WIDaT 2025
VIII WORKSHOP DE INFORMAÇÃO, DADOS E TECNOLOGIA
MARÍLIA • SP

Aprendizado de Máquina para detecção de sites de *Phishing*

Machine Learning for Phishing website detection

Pedro Henrique Del Bianco Hokama[1]
http://lattes.cnpq.br/9395192779202995
https://orcid.org/0000-0002-3734-7679

Jader Duque Figueredo[2]
http://lattes.cnpq.br/5323391336547899
https://orcid.org/0009-0005-6281-3386

Karen Cristina Soares Cavalheiro[3]
http://lattes.cnpq.br/3606856318278865
https://orcid.org/0000-0003-2513-6927

Resumo

No meio digital, o termo Phishing se refere a materiais que simulam algum serviço com o objetivo de enganar os usuários para que enviem informações pessoais. Este artigo apresenta a utilização da técnica de árvores interpretáveis para a detecção de sites de Phishing. Essas árvores têm a característica de serem compreensíveis e possivelmente utilizadas por usuários comuns, mesmo sem o auxílio de um software especializado. Foi utilizada uma formulação em Programação Linear Inteira para encontrar árvores que classificam um conjunto de treinamento da melhor forma possível. Resultados parciais apontam que observando poucos elementos sobre um site foi possível detectar um Phishing com 87% de precisão no conjunto de testes.

Palavras-chave: golpes digitais; documentos falsos; árvores de classificação.

Abstract

In the digital realm, the term Phishing refers to materials that simulate a legitimate service with the aim of deceiving users into submitting personal information. We present the approach of interpretable tree techniques for phishing website detection. These trees possess the characteristic of being potentially usable by everyday users, even without the aid of specialized software. We employ an Integer Linear Programming formulation to find trees that classify a training set in the best possible way. Partial results indicate that by observing a few elements about a website, it was possible to detect phishing with 87% accuracy in the test set.

Keywords: digital scams; fraudulent documents; classification trees.

[1] Doutor em Ciência da Computação, Universidade Estadual de Campinas (UNICAMP), Campinas, SP, Brasil. Professor do Magistério Superior, Universidade Federal de Itajubá (UNIFEI), Itajubá, MG, Brasil. Hokama@unifei.edu.br.
[2] Mestre em Ciência e Tecnologia da Computação, Universidade Federal de Itajubá (UNIFEI), Itajubá, MG, Brasil. Analista de Tecnologia da Informação, Universidade Federal de Itajubá (UNIFEI), Itajubá, MG, Brasil. jader.figueredo@unifei.edu.br.
[3] Mestre em Ciência da Informação, Universidade Federal de São Carlos (UFSCar), São Carlos, SP, Brasil. Doutoranda, Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), Marília, SP, Brasil. karen.cavalheiro@unesp.br.

1 INTRODUÇÃO

A crescente dependência da sociedade moderna em tecnologias digitais tem sido acompanhada por um aumento significativo nas ameaças cibernéticas, com destaque para a prática danosa de *phishing* (Khonji; Iraqi; Jones, 2013). O *phishing* é uma forma de ataque em que indivíduos mal-intencionados tentam enganar usuários, induzindo-os a revelar informações confidenciais, como senhas e dados pessoais, por meio de mensagens falsas e sites fraudulentos. Para combater essa ameaça, é importante utilizar técnicas eficazes de detecção de *phishing*, mas também conscientizar os usuários da internet de como evitar esse tipo de golpe.

Nesse contexto, as árvores de classificação interpretáveis emergem como uma abordagem promissora para identificar sites suspeitos associados a atividades de *phishing*. Técnicas mais comuns de aprendizado de máquinas, como as redes neurais e redes neurais profundas, têm sido largamente utilizadas com sucesso em diversas aplicações (Sharifani; Amini, 2023). Entretanto, essas técnicas sofrem da falta de interpretabilidade dos resultados, ou seja, um usuário comum ao olhar uma rede neural não consegue, por exemplo, entender o que faz um dado ser classificado de uma forma ou de outra, tão pouco conseguiria classificar uma entrada sem um programa de computador.

Em contrapartida as árvores de classificação (Bertsimas; Orfanoudaki; Wiberg, 2021) oferecem a vantagem de serem facilmente interpretáveis, permitindo que, tanto usuários comuns quanto especialistas em segurança, compreendam, validem e apliquem as decisões tomadas pelo modelo. Essa interpretabilidade é fundamental para ganhar a confiança dos usuários e criar um sistema de detecção simples. Estudos recentes destacaram a efetividade do modelo exato aplicado a decisão (Figueredo, 2024).

A técnica de Árvores de Classificação Ótimas, tem ganhado novos estudos como as Árvores de Classificação Ótimas Fortes (*Strong Optimal Classification Trees*) (Aghaei, Gómez e Vayanos, 2024) que gera árvores mais justas através da relaxação linear mais forte e uso de método de decomposição de Benders, obtendo aplicações mais práticas e poderosas de árvores de classificação.

Já a Árvore de Decisão Ótima usando Programação Dinâmica de Van Der Linden, De Weerdt e Demirović (2023) não só é capaz de lidar com grande quantidade de restrições e objetivos como ainda superar as linhas de base de eficiência e escalabilidade de abordagens baseadas em Otimização Inteira Mista.

Neste trabalho é explorado o uso de árvores de classificação como uma ferramenta para detectar *sites* de *phishing*. Serão resumidos os fundamentos teóricos por trás das árvores de classificação, explicando como elas funcionam e como podem ser aplicadas à detecção de *phishing*. Também será apresentado um modelo matemático em Programação Linear Inteira (PLI) capaz de encontrar as melhores árvores possíveis para um dado conjunto de treinamento.

Safi e Singh (2023) realizaram uma revisão sistemática da literatura sobre as diferentes abordagens de detecção de *phishing*, para isso foram analisados 80 trabalhos, dentre eles artigos científicos e trabalhos apresentados em eventos ou *sites* renomados dos últimos cinco anos; os autores apontam que as técnicas de aprendizado de máquina foram as mais aplicadas, sendo utilizadas em 57 estudos e o algoritmo que atingiu maior precisão na detecção de *sites* de *phishing* foi a Rede Neural Convolucional, com 99,98% de acerto.

O objetivo geral desta pesquisa é auxiliar usuários a detectarem se o *site* em que estão navegando possui *phishing*. O objetivo específico é: elencar orientações simples e capazes de ser identificadas com boa precisão que possam classificar um site como *phishing* ou não.

A escolha do modelo em Programação Linear Inteira se deve ao fato de que, para realizar a classificação ele tem mostrado bons resultados em trabalhos recentes (Bertsimas; Orfanoudaki; Wiberg, 2021; Figueredo, 2024). Destaca-se ainda que esse é um trabalho parcial, em trabalhos futuros os modelos podem ser expandidos e testados em diferentes aplicações.

Este trabalho está dividido em seções, são elas: a seção 2, que é apresentada a metodologia empregada; a seção 3, que contempla o desenvolvimento e a fundamentação teórica da árvore de classificação e da PLI; A seção 4, que apresenta os

resultados parciais obtidos até o momento e por fim, a seção 5, na qual constam as conclusões dos autores e trabalhos futuros.

2 PROCEDIMENTOS METODOLÓGICOS

Para o desenvolvimento da fundamentação teórica da pesquisa foi realizado um levantamento documental sobre os temas propostos: *Phishing* golpes digitais; documentos falsos; árvores de classificação.

A pesquisa utiliza o método Árvore de Classificação Ótima e se caracteriza como quali-quantitativa, de caráter descritivo e exploratório, sobre os temas de: Aprendizado de máquina, detecção de informação falsa e golpes e modelagem matemática.

Para realização da pesquisa os modelos foram implementados utilizando a linguagem de programação C++, compilado em g++ 11.3.0 e executado em ambiente Linux. O resolvedor de Programação Linear Inteira utilizado foi o CPLEX 20.1.

Os experimentos foram realizados com o *dataset* "***Phishing website Detector***" de Eswar Chand (2020) disponível na plataforma *Kaggle*. O *dataset* é formado por 11054 observações com 30 atributos (variáveis dependentes) e 1 classe (variável dependente), a qual pretende-se predizer, organizados em tabela, onde as observações são as linhas e os atributos são as colunas. Os atributos são todos numéricos, onde o valor "1" significa que o atributo acorre, "0" significa neutro ou incerto e "-1" significa que o atributo não acontece. Todos as observações são classificadas como "1" seguro, ou "-1" malicioso. O conjunto foi dividido em 2, onde a primeira metade foi usada para treinamento e a outra metade para teste.

3 DESENVOLVIMENTO

Árvores de classificação interpretáveis são modelos de aprendizado de máquina que representam um conjunto de regras de decisão hierárquicas em uma estrutura de árvore. Cada nó interno da árvore representa uma condição sobre um atributo, enquanto os nós folha representam as classes ou rótulos de saída. Essas árvores são ditas interpretáveis, pois permitem uma compreensão intuitiva das regras de classificação

utilizadas pelo modelo. Os caminhos na árvore representam as sequências de decisões tomadas para chegar a uma classificação específica, facilitando a interpretação e a explicação do processo de tomada de decisão.

Os usuários podem utilizar as árvores da seguinte forma: no topo da árvore existe uma pergunta, caso a pergunta seja respondida com “não”, o usuário segue a linha/galho da direita, caso a resposta seja “sim”, segue para a esquerda, e continua para as seguintes perguntas até que que no final seja respondido se o Localizador Uniforme de Recursos (URL) é legítimo ou não, e o usuário saberá se está sujeito à *phishing*; se a resposta final respondida for “legítimo”, significa que é seguro de ser acessado.

A interpretabilidade das árvores de classificação é um atributo valioso em diversas aplicações, especialmente em problemas nos quais é desejável que a compreensão do modelo possa ser utilizada como instruções simples, que um humano possa seguir sem o auxílio de um *software*. A capacidade de examinar as regras de decisão em cada nó permite que especialistas em segurança compreendam quais características ou indicadores são relevantes para identificar um *site* malicioso. Além disso, as árvores de classificação interpretáveis permitem que os usuários finais entendam por que um *site* foi classificado como suspeito ou seguro, aumentando a confiança e a transparência do sistema de detecção.

3.1 Treinamento e Teste

O conjunto de treinamento é uma parte dos dados disponíveis usados para ajustar um modelo de aprendizado de máquina. Ele consiste em exemplos de entradas (por exemplo: *sites*) associados às respectivas saídas desejadas (por exemplo: se é um *site* de *phishing* ou não). O objetivo do treinamento é permitir que o modelo aprenda padrões e relações entre os dados de entrada e saída, ajustando seus parâmetros e melhorando seu desempenho ao longo do tempo. O conjunto de treinamento é fundamental para criar um modelo preciso e geral, capaz de fazer previsões precisas em novos dados.

Por outro lado, o conjunto de teste é uma porção separada dos dados que não é usada durante o treinamento, mas sim para avaliar o desempenho do modelo após o treinamento. Esse conjunto de teste é composto por exemplos que não foram

apresentados ao modelo durante o treinamento, permitindo verificar a capacidade de generalização do modelo para dados não vistos anteriormente. Ao avaliar o modelo com o conjunto de teste, pode-se obter métricas de desempenho, como acurácia, que indicam quão bem o modelo está se saindo na detecção de *phishing* não observadas anteriormente. Isso ajuda a estimar o desempenho do modelo em cenários reais e a verificar se ele é capaz de generalizar efetivamente para novos dados.

3.2 Fundamentos da Programação Linear Inteira

A Programação Linear Inteira (Wolsey, 2020) é uma técnica poderosa utilizada para otimizar problemas em que as variáveis de decisão devem assumir valores inteiros. Ela combina conceitos da Programação Linear, que lida com variáveis contínuas, com a restrição adicional de que as variáveis devem ser inteiras. A PLI tem sido, por décadas (Wolsey; Nemhauser, 1999), amplamente aplicada em diferentes áreas, como logística, produção e pesquisa operacional.

Ao utilizar PLI para encontrar árvores de classificação interpretáveis, o problema de construção da árvore é formulado como um problema de otimização, em que o objetivo é encontrar a melhor árvore de classificação com base em critérios específicos, como a acurácia de classificação e a interpretabilidade das regras geradas (complexidade da árvore).

Informalmente, um “Programa Linear Inteiro” é um modelo matemático que descreve como uma solução para o problema deve ser, ou seja, quais restrições ela deve respeitar relacionadas a uma função objetivo que descreve o que deve ser otimizado. Esse modelo então é resolvido por um algoritmo e obtém-se a melhor solução possível.

3.3 Formulação do Problema de Árvores de Classificação com PLI

Para utilizar PLI na construção de árvores de classificação interpretáveis para detecção de *phishing* em URLs, o problema é formulado como um modelo de otimização. O modelo é composto por variáveis de decisão que representam as regras de divisão em

cada nó da árvore e as relações entre os nós. O objetivo é maximizar a acurácia de classificação da árvore, enquanto se impõem restrições relacionadas à interpretabilidade.

As restrições de interpretabilidade podem ser definidas de várias formas, dependendo dos critérios desejados. Por exemplo, pode-se limitar o número máximo de níveis na árvore, ou ainda punindo a quantidade de divisões, fazendo com que a solução apenas as crie quando valerem a pena. Essas restrições são essenciais para garantir que a árvore resultante seja compreensível e interpretável.

O modelo apresentado a seguir foi adaptado de Bertsimas, Orfanoudaki e Wiberg (2021), primeiramente serão definidos os dados do problema, depois as variáveis, e por fim o modelo. São dados do problema:

a) I é o conjunto dos dados de treinamento. Cada elemento i de I é um *site*;
b) P é o conjunto de atributos, e.g., (Usa Subdomínio, usa encurtador de URL e etc);
c) K é o conjunto de classes {*Phishing*, Legítimo};
d) x_{ip} é o valor do parâmetro p para o item $i \in I$;
e) t é cada nó dentre um conjunto T de nós da árvore;
f) T_B é o conjunto de nós internos da árvore;
g) T_L é o conjunto de nós folhas da árvore;
h) $A_R(t)$ são os ancestrais do nó t onde foi necessário descer à esquerda para chegar em t;
i) $A_L(t)$ são os ancestrais do nó t onde foi necessário descer à direita para chegar em t;

As variáveis do modelo são:

a) a_{tp} indica se o parâmetro p é usado para a divisão no nó t;
b) b_t indica qual o valor que faz um elemento ir para direita ou para esquerda;
c) z_{it} indica se o item i está na folha t;
d) c_{kt} indica se a folha t está classificada como classe k;
e) N_{kt} indica a quantidade de elementos da classe k que estão na folha t;
f) N_t indica a quantidade total de elementos na folha t;
g) L_t indica a quantidade de erros na folha t.

O modelo então é apresentado a seguir e posteriormente uma intuição do que significa cada uma das restrições.

$$\text{Min.} \sum_{t \in \mathcal{T}_L} L_t, \tag{1}$$

$$\text{s. a } L_t \geq N_t - N_{kt} - |I|(1 - c_{kt}), \qquad k \in K, \forall t \in \mathcal{T}_L, \tag{2}$$

$$N_{kt} = \sum_{i \in I : y_i = k} z_{it}, \qquad k \in K, \forall t \in \mathcal{T}_L, \tag{3}$$

$$N_t = \sum_{i \in I} z_{it}, \qquad \forall t \in \mathcal{T}_L, \tag{4}$$

$$\sum_{k \in K} c_{kt} = 1, \qquad \forall t \in \mathcal{T}_L, \tag{5}$$

$$\sum_{p \in P} a_{mp} x_{ip} \geq b_m - M(1 - z_{it}), \qquad i \in I, \forall t \in \mathcal{T}_L, \forall m \in A_R(t), \tag{6}$$

$$\sum_{p \in P} a_{mp} x_{ip} + \epsilon \leq b_m + M(1 - z_{it}), \quad i \in I, \forall t \in \mathcal{T}_L, \forall m \in A_L(t), \tag{7}$$

$$\sum_{p \in P} a_{tp} = 1, \qquad \forall t \in \mathcal{T}_B, \tag{8}$$

$$0 \leq b_t \leq 1, \qquad \forall t \in \mathcal{T}_B, \tag{9}$$

$$z_{it} \in \{0, 1\}, \qquad i \in I, \forall t \in \mathcal{T}_L, \tag{10}$$

$$c_{kt} \in \{0, 1\}, \qquad k \in K, \forall t \in \mathcal{T}_L, \tag{11}$$

$$a_{tp} \in \{0, 1\}, \qquad p \in P, \forall t \in \mathcal{T}_B, \tag{12}$$

A função objetivo (1) busca minimizar o número total de erros. A restrição (2) faz com que o número de erros em cada folha seja corretamente computado. A restrição (3) contabiliza o número de elementos de cada classe em cada folha. A restrição (4) computa a quantidade total de elementos por folha. A restrição (5) obriga toda folha a ser classificada como uma classe. As restrições (6) e (7) juntas fazem com que o elemento, para chegar em uma certa folha, tenha percorrido adequadamente as divisões dos ancestrais. A restrição (8) obriga todo galho a escolher um parâmetro para a divisão. Por fim as restrições (9), (10), (11) e (12) indicam o domínio de cada variável.

4 RESULTADOS PRELIMINARES

Como o número de variáveis do modelo depende do número de exemplares no conjunto de treinamento, e esse número impacta diretamente no tempo de resolução do modelo, optou-se por limitar o número de exemplares do conjunto de treinamento aos 150 primeiros, e o conjunto de testes são os 150 seguintes.

O modelo apresentado na seção 3 foi implementado em linguagem C++, compilado com g++ 11.3.0 e executado em ambiente Linux. O resolvedor de Programação Linear Inteira utilizado foi o CPLEX 20.1.

Para reduzir o número de parâmetros e preservar a interpretabilidade das árvores geradas, eliminou-se do *dataset* todos os parâmetros que um usuário leigo pudesse ter dificuldade de verificar. Dessa forma os resultados obtidos podem ser utilizados como instrução para a detecção de *phishing* mesmo sem o uso de softwares adicionais. Vale notar que se não houvesse o objetivo de tornar a árvore aplicável por um usuário, poderíamos usar um método como o *Principal Component Analysis* - PCA (Greenacre *et al.*, 2022) para reduzir o número de parâmetros. Ao final sobraram 11 parâmetros:

a) 1. A URL do *site* usa um endereço de protocolo de internet (IP)?
b) 2. A URL é demasiadamente longa?
c) 3. O *site* usa encurtador de URL?
d) 4. A URL usa o símbolo de @?
e) 5. O domínio tem um traço/hífen “-”?
f) 6. A URL tem subdomínios?
g) 7. O *site* usa Protocolo de Transferência de Hipertexto Seguro (HTTPS)?
h) 8. O *site* usa uma porta não padrão?
i) 9. O *site* exibe *pop-ups*?
j) 10. O *site* tem um *pagerank* menor que 0.2?
k) 11. O *site* está indexado pelo *Google*?

Além disso, é preciso que a altura da árvore seja pequena, por isso limitou-se a altura da árvore a 2 e 3. A Tabela 1 apresenta os resultados do treinamento e do desempenho da árvore obtida no conjunto de testes. Pode-se notar que o tempo para

treinamento é muito maior quando a altura é 3, mas apesar disso a solução obtida acerta apenas 0,7% mais classificações no conjunto de treinamento.

Tabela 1 – Treinamento e desempenho da árvore

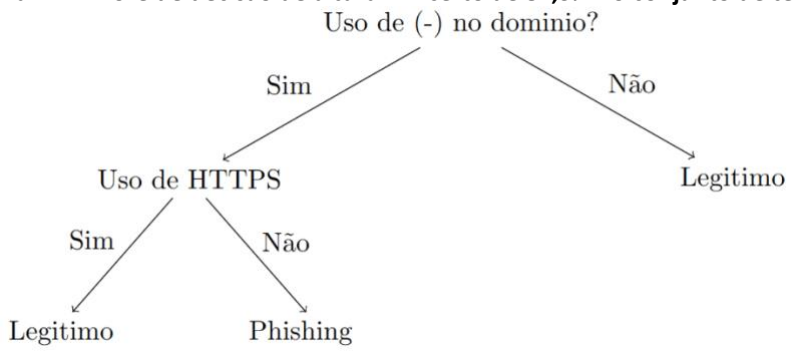
Título	Tempo de treinamento	% acertos treino	% acertos teste
Altura 2	2 segundos	88,0%	87,3%
Altura 3	1507 segundos	88,7%	84,0%

Fonte: Elaboração própria.

Pode-se notar ainda que, no conjunto de teste, a árvore mais simples com altura dois teve um desempenho maior. Conjectura-se que quando a árvore é mais complexa aumenta a possibilidade de acontecer o que é conhecido como *overfitting* (Ying, 2019), quando um sistema se adapta demasiadamente ao conjunto de treinamento, e acaba tendo dificuldade de ser generalizado para outras entradas.

A Figura 1 mostra a árvore que foi obtida quando a altura é limitada a dois, pode-se observar que apenas 2 perguntas simples são capazes de identificar se o *site* é de *Phishing* com uma precisão razoável de 87,3%.

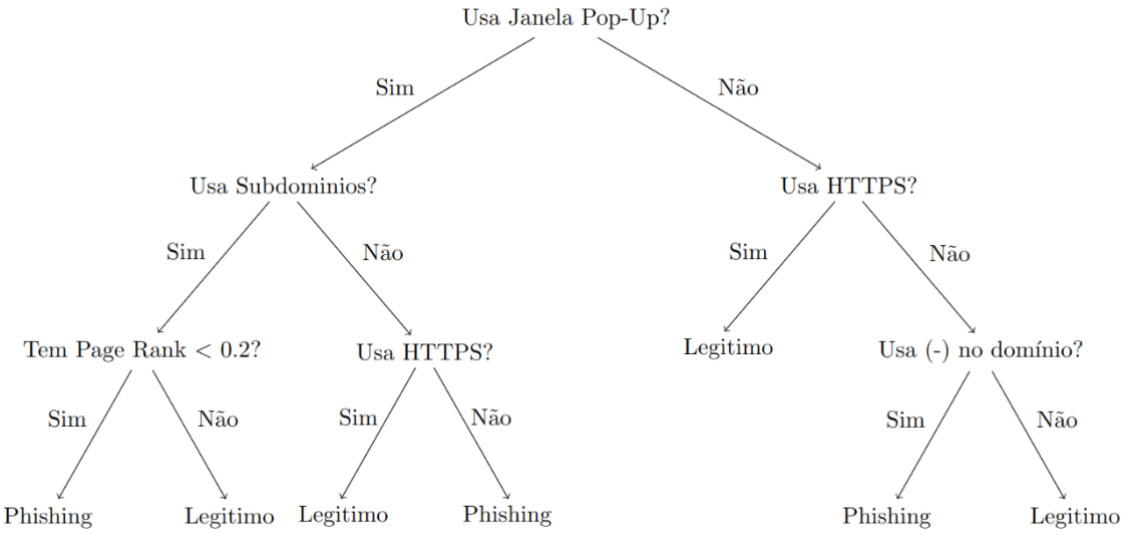
Figura 1 – Árvore de decisão de altura 2. Acerto de 87,3% no conjunto de teste



Fonte: Elaboração própria.

A Figura 2 mostra a árvore que foi obtida quando a altura é limitada a três, pode-se considerar que a complexidade da árvore é bem maior. Ao longo de toda a árvore ocorreram 6 perguntas, sendo duas repetidas. Além disso, a pergunta sobre o PageRank exige uma consulta a alguma ferramenta externa.

Figura 2 – Árvore de decisão de altura 3. Acerto de 84% no conjunto de teste



Fonte: Elaboração própria.

Dessa forma observa-se que nesse caso particular o limite aumentado da altura não trouxe benefícios que o justificassem.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentada uma abordagem para a detecção de *sites* de *Phishing* utilizando árvores de classificação ótimas. A escolha desse método foi gerar uma instrução de detecção que não dependa da execução de um *software*, ou seja, um usuário pode ser instruído para verificar poucas observações em um *site* e detectar com boa precisão se um *site* se trata de um golpe.

O objetivo geral da pesquisa foi atendido na apresentação da Tabela 1, na qual foi encontrada uma árvore de classificação que atendia os requisitos de ser simples e interpretável por usuários. Já o objetivo específico, foi atendido na Figura 1, na qual elencou-se as orientações mais precisas para a identificação de *sites* de *phishing*.

Uma das principais contribuições deste trabalho reside na demonstração da eficácia de árvores de classificação interpretáveis, construídas através de Programação Linear Inteira (PLI), para a detecção de *sites* de *Phishing*. Foram obtidas árvores de altura 2 e 3, e notavelmente, a árvore com apenas altura 2 alcançou uma precisão de 87,3% no conjunto de testes. Este resultado sublinha a capacidade de um modelo simples, baseado

em apenas duas perguntas sobre características facilmente observáveis de um *site*, de discriminar entre *sites* legítimos e fraudulentos com uma acurácia considerável. Essa simplicidade e interpretabilidade são cruciais, pois permitem que usuários comuns possam ser instruídos a verificar esses poucos elementos em um *site* para se protegerem contra golpes de *phishing*, sem a necessidade de *softwares* especializados.

A pesquisa também observou que uma árvore mais complexa (altura 3) não apresentou uma melhoria significativa na precisão no conjunto de testes, e até mesmo demonstrou sinais de *overfitting*, reforçando o valor de modelos mais simples e interpretáveis para este problema. Em suma, a identificação de uma árvore de decisão rasa e altamente precisa, utilizando um conjunto limitado de atributos facilmente verificáveis, representa um avanço significativo na criação de métodos de detecção de *phishing* acessíveis e compreensíveis para o público em geral

Como trabalhos futuros pretende-se ampliar os testes computacionais para uma melhor validação do método, por exemplo, testar outras métricas como precisão, sensibilidade e pontuação F1, para verificar a robustez do método. Também pode-se utilizar metodologia semelhante para outros tipos de detecção, como notícias e documentos falsos. Além disso, o método também pode ser utilizado para a classificação de outras naturezas, como público-alvo, classificação indicativa (faixa etária), classificação de gênero, entre outras. Pretende-se ainda publicar a implementação dos códigos para a reprodutibilidade da pesquisa.

REFERÊNCIAS

AGHAEI, Sina; GÓMEZ, Andrés; VAYANOS, Phebe. Strong optimal classification trees. **Operations research**, [*S. l.*], p. 1-19, 2024. DOI: https://doi.org/10.1287/opre.2021.0034. Disponível em: https://pubsonline.informs.org/doi/10.1287/opre.2021.0034. Acesso em: 15 maio 2025.

BERTSIMAS, Dimitris; ORFANOUDAKI, Agni; WIBERG, Holly. Interpretable clustering: an optimization approach. **Machine Learning**, [*S. l.*], v. 110, p. 89-138, 2021. Disponível em: https://link.springer.com/article/10.1007/s10994-020-05896-2. Acesso em: 10 mar. 2025.

CHAND, Eswar. **Phishing website detector**. 2020. Disponível em: https://www.kaggle.com/datasets/eswarchandt/phishing-website-detector. Acesso em: 10 mar. 2025.

FIGUEREDO, Jader Duque. **Modelos de aprendizado de máquina para árvore de decisão interpretável**: otimização vs heurística. 2024. Dissertação (Mestrado em Ciência e Tecnologia da Computação) – Universidade Federal de Itajubá, Instituto de Engenharia de Sistemas e Tecnologia da Informação, Itajubá, 2024. Disponível em: https://repositorio.unifei.edu.br/jspui/handle/123456789/4191. Acesso em: 10 mar. 2025.

GREENACRE, Michael *et al*. Principal component analysis. **Nature reviews methods primers**, [*S. l.*], v. 2, n. 100, 2022. Disponível em: https://www.nature.com/articles/s43586-022-00184-w. Acesso em: 10 mar. 2025.

KHONJI, Mahmoud; IRAQI, Youssef; JONES, Andrew. Phishing detection: a literature survey. **IEEE communications surveys and tutorials**, [*S. l.*], v. 15, n. 4, p. 2091-2121, 2013. Disponível em: http://romisatriawahono.net/lecture/rm/survey/network%20security/Khonji%20-%20Phishing%20Detection%20-%202013.pdf. Acesso em: 10 mar. 2025.

SAFI, Asadullah; SINGH, Satwinder. A systematic literature review on phishing website detection techniques. **Journal of King Saud University-Computer and Information Sciences**, Saudi Arabia, v. 35, n. 2, p. 590-611, 2023. Disponível em: https://www.sciencedirect.com/science/article/pii/S1319157823000034. Acesso em: 15 maio 2025.

SHARIFANI, Koosha; AMINI, Mahyar. Machine Learning and Deep Learning: a review of methods and applications. **World information technology and engineering journal**, [*S. l.*], v. 10, n. 7, p. 3897-3904, 2023. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4458723. Acesso em: 10 mar. 2025.

VAN DER LINDEN, Jacobus; DE WEERDT, Mathijs; DEMIROVIĆ, Emir. Necessary and sufficient conditions for optimal decision trees using dynamic programming. **Advances in neural information processing systems**, [*S. l.*], v. 36, p. 9173-9212, 2023. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2023/file/1d5fce9627e15c84db572a66e029b1fc-Paper-Conference.pdf. Acesso em: 15 maio 2025.

WOLSEY, Laurence Alexander; NEMHAUSER, George Lann. **Integer and combinatorial optimization**. [Hoboken]: Wiley, 1999.

WOLSEY, Laurence Alexander. **Integer programming**. Hoboken: Wiley, 2020.

YING, Xue. An overview of overfitting and its solutions. **Journal of Physics: Conf. Series**, [*S. l.*], v. 1168, n. 2, e022022, 2019. Disponível em:

https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf. Acesso em: 10 mar. 2025.

Como citar este trabalho:
HOKAMA, Pedro Henrique Del Bianco; FIQUEREDO, Jader Duque; CAVALHEIRO, Karen Cristina Soares. Aprendizado de Máquina para detecção de sites de Phishing. *In*: WORKSHOP DE INFORMAÇÃO DADOS E TECNOLOGIA, 8., 2025, Marília, SP. **Anais [...]**. Marília, SP: Universidade de Marília, 2025. DOI: https://doi.org/10.22477/viii.widat.271.