



DISSERTAÇÃO DE MESTRADO PROFISSIONAL

**HOOF-STACK-PUD: ARCABOUÇO HETEROGÊNEO
DE META-APRENDIZADO EMPILHADO
COM PREDIÇÕES FORA DA DOBRA
PARA DETECÇÃO DE URLs DE PHISHING**

Hernany Silveira Rocha

Programa de Pós-Graduação Profissional em Engenharia Elétrica

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

**HOOF-STACK-PUD: ARCABOUÇO HETEROGÊNEO
DE META-APRENDIZADO EMPILHADO
COM PREDIÇÕES FORA DA DOBRA
PARA DETECÇÃO DE URLs DE PHISHING**

Hernany Silveira Rocha

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

Banca Examinadora

Prof. João Paulo da Costa Javidi, Ph.D, FT/UnB
Orientador

Prof. Laerte Peotta de Mello, Ph.D, FT/UnB
Examinador interno

Kevin Herman Muraro Gularte , Ph.D, HSHL
Examinador externo

FICHA CATALOGRÁFICA

S. ROCHA, HERNANY

HOOF-STACK-PUD: ARCABOUÇO HETEROGÊNEO DE META-APRENDIZADO EMPILHADO COM PREDIÇÕES FORA DA DOBRA PARA DETECÇÃO DE URLs DE PHISHING [Distrito Federal] 2026.

xvi, 112 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2026).

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

Departamento de Engenharia Elétrica

1. Detecção de phishing

2. Aprendizado empilhado

3. Saturação de benchmark

4. Aprendizado de máquina

I. ENE/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

S. ROCHA, H. (2026). *HOOF-STACK-PUD: ARCABOUÇO HETEROGÊNEO DE META-APRENDIZADO EMPILHADO COM PREDIÇÕES FORA DA DOBRA PARA DETECÇÃO DE URLs DE PHISHING*. Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 112 p.
Publicação: PPEE.MP.124

CESSÃO DE DIREITOS

AUTOR: Hernany Silveira Rocha

TÍTULO: HOOF-STACK-PUD: ARCABOUÇO HETEROGÊNEO DE META-APRENDIZADO EMPILHADO COM PREDIÇÕES FORA DA DOBRA PARA DETECÇÃO DE URLs DE PHISHING.

GRAU: Mestre em Engenharia Elétrica ANO: 2026

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

Hernany Silveira Rocha

Depto. de Engenharia Elétrica (ENE) - FT

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Dedico este trabalho à minha esposa, Daiane Daniele, e aos meus filhos, Lorrany, Thauany e Calebe.

O que aqui está escrito nasceu do tempo que tomei de vocês e da paciência que vocês me devolveram em dobro. Nenhum título dá conta de traduzir isso; esta página, ao menos, o deixa registrado.

AGRADECIMENTOS

Agradeço, em primeiro lugar, ao Senhor Jesus, que sustentou cada etapa deste percurso e me deu forças justamente nos momentos em que elas pareciam ter acabado.

À minha esposa, Daiane Daniele, que insistiu para que eu fizesse o mestrado quando eu ainda hesitava. Se esta dissertação existe, é porque ela enxergou antes de mim aquilo de que eu seria capaz e não aceitou que eu recuasse. Aos meus filhos, Lorrany, Thauany e Calebe, pela compreensão diante das ausências e pela alegria que sempre me esperava do outro lado da porta.

Aos meus pais, minha gratidão por tudo aquilo que veio muito antes deste mestrado e que tornou possível chegar até aqui. Foram eles que me deram a oportunidade de estudar, que confiaram em mim quando escolhi cursar Sistemas de Informação e que sempre apoiaram minhas escolhas. De modo especial, agradeço pelas orações de minha mãe, que tantas vezes me deram forças nos momentos em que o caminho parecia muito difícil.

Ao meu orientador, Prof. Dr. João Paulo Javidi, e à minha coorientadora, Prof.^a Dr.^a Elena Javidi, pela orientação rigorosa e generosa, conduzida desde a Alemanha sem que a distância geográfica jamais se convertesse em distância real. Tive ainda o privilégio de conhecê-los pessoalmente na cidade de Lippstadt, encontro que guardo com especial carinho e que deu rosto, voz e gesto a uma parceria até então mediada por telas.

Aos professores da Universidade de Brasília, cujas aulas, críticas e provocações moldaram o modo como hoje leio, penso e escrevo pesquisa. Levo comigo, mais do que o conteúdo das disciplinas, um jeito mais exigente de olhar para os próprios resultados.

À Universidade de Brasília, pela oportunidade de cursar este mestrado e pelo ambiente acadêmico que tornou possível transformar inquietação em pesquisa.

À Codevasf, empresa onde trabalho, por ter autorizado o tempo dedicado aos estudos. Sem esse apoio institucional, conciliar as atribuições profissionais com as exigências do mestrado teria sido inviável.

Por fim, aos colegas que, direta ou indiretamente, contribuíram para o enriquecimento das pesquisas realizadas ao longo do mestrado. Uma conversa de corredor, uma pergunta desconfortável no momento certo, uma referência compartilhada: pouca coisa neste trabalho é fruto de uma cabeça só.

DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL GENERATIVA

Declaro que utilizei ferramentas de Inteligência Artificial Generativa neste trabalho exclusivamente como apoio à revisão e ao aprimoramento textual. O uso dessas ferramentas restringiu-se à correção de aspectos gramaticais e ortográficos, ao aperfeiçoamento do estilo, à adequação às normas da ABNT e ao auxílio na edição e na formatação do documento em $\text{\LaTeX}$.

Esclareço que tais ferramentas não foram empregadas para produzir conteúdo original, realizar análises, interpretar dados, desenvolver argumentos ou elaborar resultados e conclusões.

Todo o conteúdo apresentado nesta dissertação é de minha autoria, foi por mim revisado criticamente e reflete exclusivamente os procedimentos, as análises e as conclusões que conduzi ao longo da pesquisa. Assumo integral responsabilidade pela originalidade, pela veracidade, pela precisão e pela integridade do texto final, inclusive por eventuais erros, inconsistências ou imprecisões.

Esta declaração está em conformidade com a Portaria CNPq nº 2.664, de 6 de março de 2026.

Ataques de *phishing* seguem explorando *Uniform Resource Locators* (URLs) maliciosas como vetor escalável e de baixo custo, o que motivou o desenvolvimento de detectores baseados em aprendizado de máquina e em aprendizado profundo. A literatura frequentemente reporta desempenho próximo ao limite superior em *benchmarks* de detecção; contudo, quando atributos engenheirados tornam as classes quase separáveis, esses resultados podem refletir saturação do *benchmark*, e não necessariamente ganho arquitetural. A questão é crítica para detectores baseados em empilhamento, cuja justificativa repousa sobre a existência de erro residual passível de correção por evidência complementar.

Este trabalho apresenta o HOOF-Stack-PUD, sigla de *Heterogeneous Out-of-Fold Stacked Meta-Learning Framework for Phishing URL Defense*, arcabouço heterogêneo de meta-aprendizado empilhado com predições fora da dobra, concebido para avaliar quando a combinação de modelos tabulares e lexicais de URL melhora a detecção além de aprendizes isolados fortes. A avaliação percorreu três *benchmarks* de grande porte, com espaços de atributos e procedimentos de coleta distintos: PhiUSIIL, com 235.795 URLs; StealthPhisher, com 336.749 URLs; e HybridPhisher, conjunto construído no âmbito desta pesquisa, com 221.360 URLs. Ao longo de execuções repetidas, os resultados revelaram três regimes. No PhiUSIIL, o melhor meta-modelo mostrou-se estatisticamente indistinguível da linha de base tabular mais forte, resultado compatível com saturação empírica do *benchmark* sob o protocolo adotado. No StealthPhisher, o empilhamento produziu ganho estatisticamente sustentado, porém pequeno. No HybridPhisher, em que os modelos isolados deixam erro residual mais amplo, o meta-aprendiz LightGBM apresentou a melhoria mais clara, com aumento de 0,00608 em medida F1 sobre a melhor linha de base tabular e de 0,01021 sobre a melhor linha de base lexical, em sementes correspondentes. Em conjunto, as análises complementares com linhas de base tabulares robustas, perfilamento de eficiência, blocos de características de URL e de HTML, contribuição dos descritores URLC e *Domain Similarity* e teste de estresse adversarial indicaram maior benefício do empilhamento quando as famílias de modelos fornecem evidência complementar. Esses achados sustentam uma avaliação cautelosa dos detectores empilhados de *phishing*, fundada na saturação do *benchmark*, na complementaridade entre características, na estabilidade estatística, na eficiência e na robustez.

Palavras-chave: Detecção de *phishing*, Detecção de URLs maliciosas, Generalização empilhada, Saturação de *benchmark*, Características de URL e HTML, Complexidade de URL, Robustez adversarial, Conjuntos de dados de *phishing*.

ABSTRACT

Phishing attacks continue to exploit malicious Uniform Resource Locators (URLs) as a scalable and low-cost vector, which has motivated the development of detectors based on machine learning and deep learning. The literature frequently reports near-ceiling performance on detection benchmarks; however, when engineered features make the classes nearly separable, these results may reflect benchmark saturation rather than necessarily an architectural gain. The issue is critical for stacking-based detectors, whose rationale rests on the existence of residual error amenable to correction by complementary evidence.

This work presents HOOF-Stack-PUD, a Heterogeneous Out-of-Fold Stacked Meta-Learning Framework for Phishing URL Defense, designed to assess when combining tabular and lexical URL models improves detection beyond strong standalone learners. The evaluation spanned three large-scale benchmarks with distinct feature spaces and collection procedures: PhiUSIIL, with 235,795 URLs; StealthPhisher, with 336,749 URLs; and HybridPhisher, a dataset built within this research, with 221,360 URLs. Across repeated runs, the results revealed three regimes. On PhiUSIIL, the best meta-model proved statistically indistinguishable from the strongest tabular baseline, a result consistent with empirical benchmark saturation under the adopted protocol. On StealthPhisher, stacking produced a statistically supported but small gain. On HybridPhisher, where standalone models leave a larger residual error, the LightGBM meta-learner showed the clearest improvement, increasing F1 by 0.00608 over the best tabular baseline and by 0.01021 over the best lexical baseline across matched seeds. Taken together, the complementary analyses with strong tabular baselines, efficiency profiling, URL and HTML feature blocks, the contribution of the URLC and Domain Similarity descriptors, and adversarial stress testing indicated that stacking is most beneficial when model families provide complementary evidence. These findings support a cautious evaluation of stacked phishing detectors, grounded in benchmark saturation, feature complementarity, statistical stability, efficiency, and robustness.

Keywords: Phishing detection, Malicious URL detection, Stacked generalization, Benchmark saturation, URL and HTML features, URL complexity, Adversarial robustness, Phishing datasets.

SUMÁRIO

1	INTRODUÇÃO	1
1.1	MOTIVAÇÃO	2
1.2	PROBLEMA DE PESQUISA.....	3
1.3	OBJETIVOS DO TRABALHO.....	4
1.3.1	OBJETIVO GERAL.....	4
1.3.2	OBJETIVOS ESPECÍFICOS.....	4
1.4	CONTRIBUIÇÕES DA PESQUISA	5
1.5	TRABALHOS PUBLICADOS	5
1.6	METODOLOGIA	6
1.7	ORGANIZAÇÃO DO TRABALHO.....	6
2	FUNDAMENTAÇÃO TEÓRICA.....	8
2.1	PHISHING COMO AMEAÇA SOCIOTÉCNICA	8
2.2	A URL COMO SUPERFÍCIE DE ATAQUE E UNIDADE DE ANÁLISE	9
2.2.1	DELIMITAÇÃO TERMINOLÓGICA	10
2.3	MODALIDADES DE EVIDÊNCIA E TAXONOMIA DAS ABORDAGENS DE DETECÇÃO .	10
2.4	ENGENHARIA E SELEÇÃO DE CARACTERÍSTICAS	12
2.4.1	CARACTERÍSTICAS LÉXICAS E ESTRUTURAIS DA URL	12
2.4.2	CARACTERÍSTICAS DE DOMÍNIO, HOSPEDAGEM E SIMILARIDADE	12
2.4.3	CARACTERÍSTICAS DE HTML, CONTEÚDO E HIPERLINKS	13
2.4.4	REPRESENTAÇÕES TEXTUAIS E RECUPERAÇÃO DA INFORMAÇÃO	13
2.4.5	SELEÇÃO DE CARACTERÍSTICAS E REDUÇÃO DE DIMENSIONALIDADE	14
2.5	PARADIGMAS DE APRENDIZADO APLICADOS À DETECÇÃO	14
2.5.1	APRENDIZADO SUPERVISIONADO E MODELOS BASEADOS EM ÁRVORES	14
2.5.2	APRENDIZADO PROFUNDO E APRENDIZADO DE REPRESENTAÇÕES	15
2.5.3	TRANSFORMERS E REPRESENTAÇÕES CONTEXTUAIS.....	15
2.6	APRENDIZADO POR COMITÊS E GENERALIZAÇÃO EMPILHADA	16
2.6.1	PRINCÍPIOS DE COMBINAÇÃO DE MODELOS.....	16
2.6.2	PREDIÇÕES FORA DA DOBRA E PREVENÇÃO DE VAZAMENTO.....	17
2.7	CONJUNTOS DE DADOS E BENCHMARKS DE PHISHING	17
2.7.1	FONTES, ROTULAGEM E COMPOSIÇÃO.....	17
2.7.2	BENCHMARKS ADOTADOS NESTA PESQUISA.....	17
2.7.3	AMEAÇAS À VALIDADE RELACIONADAS AOS DADOS	18
2.8	CRITÉRIOS DE AVALIAÇÃO EXPERIMENTAL	18
2.8.1	MÉTRICAS DE CLASSIFICAÇÃO E INTERPRETAÇÃO OPERACIONAL	18
2.8.2	SATURAÇÃO DO BENCHMARK E ERRO RESIDUAL	19
2.8.3	REPETIBILIDADE, SIGNIFICÂNCIA ESTATÍSTICA E EFICIÊNCIA.....	20
2.8.4	CALIBRAÇÃO DE PROBABILIDADES	20

2.9	CONFIABILIDADE E ROBUSTEZ ADVERSARIAL	20
2.10	SÍNTESE DO REFERENCIAL E APLICAÇÃO NESTA PESQUISA.....	22
3	TRABALHOS RELACIONADOS	24
3.1	ABORDAGENS BASEADAS EM CARACTERÍSTICAS E APRENDIZADO DE MÁQUINA..	24
3.2	ABORDAGENS BASEADAS EM APRENDIZADO PROFUNDO	26
3.3	ABORDAGENS BASEADAS EM TRANSFORMERS.....	28
3.4	ABORDAGENS BASEADAS EM EMPILHAMENTO E META-APRENDIZADO	29
3.5	QUADRO COMPARATIVO DOS TRABALHOS RELACIONADOS	31
3.6	LACUNAS IDENTIFICADAS NA LITERATURA	34
3.7	POSICIONAMENTO DA PESQUISA	35
4	ARQUITETURA DA PROPOSTA	37
4.1	CARACTERIZAÇÃO DA PESQUISA	37
4.2	VISÃO GERAL DO HOOF-STACK-PUD.....	38
4.3	CONSTRUÇÃO DO CONJUNTO HYBRIDPHISHER.....	39
4.3.1	CORPUS LEGÍTIMO	40
4.3.2	CORPUS DE PHISHING	40
4.3.3	BALANCEAMENTO E CONSISTÊNCIA ENTRE CLASSES	41
4.3.4	CRITÉRIO DE INCLUSÃO DE PÁGINAS ATIVAS E VIÉS DE DISPONIBILIDADE	41
4.4	ENGENHARIA DE CARACTERÍSTICAS	42
4.4.1	CARACTERÍSTICAS DERIVADAS DA URL	42
4.4.2	CARACTERÍSTICAS DERIVADAS DO HTML	43
4.4.3	ESCORES DE ALINHAMENTO ENTRE TÍTULO E URL	44
4.4.4	DOMAIN SIMILARITY.....	45
4.4.5	URL COMPLEXITY SCORE.....	47
4.4.6	ANÁLISE EXPLORATÓRIA E REGULARIZAÇÃO DAS CARACTERÍSTICAS	49
4.4.7	ESTRUTURA DISCRIMINATIVA DO ESPAÇO DE CARACTERÍSTICAS	51
4.5	PROTOCOLO DE APRENDIZADO EMPILHADO.....	53
4.5.1	PARTICIONAMENTO E PRÉ-PROCESSAMENTO AJUSTADO NO TREINAMENTO.....	53
4.5.2	PREDITORES DE NÍVEL 0	54
4.5.3	GERAÇÃO DE PROBABILIDADES FORA DA DOBRA	54
4.5.4	META-APRENDIZES DE NÍVEL 1, CALIBRAÇÃO E SELEÇÃO DE LIMIAR	55
4.5.5	CONTROLES CONTRA VAZAMENTO DE INFORMAÇÃO	56
4.6	PROTOCOLO EXPERIMENTAL	56
4.6.1	COMPOSIÇÃO DAS EXECUÇÕES.....	56
4.6.2	CONJUNTOS DE DADOS DE AVALIAÇÃO	58
4.6.3	OTIMIZAÇÃO DE HIPERPARÂMETROS E TRAVAMENTO DAS CONFIGURAÇÕES	58
4.6.4	EXECUÇÕES REPETIDAS E COMPARAÇÕES PAREADAS.....	58
4.6.5	EXPERIMENTOS COMPLEMENTARES	59
4.6.6	MÉTRICAS DE AVALIAÇÃO	60
5	RESULTADOS	62

5.1	AMBIENTE COMPUTACIONAL.....	62
5.2	CRITÉRIO DE LEITURA DOS RESULTADOS.....	62
5.3	DESEMPENHO NAS PARTIÇÕES DE TESTE RETIDAS	63
5.4	EXECUÇÕES REPETIDAS E SIGNIFICÂNCIA ESTATÍSTICA	66
5.5	COMPARAÇÃO COM LINHAS DE BASE TABULARES ROBUSTAS	69
5.6	ANÁLISE DE COMPONENTES E BLOCOS DE ATRIBUTOS	71
5.7	CONTRIBUIÇÃO E SENSIBILIDADE DO ESCORE URLC.....	73
5.8	ANÁLISE DE ESTRESSE ADVERSARIAL	76
5.9	CONTRIBUIÇÃO DA SIMILARIDADE DE DOMÍNIO	79
5.10	DISCUSSÃO DOS RESULTADOS.....	80
5.10.1	SÍNTESE DAS EVIDÊNCIAS.....	80
5.10.2	CONFRONTO COM A LITERATURA	81
6	CONCLUSÕES.....	83
6.1	LIMITAÇÕES DO TRABALHO	85
6.2	RECOMENDAÇÕES PARA TRABALHOS FUTUROS	86
	REFERÊNCIAS BIBLIOGRÁFICAS.....	88
	APÊNDICES.....	93
A	CONFIGURAÇÕES DE HIPERPARÂMETROS DOS MODELOS.....	94
A.1	HYBRIDPHISHER.....	94
A.2	PHIUSIIL	96
A.3	STEALTHPHISHER	97
A.4	META-APRENDIZES DE NÍVEL 1	99
B	IMPLEMENTAÇÃO DOS DESCRITORES COMPOSTOS	100
B.1	URL COMPLEXITY SCORE.....	100
B.2	ÍNDICE DE SIMILARIDADE ENTRE DOMÍNIOS ADAPTADO DO USI DE PRASAD E CHANDRA (2024).....	102
C	PROTOCOLO DE EMPILHAMENTO EM PSEUDOCÓDIGO	104
C.1	SIMILARIDADE MÁXIMA CONTRA O CONJUNTO DE REFERÊNCIA.....	104
C.2	APRENDIZADO EMPILHADO COM PREDIÇÕES FORA DA DOBRA	105
D	CONSTRUÇÃO DO CONJUNTO HYBRIDPHISHER.....	107
D.1	URLS LEGÍTIMAS	107
D.2	URLS DE PHISHING	109
D.3	COMPOSIÇÃO DO CONJUNTO E CÁLCULO DOS ATRIBUTOS	109
D.4	CUSTO DE AQUISIÇÃO DA EVIDÊNCIA WEB	111
D.5	DIFERENÇAS DE COLETA ENTRE AS CLASSES	112

LISTA DE FIGURAS

4.1	Visão geral do arcabouço HOOF-Stack-PUD	39
4.2	Diagramas de caixa agrupados das características numéricas mais assimétricas identificadas na análise exploratória	49
4.3	Influência das características por classe, derivada dos modelos treinados	51
4.4	Atributos tabulares mais bem classificados pela estatística qui-quadrado após a regularização	52
4.5	Distribuições pareadas dos atributos tabulares mais bem classificados após a regularização, estratificadas por classe	53
4.6	Composição das execuções experimentais	57
5.1	Média e desvio padrão de F1 em execuções repetidas para a configuração com o conjunto completo de atributos	65
5.2	Variação de F1 entre sementes em D3 para a configuração com o conjunto completo de atributos	66
5.3	Média e desvio padrão de PR-AUC em execuções repetidas nos três <i>benchmarks</i>	67
5.4	Diferença de F1 por semente em D3 entre o melhor meta-modelo e o melhor modelo de base, com limiar de decisão fixo em 0.5	68
5.5	Variações do teste de estresse adversarial em nível de <i>pipeline</i> no HybridPhisher	78
5.6	Trajetória de Recall entre as condições limpa e adversariais no HybridPhisher	79

LISTA DE TABELAS

2.1	Modalidades de evidência empregadas na detecção de <i>phishing</i>	11
3.1	Quadro comparativo dos trabalhos relacionados	31
4.1	Atributos derivados do HTML, por grupo funcional	44
4.2	Escores representativos de similaridade de domínio para variantes visualmente manipuladas	47
4.3	Mapeamento de atributos produzido pela regularização	50
4.4	Correspondência entre manifestos de experimento e análises do Capítulo 5	57
4.5	Composição dos conjuntos de dados e dimensionalidade das características . . .	58
4.6	Protocolo de otimização de hiperparâmetros	59
5.1	Desempenho no conjunto PhiUSIIL com limiar de decisão fixo (0.5) em execuções repetidas	63
5.2	Desempenho no conjunto StealthPhisher com limiar de decisão fixo (0.5) em execuções repetidas	64
5.3	Desempenho no conjunto HybridPhisher com limiar de decisão fixo (0.5) em execuções repetidas	64
5.4	Comparação estatística entre os melhores meta-modelos e os modelos de base em execuções repetidas, com limiar de decisão fixo (0.5)	66
5.5	Perfil de eficiência em nível de modelo no HybridPhisher	69
5.6	Comparação de linhas de base tabulares isoladas em D1–D3	70
5.7	Eficiência computacional das linhas de base tabulares isoladas em D1–D3	71
5.8	Desempenho dos modelos tabulares em D1–D3 por bloco de atributos	72
5.9	Contribuição discriminativa incremental do URLC por bloco de atributos tabulares no HybridPhisher	74
5.10	Desempenho tabular com e sem o URLC no HybridPhisher	75
5.11	Análise de sensibilidade dos pesos do URLC no HybridPhisher	76
5.12	Desempenho do teste de estresse adversarial em nível de <i>pipeline</i> no HybridPhisher	77
5.13	Contribuição discriminativa incremental da similaridade de domínio por bloco de atributos tabulares no HybridPhisher	80

A.1	Hiperparâmetros finais travados no HybridPhisher.	94
A.2	Hiperparâmetros finais travados no PhiUSIL.	96
A.3	Hiperparâmetros finais travados no StealthPhisher.	97
A.4	Parâmetros fixos dos meta-aprendizes de Nível 1.	99
D.1	Etapas de construção do HybridPhisher.	107
D.2	Tempo de aquisição, tamanho do HTML e tempo de extração dos atributos HTML.	111
D.3	Atributos sensíveis à forma de coleta, por classe.	112

LISTA DE SÍMBOLOS

Símbolos Latinos

$\emptyset$	Conjunto vazio
B	Número de aprendizes de base ativos no Nível 0
d_{obs}	Domínio observado na URL sob análise
d_{ref}	Domínio legítimo tomado como referência
F_1	Média harmônica entre precisão e <i>recall</i>
F_{HTML}	Bloco de atributos derivados do HTML
F_{URL}	Bloco de atributos derivados da URL
$F_{\text{URL+HTML}}$	Bloco combinado de atributos de URL e de HTML
H	Sequência ordenada de rótulos do <i>host</i>
i	Índice de posição de caractere
$\mathbb{I}[\cdot]$	Função indicadora
J	Número de execuções pareadas, uma por semente, nas comparações estatísticas
k	Número de características retidas na seleção
L_0, L_2	Normas empregadas em medidas de perturbação adversarial
m	Aprendiz de Nível 0
$\mathcal{M}$	Conjunto de índices de caracteres correspondentes
N	Maior comprimento entre dois domínios comparados
N_{test}	Número de amostras da partição de teste
N_{train}	Número de amostras da partição de treinamento
p	Número de parâmetros da cadeia de consulta
p_{NB}	Valor p do teste t pareado com a correção de variância de Nadeau e Bengio
p_{perm}	Valor p do teste de permutação por troca de sinais
$\hat{p}_{\text{meta}}$	Probabilidade calibrada produzida no Nível 1
P_{max}	Limite superior de parâmetros na normalização da consulta
P_{OOF}	Matriz de probabilidades fora da dobra
P_{test}	Matriz de probabilidades sobre a partição de teste
r	Representação normalizada do domínio raiz
r_p	Razão entre chaves únicas e total de parâmetros de consulta
s	Rótulo de subdomínio
S_{maxtok}	Componente de dominância do maior <i>token</i> no URLC
S_{mix}	Componente de mistura alfanumérica de <i>tokens</i> no URLC
S_{query}	Componente de complexidade da cadeia de consulta no URLC
S_{subdom}	Componente de concentração de subdomínios no URLC
t	<i>Token</i> obtido da URL ou do título da página
T	Conjunto de <i>tokens</i> de <i>host</i> e de caminho
u	URL submetida à análise

$w(i; N)$	Peso posicional na comparação entre domínios
y	Rótulo verdadeiro da amostra
$\hat{y}$	Rótulo predito pelo modelo

Símbolos Gregos

Δ	Diferença pareada entre duas medidas sobre sementes correspondentes
τ	Limiar de decisão aplicado às probabilidades preditas
χ^2	Estatística qui-quadrado, empregada na seleção de características

Subscritos

c	Modelo lexical em nível de caractere
m	Modelo de meta-aprendizado, no Nível 1
$\max$	Valor máximo
t	Modelo tabular
test	Relativo à partição de teste
train	Relativo à partição de treinamento
w	Modelo lexical em nível de <i>token</i> ou de palavra

Sobrescritos

$*$	Valor derivado da partição de treinamento, como em τ^*
(m)	Relativo ao aprendiz de Nível 0 identificado por m
$\hat{}$	Estimativa produzida por um modelo

Siglas

ACC	<i>Accuracy</i> (acurácia)
ANN	<i>Artificial Neural Network</i> (rede neural artificial)
ANOVA	Análise de variância
ASR	<i>Attack Success Rate</i> (taxa de sucesso de ataque)
AUC	<i>Area Under the Receiver Operating Characteristic Curve</i> (área sob a curva ROC)
BERT	<i>Bidirectional Encoder Representations from Transformers</i>
BRNN	<i>Bidirectional Recurrent Neural Network</i> (rede neural recorrente bidirecional)
CNN	<i>Convolutional Neural Network</i> (rede neural convolucional)
D1	Conjunto de dados PhiUSIIL
D2	Conjunto de dados StealthPhisher

D3	Conjunto de dados HybridPhisher
DNN	<i>Deep Neural Network</i> (rede neural profunda)
DNS	<i>Domain Name System</i> (sistema de nomes de domínio)
DOM	<i>Document Object Model</i> (modelo de objeto de documento)
DTMS	<i>DomainTitleMatchScore</i> (escore de alinhamento entre título e domínio primário)
ERG-SVC	<i>Expandable Random Gradient Stacked Voting Classifier</i>
FNR	<i>False Negative Rate</i> (taxa de falsos negativos)
FPR	<i>False Positive Rate</i> (taxa de falsos positivos)
FT-Transformer	Arquitetura <i>Transformer</i> adaptada a dados tabulares
GBDT	<i>Gradient Boosting Decision Tree</i> (árvore de decisão com <i>gradient boosting</i>)
GNN	<i>Graph Neural Network</i> (rede neural em grafos)
HOOF-Stack-PUD	<i>Heterogeneous Out-of-Fold Stacked Meta-Learning Framework for Phishing URL Defense</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i> (protocolo de transferência de hipertexto)
IC	Intervalo de confiança
IDNA	<i>Internationalized Domain Names in Applications</i>
IoT	<i>Internet of Things</i> (Internet das Coisas)
IP	<i>Internet Protocol</i> (protocolo de Internet)
KNN	<i>K-Nearest Neighbors</i> (K-vizinhos mais próximos)
KS	Estatística de Kolmogorov–Smirnov
LDA	<i>Latent Dirichlet Allocation</i> (alocação latente de Dirichlet)
LightGBM	<i>Light Gradient Boosting Machine</i>
LSTM	<i>Long Short-Term Memory</i> (memória longa de curto prazo)
OOF	<i>Out-of-Fold</i> (fora da dobra)
PLN	Processamento de linguagem natural
PR-AUC	<i>Area Under the Precision–Recall Curve</i> (área sob a curva de precisão–recall)
Pre	<i>Precision</i> (precisão)
PRISMA	<i>Preferred Reporting Items for Systematic Reviews and Meta-Analyses</i>
RFC	<i>Request for Comments</i>
RNN	<i>Recurrent Neural Network</i> (rede neural recorrente)
ROC	<i>Receiver Operating Characteristic</i> (característica de operação do receptor)
SMOTE	<i>Synthetic Minority Over-sampling Technique</i> (sobreamostragem sintética da classe minoritária)
SSL	<i>Secure Sockets Layer</i> (camada de sockets segura)
SVM	<i>Support Vector Machine</i> (máquina de vetores de suporte)
TabNet	Arquitetura profunda para dados tabulares com seleção esparsa por atenção
TabTransformer	Arquitetura tabular baseada em autoatenção
TF-IDF	<i>Term Frequency–Inverse Document Frequency</i> (frequência do termo e frequência inversa nos documentos)
TPR	<i>True Positive Rate</i> (taxa de verdadeiros positivos)

t-SNE	<i>t-Distributed Stochastic Neighbor Embedding</i>
URL	<i>Uniform Resource Locator</i> (localizador uniforme de recursos)
URLC	<i>URL Complexity Score</i> (escore de complexidade da URL)
USI	<i>URL Similarity Index</i> (índice de similaridade de URL)
UTMS	<i>URLTitleMatchScore</i> (escore de alinhamento entre título e domínio raiz)
XGBoost	<i>eXtreme Gradient Boosting</i>

Identificação dos Modelos

CNN _c	Rede neural convolucional em nível de caractere
CNN _w	Rede neural convolucional em nível de <i>token</i>
FTTr _t	FT-Transformer sobre atributos tabulares
LGBM _m	LightGBM como meta-aprendiz de Nível 1
LGBM _t	LightGBM sobre atributos tabulares
LR _c	Regressão logística sobre n-gramas de caractere
LR _m	Regressão logística <i>elastic net</i> como meta-aprendiz de Nível 1
RF _t	<i>Random Forest</i> sobre atributos tabulares
TabNet _t	TabNet sobre atributos tabulares
TabTr _t	TabTransformer sobre atributos tabulares
XGB _t	XGBoost sobre atributos tabulares

1 INTRODUÇÃO

O *phishing* continua sendo uma das ameaças mais persistentes do ambiente digital. Sua eficácia não decorre de uma única falha técnica, mas da combinação entre recursos de dissimulação e estratégias de engenharia social que levam o usuário a revelar credenciais, informações financeiras e outros dados sensíveis. As técnicas empregadas pelos atacantes mudam com frequência, porém o mecanismo central permanece o mesmo: explorar limitações humanas, semelhanças visuais, mensagens persuasivas e artifícios técnicos que conferem aparência de legitimidade ao conteúdo malicioso (Baki; Verma, 2023; Kulkarni; Balachandran; Das, 2025). Por isso, campanhas de conscientização, embora necessárias, são insuficientes quando empregadas isoladamente, o que motiva o uso de mecanismos automatizados capazes de identificar indícios de fraude antes que a interação do usuário resulte em comprometimento.

Nessa cadeia de ataque, as *Uniform Resource Locators* (URLs) ocupam posição central. Além de conduzir o usuário ao recurso eletrônico, elas podem ser criadas e distribuídas a baixo custo, modificadas em poucos minutos e estruturadas para imitar serviços legítimos. Subdomínios enganosos, cadeias extensas de caracteres, *tokens* associados a marcas conhecidas, parâmetros de consulta, redirecionamentos, encurtadores, alterações lexicais e infraestrutura de hospedagem temporária são recursos que dificultam a distinção entre endereços legítimos e maliciosos (Tian et al., 2025; Kailas; Roopalakshmi, 2025; Zieni; Massari; Calzarossa, 2023). Relatórios periódicos de atividade confirmam a escala operacional dessas campanhas e mostram que a ameaça permanece relevante, o que exige soluções de detecção capazes de conciliar acurácia, eficiência, estabilidade e capacidade de resposta a mudanças no comportamento dos atacantes (Anti-Phishing Working Group, 2025).

Diante desse cenário, técnicas de aprendizado de máquina e de aprendizado profundo tornaram-se predominantes na detecção de URLs e de páginas de *phishing*. A literatura apresenta abordagens baseadas em atributos lexicais, estruturais, de hospedagem e de conteúdo *HyperText Markup Language* (HTML), combinadas com classificadores baseados em árvores, métodos de seleção de atributos, representações profundas de sequências de caracteres, arquiteturas convolucionais, modelos baseados em *Transformers* e técnicas de generalização empilhada (Aljofey et al., 2022; Asiri et al., 2023; Aslam et al., 2024; Aljofey et al., 2025; Kustiawan; Ghauth, 2025a; Kalabarige et al., 2022; Le et al., 2018; He et al., 2024b; Motiur Rahman; Islam; Jabiullah, 2020). Em boa parte desses estudos, os valores relatados de acurácia, medida F1 e área sob a curva característica de operação do receptor (*Receiver Operating Characteristic*, ROC) aproximam-se do limite superior da escala.

É nesse contexto que se insere esta dissertação. O trabalho investiga em quais condições a combinação heterogênea de modelos por meio de probabilidades *out-of-fold* produz ganhos mensuráveis sobre classificadores autônomos fortes e, para conduzir a investigação, propõe o HOOF-Stack-PUD, sigla de *Heterogeneous Out-of-Fold Stacked Meta-Learning Framework for Phishing URL Defense*, arcabouço de meta-aprendizado empilhado que integra modelos tabulares, modelos lexicais de URL e ramos convolucionais. As seções seguintes apresentam a motivação da pesquisa, o problema e a hipótese que a orientam, os objetivos geral e específicos, as contribuições da pesquisa, os trabalhos publicados no período, o percurso metodológico adotado e a organização do texto.

1.1 MOTIVAÇÃO

A primeira motivação desta pesquisa é a permanência do *phishing* como ameaça escalável e de baixo custo. Registrar domínios, hospedar páginas temporárias, alterar rapidamente a estrutura de uma URL e distribuir links por canais diversos são operações simples, o que permite renovar campanhas com elevada frequência. Listas de bloqueio e regras estáticas, embora úteis, dificilmente acompanham essa velocidade e deixam desprotegidas as URLs recém-criadas ou ainda não catalogadas. Métodos de aprendizado de máquina oferecem alternativa ao inferir padrões a partir de atributos observáveis, mas sua confiabilidade depende da qualidade dos dados, das representações escolhidas e do protocolo de avaliação adotado.

A segunda motivação nasce do uso crescente de conjuntos de dados com elevado grau de separabilidade. A engenharia de atributos consegue transformar sinais lexicais, estruturais, de domínio e de HTML em variáveis altamente discriminativas. Embora essa separabilidade favoreça o desempenho preditivo, ela reduz a capacidade do benchmark de discriminar contribuições arquiteturais. Quando vários modelos alcançam desempenho quase perfeito, torna-se difícil distinguir uma contribuição arquitetural genuína de um resultado induzido pelo próprio *benchmark*. A literatura da área apresenta tanto métodos fundamentados em atributos de URL e HTML quanto arquiteturas profundas e modelos empilhados, mas nem sempre explicita se o ganho decorre da complementaridade entre modelos ou da predominância de um conjunto de atributos já suficientemente discriminativo (Aljofey et al., 2022; Aslam et al., 2024; Kustiawan; Ghauth, 2025a; Kalabariga et al., 2022; Motiur Rahman; Islam; Jabiullah, 2020).

A terceira motivação diz respeito ao rigor estatístico. Comparações apoiadas em uma única divisão de treino e teste são sensíveis à semente utilizada e à composição das partições. Em cenários de desempenho elevado, diferenças mínimas podem parecer relevantes quando apresentadas apenas como valores pontuais. Execuções repetidas com sementes fixas, deltas pareados, intervalos de confiança e testes de significância permitem verificar se o comportamento observado é estável e se a magnitude do ganho tem relevância prática.

A quarta motivação é a necessidade de examinar eficiência e robustez ao lado da capacidade preditiva. Um modelo com pequena vantagem de F1 pode exigir mais tempo de treinamento, inferência mais lenta, múltiplos artefatos e maior complexidade operacional. Da mesma forma, uma solução muito precisa sobre dados não perturbados pode degradar de modo acentuado quando o atacante modifica evidências específicas. Estudos sobre ataques adversariais a detectores de URLs, de anúncios e de páginas de *phishing* mostram que as avaliações convencionais não cobrem integralmente as condições de evasão encontradas no espaço do problema (Kulkarni et al., 2025; Sabir et al., 2024; Lo Re; Morana; Rizzo, 2026; Nowroozi et al., 2023; Pillai et al., 2024). Uma proposta de defesa precisa, portanto, ser analisada não apenas pelo melhor resultado agregado, mas também por sua estabilidade, sua eficiência e sua tolerância à degradação de ramos específicos.

Por fim, a pesquisa se justifica pela escassez de conjuntos de dados que preservem simultaneamente a URL textual e os atributos derivados da URL e do HTML para as mesmas instâncias, condição necessária à comparação controlada entre representações lexicais, atributos tabulares e combinações heterogêneas. A resposta desta pesquisa a essa lacuna é apresentada na Seção 1.4.

1.2 PROBLEMA DE PESQUISA

A literatura demonstra que modelos de aprendizado de máquina, de aprendizado profundo e de generalização empilhada alcançam desempenho elevado na detecção de *phishing*. Permanece, contudo, uma dificuldade de interpretação: não está claro em quais condições o ganho de um meta-modelo decorre de complementaridade real entre famílias de classificadores e em quais condições o *benchmark* já se encontra saturado por atributos engenheirados. Comparar apenas os melhores valores de acurácia ou de F1 não resolve a questão, sobretudo quando as diferenças são pequenas e não vêm acompanhadas de análise estatística, avaliação de eficiência e testes de robustez.

Dessa lacuna decorre o problema que orientou esta pesquisa:

Em quais condições um arcabouço heterogêneo de meta-aprendizado empilhado, treinado com probabilidades *out-of-fold*, produz ganhos estatisticamente sustentados e compatíveis com o custo operacional adicional sobre modelos autônomos fortes na detecção de URLs de *phishing*?

O problema desdobra-se nas seguintes questões de pesquisa:

1. Como o grau de saturação de diferentes *benchmarks* afeta o ganho obtido pelo empilhamento?
2. A combinação de modelos tabulares e lexicais reduz de forma consistente os erros residuais deixados pelos melhores modelos de nível base?
3. Os ganhos observados permanecem estáveis em execuções repetidas e são sustentados por comparações estatísticas pareadas?
4. Qual é a contribuição dos blocos de atributos de URL, de HTML e de URL+HTML para a separabilidade dos conjuntos de dados?
5. Qual é a contribuição incremental dos descritores *URL Complexity Score* (URLC) e *Domain_Similarity* diante dos demais atributos disponíveis?
6. Quais são os custos de treinamento e de inferência associados aos modelos autônomos e aos meta-modelos?
7. Como os modelos de nível base e os meta-modelos se comportam quando as evidências derivadas da URL e do HTML são submetidas a perturbações controladas?

A hipótese formulada para orientar a investigação foi a seguinte:

O meta-aprendizado empilhado tende a produzir ganhos mensuráveis sobre modelos autônomos fortes quando as famílias de classificadores fornecem probabilidades complementares e o *benchmark* preserva erros residuais suficientes; em contrapartida, seu benefício tende a ser nulo ou marginal quando os atributos tabulares já tornam o conjunto de dados quase separável.

A hipótese foi examinada em três *benchmarks* com espaços de atributos e níveis de saturação distintos, por meio de probabilidades *out-of-fold*, partições de teste preservadas, execuções repetidas, deltas pareados, intervalos de confiança, análise de eficiência e testes de estresse adversarial.

1.3 OBJETIVOS DO TRABALHO

1.3.1 Objetivo Geral

Desenvolver e avaliar o HOOF-Stack-PUD, arcabouço heterogêneo de meta-aprendizado empilhado baseado em probabilidades *out-of-fold*, com a finalidade de caracterizar as condições em que a combinação de modelos tabulares e lexicais melhora a detecção de URLs de *phishing* em relação a classificadores autônomos fortes.

1.3.2 Objetivos Específicos

Para alcançar o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- **Construir um conjunto de dados híbrido:** reunir, a partir de fontes verificadas de *phishing*, um conjunto que preserve a URL textual e os atributos derivados da URL e do HTML para as mesmas instâncias.
- **Implementar um protocolo de empilhamento protegido contra vazamento:** treinar os meta-modelos exclusivamente sobre predições *out-of-fold*, mantendo isolada a partição reservada para o teste final.
- **Comparar o arcabouço em *benchmarks* com graus distintos de separabilidade:** avaliar o HOOF-Stack-PUD em PhiUSIIL (D1), StealthPhisher (D2) e HybridPhisher (D3), confrontando os meta-modelos com linhas de base tabulares e lexicais robustas.
- **Verificar a estabilidade estatística dos ganhos:** conduzir execuções repetidas com sementes fixas e analisar os resultados por deltas pareados, intervalos de confiança e testes de significância.
- **Quantificar a contribuição das representações:** medir o poder discriminativo dos blocos de atributos de URL, de HTML e de URL+HTML e a contribuição incremental dos descritores compostos propostos.
- **Caracterizar o custo computacional do arcabouço:** medir os tempos de treinamento e de inferência de cada componente, além da latência normalizada por amostra.
- **Avaliar a sensibilidade do *pipeline* a perturbações:** submeter as evidências derivadas da URL e do HTML a perturbações controladas e observar a degradação dos modelos de nível base e dos meta-modelos.

1.4 CONTRIBUIÇÕES DA PESQUISA

As contribuições principais desta pesquisa, demonstradas nos próximos capítulos, são as seguintes:

- **Conjunto de dados HybridPhisher:** conjunto com 221.360 URLs, construído a partir de fontes verificadas de *phishing*, entre elas o Phishing Database e o PhishTank (Phishing Database Project, 2026; Cisco Talos Intelligence Group, 2026), que preserva, para as mesmas instâncias, a URL textual e os atributos derivados da URL e do HTML. Nos experimentos desta dissertação, o HybridPhisher preservou erro residual suficiente para a comparação controlada entre representações lexicais, atributos tabulares e combinações heterogêneas, conforme a Seção 4.3.
- **Arcabouço HOOF-Stack-PUD:** arcabouço de meta-aprendizado empilhado que combina, em nível de probabilidade, ramos tabulares e lexicais de modalidades distintas, treinado com predições *out-of-fold* e protegido contra vazamento, uma vez que pré-processamento, seleção de características, calibração e limiares são ajustados exclusivamente sobre a partição de treinamento, conforme a Seção 4.5.
- **Caracterização do benefício condicional do empilhamento:** a avaliação em três *benchmarks* com níveis distintos de saturação, apoiada em execuções repetidas e em comparações estatísticas pareadas, caracterizou, sob o protocolo adotado, três regimes de benefício do empilhamento: nulo em PhiUSIIL, pequeno porém consistente em StealthPhisher e mais nítido em HybridPhisher, conforme a Seção 5.4.

Complementam essas contribuições, em caráter secundário:

- **Descritores compostos interpretáveis:** o *URL Complexity Score* (URLC), proposto nesta pesquisa, e o *Domain_Similarity*, adaptado do índice de similaridade de Prasad e Chandra (2024), tiveram sua contribuição medida por comparações pareadas com e sem o atributo em cada bloco de características, conforme as Seções 5.7 e 5.9.
- **Avaliação integrada de eficiência e robustez:** o mesmo protocolo mede os custos de treinamento e de inferência de cada componente e a degradação do *pipeline* sob perturbações controladas de URL e de HTML. Essa avaliação mostrou que o custo do detector é determinado pelos ramos de base, e não pela camada de fusão, e que a fusão reduz, sem eliminar, a sensibilidade a perturbações concentradas em uma única fonte de evidência, conforme as Seções 5.4 e 5.8.

1.5 TRABALHOS PUBLICADOS

Durante o período de desenvolvimento desta dissertação, foram publicados dois trabalhos científicos em áreas relacionadas à aplicação de métodos de aprendizado de máquina. Embora não decorram dos experimentos do HOOF-Stack-PUD, eles foram desenvolvidos em paralelo a esta pesquisa e compartilham abordagens metodológicas referentes à construção de conjuntos de dados, à aplicação de técnicas de aprendizado de máquina e à avaliação comparativa de modelos:

1. ROCHA, Hernany S.; DA COSTA, Elena J.; DUARTE, Jaqueline D.; DA COSTA, Joao Paulo J. **A New Dataset and Neural Benchmark for Multi-Label Classification of Modern Slavery Litigation**. *IEEE Access*, v. 13, p. 180739–180755, 2025. DOI: 10.1109/ACCESS.2025.3619928.
2. DUARTE, Jaqueline Damacena; DA COSTA, Elena Javidi; DA COSTA, Joao Paulo Javidi; SCHWEIZER SILVESTRE, Ana Sofia; CANEDO, Edna Dias; ROCHA, Hernany Silveira. **AI-driven modern slavery detection for supply chain: A cross-jurisdictional legal text analysis**. *Machine Learning with Applications*, v. 23, art. 100827, 2026. DOI: 10.1016/j.mlwa.2025.100827.

1.6 METODOLOGIA

Esta é uma pesquisa aplicada, de abordagem quantitativa e caráter experimental, cuja caracterização detalhada quanto à natureza, aos objetivos e aos procedimentos técnicos é apresentada na Seção 4.1. Esta seção limita-se a descrever o percurso adotado.

O trabalho partiu da revisão da literatura, que delimitou o problema e identificou as lacunas a serem investigadas. Em seguida, foram selecionados os conjuntos PhiUSIL e StealthPhisher e construído o HybridPhisher, o que forneceu três *benchmarks* com procedimentos de construção e espaços de atributos distintos. Sobre essa base, foram organizadas as representações tabulares e lexicais, treinados os modelos de nível base e gerados os meta-modelos a partir das probabilidades *out-of-fold*, sempre com a partição de teste mantida isolada até a avaliação final.

A etapa final consistiu na análise. O protocolo completo foi repetido com sementes fixas, e os resultados foram comparados de forma pareada, com o acréscimo de experimentos complementares que contextualizam o ganho do empilhamento. O detalhamento do arcabouço, das arquiteturas, dos hiperparâmetros, das métricas e dos testes estatísticos é apresentado no Capítulo 4.

1.7 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado em seis capítulos, incluindo esta introdução.

O Capítulo 2 apresenta a fundamentação teórica, abordando a natureza sociotécnica do *phishing*, o papel da URL como superfície de ataque, as modalidades de evidência utilizadas pelos detectores, a engenharia e a seleção de características, os paradigmas de aprendizado aplicados à área, os princípios do aprendizado por comitês, os conjuntos de dados adotados como *benchmark*, os critérios de avaliação experimental e a questão da robustez adversarial.

O Capítulo 3 descreve os trabalhos relacionados, organizados em quatro linhas de investigação: abordagens baseadas em características e aprendizado de máquina, abordagens baseadas em aprendizado profundo, detectores fundamentados em *Transformers* e estratégias de empilhamento e meta-aprendizado. O capítulo encerra com o quadro comparativo dos estudos, as lacunas identificadas e o posicionamento desta pesquisa.

O Capítulo 4 descreve a arquitetura proposta, detalhando a construção do conjunto HybridPhisher, o esquema de características adotado e a análise exploratória que fundamentou sua regularização, o protocolo de aprendizado empilhado com geração de probabilidades *out-of-fold* e o protocolo experimental que orienta as avaliações.

O Capítulo 5 apresenta os resultados obtidos nos três *benchmarks*, discute os regimes de saturação, compara os meta-modelos com aprendizes autônomos fortes e analisa os blocos de atributos, a contribuição dos descritores compostos, o perfil de eficiência e o comportamento do *pipeline* sob perturbações adversariais controladas.

Por fim, o Capítulo 6 consolida as conclusões do trabalho, retoma o problema, a hipótese e os objetivos, explicita as limitações do estudo e propõe direções para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo reúne os fundamentos necessários à compreensão da detecção de *phishing* baseada em URLs, em características derivadas de páginas Web e em aprendizado de máquina. A exposição parte da natureza do *phishing* e do papel que a URL desempenha na materialização e na distribuição desses ataques. Em seguida, examina as modalidades de evidência utilizadas pelos sistemas de detecção, os processos de engenharia e seleção de características, os paradigmas de aprendizado empregados na área, os princípios do aprendizado por comitês, a natureza dos conjuntos de dados adotados como *benchmark*, os critérios de avaliação experimental e a questão da robustez adversarial. Ao final, a Seção 2.10 explicita como cada um desses fundamentos orientou o desenho da pesquisa.

Neste capítulo, as obras são mobilizadas como apoio a definições, taxonomias e princípios. Os estudos primários que constituem o estado da arte imediato do objeto investigado são descritos e quantificados individualmente no Capítulo 3, de modo a evitar que um mesmo trabalho seja narrado duas vezes.

2.1 PHISHING COMO AMEAÇA SOCIOTÉCNICA

O *phishing* é uma modalidade de ataque de engenharia social na qual o atacante engana o usuário por meio de mensagens falsas, distribuídas com frequência por correio eletrônico ou por plataformas de mídia social. Essas mensagens costumam conter URLs que conduzem a vítima a páginas fraudulentas concebidas para aparentar legitimidade, com o propósito de obter informações sensíveis, como credenciais de acesso e dados de pagamento, ou de induzir o *download* e a instalação de software malicioso (Asiri et al., 2023; Kulkarni; Balachandran; Das, 2025; Zieni; Massari; Calzarossa, 2023).

O ataque é qualificado como sociotécnico porque opera em duas camadas. A camada técnica é constituída pela infraestrutura de hospedagem, pelo endereço utilizado e pelo conteúdo da página. A camada humana é constituída pela decisão que a vítima toma diante da mensagem recebida (Baki; Verma, 2023; Kulkarni; Balachandran; Das, 2025). Essa dupla natureza distingue o *phishing* dos ataques que exploram exclusivamente falhas de software e explica por que sua mitigação não pode se apoiar em um único tipo de medida.

No que diz respeito à camada humana, a revisão sistemática e a meta-análise conduzidas por Baki e Verma (2023) constituem a síntese mais abrangente disponível. Os autores revisaram estudos com usuários publicados ao longo de dezesseis anos e encontraram divergências entre os estudos primários quanto à influência de fatores demográficos: mais da metade dos trabalhos que analisaram o efeito da idade não identificou relação estatisticamente significativa, e alguns relataram melhor desempenho de usuários mais velhos, enquanto outros relataram o oposto. A análise agregada, por sua vez, indicou relação significativa entre idade e suscetibilidade, maior suscetibilidade entre participantes do sexo feminino e melhora significativa da capacidade de detecção após treinamento. O achado interessa duplamente a esta pesquisa: reconhece o valor das medidas de conscientização e, ao mesmo tempo, delimita seu alcance, pois o trei-

namento reduz mas não elimina a suscetibilidade. É essa lacuna residual que justifica a existência de mecanismos automatizados capazes de atuar antes da decisão do usuário.

Quanto à camada técnica, os relatórios periódicos do *Anti-Phishing Working Group* documentam a escala e a continuidade da atividade observada em produção (Anti-Phishing Working Group, 2025), e as revisões recentes registram que o *phishing* permanece ativo e adaptativo, com mudanças frequentes de infraestrutura, aparência, domínios e mecanismos de entrega (Kailas; Roopalakshmi, 2025; Kulkarni; Balachandran; Das, 2025; Tian et al., 2025). A revisão conduzida sob o protocolo *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA) por Kailas e Roopalakshmi (2025) acrescenta que ataques de *phishing* respondem por parcela expressiva das violações de dados reportadas e que a URL constitui o principal ponto de entrada da maioria dos vetores. Já Tian et al. (2025) descrevem a detecção de URLs maliciosas como uma corrida armamentista prolongada entre defensores e atacantes, na qual obter um panorama atualizado do estado dessa disputa é, em si, um problema de pesquisa.

Esse caráter dinâmico impõe duas exigências aos sistemas de detecção. A primeira é reconhecer padrões que não dependam apenas de listas previamente conhecidas. A segunda é verificar se o desempenho observado em um conjunto de dados permanece válido quando o cenário de coleta, o período temporal ou a estratégia do atacante se modificam (Kailas; Roopalakshmi, 2025; Skula; Kvet, 2024; Tian et al., 2025).

2.2 A URL COMO SUPERFÍCIE DE ATAQUE E UNIDADE DE ANÁLISE

A URL ocupa posição central na superfície de ataque do *phishing*. Trata-se de um artefato barato de produzir, trivial de distribuir e facilmente manipulável por estratégias léxicas, estruturais e de hospedagem que imitam serviços legítimos (Kailas; Roopalakshmi, 2025; Tian et al., 2025; Zieni; Massari; Calzarossa, 2023). Além de indicar o recurso a ser acessado, a própria cadeia de caracteres carrega sinais léxicos e estruturais úteis à detecção automatizada.

Do ponto de vista sintático, a URL é composta por partes com funções distintas: esquema, autoridade, que abrange o *host* e, quando presentes, os subdomínios, caminho, cadeia de consulta e fragmento. Essa decomposição é metodologicamente relevante porque as estratégias de ocultação empregadas pelos atacantes atuam sobre partes diferentes. Alongar artificialmente o caminho, usar subdomínios extensos ou irregulares, inserir caracteres codificados ou ofuscados, misturar letras e dígitos em um mesmo *token* e construir cadeias de consulta com múltiplos parâmetros são manipulações que se manifestam em componentes específicos da URL e que, por isso, podem ser descritas por atributos distintos (Kustiawan; Ghauth, 2025a; Kailas; Roopalakshmi, 2025; Rafsanjani et al., 2024; Tian et al., 2025).

Há ainda uma classe de manipulações que atua sobre a percepção visual do endereço, e não sobre sua estrutura. Caracteres de largura zero, ataques homográficos, codificação *punycode*, homófonos, *bit squatting* e *combosquatting* produzem cadeias que o usuário lê como legítimas, embora sejam distintas do domínio original (Prasad; Chandra, 2024). É essa categoria que justifica a inclusão, entre os atributos tabulares, de descritores de similaridade entre a URL observada e padrões de legitimidade.

Analisar exclusivamente a URL traz uma vantagem operacional relevante: a decisão pode ser tomada sem que o mecanismo de detecção visite a página suspeita, o que atende a cenários de detecção precoce

e de baixa latência (Duarte et al., 2025; Tian et al., 2025). Em contrapartida, a URL isolada não revela os sinais presentes no HTML, no JavaScript, nos formulários, nos hiperlinks ou na apresentação visual da página. A revisão de Kulkarni, Balachandran e Das (2025) indica que ampliar as fontes de evidência tende a favorecer a capacidade de detecção, mas acrescenta dependências, custo de processamento, requisitos de armazenamento e tempo de resposta. A escolha entre abordagens baseadas em URL, em conteúdo, em evidências visuais ou em combinações híbridas deve, portanto, considerar as condições do cenário de aplicação.

2.2.1 Delimitação Terminológica

Neste trabalho, a expressão *detecção de phishing baseada em URL* é empregada em sentido amplo. Ela abrange tanto modelos que recebem diretamente a sequência bruta da URL quanto modelos tabulares que utilizam atributos derivados da URL e, quando o conjunto de dados os disponibiliza, características extraídas do HTML da página correspondente. A delimitação é necessária porque a literatura emprega o mesmo rótulo para sistemas com espaços de entrada bastante diferentes (Asiri et al., 2023; Kulkarni; Balachandran; Das, 2025; Tian et al., 2025; Zieni; Massari; Calzarossa, 2023), o que compromete a comparabilidade entre resultados sempre que a delimitação não é explicitada.

2.3 MODALIDADES DE EVIDÊNCIA E TAXONOMIA DAS ABORDAGENS DE DETECÇÃO

A qualidade de um detector não depende apenas do algoritmo de classificação, mas também da forma como o objeto é representado. Duas pesquisas podem empregar o mesmo classificador e obter comportamentos distintos por utilizarem fontes de evidência, processos de coleta e transformações diferentes. Por esse motivo, a representação deve ser tratada como parte integrante do método, e não como etapa meramente preparatória (Kustiawan; Ghauth, 2025a; Skula; Kvet, 2024).

As obras de revisão da área convergem para uma organização das abordagens em famílias com características e limitações próprias. Zieni, Massari e Calzarossa (2023) estruturam a discussão em torno de três categorias, métodos baseados em listas, métodos baseados em similaridade e métodos baseados em aprendizado de máquina, descrevendo, para cada uma, os conjuntos de dados empregados na avaliação e as lacunas remanescentes. Kulkarni, Balachandran e Das (2025) propõem categorização em quatro grupos: abordagens baseadas em URL, em conteúdo da página, em serviços de terceiros e em análise visual. Asiri et al. (2023) concentram-se na combinação de atributos de HTML e de URL, revisando modelos de aprendizado profundo aplicados à detecção baseada em URL e à detecção híbrida e comparando-os quanto ao pré-processamento, à extração de atributos, ao projeto do modelo e ao desempenho. Kailas e Roopalakshmi (2025) categorizam as técnicas em listagem, heurísticas, aprendizado de máquina, engenharia de características e métodos emergentes.

A revisão mais recente propõe uma reorganização desse campo que foi adotada como referência estrutural neste capítulo. Tian et al. (2025) identificam quatro limitações nas revisões anteriores: a dependência de taxonomias centradas no algoritmo, que obscurece a compreensão de como cada abordagem explora

canais específicos de informação; a ausência de tratamento das defesas baseadas em *Transformers* e em modelos de linguagem de grande porte; a não compilação de implementações de código aberto que viabilizem *benchmarking*; e a cobertura insuficiente de conjuntos de dados. Em resposta, os autores propõem uma taxonomia baseada em modalidade, que categoriza os trabalhos segundo suas modalidades primárias de dados, cadeias léxicas de URL, estrutura HTML, comportamento JavaScript e disposição visual, agrupando, por exemplo, modelos que analisam árvores do modelo de objeto de documento (*Document Object Model*, DOM) e extraem caminhos de *tags* HTML na modalidade HTML e modelos que utilizam capturas de tela renderizadas na modalidade visual. Os autores compilam ainda os conjuntos de dados publicamente disponíveis entre 2016 e 2024 e as implementações de código aberto correspondentes, com o propósito declarado de reduzir a ausência de comparações com linhas de base adequadas.

Essa taxonomia por modalidade foi adotada aqui porque explicita a relação entre o canal de informação de entrada e o desenho do modelo, que é precisamente o eixo da investigação sobre complementaridade conduzida nesta dissertação. A Tabela 2.1 sintetiza as modalidades e suas implicações operacionais.

Tabela 2.1 – Modalidades de evidência empregadas na detecção de *phishing*

Modalidade	Exemplos de informação	Implicações para o detector
Sequência bruta da URL	Caracteres, <i>tokens</i> , segmentos, ordem e contexto local	Permite aprendizado de representações sem depender integralmente de atributos manuais; exige tratamento de sequências raras, comprimentos variáveis e vocabulário aberto.
Características léxicas e estruturais	Comprimento, quantidade e distribuição de caracteres, símbolos, dígitos, subdomínios, caminhos e parâmetros	Apresentam baixo custo de extração e boa adequação a modelos tabulares, mas podem refletir particularidades do processo de coleta.
Domínio, hospedagem e reputação	Idade, resolução, registros, endereço do protocolo de Internet (<i>Internet Protocol</i> , IP), certificado, popularidade e indicadores externos	Acrescentam contexto, porém podem depender de consultas externas, sofrer indisponibilidade ou introduzir variação temporal.
HTML e JavaScript	Formulários, <i>scripts</i> , elementos ocultos, redirecionamentos, conteúdo textual e estrutura do documento	Fornecem evidência sobre o comportamento da página, mas requerem acesso ao conteúdo e tratamento de páginas indisponíveis ou alteradas.
Hiperlinks e consistência entre fontes	Relação entre URL, domínio, links internos, texto, título e destinos de formulários	Permite identificar incoerências entre a identidade apresentada e a infraestrutura efetivamente utilizada.
Sinais visuais e multimodais	Imagem renderizada, logotipos, composição visual, texto, URL, HTML e metadados	Podem reconhecer imitação de marcas e combinar evidências complementares, com maior custo de processamento e integração.

Fonte: adaptado de Tian et al. (2025), Asiri et al. (2023), Zieni, Massari e Calzarossa (2023) e Kulkarni, Balachandran e Das (2025).

Os métodos baseados em listas de bloqueio e de permissão apresentam baixo custo computacional e alta precisão sobre entradas já catalogadas, mas são estruturalmente incapazes de cobrir campanhas recém-criadas (Le et al., 2018; Pillai et al., 2024; Zieni; Massari; Calzarossa, 2023). Os métodos baseados em similaridade e em heurísticas ampliam a cobertura, porém dependem de regras cuja manutenção se torna custosa diante da evolução das táticas de ataque (Kailas; Roopalakshmi, 2025; Zieni; Massari; Calzarossa, 2023). Essas limitações explicam o predomínio das abordagens baseadas em aprendizado de máquina nas duas últimas décadas.

Esta pesquisa opera sobre as duas primeiras modalidades da Tabela 2.1 e também sobre a quarta e a quinta. Ficaram deliberadamente fora do escopo a dependência de serviços de terceiros em tempo de inferência e a análise visual renderizada, conforme detalhado na Seção 2.10.

2.4 ENGENHARIA E SELEÇÃO DE CARACTERÍSTICAS

2.4.1 Características Léxicas e Estruturais da URL

Características léxicas descrevem propriedades observáveis na sequência da URL, como comprimento total, proporção de dígitos, quantidade de caracteres especiais, repetição de símbolos, distribuição de *tokens* e presença de termos associados a autenticação, pagamento ou segurança. Características estruturais representam a forma como esses elementos se organizam e incluem a composição do domínio, a profundidade de subdomínios, o caminho e os parâmetros de consulta (Aljofey et al., 2022; Kustiawan; Ghauth, 2025a; Rafsanjani et al., 2024).

Três fatores explicam a atratividade dessas características. O primeiro é que sua extração é rápida e independe do carregamento integral da página. O segundo é que os valores produzidos são compatíveis com classificadores tabulares maduros. O terceiro é que muitas estratégias de ocultação alteram diretamente a morfologia da URL, gerando sinais discriminativos (Kustiawan; Ghauth, 2025a; Kailas; Roopalakshmi, 2025; Tian et al., 2025). Estudos comparativos que avaliam URL, HTML e atributos derivados mostram que variáveis de comprimento, composição, subdomínio e complexidade assumem elevada importância relativa em diferentes classificadores (Aljofey et al., 2022; Kustiawan; Ghauth, 2025a; Kustiawan; Ghauth, 2025b).

A construção de características derivadas, isto é, de atributos compostos a partir de informações já disponíveis, exige cautela. Um atributo pode ser altamente discriminativo por representar um padrão genuinamente associado ao ataque, mas também pode funcionar como marcador indireto da fonte de coleta, do período temporal ou do processo de rotulagem (Kustiawan; Ghauth, 2025b; Mohanty et al., 2024; Skula; Kvet, 2024). Quando isso ocorre, o modelo aprende a distinguir a composição do conjunto de dados, e não o fenômeno que se pretende detectar. Distinguir capacidade discriminativa de capacidade de generalização é, portanto, condição para interpretar resultados muito elevados, conceito retomado na Seção 2.8.2.

2.4.2 Características de Domínio, Hospedagem e Similaridade

Informações de domínio e de hospedagem acrescentam contexto que não está contido na sequência textual. Entre os exemplos discutidos na literatura estão a idade e o registro do domínio, a associação com endereços IP, os certificados, a popularidade, o histórico e a similaridade com domínios legítimos (Asiri et al., 2023; Kailas; Roopalakshmi, 2025; Tian et al., 2025; Zieni; Massari; Calzarossa, 2023). Esses atributos são úteis porque campanhas de *phishing* recorrem com frequência a infraestrutura efêmera, a domínios recém-criados ou a nomes que procuram reproduzir marcas conhecidas (Duarte et al., 2025; Prasad; Chandra, 2024).

Há, contudo, um custo associado. Indicadores obtidos de serviços externos podem se alterar ao longo

do tempo, apresentar cobertura desigual entre regiões e registradores, impor latência à decisão ou simplesmente tornar-se indisponíveis (Kulkarni; Balachandran; Das, 2025; Kustiawan; Ghauth, 2025b; Rafsanjani et al., 2024). Essa restrição motivou, na literatura recente, a preferência por descritores de similaridade calculados diretamente sobre a cadeia da URL, sem consulta externa (Prasad; Chandra, 2024). A mesma estratégia foi adotada nesta pesquisa e é detalhada no Capítulo 4.

2.4.3 Características de HTML, Conteúdo e Hiperlinks

Analisar a página amplia a evidência para além do endereço. O HTML pode revelar formulários de coleta de credenciais, campos ocultos, redirecionamentos, *scripts*, ligações externas, inconsistências entre o domínio exibido e os destinos dos hiperlinks, além de conteúdo textual relacionado à marca imitada (Aljofey et al., 2022; Kustiawan; Ghauth, 2025a; Asiri et al., 2023).

Dessa modalidade decorrem três consequências metodológicas. A primeira é que acrescentar blocos de características não produz, por si só, um modelo melhor: blocos adicionais ampliam a cobertura de evidências, mas também introduzem redundância, custo de extração e sinais específicos da base utilizada (Kustiawan; Ghauth, 2025a; Kustiawan; Ghauth, 2025b). A segunda é que a extração de HTML exige que a página esteja acessível no momento da coleta, o que introduz o viés de disponibilidade discutido na Seção 2.7.3. A terceira é de natureza operacional e costuma ser subestimada: avaliações de tempo de execução ponta a ponta indicam que o *download* do HTML consome aproximadamente 1.000 ms em média e responde por mais de 95% do atraso total do processo de detecção (Jiang et al., 2025). Esse dado fundamenta a decisão, adotada nesta pesquisa, de extrair os atributos de HTML *offline*, a partir de instantâneos armazenados, separando o custo de coleta do custo de inferência do modelo.

2.4.4 Representações Textuais e Recuperação da Informação

URLs, textos de páginas e mensagens de correio eletrônico podem ser representados por técnicas oriundas da recuperação da informação e do processamento de linguagem natural (PLN). Modelos de espaço vetorial transformam documentos ou sequências em vetores cujas dimensões representam termos, caracteres ou componentes latentes. Manning, Raghavan e Schütze (2008) sistematizam os fundamentos dessas representações, entre as quais o modelo *Bag-of-Words*, a ponderação por frequência do termo e frequência inversa nos documentos (*Term Frequency–Inverse Document Frequency*, TF-IDF) e a análise semântica latente, que constituem a base conceitual das técnicas empregadas na representação léxica de URLs e de conteúdo textual em detecção de *phishing* (Aslam et al., 2024; Gualberto et al., 2020b; Gualberto et al., 2020a; Khalid et al., 2024).

A relevância dessas técnicas para esta pesquisa não é apenas histórica. Representações esparsas de n-gramas de caractere permanecem competitivas na detecção de URLs e apresentam padrões de erro distintos dos produzidos por modelos tabulares sobre atributos projetados, o que as torna candidatas naturais a compor um comitê heterogêneo, conforme discutido na Seção 2.6.

2.4.5 Seleção de Características e Redução de Dimensionalidade

A seleção de características procura remover atributos irrelevantes, redundantes ou ruidosos, preservando um subconjunto informativo. Métodos de filtro utilizam critérios estatísticos independentes do classificador; métodos *wrapper* avaliam subconjuntos pelo desempenho de um modelo; métodos embutidos realizam a seleção durante o próprio treinamento (Bari et al., 2025; Kalabarige et al., 2023; Mohanty et al., 2024; Nayak; Muniyal; Belavagi, 2025). Métodos híbridos combinam essas famílias justamente porque as abordagens individuais esbarram em limitações de consumo de recursos, de tempo de processamento e de tratamento de conjuntos de alta dimensionalidade (Mohanty et al., 2024).

A literatura converge na constatação de que a dimensão do vetor de entrada não é, por si só, medida de qualidade: um espaço menor pode ser mais eficiente e menos redundante, desde que preserve a informação discriminativa necessária (Duarte et al., 2025; Gualberto et al., 2020a; Indrasiri; Halgamuge; Mohammad, 2021; Mohanty et al., 2024; Nayak; Muniyal; Belavagi, 2025; Rafsanjani et al., 2024). Há, entretanto, um requisito de validade frequentemente negligenciado: quando a seleção é ajustada sobre a totalidade dos dados, incluindo a partição de teste, o desempenho reportado incorpora vazamento de informação. Nesta pesquisa, todo o pré-processamento e toda a seleção foram ajustados exclusivamente sobre a partição de treinamento, conforme detalhado no Capítulo 4.

2.5 PARADIGMAS DE APRENDIZADO APLICADOS À DETECÇÃO

2.5.1 Aprendizado Supervisionado e Modelos Baseados em Árvores

O aprendizado de máquina supervisionado formula a detecção como um problema de classificação binária: cada amostra é associada a um vetor de atributos e a um rótulo que indica *phishing* ou legitimidade. Durante o treinamento, o modelo estima uma função capaz de relacionar o espaço de entrada à classe; na inferência, essa função é aplicada a amostras não utilizadas no ajuste (Karim et al., 2023; Zieni; Massari; Calzarossa, 2023).

Modelos baseados em árvores são amplamente utilizados na área porque representam interações não lineares entre características heterogêneas e exigem menos transformações de escala do que muitos modelos lineares. Chen e Guestrin (2016) formalizam o XGBoost como um sistema escalável de *gradient tree boosting*, incorporando regularização, tratamento eficiente de esparsidade e otimizações na construção das árvores. Esse algoritmo e suas variantes de *boosting* figuram recorrentemente entre os classificadores tabulares de melhor desempenho na detecção de *phishing* (Kustiawan; Ghauth, 2025a; Duarte et al., 2025; Gualberto et al., 2020b; Karim et al., 2023; Kustiawan; Ghauth, 2025b; Mohanty et al., 2024; Nayak; Muniyal; Belavagi, 2025).

Dessa constatação decorre uma exigência metodológica central para esta dissertação: arquiteturas mais complexas devem ser comparadas com linhas de base tabulares fortes, e não com classificadores fracos. Se a representação tabular já torna as classes quase separáveis, um modelo profundo ou um comitê pode apresentar ganho mínimo ainda que aumente substancialmente o custo computacional. Essa situação é analisada nesta pesquisa pelo conceito operacional de saturação do *benchmark*, definido na Seção 2.8.2.

2.5.2 Aprendizado Profundo e Aprendizado de Representações

O aprendizado profundo utiliza redes neurais com múltiplas camadas para aprender representações hierárquicas dos dados. LeCun, Bengio e Hinton (2015) descrevem como modelos compostos por múltiplas camadas de processamento aprendem representações em diferentes níveis de abstração por meio do algoritmo de retropropagação, que indica como os parâmetros internos devem ser ajustados para calcular a representação de cada camada a partir da anterior. Os autores destacam o desempenho de redes neurais convolucionais (*Convolutional Neural Network*, CNN) no processamento de sinais estruturados e de redes neurais recorrentes (*Recurrent Neural Network*, RNN) no tratamento de dados sequenciais, propriedades diretamente aplicáveis à modelagem de URLs.

Na detecção de URLs maliciosas, o marco dessa transição foi estabelecido por Le et al. (2018), que identificam três limitações das abordagens léxicas tradicionais baseadas em atributos do tipo *Bag-of-Words* seguidos de classificadores lineares ou de margem: a incapacidade de capturar significado semântico e padrões sequenciais na cadeia da URL, a exigência de engenharia manual substancial de atributos e a incapacidade de generalizar para atributos não observados no treinamento. A resposta proposta consiste em aplicar redes convolucionais simultaneamente aos caracteres e às palavras da URL, aprendendo o *embedding* de forma conjuntamente otimizada, além de empregar *embeddings* avançados para lidar com a alta incidência de *tokens* raros. Esse trabalho fundamenta o ramo léxico convolucional adotado no arcabouço desta pesquisa.

O benefício do aprendizado de representações não elimina decisões de projeto. Permanece necessário definir tokenização, comprimento máximo, tratamento de caracteres raros, arquitetura, função de perda, regularização e estratégia de validação. Além disso, o aprendizado da representação continua dependente da distribuição observada: uma rede profunda pode memorizar regularidades específicas do conjunto de treinamento se a avaliação não preservar separação adequada entre treinamento, validação e teste (Skula; Kvet, 2024; Wei; Nakayama; Sekiya, 2025). Finalmente, modelos profundos demandam custos superiores de treinamento e de inferência, e seus benefícios tornam-se difíceis de mensurar quando atributos tabulares projetados já sintetizam grande parte da separação entre classes (Kustiawan; Ghauth, 2025a; Kustiawan; Ghauth, 2025b).

2.5.3 Transformers e Representações Contextuais

A arquitetura *Transformer* substitui a dependência exclusiva de recorrência e de convolução por mecanismos de atenção, capazes de relacionar diferentes posições de uma sequência e de processá-las de forma paralelizável. Vaswani et al. (2017) propõem uma arquitetura fundamentada exclusivamente em mecanismos de atenção, dispensando recorrência e convoluções, e demonstram ganhos simultâneos de qualidade, de paralelização e de custo de treinamento em tarefas de transdução de sequências.

Sobre essa base, Devlin et al. (2018) introduzem o BERT (*Bidirectional Encoder Representations from Transformers*), modelo de representação linguística pré-treinado de forma bidirecional profunda, no qual a representação de cada *token* é condicionada simultaneamente pelos contextos à esquerda e à direita em todas as camadas. Após o pré-treinamento, o modelo pode ser ajustado para tarefas específicas com alterações arquiteturais limitadas. Embora concebido para linguagem natural, sua capacidade de construir

representações contextuais motivou adaptações para URLs, que apresentam vocabulário aberto, *tokens* incomuns e estrutura distinta da de frases convencionais (Aljofey et al., 2025; Almousa; Anwar, 2023; He et al., 2024b; Remya et al., 2025; Wei; Nakayama; Sekiya, 2025).

O emprego de representações contextuais introduz um custo computacional que precisa ser explicitado. Modelos *Transformer* completos apresentam número elevado de parâmetros e latência de inferência superior à de modelos tabulares, o que motivou a investigação de variantes compactas na literatura de detecção (Aljofey et al., 2025; He et al., 2024b). A magnitude desse custo é mensurável: arquiteturas convolucionais aplicadas à mesma tarefa requerem aproximadamente um terço do tempo de treinamento exigido por métodos equivalentes baseados em BERT (He et al., 2024a). Também nesse caso, o maior número de parâmetros não garante ganho quando a tarefa já é resolvida por sinais tabulares altamente discriminativos, o que reforça a exigência de comparação sob protocolo idêntico enunciada na Seção 2.5.1.

2.6 APRENDIZADO POR COMITÊS E GENERALIZAÇÃO EMPILHADA

2.6.1 Princípios de Combinação de Modelos

Métodos de comitê combinam múltiplos aprendizes com o objetivo de produzir uma decisão mais estável ou mais precisa do que a obtida por um único modelo. A combinação pode ocorrer por votação, por média de probabilidades, por ponderação ou pelo treinamento de um modelo adicional (Indrasiri; Halgamuge; Mohammad, 2021; Karim et al., 2023; Naseeb et al., 2025). O ganho potencial decorre da diversidade: para que o comitê corrija erros, seus componentes precisam apresentar competências ou padrões de falha parcialmente diferentes, de modo que os aprendizes de base compensem mutuamente os erros cometidos na classificação (He et al., 2024b).

A generalização empilhada, ou *stacking*, organiza os modelos em níveis. No Nível 0, os aprendizes de base produzem previsões ou probabilidades. No Nível 1, um meta-aprendiz recebe essas saídas e aprende a combiná-las (He et al., 2024b; Indrasiri; Halgamuge; Mohammad, 2021; Kalabarige et al., 2022; Motiur Rahman; Islam; Jabiullah, 2020). O meta-aprendiz pode ser linear, o que favorece a interpretação e a regularização, ou não linear, o que permite representar interações mais complexas entre as probabilidades (Aslam et al., 2024; Kalabarige et al., 2023; Khalid et al., 2024). A formulação original do procedimento é devida a Wolpert (1992), que o descreve como esquema para reduzir a taxa de erro de generalização de um ou mais generalizadores a partir da dedução de seus vieses em relação a um conjunto de aprendizado.

Incluir muitos algoritmos, porém, não garante diversidade útil. Modelos diferentes treinados sobre o mesmo espaço altamente separável podem produzir probabilidades quase idênticas, e modelos de uma mesma família tendem a errar as mesmas amostras (Bari et al., 2025; Kalabarige et al., 2022). Daí decorre um princípio que orientou o desenho desta pesquisa: a diversidade entre modelos não deve ser inferida pelos nomes dos algoritmos, mas examinada em conjunto com a modalidade de evidência e a representação que alimentam cada aprendiz. É essa consideração que justifica organizar os preditores de Nível 0 em ramos específicos por modalidade, em vez de reunir uma lista de classificadores distintos sobre o mesmo espaço de atributos.

2.6.2 Predições Fora da Dobra e Prevenção de Vazamento

O treinamento do meta-aprendiz exige cuidados para evitar vazamento de informação. Se as predições utilizadas no Nível 1 forem geradas por modelos já ajustados sobre as mesmas amostras, elas serão excessivamente otimistas, e o meta-aprendiz aprenderá a combinar um cenário que não corresponde ao comportamento esperado em dados não vistos (Kalabarige et al., 2022; Skula; Kvet, 2024).

A estratégia apropriada consiste em gerar predições fora da dobra (*out-of-fold*, OOF), procedimento no qual cada amostra de treinamento recebe uma probabilidade produzida por um modelo que não a utilizou no ajuste, e cujo detalhamento formal é apresentado na Seção 4.5.3 (Aslam et al., 2024; Kalabarige et al., 2023; Motiur Rahman; Islam; Jabiullah, 2020).

A importância desse procedimento não está apenas na técnica de validação, mas na validade da inferência. Resultados de empilhamento só podem ser atribuídos à combinação dos modelos quando o Nível 1 não teve acesso indireto aos rótulos por meio de predições produzidas sobre amostras já observadas. Nesta dissertação, a geração OOF é tratada como requisito central do protocolo experimental, e não como detalhe de implementação.

2.7 CONJUNTOS DE DADOS E BENCHMARKS DE PHISHING

2.7.1 Fontes, Rotulagem e Composição

Conjuntos de dados de *phishing* são formados a partir de repositórios colaborativos, *feeds* de incidentes, listas de popularidade, registros de certificados, coleta Web e bases acadêmicas. O projeto Phishing.Database mantém um repositório comunitário de URLs reportadas como maliciosas (Phishing Database Project, 2026), e a plataforma PhishTank utiliza submissão e verificação colaborativa de páginas suspeitas (Cisco Talos Intelligence Group, 2026). Essas fontes são úteis para obter amostras reais, mas exigem deduplicação, validação temporal e tratamento de indisponibilidade.

A definição da classe legítima é tão determinante quanto a coleta das amostras de *phishing*. Se as URLs legítimas forem sistematicamente mais curtas, mais populares ou mais estáveis, enquanto as maliciosas provierem de fontes efêmeras e estruturalmente complexas, o classificador poderá explorar diferenças triviais de origem em vez de padrões de ataque. Skula e Kvet (2024) propõem um arcabouço para a preparação de conjuntos equilibrados e abrangentes, cobrindo a identificação e a coleta de dados de *phishing* e legítimos, a limpeza e considerações estruturais como a proporção entre as classes e o tamanho adequado do conjunto. Os autores argumentam que a comparação entre estudos é fragilizada pela heterogeneidade de fontes, transformações e técnicas de limpeza, e apontam a transparência do processo de construção como condição para a comparabilidade dos resultados.

2.7.2 Benchmarks Adotados nesta Pesquisa

Dois conjuntos de referência com atributos previamente extraídos foram adotados nesta dissertação, além do conjunto construído no escopo da pesquisa.

Prasad e Chandra (2024) propõem o PhiUSIIL, arcabouço de detecção de URLs de *phishing* baseado em índice de similaridade e aprendizado incremental. O PhiUSIIL extrai características da URL, faz o *download* da página para extrair características de HTML e deriva novas características a partir das informações existentes; o conjunto que dele resulta é adotado aqui como D1. Prasad et al. (2026) propõem o StealthPhisher, arcabouço defensivo que combina aprendizado profundo híbrido e inteligência artificial generativa; o conjunto associado é enriquecido com atributos como entropia de Shannon, estrutura da URL, complexidade de Kolmogorov e dados derivados do HTML, e é adotado como D2. O terceiro conjunto, HybridPhisher, adotado como D3, foi construído no escopo desta pesquisa a partir das fontes verificadas mencionadas na Seção 2.7.1 (Cisco Talos Intelligence Group, 2026; Phishing Database Project, 2026) e reúne, para as mesmas instâncias, a cadeia textual da URL e atributos derivados de URL e de HTML. Seu procedimento completo de coleta, filtragem e consolidação é apresentado no Capítulo 4, e a composição numérica dos três conjuntos é apresentada na Seção 4.6.2.

2.7.3 Ameaças à Validade Relacionadas aos Dados

Quatro ameaças à validade merecem destaque nesta área.

A primeira é o vazamento por duplicação ou por alta similaridade entre as partições de treinamento e de teste, que produz estimativas otimistas de desempenho (Skula; Kvet, 2024). A segunda é a deriva temporal, pois padrões de infraestrutura, de nomenclatura e de campanha se modificam ao longo do tempo, de modo que resultados obtidos sobre um recorte temporal podem não se sustentar em outro (Kailas; Roopalakshmi, 2025; Tian et al., 2025; Wei; Nakayama; Sekiya, 2025). A terceira é a indisponibilidade seletiva de páginas: quando a extração de HTML exige que o recurso esteja ativo, amostras de *phishing* expiradas, removidas ou bloqueadas são sistematicamente excluídas, o que afeta as classes de forma desigual. A quarta é o viés de fonte, situação em que o rótulo se correlaciona com a origem da amostra, e não com o fenômeno de interesse (Mohanty et al., 2024; Skula; Kvet, 2024).

O desbalanceamento entre classes constitui ameaça correlata, pois leva os modelos a privilegiar os padrões da classe majoritária durante o treinamento, reduzindo sua capacidade discriminativa sobre a classe minoritária (Jiang et al., 2025; Naseeb et al., 2025). Nenhuma dessas ameaças é eliminada pelo simples aumento do número de registros, já que bases maiores podem reproduzir os mesmos vieses em escala ampliada. Documentar o processo de construção, deduplicar, separar adequadamente as partições e analisar o desempenho por blocos de características são, portanto, componentes necessários da avaliação (Kailas; Roopalakshmi, 2025; Kulkarni; Balachandran; Das, 2025; Skula; Kvet, 2024; Tian et al., 2025).

2.8 CRITÉRIOS DE AVALIAÇÃO EXPERIMENTAL

2.8.1 Métricas de Classificação e Interpretação Operacional

A acurácia representa a proporção total de classificações corretas, mas pode ocultar desempenho inadequado quando as classes estão desbalanceadas. A precisão mede a proporção de amostras classificadas como *phishing* que efetivamente pertencem à classe positiva, e o *recall* mede a proporção de amostras

de *phishing* corretamente identificadas. A medida F1 combina precisão e *recall* por média harmônica, sendo adequada a situações de dados não balanceados (Tian et al., 2025). As áreas sob as curvas ROC e de precisão-*recall* avaliam o comportamento das probabilidades em diferentes limiares, sendo a segunda particularmente informativa quando a classe positiva é minoritária ou quando falsos positivos e falsos negativos têm custos distintos (Bari et al., 2025; Kalabarige et al., 2023; Zieni; Massari; Calzarossa, 2023).

A taxa de falsos positivos (*False Positive Rate*, FPR) merece atenção específica na detecção de *phishing*. Um detector com taxa elevada bloqueia recursos legítimos e tende a ser desativado pelos usuários, o que anula sua função protetiva. Por isso, parte da literatura reporta explicitamente essa métrica ao lado da acurácia (Aljofey et al., 2022; Bari et al., 2025; Remya et al., 2025), e há inclusive propostas de métricas compostas construídas sobre falsos positivos e falsos negativos como forma de superar as limitações da acurácia isolada (Nayak; Muniyal; Belavagi, 2025).

Estudos da área frequentemente reportam valores elevados dessas métricas, mas a comparação direta entre trabalhos exige cautela. Diferenças de fonte, limpeza, divisão dos dados, balanceamento, extração de características e seleção de limiar produzem resultados não equivalentes (Asiri et al., 2023; Kailas; Roopalakshmi, 2025; Skula; Kvet, 2024; Zieni; Massari; Calzarossa, 2023). A própria literatura de revisão reconhece essa dificuldade e registra a ausência de *benchmarking* padronizado, bem como a frequente omissão de comparações adequadas com linhas de base (Tian et al., 2025). Por esse motivo, esta dissertação privilegia comparações internas sob protocolo idêntico, nas quais todos os modelos recebem partições compatíveis e são avaliados pelas mesmas métricas.

2.8.2 Saturação do Benchmark e Erro Residual

Neste trabalho, utiliza-se a expressão ‘saturação empírica’ em sentido operacional e restrito ao protocolo adotado. Considera-se haver evidência de saturação quando modelos independentes e suficientemente fortes atingem desempenho próximo ao teto e a camada de meta-aprendizado não produz ganho pareado de F1 igual ou superior a 0,001 e, como não saturado, quando o ganho alcança esse valor com intervalo de confiança estritamente positivo. A saturação não significa que o problema real esteja resolvido; significa que a combinação entre aquele conjunto de dados, aquele espaço de características e aquela divisão experimental oferece pouca margem para distinguir arquiteturas.

Define-se ainda como separabilidade específica do conjunto de dados a situação em que um subconjunto de atributos distingue as classes com erro muito baixo dentro de determinado *benchmark*, sem que isso constitua evidência suficiente de generalização para outras distribuições. O conceito não implica que o conjunto esteja incorreto; indica que os ganhos atribuídos a arquiteturas adicionais devem ser interpretados em função do erro residual disponível e da origem da separabilidade (Kustiawan; Ghauth, 2025a; Kustiawan; Ghauth, 2025b; Skula; Kvet, 2024).

Esses conceitos são decisivos para avaliar o empilhamento. O meta-aprendiz só pode produzir ganho se os modelos de base errarem amostras diferentes e se suas probabilidades contiverem informação útil para correção. Quando todos os componentes classificam praticamente as mesmas amostras de forma correta, a complexidade adicional tende a gerar ganhos marginais ou estatisticamente instáveis. Em sentido oposto, quando o *benchmark* preserva erros residuais e as famílias de modelos capturam evidências

complementares, a fusão pode ser justificável.

A interpretação por erro residual desloca a pergunta de pesquisa. Em vez de perguntar qual modelo alcança o maior valor absoluto, pergunta-se sob quais condições a combinação melhora de forma consistente uma linha de base forte. Essa formulação orientou as comparações pareadas, os intervalos de confiança e a análise de complementaridade desenvolvidos nos Capítulos 4 e 5.

2.8.3 Repetibilidade, Significância Estatística e Eficiência

Uma única execução pode refletir a inicialização aleatória, a divisão dos dados ou a ordem de treinamento. Avaliações repetidas permitem estimar a estabilidade dos resultados e comparar modelos sob sementes correspondentes. Comparações pareadas são especialmente adequadas quando dois métodos são executados sobre as mesmas partições, pois a diferença entre eles elimina parte da variabilidade associada ao conjunto de teste. Esse controle torna-se indispensável quando o ganho esperado é da mesma ordem de grandeza do erro residual remanescente, situação característica dos *benchmarks* discutidos na Seção 2.8.2.

Além do desempenho preditivo, a viabilidade de implantação depende do tempo de treinamento, da latência de inferência, do consumo de memória, do tamanho do artefato e das dependências externas. As revisões recentes destacam que sistemas reais precisam equilibrar acurácia, velocidade, escalabilidade e adaptabilidade (Kailas; Roopalakshmi, 2025; Kulkarni; Balachandran; Das, 2025; Tian et al., 2025), e trabalhos com preocupação operacional explícita reportam medidas concretas dessa dimensão, como o tempo necessário para classificar lotes de amostras em produção (He et al., 2024a; Jiang et al., 2025; Rastakhiz; Eftekhari; Vahdati, 2024). Um ganho pequeno pode não justificar um *pipeline* substancialmente mais caro, especialmente quando a decisão deve ocorrer antes do acesso à página.

2.8.4 Calibração de Probabilidades

Modelos de comitê e classificadores de margem produzem estimativas de probabilidade que apresentam desvios sistemáticos em relação às frequências observadas. Quando a decisão final depende da comparação dessas estimativas com um limiar, tais desvios afetam diretamente o ponto de operação escolhido. Niculescu-Mizil e Caruana (2005) examinam esse comportamento em diferentes famílias de aprendizes e demonstram que métodos não paramétricos, entre os quais a regressão isotônica, figuram entre os mais efetivos para corrigi-lo quando há dados suficientes. Esse fundamento sustenta a etapa de calibração aplicada às probabilidades de meta-nível nesta pesquisa, descrita no Capítulo 4.

2.9 CONFIABILIDADE E ROBUSTEZ ADVERSARIAL

Detectores baseados em aprendizado de máquina operam em ambiente adversarial, no qual o atacante pode modificar a entrada com o propósito de reduzir a probabilidade de detecção. No caso de URLs, as alterações envolvem inserção, remoção ou substituição de caracteres, além de mudanças em *tokens*, subdomínios, caminhos e parâmetros. Em páginas Web, podem ser modificados elementos HTML, estilos,

conteúdo textual, recursos visuais ou relações entre componentes (Kulkarni et al., 2025; Pillai et al., 2024; Sabir et al., 2024).

Sabir et al. (2024) apresentam o estudo mais abrangente sobre a confiabilidade dessas soluções. Os autores propõem uma metodologia para investigar cinquenta modelos representativos de detecção de URLs de *phishing* baseados em aprendizado de máquina, desenvolvem o gerador de URLs adversariais URLBUG, reproduzem os cinquenta sistemas, registram seu desempenho de referência e os submetem ao conjunto adversarial gerado. Os resultados mostram que as URLs adversariais produzidas possuem sintaxe válida e podem ser registradas a preço anual mediano de US\$ 11,99, e que, entre os 13% de URLs adversariais já registradas, 63,94% haviam sido utilizadas para fins maliciosos. O achado mais expressivo é a queda do coeficiente de correlação de Matthews de mediana 0,92 para 0,02 quando os modelos são avaliados sobre o conjunto adversarial, o que indica que as linhas de base consideradas são pouco confiáveis em sua forma corrente.

Lo Re, Morana e Rizzo (2026) quantificam o mesmo fenômeno no espaço do problema. Os autores propõem uma estratégia de ataque que emprega algoritmos de busca heurística, *Beam Search*, *Simulated Annealing* e *Monte Carlo Tree Search*, para gerar amostras adversariais que evadem detectores do estado da arte preservando fidelidade visual e funcional, otimizando o compromisso entre o número de manipulações e o sucesso do ataque. Em dois conjuntos públicos, o método reduz a taxa média de detecção de 0,80 para 0,05 no conjunto Zenodo e de 0,82 para 0,03 no conjunto δ Phish, exigindo até 70% menos manipulações do que ataques concorrentes e mantendo as amostras mais próximas das originais nas métricas L_0 e L_2 , o que indica plausibilidade estatística.

Outros estudos detalham esse diagnóstico em recortes distintos. Nowroozi et al. (2023) conduzem análise de ataques adversariais sobre arcabouço de detecção de URLs maliciosas em anúncios publicitários. Pillai et al. (2024) tratam de ataques de evasão e de mecanismos de defesa para classificadores baseados em aprendizado de máquina, elaborando amostras adversariais direcionadas aos atributos de classificação com a preocupação explícita de preservar a funcionalidade e a aparência das páginas, avaliada por métricas de distorção de imagem, e propondo mecanismo de detecção por semelhança. Kulkarni et al. (2025) desenvolvem o PhishOracle, ferramenta que gera páginas de *phishing* adversariais pela incorporação de características diversas de *phishing* em páginas legítimas; ao avaliar a robustez de três modelos específicos da tarefa, *Stack model*, VisualPhishNet e Phishpedia, os autores observam queda significativa nas taxas de detecção, enquanto um detector baseado em modelo de linguagem de grande porte multimodal demonstra maior robustez, permanecendo, ainda assim, suscetível à evasão.

A resposta a esse cenário tem levado autores a incorporar a avaliação adversarial ao próprio protocolo experimental. Prasad et al. (2026) reportam, além da acurácia de detecção, uma robustez de 98,70% sob ataques construídos pelo método do sinal do gradiente (*Fast Gradient Sign Method*), acompanhada de um módulo dedicado de avaliação adversarial. Esse procedimento é o antecedente direto do teste de estresse conduzido nesta pesquisa.

Para os fins desta dissertação, a robustez é entendida como a capacidade de preservar desempenho útil diante de modificações controladas que mantenham a natureza maliciosa da amostra. Essa definição diferencia ruído aleatório de manipulação orientada à evasão e exige que o teste adversarial seja aplicado a um *pipeline* previamente treinado, sem reajuste sobre as amostras perturbadas, para que a degradação

observada represente efetivamente a sensibilidade do sistema.

Cabe uma delimitação final de escopo. A avaliação adversarial não deve presumir que toda perturbação tecnicamente válida permaneça eficaz contra usuários. Yuan et al. (2024) conduziram dois estudos com 470 participantes para examinar como usuários percebem páginas adversariais, tanto sintéticas quanto reais coletadas na Web, e concluíram que a maioria delas apresenta efetividade comparável à das páginas não perturbadas, embora nem todas as perturbações sejam igualmente efetivas: perturbações que introduzem erros de digitação são significativamente mais perceptíveis do que alterações de maior magnitude visual, como a substituição do plano de fundo. Os autores observaram ainda correlação negativa entre a frequência autorrelatada de visita ao sítio de uma marca e a acurácia de detecção dos participantes, atribuída a excesso de confiança. Esta pesquisa não realiza estudo comportamental; essa literatura fundamenta, contudo, uma interpretação prudente, segundo a qual o teste adversarial de modelos representa uma dimensão de risco, mas não substitui a avaliação da persuasão exercida sobre o usuário. O detector automatizado e o usuário são alvos distintos da mesma cadeia de ataque.

2.10 SÍNTESE DO REFERENCIAL E APLICAÇÃO NESTA PESQUISA

Os fundamentos reunidos neste capítulo orientaram diretamente o desenho da pesquisa. A taxonomia por modalidade de evidência, apresentada na Seção 2.3, sustenta a separação dos preditores de Nível 0 em ramos por modalidade: um ramo tabular sobre atributos derivados de URL e de HTML e um ramo léxico sobre a cadeia bruta da URL. A decisão de excluir serviços de terceiros em tempo de inferência decorre das restrições operacionais discutidas na Seção 2.4.2, e a extração dos atributos de HTML a partir de instantâneos armazenados responde tanto ao viés de disponibilidade identificado na Seção 2.7.3 quanto ao custo de coleta quantificado na Seção 2.4.3.

A discussão sobre engenharia de características, na Seção 2.4, justifica a avaliação isolada e combinada dos blocos de URL e de HTML, bem como a construção de descritores derivados interpretáveis. O exame dos modelos baseados em árvores, na Seção 2.5.1, fundamenta a adoção de linhas de base tabulares fortes, sem as quais não é possível verificar se arquiteturas mais complexas produzem ganho real. Os conceitos de aprendizado profundo e de representações contextuais, nas Seções 2.5.2 e 2.5.3, explicam o uso de modelos capazes de aprender diretamente da sequência bruta da URL e delimitam a exigência de compará-los com alternativas mais simples sob protocolo idêntico.

Os princípios de comitês e o requisito de predições fora da dobra, na Seção 2.6, fundamentam a combinação das probabilidades dos aprendizes de Nível 0 por meta-aprendizes de Nível 1 e o controle de vazamento entre os níveis. Os conceitos de saturação do *benchmark* e de erro residual, definidos na Seção 2.8.2, orientam a interpretação dos resultados, de modo que o desempenho absoluto elevado não seja tomado, isoladamente, como evidência de superioridade arquitetural. A literatura de robustez, na Seção 2.9, fundamenta o teste de estresse adversarial em nível de *pipeline*, e a discussão de eficiência, na Seção 2.8.3, sustenta a caracterização de custos de treinamento e de inferência.

Por fim, a análise dos conjuntos de dados, na Seção 2.7, justifica a utilização de *benchmarks* com procedimentos de construção e espaços de características distintos, exigência que decorre diretamente da

formulação do problema apresentada na Seção 1.2. Com esses fundamentos estabelecidos, o Capítulo 3 examina os estudos primários que constituem o estado da arte imediato do problema investigado.

3 TRABALHOS RELACIONADOS

Este capítulo analisa os estudos primários que constituem o estado da arte imediato do objeto desta dissertação. O critério de inclusão adotado foi a proposição de um detector de *phishing* ou de URLs maliciosas avaliado empiricamente. Ficam de fora, portanto, as obras de revisão, os marcos teóricos, os repositórios de dados e os estudos de robustez adversarial, já discutidos, respectivamente, nas Seções 2.3, 2.5, 2.7 e 2.9 do Capítulo 2.

Os estudos foram organizados em quatro linhas, correspondentes às famílias que compõem ou contextualizam o arcabouço HOOFF-Stack-PUD: abordagens baseadas em características e aprendizado de máquina; abordagens baseadas em aprendizado profundo; detectores baseados em *Transformers*; e estratégias de empilhamento e meta-aprendizado. Para cada trabalho, registram-se a abordagem adotada, o resultado empírico reportado pelos autores e o aspecto que permanece fora de seu escopo em relação ao problema aqui investigado. O capítulo encerra com o quadro comparativo dos estudos, as lacunas identificadas e o posicionamento desta pesquisa.

3.1 ABORDAGENS BASEADAS EM CARACTERÍSTICAS E APRENDIZADO DE MÁQUINA

Aljofey et al. (2022) propõem abordagem que combina a sequência de caracteres da URL, informações de hiperlinks e o conteúdo textual da página, extraíndo atributos TF-IDF em nível de caractere a partir de trechos ruidosos do HTML e do texto simples e alimentando um classificador XGBoost. Diante da ausência de conjuntos públicos de grande porte à época, os autores construíram um conjunto próprio com 60.252 páginas, sendo 32.972 legítimas e 27.280 de *phishing*, e obtiveram acurácia de 96,76% com taxa de falsos positivos de 1,39% nesse conjunto, além de acurácia de 98,48% com 2,09% de falsos positivos em conjunto de referência. O trabalho é referência direta para esta pesquisa por integrar evidências de URL e de HTML sem depender de serviços de terceiros. Sua combinação, entretanto, concentra-se no espaço de características e em um classificador principal, ao passo que a presente investigação examina a complementaridade entre famílias de modelos no nível das probabilidades.

Gualberto et al. (2020b) propõem abordagem de engenharia de características apoiada em processamento de linguagem natural, lematização, modelagem de tópicos, técnicas aprimoradas de reamostragem e validação cruzada, e configuração de hiperparâmetros. O primeiro método emprega a totalidade dos atributos da matriz documento-termo, enquanto o segundo aplica a alocação latente de Dirichlet (*Latent Dirichlet Allocation*, LDA) para mitigar a maldição da dimensionalidade, a esparsidade e a perda de contexto textual, alcançando medida F1 de 99,95% com o algoritmo XGBoost, considerando apenas o corpo das mensagens. Em trabalho subsequente, Gualberto et al. (2020a) formalizam abordagem multiestágio na qual o primeiro método explora as estatísticas Qui-Quadrado e a informação mútua para redução de dimensionalidade e o segundo associa análise de componentes principais e análise semântica latente; combinados a XGBoost e *Random Forest*, os conjuntos reduzidos alcançaram medida F1 de 100% nos testes

com os corpora SpamAssassin Public Corpus e Nazario Phishing Corpus, com número substancialmente menor de atributos e menor custo computacional. Embora o domínio primário seja o correio eletrônico, esses trabalhos demonstram que a qualidade da representação e do *pipeline* pode ser tão determinante quanto a escolha do classificador, e seus resultados no limite superior da escala ilustram exatamente o fenômeno formalizado na Seção 2.8.2.

Karim et al. (2023) avaliam árvore de decisão, regressão linear, *Random Forest*, *Naive Bayes*, *Gradient Boosting*, K-vizinhos mais próximos (*K-Nearest Neighbors*, KNN) e *Support Vector Classifier*, e propõem o modelo híbrido LSD, que combina regressão logística, máquina de vetores de suporte (*Support Vector Machine*, SVM) e árvore de decisão sob votação suave e rígida, associado à seleção de características por *canopy*, validação cruzada e otimização de hiperparâmetros por *Grid Search*. O trabalho emprega votação, e não um meta-aprendiz treinado sobre predições fora da dobra de famílias heterogêneas, distinção central em relação à presente pesquisa.

Rafsanjani et al. (2024) propõem arcabouço de detecção de URLs maliciosas baseado em classificação estática pré-definida de características, mediante alocação de coeficientes de prioridade e métodos de avaliação de atributos. A classificação abrange 42 classes de características, agrupadas em listas de bloqueio, características léxicas, características baseadas em hospedagem e características baseadas em conteúdo. Para a validação, os autores coletaram 5.000 URLs reais provenientes de repositórios de *phishing* e de *malware*, especificamente URLhaus e PhishTank, e avaliaram o arcabouço com máquina de vetores de suporte, *Random Forest* e rede bayesiana, alcançando acurácia de 98,95% e precisão de 98,60%. A análise de *benchmarking* foi conduzida contra três métodos abrangentes de detecção, PDRCNN, o método de Li e o URLNet, com resultado favorável em acurácia e precisão. A contribuição situa-se na valoração das características, não na fusão de evidências em nível de probabilidade.

Mohanty et al. (2024) propõem técnica híbrida de seleção de características voltada à predição de URLs suspeitas em ambientes de Internet das Coisas. O argumento central é que o uso exclusivo de características léxicas, embora favoreça a velocidade de detecção, não permite recuperar informação abrangente sobre o sítio, razão pela qual os autores combinam características léxicas com dezoito características de conteúdo de página. A técnica associa um método de filtro baseado em ganho de informação mútua a uma busca *wrapper* conduzida por algoritmo genético, cujos operadores selecionam iterativamente os subconjuntos mais aptos. O subconjunto ótimo, obtido na vigésima população e composto por 26 características, alcançou, com classificador XGBoost, acurácia de 98,3%, precisão de 99%, taxa de detecção de 98% e medida F1 de 98%. O trabalho não isola a contribuição de modalidades distintas em protocolo empilhado.

Nayak, Muniyal e Belavagi (2025) analisam um conjunto com 58.645 URLs e 111 características, com o objetivo declarado de reduzir o custo associado à extração de atributos. Utilizando apenas 14 características, um modelo *feedforward* alcançou acurácia de 94,46%. Os autores avaliam ainda uma coleção de classificadores composta por rede neural profunda (*Deep Neural Network*, DNN), arquitetura *Wide and Deep* e TabNet, introduzem uma métrica denominada *anti-phishing score*, construída sobre falsos positivos e falsos negativos como complemento à acurácia, e submetem o modelo treinado a validação em um novo conjunto para avaliar generalização. A abordagem não contempla comparação pareada repetida entre arquiteturas.

Duarte et al. (2025) tratam da detecção precoce de URLs de *phishing* em domínios recém-registrados e

estacionados no contexto de uma instituição financeira. A partir de um conjunto curado de 211.659 URLs obtidas por monitoramento em tempo real de certificados de camada de sockets segura (*Secure Sockets Layer, SSL*), listagens de domínios populares e relatos de incidentes, um classificador *Light Gradient Boosting Machine* alcançou *recall* de 96,02% e acurácia de 97,28% em conjunto balanceado, validado por validação cruzada de dez dobras. Roy, Kumar e Singh (2024) investigam arcabouço de detecção voltado à proteção de aplicações centradas no consumidor, com ênfase no setor de saúde, relatando acurácia de 99,00%. Ambos aproximam a pesquisa de cenários operacionais, nos quais latência e antecipação são relevantes, mas nenhum deles compara regimes de saturação entre *benchmarks* distintos.

Kustiawan e Ghauth (2025b) propõem o PhishOFE, arcabouço de detecção em tempo real com engenharia de características otimizada, que extrai atributos de URL e de HTML e deriva atributos compostos com o objetivo de reduzir a dependência de dados de terceiros; nos experimentos com dez modelos, o CatBoost alcançou a maior acurácia, 99,48%. Em estudo comparativo associado, Kustiawan e Ghauth (2025a) avaliam quatro configurações de características: somente URL; somente HTML; URL + HTML; e URL + HTML + características derivadas, sobre dez modelos e o conjunto PhishOFE, com 101.063 URLs, obtendo 99,45% de acurácia com CatBoost sobre o conjunto completo. Os autores registram que atributos como *URLLength* e *NoOfSubDomain* figuram consistentemente entre os mais importantes e que características derivadas, como *SuspiciousCharRatio* e *URLComplexityScore*, elevam o desempenho em modelos específicos. Esse último resultado é o antecedente empírico mais próximo do descritor interpretável proposto nesta dissertação. A diferença de recorte está em que a presente pesquisa usa a avaliação por blocos para verificar se a evidência tabular, por si só, já satura o *benchmark*, condição em que a camada de meta-aprendizado não teria erro residual a corrigir.

3.2 ABORDAGENS BASEADAS EM APRENDIZADO PROFUNDO

O marco dessa linha, estabelecido por Le et al. (2018) e já discutido na Seção 2.5.2, orienta os trabalhos aqui analisados.

Sahingoz, Buber e Kugu (2024) apresentam o DEPHIDES, sistema de detecção de *phishing* baseado em aprendizado profundo que emprega cinco algoritmos distintos: redes neurais artificiais (*Artificial Neural Network, ANN*), redes neurais convolucionais, redes neurais recorrentes, redes recorrentes bidirecionais (*Bidirectional Recurrent Neural Network, BRNN*) e redes de atenção. O sistema é orientado à classificação rápida de páginas a partir da URL. Para avaliar o desempenho, os autores coletaram e disponibilizaram um conjunto rotulado com aproximadamente cinco milhões de registros, sobre o qual as redes convolucionais obtiveram o melhor resultado, com acurácia de detecção de 98,74%. A escala do conjunto é notável, mas o estudo não avalia o modelo como componente complementar de um comitê heterogêneo.

Zonyfar, Lee e Kim (2023) propõem o HCNN-LSTM, que integra uma rede convolucional unidimensional a uma rede de memória longa de curto prazo (*Long Short-Term Memory, LSTM*) com estimação compartilhada entre subcamadas. O modelo foi avaliado sobre conjunto de referência com 11.430 URLs e 87 atributos extraídos, dos quais 56 parâmetros derivados da estrutura e da sintaxe da URL foram selecionados, alcançando acurácia, precisão, *recall* e medida F1 de 95,19%, 95,00%, 95,00% e 95,00%,

respectivamente, e superando as variantes isoladas de CNN e de LSTM. A escala reduzida do conjunto e a ausência de teste estatístico pareado impedem determinar se o ganho decorre de complementaridade entre os componentes ou da separabilidade da base.

Sruthi e Naik S (2025) propõem arcabouço fundamentado em rede siamesa que integra sub-redes LSTM gêmeas com pesos compartilhados, transformando sequências de URL em representações latentes cuja comparação, em pares, distingue amostras de *phishing* de amostras legítimas. O ponto central da abordagem é um conjunto pareado especialmente curado, projetado para viabilizar análise de similaridade de granularidade fina. Sobre os conjuntos de referência, o modelo alcançou acurácia de 99,68% com desvio de 0,08 ponto percentual, precisão de 99,87% e medida F1 de 99,68%. A contribuição mais informativa para esta dissertação está na avaliação transdomínio: submetido a um corpus de URLs geradas por inteligência artificial generativa, o modelo registrou precisão de 89,06%, *recall* de 99,75% e medida F1 de 94,10%, o que evidencia queda concentrada na precisão e, portanto, aumento de falsos positivos sob deslocamento de distribuição. O trabalho não avalia o modelo como componente de um comitê tabular-léxico.

He et al. (2024a) propõem modelo de detecção baseado em extensão de características, no qual o algoritmo TextRank é utilizado para gerar uma biblioteca de extensão cujos termos são embutidos na URL a ser analisada antes da vetorização e do processamento por rede convolucional. O modelo alcançou acurácia de 99,75%, *recall* de 99,77% e medida F1 de 99,72%. Os autores reportam ainda duas medidas de eficiência relevantes: o tempo de treinamento correspondeu a aproximadamente um terço do exigido por um método equivalente baseado em BERT, e a detecção de dez mil amostras consumiu 0,92 s, o que sustenta a adequação da proposta a execução no lado do cliente. A avaliação não contempla complementaridade entre famílias de modelos.

Rastakhiz, Eftekhari e Vahdati (2024) propõem o QuickCharNet, arquitetura de classificação de URLs que combina convolução em nível de caractere com representação em nível de *token*, motivada pelas limitações de tempo e de memória que os métodos existentes enfrentam nas etapas de pré-processamento, processamento e inferência em cenários de decisão rápida. As contribuições declaradas incluem o uso de agregação por *max pooling* e *mean pooling* para converter *embeddings* de caractere em *embeddings* de *token*, a exploração de tokenizadores de subpalavra para representação ótima da URL e a análise comparativa de cinco modelos por acurácia, medida F1 e visualizações por incorporação estocástica de vizinhos com distribuição *t* (*t-Distributed Stochastic Neighbor Embedding*, t-SNE). O modelo foi treinado sobre um conjunto desenvolvido para o estudo e sobre dois conjuntos de referência, com aplicação dupla: detecção de *spam* e classificação temática associada ao posicionamento em resultados de busca. O recorte é de modalidade única, sem integração com atributos tabulares.

Jiang et al. (2025) propõem o D-PhishNet, arquitetura de duplo ramo que combina redes neurais em grafos (*Graph Neural Network*, GNN), aplicadas à informação estruturada da URL, e modelos *Transformer*, aplicados ao conteúdo textual do HTML. O trabalho introduz um módulo aprimorado de fusão de características, denominado DBA-CMixer, destinado a mitigar a perda de informação e a lacuna semântica das estratégias tradicionais de integração, e uma estratégia de treinamento progressivo que trata conflitos de gradiente e problemas de convergência no treinamento ponta a ponta de modelos multiramo. A avaliação emprega três conjuntos: D1, de grande escala, com 60.252 amostras, sendo 27.280 de *phishing* e 32.972 legítimas, composição idêntica à do conjunto construído por Aljofey et al. (2022); D2, desbalanceado, com

10.305 amostras, sendo 605 de *phishing* e 9.700 legítimas; e D3, construído no próprio estudo com 10.000 amostras equilibradas e utilizado exclusivamente para teste, com o objetivo de avaliar a generalização sobre amostras emergentes. O D-PhishNet alcançou acurácia de 0,9930, 0,9920 e 0,9890 em D1, D2 e D3, respectivamente, superando URLNet (0,9760 em D1), URL_XGBoost, HTMLPhish, TransURL e WebPhish, e os autores registram que modelos que exploram simultaneamente URL e HTML superam consistentemente os modelos que utilizam apenas URL. Esse é o trabalho mais próximo do presente quanto à premissa de complementaridade entre modalidades. A diferença metodológica é determinante: no D-PhishNet a fusão ocorre internamente à arquitetura, por combinação de representações, ao passo que nesta dissertação a integração ocorre em nível de probabilidade, por meio de meta-aprendizes treinados sobre estimativas fora da dobra, o que preserva os aprendizes como componentes separáveis e mensuráveis individualmente.

3.3 ABORDAGENS BASEADAS EM TRANSFORMERS

Aljofey et al. (2025) propõem o BERT-PhishFinder, que adota projeto otimizado baseado em DistilBERT, incorporando *SpatialDropout1D* nas camadas de *embedding* e combinando *global average pooling* e *global max pooling*; as representações agrupadas são concatenadas e processadas por múltiplas camadas densas paralelas com parâmetros distintos e regularização por *dropout*. As avaliações, conduzidas contra algoritmos convencionais de aprendizado profundo, contra modelos *Transformer*, XLNet, RoBERTa e ALBERT, além de outros métodos existentes em cinco conjuntos de referência, indicam desempenho competitivo com menor dependência de rótulos. O trabalho não avalia conjuntamente saturação, custo, blocos de características e robustez sob um mesmo *pipeline*.

Remya et al. (2025) propõem o BGL-PhishNet, método híbrido que articula BERT para análise textual, redes neurais em grafos para análise de URLs e LightGBM para extração de características de metadados, relatando acurácia de 97,3% e redução de falsos positivos. A combinação é arquitetural, o que impede isolar o ganho atribuível ao meta-aprendizado sobre linhas de base correspondentes.

Wei, Nakayama e Sekiya (2025) propõem abordagem multimodal que combina um modelo BERT ajustado para processamento de URL a características externas auxiliares derivadas de recursos públicos da Internet, com foco explícito na capacidade de generalização. Os autores disponibilizam ainda o conjunto PhishMail, com 8.937 amostras de *phishing* coletadas ao longo de período estendido a partir de mensagens maliciosas recebidas diariamente, destinado à simulação de cenários de detecção de dia zero. A dependência de recursos públicos externos em tempo de inferência constitui a principal diferença em relação ao recorte adotado nesta pesquisa, conforme a restrição enunciada na Seção 2.4.2.

Almoussa e Anwar (2023) investigam a viabilidade de modelos de linguagem sensíveis a caracteres para a detecção de ataques semânticos sociais baseados em URL, categoria que abrange *phishing*, *spam*, desfiguração e *malware*. Os autores desenvolveram três modelos, baseado em LSTM, baseado em CNN e baseado em CharacterBERT, além de registrar a acurácia global de 99,65% pela média de validação cruzada de cinco dobras com o modelo CharacterBERT, que apresentou o melhor desempenho por classe entre os três, alcançando 99,90% na detecção de ataques de desfiguração. A avaliação não contempla comparação com linhas de base tabulares fortes sob protocolo idêntico.

3.4 ABORDAGENS BASEADAS EM EMPILHAMENTO E META-APRENDIZADO

Motiur Rahman, Islam e Jabiullah (2020) avaliam a generalização empilhada especificamente na detecção de URLs de *phishing* com o modelo PhishStack. O experimento emprega três conjuntos públicos provenientes dos repositórios UCI e Mendeley Data: D1, com 11.055 URLs, sendo 4.898 de *phishing* e 6.157 legítimas, e 30 características; D2, com 1.353 URLs e 10 características; e D3, com 10.000 URLs, sendo 5.000 de *phishing* e 5.000 legítimas, e 48 características. No Nível 1 são utilizados *Random Forest*, árvore de decisão, *Multilayer Perceptron*, máquina de vetores de suporte, gradiente descendente estocástico e *Gaussian Naive Bayes* com validação cruzada; no Nível 2, um classificador XGBoost. A generalização empilhada alcançou 97,44% de acurácia em D1 e 97,86% em D3, superando os classificadores individuais em acurácia global, embora os autores registrem que *Random Forest* e *Multilayer Perceptron* apresentam melhor precisão e *recall*. Esse resultado é ilustrativo do problema central desta dissertação: o ganho do empilhamento manifesta-se na métrica agregada, mas o estudo não caracteriza sistematicamente as condições em que ele deixa de agregar valor por saturação.

Indrasiri, Halgamuge e Mohammad (2021) propõem o *Expandable Random Gradient Stacked Voting Classifier* (ERG-SVC), avaliado com sete algoritmos de classificação, um algoritmo de agrupamento e duas técnicas de comitê sobre dois conjuntos legítimos de grande porte, com 73.575 e 100.000 URLs. Os autores relatam que o classificador *Gradient Boosting* alcançou 98,118% de acurácia com o menor número de características baseadas em processamento de linguagem natural, enquanto o modelo empilhado e o modelo de votação proposto atingiram, respectivamente, 98,23% e 98,27%, após redução do conjunto de características de 46 para 22. A análise não se centra em predições fora da dobra de modalidades heterogêneas nem em comparações pareadas repetidas.

Kalabarige et al. (2022) propõem modelo de comitê empilhado multicamada no qual as predições dos estimadores da camada corrente são fornecidas como entrada à camada seguinte, alcançando acurácia entre 96,79% e 98,90% em quatro conjuntos distintos, provenientes dos repositórios UCI e Mendeley. Em trabalho posterior, Kalabarige et al. (2023) incorporam seleção híbrida de características baseada em *boosting* ao comitê multicamada, registrando acurácia entre 96,16% e 98,95% sem seleção de características e entre 96,18% e 98,80% com seleção. Em nenhum dos dois trabalhos a complementaridade residual é avaliada explicitamente diante de linhas de base tabulares quase saturadas, nem se examinam conjuntamente robustez adversarial e eficiência do *pipeline*.

Aslam et al. (2024) propõem o AntiPhishStack, modelo de generalização empilhada em duas fases que aprende de forma simétrica características de URL e TF-IDF em nível de caractere. Na Fase I, as características são treinadas em um classificador de base com validação cruzada em K dobras; na Fase II, emprega-se uma rede LSTM empilhada em duas camadas com cinco otimizadores adaptativos, e as predições simétricas de ambas as fases são integradas para treinar um meta-classificador XGBoost, alcançando 96,04% de acurácia em dois conjuntos de referência sem conhecimento prévio de características específicas de *phishing*. O trabalho não avalia robustez adversarial em nível de *pipeline* nem compara regimes distintos de separabilidade.

He et al. (2024b) propõem método de detecção de sítios de *phishing* baseado em empilhamento com tiny-Bert. O conceito central consiste em utilizar o tiny-Bert para extrair características diretamente das

cadeias de URL, aprendendo características semânticas e de dependência de longo alcance, e em construir sobre essas representações um classificador por empilhamento com quatro aprendizes de base: CatBoost, XGBoost e LightGBM no primeiro nível e árvore de decisão com *gradient boosting* (*Gradient Boosting Decision Tree*, GBDT) no segundo. O modelo dispensa extração manual de características e, segundo os autores, permite que os aprendizes de base compensem mutuamente os erros de classificação, melhorando a generalização. Avaliado sobre um conjunto baseado em sítios reais de *phishing*, o modelo alcançou acurácia de até 99,14% e *recall* de até 99,13%, com estabilidade superior à das alternativas comparadas. Esse é o trabalho estruturalmente mais próximo do arcabouço aqui proposto. A diferença de recorte está em que o presente estudo combina modalidades distintas, tabular e léxica, em vez de uma única representação contextual, e submete o resultado a verificação estatística pareada, análise por blocos de características, perfil de eficiência e teste adversarial sob protocolo comum.

Bari et al. (2025) propõem combinação de métodos de seleção por filtro, remoção de características constantes, duplicadas, quase constantes e correlacionadas, extração de informação mútua e teste de análise de variância (ANOVA), avaliada com *Random Forest*, redes neurais artificiais, Ada-Boost, XGBoost, regressão logística, árvores de decisão, *Gradient Boosting*, SVM e dois modelos de comitê, empilhamento e votação majoritária. O classificador empilhado composto por *Gradient Boosting*, *Random Forest* e SVM alcançou taxa de falsos positivos de 1,31% e acurácia de 98,17% no primeiro conjunto, 2,81% e 97,61% no terceiro e 3,47% e 96,47% no segundo. O foco recai sobre a seleção de características, e não sobre a complementaridade entre modalidades.

Naseeb et al. (2025) propõem comitê empilhado que combina redes neurais artificiais e *Bagging* com K-vizinhos mais próximos, usando regressão logística como meta-classificador, com acurácia de 97,20%, precisão de 97% e *recall* de 97% em conjuntos de 2019 a 2024, empregando a técnica de sobreamostragem sintética da classe minoritária (*Synthetic Minority Over-sampling Technique*, SMOTE) para tratamento do desbalanceamento e validação cruzada em K dobras. Khalid et al. (2024) investigam TF-IDF, Word2Vec e Doc2Vec associados a algoritmos clássicos e ao modelo de empilhamento LogiTriBlend, que combina SVM, regressão logística, *Random Forest* e XGBoost com meta-aprendiz de regressão logística; sobre conjunto com 501 mensagens de *phishing* e 4.090 legítimas, tratado com SMOTE, o modelo alcançou 99,34% de acurácia com Doc2Vec, contra 99,12% e 98,80% com Word2Vec e TF-IDF, respectivamente. Este último estudo é relevante por evidenciar que a representação textual condiciona o desempenho do comitê, ainda que seu objeto seja a mensagem de correio eletrônico, e não a URL.

O conjunto desses trabalhos demonstra que o empilhamento é estratégia recorrente e flexível na área. Contudo, as configurações variam quanto à origem das características, à geração das predições do meta-nível, às linhas de base adotadas, ao número de execuções e à avaliação de custo. Essa heterogeneidade dificulta concluir, de forma geral, em quais condições o empilhamento agrega valor, lacuna da qual parte esta dissertação.

3.5 QUADRO COMPARATIVO DOS TRABALHOS RELACIONADOS

A Tabela 3.1 consolida os estudos primários discutidos nas seções anteriores e os contrapõe ao recorte desta pesquisa. As colunas apresentam a abordagem principal, o resultado empírico reportado pelos autores e o aspecto que permanece fora do escopo de cada trabalho em relação ao problema aqui investigado. Convém esclarecer o sentido dessa última coluna: o aspecto ali indicado não representa deficiência do trabalho citado, mas o elemento que permanece fora de seu escopo quando comparado ao problema investigado nesta dissertação.

Tabela 3.1 – Quadro comparativo dos trabalhos relacionados

Autor (ano)	Abordagem	Resultado reportado	Aspecto fora do escopo
Aljofey et al. (2022)	URL, hiperlinks e texto com XGBoost	96,76% de acurácia e 1,39% de FPR em conjunto próprio de 60.252 páginas	Combinação no espaço de características; sem meta-aprendizado sobre probabilidades
Gualberto et al. (2020b)	PLN e modelagem de tópicos (LDA)	Medida F1 de 99,95% com XGBoost	Domínio de correio eletrônico; sem combinação de famílias tabular e léxica
Gualberto et al. (2020a)	<i>Pipeline</i> textual multiestágio	Medida F1 de 100% em SpamAssassin e Nazario	Não investiga saturação de <i>benchmark</i> nem complementaridade probabilística
Karim et al. (2023)	Modelo híbrido LSD com votação	Superioridade sobre os classificadores isolados avaliados	Votação, sem meta-aprendiz treinado sobre previsões OOF
Roy, Kumar e Singh (2024)	Aprendizado de máquina para aplicações do consumidor	99,00% de acurácia	Recorte setorial; sem análise de regimes de saturação
Mohanty et al. (2024)	Filtro por informação mútua e <i>wrapper</i> genético (IoT)	98,3% de acurácia e 99% de precisão com 26 características	Não isola a contribuição de modalidades em protocolo empilhado
Rafsanjani et al. (2024)	Coefficiente de prioridade sobre 42 classes de características	98,95% de acurácia e 98,60% de precisão sobre 5.000 URLs	Valoração de características; sem fusão em nível de probabilidade
Nayak, Muniyal e Belavagi (2025)	Seleção de características, DNN, <i>Wide and Deep</i> e TabNet	94,46% com 14 de 111 características sobre 58.645 URLs	Sem comparação pareada repetida entre arquiteturas
Kustiawan e Ghauth (2025a)	Comparação de blocos URL, HTML e derivados	99,45% com CatBoost sobre 101.063 URLs	Não avalia fusão em nível de probabilidade nem estabilidade dos ganhos

Continua na próxima página

Tabela 3.1 – continuação da página anterior

Autor (ano)	Abordagem	Resultado reportado	Aspecto fora do escopo
Duarte et al. (2025)	Detecção precoce em domínios estacionados	<i>Recall</i> de 96,02% e acurácia de 97,28% sobre 211.659 URLs	Recorte operacional; não compara regimes de saturação entre <i>benchmarks</i>
Kustiawan e Ghauth (2025b)	PhishOFE com engenharia otimizada	99,48% de acurácia com CatBoost	Otimização no espaço de características; sem meta-aprendizado OOF
Zonyfar, Lee e Kim (2023)	HCNN-LSTM	95,19% de acurácia sobre 11.430 URLs e 56 atributos	Não examina se o ganho decorre de complementaridade ou de separabilidade
Sahingo, Buber e Kugu (2024)	DEPHIDES: ANN, CNN, RNN, BRNN e atenção	98,74% com CNN sobre conjunto de cerca de cinco milhões de URLs	Não avalia o modelo como componente de comitê heterogêneo
He et al. (2024a)	Extensão de características por TextRank e CNN	99,75% de acurácia; 0,92 s para 10.000 amostras	Não avalia complementaridade entre famílias de modelos
Rastakhiz, Eftekhari e Vahdati (2024)	QuickCharNet: convolução de caractere e <i>token</i>	Comparação de cinco modelos em conjunto próprio e dois de referência	Modalidade única; sem integração com atributos tabulares
Jiang et al. (2025)	D-PhishNet: GNN e <i>Transformer</i> com fusão DBA-CMixer	Acurácia de 0,9930, 0,9920 e 0,9890 em D1, D2 e D3	Fusão interna à arquitetura; sem meta-probabilidades OOF entre modelos independentes
Sruthi e Naik S (2025)	Rede siamesa com LSTM gêmeas de pesos compartilhados	99,68% de acurácia; F1 de 94,10% em teste transdomínio	Não avalia o modelo como componente de comitê tabular-léxico
Almousa e Anwar (2023)	LSTM, CNN e CharacterBERT	99,65% de acurácia média em cinco dobras	Não compara com linhas de base tabulares fortes sob protocolo idêntico
Aljofey et al. (2025)	BERT-PhishFinder com DistilBERT	Desempenho competitivo em cinco conjuntos de referência	Não avalia conjuntamente saturação, custo, blocos e robustez
Remya et al. (2025)	BGL-PhishNet: BERT, GNN e LightGBM	97,3% de acurácia com redução de falsos positivos	Combinação arquitetural; não isola o ganho do meta-aprendizado
Wei, Nakayama e Sekiya (2025)	BERT multimodal com características externas	Conjunto PhishMail com 8.937 amostras para cenário de dia zero	Depende de recursos públicos externos em tempo de inferência

Continua na próxima página

Tabela 3.1 – continuação da página anterior

Autor (ano)	Abordagem	Resultado reportado	Aspecto fora do escopo
He et al. (2024b)	tiny-Bert com empilhamento (CatBoost, XGBoost, LightGBM e GBDT)	99,14% de acurácia e 99,13% de <i>recall</i>	Representação única; sem verificação estatística pareada, blocos, eficiência e teste adversarial
Motiur Rahman, Islam e Jabiullah (2020)	PhishStack: seis aprendizes de base e meta-XGBoost	97,44% em D1 e 97,86% em D3	Não caracteriza quando o empilhamento deixa de agregar valor por saturação
Indrasiri, Halgamuge e Mohammad (2021)	ERG-SVC: votação e empilhamento	98,27% com redução de 46 para 22 características	Características exclusivamente léxicas; sem evidência de HTML
Kalabarige et al. (2022)	Comitê empilhado multicamada	96,79% a 98,90% em quatro conjuntos	Sem intervalos de confiança sobre diferenças pareadas
Kalabarige et al. (2023)	Seleção híbrida e empilhamento multicamada	96,16% a 98,95% sem seleção; 96,18% a 98,80% com seleção	Não avalia conjuntamente robustez adversarial e eficiência
Aslam et al. (2024)	AntiPhishStack com meta-XGBoost	96,04% de acurácia em dois conjuntos de referência	Não compara regimes de separabilidade nem testa robustez
Bari et al. (2025)	Filtros de seleção e empilhamento	FPR de 1,31% e acurácia de 98,17%	Foco na seleção de características; saturação não é objeto central
Khalid et al. (2024)	LogiTriBlend para correio eletrônico	99,34% de acurácia com Doc2Vec	Objeto é a mensagem, não a complementaridade URL bruta e URL/HTML
Naseeb et al. (2025)	ANN e <i>Bagging</i> KNN com meta-aprendiz linear	97,20% de acurácia, 97% de precisão e de <i>recall</i>	Sem análise por blocos, robustez de <i>pipeline</i> e perfil de eficiência
Esta dissertação	HOOF-Stack-PUD: empilhamento heterogêneo OOF	Avaliação condicional em D1, D2 e D3; conjunto HybridPhisher; descritor URLC	Articula saturação, complementaridade, estabilidade estatística, eficiência e robustez

A leitura do quadro revela três padrões. Os trabalhos baseados em características e aprendizado de máquina investem na representação e reportam desempenho elevado, sem examinar se o conjunto já é quase separável sob essa representação. Os trabalhos baseados em aprendizado profundo e em *Transformers* avançam na aprendizagem de representações, mas nem sempre comparam seus componentes a linhas de base tabulares fortes sob protocolo idêntico. Os trabalhos baseados em empilhamento exploram a complementaridade entre aprendizes, porém raramente articulam, de forma conjunta, proteção contra vazamento, verificação estatística pareada, análise de eficiência e teste de robustez.

Um segundo padrão, mais específico, emerge da coluna de resultados. Vinte e dois dos trinta estudos relacionados reportam acurácia ou medida F1 superior a 96%, e treze reportam valores superiores a 99%, obtidos sobre conjuntos de dados, protocolos de divisão e espaços de características distintos. Essa concentração no extremo superior da escala é precisamente o sintoma que motiva o conceito de saturação de *benchmark* apresentado na Seção 2.8.2: quando quase todos os métodos reportam valores próximos ao teto, a métrica agregada deixa de discriminar arquiteturas, e a comparação passa a exigir instrumentos adicionais.

3.6 LACUNAS IDENTIFICADAS NA LITERATURA

A revisão apresentada permite identificar seis lacunas convergentes, que sustentam o recorte desta pesquisa.

A primeira é a interpretação insuficiente de resultados próximos ao limite superior da escala. Parte significativa dos estudos enfatiza a maior métrica alcançada sem distinguir avanço arquitetural de separabilidade produzida pelo conjunto de características (Kustiawan; Ghauth, 2025a; Gualberto et al., 2020b; Gualberto et al., 2020a; He et al., 2024a; Kustiawan; Ghauth, 2025b; Sruthi; Naik S, 2025). Quando um modelo tabular forte já apresenta erro residual mínimo, a utilidade de uma arquitetura adicional precisa ser demonstrada por comparação pareada, e não por diferença nominal.

A segunda é a avaliação incompleta da complementaridade. Trabalhos de empilhamento demonstram que combinações podem superar componentes individuais, porém nem sempre analisam se os modelos de base utilizam modalidades efetivamente distintas ou se produzem erros diferentes (Aslam et al., 2024; Bari et al., 2025; He et al., 2024b; Kalabarige et al., 2022; Naseeb et al., 2025). Conforme discutido na Seção 2.6.1, a diversidade nominal entre algoritmos não equivale a diversidade de informação.

A terceira refere-se ao controle metodológico do meta-aprendizado. A geração de predições fora da dobra, a separação da partição de teste e o ajuste de limiares exclusivamente sobre dados de treinamento precisam ser descritos de forma explícita; sem esses controles, o ganho atribuído ao empilhamento pode estar contaminado por vazamento (Indrasiri; Halgamuge; Mohammad, 2021; Khalid et al., 2024).

A quarta é a ausência de linhas de base tabulares suficientemente fortes em parte das comparações. Modelos de *boosting* e arquiteturas profundas para dados tabulares podem saturar *benchmarks* enriquecidos por engenharia de características (Chen; Guestrin, 2016; Duarte et al., 2025; Mohanty et al., 2024; Nayak; Muniyal; Belavagi, 2025); comparar um meta-modelo apenas com classificadores mais fracos superestima a contribuição da combinação.

A quinta é a dependência de execução única. Variações de semente e de divisão podem alterar diferenças pequenas, de modo que avaliações repetidas, deltas pareados e intervalos de confiança tornam-se necessários quando o ganho esperado é da ordem do erro residual remanescente. Entre os trabalhos analisados, apenas um reporta medida de dispersão associada à métrica principal (Sruthi; Naik S, 2025), enquanto os demais apresentam valores pontuais (Jiang et al., 2025; Kalabarige et al., 2022; Kalabarige et al., 2023; Zonyfar; Lee; Kim, 2023).

A sexta é a análise fragmentada entre desempenho, robustez e eficiência. Os estudos que documentam degradação sob perturbação adversarial (Kulkarni et al., 2025; Lo Re; Morana; Rizzo, 2026; Nowroozi et al., 2023; Pillai et al., 2024; Sabir et al., 2024) raramente propõem arquiteturas de detecção, ao passo que os estudos que propõem arquiteturas raramente incorporam teste de estresse adversarial ao protocolo de avaliação, com exceções recentes (Prasad et al., 2026). Essa separação é agravada pela heterogeneidade na construção dos conjuntos de dados (Cisco Talos Intelligence Group, 2026; Phishing Database Project, 2026; Skula; Kvet, 2024) e permanece relevante mesmo quando se considera que páginas adversariais afetam também a percepção dos usuários (Yuan et al., 2024), dimensão situada fora do escopo deste trabalho.

3.7 POSICIONAMENTO DA PESQUISA

Esta dissertação se posiciona na interseção entre detecção baseada em características, aprendizado de representações e generalização empilhada. O que a distingue dos trabalhos examinados neste capítulo não está na arquitetura empregada, e sim na avaliação da saturação do *benchmark*, construída a partir das lacunas da Seção 3.6.

Às três primeiras lacunas corresponde o núcleo do protocolo. A interpretação de resultados próximos ao limite superior da escala é enfrentada por uma análise de saturação conduzida comparativamente sobre PhiUSIIL (D1) (Prasad; Chandra, 2024), StealthPhisher (D2) (Prasad et al., 2026) e HybridPhisher (D3), conjuntos cujos procedimentos de construção e espaços de atributos diferem entre si. A complementaridade entre modelos deixa de ser inferida dos nomes dos algoritmos e passa a ser medida pela decomposição do desempenho em blocos de atributos de URL, de HTML e de URL + HTML, submetidos aos classificadores tabulares isolados, o que indica se algum bloco é dispensável e se ainda resta erro residual para a camada de meta-aprendizado. E o controle metodológico do meta-aprendizado torna-se explícito por meio de um protocolo de empilhamento heterogêneo fora da dobra protegido contra vazamento, no qual pré-processamento, seleção de características, calibração e limiares se ajustam exclusivamente sobre a partição de treinamento, a exemplo dos estudos de empilhamento correlatos (Aslam et al., 2024; He et al., 2024b; Indrasiri; Halgamuge; Mohammad, 2021; Kalabarige et al., 2022; Kalabarige et al., 2023; Motiur Rahman; Islam; Jabiullah, 2020).

Às três últimas corresponde a moldura experimental. A ausência de referências suficientemente fortes é enfrentada pelo confronto sistemático dos meta-modelos com classificadores baseados em *boosting* e com arquiteturas tabulares profundas (Chen; Guestrin, 2016; Duarte et al., 2025; Mohanty et al., 2024; Nayak; Muniyal; Belavagi, 2025), o que impede que a utilidade da combinação seja aferida contra linhas de base frágeis. A dependência de execução única é tratada por execuções repetidas com sementes fixas, diferenças pareadas e intervalos de confiança, acompanhadas da análise do tempo de inferência. E desempenho, robustez e eficiência são examinados em conjunto pela incorporação, ao mesmo protocolo, do perfilamento dos custos de treinamento e de inferência e de um teste de estresse adversarial em nível de *pipeline*, que perturba evidências derivadas da URL e do HTML e mede a degradação dos modelos de base e dos meta-modelos (Kulkarni et al., 2025; Lo Re; Morana; Rizzo, 2026; Nowroozi et al., 2023; Pillai et al., 2024; Prasad et al., 2026; Sabir et al., 2024).

Duas decisões relativas aos dados completam esse posicionamento. A primeira é a construção do conjunto HybridPhisher, que preserva a cadeia textual da URL e os atributos derivados de URL e de HTML para as mesmas instâncias, a partir de fontes de *phishing* verificadas (Cisco Talos Intelligence Group, 2026; Phishing Database Project, 2026) e segundo as recomendações de equilíbrio e transparência discutidas na literatura de preparação de conjuntos (Skula; Kvet, 2024). A segunda é a incorporação, ao esquema de atributos, de dois descritores compostos interpretáveis, o URLC, proposto nesta pesquisa, e o Domain_Similarity, adaptado do índice de similaridade de Prasad e Chandra (2024), ambos submetidos a comparações pareadas com e sem o atributo sob as configurações somente URL, somente HTML e URL + HTML, e o primeiro deles submetido ainda a análise de sensibilidade dos pesos de seus componentes, em continuidade à evidência de contribuição de características derivadas reportada na literatura comparativa (Kustiawan; Ghauth, 2025a; Kustiawan; Ghauth, 2025b).

A revisão apresentada neste capítulo evidencia que a literatura dispõe de modelos eficazes para representação de URLs, exploração de HTML, seleção de características, aprendizado profundo, representações contextuais e comitês. A lacuna não está, portanto, na ausência de classificadores, mas na necessidade de um protocolo capaz de explicar em quais condições a heterogeneidade entre eles produz ganho confiável. É essa questão que orienta a arquitetura apresentada no Capítulo 4.

4 ARQUITETURA DA PROPOSTA

Este capítulo descreve a arquitetura proposta para responder ao problema formulado no Capítulo 1, tomando como base os fundamentos reunidos nos Capítulos 2 e 3. A exposição inicia-se pela caracterização da pesquisa e por uma visão geral do arcabouço HOOF-Stack-PUD, de modo que o quadro completo esteja disponível antes que cada uma de suas partes seja detalhada. Em seguida, são apresentados a construção do conjunto de dados HybridPhisher, o esquema de características adotado, a análise exploratória que orientou a regularização desse esquema e o protocolo de aprendizado empilhado responsável por integrar os dois níveis de modelos. A última seção descreve o protocolo experimental que orienta as avaliações apresentadas no Capítulo 5.

4.1 CARACTERIZAÇÃO DA PESQUISA

Quanto à natureza, esta pesquisa é aplicada, uma vez que se volta para um problema concreto de segurança cibernética, que consiste em determinar sob quais condições a combinação heterogênea de modelos de aprendizado de máquina produz ganho mensurável na detecção de URLs de *phishing*. O interesse não está no conhecimento abstrato a respeito de arquiteturas de classificação, mas na formulação de critérios de decisão que sejam úteis a quem projeta ou avalia detectores em operação.

Quanto aos objetivos, o estudo é ao mesmo tempo exploratório e descritivo. Ele é exploratório porque investiga um fenômeno ainda pouco caracterizado na literatura, que é a saturação de *benchmark* na detecção de *phishing* e o efeito que essa saturação produz sobre a interpretação dos ganhos atribuídos a novas arquiteturas, tema discutido na Seção 2.8.2. E é descritivo porque documenta de forma sistemática como nove modelos individuais e dois meta-modelos se comportam sobre três conjuntos de dados que apresentam graus distintos de separabilidade.

Quanto à abordagem, a pesquisa é quantitativa, pois todas as conclusões derivam de medidas objetivas obtidas sobre partições de teste isoladas, que foram submetidas a execuções repetidas, a comparações pareadas e à estimação de intervalos de confiança. Não houve coleta de percepções, entrevistas ou análise qualitativa de conteúdo, de maneira que a dimensão humana do *phishing*, reconhecida na Seção 2.1, permanece fora do escopo deste trabalho.

Quanto aos procedimentos técnicos, a pesquisa articula três instrumentos que se complementam. A pesquisa bibliográfica sustentou a fundamentação teórica e permitiu identificar as lacunas apresentadas na Seção 3.6, que delimitaram o recorte do trabalho. A coleta ativa de dados na Web deu origem ao conjunto HybridPhisher, cujo procedimento de construção é descrito na Seção 4.3. Por fim, o experimento controlado permitiu que as variáveis de interesse, isto é, a família de modelos, o bloco de características, a presença dos descritores derivados e a condição adversarial, variassem uma de cada vez sob um mesmo protocolo, mantidos fixos o particionamento, as sementes e o pré-processamento.

A opção pelo experimento controlado decorre diretamente da pergunta que orienta a pesquisa. Como

o objetivo não consiste em maximizar uma métrica, mas em identificar as condições sob as quais o empilhamento acrescenta informação útil, torna-se necessário isolar cada fator para que seja possível medir o efeito que ele produz sobre o erro residual, e não apenas sobre o valor absoluto alcançado pela métrica.

4.2 VISÃO GERAL DO HOOF-STACK-PUD

O HOOF-Stack-PUD (*Heterogeneous Out-of-Fold Stacked Meta-Learning Framework for Phishing URL Defense*) é um arcabouço de meta-aprendizado empilhado que gera probabilidades fora da dobra e que integra, por meio de ramos específicos para cada modalidade de evidência, tanto as representações brutas da URL quanto os atributos tabulares derivados da URL e do conteúdo HTML.

Antes de detalhar cada uma das partes, convém fixar uma distinção terminológica que percorre todo o texto. O termo arcabouço designa a concepção arquitetural recém-descrita, ao passo que *pipeline* designa a cadeia de execução que dela decorre, com as etapas, os custos e os tempos que lhe são próprios.

Fixada essa distinção, o que o arcabouço acrescenta não é mais um classificador isolado, e sim um protocolo controlado de integração e de avaliação, formulado para verificar a hipótese enunciada na Seção 1.2, segundo a qual o empilhamento heterogêneo reduz o erro residual deixado por aprendizes fortes de modalidade única somente quando a separabilidade do *benchmark* preserva alguma margem para essa correção. O princípio de combinação em si remonta à generalização empilhada, formulada originalmente por Wolpert (1992) e discutida na Seção 2.6, de modo que a contribuição deste trabalho se situa no protocolo de avaliação da saturação do *benchmark* aplicado àquela formulação clássica, e não em uma reformulação teórica dela.

A Figura 4.1 resume os ramos de Nível 0 e a forma como eles são integrados, em nível de probabilidade, pelos meta-aprendizes de Nível 1. Como se observa na figura, os ramos específicos por modalidade processam, de um lado, as cadeias de URL e, de outro, os atributos derivados da URL e do HTML. Os aprendizes de Nível 0 produzem estimativas de probabilidade fora da dobra para a partição de treinamento e estimativas alinhadas para a partição de teste isolada, que são então combinadas pelos meta-aprendizes de Nível 1. Essas saídas probabilísticas passam por uma etapa de calibração e convertem-se em predições binárias por meio de uma seleção de limiar conduzida sem qualquer acesso aos rótulos de teste.

O arcabouço organiza-se em três etapas sequenciais e interdependentes, detalhadas nas seções seguintes. A primeira delas constrói o conjunto de dados que torna possível analisar, para as mesmas instâncias, tanto as evidências extraídas da URL quanto as extraídas do HTML. A segunda define o esquema determinístico de características, que foi fixado antes de qualquer ajuste de modelo, justamente para que não dependesse da partição empregada. A terceira estabelece o protocolo de aprendizado empilhado que reúne os dois níveis de modelos. Sobre as três etapas incide um mesmo controle contra vazamento de informação, enunciado na Seção 4.5.5 e aplicado sem exceção em todos os procedimentos descritos adiante.

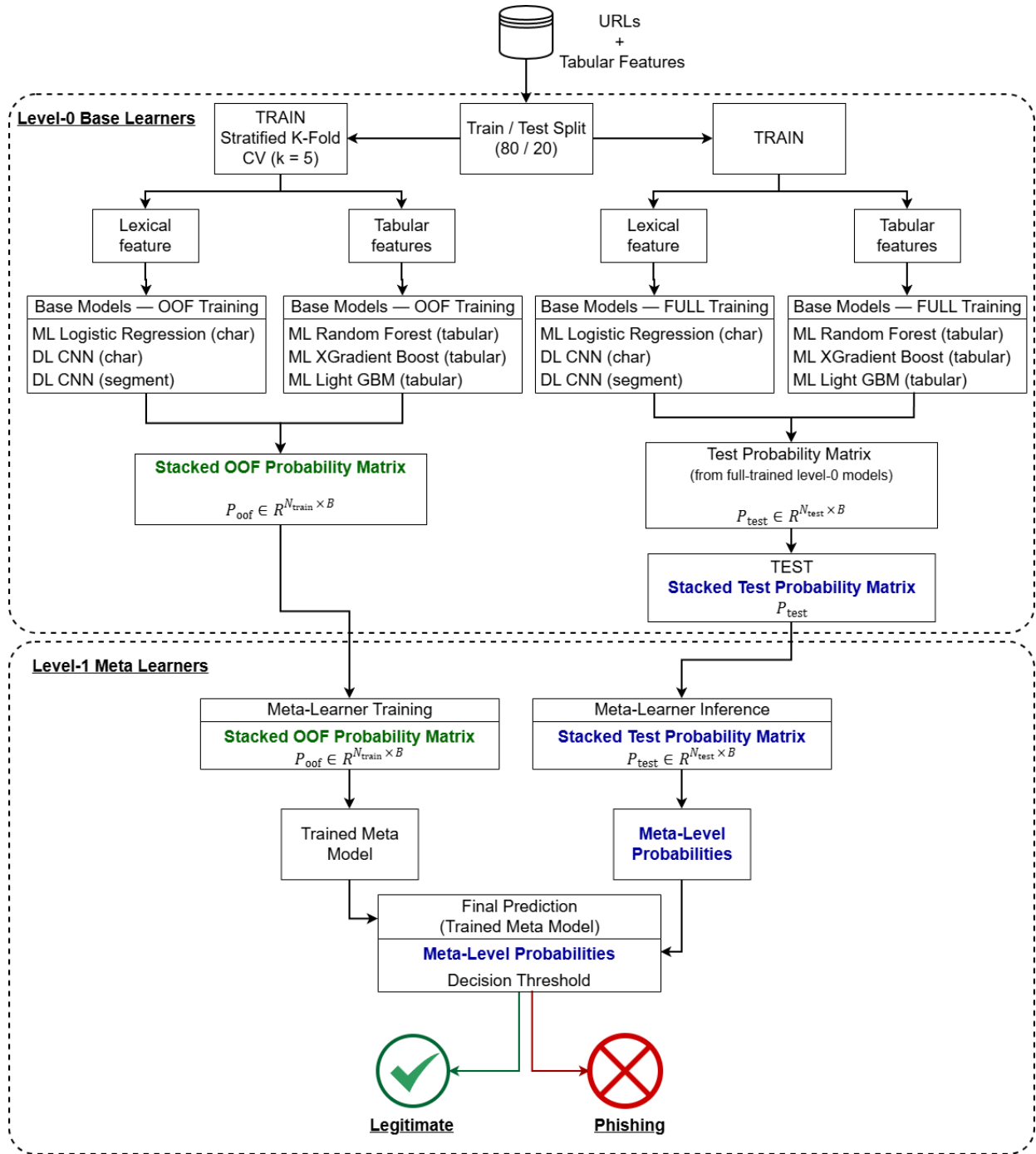


Figura 4.1 – Visão geral do arcabouço HOOF-Stack-PUD

4.3 CONSTRUÇÃO DO CONJUNTO HYBRIDPHISHER

Para que a complementaridade entre modalidades pudesse ser avaliada, era necessário que as evidências da URL e as do HTML estivessem disponíveis para as mesmas instâncias, pois, se parte das amostras dispusesse de apenas uma delas, a comparação entre as representações ficaria incompleta desde o início. Foi essa exigência que motivou a construção do HybridPhisher, conduzida segundo as recomendações de equilíbrio, de limpeza e de transparência do processo que Skula e Kvet (2024) discutem e que a Seção 2.7.1

apresenta. O procedimento adotado percorreu quatro etapas, a saber, coleta ativa, validação, consolidação e balanceamento, descritas nas subseções seguintes.

4.3.1 Corpus Legítimo

A construção do corpus legítimo partiu de domínios semente de alta reputação extraídos do ranking Tranco Top-1M, proposto por Le Pochat et al. (2019) como alternativa de pesquisa mais resistente à manipulação. Os autores demonstram que um adversário consegue alterar com facilidade a composição dos rankings de popularidade tradicionais, o que acaba por enviesar as conclusões dos estudos que se apoiam neles. A partir desses domínios, um coletor com múltiplas linhas de execução reuniu 887.782 URLs brutas. Como um número pequeno de sítios muito conectados poderia dominar o corpus, a coleta foi restringida por um limite máximo de URLs por domínio, o que resultou na cobertura de 61.855 domínios únicos.

Na etapa seguinte, foram descartadas as variantes de roteamento que impõem localidade, os recursos que não correspondiam a documentos HTML, as duplicatas excessivas em nível de domínio e as duplicatas exatas de URL, procedimento após o qual restaram 474.486 URLs legítimas candidatas.

Essas candidatas foram então submetidas a uma validação ativa leve, que combinou a resolução no sistema de nomes de domínio (*Domain Name System*, DNS), a inspeção do estado do protocolo de transferência de hipertexto (*Hypertext Transfer Protocol*, HTTP) e a verificação do tipo de conteúdo devolvido. Ao final dessa etapa, 257.204 URLs, o que corresponde a 54,2% do corpus legítimo candidato, mostraram-se acessíveis e devolveram conteúdo HTML válido.

Uma última inspeção percorreu os caminhos de URL decodificados em nível de caractere, com o objetivo de assegurar consistência de escrita entre as amostras. Os componentes de caminho que apresentavam escritas não latinas foram removidos do corpus, ao passo que os caracteres latinos com diacríticos foram mantidos. Essa etapa eliminou 2.880 URLs e encerrou o corpus legítimo validado em 254.324 URLs.

4.3.2 Corpus de Phishing

A base inicial de *phishing* veio do projeto `Phishing.Database` (Phishing Database Project, 2026), do qual foram ingeridas 782.295 URLs candidatas rotuladas como maliciosas. Cada uma dessas candidatas foi submetida à validação de alcançabilidade ativa e de conteúdo HTML, realizada sob restrições controladas de rede.

Dessas candidatas, apenas 60.878, ou seja, 7,78% do total ingerido, encontravam-se ativas e acessíveis em HTML no momento da coleta. Foram excluídas do corpus as URLs que devolveram erros HTTP permanentes, respostas de acesso negado, falhas de resolução no DNS, erros de SSL, conteúdo que não fosse HTML ou respostas repetidas de limitação de taxa. Para eliminar as duplicatas, o procedimento recorreu a um filtro de Bloom, escolha motivada pela necessidade de detectar repetições em um fluxo de centenas de milhares de URLs sem que fosse preciso manter toda a coleção em memória, e destinada sobretudo a impedir que uma mesma campanha ficasse sobrerrepresentada no corpus.

Convém registrar com precisão qual foi o estatuto dessa validação, uma vez que ela funcionou como critério de inclusão para a posterior extração de características multimodais, e não como uma reavaliação

independente dos rótulos que o repositório de origem já havia atribuído às amostras. O corpus resultante retrata, por essa razão, superfícies de ataque que estavam ativas no momento da coleta, e não artefatos já obsoletos.

As URLs retidas foram então reunidas a um instantâneo verificado do PhishTank de janeiro de 2026, composto por 49.759 URLs (Cisco Talos Intelligence Group, 2026), de modo que o corpus passasse a alcançar tanto infraestruturas transitórias quanto campanhas de duração mais longa. Após verificações de consistência em nível de fonte e uma nova deduplicação, restaram 110.637 URLs, número que uma etapa final de deduplicação global reduziu a 109.045 instâncias de *phishing* únicas.

4.3.3 Balanceamento e Consistência entre Classes

Para atenuar o desbalanceamento entre as classes, foi realizada uma amostragem sem reposição com semente 42, que selecionou 112.316 URLs legítimas e produziu um conjunto inicial de 221.361 instâncias, quase equilibrado, na proporção de 1,03:1 entre a classe legítima e a classe de *phishing*.

A verificação de consistência entre as classes identificou uma única divergência, referente à URL `https://goo.by/`, que aparecia com rótulos conflitantes. Diante desse caso, o critério adotado preservou a instância rotulada como *phishing* e descartou a duplicata legítima. O conjunto final reúne, portanto, 221.360 URLs, das quais 109.045 são instâncias de *phishing* e 112.315 são legítimas, o que corresponde a 49,26% e a 50,74% do total, respectivamente. Antes da extração de características, um embaralhamento global foi aplicado ao conjunto, a fim de eliminar eventuais efeitos de ordenação.

Toda instância retida no conjunto possui a cadeia bruta da URL e um instantâneo HTML armazenado. As 4.907 capturas sem sucesso, 81,8% delas de *phishing*, permaneceram no conjunto com atributos HTML zerados, efeito discutido no Apêndice D. É justamente essa propriedade que torna o conjunto adequado à comparação controlada entre representações lexicais, atributos tabulares e combinações heterogêneas, e é sob a designação D3 que ele integra o protocolo experimental descrito na Seção 4.6.

4.3.4 Critério de Inclusão de Páginas Ativas e Viés de Disponibilidade

O HybridPhisher não foi concebido como um instantâneo de lista de bloqueio construído apenas com URLs, mas como um conjunto capaz de sustentar experimentos nos quais os atributos derivados da URL e os derivados do HTML coexistam para as mesmas instâncias. Se URLs inativas ou respostas sem HTML recuperável fossem mantidas, o conjunto passaria a conter amostras apenas parcialmente observadas, o que comprometeria a comparabilidade entre as configurações léxica, tabular e empilhada. A proporção de 7,78% de candidatas ativas e acessíveis em HTML não mede a qualidade do conjunto, mas a volatilidade do próprio fenômeno. Páginas de *phishing* recorrem a infraestrutura efêmera (Duarte et al., 2025; Prasad; Chandra, 2024), e a lista de candidatas reúne endereços reportados em momentos distintos, muitos dos quais já não estavam ativos no momento da coleta. Sob verificação equivalente, a taxa do corpus legítimo foi de 54,2%. A mesma volatilidade reapareceu na captura do HTML, realizada depois da verificação de atividade: 3,7% das URLs de *phishing* já não responderam nessa etapa, contra 0,8% das legítimas. A principal restrição à construção de um conjunto multimodal de *phishing* está, portanto, na captura do

HTML enquanto a página permanece ativa, e não na obtenção das URLs. Nesse contexto, o HybridPhisher representa as campanhas cujo conteúdo pôde ser capturado no período da coleta, e o roteiro completo dessa coleta é apresentado no Apêndice D.

Esse critério, entretanto, introduz um viés que precisa ser declarado com clareza. Entre as candidatas rotuladas como *phishing* que acabaram descartadas podem estar campanhas genuínas que já haviam expirado, sido removidas, bloqueadas ou se tornado inacessíveis ao coletor, assim como respostas que não eram HTML, duplicatas e eventuais falsos positivos originados na própria fonte. Cabe destacar que nenhuma dessas amostras foi rerrotulada como legítima, pois elas apenas deixaram de compor o corpus experimental.

O HybridPhisher preserva, dessa maneira, os rótulos que as fontes originais atribuíram a todas as amostras retidas, ainda que carregue o viés de disponibilidade temporal que é inerente à coleta ativa de amostras de *phishing*. Esse viés corresponde à terceira ameaça à validade discutida na Seção 2.7.3 e é retomado entre as limitações apresentadas na Seção 6.1.

4.4 ENGENHARIA DE CARACTERÍSTICAS

Sobre as instâncias já validadas, a pesquisa derivou um esquema determinístico de atributos destinado a três usos distintos, que são alimentar os aprendizes tabulares, viabilizar a análise por blocos de características e permitir que os descritores compostos fossem avaliados isoladamente. A extração ocorreu somente depois de concluídas a consolidação das URLs, a verificação de consistência entre as classes, a deduplicação e o embaralhamento global do conjunto. As cadeias brutas de URL alimentam os modelos léxicos, ao passo que os 58 atributos numéricos e binários determinísticos alimentam tanto os aprendizes tabulares quanto os experimentos conduzidos por bloco de características.

Como o esquema foi fixado antes de qualquer ajuste de modelo, ele não depende da partição empregada, propriedade que torna possível definir os blocos F_{URL} , F_{HTML} e $F_{URL+HTML}$ utilizados nos experimentos complementares descritos na Seção 4.6.5.

A exposição segue a ordem em que os atributos surgiram: primeiro os derivados diretamente da URL, depois os derivados do HTML, em seguida os descritores compostos incorporados ao esquema e, por fim, a análise exploratória que fundamentou a regularização do conjunto.

4.4.1 Características Derivadas da URL

Os atributos derivados da URL capturam indícios estruturais e léxicos que costumam estar associados à atividade de *phishing*. Tomados isoladamente eles não são suficientes para a decisão, mas oferecem sinais precoces e de baixo custo, extraídos de forma determinística da cadeia bruta e sem qualquer consulta a serviços externos, o que os torna complementares às representações baseadas em conteúdo. A nomenclatura adotada procura ser transparente, pois cada nome designa diretamente a propriedade que o atributo mede, de maneira que a leitura do vetor de características não exija documentação auxiliar.

O comprimento aparece em `tld_length` e `url_length`; a estrutura do endereço, em `is_domain_`

`ip`, `num_subdomains` e `is_https`. As tentativas de ofuscação deixam rastro em `num_pct_encoded`, `num_obfuscated_char`, `has_obfuscation` e `has_punycode`, enquanto a composição de caracteres se traduz em `num_digits` e `SuspiciousCharRatio`. A cadeia de consulta contribui com `num_question_marks`, `num_equals`, `num_ampersands` e `num_percent`. Por fim, `shannon_entropy` e `compression_ratio` resumem estatísticas em nível de cadeia.

Reunidas, essas variáveis descrevem o comprimento da URL, a estrutura do *host*, o conteúdo codificado, a inflação da cadeia de consulta, a composição incomum de caracteres e a irregularidade léxica, que correspondem justamente às manifestações estruturais descritas na Seção 2.2.

4.4.2 Características Derivadas do HTML

Os atributos de HTML complementam os de URL porque modelam propriedades estruturais e comportamentais da página que o usuário efetivamente recebe. A motivação para incluí-los é simples de enunciar, uma vez que uma página de *phishing* pode ser irrepreensível no nível da URL, e nesses casos a evidência discriminativa só se torna disponível no conteúdo. É no HTML que se manifestam a coleta de credenciais, a manipulação do leiaute e os padrões de interação que caracterizam a fraude.

A extração operou sobre os instantâneos de página que haviam sido armazenados durante a validação ativa. A captura ocorreu sob condições controladas, com limite de tamanho de resposta e identificação determinística da requisição, e apenas os documentos HTML efetivamente recuperados e processáveis seguiram para a etapa seguinte. A escolha de trabalhar a partir de instantâneos, e não de novas requisições, apoia-se em duas razões. A primeira delas diz respeito à reprodutibilidade, já que o mesmo conteúdo de página serve a experimentos sucessivos e nem as condições de rede nem a disponibilidade da página alteram a entrada durante o treinamento. A segunda razão diz respeito à separação entre o custo de coleta e o custo de inferência, uma vez que o *download* do HTML pode consumir parcela dominante do tempo total de detecção, conforme quantifica a Seção 2.4.3.

A nomenclatura desses atributos obedece a uma convenção uniforme, adotada para que o nome de cada variável descreva a evidência que ela mede. O prefixo `Has` identifica os indicadores binários de presença, como ocorre em `HasTitle`, `HasDescription`, `HasFavicon` e `HasCopyrightInfo`, ao passo que o prefixo `NoOf` identifica as contagens de elementos, como em `NoOfJS`, `NoOfiFrame` e `NoOfExternalRef`. Graças a essa convenção, tanto a leitura do vetor de características quanto a das análises de importância apresentadas na Seção 4.4.7 permitem distinguir de imediato o que é ocorrência do que é intensidade.

O conjunto de indicadores de metadados merece uma explicação à parte, por ser o menos evidente entre os grupos considerados. Páginas de *phishing* costumam ser criadas às pressas, permanecer no ar por período curto e resultar de cópias parciais de sítios legítimos. Nesse processo, os elementos que não afetam a aparência imediata da página tendem a ser os primeiros a ser omitidos, entre os quais um título coerente, a metatag de descrição, o ícone do sítio e o aviso de direitos autorais no rodapé. Cada um desses elementos seria pouco custoso de reproduzir isoladamente, mas a ausência conjunta deles indica a economia de esforço que é típica de uma página descartável. Foi essa observação que levou à incorporação de `HasTitle`, `HasDescription`, `HasFavicon` e `HasCopyrightInfo` ao esquema como indicado-

res binários independentes, em vez de um único indicador agregado de completude dos metadados.

A Tabela 4.1 organiza os atributos derivados do HTML por grupo funcional e registra a evidência que cada grupo pretende capturar.

Tabela 4.1 – Atributos derivados do HTML, por grupo funcional

Grupo funcional	Atributos	Evidência capturada
Estrutura do documento	LineOfCode, LargestLineLength	Volume e formatação do código; linhas muito extensas indicam conteúdo minificado, ofuscado ou gerado automaticamente.
Metadados	HasTitle, HasDescription, HasFavicon, HasCopyrightInfo	Completude dos elementos de identidade da página; sua ausência conjunta indica reprodução apressada de um sítio legítimo.
Formulários e credenciais	HasHiddenFields, HasPasswordField, HasSubmitButton, HasExternalFormSubmit	Intenção de coleta de dados sensíveis e envio do formulário para domínio distinto do exibido.
Interação e incorporação	NoOfPopup, NoOfiFrame	Mecanismos de sobreposição e de incorporação de conteúdo externo, usados para simular telas de autenticação.
Composição de recursos	NoOfJS, NoOfCSS, NoOfImage	Perfil de carregamento da página; páginas fraudulentas tendem a apresentar composição de recursos atípica.
Distribuição de ligações	NoOfExternalRef, NoOfSelfRef, NoOfEmptyRef, NoOfURLRedirect, NoOfSelfRedirect	Incoerência entre a identidade apresentada e a infraestrutura utilizada; predomínio de referências externas ou vazias.
Palavras-chave financeiras	Bank, Pay, Crypto	Presença de termos associados a serviços financeiros, alvos preferenciais das campanhas.

Em conjunto, esses atributos caracterizam a estrutura da página, a intenção de interação e o comportamento de carregamento de recursos, de modo que acrescentam ao esquema sinais que os padrões léxicos de superfície não seriam capazes de alcançar.

4.4.3 Escores de Alinhamento entre Título e URL

Dois atributos foram construídos para medir a consistência entre o título do documento HTML e a identidade que a URL anuncia. Ambos procuram evidenciar a incoerência entre a identidade apresentada ao usuário e a infraestrutura que é efetivamente utilizada, situação que corresponde a uma das modalidades de evidência relacionadas na Tabela 2.1.

O `URLTitleMatchScore` (UTMS) quantifica o alinhamento léxico entre o título da página e uma representação normalizada do domínio raiz. Essa normalização remove o prefixo `www` quando ele está presente, descarta o rótulo final do *host*, que é tratado como domínio de topo, concatena os rótulos remanescentes, elimina os hifens e converte para minúsculas a cadeia resultante. Seja *r* a representação obtida ao final desse procedimento.

O título HTML, por sua vez, é dividido em *tokens* delimitados por espaço em branco, sendo cada um deles convertido para minúsculas e reduzido a caracteres alfanuméricos. Sempre que um *token* normalizado

t ocorre como subcadeia da representação remanescente do domínio raiz, o escore é acrescido de

$$\text{score}(t, r) = \frac{100}{|r|} |t|. \quad (4.1)$$

A cada correspondência encontrada, o procedimento retira do domínio raiz a subcadeia que já foi contabilizada, o que impede que os mesmos caracteres sejam creditados duas vezes. O resultado acumulado permanece limitado ao intervalo $[0, 100]$.

O `DomainTitleMatchScore` (DTMS) repete o mesmo procedimento de normalização de *tokens* e de correspondência ponderada por comprimento, mas toma como referência o rótulo de domínio primário, em vez da representação completa do domínio raiz. Depois de removido o prefixo `www`, o procedimento seleciona o penúltimo componente do *host* sempre que existirem ao menos dois componentes e, caso contrário, utiliza o único componente disponível. O rótulo escolhido é convertido para minúsculas e reduzido a caracteres alfanuméricos, as subcadeias correspondentes são retiradas da cadeia e o escore final permanece limitado ao intervalo $[0, 100]$.

O UTMS mede, portanto, a concordância entre o título e a representação mais ampla do domínio raiz, ao passo que o DTMS se concentra especificamente no rótulo de domínio primário. Em ambos os casos, valores mais altos indicam maior consistência léxica entre o título da página e a identidade anunciada pela URL.

4.4.4 Domain Similarity

O descritor `Domain_Similarity` foi construído para capturar desvios léxicos ou visualmente induzidos em nomes de domínio, categoria de manipulação descrita na Seção 2.2. Ele parte da intuição do índice de similaridade de URL (*URL Similarity Index*, USI) proposto por Prasad e Chandra (2024). Como os autores não especificam os detalhes de implementação do índice original em grau suficiente para que ele possa ser replicado diretamente, esta pesquisa adotou um esquema de correspondência de caracteres ponderada por posição, que preserva os princípios de alinhamento e de ponderação do USI, mas o converte em uma representação contínua no intervalo $[0, 100]$.

Sejam d_{ref} um domínio legítimo de referência e d_{obs} o domínio observado, e seja

$$N = \max(|d_{\text{ref}}|, |d_{\text{obs}}|). \quad (4.2)$$

Cada posição de caractere $i \in \{0, \dots, N - 1\}$ recebe um peso dependente da posição, dado por

$$w(i; N) = \frac{50}{N} + \frac{100(N - i)}{N(N + 1)}, \quad (4.3)$$

que soma uma componente constante a uma ênfase linearmente decrescente da esquerda para a direita. Essa escolha responde a uma propriedade perceptual, pois os caracteres iniciais de um domínio concentram a maior parte da informação que o usuário mobiliza para reconhecê-lo, razão pela qual uma substituição no começo da cadeia deve pesar mais do que uma alteração equivalente ocorrida no fim.

O procedimento alinha os domínios de forma sequencial, e cada correspondência exata verificada na posição i acrescenta $w(i; N)$ ao escore acumulado. Quando há diferença de comprimento entre as cadeias, o alinhamento remove caracteres da cadeia mais longa segundo um critério determinístico, ao passo que, nos demais casos, as divergências são penalizadas sem remoção. Sendo $\mathcal{M}$ o conjunto dos índices correspondentes, o escore é dado por

$$\text{Domain_Similarity}(d_{\text{ref}}, d_{\text{obs}}) = \sum_{i \in \mathcal{M}} w(i; N), \quad (4.4)$$

normalizado ao intervalo $[0, 100]$.

A Equação (4.4) compara apenas duas cadeias entre si. O descritor que foi efetivamente incorporado ao conjunto, no entanto, responde a uma pergunta diferente, que é saber quão próximo o domínio observado chega do domínio popular mais parecido com ele. Por essa razão, o valor atribuído a cada instância corresponde ao máximo da similaridade calculada sobre um conjunto de referência $\mathcal{R}$ formado por domínios legítimos, de acordo com

$$\text{Domain_Similarity}(d_{\text{obs}}) = \max_{d \in \mathcal{R}} \text{SI}(d, d_{\text{obs}}), \quad (4.5)$$

em que SI denota o escore definido pela Equação (4.4). Quando d_{obs} já pertence a $\mathcal{R}$, o descritor recebe diretamente o valor 100. É essa formulação que confere sentido operacional ao atributo, uma vez que um domínio de *phishing* construído para imitar uma marca conhecida obtém escore alto justamente porque existe, em algum ponto do conjunto de referência, o domínio que ele procura imitar.

O conjunto $\mathcal{R}$ adotado é a lista *Top 10 Million Domains*, publicada pelo projeto Open PageRank (Open PageRank, 2026), da qual se utiliza somente a coluna de domínios. Essa lista distingue-se da Tranco Top-1M empregada na Seção 4.3.1, e a diferença entre as duas é de finalidade, pois a primeira serve apenas como universo de comparação para o cálculo da similaridade, enquanto a segunda foi utilizada para selecionar os domínios semente da coleta do corpus legítimo. Como o objetivo aqui é localizar a marca que está sendo imitada, uma lista mais extensa amplia a chance de que ela esteja de fato presente.

Antes da comparação, cada *host* passa por uma normalização destinada a neutralizar artifícios de ocultação, que compreende a decodificação de sequências percentuais, a remoção do prefixo `www`, a conversão para minúsculas, a decodificação *punycode* e a normalização Unicode na forma NFKC, esta última responsável por aproximar os caracteres homógrafos de sua forma canônica. Optou-se por não reduzir o *host* ao domínio registrável, decisão que evita atribuir similaridade máxima a páginas hospedadas em plataformas compartilhadas, cujo sufixo coincide com o de um serviço legítimo.

Comparar cada URL contra dez milhões de domínios seria computacionalmente inviável, razão pela qual a busca opera sobre um índice de candidatos previamente construído. Nesse índice, os domínios de referência são agrupados por domínio de topo, por prefixo de cinco caracteres e por faixa de comprimento, de modo que a consulta recupere os grupos correspondentes e também os vizinhos imediatos, recorrendo a uma busca independente do domínio de topo sempre que os grupos iniciais não devolvem candidatos. Para tratar as URLs que apresentam subdomínios injetados, a consulta utiliza ainda variantes do *host* obtidas

pela remoção de até dois rótulos à esquerda, embora o escore seja sempre calculado sobre o *host* original. O custo do procedimento é contido por limites por grupo e por um teto global de candidatos, e a busca é interrompida tão logo algum candidato atinja o valor máximo.

Cabe destacar uma decisão metodológica que diferencia o uso feito aqui daquele proposto originalmente. No arcabouço PhiUSIIL, o USI opera como um pré-filtro baseado em limiar, ao passo que, nesta pesquisa, o escore não determina isoladamente qualquer decisão de classificação. Ele foi mantido como um atributo submetido ao aprendizado supervisionado, para que os modelos pudessem aprender de que forma ele interage com as demais evidências de URL e de HTML. É essa opção que torna possível estimar a contribuição do descritor por meio de comparações pareadas com e sem o atributo, conforme descreve a Seção 4.6.5.

Os valores numéricos obtidos não reproduzem exatamente os de Prasad e Chandra (2024), mas a aproximação adotada conserva a sensibilidade relativa a substituições homográficas, a omissões e a transposições de caracteres. A Tabela 4.2 ilustra esse comportamento a partir de um único domínio de referência, e a coluna DS registra o valor assumido pelo descritor em cada caso. A ordenação decrescente dos escores confirma o efeito que a ponderação posicional pretendia produzir, uma vez que as transposições próximas ao início da cadeia degradam o escore mais do que as substituições ou omissões ocorridas em posições posteriores.

Tabela 4.2 – Escores representativos de similaridade de domínio para variantes visualmente manipuladas

Domínio de referência	Domínio testado	DS (%)	Tipo de modificação
example.com	example.com	100,00	Domínio idêntico
example.com	example.com	91,67	Substituição homográfica ('e' com ponto inferior)
example.com	example.com	90,15	Substituição homográfica ('p' com ponto superior)
example.com	example.com	88,64	Substituição homográfica ('a' com ponto inferior)
example.com	example.com	88,64	Substituição homográfica ('a' com ponto superior)
example.com	exampl.com	86,36	Omissão de caractere ('e')
example.com	exapml.com	79,55	Transposição de caracteres ('m'–'p')
example.com	eaxmple.com	76,52	Transposição de caracteres ('a'–'x')

4.4.5 URL Complexity Score

O *URL Complexity Score* (URLC) é um descritor composto e determinístico que resume, por meio de componentes interpretáveis, as irregularidades estruturais presentes em uma URL. Sua formulação dialoga com trabalhos anteriores sobre complexidade de URL e sobre atributos baseados em entropia (Prasad et al., 2026), dos quais se afasta ao modelar de forma explícita quatro anomalias nomeadas, que são a mistura alfanumérica de *tokens*, a dominância do maior *token*, a concentração de subdomínios e a complexidade da cadeia de consulta. Convém esclarecer que o URLC não atua como detector autônomo nem substitui os atributos originais da URL, pois ele foi incorporado ao esquema como mais um atributo tabular, cuja função é condensar essas quatro formas complementares de complexidade em uma única dimensão contínua.

Dada uma URL *u*, a implementação a decompõe segundo a especificação *Request for Comments* (RFC) e extrai dela os componentes *host*, caminho e consulta. Os *tokens* de *host* resultam da divisão do *host* pelo separador ".", ao passo que os *tokens* de caminho resultam de uma divisão em duas etapas, realizada primeiro pelo separador "/" e, em seguida, dentro de cada segmento não vazio, pelos separadores "-", "_"

".". Os *tokens* vazios são descartados. Seja T a união dos rótulos de *host* com os *tokens* de caminho.

O primeiro componente mede a mistura alfanumérica:

$$S_{\text{mix}} = \frac{1}{|T|} \sum_{t \in T} \mathbb{I}[\text{hasAlpha}(t) \wedge \text{hasDigit}(t)], \quad (4.6)$$

com $S_{\text{mix}} = 0$ quando T é vazio. Esse componente identifica os *tokens* que combinam letras e dígitos, padrão que é comum em fragmentos de URL sintéticos, ofuscados ou gerados automaticamente por campanhas.

O segundo componente mede a dominância do maior *token*:

$$S_{\text{maxtok}} = \frac{\max_{t \in T} |t|}{\max \left(\sum_{t \in T} |t|, 1 \right)}, \quad (4.7)$$

com $S_{\text{maxtok}} = 0$ quando T é vazio. Esse termo permite verificar se um único *token* de *host* ou de caminho responde por uma fração desproporcional de todo o conteúdo tokenizado da URL.

O terceiro componente mede a concentração de subdomínios. Sendo H a sequência ordenada dos rótulos de *host*, adota-se como conjunto de subdomínios, por critério pragmático, aquele formado pelos rótulos que precedem os dois últimos, isto é,

$$\text{Sub}(H) = \begin{cases} H_{1:|H|-2}, & |H| > 2, \\ \emptyset, & |H| \leq 2. \end{cases} \quad (4.8)$$

Então,

$$S_{\text{subdom}} = \frac{\max_{s \in \text{Sub}(H)} |s|}{\max \left(\sum_{s \in \text{Sub}(H)} |s|, 1 \right)}. \quad (4.9)$$

Na ausência de subdomínio relevante, S_{subdom} assume o valor zero. Esse componente detecta os efeitos de concentração que ocorrem na porção de subdomínio do *host*, frequentes nas URLs que recorrem a prefixos longos ou irregulares.

O quarto componente mede a complexidade da cadeia de consulta. Nessa etapa, os parâmetros são lidos como pares de chave e valor, e os valores em branco são preservados. Sendo p o número de parâmetros e r_p a razão entre o número de chaves únicas e p , tem-se, quando $p > 0$,

$$S_{\text{query}} = \left(\frac{\log(1+p)}{\log(1+P_{\text{max}})} \right) \cdot (0.5 + 0.5r_p), \quad (4.10)$$

com $P_{\text{max}} = 20$ e escore limitado ao intervalo $[0, 1]$. Quando não há parâmetro de consulta, $S_{\text{query}} = 0$. A formulação combina o volume de parâmetros com a diversidade das chaves e recorre a um escalonamento logarítmico, cuja finalidade é amortecer o efeito produzido por contagens extremas.

O escore URLC final resulta da combinação ponderada dos quatro componentes:

$$\text{URLC}(u) = 100 \times (0.40 S_{\text{mix}} + 0.25 S_{\text{maxtok}} + 0.25 S_{\text{subdom}} + 0.10 S_{\text{query}}), \quad (4.11)$$

com o valor final limitado ao intervalo $[0, 100]$.

Os pesos somam a unidade e definem uma agregação heurística compacta dos quatro componentes. Eles não pretendem constituir uma parametrização globalmente ótima, uma vez que seu papel é apenas o de oferecer um resumo interpretável da complexidade estrutural da URL. A contribuição empírica do URLC é aferida separadamente, por meio da análise de sensibilidade dos pesos e de experimentos pareados com e sem o atributo, ambos descritos na Seção 4.6.5.

4.4.6 Análise Exploratória e Regularização das Características

Concluída a extração, uma análise exploratória sistemática examinou o comportamento estatístico dos atributos numéricos, à procura de propriedades distribucionais capazes de prejudicar o treinamento. A análise apoiou-se em um relatório de dispersão que avaliou cada característica por estatísticas robustas de tendência central, dispersão, assimetria e comportamento de cauda.

O diagnóstico revelou que parcela substancial dos atributos numéricos exibe distribuições esparsas e assimétricas. Muitas variáveis de contagem concentram massa dominante em zero e alongam-se em caudas à direita, produzidas por um número reduzido de observações extremas. O padrão marca sobretudo os atributos ligados à composição da cadeia de consulta, aos mecanismos de interação do HTML e aos indicadores de ofuscação.

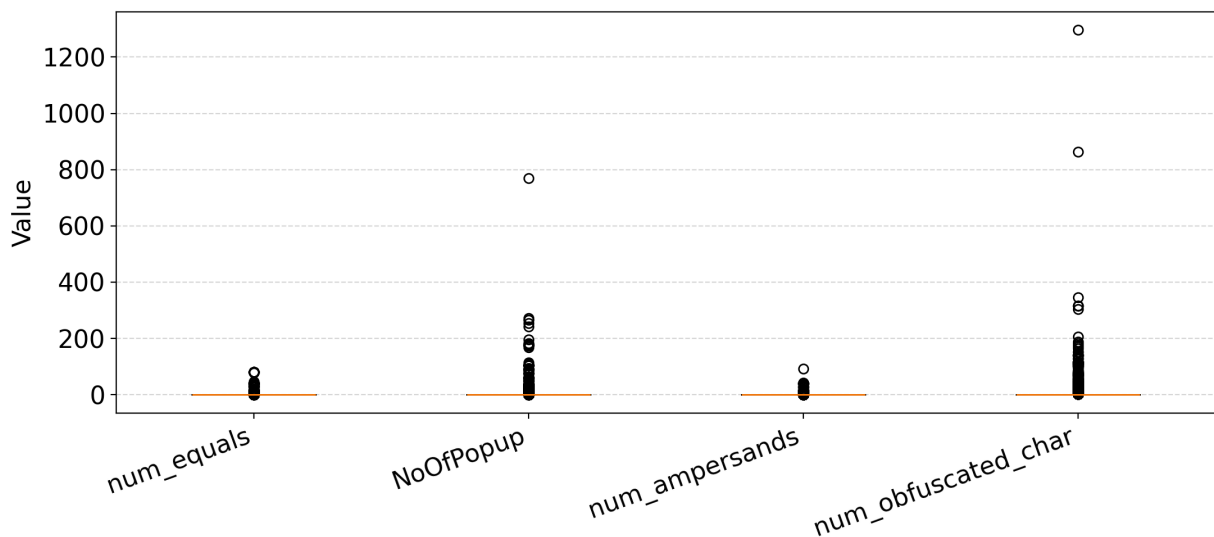


Figura 4.2 – Diagramas de caixa agrupados das características numéricas mais assimétricas identificadas na análise exploratória

A Figura 4.2 torna esse comportamento visível. Todas as variáveis ali representadas são infladas em zero, com mediana e amplitude interquartil nulas, e ainda assim atingem máximos expressivos. Atributos como `num_equals` e `num_ampersands` chegam a 80 e 92 ocorrências, respectivamente, o que reflete

cadeias de consulta extensas. O atributo `NoOfPopup` dispersa-se ainda mais: uma fração pequena de páginas ultrapassa 700 elementos de sobreposição. Já `num_obfuscated_char`, embora menos frequente, tem a cauda mais acentuada de todas, e supera mil caracteres em aproximadamente 1% das amostras.

A combinação de inflação em zero, caudas pesadas e assimetria elevada é problemática por três razões: distorce medidas baseadas em distância, enviesada fronteiras de decisão lineares e desestabiliza a otimização por gradiente. Indicadores binários e atributos de razão limitada, ao contrário, comportam-se bem e dispensam correção.

Esse diagnóstico conduziu à adoção de uma estratégia de regularização diferenciada por tipo de variável, em lugar de uma transformação uniforme aplicada a todo o espaço de características. Os indicadores binários e as características de razão limitada permaneceram inalterados. As variáveis de contagem com cauda longa foram submetidas a um escalonamento logarítmico monotônico, que reduz os valores extremos sem perturbar as relações de ordem entre as observações. Já as variáveis infladas em zero deram origem a duas componentes distintas, sendo a primeira um indicador binário de presença, identificado pelo prefixo `has_`, e a segunda uma componente de magnitude regularizada, que conserva o nome original do atributo. Esse desdobramento permite que o efeito de ocorrência e o efeito de intensidade sejam modelados separadamente.

A Tabela 4.3 registra a correspondência entre os atributos originais e as representações que passaram a compor o conjunto final depois da regularização. Duas situações distintas aparecem nessa tabela, pois, nas variáveis infladas em zero, uma coluna adicional `has_*` acompanha o nome original, que passa a carregar os valores já regularizados, enquanto nos demais atributos atua somente a compressão logarítmica, sem que haja desdobramento.

Tabela 4.3 – Mapeamento de atributos produzido pela regularização

Atributo original	Atributo transformado
<code>num_equals</code>	<code>has_num_equals, num_equals</code>
<code>num_percent</code>	<code>has_num_percent, num_percent</code>
<code>num_ampersands</code>	<code>has_num_ampersands, num_ampersands</code>
<code>num_obfuscated_char</code>	<code>has_num_obfuscated_char, num_obfuscated_char</code>
<code>num_pct_encoded</code>	<code>has_num_pct_encoded, num_pct_encoded</code>
<code>NoOfPopup</code>	<code>has_NoOfPopup, NoOfPopup</code>
<code>LargestLineLength</code>	<code>LargestLineLength</code>
<code>NoOfSelfRef</code>	<code>NoOfSelfRef</code>
<code>NoOfExternalRef</code>	<code>NoOfExternalRef</code>
<code>NoOfImage</code>	<code>NoOfImage</code>
<code>NoOfURLRedirect</code>	<code>NoOfURLRedirect</code>
<code>LineOfCode</code>	<code>LineOfCode</code>
<code>NoOfCSS</code>	<code>NoOfCSS</code>
<code>NoOfJS</code>	<code>NoOfJS</code>
<code>url_length</code>	<code>url_length</code>
<code>NoOfSelfRedirect</code>	<code>NoOfSelfRedirect</code>
<code>NoOfiFrame</code>	<code>NoOfiFrame</code>
<code>NoOfEmptyRef</code>	<code>NoOfEmptyRef</code>

4.4.7 Estrutura Discriminativa do Espaço de Características

Como a regularização descrita na subseção anterior altera a escala das variáveis, tornou-se necessário verificar se ela preserva a informação discriminativa originalmente disponível. Três análises complementares foram conduzidas para responder a essa pergunta, todas realizadas sobre o conjunto já transformado e antes de qualquer decisão de modelagem.

A primeira delas examina a influência de cada atributo por classe, apresentada na Figura 4.3, na qual os coeficientes positivos indicam associação com a classe de *phishing*, os valores negativos correspondem às URLs legítimas e a magnitude reflete o impacto relativo sobre a fronteira de decisão. Mesmo depois da compressão aplicada, os atributos ligados à complexidade da consulta e à intensidade de ofuscação continuam a exibir valores normalizados mais altos para a classe de *phishing*. Conclui-se, portanto, que o poder discriminativo não decorre apenas de magnitudes extremas, uma vez que tanto a ocorrência quanto a intensidade relativa contribuem para separar as classes.

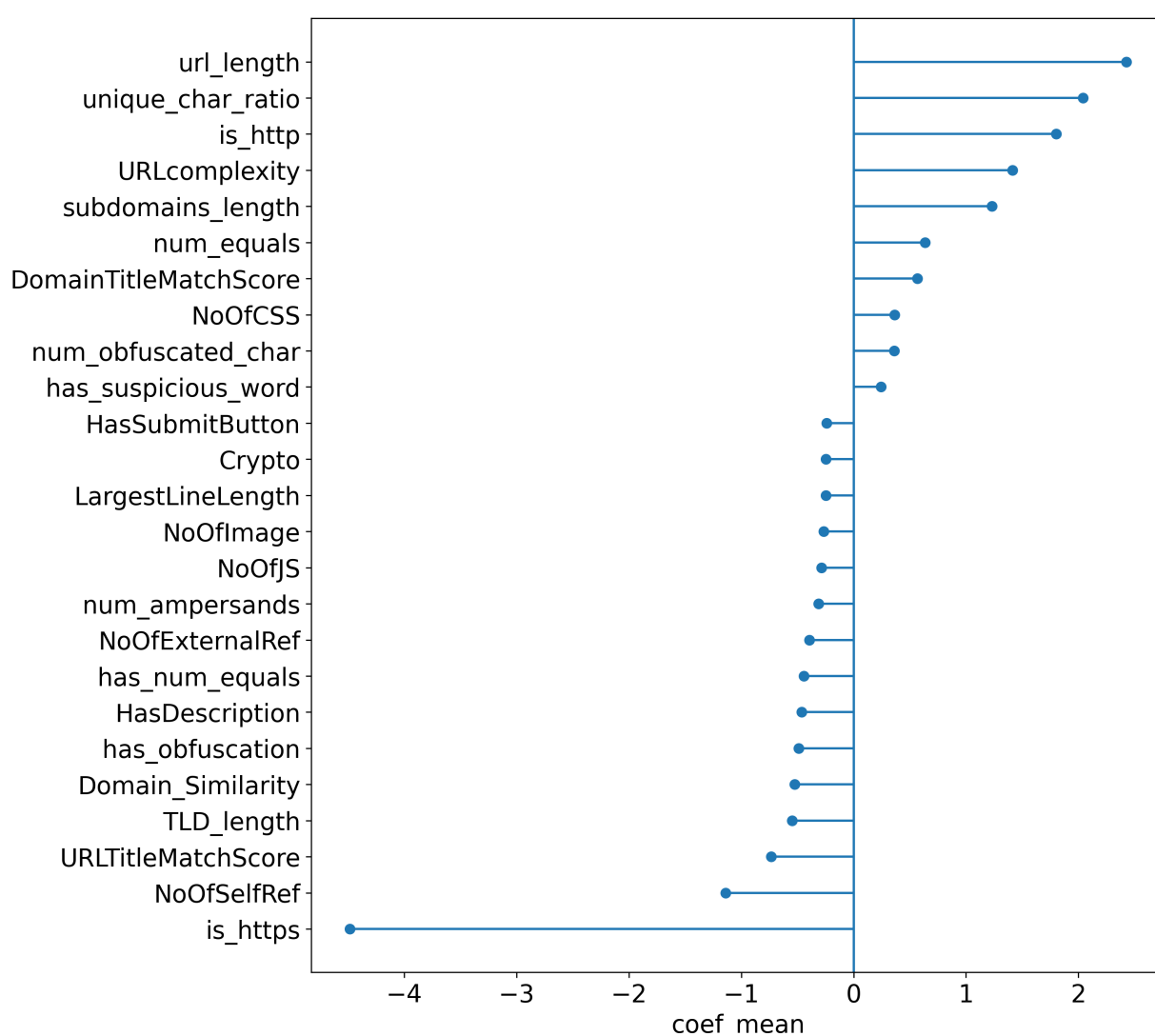


Figura 4.3 – Influência das características por classe, derivada dos modelos treinados

A segunda análise ordena os atributos pela estatística qui-quadrado calculada após a normalização,

conforme mostra a Figura 4.4, na qual os valores mais altos indicam associação mais forte com o rótulo de *phishing*. No topo dessa ordenação permanecem os atributos de alinhamento semântico entre a URL e o título da página, isto é, o UTMS e o DTMS, seguidos de perto pelos descritores estruturais apresentados nas Seções 4.4.4 e 4.4.5, o `Domain_Similarity` e o `URLC`. O que confere peso a esse resultado é o momento em que a medição foi realizada, pois, como a ordenação recai sobre valores já regularizados, a relevância desses atributos não pode ser atribuída à presença de valores brutos extremos.

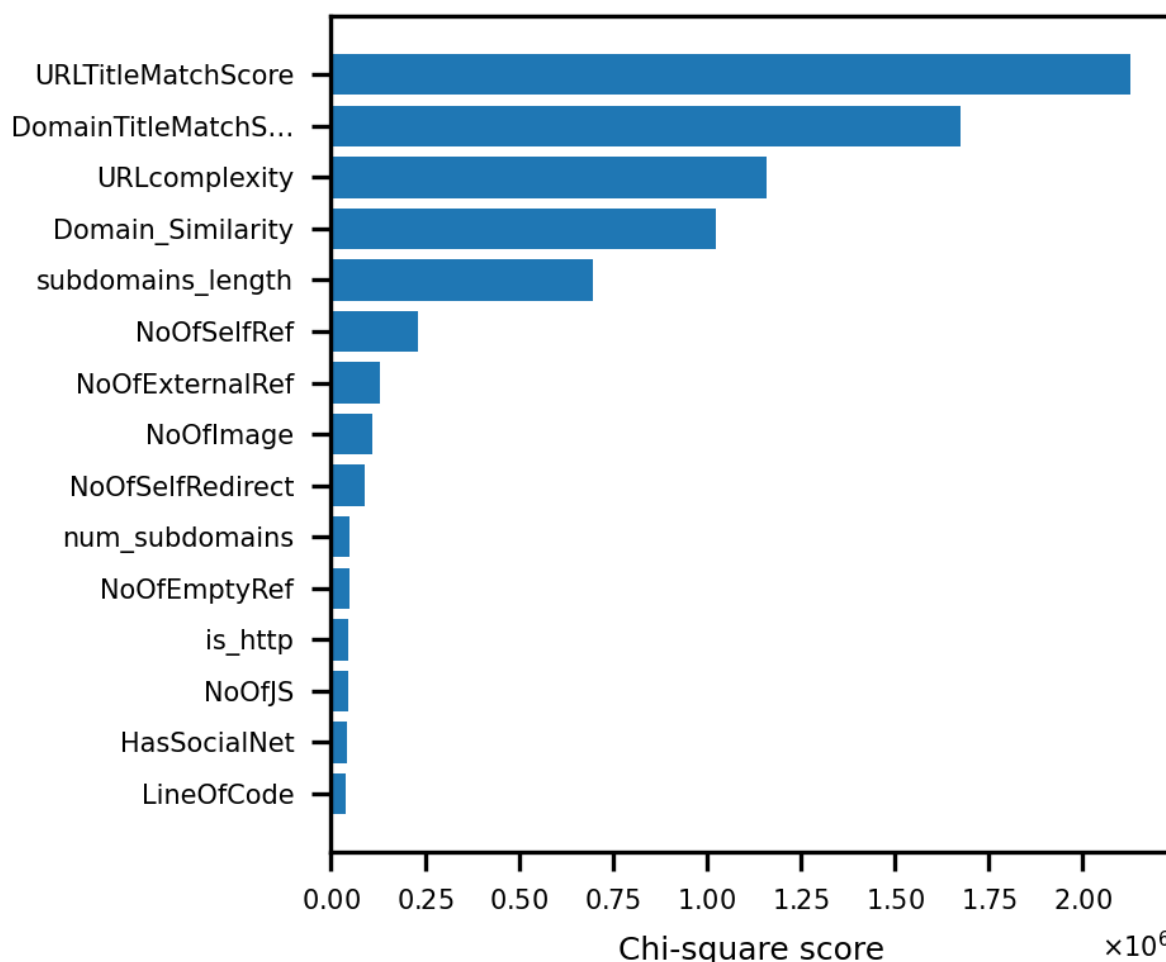


Figura 4.4 – Atributos tabulares mais bem classificados pela estatística qui-quadrado após a regularização

A terceira análise observa as distribuições conjuntas dos atributos mais bem classificados, estratificadas por classe, conforme a Figura 4.5. Os atributos UTMS, DTMS, URLC e `Domain_Similarity` deslocam-se de forma sistemática entre as classes e chegam mesmo a separá-las parcialmente, com dominância reduzida de valores extremos, o que evidencia que a normalização conteve os *outliers* sem desmontar a estrutura discriminativa. As características de interação derivadas do HTML mantêm deslocamentos igualmente distintos, o que indica que a regularização preservou também os sinais de natureza comportamental.

As três análises convergem para a mesma conclusão, segundo a qual a etapa de transformação produz um espaço de características bem condicionado e interpretável, compatível com as premissas tanto dos

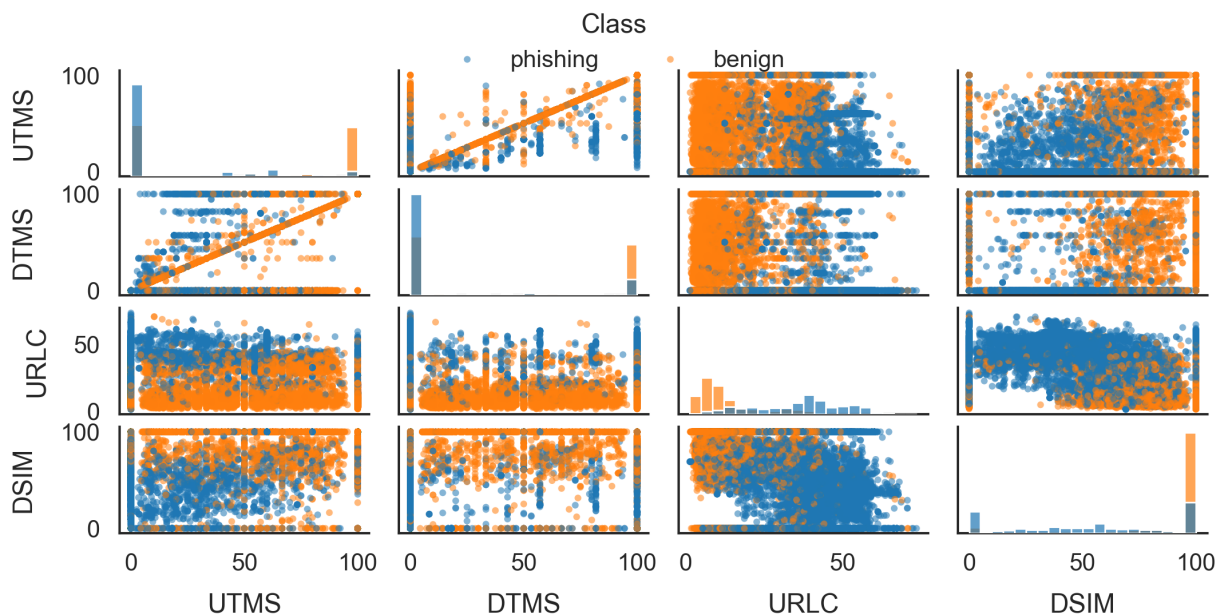


Figura 4.5 – Distribuições pareadas dos atributos tabulares mais bem classificados após a regularização, estratificadas por classe

modelos lineares quanto dos modelos não lineares. Cabe, no entanto, registrar uma delimitação importante, pois se trata de análises descritivas do conjunto construído, que serviram para validar a engenharia de características. Elas não substituem, portanto, a avaliação por blocos de atributos conduzida sob protocolo experimental controlado, que é apresentada na Seção 5.6.

4.5 PROTOCOLO DE APRENDIZADO EMPILHADO

O protocolo de aprendizado empilhado foi construído para responder a uma pergunta específica, que consiste em saber se preditores de base heterogêneos fornecem, em nível de probabilidade, evidência complementar àquela que aprendizes isolados fortes já oferecem. Para que a resposta a essa pergunta tenha valor, é preciso satisfazer uma condição necessária, segundo a qual qualquer ganho observado deve ser atribuível à combinação dos modelos, e não a vazamento de informação proveniente da partição de teste. É essa exigência que orienta cada uma das decisões descritas nas subseções seguintes.

4.5.1 Particionamento e Pré-processamento Ajustado no Treinamento

Para cada conjunto de dados e para cada semente, uma divisão estratificada separa as partições de treinamento e de teste na proporção de 80 para 20. A partição de teste permanece fora de todas as etapas que se seguem, entre as quais o pré-processamento, a seleção de características, o ajuste dos modelos de base, o treinamento dos meta-aprendizes, a calibração, a seleção de limiar e a seleção de modelos.

O pré-processamento tabular é ajustado exclusivamente sobre a partição de treinamento. Nele, as características candidatas passam por normalização Min-Max, por filtragem segundo um limiar de variância

de 10^{-5} e por ordenação obtida a partir da seleção qui-quadrado. Quando o parâmetro k não é especificado, permanecem no conjunto até 128 características depois da filtragem. Somente após a conclusão dessas etapas é que as transformações ajustadas e a ordenação delas resultante são aplicadas à partição de teste isolada, sem que sofram qualquer alteração.

4.5.2 Preditores de Nível 0

A camada de Nível 0 é composta por seis preditores, organizados em ramos segundo a modalidade de evidência que cada um processa. Essa organização operacionaliza o princípio discutido na Seção 2.6.1, segundo o qual a diversidade útil a um comitê decorre da modalidade de evidência e da representação empregadas, e não da simples pluralidade de algoritmos reunidos.

O ramo tabular reúne o *Random Forest* (RF_t) (Breiman, 2001), o *XGBoost* (XGB_t) (Chen; Guestrin, 2016) e o *LightGBM* ($LGBM_t$) (Ke et al., 2017), todos treinados sobre as características numéricas selecionadas de URL e de HTML. A reunião desses três algoritmos justifica-se porque eles diferem entre si no funcionamento interno, já que Breiman (2001) agrega os modelos por *bagging* com amostragem aleatória de atributos, ao passo que Chen e Guestrin (2016) e Ke et al. (2017) implementam *gradient boosting* com estratégias distintas de construção de árvores e de tratamento da esparsidade. A evidência de desempenho consolidada dessa família de modelos é discutida na Seção 2.5.1.

O ramo léxico, por sua vez, reúne uma regressão logística sobre n-gramas de caractere (LR_c), uma rede neural convolucional em nível de caractere (CNN_c) e uma rede convolucional em nível de *token* ou segmento (CNN_w), todas alimentadas diretamente pelas cadeias brutas de URL. A representação por n-gramas de caractere apoia-se nos fundamentos de recuperação da informação sistematizados por Manning, Raghavan e Schütze (2008), enquanto as arquiteturas convolucionais seguem o paradigma de aprendizado fim a fim sobre a cadeia da URL, estabelecido por Le et al. (2018) e discutido na Seção 2.5.2.

Esses modelos capturam, respectivamente, padrões não lineares entre os atributos projetados, padrões léxicos esparsos e padrões sequenciais locais. Espera-se, por essa razão, que eles cometam erros em amostras apenas parcialmente coincidentes, o que constitui a condição necessária para que o empilhamento venha a produzir algum ganho.

4.5.3 Geração de Probabilidades Fora da Dobra

Para cada aprendiz de Nível 0, denotado por m , o treinamento estratificado em cinco dobras gera probabilidades da classe positiva somente para as amostras que ficaram de fora do ajuste realizado em cada dobra. Desse procedimento resulta o vetor

$$P_{\text{OOF}}^{(m)} = \left[p_1^{(m)}, p_2^{(m)}, \dots, p_{N_{\text{train}}}^{(m)} \right]^T. \quad (4.12)$$

Os vetores de todos os aprendizes ativos formam

$$P_{\text{OOF}} = \left[P_{\text{OOF}}^{(1)}, P_{\text{OOF}}^{(2)}, \dots, P_{\text{OOF}}^{(B)} \right] \in \mathbb{R}^{N_{\text{train}} \times B}, \quad (4.13)$$

em que B corresponde ao número de aprendizes de base ativos. Em seguida, cada modelo de Nível 0 volta a ser treinado sobre a partição de treinamento completa e é aplicado à partição de teste isolada, o que produz

$$P_{\text{test}} \in \mathbb{R}^{N_{\text{test}} \times B}, \quad (4.14)$$

com a mesma ordenação de colunas adotada anteriormente. Dessa maneira, os aprendizes de Nível 1 recebem exclusivamente probabilidades alinhadas, provenientes dos modelos de base.

Esse procedimento concretiza o requisito de validade discutido na Seção 2.6.2. Como nenhuma amostra de treinamento recebeu predição de um modelo que a tivesse utilizado durante o ajuste, o meta-aprendiz passa a observar estimativas cujo comportamento se aproxima daquele que se espera encontrar em dados ainda não observados.

4.5.4 Meta-aprendizes de Nível 1, Calibração e Seleção de Limiar

Dois modelos foram considerados para o Nível 1, sendo o primeiro o LightGBM (LGBM_m), que funde as probabilidades de forma não linear, e o segundo a regressão logística com regularização *elastic net* (LR_m), que constitui a alternativa linear regularizada. Ambos aprendem sobre P_{OOF} e são avaliados sobre P_{test} . A decisão de manter a alternativa linear ao lado da não linear justifica-se porque somente essa comparação permite distinguir um ganho originado na fusão de evidências de outro que decorra apenas da capacidade adicional do meta-modelo. Diferentemente dos modelos de Nível 0, os dois meta-aprendizes não passaram por otimização de hiperparâmetros e foram avaliados com parâmetros fixos, apresentados na Tabela A.4 do Apêndice A.

As probabilidades de meta-nível são calibradas por uma regressão isotônica com validação cruzada interna de três dobras, pelas razões discutidas na Seção 2.8.4. A avaliação é conduzida no limiar fixo $\tau = 0,5$ e também em um limiar derivado da própria partição de treinamento, definido por

$$\tau^* = \arg \max_{\tau \in \mathcal{T}} F_1(y_{\text{train}}, \mathbb{I}[\hat{p}_{\text{meta,train}} \geq \tau]), \quad (4.15)$$

em que $\mathcal{T}$ combina quantis de probabilidade com pontos de operação fixos. O limiar assim escolhido é aplicado às probabilidades de teste isoladas sem que sofra qualquer alteração, de modo que

$$\hat{y}_{\text{test}} = \mathbb{I}[\hat{p}_{\text{meta,test}} \geq \tau^*]. \quad (4.16)$$

O protocolo calcula ainda outros limiares do lado do treinamento, voltados à acurácia e ao *recall* sob restrições de taxa de falsos positivos de 0,01 e de 0,05, embora os resultados principais obtidos com limiar ajustado recorram ao τ^* otimizado pela medida F1.

4.5.5 Controles contra Vazamento de Informação

O controle contra vazamento de informação apoia-se em uma regra única, aplicada sem exceção ao longo de todo o *pipeline*, segundo a qual o pré-processamento, a seleção de características, os modelos de Nível 0, os modelos de Nível 1, a calibração e os limiares são ajustados exclusivamente sobre dados do lado do treinamento. A partição de teste isolada só é utilizada depois que todas essas decisões já se encontram fixadas.

Esse conjunto de controles responde diretamente à terceira lacuna apontada na Seção 3.6 e é o que confere valor inferencial às comparações estatísticas apresentadas na Seção 4.6.4. Ao final de cada execução, o *pipeline* entrega, para cada conjunto de dados e para cada semente, as probabilidades calibradas de Nível 1, os limiares derivados do treinamento e as predições binárias sobre a partição de teste isolada, acompanhados dos tempos de treinamento e de inferência de cada componente.

4.6 PROTOCOLO EXPERIMENTAL

4.6.1 Composição das Execuções

Afirmou-se em diversos pontos deste capítulo que as variáveis de interesse variam isoladamente sob um protocolo idêntico. Um componente compositor lê quatro camadas de configuração, que são o manifesto do experimento, o contrato do conjunto de dados, os arquivos de hiperparâmetros travados e a semente, e funde essas camadas de forma recursiva em um único arquivo de configuração efetiva, identificado por um resumo criptográfico estável. O orquestrador recebe apenas esse arquivo e percorre sempre o mesmo caminho de execução, que compreende o particionamento estratificado, o ajuste dos preditores de Nível 0, a geração das probabilidades fora da dobra, o meta-aprendizado, a calibração e a seleção de limiar. A Figura 4.6 sintetiza esse arranjo.

A consequência desse arranjo é justamente o que sustenta o argumento, pois trocar o conjunto de dados, restringir o bloco de características ou alterar a semente modifica apenas a configuração, e não o procedimento efetivamente executado. As comparações pareadas apresentadas no Capítulo 5 são legítimas justamente porque as execuções confrontadas percorrem o mesmo código e diferem entre si somente na camada que foi declarada. Cada execução arquiva ainda uma cópia da configuração efetiva que a produziu, de modo que qualquer resultado do capítulo seguinte pode ser reconstituído a partir do arquivo correspondente.

Sob essa perspectiva, os experimentos complementares descritos na Seção 4.6.5 correspondem a manifestos distintos submetidos a um mesmo compositor, e não a implementações paralelas e independentes. A Tabela 4.4 registra a correspondência entre cada manifesto e as análises do Capítulo 5 que dele resultam.

Os arquivos de hiperparâmetros travados, reproduzidos no Apêndice A e organizados por modelo e por conjunto de dados, são produzidos em uma fase anterior, conduzida sob manifesto próprio. Uma vez concluída essa fase, tais arquivos passam a constituir entrada fixa de todas as execuções seguintes, o que assegura a separação entre a etapa de ajuste e a etapa de avaliação.

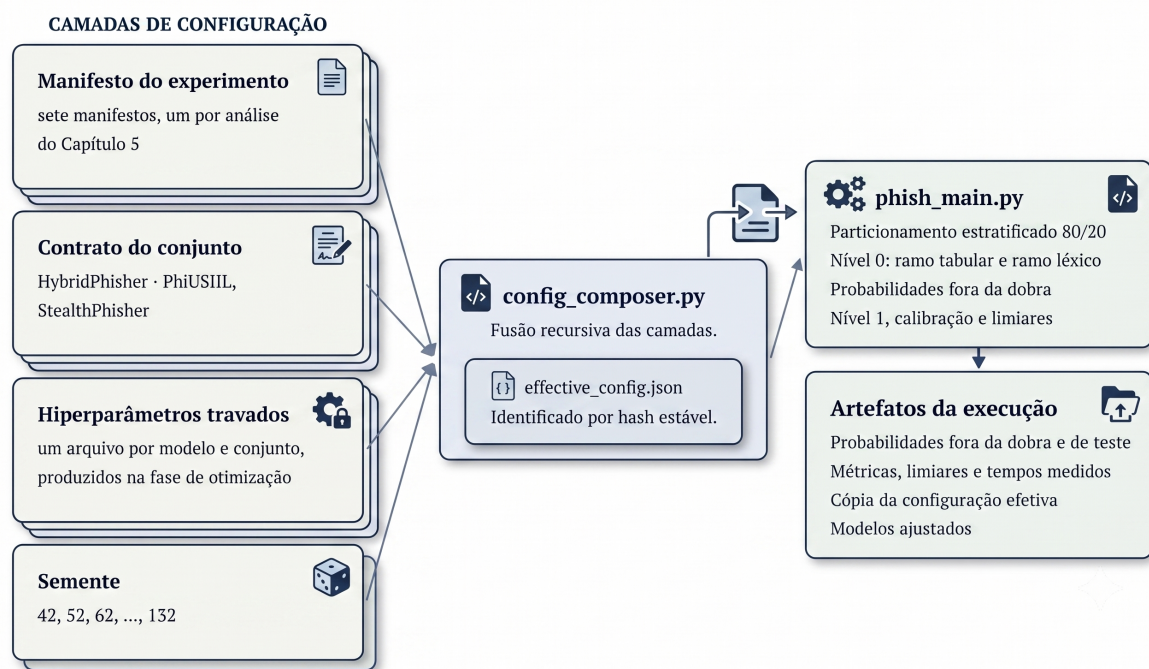


Figura 4.6 – Composição das execuções experimentais

Tabela 4.4 – Correspondência entre manifestos de experimento e análises do Capítulo 5

Manifesto	Análise correspondente
1_url_html_comparision	Análise de componentes e blocos de atributos (Seção 5.6)
2_tabular_baselines	Comparação com linhas de base tabulares robustas (Seção 5.5)
3_urlc_validation	Contribuição e sensibilidade do escore URLC (Seção 5.7)
4_multi_tests	Desempenho nas partições retidas e significância estatística (Seções 5.3 e 5.4)
5_efficiency_profile	Perfil de eficiência computacional (Seções 5.4 e 5.5)
6_adversarial_stress_test	Análise de estresse adversarial (Seção 5.8)
7_domain_similarity_validation	Contribuição da similaridade de domínio (Seção 5.9)

4.6.2 Conjuntos de Dados de Avaliação

O protocolo experimental submete o HOOF-Stack-PUD a *benchmarks* que apresentam níveis distintos de separabilidade e de erro residual, com o objetivo de determinar em quais situações o empilhamento em nível de probabilidade produz ganhos estáveis e em quais delas as representações tabulares projetadas já aproximam a tarefa da saturação.

A avaliação é composta por três *benchmarks* de grande escala, que são o PhiUSIIL (D1) (Prasad; Chandra, 2024), o StealthPhisher (D2) (Prasad et al., 2026) e o HybridPhisher (D3), este último construído conforme descreve a Seção 4.3. A Tabela 4.5 resume a composição de cada um deles.

Tabela 4.5 – Composição dos conjuntos de dados e dimensionalidade das características

Conjunto	Total	Phishing (%)	Legítimas (%)	Características
PhiUSIIL	235.795	42,81	57,19	52
StealthPhisher	336.749	52,21	47,79	62
HybridPhisher	221.360	49,26	50,74	58

A escolha desses três conjuntos obedeceu a um critério preciso, pois eles diferem entre si quanto ao procedimento de construção, ao espaço de características e à origem dos rótulos, o que os torna especialmente adequados à investigação da saturação. Como a margem de erro residual disponível varia de um conjunto para outro, espera-se que o valor agregado pelo empilhamento também varie. Em todos os casos, o particionamento dos dados, a geração fora da dobra, a calibração, a seleção de limiar e os controles contra vazamento seguem o procedimento descrito na Seção 4.5.

4.6.3 Otimização de Hiperparâmetros e Travamento das Configurações

A seleção de hiperparâmetros antecedeu as avaliações finais, e as configurações escolhidas permaneceram travadas a partir desse momento. Uma busca aleatória orientada por manifesto percorreu separadamente três grupos de modelos, a saber, os de aprendizado de máquina clássico, os de aprendizado profundo léxico e os tabulares profundos. Este último grupo reúne o TabNet (TabNet_t), o TabTransformer (TabTr_t) e o FT-Transformer (FTTr_t), arquiteturas que também servem como linhas de base autônomas e que são descritas na Seção 4.6.5. A Tabela 4.6 apresenta o protocolo adotado nessa etapa.

A PR-AUC de validação foi adotada como métrica de seleção pela razão que a Seção 2.8.1 apresenta, uma vez que ela é mais informativa do que a acurácia quando os custos associados a falsos positivos e a falsos negativos são distintos, condição que caracteriza a detecção de *phishing*. Em nenhuma etapa da seleção de hiperparâmetros foram consultados os rótulos da partição de teste. O Apêndice A reproduz as configurações finais travadas de LR_c, RF_t, XGB_t, LGBM_t, CNN_c, CNN_w, TabNet_t, TabTr_t e FTTr_t para os três *benchmarks*.

4.6.4 Execuções Repetidas e Comparações Pareadas

O *pipeline* completo foi executado sobre D1, D2 e D3 sob dez sementes distintas, que foram 42, 52, 62, 72, 82, 92, 102, 112, 122 e 132. As comparações pareadas entre sementes correspondentes confrontam

Tabela 4.6 – Protocolo de otimização de hiperparâmetros

Item	Configuração
Métrica de seleção	PR-AUC de validação
Modo de seleção	Maximizar
Protocolo de validação	Validação estratificada em cinco dobras
Semente	42
Tentativas por estágio	40
Paciência em nível de tentativa	10
Melhoria mínima	10^{-5}
Divisão principal	Divisão estratificada 80/20 entre treinamento e teste
Uso da partição de teste na otimização	Não utilizada
Modelos de aprendizado clássico	LR_c , RF_t , XGB_t , $LGBM_t$
Modelos léxicos profundos	CNN_c , CNN_w
Modelos tabulares profundos	$TabNet_t$, $TabTr_t$, $FTTr_t$
Pré-processamento tabular	$MinMaxScaler \rightarrow VarianceThreshold(10^{-5}) \rightarrow SelectKBest(\chi^2)$

os meta-modelos com o melhor aprendiz tabular, com o melhor aprendiz léxico e com o pior aprendiz de base.

O emparelhamento por semente constitui a escolha metodológica central desta etapa. Conforme discute a Seção 2.8.3, quando dois métodos são executados sobre as mesmas partições, a diferença entre eles elimina parte da variabilidade associada ao conjunto de teste. Esse controle torna-se indispensável no presente caso, porque o ganho esperado do empilhamento tem a mesma ordem de grandeza do erro residual que ainda permanece. Os testes aplicados a essas diferenças e o cálculo dos intervalos de confiança estão descritos na Seção 4.6.6.

4.6.5 Experimentos Complementares

Cinco conjuntos de experimentos adicionais foram conduzidos com a finalidade de dar contexto aos resultados obtidos pelo empilhamento.

O primeiro confronta o arcabouço com linhas de base tabulares autônomas fortes, a saber, *Random Forest* (Breiman, 2001), *XGBoost* (Chen; Guestrin, 2016), *LightGBM* (Ke et al., 2017), *TabNet* (Arik; Pfister, 2021), *TabTransformer* (Huang et al., 2020) e *FT-Transformer* (Gorishniy et al., 2021). As três últimas são arquiteturas profundas projetadas especificamente para dados tabulares: Arik e Pfister (2021) propõem seleção sequencial de atributos por máscaras esparsas com atenção, Huang et al. (2020) aplicam camadas de autoatenção sobre *embeddings* contextuais de variáveis categóricas, e Gorishniy et al. (2021) adaptam o *Transformer* ao domínio tabular e mostram que arquiteturas profundas bem ajustadas competem com modelos de *boosting*. Esse experimento responde à quarta lacuna da Seção 3.6, segundo a qual comparar um meta-modelo apenas com classificadores fracos superestima a contribuição da combinação.

O segundo avalia os blocos de características, contrastando entre si F_{URL} , F_{HTML} e $F_{URL+HTML}$. Esses blocos são fornecidos aos modelos tabulares isolados, sem os ramos lexicais e sem a camada de meta-aprendizado, com dupla finalidade. A primeira é verificar se um único bloco já responde pela separação das classes, caso em que o outro seria dispensável, ou se a junção $F_{URL+HTML}$ acrescenta ganho sobre cada bloco isolado. A segunda é verificar se a evidência tabular, por si só, já satura o *benchmark*, situação em que não restaria erro residual para a camada de meta-aprendizado corrigir e o empilhamento não traria

ganho.

O terceiro avalia os descritores compostos, uma vez que tanto o URLC quanto o `Domain_Similarity` são submetidos a comparações pareadas com e sem o atributo, mantidos fixos o particionamento, a semente e as demais características. A sensibilidade do URLC recebe ainda um exame adicional, conduzido por meio da variação dos pesos de seus quatro componentes, definidos nas Equações (4.6) a (4.10).

O quarto caracteriza a eficiência computacional sobre o HybridPhisher a partir de quatro medidas, que são o tempo de treinamento fora da dobra, o tempo de treinamento do modelo final, o tempo de predição sobre a partição isolada e a latência de inferência por amostra. Essa caracterização atende à sexta lacuna apontada na Seção 3.6, pois um ganho preditivo pequeno pode não compensar um *pipeline* substancialmente mais caro, sobretudo quando a decisão precisa ocorrer antes do acesso à página.

O quinto experimento consiste em um teste de estresse adversarial conduzido em nível de *pipeline*. Nele, as perturbações alcançam exclusivamente as amostras de *phishing* da partição de teste, e o *pipeline* previamente treinado permanece fixo, sem que receba qualquer reajuste sobre as amostras perturbadas. As condições de perturbação em nível de URL e em nível de HTML são avaliadas de forma independente uma da outra. Esse desenho concretiza a definição operacional apresentada na Seção 2.9, segundo a qual a degradação medida expressa a sensibilidade do sistema, e não sua capacidade de readaptação.

4.6.6 Métricas de Avaliação

Todos os experimentos reportam a acurácia, a precisão, a medida F1, o *recall*, a ROC-AUC e a PR-AUC, calculadas em $\tau = 0,5$ e, quando aplicável, no limiar τ^* derivado do treinamento conforme a Equação (4.15).

As análises de execuções repetidas recorrem a comparações estatísticas pareadas e a intervalos de confiança (IC) de 95%, calculados sobre as dez diferenças de F1, com limiar de decisão fixo em 0,5, entre execuções que compartilham a mesma semente e, portanto, a mesma partição de teste. A apresentação simultânea dos resultados em limiar fixo e em limiar otimizado atende a um propósito específico, uma vez que somente essa dupla leitura permite verificar se eventuais ganhos sobrevivem à mudança da política de decisão.

Na comparação entre meta-modelos e modelos de base, apresentada na Tabela 5.4, o valor p decorre do teste t de Student para amostras pareadas. A hipótese nula é a de que a diferença média de F1 entre os dois modelos é nula, e o teste é bilateral, com nove graus de liberdade. Como cada semente sorteia uma nova partição de treinamento e teste a partir do mesmo conjunto de dados, as execuções compartilham parte das amostras de treinamento, e as diferenças entre elas não são independentes. Por essa razão, a mesma tabela apresenta, na coluna p_{NB} , o resultado do teste com a correção de variância proposta por Nadeau e Bengio (2003), mais conservadora, que multiplica a variância da diferença média pelo fator $1 + J N_{\text{test}}/N_{\text{train}}$, igual a 3,5 nos três conjuntos de dados, mantidos os nove graus de liberdade.

Nas comparações com e sem atributo, apresentadas na Tabela 5.9, o valor p_{perm} decorre de um teste de permutação por troca de sinais. A hipótese nula é a de que a inclusão do atributo não altera o F1 esperado, de modo que cada diferença teria a mesma probabilidade de ser positiva ou negativa. Sob essa hipótese,

os sinais das dez diferenças são trocados aleatoriamente, e p_{perm} corresponde à proporção de trocas cuja média absoluta iguala ou supera a observada, acrescidos de uma unidade o numerador e o denominador. O teste é bilateral, e os valores foram estimados a partir de 10 mil sorteios aleatórios de sinais, com semente fixa, entre as 1.024 combinações possíveis para dez diferenças. O valor 0,0016, comum às linhas em que as dez diferenças têm o mesmo sinal, corresponde à estimativa obtida por esses sorteios para esse caso extremo.

Os intervalos de confiança não constituem teste de hipótese, pois expressam a incerteza associada à diferença média. Na Tabela 5.4, foram obtidos pela distribuição t de Student com nove graus de liberdade, a mesma que fundamenta o teste pareado; na Tabela 5.9, por reamostragem *bootstrap* das dez diferenças por semente, com mil repetições.

Os resultados produzidos sob esse protocolo são apresentados no Capítulo 5, ao passo que as limitações que delimitam o alcance deles são retomadas na Seção 6.1.

5 RESULTADOS

Este capítulo apresenta os resultados experimentais que o HOOF-Stack-PUD produziu nos três conjuntos de dados adotados nesta pesquisa: PhiUSIIL (D1) (Prasad; Chandra, 2024), StealthPhisher (D2) (Prasad et al., 2026) e HybridPhisher (D3), este último construído no âmbito deste trabalho conforme descreve o Capítulo 4. Todas as execuções seguiram o protocolo da Seção 4.5, sob os controles contra vazamento de informação enunciados na Seção 4.5.5.

A exposição não se limita a relatar os valores obtidos. Cada resultado vem acompanhado da explicação do mecanismo que o produziu, isto é, das propriedades do conjunto de dados, do espaço de atributos e da arquitetura de empilhamento que tornam aquele comportamento esperado. A tese central deste trabalho, de que o ganho do empilhamento é condicionado ao nível de saturação do *benchmark*, só se torna verificável quando cada tabela é lida imediatamente à luz do mecanismo que a explica. A Seção 5.10 encerra o capítulo confrontando o conjunto das evidências com a literatura revisada nos Capítulos 2 e 3.

5.1 AMBIENTE COMPUTACIONAL

Todos os experimentos rodaram em processador AMD Ryzen 7 7800X3D, com 96 GB de memória RAM e placa gráfica NVIDIA GeForce RTX 4060 Ti com 8 GB de memória dedicada, sobre Python 3.11.9, PyTorch 2.8.0, CUDA 12.8 e controlador NVIDIA 580.88.

Quatro elementos sustentam a reprodutibilidade dos resultados: as dez sementes fixas listadas na Seção 4.6.4, o travamento das configurações de hiperparâmetros antes das avaliações finais, o esquema determinístico de características da Seção 4.4 e o registro completo do ambiente de execução. Como todas as medições de tempo deste capítulo vieram desse mesmo ambiente, seus valores absolutos comparam-se entre si, mas não se transferem diretamente a outras configurações de *hardware*.

5.2 CRITÉRIO DE LEITURA DOS RESULTADOS

A análise separa deliberadamente duas coisas que a leitura apressada confunde: desempenho preditivo e saturação do *benchmark*. Um valor elevado de métrica não atesta apenas a capacidade do modelo, ele pode revelar que a representação de atributos já torna o conjunto de dados praticamente separável. A distinção importa porque uma arquitetura de empilhamento só produz ganho mensurável quando os aprendizes de base deixam erro residual que evidência complementar consiga corrigir. Se o melhor modelo de base já opera no limite superior da escala métrica, falta margem aritmética para que a camada de meta-aprendizado demonstre utilidade, por melhor que seja o projeto do empilhamento.

Daí decorrem as três perguntas que guiam a discussão: os ramos tabular e lexical fornecem informação complementar? As melhorias resistem a execuções repetidas? Os ganhos sobrevivem quando a comparação

enfrenta linhas de base tabulares fortes, e não apenas ramos individualmente mais fracos? As Seções 5.3 a 5.9 aplicam esse critério de maneira uniforme.

5.3 DESEMPENHO NAS PARTIÇÕES DE TESTE RETIDAS

As Tabelas 5.1, 5.2 e 5.3 apresentam os resultados principais de execuções repetidas com limiar de decisão fixo em 0.5. Cada célula traz média $\pm$ desvio padrão ao longo de dez sementes aleatórias distintas, o que separa o desempenho médio da variabilidade que a inicialização e o particionamento interno induzem. Lidas em conjunto, as três tabelas revelam três regimes distintos de comportamento.

O conjunto D1 exhibe um padrão de saturação. Os aprendizes tabulares mais fortes alcançam valores perfeitos ou quase perfeitos em todas as métricas reportadas, com variabilidade desprezível entre execuções: RF_t e $LGBM_t$ atingem F1 de 0.99999 ± 0.00002 , e a precisão permanece em 1.00000 ± 0.00000 para os três modelos tabulares. Esse resultado deve ser interpretado primariamente como propriedade da combinação entre o conjunto de dados e o espaço de atributos, e não como evidência isolada de superioridade arquitetural. Uma vez disponíveis os atributos tabulares projetados, D1 apresenta ambiguidade residual muito limitada, de modo que o modelo empilhado tem pouca margem prática para produzir melhoria mensurável sobre o melhor ramo tabular.

Tabela 5.1 – Desempenho no conjunto PhiUSIIL com limiar de decisão fixo (0.5) em execuções repetidas

Modelo	ACC	Prec	Recall	F1	PR-AUC	Execuções
<i>Modelos de base — tabulares</i>						
RF_t	0.99999 ± 0.00001	1.00000 ± 0.00000	0.99998 ± 0.00004	0.99999 ± 0.00002	0.99999 ± 0.00002	10
XGB_t	0.99996 ± 0.00003	1.00000 ± 0.00000	0.99991 ± 0.00007	0.99996 ± 0.00003	0.99999 ± 0.00002	10
$LGBM_t$	0.99999 ± 0.00002	1.00000 ± 0.00000	0.99998 ± 0.00004	0.99999 ± 0.00002	1.00000 ± 0.00000	10
<i>Modelos de base — lexicais</i>						
LR_c	0.99785 ± 0.00022	0.99975 ± 0.00013	0.99523 ± 0.00054	0.99748 ± 0.00026	0.99922 ± 0.00014	10
CNN_c	0.99796 ± 0.00050	0.99928 ± 0.00120	0.99596 ± 0.00045	0.99762 ± 0.00058	0.99904 ± 0.00015	10
CNN_w	0.99626 ± 0.00029	0.99832 ± 0.00096	0.99294 ± 0.00082	0.99562 ± 0.00034	0.99828 ± 0.00012	10
<i>Meta-modelos</i>						
$LGBM_m$	1.00000 ± 0.00001	1.00000 ± 0.00000	0.99999 ± 0.00003	1.00000 ± 0.00002	1.00000 ± 0.00000	10
LR_m	1.00000 ± 0.00001	1.00000 ± 0.00000	0.99999 ± 0.00003	1.00000 ± 0.00002	1.00000 ± 0.00000	10

O conjunto D2 também é saturado pelo critério da Seção 2.8.2, embora pequenas diferenças entre famílias de modelos permaneçam observáveis. Os ramos tabulares, em especial $LGBM_t$ e XGB_t , mantêm-se mais fortes que os ramos lexicais, com F1 de 0.99960 ± 0.00005 e 0.99952 ± 0.00005 contra 0.99827 ± 0.00037 do melhor ramo lexical, CNN_c . Isso indica que a representação tabular projetada explica a maior parte da separação de classes nesse *benchmark*. Os meta-modelos alcançam os maiores valores de F1, mas a margem absoluta sobre o melhor modelo tabular é pequena. O comportamento sugere que o empilhamento fornece refinamento incremental consistente em D2, e não separação prática expressiva, justamente porque o aprendiz de base mais forte já se encontra próximo ao limite superior da escala métrica.

O conjunto D3 constitui o cenário mais informativo para avaliar a contribuição do arcabouço de empilhamento. Diferentemente de D1 e D2, os modelos isolados não saturam o *benchmark*. Os ramos tabulares mais fortes permanecem competitivos, mas deixam erro residual maior do que nos demais conjuntos:

Tabela 5.2 – Desempenho no conjunto StealthPhisher com limiar de decisão fixo (0.5) em execuções repetidas

Modelo	ACC	Prec	Recall	F1	PR-AUC	Execuções
<i>Modelos de base — tabulares</i>						
RF _t	0.99910 ± 0.00009	0.99949 ± 0.00009	0.99879 ± 0.00015	0.99914 ± 0.00009	0.99997 ± 0.00001	10
XGB _t	0.99950 ± 0.00006	0.99956 ± 0.00008	0.99948 ± 0.00010	0.99952 ± 0.00005	0.99999 ± 0.00001	10
LGBM _t	0.99958 ± 0.00006	0.99966 ± 0.00009	0.99954 ± 0.00008	0.99960 ± 0.00005	0.99998 ± 0.00001	10
<i>Modelos de base — lexicais</i>						
LR _c	0.99753 ± 0.00024	0.99951 ± 0.00008	0.99576 ± 0.00045	0.99763 ± 0.00023	0.99926 ± 0.00009	10
CNN _c	0.99820 ± 0.00038	0.99937 ± 0.00072	0.99718 ± 0.00059	0.99827 ± 0.00037	0.99955 ± 0.00006	10
CNN _w	0.99542 ± 0.00037	0.99858 ± 0.00062	0.99263 ± 0.00084	0.99560 ± 0.00036	0.99846 ± 0.00014	10
<i>Meta-modelos</i>						
LGBM _m	0.99974 ± 0.00006	0.99970 ± 0.00010	0.99979 ± 0.00012	0.99975 ± 0.00005	0.99998 ± 0.00002	10
LR _m	0.99971 ± 0.00005	0.99962 ± 0.00008	0.99983 ± 0.00009	0.99973 ± 0.00004	0.99997 ± 0.00002	10

XGB_t alcança F1 de 0.98696 ± 0.00041 e LGBM_t, 0.98646 ± 0.00034 . Os ramos lexicais não são os melhores detectores isolados, mas suas estimativas de probabilidade fornecem evidência adicional que a camada de meta-aprendizado consegue explorar. Como consequência, ambos os meta-modelos superam os ramos isolados mais fortes, com LGBM_m e LR_m atingindo F1 de 0.99304 e 0.99300, respectivamente. Esse padrão indica que o benefício do empilhamento é mais visível quando preditores tabulares e lexicais capturam aspectos parcialmente complementares do sinal de *phishing*.

Tabela 5.3 – Desempenho no conjunto HybridPhisher com limiar de decisão fixo (0.5) em execuções repetidas

Modelo	ACC	Prec	Recall	F1	PR-AUC	Execuções
<i>Modelos de base — tabulares</i>						
RF _t	0.98314 ± 0.00051	0.98735 ± 0.00091	0.97831 ± 0.00072	0.98281 ± 0.00052	0.99850 ± 0.00010	10
XGB _t	0.98719 ± 0.00041	0.98995 ± 0.00070	0.98398 ± 0.00056	0.98696 ± 0.00041	0.99886 ± 0.00009	10
LGBM _t	0.98670 ± 0.00033	0.98968 ± 0.00063	0.98326 ± 0.00066	0.98646 ± 0.00034	0.99889 ± 0.00009	10
<i>Modelos de base — lexicais</i>						
LR _c	0.98320 ± 0.00059	0.99012 ± 0.00103	0.97564 ± 0.00117	0.98283 ± 0.00061	0.99811 ± 0.00011	10
CNN _c	0.98234 ± 0.00102	0.98825 ± 0.00440	0.97578 ± 0.00495	0.98196 ± 0.00107	0.99811 ± 0.00011	10
CNN _w	0.97915 ± 0.00082	0.98386 ± 0.00490	0.97369 ± 0.00428	0.97873 ± 0.00080	0.99804 ± 0.00012	10
<i>Meta-modelos</i>						
LGBM _m	0.99316 ± 0.00038	0.99478 ± 0.00101	0.99131 ± 0.00115	0.99304 ± 0.00038	0.99971 ± 0.00003	10
LR _m	0.99312 ± 0.00044	0.99506 ± 0.00071	0.99095 ± 0.00117	0.99300 ± 0.00045	0.99964 ± 0.00004	10

As Figuras 5.1 e 5.2 complementam as tabelas ao sintetizar a estabilidade entre execuções repetidas. Nelas, cada ponto representa o F1 médio de um modelo em um dado *benchmark* e a barra horizontal corresponde a um desvio padrão entre sementes; os sufixos *t*, *c*, *w* e *m* identificam, respectivamente, os modelos tabulares, os lexicais em nível de caractere, os lexicais em nível de palavra e os de meta-aprendizado. A Figura 5.1 mostra que D1 e D2 concentram-se junto ao limite superior de F1, o que reforça visualmente a saturação observada nas tabelas; em contraste, D3 apresenta separação mais nítida entre famílias de modelos.

A Figura 5.2 detalha o comportamento por semente em D3, com um ponto por semente e curvas mais espessas para os meta-modelos, e evidencia que estes permanecem consistentemente acima dos ramos isolados em todas as execuções. Esse detalhamento é relevante porque sustenta a interpretação de que o ganho observado em D3 não decorre de uma semente favorável isolada, mas de redução estável do erro

residual por meio da combinação heterogênea de modelos.

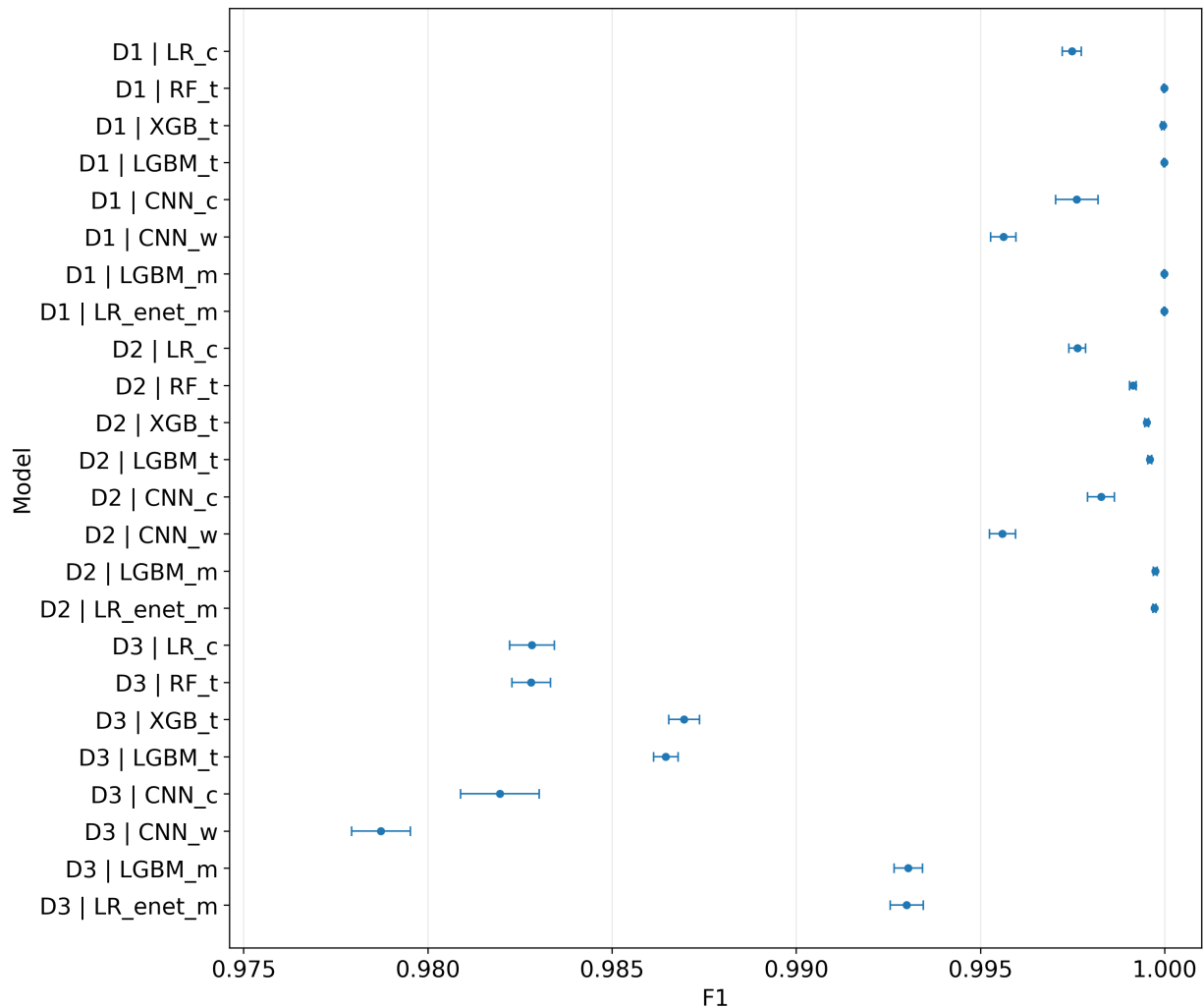


Figura 5.1 – Média e desvio padrão de F1 em execuções repetidas para a configuração com o conjunto completo de atributos

A Figura 5.3 acrescenta a mesma leitura sob a perspectiva de PR-AUC, métrica orientada a ordenação e adequada a cenários de detecção com desbalanceamento de classes; seus marcadores denotam o valor médio de cada modelo em cada conjunto e as barras horizontais, um desvio padrão entre sementes. O comportamento é coerente com o observado para F1: em D1 e D2 as distribuições colapsam junto ao limite superior, enquanto em D3 os meta-modelos se destacam dos ramos isolados. A convergência entre uma métrica dependente de limiar (F1) e uma métrica independente de limiar (PR-AUC) indica que o ganho do empilhamento em D3 não é artefato da escolha do ponto de corte, mas reflete melhoria efetiva na ordenação das probabilidades preditas.

Em síntese, os resultados principais devem ser lidos como evidência de utilidade condicional, e não de superioridade uniforme. Quando o espaço de atributos tabulares já satura o *benchmark*, como em D1, o empilhamento não proporciona ganho adicional relevante. Quando o *benchmark* está saturado, mas ainda deixa erro residual mensurável, como em D2, o ganho é pequeno, porém consistente. Quando o *benchmark* preserva ambiguidade residual, como em D3, a representação empilhada produz a melhoria mais nítida ao

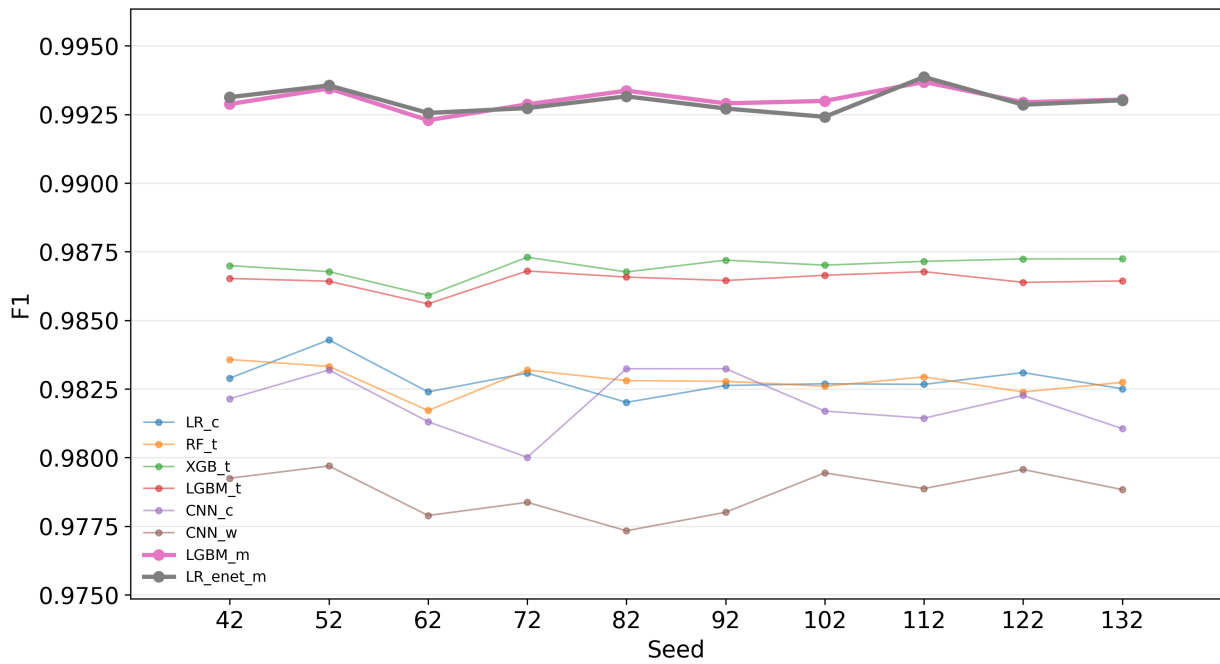


Figura 5.2 – Variação de F1 entre sementes em D3 para a configuração com o conjunto completo de atributos

combinar evidência tabular e lexical na camada final de decisão.

5.4 EXECUÇÕES REPETIDAS E SIGNIFICÂNCIA ESTATÍSTICA

A Tabela 5.4 reporta as diferenças pareadas de F1, com limiar de decisão fixo em 0.5, entre o meta-modelo selecionado e os modelos de base, calculadas sobre sementes correspondentes. Três comparações distintas figuram lado a lado, contra o melhor tabular, contra o melhor lexical e contra o pior modelo de base, de modo que a tabela delimite a faixa de ganho do meta-aprendizado: as duas primeiras mostram o ganho em relação ao ramo isolado mais forte disponível, e a terceira, o ganho máximo. O cuidado tem motivo: confrontar o meta-modelo apenas com o ramo lexical, ou apenas com o pior modelo de base, produziria uma leitura otimista e metodologicamente frágil do benefício do empilhamento. Em D1, LR_m e LGBM_m produzem valores idênticos em todas as sementes, e manteve-se LR_m como referência.

Tabela 5.4 – Comparação estatística entre os melhores meta-modelos e os modelos de base em execuções repetidas, com limiar de decisão fixo (0.5)

Conjunto	Comparação	Meta	Base	Métrica	Δ	IC 95%	Valor p	p_{NB}
D1	Meta vs. melhor tabular	LR _m	LGBM _t	F1	0.00000	[-0.00001, 0.00002]	0.343436	0.605931
D1	Meta vs. melhor lexical	LR _m	CNN _c	F1	0.00238	[0.00196, 0.00279]	$< 10^{-4}$	$< 10^{-4}$
D1	Meta vs. pior base	LR _m	CNN _w	F1	0.00437	[0.00412, 0.00462]	$< 10^{-4}$	$< 10^{-4}$
D2	Meta vs. melhor tabular	LGBM _m	LGBM _t	F1	0.00015	[0.00009, 0.00020]	0.000257	0.012632
D2	Meta vs. melhor lexical	LGBM _m	CNN _c	F1	0.00147	[0.00123, 0.00171]	$< 10^{-4}$	$< 10^{-4}$
D2	Meta vs. pior base	LGBM _m	CNN _w	F1	0.00415	[0.00390, 0.00440]	$< 10^{-4}$	$< 10^{-4}$
D3	Meta vs. melhor tabular	LGBM _m	XGB _t	F1	0.00608	[0.00578, 0.00638]	$< 10^{-4}$	$< 10^{-4}$
D3	Meta vs. melhor lexical	LGBM _m	LR _c	F1	0.01021	[0.00976, 0.01067]	$< 10^{-4}$	$< 10^{-4}$
D3	Meta vs. pior base	LGBM _m	CNN _w	F1	0.01431	[0.01374, 0.01489]	$< 10^{-4}$	$< 10^{-4}$

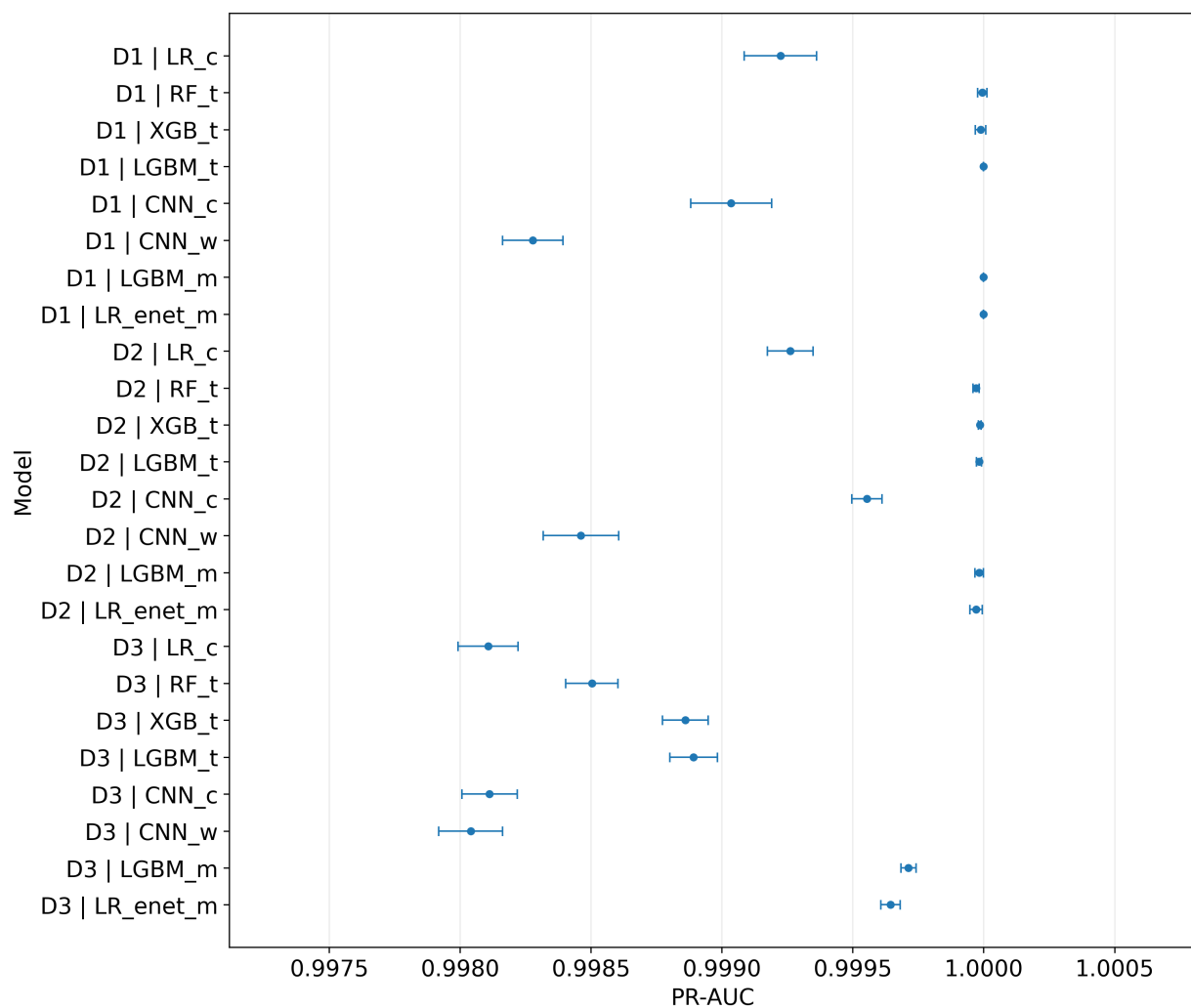


Figura 5.3 – Média e desvio padrão de PR-AUC em execuções repetidas nos três *benchmarks*

Os resultados confirmam que a evidência estatística em favor do empilhamento depende da ambiguidade residual de cada *benchmark*. Em D1, o melhor modelo tabular, $LGBM_t$, erra apenas quatro URLs em 471.590 predições nas dez sementes, e LR_m não acrescenta ganho ($\Delta = 0.00000$; IC 95% $[-0.00001, 0.00002]$; $p = 0.343436$; $p_{NB} = 0.605931$). Esse resultado é coerente com a interpretação de que D1 já está saturado pela evidência tabular, o que não deixa margem significativa para que a camada empilhada altere a decisão.

Cabe observar que a ausência de diferença estatística, nesse caso, não indica falha do arcabouço: ela indica que o teste está sendo aplicado a um cenário em que o erro residual é praticamente nulo, de modo que qualquer diferença mensurável seria numericamente irrelevante.

Em D2, $LGBM_m$ acrescenta ao F1 de $LGBM_t$ um ganho pequeno, porém consistente ($\Delta = 0.00015$; IC 95% $[0.00009, 0.00020]$; $p = 0.000257$; $p_{NB} = 0.012632$). O intervalo de confiança estritamente positivo mostra que o ganho é sistemático entre sementes, e não fruto de flutuação amostral. Ainda assim, sua magnitude prática permanece limitada pelo desempenho da linha de base tabular mais forte, já próximo do limite superior da escala, o que ilustra a distinção entre significância estatística e relevância prática.

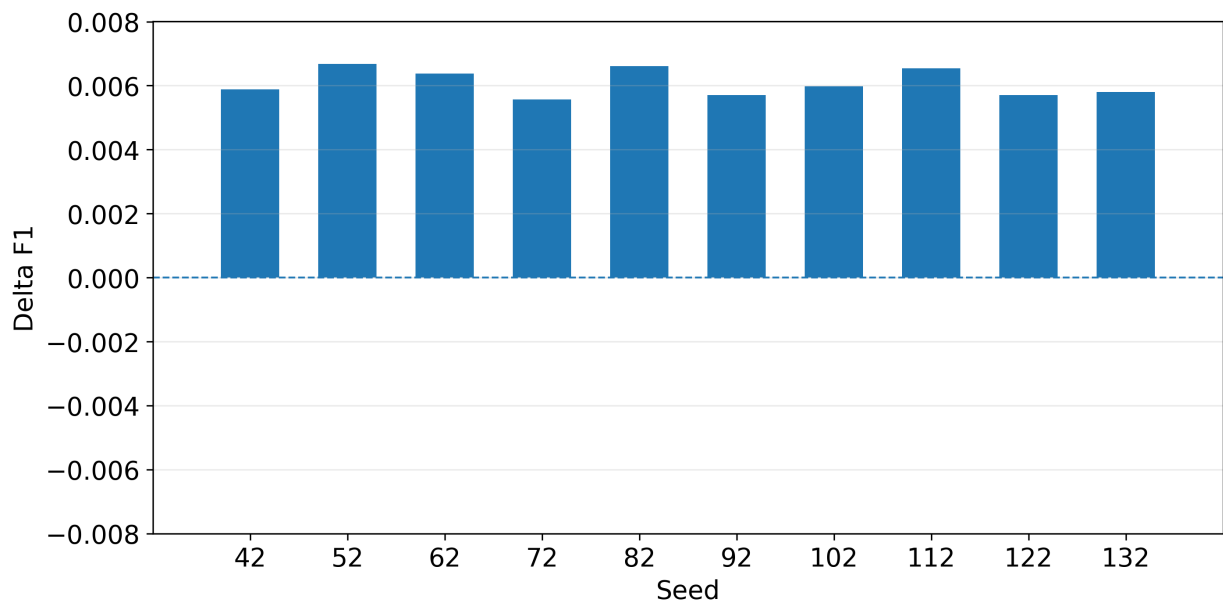


Figura 5.4 – Diferença de F1 por semente em D3 entre o melhor meta-modelo e o melhor modelo de base, com limiar de decisão fixo em 0.5

O conjunto D3 fornece a evidência mais clara de que o empilhamento heterogêneo reduz o erro residual. Nesse *benchmark*, $LGBM_m$ acrescenta ao F1 da linha de base tabular mais forte, XGB_t , um ganho de $\Delta = 0.00608$, com intervalo de confiança estreito e positivo ($[0.00578, 0.00638]$; $p < 10^{-4}$; $p_{NB} < 10^{-4}$). O ganho sobre a melhor linha de base lexical é maior ($\Delta = 0.01021$; IC 95% $[0.00976, 0.01067]$; $p < 10^{-4}$; $p_{NB} < 10^{-4}$), o que demonstra que os modelos lexicais não são suficientes como detectores isolados, mas ainda assim fornecem estimativas de probabilidade complementares ao meta-aprendiz. Em relação ao pior modelo de base, CNN_w , o ganho chega a $\Delta = 0.01431$ (IC 95% $[0.01374, 0.01489]$). Diferentemente de D1 e D2, D3 preserva erro residual suficiente para que as diferenças entre famílias de modelos se tornem observáveis e estatisticamente estáveis ao longo de execuções repetidas.

A Figura 5.4 traz essa mesma evidência em nível de semente, com a diferença de F1 entre o melhor meta-modelo e o melhor modelo de base em cada execução de D3. Todas as diferenças são positivas, não apenas a média, e é isso que sustenta a afirmação de ganho estável. Se o benefício viesse de variabilidade aleatória, o sinal das diferenças alternaria entre sementes.

A Tabela 5.5 complementa esse resultado com os tempos medidos em nível de modelo na execução completa do *pipeline* em D3. A coluna de inferência tem significado distinto nas duas camadas: nos modelos de base, ela mede o tempo de predição sobre a partição de teste retida; nos meta-modelos, o tempo da predição final sobre a matriz empilhada de probabilidades.

Os meta-modelos mais fortes melhoram F1 e PR-AUC a um custo modesto no estágio final de inferência, uma vez disponíveis as probabilidades dos modelos de base: LR_m alcança $F1 = 0.99313$ com 0.0110 s de inferência em nível meta, e $LGBM_m$ alcança $F1 = 0.99288$ com 0.3730 s. Esses valores não representam o custo total do detector empilhado, já que as predições de base precisam ser computadas antes. Eles mostram, todavia, que o estágio de fusão final não domina a latência, distinção que importa à leitura de viabilidade operacional, porque quem dita o custo agregado do *pipeline* são os ramos de base, e não a camada de meta-aprendizado.

Tabela 5.5 – Perfil de eficiência em nível de modelo no HybridPhisher

Modelo	F1	PR-AUC	Treino (s)	Inferência (s)	ms/amostra	Inf. rel.
LR_c	0.98289	0.99824	21.72	0.0330	0.0007	3.00x
RF_t	0.98357	0.99853	43.11	4.5460	0.1027	413.27x
XGB_t	0.98699	0.99889	37.99	0.8550	0.0193	77.73x
$LGBM_t$	0.98653	0.99895	36.12	2.8370	0.0641	257.91x
CNN_c	0.98214	0.99827	303.95	4.8430	0.1094	440.27x
CNN_w	0.97925	0.99830	202.69	1.8880	0.0426	171.64x
$LGBM_m$	0.99288	0.99972	96.94	0.3730	0.0084	33.91x
LR_m	0.99313	0.99970	36.18	0.0110	0.0002	1.00x

De modo geral, a análise de execuções repetidas sustenta uma interpretação condicional do HOOF-Stack-PUD: o empilhamento acrescenta pouco quando os atributos tabulares já saturam o *benchmark*, mas proporciona ganhos estáveis quando persiste erro residual, como observado em D3.

5.5 COMPARAÇÃO COM LINHAS DE BASE TABULARES ROBUSTAS

A Tabela 5.6 avalia se o arcabouço empilhado permanece relevante quando comparado a aprendizes tabulares isolados de alta capacidade, treinados sobre o mesmo espaço de atributos derivados de URL e HTML. Nela, a coluna de treino mede o tempo de ajuste do modelo final sobre a partição completa de treinamento, e a coluna de inferência, o tempo de predição sobre a partição de teste retida. Essa comparação foi incluída para evitar que ganhos sejam atribuídos ao empilhamento em cenários nos quais o *benchmark* já pode ser resolvido por um único modelo tabular suficientemente expressivo. Sem esse controle, a conclusão sobre a utilidade da arquitetura ficaria condicionada à escolha de linhas de base fracas.

Tabela 5.6 – Comparação de linhas de base tabulares isoladas em D1–D3

Conjunto	Modelo	ACC	Pre	F1	Recall	AUC	PR-AUC	Treino (s)	Inferência (s)
D1	RF _t	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	46.49	0.2941
D1	XGB _t	1.0000	1.0000	1.0000	0.9999	1.0000	1.0000	9.39	0.1231
D1	LGBM _t	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	7.42	0.3629
D1	TabNet _t	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	541.80	0.8898
D1	TabTr _t	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	516.98	0.9260
D1	FTTr _t	0.9997	1.0000	0.9997	0.9993	1.0000	1.0000	415.99	0.8116
D2	RF _t	0.9990	0.9995	0.9990	0.9986	1.0000	1.0000	23.12	0.3596
D2	XGB _t	0.9995	0.9997	0.9995	0.9994	1.0000	1.0000	15.14	0.1815
D2	LGBM _t	0.9997	0.9997	0.9997	0.9996	1.0000	1.0000	11.81	0.6914
D2	TabNet _t	0.9978	0.9998	0.9978	0.9959	1.0000	1.0000	783.84	1.2574
D2	TabTr _t	0.9991	0.9993	0.9991	0.9989	1.0000	1.0000	818.64	1.3635
D2	FTTr _t	0.9984	0.9992	0.9985	0.9978	1.0000	1.0000	651.28	1.1480
D3	RF _t	0.9848	0.9910	0.9844	0.9779	0.9986	0.9987	14.91	0.4534
D3	XGB _t	0.9879	0.9919	0.9876	0.9834	0.9989	0.9989	14.67	0.2732
D3	LGBM _t	0.9879	0.9924	0.9877	0.9830	0.9989	0.9991	13.45	0.9042
D3	TabNet _t	0.9841	0.9874	0.9838	0.9802	0.9972	0.9976	818.46	1.0515
D3	TabTr _t	0.9817	0.9803	0.9814	0.9826	0.9969	0.9973	672.85	0.6374
D3	FTTr _t	0.9822	0.9816	0.9820	0.9823	0.9978	0.9981	423.33	0.7043

Os resultados mostram que os modelos de *boosting* são as linhas de base tabulares isoladas mais fortes nos cenários mais informativos. Em D1, praticamente todos os modelos tabulares atingem o limite superior da escala métrica, com cinco modelos obtendo $F1 = 1.0000$ e FTTr_t alcançando $F1 = 0.9997$. Isso confirma que D1 não discrimina de forma significativa entre aprendizes tabulares fortes e, portanto, não deve ser utilizado isoladamente como critério de escolha arquitetural. Em D2, as diferenças entre modelos tornam-se mais visíveis, mas a melhor linha de base permanece próxima do limite superior da escala: LGBM_t alcança $F1 = 0.9997$, seguido por XGB_t, com $F1 = 0.9995$. Os modelos tabulares profundos são competitivos, mas não superam os modelos de *boosting* mais fortes, com valores de $F1$ entre 0.9978 e 0.9991.

O conjunto D3 fornece a comparação mais clara entre aprendizes tabulares, pois as métricas deixam de colapsar no limite superior. Nesse conjunto, LGBM_t e XGB_t são os modelos tabulares isolados mais fortes, com $F1 = 0.9877$ e 0.9876 , respectivamente. As linhas de base neurais permanecem inferiores, com valores de $F1$ entre 0.9814 e 0.9838. Esse ordenamento indica que, para o espaço de atributos projetado neste trabalho, as árvores impulsionadas oferecem referência tabular mais forte do que TabNet, TabTransformer ou FT-Transformer. Uma explicação plausível está na natureza dos atributos utilizados: trata-se de descritores heterogêneos, majoritariamente numéricos, com muitas relações não lineares por limiar, condição na qual árvores impulsionadas costumam ser particularmente eficazes sem exigir grande volume de dados de treinamento. Como consequência, qualquer alegação de melhoria por meta-aprendizado precisa ser avaliada contra essas linhas de base de *boosting*, e não contra ramos isolados mais fracos.

A Tabela 5.7 mostra que essa vantagem preditiva vem acompanhada de um compromisso computacio-

nal mais favorável. Ela decompõe o custo em quatro parcelas, o treinamento fora de dobra, o treinamento do modelo final, a predição sobre a partição de teste retida e a latência por amostra e as agrega na coluna de total. Em D3, LGBM_t e XGB_t exigem 13.45 s e 14.67 s de treinamento do modelo final, respectivamente, enquanto TabNet_t, TabTr_t e FTTr_t consomem 818.46 s, 672.85 s e 423.33 s sem melhorar F1 ou PR-AUC. D1 e D2 repetem o padrão: ali os modelos tabulares profundos também demandam tempos de treinamento substancialmente maiores, e ainda operam em regime já saturado. Nesta avaliação, portanto, as árvores impulsionadas definem a linha de base tabular isolada mais forte tanto em acurácia quanto em eficiência.

Tabela 5.7 – Eficiência computacional das linhas de base tabulares isoladas em D1–D3

Conjunto	Modelo	OOF (s)	Treino (s)	Inferência (s)	ms/amostra	Total (s)
D1	RF _t	173.78	46.49	0.2941	0.0062	220.81
D1	XGB _t	41.03	9.39	0.1231	0.0026	50.79
D1	LGBM _t	31.57	7.42	0.3629	0.0077	39.59
D1	TabNet _t	1764.20	541.80	0.8898	0.0189	2307.13
D1	TabTr _t	1499.04	516.98	0.9260	0.0196	2017.18
D1	FTTr _t	1207.67	415.99	0.8116	0.0172	1624.71
D2	RF _t	84.62	23.12	0.3596	0.0053	108.55
D2	XGB _t	65.08	15.14	0.1815	0.0027	80.83
D2	LGBM _t	51.36	11.81	0.6914	0.0103	64.30
D2	TabNet _t	2325.28	783.84	1.2574	0.0187	3110.82
D2	TabTr _t	2962.03	818.64	1.3635	0.0202	3782.46
D2	FTTr _t	2703.47	651.28	1.1480	0.0170	3356.34
D3	RF _t	57.20	14.91	0.4534	0.0102	72.83
D3	XGB _t	65.30	14.67	0.2732	0.0062	80.51
D3	LGBM _t	65.64	13.45	0.9042	0.0204	80.25
D3	TabNet _t	1721.20	818.46	1.0515	0.0238	2540.98
D3	TabTr _t	1468.75	672.85	0.6374	0.0144	2142.49
D3	FTTr _t	1806.92	423.33	0.7043	0.0159	2231.22

5.6 ANÁLISE DE COMPONENTES E BLOCOS DE ATRIBUTOS

A Tabela 5.8 examina se o próprio espaço de atributos tabulares já explica a maior parte da separação de classes antes da introdução dos ramos lexicais e do meta-aprendizado. A análise é conduzida por blocos: F_{URL} , contendo apenas atributos derivados da URL; F_{HTML} , contendo apenas atributos derivados da página; e $F_{\text{URL+HTML}}$, com a representação combinada. Essa decomposição permite identificar em qual conjunto de dados existe, de fato, informação residual passível de ser explorada por combinação de evidências.

Os resultados mostram que D1 é efetivamente saturado por atributos tabulares derivados da URL: F_{URL} alcança ao menos 99.99% de F1 para todos os modelos avaliados, com AUC e PR-AUC igualmente no limite superior. A adição de atributos derivados de HTML não produz, portanto, margem significativa

Tabela 5.8 – Desempenho dos modelos tabulares em D1–D3 por bloco de atributos

Conjunto	Modelo	Atributos	ACC (%)	Pre (%)	F1 (%)	Recall (%)	AUC (%)	PR-AUC (%)
D1	RF _t	F_{URL}	99.99	100.00	99.99	99.98	99.99	99.99
		F_{HTML}	99.71	99.71	99.66	99.60	99.99	99.99
		$F_{URL+HTML}$	100.00	100.00	100.00	100.00	100.00	100.00
	XGB _t	F_{URL}	99.99	100.00	99.99	99.98	99.99	99.99
		F_{HTML}	99.81	99.84	99.77	99.71	99.99	99.99
		$F_{URL+HTML}$	100.00	100.00	100.00	99.99	100.00	100.00
	LGBM _t	F_{URL}	99.99	100.00	99.99	99.98	99.99	99.99
		F_{HTML}	99.85	99.89	99.82	99.76	99.99	99.99
		$F_{URL+HTML}$	100.00	100.00	100.00	100.00	100.00	100.00
	TabNet _t	F_{URL}	99.99	100.00	99.99	99.98	99.99	99.99
		F_{HTML}	96.71	99.94	96.01	92.37	99.97	99.96
		$F_{URL+HTML}$	99.99	100.00	99.99	99.98	100.00	100.00
	TabTr _t	F_{URL}	99.99	100.00	99.99	99.98	99.99	99.99
		F_{HTML}	99.67	99.91	99.61	99.32	99.99	99.99
		$F_{URL+HTML}$	99.44	98.70	99.35	100.00	100.00	100.00
	FTTr _t	F_{URL}	99.99	100.00	99.99	99.98	99.99	100.00
		F_{HTML}	99.66	99.59	99.61	99.62	99.99	99.99
		$F_{URL+HTML}$	100.00	100.00	100.00	100.00	100.00	100.00
D2	RF _t	F_{URL}	99.78	99.97	99.79	99.61	99.96	99.97
		F_{HTML}	99.39	99.37	99.41	99.46	99.93	99.95
		$F_{URL+HTML}$	99.90	99.95	99.90	99.85	100.00	99.99
	XGB _t	F_{URL}	99.77	99.95	99.77	99.60	99.97	99.98
		F_{HTML}	99.35	99.32	99.38	99.44	99.93	99.95
		$F_{URL+HTML}$	99.95	99.97	99.95	99.94	100.00	100.00
	LGBM _t	F_{URL}	99.78	99.97	99.78	99.60	99.97	99.98
		F_{HTML}	99.36	99.39	99.39	99.39	99.93	99.94
		$F_{URL+HTML}$	99.96	99.97	99.96	99.96	100.00	100.00
	TabNet _t	F_{URL}	99.79	99.98	99.79	99.61	99.97	99.97
		F_{HTML}	97.97	97.08	98.08	99.09	99.48	99.36
		$F_{URL+HTML}$	99.82	99.95	99.83	99.72	100.00	100.00
	TabTr _t	F_{URL}	99.80	99.97	99.80	99.64	99.97	99.98
		F_{HTML}	98.33	97.51	98.42	99.35	99.73	99.73
		$F_{URL+HTML}$	99.88	99.90	99.88	99.87	100.00	100.00
	FTTr _t	F_{URL}	99.79	99.95	99.79	99.64	99.97	99.98
		F_{HTML}	98.27	97.45	98.36	99.28	99.71	99.71
		$F_{URL+HTML}$	99.85	99.94	99.85	99.76	100.00	100.00
D3	RF _t	F_{URL}	96.44	97.71	96.34	95.00	99.28	99.39
		F_{HTML}	96.21	96.76	96.13	95.51	99.28	99.30
		$F_{URL+HTML}$	98.39	98.84	98.36	97.88	99.84	99.85
	XGB _t	F_{URL}	96.41	97.87	96.30	94.77	99.20	99.34
		F_{HTML}	96.70	96.86	96.64	96.42	99.24	99.27
		$F_{URL+HTML}$	98.72	99.11	98.70	98.29	99.88	99.89
	LGBM _t	F_{URL}	96.48	97.81	96.37	94.98	99.28	99.39
		F_{HTML}	96.16	96.39	96.09	95.80	99.13	99.19
		$F_{URL+HTML}$	98.68	99.11	98.65	98.20	99.88	99.90
	TabNet _t	F_{URL}	96.15	97.23	96.05	94.89	99.19	99.29
		F_{HTML}	95.39	95.71	95.30	94.89	98.61	98.74
		$F_{URL+HTML}$	98.21	98.87	98.17	97.47	99.77	99.80
	TabTr _t	F_{URL}	95.97	97.19	95.86	94.56	99.14	99.25
		F_{HTML}	95.04	94.83	94.97	95.11	98.36	98.51
		$F_{URL+HTML}$	97.96	98.12	97.93	97.74	99.73	99.77
	FTTr _t	F_{URL}	96.10	97.61	95.98	94.39	99.16	99.28
		F_{HTML}	94.85	95.09	94.75	94.42	98.47	98.64
		$F_{URL+HTML}$	98.12	98.35	98.09	97.82	99.75	99.78

de discriminação adicional, uma vez que a fronteira de decisão já está quase inteiramente determinada pela evidência tabular em nível de URL. O conjunto D2 segue o mesmo padrão, embora de forma menos extrema. O bloco somente de URL produz valores de F1 em torno de 99.77% a 99.80%, e a configuração combinada $F_{URL+HTML}$ eleva os modelos baseados em árvores mais fortes apenas para a faixa de 99.95% a 99.96% de F1. Esses resultados indicam que D1 e D2 não são limitados primariamente pela capacidade do modelo nem pela ausência de um meta-aprendiz: são *benchmarks* nos quais os atributos tabulares projetados já respondem pela maior parte da separabilidade.

O conjunto D3 é o único em que a análise por blocos expõe margem residual mais ampla. Nele, os blocos isolados de URL e de HTML permanecem abaixo da saturação: os modelos baseados em árvores obtêm aproximadamente 96.30% a 96.37% de F1 com F_{URL} e 96.09% a 96.64% com F_{HTML} , enquanto a representação combinada eleva o F1 para 98.36% em RF_t , 98.70% em XGB_t e 98.65% em $LGBM_t$. Os modelos tabulares neurais apresentam a mesma direção, melhorando de valores de F1 entre 94.75% e 96.05% nos blocos isolados para 97.93% a 98.17% sob $F_{URL+HTML}$. O ganho consistente obtido pela união dos blocos, e não por um deles isoladamente, é a evidência direta de que URL e HTML carregam sinais parcialmente independentes em D3.

Essa constatação sustenta a interpretação central do arcabouço. Onde os atributos tabulares já saturam o *benchmark*, como em D1 e, em larga medida, em D2, resta pouca incerteza residual para a camada de meta-aprendizado explorar. Onde os blocos de atributos permanecem individualmente incompletos, como em D3, a combinação de evidências complementares ganha valor informativo e cria as condições sob as quais o empilhamento produz ganhos mensuráveis.

5.7 CONTRIBUIÇÃO E SENSIBILIDADE DO ESCORE URLC

A análise do URLC avalia se o escore composto proposto acrescenta informação mensurável além dos demais atributos tabulares disponíveis em D3. Como o URLC é um atributo introduzido como parte do conjunto HybridPhisher, a avaliação restringe-se a esse *benchmark*. São reportadas três visões complementares: desempenho absoluto com e sem o URLC, diferenças pareadas de desempenho sobre sementes correspondentes e sensibilidade do esquema de ponderação do escore.

As Tabelas 5.9 e 5.10 mostram que a contribuição do URLC depende do bloco de atributos que o circunda. Na primeira, os valores são diferenças pareadas em pontos percentuais entre as configurações com e sem o descritor, de modo que valores positivos indicam contribuição discriminativa acrescentada por ele, e o intervalo de confiança de 95% refere-se ao $\Delta F1$ em nível de semente; a segunda apresenta o desempenho absoluto das mesmas configurações correspondentes. Quando o URLC é adicionado a F_{URL} , todos os seis modelos tabulares melhoram em F1 nas execuções correspondentes. Os ganhos variam de +0.24 ponto percentual em $TabTr_t$ a +0.38 ponto percentual em XGB_t , e todos os intervalos de confiança permanecem estritamente positivos. Isso indica que o URLC captura parte da estrutura em nível de URL que não é plenamente representada pelos demais atributos derivados da URL, provavelmente porque sintetiza, em uma única dimensão contínua, relações entre componentes que os atributos individuais expressam apenas de forma fragmentada.

Tabela 5.9 – Contribuição discriminativa incremental do URLC por bloco de atributos tabulares no HybridPhisher

Bloco de atributos	Modelo	ΔACC	ΔPre	ΔF1	ΔRecall	ΔAUC	$\Delta\text{PR-AUC}$	IC 95% ΔF1	p_{perm}	Execuções
F_{URL}	RF_t	+0.34	+0.12	+0.36	+0.59	+0.15	+0.13	[+0.33, +0.40]	0.0016	10
F_{URL}	XGB_t	+0.36	+0.10	+0.38	+0.64	+0.19	+0.14	[+0.36, +0.41]	0.0016	10
F_{URL}	LGBM_t	+0.35	+0.16	+0.37	+0.56	+0.15	+0.12	[+0.33, +0.41]	0.0016	10
F_{URL}	TabNet_t	+0.33	-0.01	+0.36	+0.69	+0.14	+0.11	[+0.31, +0.40]	0.0016	10
F_{URL}	TabTr_t	+0.22	-0.06	+0.24	+0.53	+0.13	+0.10	[+0.20, +0.28]	0.0016	10
F_{URL}	FTTr_t	+0.33	-0.11	+0.36	+0.80	+0.15	+0.12	[+0.30, +0.42]	0.0016	10
F_{HTML}	RF_t	+0.31	+0.38	+0.31	+0.25	+0.19	+0.21	[+0.27, +0.35]	0.0016	10
F_{HTML}	XGB_t	+0.22	+0.52	+0.21	-0.09	+0.22	+0.23	[+0.17, +0.26]	0.0016	10
F_{HTML}	LGBM_t	+0.51	+0.84	+0.51	+0.19	+0.30	+0.29	[+0.48, +0.54]	0.0016	10
F_{HTML}	TabNet_t	+0.52	+0.69	+0.52	+0.34	+0.47	+0.43	[+0.37, +0.66]	0.0016	10
F_{HTML}	TabTr_t	+0.64	+1.13	+0.63	+0.11	+0.53	+0.45	[+0.49, +0.75]	0.0016	10
F_{HTML}	FTTr_t	+0.67	+0.68	+0.68	+0.69	+0.50	+0.43	[+0.62, +0.74]	0.0016	10
$F_{\text{URL+HTML}}$	RF_t	-0.03	+0.00	-0.03	-0.06	+0.00	+0.00	[-0.04, -0.02]	0.0075	10
$F_{\text{URL+HTML}}$	XGB_t	+0.03	+0.03	+0.03	+0.04	+0.01	+0.01	[+0.02, +0.05]	0.0034	10
$F_{\text{URL+HTML}}$	LGBM_t	+0.02	+0.00	+0.02	+0.04	+0.01	+0.00	[+0.01, +0.04]	0.0218	10
$F_{\text{URL+HTML}}$	TabNet_t	+0.02	+0.17	+0.02	-0.14	+0.02	+0.01	[-0.05, +0.09]	0.6657	10
$F_{\text{URL+HTML}}$	TabTr_t	+0.02	-0.09	+0.02	+0.14	-0.00	+0.00	[-0.06, +0.10]	0.5745	10
$F_{\text{URL+HTML}}$	FTTr_t	-0.01	+0.03	-0.02	-0.06	+0.01	+0.01	[-0.07, +0.04]	0.5873	10

O bloco F_{HTML} repete o padrão, e de forma menos intuitiva. Embora o URLC nasça exclusivamente da cadeia de caracteres da URL, ele melhora as configurações derivadas de HTML em todos os modelos avaliados, com ganhos de F1 entre +0.21 e +0.68 ponto percentual. Os modelos tabulares neurais colhem os maiores ganhos, sobretudo FTTr_t , TabTr_t e TabNet_t . O resultado sugere que o URLC leva evidência de nível de URL a um modelo que aprende predominantemente sobre atributos da página, funcionando como ponte compacta entre a estrutura do endereço e o comportamento observado no HTML.

O efeito encolhe substancialmente na configuração completa $F_{\text{URL+HTML}}$. Com atributos de URL e de HTML já à disposição, a contribuição adicional do URLC torna-se marginal e passa a depender do modelo. XGB_t e LGBM_t registram pequenos ganhos positivos de F1, de +0.03 e +0.02 ponto percentual, ao passo que RF_t e FTTr_t chegam a diferenças médias levemente negativas. Nos modelos tabulares neurais, os intervalos de confiança incluem zero, sinal de que o valor agregado do URLC perde estabilidade na representação plenamente combinada. Nada disso invalida o atributo. Mostra apenas que parte da informação que ele sintetiza se sobrepõe a outros atributos de URL e de HTML quando o conjunto completo entra em jogo, situação típica de descritores agregados em espaços de atributos redundantes.

A Tabela 5.11 acrescenta uma terceira leitura: o URLC responde à ênfase relativa entre seus componentes, mas nenhuma parametrização exata o domina. Ela avalia oito esquemas de pesos diretamente sobre os quatro componentes do score, com a AUC orientada de modo que valores maiores indiquem evidência mais forte da classe positiva, e reporta ainda a taxa de verdadeiros positivos (*True Positive Rate*, TPR) no ponto de operação em que a FPR é fixada em 0,05. O esquema *mix-heavy* obtém a maior AUC univariada, 90.10%, enquanto o esquema arredondado proposto alcança 89.58% e fica muito próximo do esquema em-

Tabela 5.10 – Desempenho tabular com e sem o URLC no HybridPhisher

Bloco de atributos	Modelo	URLC	ACC	Pre	F1	Recall	AUC	PR-AUC	Execuções
F_{URL}	RF _t	Com URLC	96.46	97.79	96.35	94.96	99.31	99.40	10
F_{URL}	RF _t	Sem URLC	96.11	97.67	95.99	94.37	99.15	99.28	10
F_{URL}	XGB _t	Com URLC	96.44	97.95	96.33	94.75	99.25	99.37	10
F_{URL}	XGB _t	Sem URLC	96.08	97.86	95.95	94.11	99.07	99.23	10
F_{URL}	LGBM _t	Com URLC	96.41	97.80	96.30	94.84	99.32	99.41	10
F_{URL}	LGBM _t	Sem URLC	96.06	97.64	95.94	94.29	99.17	99.29	10
F_{URL}	TabNet _t	Com URLC	96.08	97.27	95.96	94.69	99.19	99.30	10
F_{URL}	TabNet _t	Sem URLC	95.75	97.28	95.61	94.00	99.05	99.19	10
F_{URL}	TabTr _t	Com URLC	95.97	97.19	95.85	94.55	99.17	99.28	10
F_{URL}	TabTr _t	Sem URLC	95.75	97.25	95.61	94.02	99.04	99.18	10
F_{URL}	FTTr _t	Com URLC	95.97	97.27	95.85	94.47	99.18	99.29	10
F_{URL}	FTTr _t	Sem URLC	95.64	97.38	95.49	93.67	99.03	99.17	10
F_{HTML}	RF _t	Com URLC	96.52	97.07	96.44	95.83	99.45	99.49	10
F_{HTML}	RF _t	Sem URLC	96.21	96.70	96.13	95.57	99.26	99.29	10
F_{HTML}	XGB _t	Com URLC	96.89	97.29	96.83	96.37	99.45	99.50	10
F_{HTML}	XGB _t	Sem URLC	96.67	96.77	96.61	96.46	99.23	99.27	10
F_{HTML}	LGBM _t	Com URLC	96.62	97.05	96.55	96.06	99.42	99.47	10
F_{HTML}	LGBM _t	Sem URLC	96.11	96.22	96.04	95.87	99.11	99.17	10
F_{HTML}	TabNet _t	Com URLC	95.83	95.96	95.76	95.56	99.07	99.13	10
F_{HTML}	TabNet _t	Sem URLC	95.31	95.27	95.24	95.22	98.60	98.70	10
F_{HTML}	TabTr _t	Com URLC	95.58	95.67	95.51	95.36	98.96	99.06	10
F_{HTML}	TabTr _t	Sem URLC	94.94	94.54	94.89	95.24	98.44	98.61	10
F_{HTML}	FTTr _t	Com URLC	95.58	95.75	95.50	95.25	98.99	99.09	10
F_{HTML}	FTTr _t	Sem URLC	94.91	95.07	94.81	94.56	98.49	98.66	10
$F_{URL+HTML}$	RF _t	Com URLC	98.31	98.74	98.28	97.82	99.84	99.85	10
$F_{URL+HTML}$	RF _t	Sem URLC	98.34	98.74	98.31	97.89	99.84	99.85	10
$F_{URL+HTML}$	XGB _t	Com URLC	98.72	99.00	98.70	98.40	99.88	99.89	10
$F_{URL+HTML}$	XGB _t	Sem URLC	98.69	98.97	98.66	98.36	99.87	99.88	10
$F_{URL+HTML}$	LGBM _t	Com URLC	98.67	98.97	98.65	98.33	99.88	99.89	10
$F_{URL+HTML}$	LGBM _t	Sem URLC	98.65	98.96	98.62	98.29	99.87	99.88	10
$F_{URL+HTML}$	TabNet _t	Com URLC	98.21	98.53	98.18	97.83	99.76	99.79	10
$F_{URL+HTML}$	TabNet _t	Sem URLC	98.19	98.35	98.16	97.97	99.74	99.78	10
$F_{URL+HTML}$	TabTr _t	Com URLC	98.05	98.01	98.02	98.03	99.73	99.77	10
$F_{URL+HTML}$	TabTr _t	Sem URLC	98.02	98.10	97.99	97.89	99.73	99.77	10
$F_{URL+HTML}$	FTTr _t	Com URLC	98.11	98.12	98.08	98.05	99.76	99.79	10
$F_{URL+HTML}$	FTTr _t	Sem URLC	98.13	98.09	98.10	98.11	99.75	99.78	10

pírico exato, 89.52%. Ao longo de todos os esquemas testados, a AUC oscila em faixa estreita, de 88.60% a 90.10%, e a estatística de Kolmogorov–Smirnov (KS) permanece elevada. O conceito do URLC não depende, portanto, de um conjunto frágil e único de pesos. Em contrapartida, esses mesmos números situam a formulação arredondada como escolha operacional voltada à interpretabilidade, e não como esquema de ponderação matematicamente ótimo.

Tabela 5.11 – Análise de sensibilidade dos pesos do URLC no HybridPhisher

Esquema	w_{mix}	w_{maxtok}	w_{subdom}	w_{query}	AUC	IC 95% AUC	KS	TPR@FPR=0.05
<i>Mix-heavy</i>	0.55	0.15	0.20	0.10	90.10	[89.97, 90.22]	0.6385	0.5810
Arredondado proposto	0.40	0.25	0.25	0.10	89.58	[89.46, 89.71]	0.6384	0.5701
Empírico exato	0.36	0.26	0.25	0.13	89.52	[89.40, 89.65]	0.6377	0.5662
<i>No-query</i>	0.45	0.28	0.28	0.00	89.31	[89.19, 89.44]	0.6375	0.5655
<i>Query-heavy</i>	0.25	0.20	0.20	0.35	89.28	[89.16, 89.42]	0.6323	0.5696
Pesos iguais	0.25	0.25	0.25	0.25	89.17	[89.05, 89.30]	0.6327	0.5610
<i>Subdomain-heavy</i>	0.25	0.20	0.40	0.15	89.13	[89.00, 89.26]	0.6422	0.5533
<i>Max-token-heavy</i>	0.25	0.45	0.20	0.10	88.60	[88.47, 88.73]	0.6207	0.5450

O URLC rende mais, em suma, quando o bloco de atributos disponível ainda não reúne um conjunto amplo de descritores de URL e de HTML. Ele contribui de forma consistente nas configurações tabulares restritas a URL ou a HTML, e encolhe até a redundância na representação combinada completa. Cabe lê-lo, portanto, como resumo estrutural compacto que socorre cenários de atributos reduzidos e acrescenta informação em contextos menos saturados, não como detector isolado suficiente nem como atributo universalmente dominante.

5.8 ANÁLISE DE ESTRESSE ADVERSARIAL

A Tabela 5.12 e a Figura 5.5 reportam um teste de estresse adversarial controlado no HybridPhisher. O experimento avalia se a perturbação da evidência observável de URL ou de HTML altera o comportamento dos ramos isolados e dos meta-modelos empilhados. Essa análise complementa os resultados limpos de execuções repetidas ao medir a sensibilidade do *pipeline* sob famílias determinísticas de perturbação configuradas no manifesto do experimento, e não ao propor um novo ataque adversarial.

As perturbações alcançaram somente as amostras de *phishing* da partição de teste retida, e nem os dados de treinamento nem a configuração do *pipeline* sofreram alteração. A restrição tem peso metodológico: ela isola o efeito da perturbação sobre a capacidade de detecção e impede que o modelo se adapte às modificações durante o treinamento.

Em nível de URL, os candidatos abrangeram substituição de caracteres no rótulo de *host* registrável, substituição homográfica em nomes de domínio internacionalizados (*Internationalized Domain Names in Applications*, IDNA), injeção de subdomínios relacionados à segurança e ofuscação de caminho e de consulta. Cada candidato passou por validação sintática; o procedimento escolheu a primeira variante determinística válida que o manifesto do experimento habilitava e, em seguida, recomputou os atributos

derivados da URL. Em nível de HTML, os candidatos abrangeram modificação da ação de formulários, injeção de elementos ocultos, ofuscação de *links* em âncoras e inserção de *scripts* e recursos fictícios nas cópias armazenadas das páginas. Também aqui o procedimento aplicou a primeira transformação determinística possível, guardou a cópia modificada para auditoria e recomputou todos os atributos de HTML afetados antes da inferência.

A Tabela 5.12 reúne quatro medidas para cada condição. As variações de Recall e de escore são diferenças entre a condição adversarial e a condição limpa, no limiar fixo de 0.5. A taxa de falsos negativos (*False Negative Rate*, FNR) considera todas as amostras de *phishing* após a perturbação. Já a taxa de sucesso de ataque condicional (*Attack Success Rate*, ASR) mede a fração das amostras de *phishing* corretamente detectadas na condição limpa que se tornam falsos negativos depois dela.

Tabela 5.12 – Desempenho do teste de estresse adversarial em nível de *pipeline* no HybridPhisher

Atributos	Modelo	Recall limpo	Recall adv.	Δ Recall	FNR adv.	ASR cond.	Δ Escore
Nível de URL	RF _t	97.89	97.91	0.03	2.09	1.57	-3.96
Nível de URL	XGB _t	98.29	97.89	-0.41	2.11	1.63	-1.01
Nível de URL	LGBM _t	98.20	98.18	-0.02	1.82	1.36	-1.14
Nível de URL	LR _c	97.49	96.68	-0.80	3.32	1.16	-1.42
Nível de URL	CNN _c	97.10	96.82	-0.28	3.18	0.68	-0.34
Nível de URL	CNN _w	97.59	85.71	-11.89	14.29	12.41	-9.79
Nível de URL	LGBM _m	99.09	98.98	-0.11	1.02	0.54	-0.46
Nível de URL	LR _m	99.08	98.99	-0.09	1.01	0.49	-0.38
Nível de HTML	RF _t	97.89	89.61	-8.27	10.39	8.47	-12.74
Nível de HTML	XGB _t	98.29	92.40	-5.90	7.60	6.15	-6.81
Nível de HTML	LGBM _t	98.20	91.44	-6.76	8.56	7.00	-7.36
Nível de HTML	LR _c	97.49	97.49	0.00	2.51	0.00	0.00
Nível de HTML	CNN _c	97.10	97.10	0.00	2.90	0.00	0.00
Nível de HTML	CNN _w	97.59	97.59	0.00	2.41	0.00	0.00
Nível de HTML	LGBM _m	99.09	98.50	-0.59	1.50	0.67	-0.63
Nível de HTML	LR _m	99.08	98.67	-0.41	1.33	0.45	-0.68

Sob perturbações em nível de URL, os meta-modelos empilhados permanecem os mais estáveis. A variação de Recall limita-se a -0.11 ponto percentual (p.p.) para LGBM_m e -0.09 p.p. para LR_m, com ASR inferior a 0.6%. Os modelos baseados em caracteres também permanecem relativamente estáveis, ao passo que CNN_w é o ramo mais afetado, com queda de 11.89 p.p. em Recall e ASR de 12.41%. Essa assimetria tem explicação direta: o modelo em nível de palavra depende de um vocabulário fixo de *tokens*, de modo que substituições de caracteres e injeções de subdomínio deslocam a entrada para termos fora do vocabulário, degradando a representação. Os modelos tabulares apresentam deslocamentos de escore maiores que os meta-modelos, mas seu Recall permanece próximo à condição limpa, o que indica que as perturbações de URL reduzem a confiança mais do que induzem consistentemente falsos negativos nesses ramos.

As perturbações em nível de HTML produzem padrão distinto. Como os ramos lexicais de URL não consomem evidência derivada de HTML, suas métricas permanecem inalteradas nessa condição. Em contraste, os ramos tabulares são diretamente afetados: o Recall diminui 8.27 p.p. para RF_t , 5.90 p.p. para XGB_t e 6.76 p.p. para $LGBM_t$, com valores correspondentes de ASR entre 6.15% e 8.47%. Apesar dessa degradação na camada tabular, os meta-modelos empilhados absorvem a maior parte do efeito da perturbação, com perdas de Recall de 0.59 p.p. para $LGBM_m$ e 0.41 p.p. para LR_m . Esse comportamento é justamente o esperado de uma arquitetura heterogênea: quando uma fonte de evidência é comprometida, a camada de fusão pode se apoiar nas fontes preservadas, o que atenua a queda global de desempenho.

O mapa de calor da Figura 5.5 fornece a mesma conclusão sob perspectiva métrica mais ampla; cada célula reporta a variação, em pontos percentuais, entre as condições adversarial e limpa, e valores negativos indicam degradação. Os modelos tabulares isolados exibem as maiores degradações em acurácia e F1 sob perturbações em nível de URL, e as maiores perdas de Recall sob perturbações em nível de HTML. Em contraste, os meta-modelos apresentam variações pequenas em acurácia, F1, Recall e PR-AUC em ambas as condições adversariais. A Figura 5.6 complementa essa leitura ao ligar, para cada modelo, o Recall na condição limpa aos valores sob perturbação de URL e de HTML: inclinações descendentes acentuadas revelam os ramos que deixam de detectar amostras de *phishing*, e trajetórias praticamente planas, os que resistem.

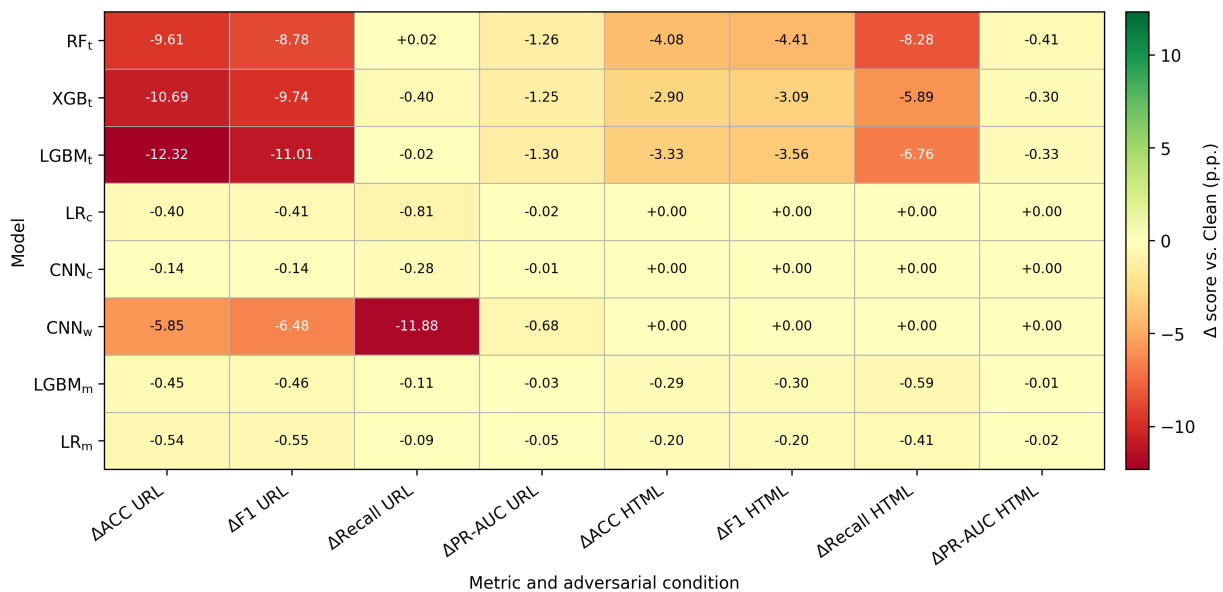


Figura 5.5 – Variações do teste de estresse adversarial em nível de *pipeline* no HybridPhisher

A camada empilhada, portanto, não elimina a sensibilidade adversarial: ela apenas amortece o impacto das perturbações, ao combinar fontes heterogêneas de evidência e compensar a degradação específica de cada ramo. O que se ganha é estabilidade relativa, não garantia de robustez.

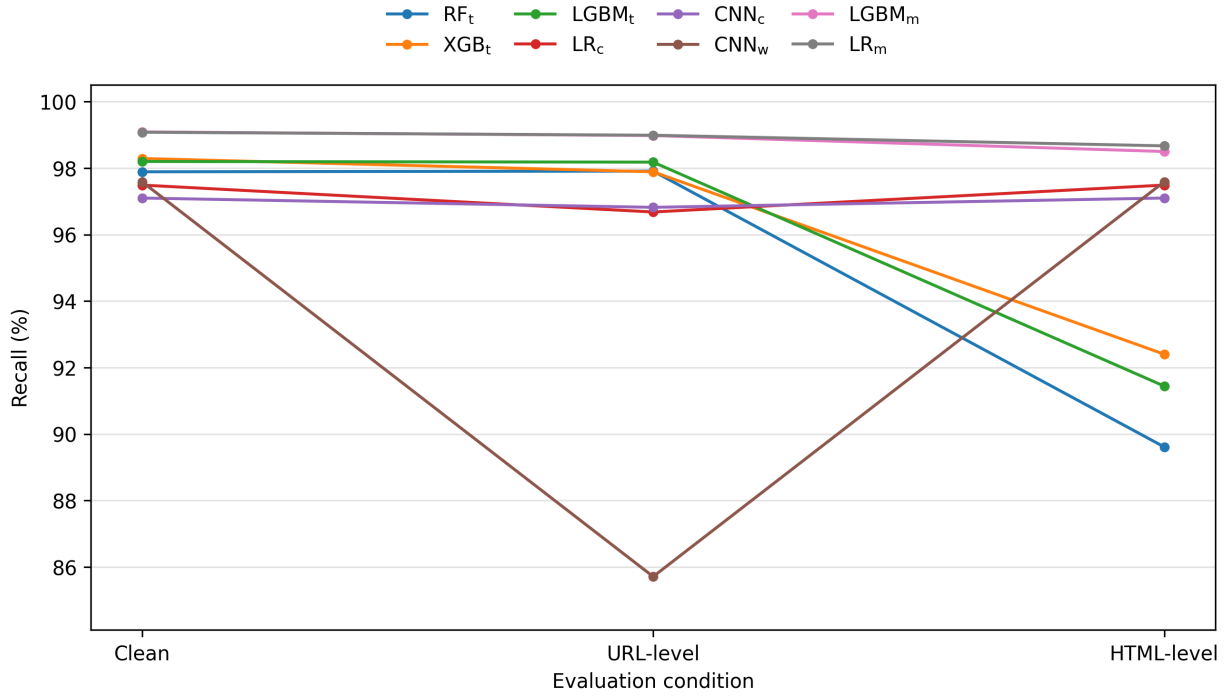


Figura 5.6 – Trajetória de Recall entre as condições limpa e adversariais no HybridPhisher

5.9 CONTRIBUIÇÃO DA SIMILARIDADE DE DOMÍNIO

A Tabela 5.13 avalia se o atributo `Domain_Similarity` fornece informação discriminativa além dos demais atributos tabulares do HybridPhisher. Diferentemente das demais análises deste capítulo, conduzidas sobre dez sementes, esta avaliação apoia-se em três execuções pareadas, em razão do orçamento experimental disponível: um ciclo completo de execuções repetidas demanda aproximadamente uma semana de processamento contínuo na infraestrutura descrita na Seção 5.1. Como se verá adiante, essa redução afeta a precisão das estimativas, mas não compromete a direção do efeito observado, uma vez que os intervalos de confiança de $\Delta F1$ permanecem estritamente positivos nas dezoito configurações avaliadas. Como nas análises do URLC, os valores reportados são diferenças médias pareadas, em pontos percentuais, entre as configurações com e sem o atributo, e o intervalo de confiança de 95% refere-se ao $\Delta F1$ em nível de semente. Os resultados indicam que esse descritor é informativo em todos os blocos de atributos e famílias de modelos, uma vez que sua inclusão melhora de forma consistente tanto métricas dependentes de limiar quanto métricas orientadas a ordenação.

Sua contribuição discriminativa é mais forte quando a representação disponível está restrita a uma única fonte de evidência. No bloco somente de URL, o maior ganho de F1 é obtido por LGBM_t (+1.13), acompanhado do maior ganho de Recall (+1.91), enquanto XGB_t e FTTr_t apresentam os maiores ganhos de PR-AUC (+0.56 e +0.55, respectivamente). Isso indica que o atributo auxilia tanto aprendizes de *boosting* quanto aprendizes tabulares baseados em *Transformer* a separar domínios legítimos de domínios de *phishing* quando a decisão se apoia primordialmente em evidência derivada da URL.

O bloco de HTML também acolhe bem o atributo. Ali, FTTr_t colhe os maiores ganhos de F1 e de acurácia (+1.25 e +1.24), enquanto TabTr_t avança mais em precisão (+1.97), AUC (+0.63) e PR-

Tabela 5.13 – Contribuição discriminativa incremental da similaridade de domínio por bloco de atributos tabulares no HybridPhisher

Bloco de atributos	Modelo	ΔACC	ΔPre	ΔF1	ΔRecall	ΔAUC	$\Delta\text{PR-AUC}$	IC 95% ΔF1	Execuções
F_{URL}	RF_t	+0.95	+0.21	+1.02	+1.77	+0.74	+0.52	[+0.92, +1.09]	3
F_{URL}	XGB_t	+1.01	+0.22	+1.08	+1.87	+0.80	+0.56	[+1.07, +1.08]	3
F_{URL}	LGBM_t	+1.06	+0.29	+1.13	+1.91	+0.73	+0.52	[+1.07, +1.23]	3
F_{URL}	TabNet_t	+0.83	-0.06	+0.91	+1.81	+0.71	+0.49	[+0.80, +0.96]	3
F_{URL}	TabTr_t	+0.85	+0.63	+0.88	+1.13	+0.74	+0.51	[+0.79, +0.93]	3
F_{URL}	FTTr_t	+1.05	+0.33	+1.12	+1.85	+0.79	+0.55	[+0.99, +1.26]	3
F_{HTML}	RF_t	+0.96	+1.13	+0.98	+0.83	+0.31	+0.31	[+0.90, +1.02]	3
F_{HTML}	XGB_t	+0.86	+1.20	+0.86	+0.54	+0.31	+0.30	[+0.82, +0.89]	3
F_{HTML}	LGBM_t	+1.15	+1.49	+1.16	+0.83	+0.40	+0.38	[+1.08, +1.21]	3
F_{HTML}	TabNet_t	+1.06	+1.47	+1.07	+0.67	+0.51	+0.47	[+0.89, +1.36]	3
F_{HTML}	TabTr_t	+1.18	+1.97	+1.17	+0.38	+0.63	+0.57	[+1.03, +1.28]	3
F_{HTML}	FTTr_t	+1.24	+1.50	+1.25	+1.01	+0.62	+0.54	[+0.99, +1.40]	3
$F_{\text{URL}+\text{HTML}}$	RF_t	+0.33	+0.36	+0.33	+0.30	+0.07	+0.06	[+0.24, +0.40]	3
$F_{\text{URL}+\text{HTML}}$	XGB_t	+0.31	+0.13	+0.32	+0.50	+0.06	+0.05	[+0.29, +0.36]	3
$F_{\text{URL}+\text{HTML}}$	LGBM_t	+0.31	+0.18	+0.32	+0.45	+0.06	+0.05	[+0.29, +0.35]	3
$F_{\text{URL}+\text{HTML}}$	TabNet_t	+0.30	+0.31	+0.30	+0.30	+0.08	+0.07	[+0.26, +0.35]	3
$F_{\text{URL}+\text{HTML}}$	TabTr_t	+0.34	+0.31	+0.35	+0.39	+0.10	+0.09	[+0.25, +0.46]	3
$F_{\text{URL}+\text{HTML}}$	FTTr_t	+0.19	-0.11	+0.20	+0.50	+0.07	+0.06	[+0.10, +0.34]	3

AUC (+0.57). A similaridade em nível de domínio parece, assim, complementar os atributos da página com evidência de consistência de identidade que a estrutura do HTML, sozinha, não alcança. No bloco combinado $F_{\text{URL}+\text{HTML}}$, os ganhos continuam positivos, mas encolhem bastante, e a maior melhoria fica com TabTr_t em F1 (+0.35), AUC (+0.10) e PR-AUC (+0.09). *Domain_Similarity* não é, portanto, mero atributo redundante: ele discrimina, sobretudo sob representações reduzidas a URL ou a HTML, e sua contribuição residual na representação completa apenas denuncia sobreposição parcial com os demais sinais de URL e de HTML.

5.10 DISCUSSÃO DOS RESULTADOS

Esta seção consolida as evidências apresentadas ao longo do capítulo e as confronta com a literatura revisada nos Capítulos 2 e 3.

5.10.1 Síntese das Evidências

As seções anteriores convergem para uma leitura unificada: o nível de saturação do *benchmark* emerge, nos três conjuntos avaliados, como o principal fator associado à magnitude do ganho obtido pelo empilhamento, e a decomposição por blocos de atributos identifica a origem dessa margem residual. O confronto com linhas de base tabulares fortes afasta a hipótese de que o ganho viesse de referências fracas. Os des-

critérios compostos obedecem ao mesmo mecanismo dos blocos que os circundam, e o teste de estresse adversarial delimita o alcance da estabilidade observada. É esse conjunto de evidências que a subseção seguinte confronta com a literatura revisada.

5.10.2 Confronto com a Literatura

Os achados deste trabalho dialogam com a literatura em quatro pontos principais.

O primeiro diz respeito à interpretação de métricas elevadas em *benchmarks* de *phishing*. Estudos anteriores já haviam alertado que a engenharia de atributos pode introduzir separabilidade fortemente específica do conjunto de dados quando os descritores discriminativos estão muito alinhados ao processo de coleta (Skula; Kvet, 2024; Mohanty et al., 2024; Kustiawan; Ghauth, 2025b). Os resultados das Tabelas 5.6 e 5.8 fornecem evidência interna consistente com essa interpretação: em D1, o bloco F_{URL} isoladamente já produz ao menos 99.99% de F1 para todos os modelos avaliados, e cinco modelos tabulares distintos atingem $F1 = 1.0000$ na configuração completa. Sob essas condições, o *benchmark* oferece margem muito limitada para discriminar entre arquiteturas; por isso, comparações conduzidas exclusivamente sobre ele devem ser interpretadas com cautela. Esse resultado sustenta a recomendação metodológica de reportar o nível de saturação do conjunto de dados como parte da avaliação, e não apenas as métricas agregadas.

O segundo ponto toca a força das linhas de base tabulares. A literatura relata desempenho consistentemente forte de modelos baseados em árvores e *boosting* na captura de interações não lineares entre atributos projetados (Karim et al., 2023; Nayak; Munial; Belavagi, 2025; Duarte et al., 2025; Kustiawan; Ghauth, 2025b; Chen; Guestrin, 2016). Os resultados aqui obtidos confirmam essa observação e a estendem em duas direções. A primeira: no espaço de atributos adotado, $LGBM_t$ e XGB_t superam as três arquiteturas tabulares profundas avaliadas nos três conjuntos. A segunda: essa superioridade preditiva custa uma a duas ordens de grandeza menos, como mostra a Tabela 5.7. Daí decorre uma implicação metodológica direta, em detecção de *phishing* com atributos tabulares projetados; a avaliação de componentes neurais deve incluir linhas de base de *boosting* devidamente otimizadas sob o mesmo protocolo, para que eventuais ganhos possam ser atribuídos à arquitetura e não à escolha da referência.

O terceiro ponto diz respeito ao empilhamento propriamente dito. A generalização empilhada é classicamente motivada pela exploração de padrões de decisão complementares entre aprendizes heterogêneos (Wolpert, 1992), e sua aplicação à detecção de *phishing* tem reportado desempenho elevado em diferentes combinações de classificadores, algoritmos de *boosting*, redes profundas e etapas de seleção de atributos (Aslam et al., 2024; Kalabarige et al., 2023; Bari et al., 2025; Kalabarige et al., 2022; Khalid et al., 2024; Motiur Rahman; Islam; Jabiullah, 2020; Indrasiri; Halgamuge; Mohammad, 2021). Boa parte desses estudos, entretanto, enfatiza métricas agregadas sem analisar explicitamente se o *benchmark* subjacente já está saturado por atributos isolados fortes. A contribuição deste trabalho, nesse ponto, não é reportar um valor de métrica superior, mas avaliar a saturação do *benchmark* para identificar a condição sob a qual o empilhamento é útil. As Tabelas 5.4 e 5.3 mostram que o mesmo arcabouço, sob protocolo idêntico, produz ganho nulo em D1 ($\Delta = 0.00000$; $p = 0.343436$), ganho estatisticamente detectável porém praticamente pequeno em D2 ($\Delta = 0.00015$) e ganho estável e perceptível em D3 ($\Delta = 0.00608$ sobre a melhor linha de base tabular). Essa gradação sustenta a leitura do empilhamento heterogêneo como mecanismo condicional de redução de erro residual, e não como arquitetura uniformemente superior.

O quarto ponto trata da robustez. Estudos de confiabilidade indicam que detectores de URL baseados em aprendizado de máquina degradam sob perturbações adversariais e manipulação de atributos (Sabir et al., 2024; Nowroozi et al., 2023; Pillai et al., 2024). A Tabela 5.12 confirma essa vulnerabilidade nos ramos isolados e acrescenta uma observação específica sobre a arquitetura empilhada: a degradação concentra-se no ramo cuja fonte de evidência sofreu a perturbação, enquanto a camada de fusão preserva desempenho apoiando-se nas fontes íntegras. O caso de CNN_w sob perturbação de URL, com queda de 11.89 p.p. em Recall, ilustra o problema de *tokens* fora do vocabulário que a literatura já aponta em modelos lexicais de nível de palavra (Almousa; Anwar, 2023; Sahingoz; Buber; Kugu, 2024; Sruthi; Naik S, 2025; He et al., 2024b), e contrasta com a variação de apenas -0.11 p.p. que $LGBM_m$ registra na mesma condição.

6 CONCLUSÕES

Esta pesquisa parte de uma questão metodológica central na detecção de *phishing* baseada em URL: métricas próximas ao limite superior podem ter baixo poder discriminativo entre arquiteturas quando o *benchmark* é quase separável sob a representação adotada. A literatura reporta, com regularidade, valores de desempenho junto ao limite superior da escala métrica, mas esses valores dizem pouco quando o conjunto de dados já é quase separável sob a representação de atributos adotada. Nessas condições, um escore quase perfeito pode denunciar saturação do *benchmark* em vez de anunciar vantagem arquitetural.

O problema atinge em cheio os detectores baseados em empilhamento, cuja justificativa teórica repousa sobre a existência de erro residual que evidências complementares consigam corrigir. Se os aprendizes isolados já esgotam a separabilidade do conjunto de dados, a margem disponível para melhoria torna-se reduzida, e ganhos marginais passam a exigir comparação rigorosa com linhas de base fortes para que possam ser atribuídos à meta-aprendizagem. Desse diagnóstico veio a pergunta que orientou o trabalho: sob quais condições o empilhamento heterogêneo produz ganho mensurável, estável e estatisticamente sustentado além de aprendizes isolados fortes treinados sobre o mesmo espaço de atributos?

Para respondê-la, o trabalho propôs e avaliou o HOOF-Stack-PUD, arcabouço de meta-aprendizado empilhado heterogêneo com predições fora da dobra. O arcabouço integra aprendizes tabulares, modelos lexicais de URL e ramos convolucionais sob um protocolo protegido contra vazamento, no qual os meta-aprendizes veem exclusivamente estimativas de probabilidade fora da dobra. Esse desenho separa de forma estrita três momentos que a literatura por vezes embaralha: o treinamento dos modelos de base, o ajuste da camada de meta-aprendizado e a avaliação final sobre a partição de teste retida. A avaliação empírica percorreu três *benchmarks* de grande porte com procedimentos de construção e espaços de atributos distintos, PhiUSIIL, StealthPhisher e HybridPhisher, este último construído no âmbito da pesquisa com 221.360 URLs a partir de fontes verificadas, e recorreu a comparações com linhas de base tabulares robustas, análise por blocos de atributos, execuções repetidas com testes estatísticos pareados, perfilamento de eficiência computacional e teste de estresse adversarial em nível de *pipeline*.

Os resultados oferecem uma resposta condicional à questão formulada. O benefício do empilhamento depende do nível de saturação do *benchmark*, e essa dependência apareceu de forma gradativa e coerente nos três conjuntos. Em PhiUSIIL, os modelos tabulares mais fortes já alcançam desempenho perfeito ou quase perfeito, e a camada de meta-aprendizado não acrescentou ganho ao melhor modelo tabular, que errou apenas quatro URLs em 471.590 predições nas dez sementes, com diferença pareada nula e valor p de 0.343436 sobre sementes correspondentes. Em StealthPhisher, o padrão se repete de forma menos extrema: a camada de meta-aprendizado produz melhoria estatisticamente sustentada, com $\Delta F1$ de 0.00015 e $p = 0.000257$, cuja magnitude prática permanece limitada porque o aprendiz tabular mais forte já opera junto ao teto da escala. Em HybridPhisher, onde os modelos isolados deixam erro residual mais amplo, o meta-aprendiz acrescentou ao F1 da melhor linha de base tabular um ganho de 0.00608 e ao da melhor linha de base lexical um ganho de 0.01021, com intervalos de confiança estritamente positivos, $p < 10^{-4}$ e F1 de 0.99304. Cabe registrar que o resultado nulo obtido em PhiUSIIL não denuncia falha do arcabouço nem do desenho experimental: ele constitui um achado relevante, pois indica que, sob o

protocolo adotado, *benchmarks* com desempenho próximo ao teto podem oferecer poder discriminativo limitado para comparar arquiteturas. Conjuntos nessa condição permanecem úteis para a verificação de correteza do *pipeline* e para a comparação com trabalhos anteriores, mas não autorizam conclusões sobre arquiteturas, e foi essa constatação que motivou a construção do HybridPhisher como conjunto adicional com ambiguidade residual preservada.

As análises de componentes esclareceram a origem dessa gradação e deram sustentação mecanicista à conclusão anterior. A decomposição por blocos de atributos evidenciou que, em PhiUSIIL e, em larga medida, em StealthPhisher, os atributos tabulares projetados respondem sozinhos por quase toda a separabilidade, de modo que a fronteira de decisão já está determinada antes mesmo de os ramos lexicais entrarem em cena. Em HybridPhisher, ao contrário, os blocos derivados de URL e de HTML permanecem individualmente incompletos, e só a união deles eleva o desempenho de forma expressiva. É precisamente essa incompletude parcial que gera a complementaridade explorada pela camada de meta-aprendizado. Nos três *benchmarks* avaliados, os resultados indicam que a utilidade do empilhamento depende não apenas da arquitetura, mas também da estrutura informacional do conjunto de dados sobre o qual ela opera.

A comparação com linhas de base tabulares robustas afastou a hipótese de que os ganhos observados viessem da escolha de referências fracas. No espaço de atributos adotado, as árvores impulsionadas constituíram a referência tabular mais forte nos três conjuntos e superaram as arquiteturas tabulares profundas tanto em desempenho preditivo quanto, de maneira acentuada, em custo computacional: em HybridPhisher, o treinamento do modelo final exigiu 13.45 s do aprendiz de *boosting* mais eficiente, contra 818.46 s do modelo tabular profundo mais custoso, sem ganho métrico correspondente. Confirma-se, assim, em escala mais ampla que a do arcabouço proposto, a recomendação metodológica formulada na Seção 5.10.2: sem o confronto prévio com aprendizes de *boosting* devidamente otimizados, a adoção de componentes neurais sobre atributos tabulares projetados carece de justificativa empírica.

Os dois descritores compostos obedeceram ao mesmo mecanismo. Tanto o URLC, proposto nesta pesquisa, quanto o *Domain Similarity*, adaptado do índice de similaridade de Prasad e Chandra (2024), contribuíram de forma consistente e estatisticamente sustentada quando a representação disponível se restringia a uma única fonte de evidência, e pouco ou de forma instável na representação combinada completa. Esse comportamento delimita o contexto em que esses descritores agregam informação adicional e evidencia sua redundância parcial na representação completa. A análise de sensibilidade dos pesos do URLC mostrou ainda que o conceito não depende de uma parametrização frágil e única, pois a AUC univariada oscilou em faixa estreita, entre 88.60% e 90.10%, ao longo de oito esquemas de ponderação distintos. Cabe interpretar a formulação arredondada adotada, portanto, como escolha operacional orientada à interpretabilidade, e não como ótimo matemático.

No teste de estresse considerado, a fusão heterogênea apresentou menor degradação que os ramos mais diretamente afetados pelas perturbações, sem eliminar a sensibilidade do *pipeline*. A degradação concentrou-se de modo previsível no ramo cuja fonte de evidência sofreu a perturbação, enquanto a camada de fusão preservou desempenho apoiando-se nas fontes íntegras: sob perturbação em nível de URL, o ramo lexical em nível de palavra perdeu 11.89 pontos percentuais de Recall, contra variação de apenas -0.11 ponto percentual no meta-modelo baseado em *boosting*; sob perturbação em nível de HTML, os ramos tabulares perderam entre 5.90 e 8.27 pontos percentuais, contra perdas inferiores a 0.60 ponto per-

centual nos meta-modelos. Essa robustez, entretanto, é relativa e se explica pela diversidade das fontes de evidência, não por qualquer mecanismo explícito de defesa, de modo que um adversário capaz de perturbar URL e HTML ao mesmo tempo não encontraria o mesmo efeito compensatório. O perfilamento de eficiência acrescentou que quem dita o custo agregado do detector empilhado são os ramos de base, e não a camada de fusão, cujo tempo de inferência final ficou entre 0.0110 s e 0.3730 s uma vez disponíveis as probabilidades dos modelos de base. Em cenários operacionais, portanto, a redução de custo passa pela seleção e pela execução dos ramos de base, e não pela camada de fusão.

Em síntese, a principal contribuição desta dissertação não está em reportar o maior valor de F1 entre as abordagens comparadas, e sim em deslocar a pergunta de pesquisa. Em vez de indagar qual arquitetura alcança o desempenho mais elevado, o trabalho investigou sob quais condições determinada arquitetura é efetivamente útil e ofereceu um protocolo capaz de responder a essa pergunta de maneira verificável, combinando análise de saturação, decomposição por blocos de atributos, comparação com linhas de base robustas, testes estatísticos pareados sobre execuções repetidas, perfilamento de eficiência e estresse adversarial controlado. A resposta obtida é que, nos experimentos realizados, o empilhamento heterogêneo apresentou maior benefício quando modelos isolados fortes preservaram erros passíveis de correção por evidências complementares, e benefício marginal ou nulo quando os atributos projetados já aproximavam o *benchmark* do teto de desempenho. Como contribuições adicionais, o trabalho apresenta o conjunto HybridPhisher, com ambiguidade residual preservada e, por isso, adequado à comparação da arquitetura e dois descritores interpretáveis, cujo escopo de utilidade foi caracterizado experimentalmente.

6.1 LIMITAÇÕES DO TRABALHO

As conclusões apresentadas comportam limitações que delimitam seu alcance.

A avaliação empregou partições retidas dentro de cada *benchmark*. Execuções repetidas, testes estatísticos pareados e empilhamento fora da dobra reduzem o viés otimista, mas esse desenho não captura deriva temporal nem condições de operação contínua, nas quais as campanhas de *phishing* evoluem e a distribuição dos dados se altera depois que o modelo entra em produção.

PhiUSIIL e StealthPhisher encontram-se fortemente saturados pelos atributos tabulares projetados, o que restringe sua capacidade de demonstrar o valor agregado do empilhamento e concentra em HybridPhisher boa parte da evidência que sustenta as conclusões principais. O próprio HybridPhisher pode carregar vieses da seleção de fontes, dos critérios de filtragem, da validação de páginas ativas, do balanceamento entre classes e da disponibilidade temporal das URLs de *phishing* no momento da coleta, fatores que pesam na generalização dos resultados.

Duas delimitações adicionais decorrem do desenho do arcabouço. A primeira diz respeito à origem dos rótulos: a validação executada sobre o HybridPhisher aferiu acessibilidade e conteúdo, não veracidade da rotulagem, de modo que eventuais falsos positivos presentes nos repositórios comunitários de origem foram transmitidos ao conjunto. A segunda diz respeito ao escopo das modalidades de evidência: o arcabouço opera sobre URL e HTML e deixa deliberadamente de fora tanto os serviços de terceiros em tempo de inferência quanto a análise visual da página renderizada, razão pela qual as conclusões sobre

complementaridade entre modalidades não se estendem às modalidades não avaliadas.

O URLC e o *Domain Similarity* foram avaliados exclusivamente em HybridPhisher e devem ser lidos como descritores determinísticos daquele espaço de atributos, e não como formulações globalmente ótimas ou independentes do conjunto de dados. Acrescente-se que a avaliação do *Domain Similarity* recorreu a três execuções pareadas, e não às dez adotadas nas demais análises, pela razão de orçamento experimental exposta na Seção 5.9. A restrição reduz a precisão das estimativas e recomenda cautela na interpretação e na generalização do efeito observado.

A avaliação adversarial restringiu-se a um teste de estresse controlado em HybridPhisher, com perturbações determinísticas em nível de URL e sobre cópias armazenadas de páginas. Ela não alcança ataques adaptativos de caixa branca, otimização por busca heurística, geração de perturbações baseada em gradiente ou em redes adversárias generativas, cadeias de redirecionamento ativas, *cloaking*, transformações visuais de logotipos nem ofuscação específica de campanha.

Por fim, as medições de tempo vieram de um ambiente de execução único e idêntico para todos os modelos. Essa condição assegura a comparabilidade relativa entre eles, mas restringe a leitura dos valores absolutos, que não se transferem diretamente a outras configurações de *hardware*.

6.2 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

As limitações apontadas delineiam a agenda de continuidade da pesquisa.

Em primeiro lugar, convém avaliar o arcabouço sob protocolos de coleta temporalmente separados e continuamente atualizados. Esse desenho permitiria mensurar a degradação de desempenho diante da evolução das campanhas de *phishing* e estabelecer critérios para o retreinamento periódico dos modelos.

Em segundo lugar, a avaliação adversarial precisa ir além do teste de estresse controlado aqui considerado, incorporando modelos de ameaça adaptativos, cadeias de redirecionamento ativas, *cloaking*, transformações visuais e de logotipos, políticas de perturbação por busca heurística, ofuscação específica de campanha, encurtadores de URL, homóglifos, *typosquatting* e ataques baseados em redirecionamento. Essa extensão permitiria distinguir com maior precisão a estabilidade que decorre da diversidade de evidências da robustez efetiva.

Em terceiro lugar, o URLC e o *Domain Similarity* podem ser refinados com partições de desenvolvimento separadas para a estimação de seus parâmetros e avaliados sobre conjuntos externos e temporalmente distintos, preservada a interpretabilidade que motivou suas formulações.

Por fim, a eficiência operacional do *pipeline* comporta aprimoramento em duas frentes complementares. A primeira é a compressão dos modelos de base, por poda (*pruning*) e por destilação de conhecimento (*knowledge distillation*). A segunda é o roteamento adaptativo da inferência, mediante cascatas leves (*lightweight cascades*) que condicionem a execução seletiva de ramos ao grau de incerteza da predição, associadas ao retreinamento incremental. O ganho é maior em cenários de implantação nos quais os ramos computacionalmente mais custosos devem ser acionados apenas para os casos ambíguos, o que preservaria a capacidade de detecção demonstrada nesta pesquisa a um custo compatível com restrições operacionais

reais.

REFERÊNCIAS BIBLIOGRÁFICAS

- ALJOFEY, A.; BELLO, S. A.; LU, J.; XU, C. BERT-PhishFinder: A robust model for accurate phishing URL detection with optimized DistilBERT. *IEEE Trans. Dependable Secure Comput.*, Institute of Electrical and Electronics Engineers (IEEE), v. 22, n. 4, p. 4315–4329, jul 2025.
- ALJOFEY, A.; JIANG, Q.; RASOOL, A.; CHEN, H.; LIU, W.; QU, Q.; WANG, Y. An effective detection approach for phishing websites using URL and HTML features. *Scientific Reports*, v. 12, n. 1, p. 8842, may 2022.
- ALMOUSA, M.; ANWAR, M. A URL-based social semantic attacks detection with character-aware language model. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, p. 10654–10663, 2023.
- Anti-Phishing Working Group. *Phishing Activity Trends Report, 2nd Quarter 2025*. 2025. <https://docs.apwg.org/reports/apwg_trends_report_q2_2025.pdf>. [Accessed 19-12-2025].
- ARIK, S. Ö.; PFISTER, T. TabNet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 35, n. 8, p. 6679–6687, 2021.
- ASIRI, S.; XIAO, Y.; ALZHRANI, S.; LI, S.; LI, T. A survey of intelligent detection designs of HTML URL phishing attacks. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, p. 6421–6443, 2023.
- ASLAM, S.; ASLAM, H.; MANZOOR, A.; CHEN, H.; RASOOL, A. AntiPhishStack: LSTM-based stacked generalization model for optimized phishing URL detection. *Symmetry (Basel)*, MDPI AG, v. 16, n. 2, p. 248, feb 2024.
- BAKI, S.; VERMA, R. M. Sixteen years of phishing user studies: What have we learned? *IEEE Trans. Dependable Secure Comput.*, Institute of Electrical and Electronics Engineers (IEEE), v. 20, n. 2, p. 1200–1212, mar 2023.
- BARI, N.; SALEEM, T.; SHAH, M.; ALGARNI, A.; PATEL, A.; ULLAH, I. A filter-based feature selection framework to detect phishing URLs using stacking ensemble machine learning. *Comput. Model. Eng. Sci.*, Tech Science Press, v. 145, n. 1, p. 1167–1187, 2025.
- BREIMAN, L. Random forests. *Machine Learning*, Springer Science and Business Media LLC, v. 45, n. 1, p. 5–32, 2001.
- CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.]: Association for Computing Machinery, 2016. p. 785–794.
- Cisco Talos Intelligence Group. *PhishTank: plataforma colaborativa de verificação de phishing*. 2026. <<https://phishtank.org>>. Acesso em: 1 jan. 2026.
- DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. oct 2018.
- DUARTE, J. D.; JUNIOR, P. C.; COSTA, J. Paulo Javidi da; COSTA, E. J. da; MELO, L. Peotta de; NUNES, R. R.; SOARES, C. V. N. G.; SILVA, T. Erivan da C. Machine learning for early detection of phishing URLs in parked domains: An approach applied to a financial institution. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 145736–145753, 2025.

GORISHNIY, Y.; RUBACHEV, I.; KHRULKOV, V.; BABENKO, A. Revisiting deep learning models for tabular data. In: *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. [S.l.]: Curran Associates, 2021. p. 18932–18943.

GUALBERTO, E. S.; SOUSA, R. T. D.; VIEIRA, T. P. D. B.; COSTA, J. P. C. L. D.; DUQUE, C. G. The answer is in the text: Multi-stage methods for phishing detection based on feature engineering. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 8, p. 223529–223547, 2020.

GUALBERTO, E. S.; SOUSA, R. T. D.; VIEIRA, T. P. D. B.; COSTA, J. P. C. L. D.; DUQUE, C. G. From feature engineering and topics models to enhanced prediction rates in phishing detection. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 8, p. 76368–76385, 2020.

HE, D.; LIU, Z.; LV, X.; CHAN, S.; GUIZANI, M. On phishing URL detection using feature extension. *IEEE Internet of Things Journal*, v. 11, n. 24, p. 39527–39536, dec 2024.

HE, D.; LV, X.; ZHU, S.; CHAN, S.; CHOO, K.-K. R. A method for detecting phishing websites based on tiny-bert stacking. *IEEE Internet Things J.*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, n. 2, p. 2236–2243, jan 2024.

HUANG, X.; KHETAN, A.; CVITKOVIC, M.; KARNIN, Z. TabTransformer: Tabular data modeling using contextual embeddings. 2020.

INDRASIRI, P. L.; HALGAMUGE, M. N.; MOHAMMAD, A. Robust ensemble machine learning model for filtering phishing URLs: Expandable random gradient stacked voting classifier (ERG-SVC). *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 9, p. 150142–150161, 2021.

JIANG, H.; CHEN, Y.; ZHU, Y.; XU, X.; SONG, Y.; CHEN, Q. D-PhishNet: A dual-branch network for URL and HTML feature fusion in phishing webpage detection. *Comput. Netw.*, Elsevier BV, v. 271, n. 111648, p. 111648, oct 2025.

KAILAS, S.; ROOPALAKSHMI, R. “think before you click”—malicious URL detection in cybersecurity: A systematic review and research roadmap. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 154305–154325, 2025.

KALABARIGE, L. R.; RAO, R. S.; ABRAHAM, A.; GABRALLA, L. A. Multilayer stacked ensemble learning model to detect phishing websites. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 10, p. 79543–79552, 2022.

KALABARIGE, L. R.; RAO, R. S.; PAIS, A. R.; GABRALLA, L. A. A boosting-based hybrid feature selection and multi-layer stacked ensemble learning model to detect phishing websites. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, p. 71180–71193, 2023.

KARIM, A.; SHAHROZ, M.; MUSTOFA, K.; BELHAOUARI, S. B.; JOGA, S. R. K. Phishing detection system through hybrid machine learning based on URL. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, p. 36805–36822, 2023.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Long Beach, CA, USA: Curran Associates, 2017. p. 3149–3157.

KHALID, A.; HANIF, M.; HAMEED, A.; ASHRAF, Z.; ALNFIAI, M. M.; ALNEFAIE, S. M. M. LogiTriBlend: A novel hybrid stacking approach for enhanced phishing email detection using ML models and vectorization approach. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 12, p. 193807–193821, 2024.

KULKARNI, A.; BALACHANDRAN, V.; DAS, T. Phishing webpage detection: Unveiling the threat landscape and investigating detection techniques. *IEEE Commun. Surv. Tutor.*, Institute of Electrical and Electronics Engineers (IEEE), v. 27, n. 2, p. 974–1007, apr 2025.

KULKARNI, A.; BALACHANDRAN, V.; DIVAKARAN, D. M.; DAS, T. From ML to LLM: Evaluating the robustness of phishing web page detection models against adversarial attacks. *Digit. Threat.*, Association for Computing Machinery (ACM), v. 6, n. 2, p. 1–25, jun 2025.

KUSTIAWAN, Y. A.; GHAUTH, K. I. Evaluating the impact of feature engineering in phishing URL detection: A comparative study of URL, HTML, and derived features. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 126756–126768, 2025.

KUSTIAWAN, Y. A.; GHAUTH, K. I. PhishOFE: A novel machine learning framework for real-time phishing URL detection with optimized feature engineering. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 1–1, 2025.

LE, H.; PHAM, Q.; SAHOO, D.; HOI, S. C. H. URLNet: Learning a URL representation with deep learning for malicious URL detection. feb 2018.

Le Pochat, V.; GOETHEM, T. V.; TAJALIZADEHKHOOB, S.; KORCZYŃSKI, M.; JOOSEN, W. Tranco: A research-oriented top sites ranking hardened against manipulation. In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*. San Diego, CA, USA: [s.n.], 2019.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, may 2015.

Lo Re, G.; MORANA, M.; RIZZO, G. Adversarial attacks on phishing webpage detectors via heuristic search techniques. *J. Inf. Secur. Appl.*, Elsevier BV, v. 100, n. 104486, p. 104486, jul 2026.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. 1–482 p. ISBN 978-0-521-86571-5.

MOHANTY, S.; ACHARYA, A. A.; GABER, T.; PANDA, N.; ELDESOUKY, E.; HAMEED, I. A. An efficient hybrid feature selection technique toward prediction of suspicious URLs in IoT environment. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 12, p. 50578–50594, 2024.

Motiur Rahman, S. S. M.; ISLAM, T.; JABIULLAH, M. I. PhishStack: Evaluation of stacked generalization in phishing URLs detection. *Procedia Comput. Sci.*, Elsevier BV, v. 167, p. 2410–2418, 2020.

NADEAU, C.; BENGIO, Y. Inference for the generalization error. *Machine Learning*, Kluwer Academic Publishers, v. 52, p. 239–281, 2003.

NASEEB, S.; RAMZAN, S.; RAZA, A.; HASHMI, M. S. A.; GU, Y.; SYAFRUDIN, M.; FITRIYANI, N. L. Website phishing attack detection using innovative meta learning-based ensemble approach. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 164249–164264, 2025.

NAYAK, G. S.; MUNIYAL, B.; BELAVAGI, M. C. Enhancing phishing detection: A machine learning approach with feature selection and deep learning models. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 33308–33320, 2025.

NICULESCU-MIZIL, A.; CARUANA, R. Predicting good probabilities with supervised learning. In: *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*. [S.l.]: Association for Computing Machinery, 2005. p. 625–632.

NOWROOZI, E.; Abhishek; MOHAMMADI, M.; CONTI, M. An adversarial attack analysis on malicious advertisement URL detection framework. *IEEE Trans. Netw. Serv. Manag.*, Institute of Electrical and Electronics Engineers (IEEE), v. 20, n. 2, p. 1332–1344, jun 2023.

Open PageRank. *Top 10 million domains*. 2026. <<https://openpagerank.keywordseverywhere.com/top-10-million-domains>>. Acesso em: 1 jan. 2026.

Phishing Database Project. *Phishing.Database: base comunitária de URLs de phishing*. 2026. <<https://github.com/{P}hishing-{D}atabase>>. Acesso em: 1 jan. 2026.

PILLAI, M. J.; REMYA, S.; DEVIKA, V.; RAMASUBBAREDDY, S.; CHO, Y. Evasion attacks and defense mechanisms for machine learning-based web phishing classifiers. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 12, p. 19375–19387, 2024.

PRASAD, A.; CHANDRA, S. PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Comput. Secur.*, Elsevier BV, v. 136, n. 103545, p. 103545, jan 2024.

PRASAD, A.; YADAV, V.; SOLANKI, C.; GOSWAMI, H.; JHA, T.; NAGAL, D. StealthPhisher: A defensive framework against phishing attack using hybrid deep learning and GenAI. *Expert Syst. Appl.*, Elsevier BV, v. 299, n. 130205, p. 130205, mar 2026.

RAFSANJANI, A. S.; KAMARUDDIN, N. B.; BEHJATI, M.; ASLAM, S.; SARFARAZ, A.; AMPHAWAN, A. Enhancing malicious URL detection: A novel framework leveraging priority coefficient and feature evaluation. *IEEE Access*, v. 12, p. 85001–85026, 2024.

RASTAKHIZ, F.; EFTEKHARI, M.; VAHDATI, S. QuickCharNet: An efficient URL classification framework for enhanced search engine optimization. *IEEE Access*, v. 12, p. 156965–156979, 2024.

REMYA, S.; PILLAI, M. J.; APARNA, B. S.; SUBBAREDDY, S. R.; CHO, Y. Y. BGL-PhishNet: Phishing website detection using hybrid model-BERT, GNN, and LightGBM. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 47552–47569, 2025.

ROY, P. K.; KUMAR, A.; SINGH, A. Advanced learning for phishing URLs detection to secure consumer-centric applications. *IEEE Trans. Consum. Electron.*, Institute of Electrical and Electronics Engineers (IEEE), v. 70, n. 3, p. 5756–5763, aug 2024.

SABIR, B.; BABAR, M. A.; GAIRE, R.; ABUADBBA, A. Reliability and robustness analysis of machine learning based phishing URL detectors. *IEEE Trans. Dependable Secure Comput.*, Institute of Electrical and Electronics Engineers (IEEE), p. 1–18, 2024.

SAHINGOZ, O. K.; BUBER, E.; KUGU, E. DEPHIDES: Deep learning based phishing detection system. *IEEE Access*, v. 12, p. 8052–8070, 2024.

SKULA, I.; KVET, M. A framework for preparing a balanced and comprehensive phishing dataset. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 12, p. 53610–53622, 2024.

Sruthi; Naik S, M. A novel framework for effective phishing URL detection using an LSTM-based siamese network. *Knowl. Based Syst.*, Elsevier BV, v. 329, n. 114271, p. 114271, nov 2025.

TIAN, Y.; YU, Y.; SUN, J.; WANG, Y. From past to present: A survey of malicious URL detection techniques, datasets and code repositories. *Comput. Sci. Rev.*, Elsevier BV, v. 58, n. 100810, p. 100810, nov 2025.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. jun 2017.

WEI, Y.; NAKAYAMA, M.; SEKIYA, Y. Enhancing generalization in phishing URL detection via a fine-tuned BERT-based multimodal approach. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, p. 131197–131216, 2025.

WOLPERT, D. H. Stacked generalization. *Neural Networks*, Elsevier BV, v. 5, n. 2, p. 241–259, 1992.

YUAN, Y.; HAO, Q.; APRUZZESE, G.; CONTI, M.; WANG, G. “are adversarial phishing webpages a threat in reality?” understanding the users’ perception of adversarial webpages. In: *Proceedings of the ACM Web Conference 2024*. New York, NY, USA: ACM, 2024. v. 20, p. 1712–1723.

ZIENI, R.; MASSARI, L.; CALZAROSSA, M. C. Phishing or not phishing? a survey on the detection of phishing websites. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, p. 18499–18519, 2023.

ZONYFAR, C.; LEE, J.-B.; KIM, J.-D. HCNN-LSTM: Hybrid convolutional neural network with long short-term memory integrated for legitimate web prediction. *J. Web Eng.*, River Publishers, dec 2023.

APÊNDICES

A. CONFIGURAÇÕES DE HIPERPARÂMETROS DOS MODELOS

Este apêndice reproduz as configurações de hiperparâmetros selecionadas antes das avaliações finais e mantidas fixas durante todas as execuções repetidas, conforme o protocolo descrito na Seção 4.6.3. Em todos os casos, a seleção maximizou a PR-AUC média de validação em dobras estratificadas, e os rótulos da partição de teste não foram utilizados em nenhuma etapa. Os meta-aprendizes de Nível 1 não passaram por essa otimização e foram avaliados com parâmetros fixos, apresentados na Seção A.4.

A coluna *Tentativas* indica o número de tentativas concluídas antes do atendimento ao critério de parada da otimização, e a coluna *Melhor* identifica a tentativa cuja configuração foi travada. Os modelos tabulares e tabulares profundos utilizaram o *pipeline* de pré-processamento `MinMaxScaler` $\rightarrow$ `VarianceThreshold(10-5)` $\rightarrow$ `SelectKBest( $\chi^2$ )`, ajustado exclusivamente sobre a partição de treinamento.

Os nomes dos modelos seguem a notação adotada no Capítulo 4: LR_c para a regressão logística sobre n-gramas de caractere, RF_t para o *Random Forest*, XGB_t para o XGBoost, $LGBM_t$ para o LightGBM, CNN_c e CNN_w para as redes convolucionais em nível de caractere e de *token*, e $TabNet_t$, $TabTr_t$ e $FTTr_t$ para os modelos tabulares profundos, além de $LGBM_m$ e LR_m para os meta-aprendizes de Nível 1.

A.1 HYBRIDPHISHER

Tabela A.1 – Hiperparâmetros finais travados no HybridPhisher.

Modelo	Tentativas	Melhor	Hiperparâmetros travados
LR_c	14	6	Vetorizador TF-IDF de caractere com intervalo de n-gramas (3,6), máximo de características = 300000, conversão para minúsculas habilitada, remoção de acentos Unicode, min-df = 1; regressão logística com penalidade L2, $C = 30,0$, <i>solver</i> = liblinear, max-iter = 2000.
RF_t	15	5	<i>Random Forest</i> com $n_estimators = 1200$, max-depth = 30, min-samples-leaf = 1, min-samples-split = 2, max-features = log2, bootstrap = true, max-samples = 0,7, $n_jobs = -1$.
XGB_t	14	4	XGBoost com <i>objective</i> = binary:logistic, <i>eval-metric</i> = logloss, $n_estimators = 4000$, learning-rate = 0,1, max-depth = 7, min-child-weight = 1, subsample = 0,85, colsample-bytree = 0,85, gamma = 0,0, reg-alpha = 0,01, reg-lambda = 1,0, $n_jobs = -1$.

Continua na próxima página

Tabela A.1 – continuação da página anterior

Modelo	Tentativas	Melhor	Hiperparâmetros travados
LGBM _t	16	6	LightGBM com objective = binary, $n_estimators = 3000$, learning-rate = 0,02, num-leaves = 127, max-depth = 9, min-child-samples = 120, subsample = 1,0, colsample-bytree = 0,7, reg-alpha = 0,5, reg-lambda = 1,0, random-state = 42, $n_jobs = -1$.
CNN _c	14	10	CNN em nível de caractere com vocabulário ASCII imprimível, comprimento de caractere = 800, dimensão de <i>embedding</i> = 64, blocos convolucionais (256, 3, 2), (256, 5, 2) e (256, 7, 2), unidades densas = 256, <i>dropout</i> do modelo = 0,3; otimizador Adam, learning-rate = 0,001, épocas = 8, batch-size = 128, weight-decay = 10^{-5} , <i>dropout</i> de treinamento = 0,4.
CNN _w	20	10	CNN em nível de <i>token</i> com tokenização por segmento, comprimento de <i>token</i> = 320, dimensão de <i>embedding</i> = 128, blocos convolucionais (256, 3, 2) e (256, 5, 2), unidades densas = 256, <i>dropout</i> do modelo = 0,3; otimizador Adam, learning-rate = 0,001, épocas = 12, batch-size = 128, weight-decay = 10^{-5} , <i>dropout</i> de treinamento = 0,1.
TabNet _t	15	8	TabNet com $n_d = 16$, $n_a = 64$, $n_steps = 4$, $\gamma = 1,2$, $n_independent = 2$, $n_shared = 2$, $\lambda_sparse = 10^{-4}$, tipo de máscara = entmax; otimizador Adam, learning-rate = 0,02, weight-decay = 0,0, batch-size = 1024, virtual-batch-size = 128, max-epochs = 100, paciência = 20.
TabTr _t	12	2	TabTransformer com $d_token = 64$, $n_blocks = 2$, cabeças de atenção = 8, <i>dropout</i> de atenção = 0,1, <i>dropout</i> FFN = 0,1, <i>dropout</i> residual = 0,1, multiplicador oculto do MLP = 4,0, fator oculto FFN = 2,0; otimizador AdamW, learning-rate = 0,0003, weight-decay = 10^{-5} , batch-size = 512, épocas = 100, paciência = 20.
FTTr _t	11	1	FT-Transformer com $d_token = 64$, $n_blocks = 2$, cabeças de atenção = 8, <i>dropout</i> de atenção = 0,2, fator oculto FFN = 1,0, <i>dropout</i> FFN = 0,0, <i>dropout</i> residual = 0,0; otimizador AdamW, learning-rate = 0,0003, weight-decay = 10^{-4} , batch-size = 1024, épocas = 60, paciência = 20.

A.2 PHIUSIIL

Tabela A.2 – Hiperparâmetros finais travados no PhiUSIIL.

Modelo	Tentativas	Melhor	Hiperparâmetros travados
LR _c	11	1	Vetorizador TF-IDF de caractere com intervalo de n-gramas (3, 5), máximo de características = 200000, conversão para minúsculas habilitada, remoção de acentos Unicode, min-df = 1; regressão logística com penalidade L2, $C = 30,0$, <i>solver</i> = liblinear, max-iter = 500.
RF _t	11	1	<i>Random Forest</i> com $n_estimators = 2000$, max-depth = 30, min-samples-leaf = 1, min-samples-split = 10, max-features = none, bootstrap = true, max-samples = 0,95, $n_jobs = -1$.
XGB _t	11	4	XGBoost com objective = binary:logistic, eval-metric = logloss, $n_estimators = 6000$, learning-rate = 0,05, max-depth = 9, min-child-weight = 5, subsample = 0,6, colsample-bytree = 0,85, gamma = 2,0, reg-alpha = 0,001, reg-lambda = 1,0, $n_jobs = -1$.
LGBM _t	11	1	LightGBM com objective = binary, $n_estimators = 3000$, learning-rate = 0,03, num-leaves = 15, max-depth = 9, min-child-samples = 80, subsample = 1,0, colsample-bytree = 0,7, reg-alpha = 0,001, reg-lambda = 5,0, random-state = 42, $n_jobs = -1$.
CNN _c	20	10	CNN em nível de caractere com vocabulário ASCII imprimível, comprimento de caractere = 800, dimensão de <i>embedding</i> = 64, blocos convolucionais (256, 3, 2), (256, 5, 2) e (256, 7, 2), unidades densas = 256; otimizador Adam, learning-rate = 0,001, épocas = 12, batch-size = 128, weight-decay = 10^{-5} , <i>dropout</i> de treinamento = 0,1.
CNN _w	15	5	CNN em nível de <i>token</i> com tokenização por segmento, comprimento de <i>token</i> = 320, dimensão de <i>embedding</i> = 128, blocos convolucionais (256, 3, 2) e (256, 5, 2), unidades densas = 256; otimizador Adam, learning-rate = 0,001, épocas = 5, batch-size = 128, weight-decay = 0,0, <i>dropout</i> de treinamento = 0,4.

Continua na próxima página

Tabela A.2 – continuação da página anterior

Modelo	Tentativas	Melhor	Hiperparâmetros travados
TabNet _t	11	3	TabNet com $n_d = 64$, $n_a = 16$, $n_steps = 5$, $\gamma = 2,0$, $n_independent = 2$, $n_shared = 1$, $\lambda_sparse = 10^{-5}$, tipo de máscara = entmax; otimizador Adam, learning-rate = 0,01, weight-decay = 10^{-5} , batch-size = 2048, virtual-batch-size = 128, max-epochs = 80, paciência = 20.
TabTr _t	11	11	TabTransformer com $d_token = 64$, $n_blocks = 2$, cabeças de atenção = 8, $dropout$ de atenção = 0,2, $dropout$ FFN = 0,0, $dropout$ residual = 0,0, multiplicador oculto do MLP = 2,0, fator oculto FFN = 4,0; otimizador AdamW, learning-rate = 0,0003, weight-decay = 10^{-4} , batch-size = 1024, épocas = 60, paciência = 20.
FTTr _t	11	2	FT-Transformer com $d_token = 64$, $n_blocks = 2$, cabeças de atenção = 8, $dropout$ de atenção = 0,2, fator oculto FFN = 1,0, $dropout$ FFN = 0,0, $dropout$ residual = 0,0; otimizador AdamW, learning-rate = 0,0003, weight-decay = 10^{-4} , batch-size = 1024, épocas = 60, paciência = 20.

A.3 STEALTHPHISHER

Tabela A.3 – Hiperparâmetros finais travados no StealthPhisher.

Modelo	Tentativas	Melhor	Hiperparâmetros travados
LR _c	11	1	Vetorizador TF-IDF de caractere com intervalo de n-gramas (3,5), máximo de características = 200000, conversão para minúsculas habilitada, remoção de acentos Unicode, min-df = 1; regressão logística com penalidade L2, $C = 30,0$, $solver = liblinear$, max-iter = 500.
RF _t	15	11	<i>Random Forest</i> com $n_estimators = 1200$, max-depth = 30, min-samples-leaf = 1, min-samples-split = 2, max-features = log2, bootstrap = true, max-samples = 0,7, $n_jobs = -1$.
XGB _t	11	6	XGBoost com objective = binary:logistic, eval-metric = logloss, $n_estimators = 6000$, learning-rate = 0,05, max-depth = 9, min-child-weight = 5, subsample = 0,6, colsample-bytree = 0,85, gamma = 2,0, reg-alpha = 0,001, reg-lambda = 1,0, $n_jobs = -1$.

Continua na próxima página

Tabela A.3 – continuação da página anterior

Modelo	Tentativas	Melhor	Hiperparâmetros travados
LGBM _t	11	7	LightGBM com objective = binary, $n_estimators = 3000$, learning-rate = 0,03, num-leaves = 15, max-depth = 9, min-child-samples = 80, subsample = 1,0, colsample-bytree = 0,7, reg-alpha = 0,001, reg-lambda = 5,0, random-state = 42, $n_jobs = -1$.
CNN _c	14	10	CNN em nível de caractere com vocabulário ASCII imprimível, comprimento de caractere = 800, dimensão de <i>embedding</i> = 64, blocos convolucionais (256, 3, 2), (256, 5, 2) e (256, 7, 2), unidades densas = 256; otimizador Adam, learning-rate = 0,001, épocas = 8, batch-size = 128, weight-decay = 10^{-5} , <i>dropout</i> de treinamento = 0,4.
CNN _w	15	5	CNN em nível de <i>token</i> com tokenização por segmento, comprimento de <i>token</i> = 320, dimensão de <i>embedding</i> = 128, blocos convolucionais (256, 3, 2) e (256, 5, 2), unidades densas = 256; otimizador Adam, learning-rate = 0,001, épocas = 5, batch-size = 128, weight-decay = 0,0, <i>dropout</i> de treinamento = 0,4.
TabNet _t	11	10	TabNet com $n_d = 64$, $n_a = 16$, $n_steps = 5$, $\gamma = 2,0$, $n_independent = 2$, $n_shared = 1$, $\lambda_sparse = 10^{-5}$, tipo de máscara = entmax; otimizador Adam, learning-rate = 0,01, weight-decay = 10^{-5} , batch-size = 2048, virtual-batch-size = 128, max-epochs = 80, paciência = 20.
TabTr _t	11	4	TabTransformer com $d_token = 64$, $n_blocks = 2$, cabeças de atenção = 8, <i>dropout</i> de atenção = 0,2, <i>dropout</i> FFN = 0,0, <i>dropout</i> residual = 0,0, multiplicador oculto do MLP = 2,0, fator oculto FFN = 4,0; otimizador AdamW, learning-rate = 0,0003, weight-decay = 10^{-4} , batch-size = 1024, épocas = 60, paciência = 20.
FTTr _t	11	5	FT-Transformer com $d_token = 64$, $n_blocks = 2$, cabeças de atenção = 8, <i>dropout</i> de atenção = 0,2, fator oculto FFN = 1,0, <i>dropout</i> FFN = 0,0, <i>dropout</i> residual = 0,0; otimizador AdamW, learning-rate = 0,0003, weight-decay = 10^{-4} , batch-size = 1024, épocas = 60, paciência = 20.

A.4 META-APRENDIZES DE NÍVEL 1

Os meta-aprendizes não passaram pela otimização descrita nas seções anteriores. Ambos foram avaliados com os parâmetros fixos da Tabela A.4, idênticos nos três conjuntos de dados, e suas probabilidades foram calibradas por regressão isotônica com validação cruzada interna de três dobras, conforme a Seção 4.5.4.

Tabela A.4 – Parâmetros fixos dos meta-aprendizes de Nível 1.

Modelo	Parâmetros fixos
$LGBM_m$	LightGBM com <code>objective = binary</code> , $n_estimators = 800$, <code>learning-rate = 0,05</code> , <code>num-leaves = 31</code> , <code>min-child-samples = 100</code> , <code>subsample = 0,8</code> , <code>colsample-bytree = 0,8</code> , <code>reg-alpha = 0,5</code> , <code>reg-lambda = 5,0</code> , <code>random-state = 42</code> , $n_jobs = -1$.
LR_m	Regressão logística com penalidade <i>elastic net</i> , <code>l1-ratio = 0,3</code> , $C = 0,5$, <code>solver = saga</code> , <code>max-iter = 5000</code> , <code>random-state = 42</code> , $n_jobs = -1$.

B. IMPLEMENTAÇÃO DOS DESCRITORES COMPOSTOS

Este apêndice reproduz a implementação dos dois descritores compostos incorporados ao esquema de características: o *URL Complexity Score* da Seção 4.4.5, proposto nesta pesquisa, e o índice de similaridade que sustenta o *Domain_Similarity* da Seção 4.4.4, adaptado do *URL Similarity Index* de Prasad e Chandra (2024). A opção por reproduzir estes dois trechos em código, e não em pseudocódigo, decorre de sua natureza: o primeiro é contribuição original do trabalho e o segundo é reformulação de um índice cuja especificação original não é detalhada em grau suficiente para replicação direta, de modo que, em ambos os casos, a correspondência entre as equações do Capítulo 4 e a forma como foram efetivamente calculados é parte do que se submete à verificação.

As listagens omitem tratamento de exceção, registro de execução, leitura e escrita de arquivos e as camadas de importação do projeto, preservando integralmente a lógica de cálculo. Os comentários originais foram mantidos, com acentuação removida por exigência do ambiente de listagem.

B.1 URL COMPLEXITY SCORE

A função `_components` calcula os quatro componentes normalizados definidos nas Equações (4.6) a (4.10), e a função `urlc_v3` aplica a combinação ponderada da Equação (4.11). Os pesos e o parâmetro $P_{\max}$ são constantes do módulo, o que garante que o descritor seja determinístico e independente da partição.

Listing B.1 – Cálculo do URL Complexity Score

```
1 W_MIX, W_MAXTOK, W_SUBDOM, W_QUERY = 0.40, 0.25, 0.25, 0.10
2 P_MAX = 20 # normalizacao suave da contagem de parametros
3
4 def _tokenize_path(path):
5     """Quebra o caminho por '/' e depois por '-', '_' e '.'"
6     parts = [p for p in path.split("/") if p]
7     tokens = []
8     for seg in parts:
9         buf = ""
10        for ch in seg:
11            if ch in "-_." :
12                if buf:
13                    tokens.append(buf); buf = ""
14            else:
15                buf += ch
16        if buf:
17            tokens.append(buf)
18    return tokens
19
20 def _components(url):
21     """Retorna (S_mix, S_maxtok, S_subdom, S_query), cada um em [0,1]."""
```

```

22     s = urlsplit(url)
23     host  = (s.netloc or "").lower()
24     path  = s.path or ""
25     query = s.query or ""
26
27     host_labels = [t for t in host.split(".") if t]
28     all_tokens  = host_labels + _tokenize_path(path)
29
30     # S_mix: fracao de tokens que misturam letras e digitos
31     if all_tokens:
32         mix = sum(1 for t in all_tokens
33                   if any(c.isalpha() for c in t) and any(c.isdigit() for c in t))
34         S_mix = mix / len(all_tokens)
35     else:
36         S_mix = 0.0
37
38     # S_maxtok: dominancia do maior token
39     if all_tokens:
40         lengths = [len(t) for t in all_tokens]
41         S_maxtok = max(lengths) / max(sum(lengths), 1)
42     else:
43         S_maxtok = 0.0
44
45     # S_subdom: concentracao entre os rotulos que precedem os dois ultimos
46     subdoms = host_labels[:-2] if len(host_labels) > 2 else []
47     if subdoms:
48         lens = [len(x) for x in subdoms]
49         S_subdom = max(lens) / max(sum(lens), 1)
50     else:
51         S_subdom = 0.0
52
53     # S_query: volume de parametros com diversidade de chaves
54     params = parse_qsl(query, keep_blank_values=True)
55     p = len(params)
56     if p > 0:
57         r_p = len(set(k for k, _ in params)) / p
58         S_query = (math.log1p(p) / math.log1p(P_MAX)) * (0.5 + 0.5 * r_p)
59         S_query = min(max(S_query, 0.0), 1.0)
60     else:
61         S_query = 0.0
62
63     return S_mix, S_maxtok, S_subdom, S_query
64
65 def urlc_v3(url):
66     """URLC em [0,100]."""
67     S_mix, S_maxtok, S_subdom, S_query = _components(url)
68     score = 100.0 * (W_MIX * S_mix + W_MAXTOK * S_maxtok
69                     + W_SUBDOM * S_subdom + W_QUERY * S_query)
70     return min(max(score, 0.0), 100.0)

```

B.2 ÍNDICE DE SIMILARIDADE ENTRE DOMÍNIOS ADAPTADO DO USI DE PRASAD E CHANDRA (2024)

A função `normalize_host_for_usi_full` aplica a normalização descrita na Seção 4.4.4, e `usi_score` implementa a ponderação posicional da Equação (4.3) e o alinhamento sequencial da Equação (4.4). Observe-se, na linha em que o escore é acumulado, a correspondência direta com a expressão $50/N + 100(N - i) / (N(N + 1))$, uma vez que a variável `nsum` armazena $N(N + 1)/2$.

Listing B.2 – Normalização do *host* e índice de similaridade entre domínios

```
1 def normalize_host_for_usi_full(host):
2     """Percent-decode, minusculas, remocao de 'www.', IDNA e NFKC.
3     Nao reduz ao dominio registravel, para evitar similaridade maxima
4     espuria em plataformas de hospedagem compartilhada."""
5     h = (host or "").strip().strip(".")
6     if not h:
7         return ""
8     h = unquote(h).strip().strip(".").lower()
9     if h.startswith("www."):
10         h = h[4:]
11     h = _decode_idna(h)
12     h = unicodedata.normalize("NFKC", h).casefold()
13     return h.strip().strip(".")
14
15 def usi_score(src, tar):
16     """Similaridade entre dois dominios, em [0,100].
17
18     N         = max(len(src), len(tar)), constante durante o alinhamento
19     w(i)      = 50/N + 50*(N-i)/nsum, com nsum = N(N+1)/2
20     match     : acumula w(i) e avanca
21     mismatch  : comprimentos iguais -> penaliza sem remover
22                 comprimentos difer. -> remove do mais longo e avanca
23
24     if not src or not tar:
25         return 0.0
26
27     N = max(len(src), len(tar))
28     a = 50.0 / float(N)
29     nsum = (float(N) * float(N + 1)) / 2.0
30
31     src_cur, tar_cur = src, tar
32     n0 = min(len(src_cur), len(tar_cur))
33     SI, i = 0.0, 0
34
35     while i < n0 and i < len(src_cur) and i < len(tar_cur):
36         if _char_match(src_cur[i], tar_cur[i]):
37             SI += a + (50.0 * float(N - i)) / nsum
38             i += 1
39             continue
40
41         if len(src_cur) == len(tar_cur):
42             i += 1                                # substituicao penalizada
43             continue
44
45         if len(src_cur) > len(tar_cur): # remove do mais longo
46             src_cur = src_cur[:i] + src_cur[i + 1:]
47     else:
```

```
48         tar_cur = tar_cur[:i] + tar_cur[i + 1:]
49         i += 1
50
51         if not src_cur or not tar_cur:
52             break
53
54     return min(max(SI, 0.0), 100.0)
```

C. PROTOCOLO DE EMPILHAMENTO EM PSEUDOCÓDIGO

Este apêndice descreve, em pseudocódigo, dois procedimentos cuja implementação é extensa mas cuja lógica é compacta: a busca do domínio mais semelhante em um conjunto de referência de grande porte e o protocolo de aprendizado empilhado com previsões fora da dobra. Diferentemente do Apêndice B, aqui a opção pelo pseudocódigo é deliberada. Ambos são procedimentos de orquestração, e não formulações cujo detalhe aritmético precise ser conferido contra as equações do Capítulo 4; sua implementação dedica a maior parte das linhas a indexação, nomeação de colunas, conversão de estruturas e compatibilidade entre versões de biblioteca, elementos que obscureceriam a lógica sem acrescentar informação verificável.

C.1 SIMILARIDADE MÁXIMA CONTRA O CONJUNTO DE REFERÊNCIA

O procedimento a seguir implementa a Equação (4.5). Comparar cada *host* observado contra a totalidade do conjunto de referência seria proibitivo, e por isso a busca percorre apenas candidatos plausíveis, recuperados de um índice construído uma única vez. O score, contudo, é sempre calculado sobre o *host* original, e as variantes servem apenas para localizar candidatos.

Listing C.1 – Similaridade máxima contra o conjunto de referência

```
1 Entrada : host observado h, conjunto de referencia R, indice I
2 Saida   : Domain_Similarity(h) em [0,100]
3
4 h <- normalizar(h)                # percent-decode, minusculas,
5                                     # sem 'www.', IDNA, NFKC
6 se h esta em R entao
7     devolver 100                  # correspondencia exata
8
9 melhor <- 0
10 para cada variante v de h obtida removendo ate 2 rotulos a esquerda faca
11     B <- grupos de I indexados por (dominio de topo de v,
12                                     prefixo de 5 caracteres de v,
13                                     faixa de comprimento de v)
14     B <- B uniao grupos de faixas de comprimento vizinhas
15     se B esta vazio entao
16         B <- grupos indexados apenas por (prefixo, faixa), qualquer topo
17     fim se
18
19     para cada candidato d em B, respeitando o teto de candidatos, faca
20         se |comprimento(d) - comprimento(h)| excede a janela entao
21             seguir para o proximo candidato
22         fim se
23         s <- SI(d, h)                # Equacao do alinhamento ponderado
24         se s > melhor entao
25             melhor <- s
26         se melhor >= 100 - tolerancia entao
27             devolver 100            # encerramento antecipado
```

```

28         fim se
29     fim se
30 fim para
31 fim para
32
33 devolver melhor

```

C.2 APRENDIZADO EMPILHADO COM PREDIÇÕES FORA DA DOBRA

O procedimento a seguir corresponde às Seções 4.5.3 e 4.5.4, e observa os controles contra vazamento da Seção 4.5.5. Duas propriedades merecem atenção na leitura. A primeira é que cada amostra de treinamento recebe uma probabilidade produzida por um modelo que não a utilizou no ajuste, condição de validade do meta-aprendizado. A segunda é que o reajuste de cada preditor de Nível 0 sobre a partição de treinamento completa ocorre em passo separado do laço de dobras, e é esse modelo reajustado que produz as probabilidades da partição de teste.

Listing C.2 – Protocolo de aprendizado empilhado com previsões fora da dobra

```

1 Entrada : conjunto D, semente s, numero de dobras k = 5
2 Saida   : predicoes binarias sobre a particao de teste retida
3
4 (Tr, Te) <- divisao estratificada 80/20 de D, com semente s
5 ajustar pre-processamento e selecao de caracteristicas apenas em Tr
6 aplicar as transformacoes ajustadas a Te, sem reajuste
7
8 # ----- Nivel 0 -----
9 para cada preditor m dos ramos tabular e lexico faca
10     P_oof[m] <- vetor vazio de tamanho |Tr|
11
12     para cada dobra (tr, va) de uma particao estratificada de Tr em k partes faca
13         ajustar m sobre tr
14         P_oof[m][va] <- probabilidade da classe positiva predita para va
15     fim para
16
17     reajustar m sobre Tr completo # passo separado do laço
18     P_test[m] <- probabilidade predita para Te
19 fim para
20
21 P_OOF <- matriz cujas colunas sao P_oof[m], ordenacao fixa dos modelos
22 P_TEST <- matriz cujas colunas sao P_test[m], mesma ordenacao
23
24 # ----- Nivel 1 -----
25 para cada meta-aprendiz g em {LightGBM, regressao logistica elastic net} faca
26     ajustar g sobre (P_OOF, rotulos de Tr)
27     g <- calibrar g por regressao isotonica,
28         com validacao cruzada interna de 3 dobras sobre Tr
29
30     p_tr <- probabilidade calibrada de g sobre P_OOF
31     tau_e <- argmax sobre uma grade de limiares da medida F1 em (rotulos de Tr, p_tr)
32     # limiares adicionais para acuracia e para recall sob FPR <= 0,01 e <= 0,05
33
34     p_te <- probabilidade calibrada de g sobre P_TEST
35     y_te <- indicador de p_te >= tau, para tau em {0,5 ; tau_e}

```

```
36 fim para
37
38 devolver y_te e as metricas calculadas sobre Te
```

D. CONSTRUÇÃO DO CONJUNTO HYBRIDPHISHER

Este apêndice documenta a construção do HybridPhisher, conjunto de dados proposto nesta pesquisa. A construção foi feita em seis etapas executadas em sequência. Cada etapa tem um *script* próprio e gera um arquivo CSV que alimenta a etapa seguinte, conforme a Tabela D.1. Pelo mesmo motivo exposto no Apêndice C, as etapas são descritas em pseudocódigo: o código dedica a maior parte das linhas a paralelismo, retomada por *checkpoint*, registro de execução e tratamento de exceções. Os valores numéricos citados nas listagens são os parâmetros configurados nos *scripts*. O conjunto final reúne 221.360 URLs, das quais 112.315 legítimas e 109.045 de phishing.

Tabela D.1 – Etapas de construção do HybridPhisher.

Etapa	Finalidade	Script
1	Coleta e curadoria das URLs legítimas	d1_download_legit_full_urls, d2_clean_and_verify_benign_urls, d3_dataset_tools_remove_outros_idiomas_url
2	Seleção das URLs de phishing ativas	filter_active_phish_urls
3	Filtro de escrita não latina nas URLs de phishing	d3_dataset_tools_remove_outros_idiomas_url
4	Composição do conjunto rotulado	merge_legit_and_phish_balanced
5	Cálculo dos 25 atributos de URL	url_features
6	Captura do HTML e cálculo dos atributos HTML	dataset_builder_HTML_only

D.1 URLS LEGÍTIMAS

As sementes da classe legítima vieram da lista Tranco *top-1M* (Le Pochat et al., 2019). De cada domínio ativo foram extraídas URLs internas reais, e não apenas a página inicial, com limite de 15 URLs por domínio para que poucos sítios não dominassem a classe. Domínios organizados por prefixos de idioma, como /en ou /pt-BR, foram descartados por inteiro, pois gerariam várias versões da mesma página (Listagem D.1). As URLs restantes passaram por normalização, deduplicação, remoção de recursos estáticos e verificação leve de disponibilidade, sem download da página completa (Listagem D.2).

Listing D.1 – Coleta das URLs legítimas a partir da lista Tranco

```
1 Entrada : lista Tranco T (900.000 dominios), k = 15 URLs por dominio
2 Saída   : conjunto L0 de pares (u, 0)
3
```

```

4 L0 <- vazio
5 para cada dominio d em T, em paralelo (128 threads, timeout de 8 s), faca
6   r <- HEAD(https://d/)
7   se r falhou ou status(r) >= 400 entao
8     descartar d # dominio inativo
9   fim se
10  se o robots.txt de d referencia caminhos de idioma entao
11    descartar d
12  fim se
13  U <- URLs de ate 6 sitemaps (indicados no robots.txt ou em
14    caminhos usuais), com ate 5.000 URLs por sitemap
15  se U esta vazio entao
16    U <- ate 80 links internos da pagina inicial
17  fim se
18  se alguma u em U tem primeiro segmento de caminho de idioma entao
19    descartar d
20  fim se
21  L0 <- L0 uniao {(u, 0)} para as k primeiras u ineditas de U,
22    com esquema https e sem fragmento
23 fim para

```

Listing D.2 – Curadoria e verificação das URLs legítimas

```

1 Entrada : conjunto L0
2 Saída   : conjunto L de URLs legitimas verificadas
3
4 rho <- [a-z]{2,3}([_][a-zA-Z]{2,4})? # segmento inteiro de idioma
5 D_idioma <- dominios de L0 com algum primeiro segmento de caminho em rho
6
7 para cada u em L0 faca
8   u <- normalizar(u) # https, host minusculo, sem 'www.',
9   # barras repetidas colapsadas
10  se u invalida ou dominio(u) em D_idioma
11    ou u aponta para recurso estatico # css, js, imagem, fonte, pdf,
12    # compactado, midia, executavel
13    ou u repetida ou dominio(u) ja tem 15 URLs entao
14    descartar u
15  fim se
16 fim para
17
18 para cada u restante, em paralelo (120 threads; conexao 1,5 s; leitura 3,5 s), faca
19   r <- HEAD(u), sem seguir redirecionamentos
20   se r falhou ou status(r) >= 400 entao
21     r <- GET(u) com cabecalho Range: bytes=0-2047
22   fim se
23   se status(r) em [200, 399] e Content-Type em {text/html,
24     application/xhtml+xml, text/plain} entao
25     marcar u como ok
26   senao
27     marcar u como suspeita e registrar o motivo
28     (codigo HTTP, erro SSL, timeout ou tipo de conteudo)
29   fim se
30 fim para
31
32 L <- URLs ok cujo caminho decodificado nao contem letras de escrita nao latina

```

D.2 URLS DE PHISHING

As URLs candidatas de phishing foram reunidas a partir do Phishing Database (Phishing Database Project, 2026) e do PhishTank (Cisco Talos Intelligence Group, 2026) e só foram aceitas quando ainda respondiam com conteúdo HTML no momento da verificação. O procedimento da Listagem D.3 distingue três situações: página removida (códigos 404 e 410), acesso bloqueado (401 e 403) e limitação de taxa (429). Nesta última, a URL recebeu novas tentativas com espera, em vez de ser descartada de imediato. As URLs aceitas passaram, em seguida, pelo mesmo filtro de escrita não latina aplicado à classe legítima.

Listing D.3 – Seleção das URLs de phishing ativas

```
1 Entrada : candidatas C; filtro de Bloom B (2^28 bits, 7 funcoes de hash)
2 Saida   : conjunto P de pares (u, 1)
3
4 para cada lote de 5.000 candidatas ausentes de B, em paralelo (128 threads), faca
5     para cada c em {u} uniao {u com http, se u usa https} faca
6         r <- HEAD(c), seguindo ate 10 redirecionamentos
7         se status(r) em {404, 410} entao descartar u           # removida
8         senao se status(r) em {401, 403} ou outro 4xx exceto 405 entao
9             descartar u                                       # bloqueada
10        senao se status(r) >= 500 entao descartar u
11        senao se status(r) em [200, 399] e Content-Type HTML ou ausente entao
12            aceitar c
13        senao
14            r <- GET(c) com Range: bytes=0-2047
15            se Content-Type ausente entao
16                r <- GET(c) com Range: bytes=0-8191
17            fim se
18            aceitar c se a resposta for HTML, pelas mesmas regras
19        fim se
20    fim para
21 fim para
22 # em cada requisicao: codigo 429 -> ate 3 novas tentativas, respeitando
23 #   Retry-After (maximo de 60 s) e pausa por host;
24 #   codigos 408, 425, 5xx, timeout ou falha de conexao -> ate 2 novas
25 #   tentativas com espera exponencial
26
27 P <- URLs aceitas cujo caminho nao contem letras de escrita nao latina
```

D.3 COMPOSIÇÃO DO CONJUNTO E CÁLCULO DOS ATRIBUTOS

As duas classes foram unidas com 3% a mais de URLs legítimas que de phishing. Quando uma mesma URL aparecia nas duas classes, prevalecia o rótulo de phishing, e a amostra legítima era completada com outra URL (Listagem D.4). Cada registro recebeu então um identificador determinístico, os 25 atributos de URL e os atributos extraídos do HTML capturado (Listagem D.5). O URL Complexity Score e o Domain_Similarity seguem as Listagens B.1 e C.1.

Listing D.4 – Composição do conjunto rotulado

```
1 Entrada : P (phishing), L (legitimas), razao lambda = 1,03, semente s = 42
```

```

2 Saida : conjunto D de pares (u, y), embaralhado
3
4 remover duplicatas de P e de L
5 n_L <- arredondar(lambda * |P|)
6 S <- amostra sem reposicao de n_L URLs de L, com semente s
7 X <- S intersecao P
8 S <- S - X # conflito de rotulo: vale phishing
9 completar S ate n_L com URLs de L - (P uniao S), com semente s + 1
10 registrar X em arquivo de auditoria
11 D <- {(u, 1) : u em P} uniao {(u, 0) : u em S}
12 embaralhar D com semente s

```

Listing D.5 – Cálculo dos atributos de URL e de HTML

```

1 Entrada : conjunto D; conjunto de referencia R (Top 10 Million Domains,
2 Open PageRank)
3 Saida : tabela com id, url, atributos de URL, atributos HTML e status
4
5 para cada (u, y) em D faca
6 id <- 12 primeiros digitos hexadecimais de BLAKE2b("42" | u | y)
7 fim para
8 interromper se houver colisao de id
9
10 # ----- atributos de URL (sem acesso a rede) -----
11 para cada (u, y) em D faca
12 calcular os 25 atributos lexicais, incluindo URLC (Listagem urlc)
13 Domain_Similarity <- similaridade maxima de host(u) contra R
14 (Listagem usimax)
15 fim para
16
17 # ----- captura do HTML (com checkpoint) -----
18 para cada id ainda sem snapshot, em paralelo (32 threads), faca
19 t0 <- relógio
20 r <- GET(u), seguindo redirecionamentos; conexao 3 s, leitura 10 s;
21 corpo truncado em 2 MB
22 salvar snapshot JSON com HTML, codigo HTTP, URL final, Content-Type,
23 elapsed_ms = relógio - t0 e data da captura
24 se a requisicao falhou entao
25 salvar snapshot com HTML vazio e o tipo de erro # registro mantido
26 fim se
27 fim para
28
29 # ----- atributos HTML (sem acesso a rede) -----
30 para cada bloco de 1.600 snapshots, em paralelo (24 threads), faca
31 se HTML vazio entao
32 atributos HTML <- 0 (modo 'empty')
33 senao se HTML tem ate 500.000 caracteres e nao parece XML entao
34 analisar com BeautifulSoup; em caso de erro, usar expressoes regulares
35 senao
36 analisar com expressoes regulares
37 fim se
38 calcular os atributos HTML e registrar o modo de analise usado
39 fim para
40
41 unir atributos de URL e HTML por id (relacao 1:1), verificando
42 ausencias e duplicatas

```

D.4 CUSTO DE AQUISIÇÃO DA EVIDÊNCIA WEB

Durante a captura, cada *snapshot* registrou o tempo decorrido entre o envio da requisição e o recebimento completo do corpo da resposta (*elapsed_ms*). Esse intervalo reúne, sem separá-las, a resolução DNS, o estabelecimento da conexão TCP, a negociação TLS, os redirecionamentos e o *download* do HTML. A Tabela D.2 resume esse tempo nas 216.453 capturas concluídas, ao lado do tamanho do HTML recebido e do tempo de extração dos atributos. A mediana ficou em 1.390 ms, e uma em cada vinte páginas levou mais de 4,8 s. As páginas legítimas demoraram mais que as de phishing (mediana de 1.868 ms contra 959 ms), o que acompanha o tamanho do conteúdo: a mediana do HTML legítimo foi de 96,3 KB, e a do phishing, de 7,5 KB.

Tabela D.2 – Tempo de aquisição, tamanho do HTML e tempo de extração dos atributos HTML.

Classe	<i>n</i>	Média	P50	P90	P95	P99
<i>Tempo de aquisição (ms)</i>						
Legítima	111.421	2.299	1.868	4.374	5.406	8.303
Phishing	105.032	1.392	959	2.939	3.946	6.450
Total	216.453	1.859	1.390	3.851	4.843	7.697
<i>Tamanho do HTML (KB)</i>						
Legítima	111.421	179,8	96,3	405,7	632,5	1.496,6
Phishing	105.032	61,9	7,5	120,5	217,1	1.233,1
Total	216.453	122,6	46,3	297,7	503,5	1.233,4
<i>Extração dos atributos HTML (ms), todas as páginas</i>						
Legítima	112.315	27,26	20,18	57,68	79,29	130,01
Phishing	109.045	5,86	1,96	15,24	25,93	56,89
Total	221.360	16,72	7,40	42,22	60,84	109,24

O *builder* da etapa 6 não registrou quanto tempo cada página levou para ser analisada. Esse custo foi medido depois, sem acesso à rede, repetindo a extração dos atributos em todos os 221.360 *snapshots*, um a um e em uma única *thread*. A leitura do arquivo em disco consumiu em média 0,56 ms por página, e a análise do HTML com o cálculo dos atributos, 16,72 ms, com mediana de 7,40 ms e percentil 95 de 60,84 ms. As páginas legítimas, maiores, exigiram em média 27,26 ms, contra 5,86 ms das de phishing. Do total, 91,3% das páginas foram analisadas com BeautifulSoup, 5,4% recorreram às expressões regulares e 3,3% (7.224) não tinham HTML a analisar. Esse último grupo reúne as 4.907 capturas sem sucesso e as respostas concluídas que chegaram com corpo vazio.

A aquisição da página custa, portanto, mais de cem vezes o que custa que a extração dos atributos, e é esse custo, e não o da inferência dos modelos, que domina uma análise que dependa do HTML. Em uso real, ele se soma ao tempo de inferência reportado no capítulo de resultados. Os valores devem ser lidos como aproximação: a captura rodou com 32 requisições simultâneas a partir de uma única máquina, o que acrescenta alguma espera do lado do cliente, e os tempos dependem das condições da rede e dos servidores no período da coleta. Separar a duração de cada fase (DNS, TLS, primeiro *byte* e *download*)

exigiria instrumentar a conexão, o que não foi feito neste trabalho.

D.5 DIFERENÇAS DE COLETA ENTRE AS CLASSES

Como as duas classes vieram de fontes distintas e passaram por tratamentos distintos, parte das diferenças entre elas decorre da própria coleta, e não do comportamento do phishing. A Tabela D.3 compara as classes em atributos sensíveis a esse efeito.

Tabela D.3 – Atributos sensíveis à forma de coleta, por classe.

Atributo	Legítima	Phishing
URLs com esquema <code>https</code>	100,0%	58,5%
URLs com esquema <code>http</code>	0,0%	41,5%
<i>Host</i> iniciado por <code>www.</code>	0,0%	3,2%
URLs com <i>query string</i>	4,9%	16,6%
Média de <code>num_subdomains</code>	0,14	0,79
Média de <code>Domain_Similarity</code>	90,5	53,9
<code>Domain_Similarity</code> igual a 100	76,0%	24,8%
Média do URLC	13,4	36,1
Capturas de HTML sem sucesso	894 (0,8%)	4.013 (3,7%)

Três pontos pedem cuidado. O primeiro diz respeito ao esquema e ao prefixo `www.`. A lista Tranco registra domínios sem esse prefixo (apenas 95 das 999.999 entradas o trazem), e as URLs legítimas foram coletadas a partir desses domínios, como `https://domínio`; a curadoria manteve o esquema `https` e o *host* sem `www.` (Listagens D.1 e D.2). As URLs de phishing, por sua vez, foram mantidas como publicadas. Por essa razão, os atributos `is_https` e `is_http` separam as classes, em boa parte, pela forma de coleta, e a ausência de `www.` contribui para a menor contagem de subdomínios da classe legítima. O segundo é que as URLs legítimas partiram da lista Tranco (Le Pochat et al., 2019), formada por domínios populares, e o `Domain_Similarity` compara cada *host* com o Top 10 Million Domains da Open PageRank (Open PageRank, 2026), lista do mesmo tipo e que também registra os nomes sem o prefixo `www.`. Como boa parte dos domínios legítimos consta das duas listas, a correspondência exata é frequente nessa classe. Por isso, 76% delas alcançam o valor máximo, contra 24,8% das URLs de phishing, e parte do poder discriminativo desse atributo vem do critério de amostragem. O terceiro é que as 4.907 capturas sem sucesso permaneceram no conjunto com atributos HTML zerados, e 81,8% delas são de phishing. A ausência de HTML acaba funcionando como indício de classe. Situação parecida ocorre com ao menos 5.910 capturas concluídas cuja resposta final trouxe código de erro (4xx ou 5xx): nesses casos, os atributos foram extraídos da página de erro devolvida pelo servidor, e não do conteúdo original.

Nenhum experimento específico foi conduzido para isolar esses efeitos da característica de classe. Os atributos de URL diretamente afetados pela normalização e o `Domain_Similarity` permanecem no esquema final, e a leitura dos resultados obtidos no HybridPhisher deve levar essa limitação em conta.