

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS - CAMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARINA RIJO DE OLIVEIRA

**FERRAMENTA DE DETECÇÃO DE PHISHING COM
APRENDIZADO DE MÁQUINA**

BAURU

Novembro/2024

MARINA RIJO DE OLIVEIRA

**FERRAMENTA DE DETECÇÃO DE PHISHING COM
APRENDIZADO DE MÁQUINA**

Trabalho de Conclusão de Curso do Curso
de Ciência da Computação da Universidade
Estadual Paulista "Júlio de Mesquita Filho",
Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Kelton Augusto Pontara
da Costa

BAURU
Novembro/2024

O48f

Oliveira, Marina Rijo

Ferramenta de detecção de phishing com aprendizado de máquina /
Marina Rijo Oliveira. -- Bauru, 2024

32 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da
Computação) - Universidade Estadual Paulista (UNESP), Faculdade
de Ciências, Bauru

Orientador: Kelton Augusto Pontara da Costa

1. Inteligência Artificial. 2. Aprendizado de Máquinas. 3. Fraudes
na Internet. 4. Árvores de Decisão. 5. Software Desenvolvimento. I.
Título.

Marina Rijo de Oliveira

Ferramenta de detecção de phishing com aprendizado de máquina

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Kelton Augusto Pontara da Costa

Orientador

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Prof. Dra. Simone Prado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Prof. Douglas Rodrigues

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Bauru, _____ de _____ de _____.

Resumo

Com a popularização da internet, crimes virtuais como o *phishing* estão se tornando cada vez mais comuns. Em função disto, torna-se necessário o desenvolvimento de métodos de detecção simples para proteger a população. Este trabalho propõe a criação de uma aplicação simplificada para a detecção destes ataques, utilizando-se de modelos de aprendizado de máquina para validar os endereços *web* apresentados pelo usuário. Os modelos utilizados foram Árvore de Decisão, Floresta Aleatória, Árvores Extremamente Aleatórias, XGBoost, CatBoost e Regressão Logística, orquestrados de forma a poupar poder computacional e minimizar o tempo de execução.

Palavras-chave: Aprendizado de máquina; Crimes Virtuais; Inteligência Artificial; Árvore de Decisão; XGBoost.

Abstract

With the popularization of the internet, virtual crimes such as phishing are becoming increasingly common. Because of this, it is necessary to develop simple detection methods to protect the population. This work proposes the creation of a simplified application for detecting these attacks, using machine learning models to validate the web addresses presented by the user. The models used were Decision Tree, Random Forest, Extremely Randomized Trees, XGBoost, CatBoost, and Logistic Regression, orchestrated to save computational power and minimize execution time.

Keywords: Machine Learning; Cybercrime; Artificial Intelligence; Decision Tree; XGBoost.

Lista de figuras

Figura 1 – Árvore de decisão simplificada.	13
Figura 2 – Floresta aleatória com 4 sub-árvores.	14
Figura 3 – Estrutura de um modelo do tipo <i>Boosting</i>	15
Figura 4 – Matriz de Confusão	17
Figura 5 – Ordem de execução dos modelos	21
Figura 6 – Matriz de confusão da Árvore de Decisão	23
Figura 7 – Matriz de confusão do XGBoost	23
Figura 8 – Matriz de confusão da Regressão Logística	24
Figura 9 – Matriz de confusão da Floresta Aleatória	25
Figura 10 – Matriz de confusão do Extra Trees	25
Figura 11 – Matriz de confusão do CatBoost	26
Figura 12 – Tela inicial da aplicação	28
Figura 13 – Tela apresentada caso a página seja segura	28
Figura 14 – Tela apresentada caso a página seja duvidosa	29
Figura 15 – Tela apresentada caso a página seja perigosa	29

Lista de quadros

Quadro 1 – Exemplos de atributos baseados no endereço completo	20
Quadro 2 – Exemplos de atributos baseados no domínio do endereço	20
Quadro 3 – Classificação dos modelos	22
Quadro 4 – Resultado dos modelos	27

Lista de tabelas

Tabela 1 – Resultados da Árvore de Decisão	22
Tabela 2 – Resultados do XGBoost	23
Tabela 3 – Resultados da Regressão Logística	24
Tabela 4 – Resultados da Floresta Aleatória	24
Tabela 5 – Resultados do Extra Trees	25
Tabela 6 – Resultados do CatBoost	26

Sumário

1	INTRODUÇÃO	10
1.1	Objetivos	11
1.1.1	Objetivo Geral	11
1.1.2	Objetivos Específicos	11
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	Phishing	12
2.2	Aprendizado de Máquina	12
2.2.1	Árvore de Decisão	12
2.2.2	Floresta Aleatória	13
2.2.3	Árvores Extremamente Aleatórias	14
2.2.4	XGBoost	14
2.2.5	CatBoost	15
2.2.6	Regressão Logística	15
2.3	Métodos de Avaliação dos Modelos	16
2.3.1	Ácurácia	16
2.3.2	Precisão	16
2.3.3	Matriz de Confusão	17
3	METODOLOGIA	18
3.1	Ferramentas	18
3.1.1	Python	18
3.1.2	Google Colab	19
3.1.3	Streamlit	19
3.2	Conjunto de Dados	19
3.2.1	Características do Conjunto	20
3.3	Classificação dos modelos	20
3.4	Desenvolvimento da aplicação	21
4	RESULTADOS	22
4.1	Árvore de Decisão	22
4.2	XGBoost	23
4.3	Regressão Logística	24
4.4	Floresta Aleatória	24
4.5	Árvores Extremamente Aleatórias	25
4.6	CatBoost	26

4.7	Resultados Gerais	27
4.8	Aplicação	27
5	CONCLUSÃO	30
	REFERÊNCIAS	31

1 Introdução

O crescimento da internet nos dias atuais a tornou uma ferramenta presente no dia a dia de boa parte da população mundial, estando presente em muitos momentos da rotina de uma pessoa. Todavia, este crescimento também aumentou o número de crimes virtuais, sendo um dos grandes desafios em relação à segurança virtual.

Entre esses crimes está o *phishing*, sendo definido como o uso de engenharia social a fim de captar dados pessoais das pessoas afetadas. São utilizados sites e *e-mails* falsos, em conjunto com a replicação da identidade visual de muitas entidades confiáveis, como bancos e lojas virtuais, por exemplo, para enganar os afetados. Com a pandemia de COVID-19 em 2020, despertou um novo nível de ansiedade, medo e *stress* na sociedade, e, com isso, pessoas mal intencionadas buscam tirar proveito de pessoas que possuem tais problemas, facilitando os ataques de *phishing* (MONTAGNER; WESTPHALL, 2022).

Em função desse aumento nos casos, detectar sites falsos se torna algo interessante do ponto de vista da segurança da informação. Se for possível detectar se um *link* leva à um site de phishing, pode-se utilizar dessa detecção para prevenir os ataques e proteger os dados de muitas pessoas.

Pesquisadores como Coutinho (2023), Resnick e Bastos-Filho (2024) e Souza e Mascarenhas (2023) abordaram o tema de diferentes maneiras, explorando distintos modelos de aprendizado de máquina no processo de detecção dos ataques. O objetivo deste projeto é aplicar em um produto as ideias e modelos apresentados pelos artigos citados.

1.1 Objetivos

Serão listados a seguir o objetivo geral e os objetivos específicos deste projeto.

1.1.1 Objetivo Geral

Criar uma ferramenta que consiga identificar endereços web utilizados para *phishing*, usando vários modelos de aprendizado de máquina.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Encontrar e tratar um *dataset* que será usado como base de treinamento dos modelos.
- Identificar e classificar 6 modelos de aprendizado de máquina conforme sua taxa de sucesso.
- Montar um filtro por meio do ordenamento dos modelos a fim de poupar poder computacional quando possível.
- Desenvolver uma interface gráfica que acionará o filtro, a fim de identificar se o endereço web apresentado é *phishing* ou não.

2 Fundamentação Teórica

2.1 Phishing

[Aleroud e Zhou \(2017\)](#) definem como *phishing* o uso de técnicas de engenharia social em conjunto com a clonagem de páginas conhecidas e confiáveis para captar dados sensíveis das pessoas afetadas.

Este ataque se aproveita da falta de conhecimento sobre computadores e segurança da informação para atrair vítimas e roubar informações sensíveis, como login, senha, dados bancários, entre outros.

2.2 Aprendizado de Máquina

Aprendizado de Máquina é uma área das Ciências da Computação que engloba o desenvolvimento de ambientes e sistemas que consigam aprender, com o mínimo de intervenção humana, a identificar padrões e tomar decisões.

Os modelos de aprendizado de máquina podem ser classificados entre três tipos principais: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Para este trabalho, será dado foco ao aprendizado supervisionado.

Aprendizado supervisionado é definido como um algoritmo que recebe dados já categorizados, buscando treinar o modelo para classificar novos dados por meio do reconhecimento de padrões.

2.2.1 Árvore de Decisão

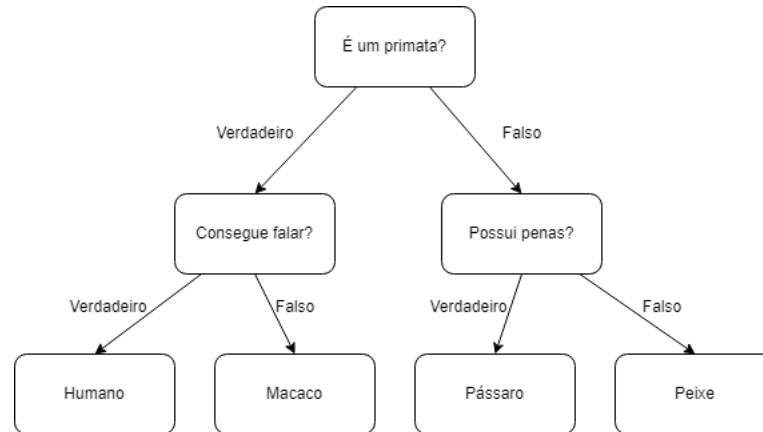
Árvore de Decisão é um modelo que se baseia numa série de questionamentos e validações, realizados de forma hierárquica e com o intuito de chegar em uma conclusão ou classificação.

Este algoritmo é muito popular devido ao fato de, segundo [Müller e Guido \(2016\)](#), ser simples de entender e interpretar, além de necessitar de pouco preparo em relação aos dados de entrada. Porém, as árvores de decisão são propensas a sofrer sobreajuste, ou seja, podem não conseguir realizar previsões precisas com dados que não estão presentes no *dataset* de treinamento.

A Figura 1 mostra uma árvore de decisão simplificada, com ela é possível classificar um animal entre humano, macaco, pássaro e peixe. Cada nó interno representa uma *feature* do

dataset, cada nó folha representa uma das classificações possíveis e cada galho representa uma regra de decisão.

Figura 1 – Árvore de decisão simplificada.



Fonte: Adaptado de Müller e Guido (2016).

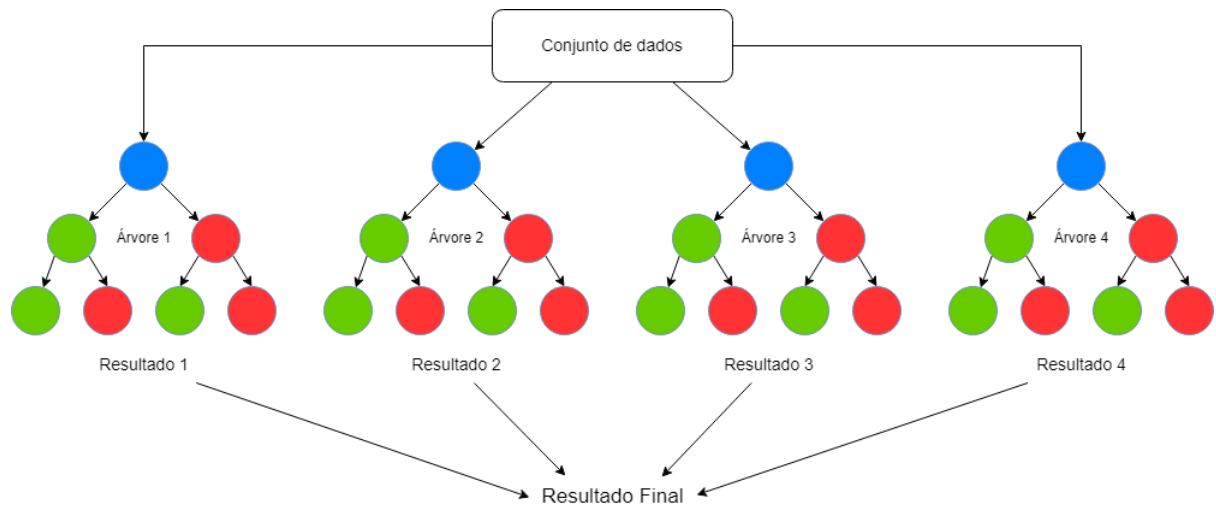
2.2.2 Floresta Aleatória

A Floresta Aleatória é um modelo do tipo *ensemble*, sendo definido algoritmos *ensemble* como a combinação de diversos modelos de aprendizado de máquina com o objetivo de criar modelos mais poderosos.

Raschka e Mirjalili (2017) definem uma Floresta Aleatória em 4 etapas, exemplificadas na Figura 2:

1. Selecionar de forma aleatória uma amostra de tamanho n do *dataset*, permitindo que o mesmo dado seja selecionado diversas vezes;
2. Montar uma Árvore de Decisão a partir desta amostra, onde cada nó deve:
 - Selecionar de forma aleatória d características do conjunto de dados;
 - Separar o nó utilizando a característica que realiza a melhor divisão segundo alguma função objetiva.
3. Repetir as etapas 1 e 2 k vezes;
4. O resultado final é obtido por meio da média dos resultados de todas as árvores ou pela previsão mais frequente.

Figura 2 – Floresta aleatória com 4 sub-árvores.



Fonte: Adaptado de [Coutinho \(2023\)](#).

2.2.3 Árvores Extremamente Aleatórias

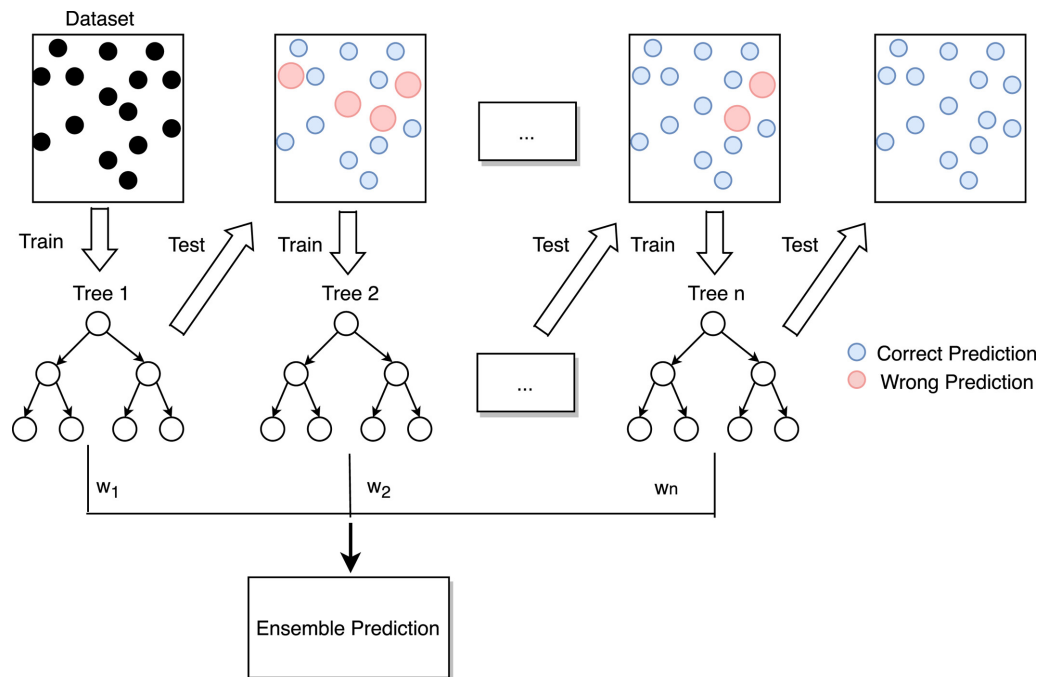
Árvores Extremamente Aleatórias ou *Extra Trees* pode ser visto como uma variação da Floresta Aleatória proposta por [Geurts, Ernst e Wehenkel \(2006\)](#). O funcionamento é essencialmente o mesmo, exceto que o *Extra Trees* separa os nós de forma aleatória e utiliza o conjunto de dados inteiro na criação das sub-árvores.

Essa aleatoriedade provém uma menor variância ao modelo, assim como um maior viés quando comparado com o Floresta Aleatória. Segundo [Geurts, Ernst e Wehenkel \(2006\)](#), a aleatoriedade do modelo pode ser ajustada, fazendo com que a variância se aproxime de zero e o viés aumente levemente quando comparada com outros tipos de árvore.

2.2.4 XGBoost

O XGBoost ou *Extreme Gradient Boosting* é um modelo de *gradient boosting* desenvolvido por [Chen e Guestrin \(2016\)](#). Uma técnica *ensemble*, *Boosting* utiliza a combinação de vários modelos de forma sequencial. Cada modelo adicional é treinado para corrigir os erros dos modelos anteriores, como exemplificado na Figura 3. O *Gradient Boosting* encaixa-se como um modelo do tipo *boosting*, criando novos modelos para calcular os resíduos das iterações anteriores, produzindo ao final uma combinação ponderada que melhora a acurácia do modelo.

Figura 3 – Estrutura de um modelo do tipo *Boosting*.



Fonte: [Zhang et al. \(2021\)](#).

O XGBoost procura otimizar o *gradient boosting* utilizando técnicas de paralelização e regularização no processo de treinamento para obter uma maior eficiência computacional.

2.2.5 CatBoost

CatBoost é uma implementação do *gradient boosting*, que faz uso de árvores de decisão binárias como base de predição (IBRAHIM et al., 2020). Ele é otimizado para trabalhar com variáveis categóricas e, por ser capaz de processar dados categóricos diretamente, não existe a necessidade de transformá-los em dados numéricos, o que diminui a complexidade do modelo.

Prokhorenkova et al. (2018) realizou testes comparando o CatBoost à outros modelos de *gradient boosting* como o XGBoost e concluiu que o ele supera a performance dos principais algoritmos de *boosting* por gradiente do mercado.

2.2.6 Regressão Logística

A Regressão Logística é um modelo estatístico linear comumente aplicado à problemas de classificação binária. Ele utiliza-se de uma função logística para determinar a probabilidade de uma entrada pertencer a uma classe determinada, restringindo a saída entre 0 e 1.

Ibrahim et al. (2020) define 5 pontos essenciais para a utilização coesa do modelo de Regressão Logística:

1. O resultado deve ser binário;

2. A relação entre a característica de classificação e as demais características não define uma relação linear;
3. Uma amostra grande normalmente é necessária;
4. Deve haver pouca ou nenhuma multicolinearidade;
5. As categorias devem ser mutuamente exclusivas e exaustivas.

Devido à sua simplicidade e facilidade de interpretação, esse modelo é amplamente usado em contextos de classificação.

2.3 Métodos de Avaliação dos Modelos

O método utilizado para avaliar o desempenho dos modelos foi a classificação binária. [Fawcett \(2006\)](#) afirma que um classificador binário possui quatro resultados possíveis para cada entrada no modelo:

- Verdadeiro Positivo (TP): Quando a entrada é verdadeira e o modelo identifica como verdadeira;
- Verdadeiro Negativo (TN): Quando a entrada é falsa e o modelo identifica como falsa;
- Falso Positivo (FP): Quando a entrada é falsa e o modelo identifica como verdadeira;
- Falso Negativo (FN): Quando a entrada é verdadeira e o modelo identifica como falsa;

2.3.1 Ácurácia

A acurácia define a porcentagem de resultados que foram classificados corretamente. Esta métrica é calculada seguindo a Equação 2.1:

$$\text{Acurácia} = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.1)$$

2.3.2 Precisão

A precisão define a porcentagem de resultados do tipo verdadeiro positivo. Esta métrica é calculada seguindo a Equação 2.2:

$$\text{Precisão} = \frac{TP}{TP + FP} \quad (2.2)$$

2.3.3 Matriz de Confusão

A matriz de confusão de um modelo é uma matriz que representa os resultados da classificação binária de um modelo. A Figura 4 apresenta um exemplo de uma matriz de confusão.

Figura 4 – Matriz de Confusão

Classe Predita	Classe Verdadeira	
	Positivo	Negativo
	Positivo	Negativo
Positivo	TP	FP
Negativo	FN	TN

Fonte: Elaborada pela autora.

3 Metodologia

Este trabalho foi realizado em três etapas. A primeira foi a revisão bibliográfica, com o objetivo de encontrar algoritmos de aprendizado de máquina e um *dataset* compatíveis com a ideia de classificar endereços *web* em relação à probabilidade de serem *phishing* ou não.

A segunda parte foi o treinamento e classificação dos modelos escolhidos. Foram selecionados seis modelos, e a classificação foi feita em relação à acurácia, precisão e tempo médio de execução de cada um. O conjunto de dados utilizado teve de ser alterado para se adequar à proposta deste projeto.

Por fim, foi desenvolvida uma aplicação que utiliza de algoritmos e lógica para classificar *links* apresentados via interface de usuário como seguros, duvidosos e perigosos.

A seguir estão as ferramentas e métodos utilizados no desenvolvimento deste trabalho.

3.1 Ferramentas

Nesta seção serão apresentadas as ferramentas utilizadas na criação deste trabalho.

3.1.1 Python

Python é uma linguagem de programação interpretada, e uma das mais utilizadas na área de ciência de dados.

A escolha desta linguagem foi motivada principalmente pela simplicidade do desenvolvimento, pelas diversas bibliotecas voltadas para o aprendizado de máquina e por ser compatível com o *framework* Streamlit.

as principais bibliotecas utilizadas são:

- **Scikit-Learn:** Desenvolvida por [Kramer e Kramer \(2016\)](#), Scikit-Learn é uma biblioteca Python *open-source* utilizada para o aprendizado de máquina. Ela fornece vários métodos de classificação, regressão, pré-processamento de dados e métricas de desempenho.
- **Pandas:** Pandas é focada em análise e manipulação de dados. É muito utilizada na área de ciência de dados por, segundo [McKinney et al. \(2011\)](#), ser código aberto, flexível, fácil de usar, muito eficiente.
- **XGBoost:** XGBoost faz a implementação do classificador XGBoost, utilizado neste projeto.

- **CatBoost:** CatBoost faz a implementação do classificador CatBoost, também utilizado neste projeto.
- **Matplotlib:** Matplotlib é utilizada para visualização de dados, e seu uso teve foco no processo de teste e classificação dos modelos.

3.1.2 Google Colab

O Google Colab é um ambiente de programação Python hospedado pela Google, possibilitando o desenvolvimento de *scripts* em qualquer local e em qualquer computador, sendo necessária apenas a conexão à internet.

Esta ferramenta foi utilizada para o treinamento e a avaliação dos modelos utilizados no projeto.

3.1.3 Streamlit

Streamlit é um *framework* em Python que possibilita o desenvolvimento de páginas *web* de forma simples e prática. Streamlit transforma *data scripts* em páginas *web* compartilháveis em minutos, tudo usando apenas Python e sem a necessidade de experiência com *front-end* (STREAMLIT, 2024).

Ele permite a criação de um cliente *web* diretamente conectado a um servidor Python, possibilitando atualizações e alterações em tempo real entre o *backend* e o *frontend*. Além disso, o *framework* também facilita no desenvolvimento da interface de usuário disponibilizando componentes e retirando a necessidade do uso de HTML e CSS.

3.2 Conjunto de Dados

Inicialmente foram selecionados dois conjuntos de dados, um estruturado por Vrbaničič, Fister e Podgorelec (2020) e o outro estruturado por Prasad e Chandra (2024).

Durante os testes dos modelos, devido à inconsistências e dificuldades do uso do *dataset* de Prasad e Chandra (2024), o conjunto estruturado por Vrbaničič, Fister e Podgorelec (2020) foi o escolhido para ser utilizado no sistema final. Ele possui 88.647 entradas de dados, divididas da seguinte forma:

- 58.000 entradas rotuladas como legítimas
- 30.647 entradas rotuladas como *phishing*

3.2.1 Características do Conjunto

O conjunto original possui 112 características, e uma delas é a classificação em relação à legitimidade da entrada.

Foram selecionadas 41 delas para o treinamento dos modelos, utilizando o critério de serem características obtíveis a partir de um endereço *web*. Elas podem ser divididas entre atributos baseados no endereço completo e no domínio do endereço.

- Atributos baseados no endereço completo: são características obtidas a partir do endereço completo da página. Algumas delas apresentam-se no Quadro 1.

Quadro 1 – Exemplos de atributos baseados no endereço completo

Atributo	Descrição	Formato
qty_questionmark_url	Quantidade de '?' no endereço	Inteiro
qty_tilde_url	Quantidade de '~' no endereço	Inteiro
qty_tld_url	Número de caracteres no domínio de nível superior	Inteiro
length_url	Número de caracteres no endereço	Inteiro

Fonte: Elaborado pela autora.

- Atributos baseados no domínio do endereço: são características obtidas somente a partir do domínio do endereço. Algumas delas apresentam-se no Quadro 2.

Quadro 2 – Exemplos de atributos baseados no domínio do endereço

Atributo	Descrição	Formato
qty_and_domain	Quantidade de '&' no endereço	Inteiro
qty_plus_domain	Quantidade de '+' no endereço	Inteiro
domain_length	Número de caracteres no domínio	Inteiro
domain_in_ip	Se o domínio está no formato de endereço de IP	Booleano

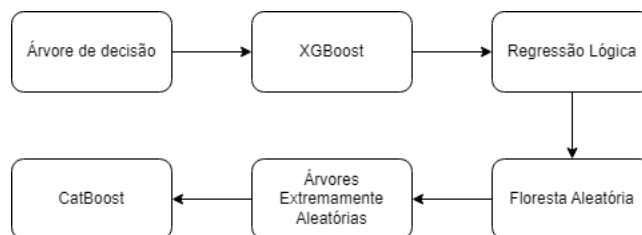
Fonte: Elaborado pela autora.

3.3 Classificação dos modelos

Após selecionar o *dataset*, foi realizado o treinamento e o teste dos modelos. O conjunto de dados foi dividido em 70% das entradas para treinamento e 30% para testes.

A classificação dos modelos foi feita com base na acurácia, precisão e tempo de execução médio, sendo este a média de 10 tempos de execução distintos. A ordem obtida está apresentada na Figura 5:

Figura 5 – Ordem de execução dos modelos



Fonte: Elaborada pela autora.

3.4 Desenvolvimento da aplicação

O desenvolvimento da aplicação foi realizado utilizando o *framework* Streamlit como base, aplicando componentes e conceitos disponibilizados para facilitar a codificação do *backend* e do *frontend*.

O sistema recebe um endereço a ser validado, obtém todas as características necessárias a partir dele e executa os modelos de aprendizado de máquina, retornando o resultado por meio de mudanças na interface gráfica da aplicação *web*.

4 Resultados

Os resultados obtidos serão apresentados conforme foram abordados na fundamentação teórica, sendo a acurácia e a precisão os melhores resultados fornecidos pelos modelos, e o tempo de execução a média de 10 tempos de execução distintos. Os valores foram obtidos a partir dos testes realizados durante o projeto.

Nos algoritmos utilizados, a classificação segue o Quadro 3:

Quadro 3 – Classificação dos modelos

Classificação	Tipo de página
Negativa	Legítima
Positiva	<i>Phishing</i>

Fonte: Elaborado pela autora.

4.1 Árvore de Decisão

A Tabela 1 apresenta os resultados obtidos durante os testes do modelo Árvore de Decisão.

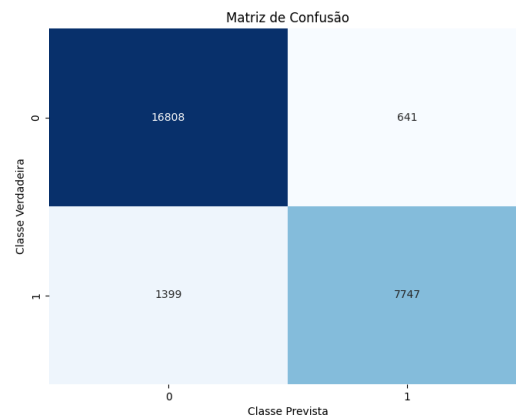
Tabela 1 – Resultados da Árvore de Decisão

	Acurácia	Precisão	Tempo de execução
Árvore de Decisão	92,42%	92,53%	1,1 segundos

Fonte: Elaborada pela autora.

O modelo apresentou resultados bons nos testes executados, com 1399 falsos positivos e 641 falsos negativos. A Figura 6 mostra a matriz de confusão referente ao modelo.

Figura 6 – Matriz de confusão da Árvore de Decisão



Fonte: Elaborada pela autora.

4.2 XGBoost

A Tabela 2 apresenta os resultados obtidos durante os testes do modelo XGBoost.

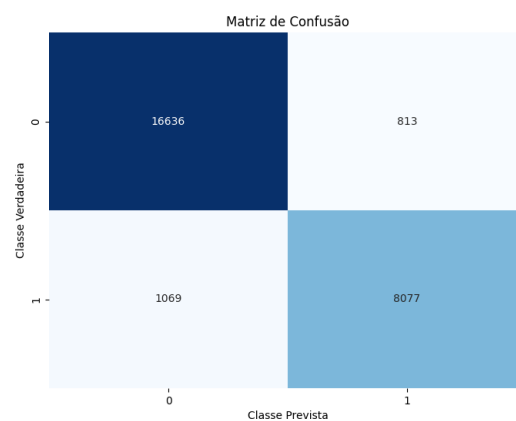
Tabela 2 – Resultados do XGBoost

	Acurácia	Precisão	Tempo de execução
XGBoost	92,92%	90,85%	1,63 segundos

Fonte: Elaborada pela autora.

O modelo apresentou resultados bons nos testes executados, com 1069 falsos positivos e 813 falsos negativos. A Figura 7 mostra a matriz de confusão referente ao modelo.

Figura 7 – Matriz de confusão do XGBoost



Fonte: Elaborada pela autora.

4.3 Regressão Logística

A Tabela 3 apresenta os resultados obtidos durante os testes do modelo Regressão Logística.

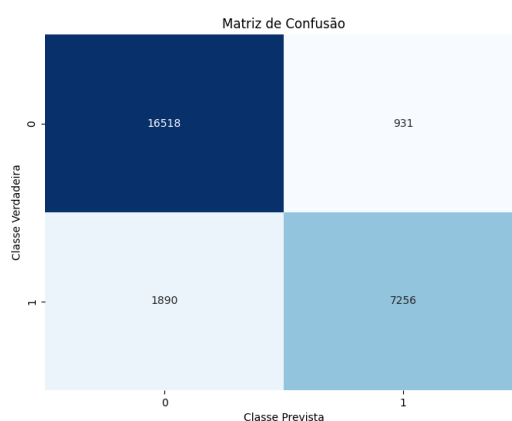
Tabela 3 – Resultados da Regressão Logística

	Acurácia	Precisão	Tempo de execução
Regressão Logística	89,39%	88,63%	5,91 segundos

Fonte: Elaborada pela autora.

O modelo apresentou resultados bons nos testes executados, com 1890 falsos positivos e 931 falsos negativos. A Figura 8 mostra a matriz de confusão referente ao modelo.

Figura 8 – Matriz de confusão da Regressão Logística



Fonte: Elaborada pela autora.

4.4 Floresta Aleatória

A Tabela 4 apresenta os resultados obtidos durante os testes do modelo Floresta Aleatória.

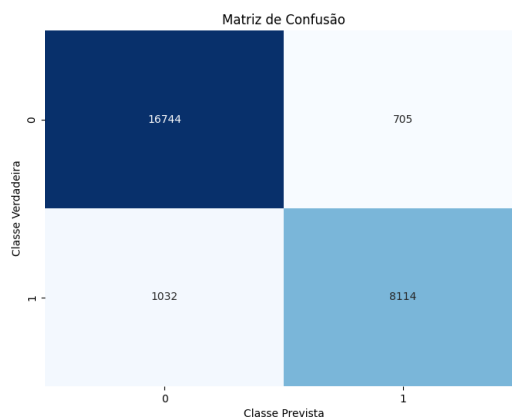
Tabela 4 – Resultados da Floresta Aleatória

	Acurácia	Precisão	Tempo de execução
Floresta Aleatória	93,45%	91,99%	6,77 segundos

Fonte: Elaborada pela autora.

O modelo apresentou resultados bons nos testes executados, com 1032 falsos positivos e 705 falsos negativos. A Figura 9 mostra a matriz de confusão referente ao modelo.

Figura 9 – Matriz de confusão da Floresta Aleatória



Fonte: Elaborada pela autora.

4.5 Árvores Extremamente Aleatórias

A Tabela 5 apresenta os resultados obtidos durante os testes do modelo Extra Trees.

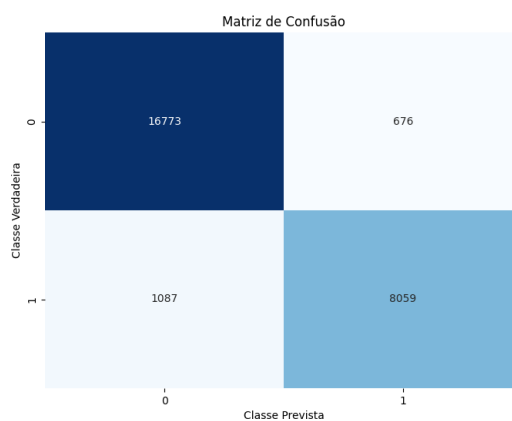
Tabela 5 – Resultados do Extra Trees

	Acurácia	Precisão	Tempo de execução
Extra Trees	93,41%	93,87%	8,35 segundos

Fonte: Elaborada pela autora.

O modelo apresentou resultados bons nos testes executados, com 1087 falsos positivos e 676 falsos negativos. A Figura 10 mostra a matriz de confusão referente ao modelo.

Figura 10 – Matriz de confusão do Extra Trees



Fonte: Elaborada pela autora.

4.6 CatBoost

A Tabela 6 apresenta os resultados obtidos durante os testes do modelo CatBoost.

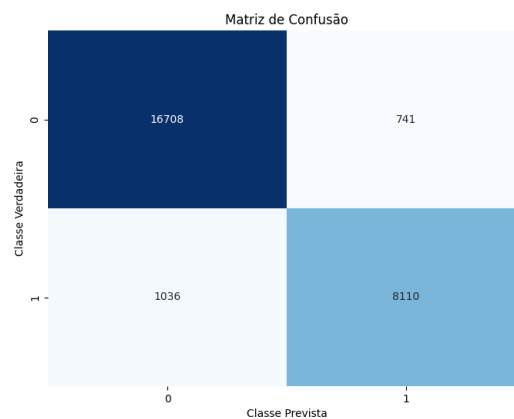
Tabela 6 – Resultados do CatBoost

	Acurácia	Precisão	Tempo de execução
CatBoost	93,32%	91,63%	22,7 segundos

Fonte: Elaborada pela autora.

O modelo apresentou resultados bons nos testes executados, com 1036 falsos positivos e 741 falsos negativos. A Figura 11 mostra a matriz de confusão referente ao modelo.

Figura 11 – Matriz de confusão do CatBoost



Fonte: Elaborada pela autora.

4.7 Resultados Gerais

O Quadro 4 apresenta os resultados obtidos durante o desenvolvimento deste projeto. Os modelos atingiram resultados bons em relação ao tempo de execução de cada um, sendo este o fator principal na ordenação deles.

Quadro 4 – Resultado dos modelos

	Acurácia	Precisão	Tempo de Execução
Árvore de Decisão	92,42%	92,53%	1,1 segundos
XGBoost	92,92%	90,85%	1,63 segundos
Regressão Logística	89,39%	88,63%	5,91 segundos
Floresta Aleatória	93,45%	91,99%	6,77 segundos
Extra Trees	93,41%	93,87%	8,35 segundos
CatBoost	93,32%	91,63%	22,7 segundos

Fonte: Elaborado pela autora.

Porém, quando trabalhando com entradas novas, os modelos retornaram resultados divergentes dos obtidos durante os testes realizados utilizando o *dataset* como base. A principal causa é o fato do conjunto de dados sido modificado, retirando 71 *features* que não se encaixavam com a proposta do projeto.

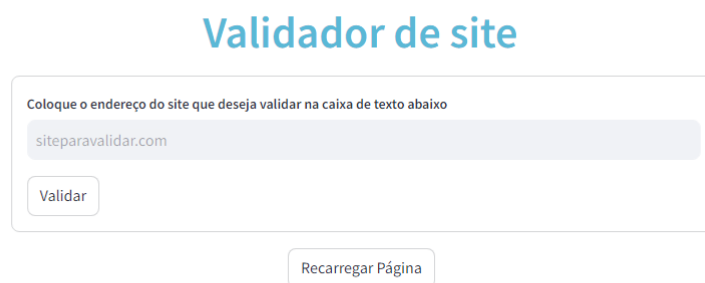
4.8 Aplicação

O código fonte da aplicação desenvolvida está disponível no repositório <<https://github.com/Maary11/Orchestrator>>.

A seguir estão apresentadas as telas desenvolvidas para cada tipo de classificação de *link*.

A Figura 12 apresenta a tela inicial da aplicação. Nessa tela o usuário insere o endereço *web* que deseja testar e pode tanto clicar no botão *Validar* ou apertar a tecla *Enter* do teclado para iniciar a validação.

Figura 12 – Tela inicial da aplicação



A tela inicial da aplicação, intitulada "Validador de site" em azul. Abaixo do título, há uma caixa de texto com o placeholder "Coloque o endereço do site que deseja validar na caixa de texto abaixo". O campo de texto contém o exemplo "siteparavalidar.com". Abaixo do campo, há um botão "Validar". Na parte inferior da tela, há um botão "Recarregar Página".

Fonte: Elaborada pela autora.

A Figura 13 apresenta a tela de página segura. Ela é apresentada ao usuário caso o retorno da aplicação indique que é uma página segura e a confiança do modelo esteja entre 90% e 98,5%.

Figura 13 – Tela apresentada caso a página seja segura



A tela de página segura, intitulada "Validador de site" em verde. Abaixo do título, há uma barra verde com o texto "Site Seguro". Abaixo da barra, há o texto "O site foi validado e pode ser acessado sem riscos à segurança". Na parte inferior da tela, há um botão "Recarregar Página".

Fonte: Elaborada pela autora.

A Figura 14 apresenta a tela de página duvidosa. Ela é apresentada ao usuário caso o retorno da aplicação esteja com a confiança abaixo de 90% ou acima de 98,5%, independente da classificação obtida.

Figura 14 – Tela apresentada caso a página seja duvidosa



Fonte: Elaborada pela autora.

A Figura 15 apresenta a tela de página perigosa. Ela é apresentada ao usuário caso o retorno da aplicação indique que é uma página perigosa e a confiança do modelo esteja entre 90% e 98,5%.

Figura 15 – Tela apresentada caso a página seja perigosa



Fonte: Elaborada pela autora.

5 Conclusão

A tecnologia se tornou parte intrínseca da vida das pessoas nos últimos anos, e as questões ligadas à cibersegurança tornaram-se extremamente pertinentes. Combater e prevenir crimes cibernéticos tem sido um desafio cada vez maior, e tais crimes afetam milhares de pessoas diariamente.

Este trabalho buscou auxiliar na prevenção de ataques de *phishing* por meio da criação de um sistema que consiga detectar endereços maliciosos de forma simples para o usuário final.

Os objetivos traçados foram alcançados. Porém, ficam evidentes vários pontos de melhoria para trabalhos futuros. A precisão dos modelos, quando acoplados ao sistema, mostrou-se inferior ao esperado durante os testes. A principal causa foi o uso de um *dataset* que não se encaixou tão bem à proposta do projeto, sendo necessárias alterações e remoções de características durante o desenvolvimento.

Futuros trabalhos podem seguir pela abordagem de aplicar outros algoritmos de aprendizado de máquina e detecção de *phishing*. Também é possível fazer a estruturação de um conjunto de dados que tenha maior sintonia com o funcionamento do sistema, aumentando a precisão e performance dos modelos.

Referências

- ALEROUD, A.; ZHOU, L. Phishing environments, techniques, and countermeasures: A survey. *Computers & Security*, Elsevier, v. 68, p. 160–196, 2017.
- 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.: s.n.], 2016. p. 785–794.
- COUTINHO, V. M. Detecção de páginas de phishing utilizando aprendizado de máquina. 2023.
- FAWCETT, T. An introduction to roc analysis. *Pattern recognition letters*, Elsevier, v. 27, n. 8, p. 861–874, 2006.
- GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine learning*, Springer, v. 63, p. 3–42, 2006.
- IBRAHIM, A. A.; RIDWAN, R. L.; MUHAMMED, M. M.; ABDULAZIZ, R. O.; SAHEED, G. A. Comparison of the catboost classifier with other machine learning methods. *International Journal of Advanced Computer Science and Applications*, v. 11, 2020. Disponível em: <<https://api.semanticscholar.org/CorpusID:232846952>>.
- KRAMER, O.; KRAMER, O. Scikit-learn. *Machine learning for evolution strategies*, Springer, p. 45–53, 2016.
- MCKINNEY, W. et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, Seattle, v. 14, n. 9, p. 1–9, 2011.
- MONTAGNER, A. S.; WESTPHALL, C. M. Uma breve análise sobre phishing. *Revista ComInG-Communications and Innovations Gazette*, v. 6, n. 1, p. 46–56, 2022.
- MÜLLER, A. C.; GUIDO, S. *Introduction to machine learning with Python: a guide for data scientists*. [S.l.]: "O'Reilly Media, Inc.", 2016.
- PRASAD, A.; CHANDRA, S. *PhiUSIIL Phishing URL (Website)*. 2024. UCI Machine Learning Repository. DOI: <https://doi.org/10.1016/j.cose.2023.103545>.
- PROKHORENKOVA, L.; GUSEV, G.; VOROBIEV, A.; DOROGUSH, A. V.; GULIN, A. Catboost: unbiased boosting with categorical features. In: BENGIO, S.; WALLACH, H.; LAROCHELLE, H.; GRAUMAN, K.; CESA-BIANCHI, N.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. v. 31. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf>.
- RASCHKA, S.; MIRJALILI, V. Python machine learning second edition. *Birmingham, England: Packt Publishing*, 2017.
- RESNICK, N. E.; BASTOS-FILHO, C. J. A. Aplicação de aprendizado de máquinas para detecção de urls phishing. *Revista de Engenharia e Pesquisa Aplicada*, v. 9, n. 1, p. 41–49, 2024.

SOUZA, J. A.; MASCARENHAS, D. M. Detecção de ataques de phishing em tempo real utilizando algoritmos de aprendizado de máquina. In: SBC. *Anais Estendidos do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2023. p. 165–176.

STREAMLIT. *Streamlit documentation*. 2024. Disponível em: <<https://docs.streamlit.io/>>. Acesso em: 3 abr. 2024.

VRBANČIČ, G.; FISTER, I.; PODGORELEC, V. Datasets for phishing websites detection. *Data in Brief*, v. 33, p. 106438, 2020. ISSN 2352-3409. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2352340920313202>>.

ZHANG, T.; LIN, W.; VOGELMANN, A. M.; ZHANG, M.; XIE, S.; QIN, Y.; GOLAZ, J.-C. Improving convection trigger functions in deep convective parameterization schemes using machine learning. *Journal of Advances in Modeling Earth Systems*, Wiley Online Library, v. 13, n. 5, p. e2020MS002365, 2021.