantes pois atacantes costumam manipular a forma da URL para confundir o usuário, disfarçar o destino real ou burlar sistemas automatizados de detecção. A Tabela 1 apresenta as verificações aplicadas com base nesse tipo de característica.

Tabela 1. Características léxicas da URL.

Nome da Verificação	Método de Coleta
Domínio curto	Extrai o domínio principal via parsing e verifica se o comprimento da string ≤ 3 caracteres.
Presença do símbolo "@"	Realiza verificação direta por correspondência de caractere na string da URL.
Palavras suspeitas na URL	Faz varredura por palavras-chave suspeitas usando comparação por substring em minúsculas.
Uso incomum de "//"	Aplica substituição do protocolo e busca por ocorrência adicional de "//"via análise de string.
Hífen no nome do domínio	Realiza parsing do domínio e verifica a presença do caractere "hífen"por correspondência direta.
Quantidade de subdomínios	Segmenta os subdomínios por ponto e conta os níveis utilizando split da string.
Número de parâmetros na URL	Faz parsing da query string e contabiliza o número de parâmetros extraídos.
Parâmetros de consulta suspeitos	Avalia os nomes dos parâmetros extraídos via parsing e verifica correspondência com lista de palavras-chave.

- Domínio curto: Avalia o comprimento do nome de domínio (excluindo subdomínios e o TLD). Domínios muito curtos, aqui definidos como até 3 caracteres, podem ser tratados como suspeitos em certos contextos por sua raridade e potencial de serem usados em ataques de *phishing*. São fáceis de memorizar, digitar e podem simular visualmente domínios legítimos, sendo úteis para criar URLs enganosas, como em encurtadores maliciosos, serviços temporários ou campanhas de phishing [Ma et al. 2009].
- Presença do caractere "@" na URL: URLs que contêm o símbolo "@" podem levar navegadores a ignorar o que vem antes dele, acessando diretamente o trecho posterior. Trata-se de uma técnica clássica em ataques de *phishing*, usada para esconder o destino real e simular autenticidade. Um exemplo comum dessa técnica é o uso de URLs como `https://www.legitimo.com@malicioso-site.com/login`, que exibem visualmente um domínio confiável antes do caractere "@", mas na verdade direcionam o usuário ao site malicioso. Essa abordagem explora o comportamento dos navegadores e facilita golpes de phishing.
- Palavras suspeitas na URL: Verifica se a URL contém termos como "login", "secure", "verify" ou "update". Essas palavras são comuns em tentativas de enganar o usuário, simulando páginas de autenticação ou atualização [Kaspersky 2023a]. Um caso comum envolve URLs como `http://secure-login.account-verification.com`, onde termos como "secure" e "login" são empregados para simular legitimidade e segurança. Essa composição é típica em ataques de *phishing* que imitam páginas bancárias ou de *e-commerce*.
- Uso indevido de barras duplas ("//"): URLs que apresentam o símbolo "//" fora do protocolo (após "https://") podem estar utilizando técnicas de disfarce visual ou redirecionamento malicioso. Esse padrão é frequentemente observado em links ocultos ou manipulados [Trustwave 2022]. O navegador interpreta tudo após o domínio como parte do caminho da URL. Assim, o uso de barras duplas após o protocolo, como em `https://exemplo.com//paypal.com`, não redireciona para "paypal.com", mas continua dentro do domínio "exemplo.com". A aparência da URL engana visualmente, mas o destino real permanece oculto.
- Hífen no nome do domínio: Embora permitido, o uso de hífens no domínio é incomum em sites legítimos. Ele é frequentemente usado para imitar nomes populares, como "pay-pal.com" no lugar de "paypal.com", prática associada ao *typosquatting* [Kaspersky 2023b].
- Quantidade de subdomínios: Domínios que possuem mais de três subdomínios são considerados suspeitos. Essa técnica pode ser usada para camuflar o domínio principal ou para confundir a leitura na barra de endereço. Uma URL como `login.security.client.bank.example.com` contém cinco subdomínios encadeados, o que pode dificultar a leitura do endereço e mascarar o domínio principal (`example.com`). Essa estrutura é frequentemente usada para simular legitimidade e esconder o real destino em campanhas de *phishing*.
- Número de parâmetros na URL: URLs com mais de cinco parâmetros de consulta na *query string* podem indicar redirecionamentos encobertos, rastreamento abusivo ou tentativas de mascarar comportamentos maliciosos [Sitebulb 2023]. Uma URL como

`https://check-secure.com/?id=4356&token=fa9b2&user=guest&track=ad72&redirect=home&ref=phish&session=x812` apresenta mais de cinco parâmetros de consulta. Essa complexidade pode esconder redirecionamentos indesejados, rastreamento abusivo ou tentativas de captura de dados sensíveis, como *token* e *session*, em páginas falsas com aparência confiável.

- Parâmetros de consulta suspeitos: Detecta a presença de parâmetros como "id", "session", "user" ou "login" na URL. Eles são fortemente associados a tentativas de coleta de dados sensíveis ou simulações de sessões de login fraudulentas. A URL `https://auth-check.com/?id=21&login=user1&session=9a3z` é um exemplo que contém múltiplos parâmetros ligados a autenticação, como *id*, *login* e *session*. Essa composição é típica de páginas falsas que simulam processos de login para capturar dados confidenciais do usuário em ataques de *phishing*.

4.4.2. Características de Conectividade

As características de conectividade avaliam propriedades da rede envolvida no acesso à URL, como uso de protocolos seguros, tempo de resposta do servidor, redirecionamentos intermediários e uso de portas não convencionais. Tais atributos refletem o comportamento da infraestrutura que hospeda o conteúdo, sendo úteis para identificar práticas incomuns, instabilidades ou padrões associados a campanhas maliciosas. A Tabela 2 resume as verificações que exploram esse aspecto técnico da URL.

Tabela 2. Coleta de características de conectividade.

Nome da Verificação	Método de Coleta
Uso de porta não padrão	Aplica parsing na URL para extrair o esquema e a porta, verificando se a porta é diferente das convencionais 80 (HTTP) ou 443 (HTTPS).
Uso de encurtadores de URL	Extrai o domínio registrado via parsing e verifica sua presença em uma lista de serviços de encurtamento conhecidos.
Presença do protocolo HTTPS	Verifica se a URL inicia com o prefixo "https://" por correspondência direta de string.
Tempo de resposta elevado	Mede o tempo de resposta de uma requisição HTTP e compara com um limiar definido em milissegundos.
Muitos redirecionamentos	Realiza requisição HTTP com redirecionamento habilitado e contabiliza o número de redirecionamentos via histórico de resposta.
Domínio é um endereço IP	O campo de hostname da URL é avaliado para identificar se representa diretamente um endereço IP válido, em vez de um nome de domínio tradicional.

- Uso de porta não padrão: Domínios que operam em portas diferentes das padrão, 80 (HTTP) e 443 (HTTPS), podem estar utilizando configurações incomuns como 8080 ou 8443, com objetivo de ocultar serviços ou dificultar a detecção automática. Um exemplo seria a URL `https://secure-check.com:8443/update`, que utiliza a porta 8443, frequentemente associada a servidores de testes ou ambientes não oficiais. O uso

dessa porta pode sinalizar uma tentativa de desvio de protocolos legítimos ou a operação de infraestrutura obscura.

- Uso de encurtadores de URL: Verifica se o domínio pertence a serviços de encurtamento, como `bit.ly`, `cutt.ly` ou `tinyurl`. Esses serviços dificultam a visualização do destino final e são frequentemente usados por atacantes para disfarçar URLs maliciosas. Uma URL como `https://bit.ly/3xU1oPq` pode redirecionar para um site malicioso, mas sem revelar o destino final ao usuário.
- Presença do protocolo HTTPS: URLs legítimas costumam utilizar o protocolo seguro HTTPS, que exige um certificado válido. A ausência desse protocolo pode indicar falta de segurança mínima ou a tentativa de mascarar uma conexão insegura. Uma URL como `http://user-update.com/login` utiliza o protocolo HTTP sem criptografia, o que torna a comunicação vulnerável à interceptação. Essa ausência do HTTPS pode indicar um site malicioso tentando simular uma página de autenticação legítima, sem oferecer segurança real na transmissão de dados sensíveis como senhas ou informações pessoais.
- Tempo de resposta elevado: URLs cujo servidor demora mais de 1000 milissegundos para responder são tratadas como suspeitas [Odown 2023]. Tempos altos podem indicar infraestruturas instáveis, temporárias ou mal configuradas, comuns em sites de baixa reputação.
- Múltiplos redirecionamentos: URLs que redirecionam o usuário diversas vezes antes de chegar ao destino final podem estar ocultando o endereço real ou executando manipulações indesejadas durante o processo de navegação. Uma URL como `https://click-track.net/redirect?target=https://login-update.com/ref?continue=https://phish-final.com` pode envolver múltiplos redirecionamentos encadeados entre diferentes domínios. Esse padrão é usado para mascarar o destino final, contornar sistemas de segurança automatizados e confundir o usuário, dificultando a identificação de comportamentos maliciosos presentes na etapa final do carregamento.
- Domínio em formato IP: Detecta se o domínio é representado diretamente por um endereço IP numérico, em vez de um nome. Essa prática é comum entre sites maliciosos, pois evita rastreamento por mecanismos baseados em domínios. Um exemplo é a URL `http://192.185.217.116/login` que utiliza um endereço IP numérico em vez de um nome de domínio. Isso pode indicar tentativa de ocultação, uma vez que mecanismos de reputação também são baseados em nomes.

4.4.3. Características Contextuais

As características contextuais consideram informações complementares que cercam a URL, como reputação pública do domínio, idade de registro, localização geográfica do IP e indexação em mecanismos de busca. Diferente dos aspectos visuais ou técnicos imediatos, essas métricas ajudam a compreender o histórico e o ambiente da URL, permitindo uma análise mais robusta e contextualizada da sua confiabilidade. As verificações baseadas nesse grupo estão organizadas na Tabela 3.

Tabela 3. Coleta de características contextuais da URL.

Nome da Verificação	Método de Coleta
Domínio listado como malicioso	Resolve o domínio para IP e verifica sua presença em servidores DNS de listas negras (RBL) por meio de consulta reversa.
IP de país não confiável	Resolve o IP do domínio e consulta sua geolocalização via `IPinfo.io` e avalia se está em lista de países não confiáveis.
Indexação no Google	Realiza busca no Google usando operador `site:` e verifica se o domínio retorna resultados válidos.
Domínio recém-criado	Obtém a data de criação do domínio via consulta WHOIS e calcula a idade em dias e compara com limiar mínimo (180 dias).
Domínio próximo da expiração	Obtém a data de expiração do domínio via WHOIS e calcula dias restantes até o vencimento e compara com limiar mínimo (90 dias).
Verificação no Google Safe Browsing	Realiza consulta na API do Google Safe Browsing com a URL final resolvida para verificar alerta de risco.

- Presença em listas negras (RBLs): Verifica se o domínio ou endereço IP está presente em bases públicas que mantêm registros associados a *spam, malware* ou *phishing*. A inclusão em uma dessas listas é um forte indicativo de ameaça. Entre os servidores mais utilizados destacam-se:
  - `zen.spamhaus.org`: agrega diversas listas da *Spamhaus*, incluindo registros de envio de *spam, botnets* e atividades maliciosas recentes.
  - `bl.spamcop.net`: monitora IPs que participaram do envio de *spam*, permitindo filtragem em tempo real de tráfego suspeito.

  URLs como `http://malicious-site.net` podem estar associadas a IPs registrados nessas listas negras, o que reforça o alerta de reputação negativa.
  A consulta é feita por meio de verificação DNS reversa. Para isso, inverte-se a ordem dos octetos do IP suspeito e concatena-se ao domínio RBL:

  ```
  nslookup 116.217.185.192.zen.spamhaus.org
  nslookup 116.217.185.192.bl.spamcop.net
  ```

  Se o IP estiver listado, o servidor retorna um endereço de resposta indicando o tipo de ameaça. Caso contrário, a resposta "not found" indica que o IP não está presente na base consultada.
- Localização geográfica suspeita do IP: Avalia se o endereço IP da URL está localizado em países com histórico elevado de fraudes ou abuso de infraestrutura. Essa métrica foi baseada em estudos de detecção de ameaças em redes nacionais [Costa et al. 2021]. Um exemplo é a URL `http://secure-login.net`, ao ser resolvida para o IP `192.0.2.45`, possui localização geográfica nos Estados Unidos (US). Como o país está incluído na lista de regiões com histórico elevado de abuso de infraestrutura digital [Ayres et al. 2019a], a URL é classificada como suspeita. Essa métrica reforça o risco quando combinada com outros fatores como ausência de HTTPS ou presença em listas negras.
- Indexação pelo Google: Domínios legítimos tendem a estar indexados pelo Google. A ausência de indexação pode indicar que o site é recente, oculto ou contém conteúdo malicioso que não é rastreado por mecanismos de busca. Esse tipo de verificação pode ser feita utilizando o operador `site:` no Google, como por exemplo: `site:secure-authentication-xjs.com`

- Idade do domínio: Domínios com menos de 180 dias de registro são mais suscetíveis a uso malicioso, por não possuírem histórico de reputação. Campanhas de phishing e golpes automatizados fazem uso frequente de domínios recém-criados [NordVPN 2023]. Essa verificação pode ser feita consultando a data de criação via WHOIS, ou com serviços semelhantes. A combinação da baixa idade com outras características suspeitas pode reforçar uma classificação de risco no seu sistema de análise.
- Tempo restante para expiração do domínio: Verifica se o domínio está próximo do vencimento. Domínios com menos de 90 dias para expirar tendem a ser descartáveis e indicam pouca intenção de manutenção, prática típica de atacantes [SerpNames 2023]. Essa métrica também pode ser obtida por meio de consulta WHOIS, e combinada com outros fatores para reforçar uma classificação de risco.
- Verificação via *Google Safe Browsing*: Identifica se uma URL está marcada como perigosa na base do *Google Safe Browsing*. Essa verificação é essencial para detectar links conhecidos por hospedar *phishing*, malware ou outras ameaças online. A título de exemplo, a URL `http://update-authentication.net` é marcada como perigosa pelo serviço *Google Safe Browsing*. Essa classificação indica que o endereço foi identificado como fonte de *phishing* ou malware, oferecendo risco direto ao usuário que o acessa. Essa verificação pode ser feita por meio da API do *Google Safe Browsing* ou por plataformas que integram esse serviço, como navegadores, serviços de email ou ferramentas de análise de reputação de URLs.

5. Resultados

Nesta seção, são apresentados os resultados obtidos a partir dos experimentos realizados com os modelos de aprendizado de máquina aplicados à tarefa de detecção de URLs maliciosas. Para garantir uma avaliação mais justa entre as classes, foram aplicadas técnicas de balanceamento durante o treinamento dos modelos, reduzindo o impacto da distribuição desigual de amostras. O objetivo é avaliar o desempenho dos algoritmos selecionados (*Random Forest, XGBoost e Support Vector Machine*) por meio de métricas clássicas de classificação binária. O resultado do modelo escolhido para análise da URL é apresentados por meio de uma interface web interativa, desenvolvida com a ferramenta *Streamlit*

5.1. Métricas de Avaliação

Esta seção apresenta e descreve as principais métricas utilizadas para avaliação dos modelos de classificação implementados no projeto. As métricas selecionadas foram: acurácia, precisão, revocação (*recall*), F1-score e matriz de confusão.

- Precisão (*Precision*): Avalia a proporção de predições positivas corretas, ou seja, quantos dos exemplos classificados como “phishing” realmente pertencem a essa classe.
- Revocação (*Recall*): Mede a capacidade do modelo de identificar corretamente os exemplos positivos reais, sendo particularmente relevante em problemas de segurança, como a detecção de *phishing*.
- *F1-Score*: Representa a média harmônica entre precisão e revocação, equilibrando ambas as métricas. É especialmente útil em cenários com desbalanceamento entre as classes.

- Acurácia (*Accuracy*): Expressa a proporção de acertos totais entre todas as previsões feitas. Embora amplamente utilizada, pode ser pouco representativa em cenários com classes desproporcionais.
- Matriz de Confusão: Estrutura tabular que evidencia os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, oferecendo uma visão detalhada do desempenho do modelo por classe.

5.2. Comparação entre Modelos

Com o intuito de identificar o modelo com melhor desempenho para a tarefa de detecção de sites maliciosos, foram avaliados três algoritmos amplamente utilizados: *Random Forest, XGBoost e Support Vector Machine (SVM)*. A Tabela 4 resume as principais métricas obtidas durante os testes. A seguir, são apresentados os resultados individualmente.

Tabela 4. Resumo das métricas de avaliação para cada modelo.

Modelo	Acurácia	Precisão	Recall	F1-score
Random Forest	76%	0,76	0,74	0,75
XGBoost	67%	0,73	0,63	0,68
SVM	71%	0,69	0,84	0,76

5.2.1. Random Forest

O modelo *Random Forest* apresentou uma acurácia global de 76%, com *F1-score* de 0,75 para a classe "phishing". A matriz de confusão, ilustrada na Figura 5, indica que o modelo identificou corretamente 103 casos positivos e 203 negativos, cometendo 65 falsos negativos e 32 falsos positivos. O desempenho revela um equilíbrio satisfatório entre *precisão* (0,76) e *revocação* (0,74), evidenciando boa capacidade de detecção de ameaças, ainda que com certo risco de subdetecção.

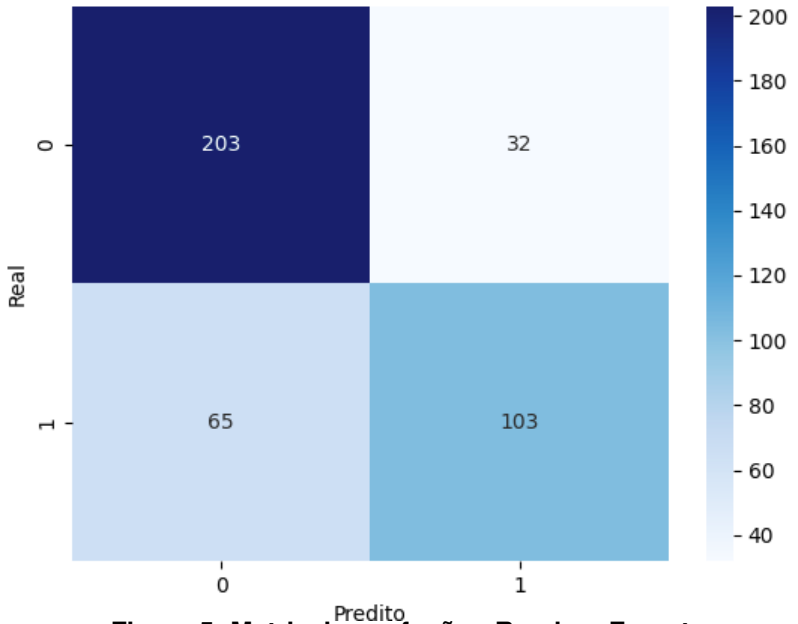

Figura 5. Matriz de confusão - Random Forest.

5.2.2. XGBoost

O algoritmo *XGBoost* obteve uma acurácia de 67% e *F1-score* de 0,68 para a classe positiva. Identificou corretamente 123 casos positivos e 117 negativos, com 71 falsos negativos e 45 falsos positivos. A matriz de confusão na Figura 6 evidencia esses resultados. O modelo destacou-se por sua maior *precisão* (0,73), porém apresentou *recall* inferior (0,63), tornando-se mais conservador e suscetível a falhas na detecção de algumas ameaças reais.

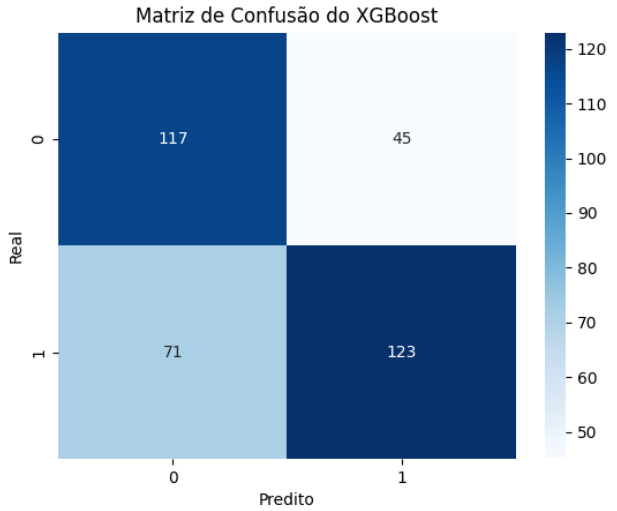

Figura 6. Matriz de confusão - XGBoost.

5.2.3. Support Vector Machine (SVM)

O modelo *SVM* apresentou uma acurácia de (71%) e um *F1-score* (0,76) para a classe "phishing". Com 162 verdadeiros positivos e 91 verdadeiros negativos, apresentou 32 falsos negativos e 71 falsos positivos. A Figura 7 apresenta sua matriz de confusão.

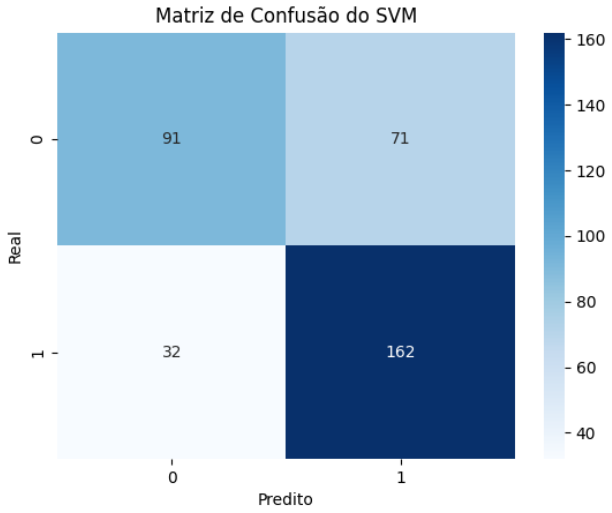

Figura 7. Matriz de confusão - SVM.

5.3. Importância das Features

A análise da importância das variáveis forneceu subsídios valiosos para compreender os critérios utilizados pelos modelos na tomada de decisão. A seguir são destacados os principais atributos segundo cada algoritmo.

5.3.1. Random Forest

No modelo, cinco atributos se destacaram como os mais influentes na classificação de URLs: presença em listas negras (`listed_in_rbl`), origem do IP em país não confiável (`ip_from_untrusted_country`), verificação via *Safe Browsing* (`safe_browsing`), idade do domínio (`has_low_domain_age`) e uso de HTTPS (`uses_https`). Esses fatores refletem aspectos técnicos e contextuais que ajudam a diferenciar URLs legítimas de maliciosas, indicando a importância da reputação externa, origem geográfica e segurança da conexão. A Tabela 5 apresenta as cinco variáveis mais relevantes, com os respectivos pesos, em porcentagem, atribuídos pelo modelo.

Tabela 5. Top 5 atributos mais relevantes segundo o Random Forest.

Atributo	Importância (%)
`listed_in_rbl`	0,12
`ip_from_untrusted_country`	0,11
`has_low_domain_age`	0,10
`safe_browsing`	0,09
`uses_https`	0,08

A Figura 8 apresenta a relevância dos atributos utilizados pelo modelo, permitindo compreender como cada fator contribui para a decisão do modelo, reforçando a eficácia da abordagem na classificação binária de URLs.

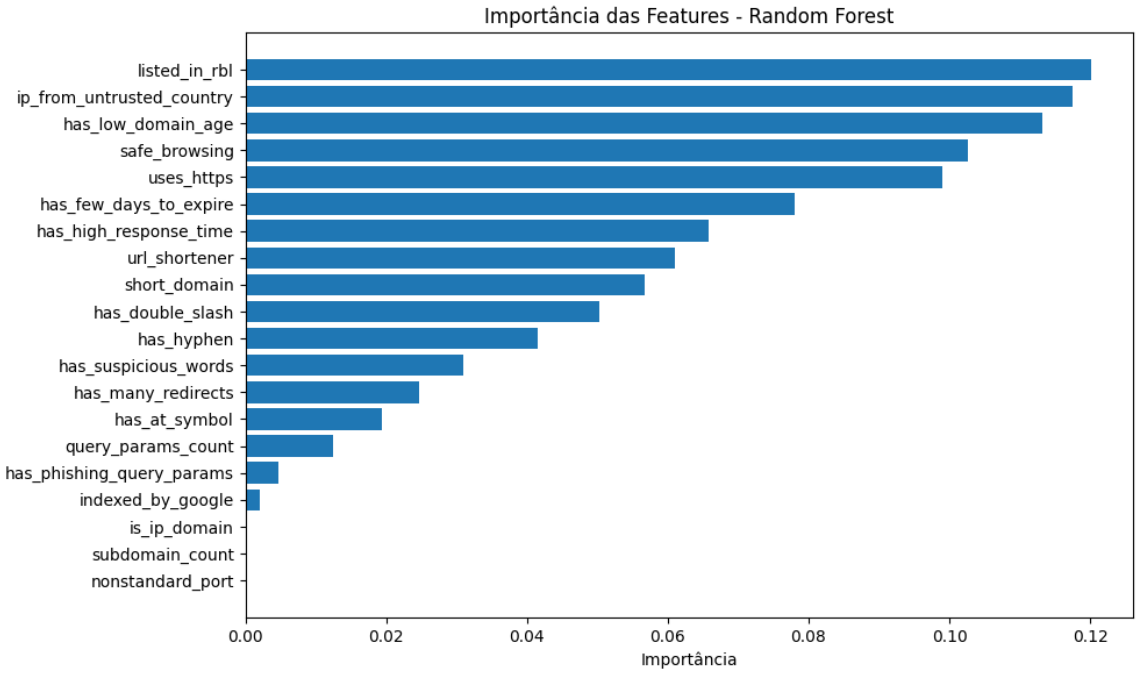

Figura 8. Importância das features - Random Forest.

5.3.2. XGBoost

Os atributos destacados por esse modelo evidenciam aspectos reputacionais, técnicos e geográficos das URLs. O uso do *Safe Browsing* (`safe_browsing`) como o fator mais relevante (12,78%) reforça a importância de fontes externas para validação de segurança. A idade do domínio (`has_low_domain_age`), com 8,5% de importância, está associada a padrões comuns em domínios fraudulentos, frequentemente registrados recentemente para fins maliciosos. O protocolo de conexão (`uses_https`) e o uso de encurtadores de links (`url_shortener`), com pesos de 6,5% e 5,5% respectivamente, refletem características técnicas que podem facilitar o ocultamento de URLs maliciosas. Por fim, o atributo `ip_from_untrusted_country` (4,5%) introduz um aspecto geográfico ao apontar possíveis origens suspeitas de tráfego, o que pode indicar conexões provenientes de locais com baixa reputação digital. A Tabela 6 apresenta essas cinco variáveis com os respectivos pesos, em porcentagem, atribuídos pelo modelo.

Tabela 6. Top 5 atributos mais relevantes segundo o XGBoost.

Atributo	Importância (%)
`safe_browsing`	12,78
`has_low_domain_age`	8,5
`uses_https`	6,5
`url_shortener`	5,5
`ip_from_untrusted_country`	4,5

Tais atributos reforçam a importância de fontes de reputação e da composição técnica da URL para a detecção de ameaças. A visualização completa da importância das *features* é apresentada na Figura 9.

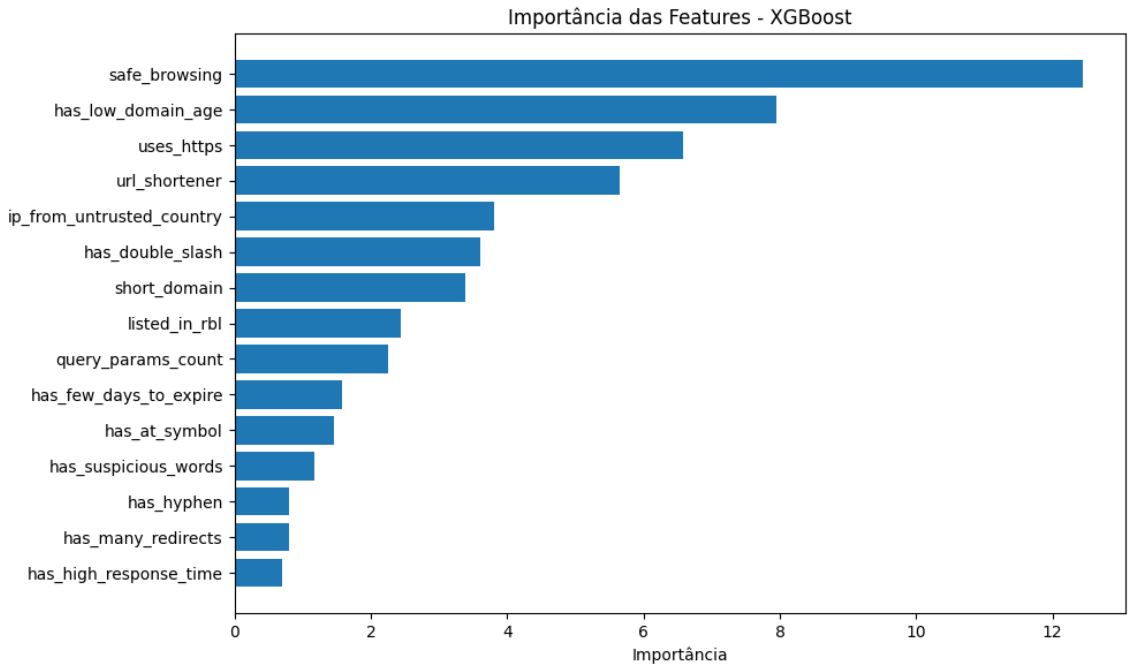

Figura 9. Importância das features - XGBoost.

5.3.3. Support Vector Machine (SVM)

A análise do modelo evidenciou um conjunto de atributos com diferentes níveis de influência na classificação. Variáveis como `query_params_count` e `nonstandard_port` apresentaram alta relevância, indicando a importância da estrutura de parâmetros na URL e do uso de portas não convencionais. Características como uso de HTTPS, domínios representados por IP e presença de palavras suspeitas tiveram contribuição moderada, refletindo aspectos técnicos e semânticos. Já atributos como indexação por mecanismos de busca, presença em listas negras e quantidade de subdomínios apresentaram impacto reduzido. A Tabela 7 sintetiza essa distribuição.

Tabela 7. Distribuição dos principais atributos segundo o modelo SVM.

Atributo	Grau de Influência
`query_params_count`	Alta
`nonstandard_port`	Alta
`uses_https`	Moderada
`is_ip_domain`	Moderada
`has_phishing_query_params`	Moderada
`has_suspicious_words`	Moderada
`safe_browsing`	Moderada
`url_shortener`	Moderada
`indexed_by_google`	Baixa
`listed_in_rbl`	Baixa
`subdomain_count`	Baixa

Essa distribuição indica que o SVM valorizou aspectos estruturais da URL e padrões típicos de *phishing*, como uso de portas não convencionais e parâmetros suspeitos. A Figura 10 apresenta a visualização completa da importância das *features*.

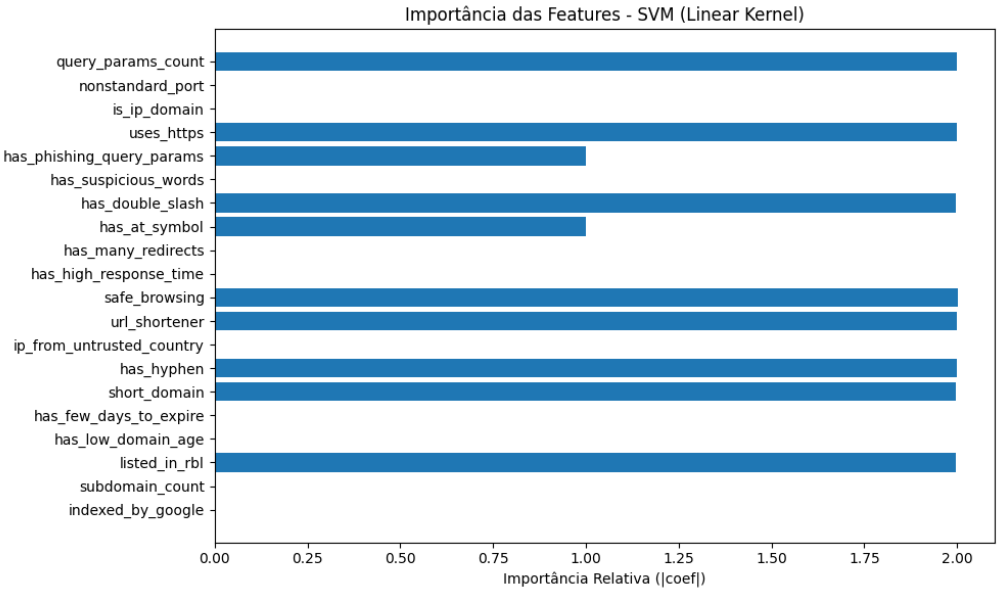

Figura 10. Importância das features - SVM.

5.4. Análise dos Resultados

Os resultados obtidos demonstram que todos os modelos testados apresentaram desempenho satisfatório, com variações relevantes em suas métricas de avaliação. O modelo *Random Forest* apresentou o melhor desempenho global, com acurácia de 76%, *precisão* de 0,76, *recall* de 0,74 e *F1-score* de 0,75. Esses resultados indicam um equilíbrio entre sensibilidade e especificidade, essencial para aplicações voltadas à segurança digital, nas quais é desejável detectar o maior número possível de ameaças mantendo um nível controlado de alarmes falsos. A robustez frente à variabilidade dos dados reforça sua adequação em ambientes reais, especialmente em sistemas com ruído ou atributos redundantes.

O algoritmo *SVM (Support Vector Machine)* obteve o maior *recall* (0,84) e maior *F1-score* (0,76), superando os demais na capacidade de identificar ameaças reais, o que é fundamental em aplicações onde minimizar falsos negativos é prioridade. No entanto, sua *precisão* foi a mais baixa (0,69), sugerindo maior propensão à geração de falsos positivos. Esse comportamento está alinhado à natureza do algoritmo, que tende a construir fronteiras de decisão mais sensíveis em cenários com classes desbalanceadas.

O modelo *XGBoost* obteve o menor desempenho entre os três algoritmos, com acurácia de 67%, *recall* de 0,63 e *F1-score* de 0,68. Apesar de ter alcançado a maior *precisão* (0,73), o baixo *recall* compromete sua capacidade de identificar exemplos maliciosos, o que é um fator limitante em sistemas de defesa contra *phishing*. Sua tendência a favorecer decisões mais conservadoras pode explicar tal comportamento, ainda que se beneficie de técnicas avançadas de regularização.

A análise das *features* mais relevantes revelou que atributos como presença em listas negras, origem geográfica do IP, informações do *Safe Browsing*, idade do domínio e uso de HTTPS são cruciais para a classificação. Esses achados reforçam os padrões descritos na literatura, validam a estratégia de extração de atributos adotada [Ma et al. 2009], e indicam que fatores reputacionais e técnicos têm papel central na identificação de URLs maliciosas.

Dessa forma, conclui-se que o modelo *Random Forest* é o mais indicado para esta aplicação, especialmente pela sua consistência global. Embora o *SVM* tenha se destacado na detecção de ameaças, sua menor precisão pode impactar negativamente em sistemas que demandam maior confiabilidade. O *XGBoost*, com desempenho mais modesto, deve ser utilizado com cautela em contextos que exigem alta sensibilidade. Esse conjunto de resultados foi consolidado em uma aplicação web construída com *Streamlit*, permitindo aos usuários explorar e testar os modelos diretamente pela interface.

5.5. Demonstração da Plataforma

A aplicação desenvolvida possui uma interface interativa construída com o *framework Streamlit*, cujo objetivo é permitir a verificação da legitimidade de URLs com base em uma série de características analisadas previamente pelas etapas de extração e classificação.

Ao acessar a interface, o usuário é apresentado a uma tela inicial com título informativo e campo de entrada para a URL desejada. O sistema aguarda a entrada da URL e, ao receber o dado, inicia o processo de classificação em tempo real. A Figura 11 ilustra a tela de entrada da aplicação.

Figura 11. Tela inicial da aplicação Streamlit

Ao submeter uma URL, o sistema invoca a função `rate_site(url)` responsável por coletar características léxicas, de conectividade e contexto associadas à URL. Esses dados são convertidos em um vetor de entrada compatível com o classificador previamente treinado, que pode ser do tipo *SVM, Random Forest* ou *XGBoost*, dependendo do modelo selecionado.

O modelo retorna uma predição binária indicando se a URL submetida apresenta características associadas a comportamentos maliciosos. Quando isso ocorre, a interface exibe uma mensagem de alerta em vermelho, acompanhada dos motivos que sustentam a decisão.

Essas informações visam fornecer ao usuário subsídios técnicos para compreender o motivo da classificação. A Figura 12 apresenta um exemplo visual da interface ao identificar uma URL maliciosa.

Figura 12. Resultado de verificação de uma URL maliciosa

Por outro lado, quando a URL é considerada segura, a aplicação apresenta uma resposta afirmativa em verde, que refletem a ausência de padrões suspeitos. Os critérios de características adotadas ajudam a sustentar a confiabilidade da classificação sem implicar garantia absoluta.

A interface também oferece acesso às justificativas e detalhes técnicos da verificação, promovendo transparência no processo de decisão. A Figura 13 mostra um exemplo visual da interface para uma URL classificada como legítima.

Figura 13. Resultado de verificação de uma URL legítima

Para fins de avaliação de desempenho, foi incorporado um temporizador à função de classificação. A medição é feita utilizando o módulo *time*, sendo contabilizado o tempo entre o início da análise e o momento em que o resultado é exibido na *interface*.

Na média, o tempo de execução para processar e classificar uma URL é de aproximadamente 2.6 segundos, dependendo da complexidade das consultas envolvidas, como verificação de reputação, consultas WHOIS ou chamadas à API externa do *Google Safe Browsing*. Esse resultado garante uma experiência ágil e condizente com aplicações em tempo quase real.

A interface foi construída para manter a simplicidade e clareza, priorizando o feedback visual imediato. Por se tratar de uma aplicação baseada em *Streamlit*, o layout é responsivo e compatível com diversos navegadores. O sistema evita sobrecarga visual e apresenta as informações de forma escalonada: primeiro a classificação, depois os motivos, e dados complementares como tempo de execução.

6. Conclusões

Este trabalho teve como objetivo principal desenvolver e avaliar modelos de *Machine Learning* capazes de classificar URLs como legítimas ou maliciosas, contribuindo para o fortalecimento de mecanismos de defesa contra ataques de *phishing*. Através da construção de um conjunto robusto de atributos derivados de características técnicas e comporta-

mentais das URLs, foi possível aplicar e comparar o desempenho de três algoritmos amplamente utilizados na literatura: *Random Forest, XGBoost e Support Vector Machine (SVM)*. Como forma de consolidar os resultados e viabilizar a interação com os modelos, foi desenvolvida uma aplicação web utilizando a ferramenta *Streamlit*, permitindo ao usuário analisar URLs diretamente pela interface e visualizar o modelo preditivo em funcionamento.

Os resultados experimentais demonstraram que o modelo baseado em *Random Forest* apresentou o melhor desempenho global, com destaque para o equilíbrio entre *precisão*, *recall* e *F1-score*, evidenciando sua eficácia na tarefa de identificação de ameaças. Esse resultado é especialmente relevante no contexto de segurança da informação, em que é desejável maximizar a detecção de ameaças mantendo níveis aceitáveis de alarmes falsos. Embora o modelo tenha apresentado uma taxa de falsos negativos superior à do SVM, sua performance geral foi mais consistente, tornando-o a alternativa mais robusta entre os três algoritmos avaliados.

O algoritmo *Support Vector Machine* obteve bom desempenho em termos de *recall*, porém gerou mais falsos positivos, o que pode impactar negativamente sistemas que demandam maior confiabilidade. O *XGBoost*, apresentou o menor *recall*, o que limita sua aplicabilidade em ambientes onde a identificação correta de ameaças é crítica.

A análise de importância das *features* revelou que atributos como presença em listas negras, origem geográfica do IP, informações do *safe_browsing*, idade do domínio e uso de HTTPS são cruciais para a classificação, estando alinhados com estudos prévios sobre o comportamento de sites maliciosos. Essa análise reforça a validade das *features* extraídas e abre espaço para futuras otimizações.

Um diferencial importante do projeto foi o desenvolvimento de uma interface gráfica interativa utilizando a biblioteca *Streamlit*, permitindo que o sistema treinado seja acessado por meio de uma aplicação web simples. Essa aplicação possibilita que o usuário final insira qualquer URL e receba, em tempo real, um diagnóstico automatizado indicando se o endereço é legítimo ou não. Tal funcionalidade amplia significativamente a utilidade prática do sistema.

Como limitações do estudo, destaca-se a dependência do conjunto de dados utilizado, bem como a necessidade de atualização constante das fontes de reputação de domínio, uma vez que ataques de *phishing* evoluem com rapidez. Além disso, a ausência de otimização de hiperparâmetros pode ter impactado marginalmente os resultados.

Para trabalhos futuros, uma possível melhoria consiste na implementação de um servidor DNS próprio, integrado ao sistema de detecção desenvolvido. Essa abordagem permitiria interceptar e validar requisições DNS antes da resolução de nomes de domínio, utilizando o modelo treinado para classificar URLs em tempo real. Ao identificar uma URL maliciosa, o servidor poderia bloquear automaticamente a requisição, impedindo que o usuário acesse o site suspeito. Esse mecanismo de filtragem na camada de rede atuaria como uma linha adicional de defesa proativa, ampliando a eficácia do sistema ao prevenir o acesso a páginas fraudulentas mesmo antes da renderização no navegador.

Referências

Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 3 edition.

Ayres, C. et al. (2019a). Utilizando Aprendizado de Máquina para Detecção Automática de URLs Maliciosas Brasileiras. In *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Ayres, F. et al. (2019b). Malicious URL Detection Using Feature Extraction and Machine Learning. *Anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Buczak, A. L. and Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 18(2).

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Costa, L. et al. (2021). Classificação Automática de URLs Maliciosas a Partir da Origem do IP e Dados WHOIS. *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Coutinho, L. G. (2022). Detecção de Páginas de Phishing Utilizando Aprendizado de Máquina. Trabalho de Conclusão de Curso, Universidade Estadual Paulista (UNESP).

Domingos, A. and Dalbert, C. (2023). Identificação de Ataques de Phishing através de Machine Learning. In *Anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSEG)*.

Freitas, R. V. and Loiola Junior, E. d. (2023). Crimes cibernéticos durante a pandemia de covid-19. *Revista Acadêmica de Iniciação Científica*, 1(1):58–69. Acesso em: 13 ago. 2025.

Jakobsson, M. and Myers, S. (2006). *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience.

Kaspersky (2023a). Malicious URL Redirect Methods. Disponível em: `https://www.kaspersky.com/blog/malicious-redirect-methods/50045`. Acesso em: 27 jun. 2025.

Kaspersky (2023b). What is Typosquatting? Disponível em: `https://www.kaspersky.com.br/resource-center/definitions/what-is-typosquatting`. Acesso em: 27 jun. 2025.

Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L. F., and Hong, J. (2010). Teaching Johnny Not to Fall for Phish. *ACM Transactions on Internet Technology (TOIT)*, 10(2):1–31.

Le, H., Pham, Q., Sahoo, D., and Hoi, S. C. H. (2018). URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection. *arXiv preprint arXiv:1802.03162*.

Liaw, A. and Wiener, M. (2002). Classification and Regression by RandomForest. *R News*, 2(3):18–22.

Ma, J., Saul, L., Savage, S., and Voelker, G. (2009). Beyond blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 1245–1254.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Natekin, A. and Knoll, A. (2013). Gradient Boosting Machines, A Tutorial. *Frontiers in Neurorobotics*, 7:21.

NordVPN (2023). Amazon Phishing Email: How to Spot and Report it. Disponível em: `https://nordvpn.com/pt-br/blog/amazon-phishing-email/`. Acesso em: 27 jun. 2025.

Odown (2023). API Response Time Standards. Disponível em: `https://odown.com/blog/api-response-time-standards`. Acesso em: 27 jun. 2025.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Édouard Duchesnay (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sahoo, D., Liu, C., and Hoi, S. (2017). Detecting Phishing Websites Using Machine Learning Technique. *PLOS ONE*, 12(10).

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3).

SerpNames (2023). Domain Expiry: Why it Matters and how to Save Expired Domains. Disponível em: `https://serpnames.com/domain-expiry-why-it-matters-and-how-to-save-expired-domains/`. Acesso em: 27 jun. 2025.

Shaikh, S. et al. (2023). AntiPhishStack: LSTM-based Stacked Generalization Model for Optimized Phishing URL Detection. *arXiv preprint arXiv:2401.08947*.

Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press.

Sitebulb (2023). Query String Contains More than Three Parameters. Disponível em: `https://sitebulb.com/hints/internal/query-string-contains-more-than-three-parameters`. Acesso em: 27 jun. 2025.

Trustwave (2022). Deceptive URL Redirections in Phishing Attacks. Disponível em: `https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/trusted-domain-hidden-danger-deceptive-url-redirections-in-email-phishing-attacks`. Acesso em: 27 jun. 2025.

van Rossum, G. and Drake, F. L. (2009). *The Python Language Reference Manual*. Network Theory Ltd.

Zhao, L., Yang, J., and Zhang, Y. (2021a). An Empirical Study of XGBoost for Malware Detection. *Journal of Information Security and Applications*, 59:102850.

Zhao, Z., Xu, W., Wang, S., Xu, H., and Liu, B. (2021b). A Comparison of Machine Learning Algorithms for Phishing Website Detection. *Security and Communication Networks*, 2021.