UNIVERSIDADE FEDERAL DO PARANÁ

ALEIDA MOPI LAFUENTE

CLASSIFICAÇÃO DE URLS MALICIOSAS COM BASE EM ATRIBUTOS LEXICAIS USANDO *Random Forest* E *XGBoost*

CURITIBA, PARANÁ
2025

ALEIDA MOPI LAFUENTE

CLASSIFICAÇÃO DE URLS MALICIOSAS COM BASE EM ATRIBUTOS LEXICAIS USANDO *Random Forest* E *XGBoost*

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada no Programa de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, da Universidade Federal do Paraná.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño.

CURITIBA, PARANÁ
2025

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **ALEIDA MOPI LAFUENTE**, intitulada: **CLASSIFICAÇÃO DE URLS MALICIOSAS COM BASE EM ATRIBUTOS LEXICAIS USANDO Random Forest E XGBoost**, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 23 de Junho de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Classificação de URLs maliciosas com base em atributos lexicais usando *Random Forest* e *XGBoost*

Aleida Mopi Lafuente
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
aleida.lafuente@ufpr.br

Prof. Dr. Razer Anthom Nizer Rojas Montaño
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
razer@ufpr.br

***Resumo*—Com o avanço das ameaças cibernéticas, a automação do processo de *takedown* de URLs maliciosas tornou-se uma estratégia essencial na segurança digital. Este trabalho realiza uma análise comparativa dos algoritmos *Random Forest* e *XGBoost* na classificação de URLs como benignas ou maliciosas, utilizando exclusivamente atributos lexicais. A base de dados empregada foi a *ISCX-URL2016*, composta por 36.708 URLs distribuídas em cinco categorias (benignas, *phishing*, *spam*, *malware* e *defacement*) e descritas por 79 atributos estruturais. A metodologia adotou divisão estratificada dos dados (80/20), validação cruzada *k-fold* (*k*=10) e otimização de hiperparâmetros via *grid search*. Os resultados indicam que ambos os modelos apresentaram alto desempenho, sendo que o *Random Forest* obteve acurácia de 98,12% e *F1-score* macro de 0,98, enquanto o *XGBoost* superou ligeiramente com acurácia de 98,62% e *F1-score* macro de 0,99. Já a análise das matrizes de confusão revelou que o *XGBoost* apresentou menor taxa de erros, especialmente nas classes "*phishing*" e "*spam*". A importância dos atributos evidenciou que características como comprimento da URL, número de *tokens* e entropia dos caracteres foram as mais relevantes para a classificação, com os resultados comprovando que é possível atingir alto desempenho utilizando apenas atributos lexicais e deste modo contribuindo para processos de *takedown* mais rápidos, eficientes e com menor custo computacional.**

***Palavras-chave*—segurança cibernética, takedown, URLs maliciosas, Random Forest, XGBoost**

***Resumo*—With the advancement of cyber threats, automating the takedown process of malicious URLs has become an essential strategy in digital security. This study presents a comparative analysis of the Random Forest and XGBoost algorithms for classifying URLs as benign or malicious, using exclusively lexical features. The ISCX-URL2016 dataset, with 36,708 URLs across five categories and 79 structural attributes, was used. The methodology included stratified data splitting (80/20), 10-fold cross-validation, and hyperparameter optimization via grid search. Both models performed well: Random Forest achieved 98.12% accuracy and a macro F1-score of 0.98, while XGBoost reached 98.62% accuracy and a macro F1-score of 0.99. XGBoost showed lower error rates, notably in "phishing" and "spam" classes. Feature importance analysis highlighted URL length, token count, and character entropy as most relevant. The findings confirm high performance using only lexical features, aiding faster, more efficient, and less resource-intensive takedown efforts.**

***Index Terms*—CyberSecurity, takedown, malicious URLs, Random Forest, XGBoost**

I. DESENVOLVIMENTO

Restando quase duas décadas para o centenário de um processo que remonta à Segunda Grande Guerra e à Guerra Fria e que se aprofunda em complexidade em um ritmo evolucionário de mais de 50 anos, o uso de dados computacionais para práticas de inteligência é um anseio que se fomenta igualmente na esfera da *Inteligência de Ameaças Cibernéticas*.

Para efeito de assertividade no entendimento da finalidade da pesquisa que será exposta neste artigo, é justo definir do que trata esta disciplina da segurança cibernética. Conforme descrito pelo reputado *Fórum de Equipes de Segurança e Resposta a Incidentes* (*FIRST*) [1], ela é

> "a coleta, a análise e o compartilhamento de informações relacionadas às operações de uma empresa no espaço cibernértico e, até certo ponto, no espaço físico. É projetada para informar todos os níveis de tomadores de decisões. Essa análise é desenhada para ajudar a manter a consciência de ameaças atuais e futuras."

Neste âmbito, a análise de URLs (*Uniform Resource Locator*) é uma técnica significativamente difundida como parte da tática de defesa denominada *takedown*. Conforme o seu nome sugere, o *takedown* é o processo que remove ou inativa conteúdos digitais maliciosos, estando dentre seus cenários mais recorrentes os casos de *phishing*, *spam*, *defacement*, download de arquivos de *malware* e páginas e aplicações falsas [2].

Entre os principais tipos de ameaças combatidas pelo *takedown*, destacam-se diversas categorias de conteúdo malicioso que visam enganar, explorar ou prejudicar usuários finais e organizações. Na lista à sequência, apresentam-se os formatos mais recorrentes dessas ameaças:

- *Phishing*: tentativa de enganar usuários para roubar dados sensíveis por meio de páginas falsas [3].
- *Spam*: envio massivo de mensagens não solicitadas, geralmente com conteúdo publicitário ou malicioso [4].
- *Defacement*: ataque que altera visualmente um site, substituindo seu conteúdo original [5].
- *Malware*: disponibilização de arquivos maliciosos para infectar dispositivos [6].

Considerando as inúmeras possibilidades de emprego indevido de tais artifícios para interesses ilegítimos e de eventuais

incidentes cibernéticos derivados desse tipo de atividade, o impacto da existência de URLs maliciosas é significativo em várias instâncias, sejam elas individuais ou a níveis corporativos [7]. Logo, a lógica em que se automatiza o processo de *takedown* não surpreendentemente é explorada como um negócio lucrativo em um número significativo de empreendimentos, promovendo um uso mais seguro da internet na medida em que minimiza os riscos de possíveis danos financeiros, reputacionais e de privacidade.

Quando automatizado, o *takedown* contempla o estágio em que se identifica, quando constatado que não se trata de uma URL benigna, em que classe de prática maliciosa se enquadra determinada URL em vista de atributos variados, analisando-se referências textuais a marcas registradas no domínio descrito pela URL, referências textuais e gráficas a tais marcas no corpo da página referenciada, downloads automáticos de arquivos, presença na página de URLs listadas como *Indicadores de Comprometimento* (*IoC*) em bases de dados de segurança, dentre outras variáveis pertinentes a esse escrutínio [2]. Depois de verificado o gênero da atividade maliciosa em curso, o pedido de retirada da URL e de seu respectivo conteúdo implica ainda na conferência de políticas da plataforma em que a URL foi registrada e no envio de resposta a um formulário detalhado em que se evidencia a regra infringida, resultando finalmente no início do processo de validações da plataforma ante as alegações e o retorno de seu veredito.

Neste contexto, este trabalho se dispõe à análise comparativa entre dois modelos de *Inteligência Artificial* (*IA*) quanto à sua eficiência classificando um conjunto de URLs, privilegiando para tanto o uso de atributos lexicais (propriedades associadas às palavras em uma língua natural) para a referida análise – o conjunto de dados utilizado abstrai uma série de características puramente lexicais, decompondo-as em diversos níveis [8]. Em identificando os materiais infratores somente fundamentado em tais atributos, o propósito do estudo é que o modelo que se demonstre mais eficiente contribua reduzindo os tempos de análises de URLs no processo de *takedown*, idealmente até mesmo eliminando a necessidade de verificações mais onerosas envolvendo atributos da página referenciada pela URL, como textos, imagens e vários tipos de arquivos.

O estudo proposto foi desenvolvido em *Python*, e o relatório técnico expondo os devidos detalhamentos a respeito de cada etapa encontra-se à sequência deste tópico, pontuando descrições fundamentais do conjunto de dados elegido, de métodos empregados, tecnologias utilizadas e resultados obtidos.

A. Descrição dos dados

Como insumo para este trabalho, o conjunto de dados selecionado é o *URL dataset* (*ISCX-URL2016*), compilado em uma pesquisa realizada pelo *Instituto Canadense para Cibersegurança* (*Canadian Institute for Cybersecurity*) e publicado no sítio eletrônico da Universidade de *New Brunswick*, localizada no Canadá [9]. Esta base categoriza exatamente 36.708 tuplas – exemplares de URLs classificadas em 5 (cinco) categorias:

- benignas (totalizando 7.781 registros)
- *defacements* (totalizando 7.930 registros)
- *malwares* (totalizando 6.712 registros)
- *phishings* (totalizando 7.586 registros)
- *spams* (totalizando 6.698 registros)

Cada URL é descrita em 79 atributos lexicais representados em valores numéricos, consoante especificado na Tabela I e na Tabela II. Estes atributos abstraem características significativamente diversificadas de uma URL como comprimento e contagem de *tokens*, informações de domínio, caminho e arquivo, indicadores de risco (uso de palavras consideradas sensíveis), quantidades de letras e dígitos, medidas de aleatoriedade de caracteres e muitos outros elementos levantados para a análise.

B. Métodos

Visando o desenvolvimento da análise de atributos lexicais de URLs e sua consequente influência em classificações, é executada uma rotina de procedimentos técnicos ordenados sistematicamente para o devido refinamento do resultado pretendido.

Esse processo consiste em uma sequência de 5 etapas centrais fundadas na base de dados definida:

1) composição da base de dados
2) pré-processamento de dados
3) preparo do treinamento
4) treinamento
5) predições

Como ilustrado na Figura 1, a estrutura de métodos aplicados culmina no entendimento do modelo de *IA* mais eficiente classificando URLs como uma de 5 classes elencadas, indicando igualmente os atributos mais dominantes no resultado em ambos os modelos estudados. À continuidade deste tópico, apresentam-se as descrições de cada etapa mencionada anteriormente.

1) Composição da base de dados: conforme exposto anteriormente, o conjunto de dados consolidado para o presente estudo é proveniente de um acervo disponibilizado pelo *Instituto Canadense para Cibersegurança*, que previamente realizou a coleta de URLs, extraiu características textuais de tais registros, compilou os dados obtidos e os organizou em uma mesma base pública.

Os dados advêm de diversas fontes: as benignas foram coletadas via uma ferramenta para análise de dados de tráfego web, a *Alexa* (subsidiária da *Amazon* e descontinuada desde 2022), tendo sido selecionadas as 35.300 URLs que registravam os maiores volumes de acessos no período da coleta. Depois de um processo de *web crawling* (processo de um software que navega e coleta dados de sites na internet) executado via *Heritrix*, os domínios listados pela *Alexa* originaram um segundo compilado de URLs associadas a tais domínios, havendo um terceiro momento de processamento de dados em que foram removidas as duplicações. Finalmente, as URLs filtradas foram submetidas a verificações quanto à sua legitimidade via *VirusTotal* [10], restando então exclusivamente as URLs rotuladas como benignas.

Tabela I
ATRIBUTOS DA BASE DE DADOS | PARTE 1

Atributo	**Descrição**
Querylength	Comprimento total da URL consultada
domain_token_count	Número de tokens no domínio
path_token_count	Número de tokens no caminho (path)
avgdomaintokenlen	Comprimento médio dos tokens no domínio
longdomaintokenlen	Comprimento do maior token no domínio
avgpathtokenlen	Comprimento médio dos tokens no caminho (path)
tld	Top Level Domain (ex.: .com, .org)
charcompvowels	Proporção de vogais na URL
charcompace	Proporção de caracteres especiais na URL
ldl_url	Comprimento do token mais longo na URL
ldl_domain	Comprimento do token mais longo no domínio
ldl_path	Comprimento do token mais longo no caminho (path)
ldl_filename	Comprimento do token mais longo no nome do arquivo
ldl_getArg	Comprimento do token mais longo nos argumentos da URL
dld_url	Distância de Levenshtein do token mais longo na URL
dld_domain	Distância de Levenshtein do token mais longo no domínio
dld_path	Distância de Levenshtein do token mais longo no caminho (path)
dld_filename	Distância de Levenshtein do token mais longo no nome do arquivo
dld_getArg	Distância de Levenshtein do token mais longo nos argumentos
urlLen	Comprimento total da URL
domainlength	Comprimento do domínio
pathLength	Comprimento do caminho (path)
subDirLen	Comprimento dos subdiretórios na URL
fileNameLen	Comprimento do nome do arquivo na URL
this.fileExtLen	Comprimento da extensão do arquivo
ArgLen	Comprimento total dos argumentos (query string)
pathurlRatio	Razão entre o comprimento do caminho e da URL
ArgUrlRatio	Razão entre o comprimento dos argumentos e da URL
argDomanRatio	Razão entre o comprimento dos argumentos e do domínio
domainUrlRatio	Razão entre o comprimento do domínio e da URL
pathDomainRatio	Razão entre o comprimento do caminho e do domínio
argPathRatio	Razão entre o comprimento dos argumentos e do caminho
executable	Indica se há executável (.exe, .bat, etc) na URL (0 ou 1)
isPortEighty	Indica se a URL utiliza a porta 80 (0 ou 1)
NumberofDotsinURL	Número de pontos (.) na URL
ISIpAddressInDomainName	Indica se há um IP no nome do domínio (0 ou 1)
CharacterContinuityRate	Taxa de continuidade dos caracteres na URL
LongestVariableValue	Comprimento do maior valor de variável na query
URL_DigitCount	Número total de dígitos na URL
host_DigitCount	Número de dígitos no domínio (host)
Directory_DigitCount	Número de dígitos no diretório
File_name_DigitCount	Número de dígitos no nome do arquivo
Extension_DigitCount	Número de dígitos na extensão do arquivo
Query_DigitCount	Número de dígitos nos argumentos da URL
URL_Letter_Count	Número total de letras na URL
host_letter_count	Número de letras no domínio
Directory_LetterCount	Número de letras no diretório
Filename_LetterCount	Número de letras no nome do arquivo
Extension_LetterCount	Número de letras na extensão do arquivo
Query_LetterCount	Número de letras nos argumentos da URL

Fonte: a autora (2025)

Tabela II
ATRIBUTOS DA BASE DE DADOS | PARTE 2

Atributo	Descrição
LongestPathTokenLength	Comprimento do maior token no caminho (path)
Domain_LongestWordLength	Comprimento do maior token no domínio
Path_LongestWordLength	Comprimento do maior token no caminho (path)
sub-Directory_LongestWordLength	Comprimento do maior subdiretório
Arguments_LongestWordLength	Comprimento do maior argumento da query string
URL_sensitiveWord	Indica se há palavras sensíveis/suspeitas na URL
URLQueries_variable	Número de variáveis na query string
spcharUrl	Número de caracteres especiais na URL
delimeter_Domain	Número de delimitadores no domínio
delimeter_path	Número de delimitadores no caminho (path)
delimeter_Count	Número total de delimitadores na URL
NumberRate_URL	Proporção de números na URL
NumberRate_Domain	Proporção de números no domínio
NumberRate_DirectoryName	Proporção de números no diretório
NumberRate_FileName	Proporção de números no nome do arquivo
NumberRate_Extension	Proporção de números na extensão do arquivo
NumberRate_AfterPath	Proporção de números após o caminho (nos argumentos)
SymbolCount_URL	Número de símbolos na URL
SymbolCount_Domain	Número de símbolos no domínio
SymbolCount_Directoryname	Número de símbolos no diretório
SymbolCount_FileName	Número de símbolos no nome do arquivo
SymbolCount_Extension	Número de símbolos na extensão do arquivo
SymbolCount_Afterpath	Número de símbolos após o caminho (nos argumentos)
Entropy_URL	Entropia da URL (nível de aleatoriedade)
Entropy_Domain	Entropia do domínio
Entropy_DirectoryName	Entropia do diretório
Entropy_Filename	Entropia do nome do arquivo
Entropy_Extension	Entropia da extensão do arquivo
Entropy_Afterpath	Entropia dos argumentos após o caminho
URL_Type_obf_Type	Tipo de ofuscação presente na URL

Fonte: a autora (2025)

As URLs de *spams* foram coletadas da base de dados pública *WEBSPAM-UK2007*; as de *phishings* retiradas do repositório de sítios eletrônicos ativos de *phishing* como o *OpenPhish* [11]; as vinculadas a *malwares* obtidas do *DNS-BH*, um projeto que mantém uma lista de websites de *malware*, e as URLs catalogadas como *defacements* pertencem a uma lista de sítios eletrônicos mais acessados e considerados confiáveis pela *Alexa* e que, no entanto, hospedam ou ofuscam conteúdo malicioso em suas páginas e URLs.

Destarte, os autores da base de dados apuraram 79 características lexicais de cada URL compilada, traduzindo em valores numéricos cada atributo selecionado e tratando possíveis inconsistências como dados faltantes e/ou desestruturados.

Em face de tais especificações, fica evidente que este estudo trata de um *aprendizado supervisionado*, em que o conjunto de dados está rotulado precisamente; para cada amostra de URL há, portanto, uma classe atribuída dentre as 5 delineadas. Sendo as variáveis de saída rótulos, a tarefa de aprendizado é do tipo *classificatória*, dado que o modelo de *IA* associa atributos lexicais a cada uma dessas classes e potencialmente prevê a classe de novas URLs.

2) Pré-processamento dos dados: o processo de refinamento do conjunto de dados para o seu uso no treinamento do *Random Forest* e do *XGBoost* observou uma série de fundamentos ante as suas especificidades.

Dentre os cenários típicos relativos a tratamento de dados, uma quantidade significativa de condições não se aplicavam à base de dados *ISCX-URL2016* – dados faltantes ou incompletos (visto que a base foi arquitetada satisfazendo parâmetros de qualidade de pesquisas anteriores e de fato apresenta nativamente valores para todos os atributos descritos), dados inconsistentes (existe lógica em todos os valores apresentados para cada atributo), dados redundantes (não há duplicações de registros) e dados ruidosos (eles se encontram padronizados dentro da amplitude de valores esperada).

Nenhuma coluna (atributo) foi removida para efeito da

Figura 1. Etapas do método aplicado no estudo.
Fonte: a autora (2025)

análise, posto que todas as características elencadas possuem significados pertinentes no contexto de atributos lexicais e consequentemente contribuem para as fases de treinamento e *predições*. Não foi realizada análise de correlação de atributos neste primeiro momento, ficando este estudo como indicação para trabalho futuro.

Sendo algoritmos baseados em árvores, o *Random Forest* e o *XGBoost* não demandam que os dados sejam normalizados, pois não se baseiam na distância entre pontos nem são sensíveis às escalas e magnitudes de determinados atributos. Em seu tópico de conclusões, o artigo *The Impact of Feature Scaling In Machine Learning: Effects on Regression and Classification Tasks*, que averigua o impacto do escalonamento de atributos em *machine learning*, denota que métodos de *ensemble* como o *Random Forest* e o *XGBoost* mantêm alto desempenho de modo consistente independentemente de normalizações, sugerindo então estes procedimentos de fato possuem pouco efeito em classificações [12].

3) Preparação do treinamento: na fase de preparamento para os treinamentos, é definida a estratégia de particionamento da base de dados previamente tratada, visando adicionalmente à qualidade da etapa de treinamento as devidas avaliações dos modelos preditivos. Considerando que o problema abordado neste estudo é do tipo **classificatório** e **multiclasse** (em que se identifica o tipo de URL maliciosa), optou-se pelo uso do conjunto de métricas clássicas e adequadas a este cenário baseadas na matriz de confusão: *acurácia* e *F1-score* [13].

Inicialmente, aplicou-se a técnica de *Holdout*, que fragmenta o conjunto de dados em duas partes distintas: uma utilizada para o treinamento e a segunda para o teste de modelos. A base de dados foi segmentada de modo estratificado, assim dispondo-se 80% de tais dados para o conjunto de treinamento e 20% para o conjunto de teste, mantendo-se proporcionalmente o tamanho de cada classe em ambos os subconjuntos. Para tais fins, o método `train_test_split` da biblioteca *scikit-learn* foi utilizado junto a um de seus possíveis parâmetros, o `stratify`, cuja lógica preserva as distribuições de cada classe.

Para incremento da qualidade do treinamento, na medida em que mitiga problemas de sobreajuste (*overfitting*) e assegura cálculos mais confiáveis acerca da performance de ambos os modelos, foi igualmente aplicada a técnica conhecida como *10-fold Stratified Cross-Validation*. Este mecanismo divide o conjunto de treino em 10 subconjuntos mutuamente exclusivos, utilizando nove subconjuntos a cada ciclo iterativo para treinamento e um para validações. À vista disto, o desempenho conclusivo é calculado mediante a média de tais performances individuais em cada rodada, reduzindo a variância em várias métricas.

Tendo finalizado essa etapa, os modelos finais foram treinados empregando 80% de dados originais e reaplicando os parâmetros anteriormente avaliados. Posteriormente, os modelos foram testados usando os 20% de dados restantes, sendo examinados perante as suas matrizes de confusão, relatórios de classificação e curvas *ROC* multiclasse; conjunto de métricas este que favoreceu uma análise mais aprofundada do desempenho em cada classe.

Não menos importante, foi desenhada, finalmente, a análise de atributos mais relevantes, utilizando os valores retornados pelos modelos *Random Forest* e *XGBoost*. Esta análise é a base para interpretar quais características possuem maior impacto em decisões do modelo, podendo potencialmente nortear aprimoramentos ou seleções de variáveis em ambientes de pesquisa onde essas abordagens seriam de muita valia para decisões de negócio.

4) Treinamento: as técnicas de *IA* adotadas neste estudo foram os algoritmos de aprendizado supervisionado **Random Forest** e **XGBoost**, ambos utilizados massivamente em classificações devido à sua assertividade e capacidade para generalizações [14] [15]. O treinamento de cada modelo contemplou a escolha criteriosa de hiperparâmetros, que foram ajustados mediante uma busca em *grid* seguida de análise de sensibilidade, conforme esmiuçado à continuidade.

No caso do *Random Forest*, dois principais hiperparâmetros foram averiguados: o número de árvores na floresta, `n_estimators`, e a quantidade de atributos considerados em ramificações da árvore, `max_features`, seguindo o indicado pelo estudo *Hyperparameters and Tuning Strategies for Random Forest* [16]. Considerando que valores mais altos de `n_estimators` tendem à melhora da estabilidade do modelo (desde que os custos computacionais sejam viáveis), foram testados os valores de 500, 1000, 1500 e 2000 para este hiperparâmetro. Para `max_features`, foram adotados valores baseados na raiz quadrada do número de atributos, uma prática considerada comum em tarefas de classificações, bem como valores superiores para efeito comparativo: 3 (aproximadamente $\sqrt{n}$), 10, 15 e 20, como mostrado na Tabela III. Depois de executada a grade de busca, os resultados foram apurados de modo que o segmento de melhor desempenho fosse identificado, possibilitando eventuais realizações de uma segunda rodada mais refinada de treinamento.

Para o *XGBoost*, os principais hiperparâmetros avaliados foram: `learning_rate` (taxa de aprendizado), `max_depth` (profundidade máxima de cada árvore), `n_estimators` e `subsample` (amostras de dados usadas em cada árvore), todas citadas como hiperparâmetros que previnem o *overfitting* em *XGBoost: A Scalable Tree Boosting System* [17]. Estes parâmetros impactam diretamente a capacidade do modelo na captura de padrões complexos desprovida de sobreajustes. Os valores testados incluíram: `learning_rate` = {0.01, 0.1, 0.2}; `max_depth` = {3, 6, 10}; `n_estimators` = {500, 1000, 1500}; e `subsample` = {0.7, 0.8, 1.0}, estando todos estes dados denotados na Tabela III, também. Regulando estes hiperparâmetros, foi realizada uma busca em *grid* com validação cruzada estratificada de 10 dobras, buscando-se o modelo com melhor *acurácia* média.

Tabela III
HIPERPARÂMETROS UTILIZADOS NOS MODELOS RANDOM FOREST E XGBOOST

Algoritmo	Hiperparâmetro	Valores testados
Random Forest	`n_estimators`	500, 1000, 1500, 2000
	`max_features`	3, 10, 15, 20
XGBoost	`learning_rate`	0.01, 0.1, 0.2
	`max_depth`	3, 6, 10
	`n_estimators`	500, 1000, 1500
	`subsample`	0.7, 0.8, 1.0

Fonte: a autora (2025)

Para ambos os modelos, o processo de ajuste de hiperparâmetros foi conduzido fazendo uso da biblioteca `scikit-learn` integrada às bibliotecas `GridSearchCV` e `StratifiedKFold`, apoiando na qualidade do trabalho de estatística e na consistência de proporções de classe durante as dobras para validações.

Partindo da análise de resultados, os melhores hiperparâmetros identificados foram utilizados no treinamento para os modelos finais, em que foi utilizado o conjunto completo de treino (80% da base); posteriormente, tais modelos foram avaliados usando o conjunto de teste (20%). O desempenho foi medido utilizando as métricas mencionadas no tópico (*Preparação do treinamento*), permitindo as análises comparativa e quantitativa entre os modelos.

5) Predições: à sequência do treinamento e ajuste de hiperparâmetros, os modelos aprimorados foram avaliados utilizando o conjunto de teste previamente separado (20% da base de dados nativa) no objetivo de estimar a capacidade de generalizações para novos dados. Esta etapa corresponde às *predições*, em que os rótulos de amostras do conjunto de teste foram previstos usando os modelos treinados e os resultados comparados a seus rótulos reais para análise de desempenho.

As principais métricas utilizadas nesta análise foram a **acurácia** e o **F1-score macro**, indicadas para classificações multiclasse e seus problemas. A *acurácia* fornece um panorama do desempenho do modelo, enquanto o *F1-score macro* trata mais do desempenho em classes desbalanceadas, pois calcula a média de *F1-scores* individuais em cada classe.

C. Tecnologias

Os experimentos foram desenvolvidos em um notebook próprio da fabricante Dell, modelo Inspiron 3520 contendo um Intel Core i7-1255U de 10 núcleos e 12 threads a 1.7 GHz, 16 GB de memória RAM, 512 GB de SSD e sistema Microsoft Windows 11 Pro (10.0.22000).

Foram utilizadas as seguintes aplicações e bibliotecas:

- Linguagem Python, versão 3.11.9;
- Pacote *pandas*, versão 2.2.2;
- Pacote *numpy*, versão 1.26.4;
- Pacote *scikit-learn*, versão 1.4.2;
- Pacote *xgboost*, versão 2.0.3;
- Pacote *matplotlib*, versão 3.8.4;
- Pacote *seaborn*, versão 0.13.2;

II. RESULTADOS E DISCUSSÕES

Esta seção apresenta os resultados obtidos nos experimentos comparativos entre *Random Forest* e *XGBoost* na tarefa de classificação de URLs maliciosas e benignas. Será discutido como a escolha de hiperparâmetros, o uso de apenas atributos lexicais e as técnicas aplicadas influenciaram as saídas, apresentadas de forma comparativa e individualmente (para cada modelo).

Os resultados denotam que o modelo *XGBoost* superou o *Random Forest* em ambas as métricas de desempenho estabelecidas: *acurácia* e *F1-score macro*. Essa diferença suscita a ideia de que o *XGBoost* foi mais efetivo na captura de padrões de dados de modo equilibrado entre as classes, comportamento

coerente perante o mecanismo de *boosting*, que corrige erros cometidos em iterações anteriores e consequentemente termina produzindo modelos mais ajustados às gradações do conjunto de dados.

Para o *Random Forest*, sendo n número total de atributos disponíveis nos dados de entrada do modelo, foram testados `n_estimators` $= \{500, 1000, 1500, 2000\}$ e `max_features` $= \{\sqrt{n}, 10, 15, 20\}$; constatou-se que todos os valores de `n_estimators` resultaram em *acurácia* média de 98,12% na validação cruzada, optando-se por `n_estimators` = 500 (princípio da economia de recursos) e `max_features` = $\sqrt{n}$. Para o *XGBoost*, a *grid* incluiu `learning_rate` $= \{0{,}01; 0{,}1; 0{,}2\}$, `max_depth` $= \{3; 6; 10\}$, `n_estimators` $= \{500; 1000; 1500\}$ e `subsample` $= \{0{,}7; 0{,}8; 1{,}0\}$, obtendo-se o melhor desempenho com `learning_rate` $= 0{,}1$, `max_depth` $= 6$, `n_estimators` $= 1000$ e `subsample` $= 0{,}8$. Todos estas métricas e os respectivos *F1-scores* aferidos encontram-se na Tabela IV.

Tabela IV
DESEMPENHO DOS MODELOS RANDOM FOREST E XGBOOST

Modelo	Hiperparâmetros	Acurácia	F1-score
RF	500 trees, mf=$\sqrt{n}$	98.12%	0.98
XGB	lr=0.1, depth=6,1000 trees, subs=0.8	98.62%	0.99

Fonte: a autora (2025)

A. Medida de qualidade dos modelos

A *acurácia* e o *F1-score macro* foram as métricas de referência, adequadas ao caráter *multiclasse* do problema. O *Random Forest* alcançou 98,12% de *acurácia* e *F1-score macro* de 0,98; o *XGBoost* obteve 98,62% de *acurácia* e *F1-score macro* de 0,99.

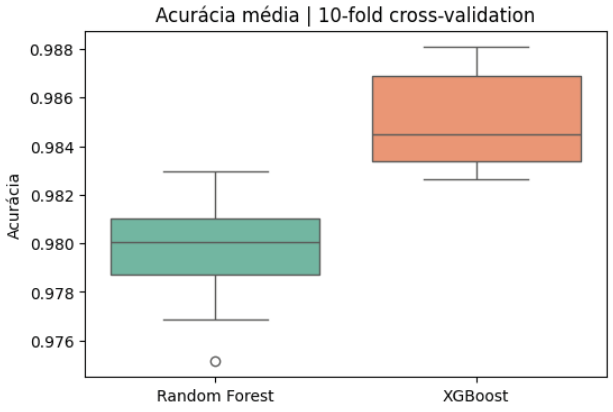

Figura 2. Acurácia média do Random Forest e XGBoost | 10-fold Stratified Cross-Validation
Fonte: a autora (2025)

B. Matrizes de Confusão

A matriz de confusão do *Random Forest* está na Figura 3, mostrando que as principais confusões ocorreram entre as classes *spam* e *phishing*. Já a matriz do *XGBoost* (Figura 4) revela melhor equilíbrio, ainda que em proporções não muito superiores. As classes, em sua maioria, apresentam menores taxas de falsos negativos e falsos positivos, havendo duas exceções pontuais: as confusões entre *defacement* e *phishing* e entre *defacement*. e *spam*. Quanto às linhas diagonais de ambas as matrizes, comparando as intersecções correspondentes, infere-se que o XGBoost de fato obteve mais eficácia classificando as URLs, já que a única métrica de desempenho abaixo da métrica alcançada pelo Random Forest foi a referente a acertos classificando *defacement*.

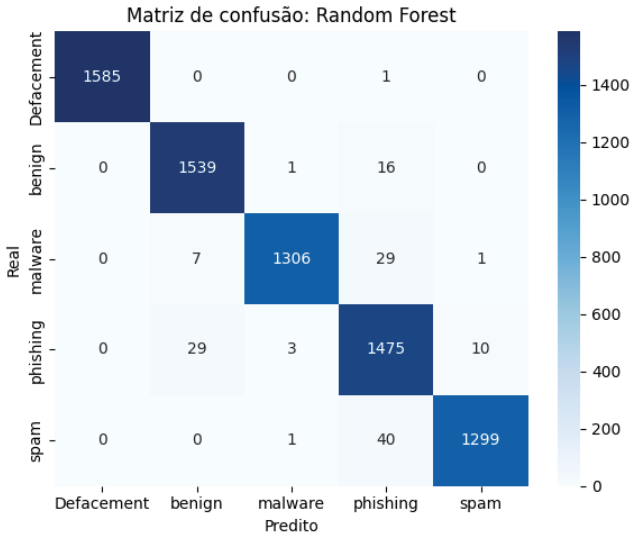

Figura 3. Matriz de confusão do Random Forest
Fonte: a autora (2025)

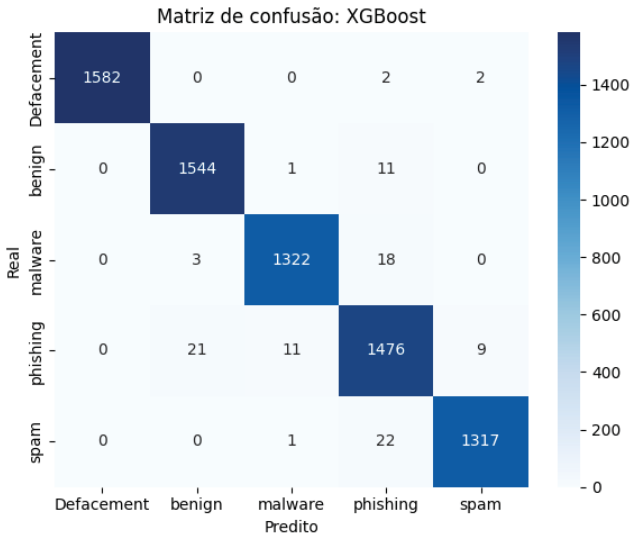

Figura 4. Matriz de confusão do XGBoost
Fonte: a autora (2025)

C. Discussões

Os resultados indicam que o *XGBoost* superou o *Random Forest* em todas as métricas avaliadas, sugerindo que o mecanismo de *boosting* capturou padrões lexicais mais sutis. Ainda assim, o *Random Forest* mostrou-se bastante competitivo, com vantagem em simplicidade de interpretação. A seleção de hiperparâmetros foi o ponto chave para os resultados: no *Random Forest*, a invariância de desempenho para diferentes valores de `n_estimators` permitiu economizar recursos computacionais; no *XGBoost*, ajustes como o

$$\texttt{learning\_rate} = 0{,}1 \quad \text{e} \quad \texttt{subsample} = 0{,}8$$

equilibraram precisão e variância.

Comparando estes resultados a um estudo semelhante, o *Detection of Malicious URLs using Machine Learning based on Lexical Features* [18], em que a mesma base de dados utilizada neste trabalho foi reduzida a 23 atributos lexicais e cuja análise comparativa envolveu dentre mais modelos o Random Forest e o XGBoost, diferentemente do aferido aqui, o Random Forest obteve o desempenho superior em suas classificações, atingindo uma acurácia de 96.6% frente à acurácia de 93.2% do XGBoost. Considerando que os experimentos de ambos os trabalhos foram executados em ambientes muito similares, levanta-se a hipotése primária de que essa diferença no resultado se deve em alguma medida a esse uso distinto de atributos, que contrastam em quantidade e qualidade (categoria, especificidade).

Já quanto à análise de importância de atributos (Figura 5 e Figura 6), ela aponta que as medidas de aleatoriedade dos caracteres, número de pontos usados em URLs e a contagem de *tokens* de domínio são os fatores mais determinantes na análise de URLs dentre os 15 levantados como mais influentes, constando em posições mais altas de relevância em ambos os gráficos. Ou seja, em um cenário de uso pragmático deste resultado para automações de *takedown*, vislumbra-se que estes atributos seriam pontos chave para o aprimoramento de futuros modelos de clasificação que tenham como foco agilidade e custos computacionais reduzidos, deste modo reduzindo o escopo nativo de 79 atributos para cada URL.

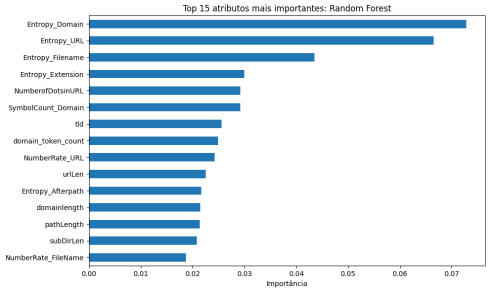

Figura 5. Top 15 atributos mais importantes usando o modelo Random Forest
Fonte: a autora (2025)

Para trabalhos futuros, almeja-se explorar outros conjuntos que combinem as duas técnicas e investigar adições de outros atributos lexicais cuja importância seja potencialmente significativa para a análise objetivada, possivelmente incluindo até mesmo atributos baseados em conteúdo de página de modo a aprimorar ainda mais a capacidade de generalização dos modelos.

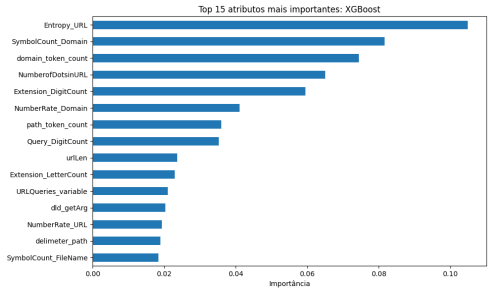

Figura 6. Top 15 atributos mais importantes usando o modelo XGBoost
Fonte: a autora (2025)

Referências

[1] FIRST, "Forum of incident response and security teams - first | introduction to cti as a general topic." Disponível em: <https://www.first.org/global/sigs/cti/curriculum/cti-introduction>. Acesso em: 20 abr. 2025.

[2] Axur, "Experimente o melhor takedown do mundo | o que é um takedown?." Disponível em: <https://www.axur.com/pt-br/takedown/>. Acesso em: 20 abr. 2025.

[3] K. I. Encyclopedia, "Phishing." Disponível em: <https://encyclopedia.kaspersky.com/glossary/phishing/>. Acesso em: 20 mai. 2025.

[4] ScienceDirect, "Spam." Disponível em: <https://www.sciencedirect.com/topics/computer-science/spam>. Acesso em: 20 mai. 2025.

[5] M. Albalawi, R. Aloufi, N. Alamrani, N. Albalawi, A. Aljaedi, and A. R. Alharbi, "Website defacement detection and monitoring methods: a review." Disponível em: https://www.researchgate.net/publication/365055780_Website_Defacement_Detection_and_Monitoring_Methods_A_Review. Acesso em: 20 mai. 2025.

[6] S. Kramer and J. C. Bradfield, "A general definition of malware." Disponível em: <https://link.springer.com/article/10.1007/s11416-009-0137-1>. Acesso em: 20 mai. 2025.

[7] K. Oloyede, C. Obunadike, S. Yufenyuy, E. Elom, A.-W. Bello, S. Olisah, C. Obunadike, O. Ogunleye, and S. Adeniji, "Impact of web (url) phishing and its detection," *International Journal of Scientific Research and Management (IJSRM)*, vol. 12, pp. 484–493, 2024. Disponível em: https://www.semanticscholar.org/reader/024254636c5b652794d6b5dab00b60133cb13e8b. Acesso em: 28 jun. 2025.

[8] M. S. I. Mamun, M. A. Rathore, A. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious urls using lexical analysis." Disponível em: https://cyberlab.usask.ca/papers/Mamun2016_Chapter_DetectingMaliciousURLsUsingLex.pdf. Acesso em: 24 abr. 2025.

[9] C. I. for Cybersecurity, "Iscx url dataset 2016." Disponível em: <https://www.unb.ca/cic/datasets/url-2016.html>. Acesso em: 12 mai. 2025.

[10] VirusTotal, "Virustotal - free online virus, malware and url scanner." Disponível em: <https://www.virustotal.com/>. Acesso em: 04 abr. 2025.

[11] OpenPhish, "Openphish - automated phishing intelligence." Disponível em: <https://openphish.com/>. Acesso em: 02 jun. 2025.

[12] J. M. H. Pinheiro, S. V. B. de Oliveira, T. H. S. Silva, P. A. R. Saraiva, E. F. de Souza, L. A. Ambrosio, and M. Becker, "The impact of feature scaling in machine learning: effects on regression and classification tasks." Disponível em: <https://arxiv.org/pdf/2506.08274>. Acesso em: 20 jun. 2025.

[13] E. B. Margherita Grandini and G. Visani, "Metrics for multi-class classification: an overview." Disponível em: <https://arxiv.org/pdf/2008.05756>. Acesso em: 18 jun. 2025.

[14] E. S. Gérard Biau, “A random forest guided tour.” Disponível em: <https://arxiv.org/pdf/1511.05741>. Acesso em: 30 mai. 2025.
[15] L. C. Qing Su and L. Qian, “Optimization of big data analysis resources supported by xgboost algorithm: comprehensive analysis of industry 5.0 and esg performance.” Disponível em: <https://www.sciencedirect.com/science/article/pii/S2665917424002861>. Acesso em: 28 jun. 2025.
[16] M. W. Philipp Probst and A.-L. Boulesteix, “Hyperparameters and tuning strategies for random forest.” Disponível em: <https://arxiv.org/pdf/1804.03515>. Acesso em: 28 jun. 2025.
[17] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system.” Disponível em: <https://arxiv.org/pdf/1603.02754>. Acesso em: 20 mai. 2025.
[18] P. Abeynayake and U. Wijenayake, “Detection of malicious urls using machine learning based on lexical features.” Disponível em: <https://journals.sjp.ac.lk/index.php/contre/article/view/7384/5265>. Acesso em: 28 jun. 2025.