

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA  
FILHO”  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE COMPUTAÇÃO  
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

## Aprendizado de máquina para detecção de ransomware

Ian Marques Breda

20 de novembro de 2024

Ian Marques Breda

# Aprendizado de máquina para detecção de ransomware

Proposta para Trabalho de Conclusão de  
Curso do Curso de Bacharelado em Ciên-  
cia da Computação da Universidade Esta-  
dual Paulista “Júlio de Mesquita Filho”, Fa-  
culdade de Ciências, campus Bauru.

Orientador: Kelton A. P. Costa

Co-Orientador: Douglas Rodrigues

Bauru

Abril/2024

# Resumo

Este trabalho apresenta o desenvolvimento de um *protótipo* de *ransomware*, seguido pela criação e aplicação de um *modelo* de *aprendizado de máquina* baseado em *floresta aleatória* para a detecção de *ransomwares*. O *protótipo* visa simular o comportamento malicioso do *ransomware*, enquanto o *modelo* de detecção é treinado para identificar padrões associados a atividades de *ransomware*, permitindo uma abordagem preventiva eficaz. Os resultados obtidos demonstram a capacidade do *modelo* de *floresta aleatória* em detectar ataques com alta acurácia, contribuindo para o fortalecimento das *defesas cibernéticas*.

**Palavras-chave:** Ransomware; Aprendizado de máquina; Floresta aleatória; Protótipo; Modelo; Defesas cibernéticas.

# Abstract

This work presents the development of a *prototype* of *ransomware*, followed by the creation and application of a *machine learning model* based on *random forest* for *ransomware* detection. The *prototype* aims to simulate the malicious behavior of *ransomware*, while the detection *model* is trained to identify patterns associated with *ransomware* activities, enabling an effective preventive approach. The results demonstrate the ability of the *random forest model* to detect attacks with high accuracy, contributing to the strengthening of *cyber defenses*.

**Keywords:** Ransomware; Machine learning; Random forest; Prototype; Model; Cyber defenses.

## Lista de Figuras

1	Tela de pagamento do WannaCry . . . . .	7
2	Simplificação da floresta aleatória . . . . .	9
3	Tabela sobre o tempo de quebra da criptografia RSA . . . . .	12
4	Fluxograma de funcionamento do protótipo . . . . .	15
5	Chaves de criptografia . . . . .	16
6	Fluxograma do modelo de detecção . . . . .	17
7	Arquivo .png antes da criptografia . . . . .	19
8	Arquivo .png após criptografia . . . . .	19
9	Arquivo .pdf antes da criptografia . . . . .	20
10	Arquivo .pdf após criptografia . . . . .	20
11	Arquivo .txt antes da criptografia . . . . .	21
12	Arquivo .txt após criptografia . . . . .	21
13	Nota de resgate . . . . .	22
14	Matriz de confusão . . . . .	24
15	Importância por característica (em porcentagem) . . . . .	25
16	Análise da característica 'Machine' . . . . .	26
17	Métricas do modelo . . . . .	27
18	Resultados após a nova escolha das características . . . . .	27
19	Resultados gerados pelo modelo . . . . .	28
20	Ilustração do SHAP Beeswam (Ilustrativo) . . . . .	29
21	Ilustração do SHAP Beeswam (Real) . . . . .	30

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>5</b>
1.1	Problema . . . . .	5
1.2	Justificativa . . . . .	6
1.3	Objetivos . . . . .	6
1.3.1	Objetivo Geral . . . . .	6
1.3.2	Objetivos Específicos . . . . .	6
<b>2</b>	<b>Fundamentação teórica</b>	<b>7</b>
2.1	<i>ransomware</i> . . . . .	7
2.1.1	Tipos principais . . . . .	7
2.1.2	Funcionamento do <i>ransomware</i> . . . . .	8
2.2	Aprendizado de Máquina . . . . .	8
2.2.1	Tipos principais . . . . .	8
2.2.2	Aplicações do Aprendizado de Máquina . . . . .	8
2.3	Floresta Aleatória . . . . .	8
2.3.1	Funcionamento da Floresta Aleatória . . . . .	9
2.3.2	Componentes Principais . . . . .	9
2.3.3	Vantagens e limitações . . . . .	10
2.4	Criptografia . . . . .	10
2.4.1	Criptografia AES . . . . .	11
2.4.2	Criptografia RSA . . . . .	12
2.5	Trabalhos correlatos . . . . .	13
<b>3</b>	<b>Metodologia</b>	<b>14</b>
3.1	Materiais e ambiente . . . . .	14
3.1.1	Descrição das bibliotecas Python . . . . .	14
3.2	Protótipo . . . . .	15
3.3	Modelo . . . . .	17
<b>4</b>	<b>Resultados do <i>ransomware</i></b>	<b>19</b>
4.1	Arquivos criptografados . . . . .	19
4.2	Arquivos gerados pelo <i>ransomware</i> . . . . .	22
<b>5</b>	<b>Resultados do modelo</b>	<b>23</b>
5.1	Métricas utilizadas . . . . .	23
5.1.1	Acurácia . . . . .	23
5.1.2	Validação Cruzada . . . . .	23
5.1.3	Precisão . . . . .	23
5.1.4	<i>Recall</i> . . . . .	24
5.1.5	F1-Score . . . . .	24
5.1.6	Matriz de Confusão . . . . .	24
5.2	Resultados obtidos . . . . .	25
5.3	Testando novos dados . . . . .	28
<b>6</b>	<b>Conclusão</b>	<b>31</b>
	<b>Referências</b>	<b>32</b>

# 1 Introdução

A crescente dependência de sistemas computacionais em praticamente todos os setores da sociedade moderna trouxe consigo uma série de desafios relacionados à segurança cibernética. De pequenos negócios a grandes corporações, a infraestrutura tecnológica atual permite uma comunicação rápida, armazenamento de dados massivo e execução de operações complexas de forma quase instantânea. No entanto, essa evolução tecnológica também abriu portas para uma gama cada vez maior de ameaças digitais, das quais os *malwares*, e em especial os *ransomwares*, se destacam pela sua gravidade e alcance.

O termo "*malware*" refere-se a qualquer *software* malicioso criado com o objetivo de causar danos a sistemas computacionais, usuários ou dados. Dentro desta categoria, o *ransomware* se distingue por sua capacidade de extorquir vítimas através da encriptação de dados essenciais, bloqueando o acesso a sistemas inteiros até que um resgate seja pago. Esse resgate, geralmente exigido em criptomoedas como o Bitcoin, torna a transação praticamente impossível de rastrear, favorecendo os criminosos por oferecer-lhes anonimato e impunidade.

O primeiro registro de ataques de *ransomware* data de 2005, na Rússia, mas sua disseminação global e sofisticação técnica evoluíram exponencialmente nas décadas seguintes. Exemplos notórios incluem o ataque "WannaCry" de 2017, que afetou hospitais, instituições governamentais e empresas em vários países, causando perdas financeiras incalculáveis e expondo a vulnerabilidade dos sistemas em ambientes críticos. Desde então, variantes como o Locky, CryptoShield e Ryuk continuaram a impactar organizações, inclusive governos e hospitais, destacando o potencial devastador desse tipo de ameaça.

Atualmente, o *ransomware* representa uma indústria criminosa de baixo risco e alta recompensa, que continua a prosperar com o avanço de novas técnicas de ataque. Ao atacar alvos sensíveis, como hospitais e órgãos governamentais, os atacantes garantem que as vítimas muitas vezes não tenham escolha a não ser pagar o resgate, devido às implicações críticas que a perda de dados pode trazer. Além disso, a rápida evolução das técnicas de ataque tornou ineficazes muitas das medidas de segurança convencionais, forçando as equipes de segurança cibernética a buscarem novas abordagens.

## 1.1 Problema

O propósito deste trabalho é abordar o seguinte problema de pesquisa: Como os *ransomwares* se comportam e como o uso de aprendizado de máquina pode se tornar útil para a detecção em tempo real de *ransomwares* e ameaças, através da análise de características.

## 1.2 Justificativa

A produção deste trabalho se justifica pela crescente ameaça representada pelo *ransomware*. Com a dependência cada vez maior de sistemas digitais, a proteção contra esses ataques é essencial para garantir a continuidade de serviços e a segurança de informações críticas.

Nesse contexto, a detecção de *ransomwares* por meio de técnicas de aprendizado de máquina tem emergido como uma solução promissora. A capacidade de aprender com padrões de comportamento malicioso e identificar anomalias em tempo real oferece uma camada adicional de proteção, que pode superar as limitações das soluções tradicionais, como antivírus e *firewalls*.

Portanto, este trabalho é de extrema relevância ao fornecer soluções que visam preservar a integridade dos dados, proteger redes contra invasões e fortalecer a defesa cibernética, contribuindo significativamente para a segurança digital ao aplicar técnicas de aprendizado de máquina para a detecção de *ransomware* em arquivos executáveis, com foco na identificação de suas características.

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

O presente trabalho tem a finalidade de produzir um protótipo que simula um *ransomware*, estudar e analisar sobre o seu funcionamento, suas condutas dentro do sistema, os tipos de arquivos que podem ser afetados pelo *malware*, sua geração de chaves de criptografia, as formas de infecção e desenvolver um modelo para sua detecção e prevenção, fazendo uso de técnicas de aprendizado de máquina.

### 1.3.2 Objetivos Específicos

- Levantar material sobre o assunto.
- Estudar sobre a geração de *malwares* e prevenção de intrusão.
- Construir um protótipo de *ransomware*.
- Pesquisar trabalhos e técnicas recentes de fundamentos de segurança.
- Estudar sobre a geração de chaves de criptografia nos *ransomwares* modernos.
- Propor um modelo de aprendizado de máquina para análise e detecção.
- Aprender sobre técnicas de classificação.
- Avaliar o método da Floresta Aleatória.



## 2 Fundamentação teórica

Neste capítulo, são apresentados os principais conceitos relacionados ao desenvolvimento do projeto, incluindo uma introdução aos *ransomwares*, aprendizado de máquina e o algoritmo de floresta aleatória.

### 2.1 *ransomware*

*ransomware* é um tipo de *malware* projetado para bloquear o acesso a dados ou sistemas até que um resgate seja pago. Com o tempo, esse tipo de ataque cibernético tornou-se uma das ameaças mais sofisticadas e lucrativas para os cibercriminosos. Ataques de *ransomware* criptografam arquivos das vítimas, exigindo pagamento, e sua popularidade aumentou devido ao uso de criptomoedas, que garantem o anonimato dos criminosos (ABOUD & MARIYAPPN, 2021). A figura 1 exemplifica a tela de pagamento do *ransomware* WannaCry.

Figura 1: Tela de pagamento do WannaCry



Fonte: CICALA & BERTINO.

#### 2.1.1 Tipos principais

**Locker ransomware:** Esse tipo impede o acesso ao dispositivo da vítima, mas não criptografa arquivos. O dispositivo continua funcional, mas a vítima é bloqueada de acessá-lo até que o resgate seja pago. Esse método é comum em dispositivos móveis e Internet das Coisas (CICALA & BERTINO, 2020).

**Crypto ransomware:** Este é o tipo mais comum e perigoso de *ransomware*, pois criptografa arquivos importantes, exigindo um resgate para a sua recuperação. O ataque WannaCry, que causou prejuízos globais em 2017, é um exemplo notável de *crypto ransomware* (GENÇ et al., 2021).

### 2.1.2 Funcionamento do *ransomware*

*ransomwares* modernos utilizam algoritmos de criptografia robustos que combinam técnicas simétricas e assimétricas para bloquear arquivos de forma eficaz. A criptografia simétrica, que é rápida, protege os dados, enquanto a criptografia assimétrica, mais segura, protege a chave de descryptografia. Esse modelo, segundo HSU et al. (2021), garante que, mesmo que a vítima descubra a chave simétrica, a recuperação completa dos dados é impossível sem a chave privada do atacante (ABOUD & MARIYAPPN, 2021).

## 2.2 Aprendizado de Máquina

Aprendizado de Máquina, é uma sub-área da inteligência artificial voltada para o desenvolvimento de algoritmos que aprendem com os dados, permitindo que sistemas realizem tarefas de forma autônoma. De acordo com GÉRON (2019), o aprendizado de máquina é essencial para lidar com o aumento exponencial de dados, tornando possível realizar tarefas como reconhecimento de padrões e previsão de resultados.

### 2.2.1 Tipos principais

**Aprendizado Supervisionado:** Nesse tipo de aprendizado, o modelo é treinado com dados rotulados, aprendendo a mapear entradas para saídas. Isso possibilita fazer previsões sobre novos dados, e técnicas comuns incluem a regressão linear e árvores de decisão (BIAU & SCORNET, 2016).

**Aprendizado Não Supervisionado:** Aqui, o modelo lida com dados sem rótulos, buscando padrões ocultos. Exemplos incluem o *clustering* e a análise de componentes principais (PCA) (GÉRON, 2019).

**Aprendizado por Reforço:** Nesta abordagem, um agente aprende com base em recompensas ou punições, sendo amplamente utilizado em aplicações como robótica e jogos, com um exemplo notável sendo o AlphaGo (SUTTON & BARTO, 2018).

### 2.2.2 Aplicações do Aprendizado de Máquina

O aprendizado de máquina possui aplicações em áreas como medicina, finanças e ciência de dados. Em diagnósticos médicos, algoritmos de aprendizado supervisionado auxiliam na detecção de tumores em exames de imagem (BIAU & SCORNET, 2016). Na ciência de dados, o algoritmo de floresta aleatória combina várias árvores de decisão para lidar com grandes volumes de dados de forma eficiente, aplicando-se a tarefas de classificação e regressão (GÉRON, 2019).

## 2.3 Floresta Aleatória

A Floresta Aleatória é um algoritmo de aprendizado supervisionado amplamente utilizado em classificação e regressão. Desenvolvido por BREIMAN (2001), o algoritmo combina diversas árvores de decisão para criar um modelo robusto e preciso, capaz de lidar com grandes volumes de dados e variáveis.

### 2.3.1 Funcionamento da Floresta Aleatória

O algoritmo de floresta aleatória é baseado no método *ensemble* conhecido como *bagging* (*Bootstrap Aggregating*). Nesse método, várias árvores de decisão são construídas a partir de subconjuntos dos dados de treino, gerados por amostragem com reposição, e suas previsões são combinadas para gerar o resultado final (GÉRON, 2019). A diversidade entre as árvores é garantida pela aleatoriedade introduzida tanto nos dados quanto na seleção de variáveis, reduzindo a correlação entre as árvores e melhorando a precisão geral do modelo (BIAU & SCORNET, 2016).

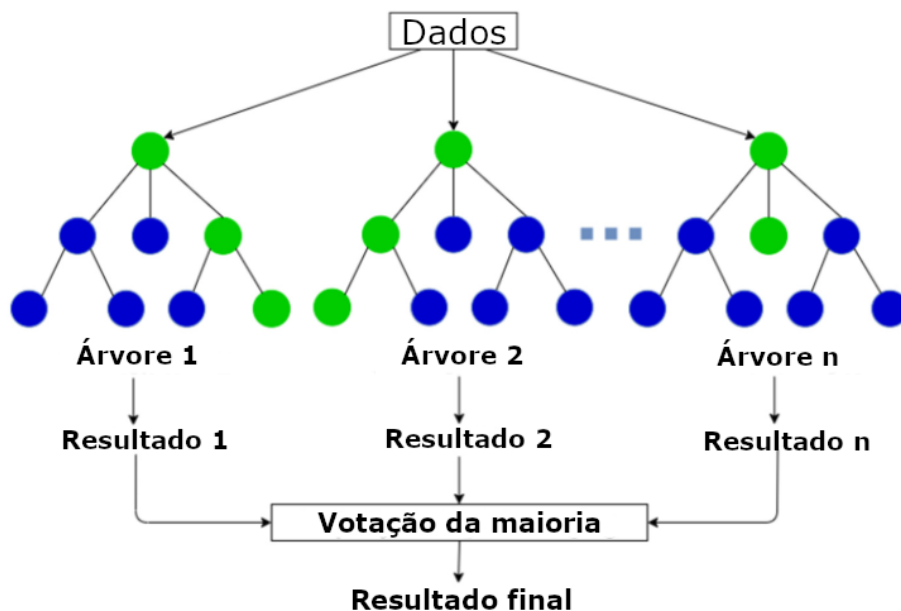
### 2.3.2 Componentes Principais

**Bagging:** Esse processo consiste em treinar várias árvores de decisão independentes com subconjuntos aleatórios dos dados, criando um modelo robusto e menos suscetível ao *overfitting* (BREIMAN, 2001).

**Árvores de Decisão:** Cada árvore é construída com o algoritmo *CART* (*Classification and Regression Trees*), que divide os dados com base nas variáveis preditoras, utilizando critérios como impureza de Gini ou erro quadrático médio (BIAU & SCORNET, 2016).

**Votação:** A Floresta Aleatória combina os resultados das árvores para gerar a previsão final. Em tarefas de classificação, o resultado é decidido por votação; para regressão, a média das previsões é utilizada (GÉRON, 2019). Podemos observar o processo na figura 2.

Figura 2: Simplificação da floresta aleatória



Fonte: Elaborado pelo autor.

### 2.3.3 Vantagens e limitações

A Floresta Aleatória oferece diversas vantagens, incluindo a redução do *overfitting*, uma vez que a combinação de várias árvores cria um modelo mais generalizado (BIAU& SCORNET, 2016). Outra vantagem é a capacidade de lidar com dados desbalanceados, devido ao treinamento independente das árvores, o que garante a consideração de classes minoritárias (BREIMAN, 2001). Além disso, o algoritmo permite medir a importância das variáveis, identificando as mais relevantes para a predição.

Apesar de sua eficiência, a Floresta Aleatória possui algumas limitações, como o alto custo computacional em grandes conjuntos de dados. Além disso, em conjuntos de dados com muitas variáveis irrelevantes, outros métodos mais simples podem ser mais eficientes (GÉRON, 2019; BREIMAN, 2001).

## 2.4 Criptografia

A criptografia é uma técnica essencial para proteger informações, garantindo a confidencialidade, integridade e autenticidade de dados em sistemas digitais. Seu uso abrange desde a proteção de senhas e dados bancários até a segurança de comunicações e armazenamento em nuvem. Existem dois tipos principais de criptografia:

**Criptografia simétrica:** utiliza a mesma chave para criptografar e descriptografar dados. É mais rápida e eficiente em termos de processamento, sendo amplamente empregada em sistemas como o AES (*Advanced Encryption Standard*). A principal desvantagem é a necessidade de compartilhar a chave entre as partes, o que pode expor os dados ao risco de interceptação.

**Criptografia assimétrica:** emprega um par de chaves, uma pública para criptografar e outra privada para descriptografar. Esse modelo é usado em algoritmos como o RSA (*Rivest-Shamir-Adleman*), que garantem maior segurança em trocas de dados sensíveis, como assinaturas digitais e certificados SSL/TLS. Contudo, é mais lento devido à sua complexidade matemática.

Atualmente, a criptografia é fundamental em áreas como comércio eletrônico, comunicações seguras e autenticação digital, contribuindo para a privacidade e segurança de milhões de usuários em todo o mundo.

### 2.4.1 Criptografia AES

A criptografia AES (*Advanced Encryption Standard*) de 128 bits é amplamente reconhecida como um dos métodos mais seguros e eficientes para a proteção de dados. Adotada pelo Instituto Nacional de Padrões e Tecnologia (NIST) dos Estados Unidos em 2001, o AES substituiu o DES (*Data Encryption Standard*), tornando-se o padrão para criptografia simétrica devido à sua robustez e desempenho (NICKY MOUHA, 2021).

O AES de 128 bits opera em blocos de dados de 128 bits, utilizando uma chave de mesmo tamanho para criptografar e descriptografar as informações. Esse algoritmo é baseado no princípio de substituição-permutação e utiliza rodadas de transformação para aumentar a segurança. Cada rodada inclui quatro operações principais:

***SubBytes***: substituição não linear de bytes utilizando uma tabela de substituição (*S-box*) que introduz complexidade ao processo.

***ShiftRows***: reorganização das linhas da matriz de dados para misturar os bytes e evitar padrões simples.

***MixColumns***: combinação linear das colunas, dificultando a reconstrução dos dados originais.

***AddRoundKey***: combinação do bloco de dados com a chave da rodada por meio de uma operação XOR.

Para o AES de 128 bits, o processo completo consiste em 10 rodadas dessas transformações, garantindo a segurança contra ataques de força bruta e outros métodos de violação.

o AES é amplamente utilizado em dispositivos de armazenamento, como discos rígidos, e na proteção de redes, como conexões Wi-Fi seguras (WPA2). É importante também reforçar a eficiência do algoritmo em sistemas que demandam alta performance devido à sua rápida execução e compatibilidade com dispositivos modernos. Por fim, destaca-se a resistência do AES a ataques de força bruta, mesmo quando chaves menores, como a de 128 bits, são utilizadas. Além de sua aplicação prática, o AES é uma escolha estratégica em segurança devido à sua simplicidade arquitetural e capacidade de adaptação a diferentes plataformas de *hardware e software*, tornando-se um dos pilares da segurança cibernética contemporânea.

### 2.4.2 Criptografia RSA

A criptografia RSA (Rivest-Shamir-Adleman) foi apresentada em 1978 por Ron Rivest, Adi Shamir e Leonard Adleman. Este sistema utiliza a dificuldade de fatorar números inteiros muito grandes como base para sua segurança, sendo um dos métodos de criptografia assimétrica mais amplamente utilizados (RIVEST et al., 1978). O RSA é dividido em três etapas principais: geração de chaves, criptografia e descryptografia.

- **Geração de chaves:** Escolhem-se dois números primos grandes  $p$  e  $q$ . Calcula-se  $n = p \cdot q$ , que define o módulo. Determina-se a função totiente de Euler  $\phi(n) = (p - 1) \cdot (q - 1)$ . Escolhe-se um número  $e$  que seja coprimo com  $\phi(n)$ . Por fim, calcula-se  $d$ , o inverso modular de  $e$  em relação a  $\phi(n)$ .
- **Criptografia:** Uma mensagem  $M$ , representada como um inteiro  $0 < M < n$ , é transformada no texto cifrado  $C$  por meio da equação  $C \equiv M^e \pmod{n}$ .
- **Descryptografia:** O receptor recupera a mensagem original usando  $M \equiv C^d \pmod{n}$ .

O RSA é amplamente utilizado para proteger comunicações digitais em protocolos como SSL/TLS. Sua segurança está diretamente ligada ao tamanho de  $n$ , sendo recomendadas chaves de pelo menos 2048 bits para resistir a ataques de força bruta (RIVEST et al., 1978).

Apesar de ser seguro, o RSA apresenta limitações, como a lentidão em comparação a algoritmos simétricos e a necessidade de números primos grandes para evitar vulnerabilidades específicas, como ataques baseados em chaves privadas pequenas (RIVEST et al., 1978). A figura 3 mostra uma tabela criada pelos autores da criptografia RSA, que relaciona os dígitos da chave com a quantidade de operações e o tempo necessário para a quebra da criptografia.

Figura 3: Tabela sobre o tempo de quebra da criptografia RSA

Table I.

Digits	Number of operations	Time
50	$1.4 \times 10^{10}$	3.9 hours
75	$9.0 \times 10^{12}$	104 days
100	$2.3 \times 10^{15}$	74 years
200	$1.2 \times 10^{23}$	$3.8 \times 10^9$ years
300	$1.5 \times 10^{29}$	$4.9 \times 10^{15}$ years
500	$1.3 \times 10^{39}$	$4.2 \times 10^{25}$ years

Fonte: RIVEST et al.

## 2.5 Trabalhos correlatos

GANTA et al. (2020) desenvolveram um método para classificar arquivos executáveis como benignos ou infectados por *ransomware* em tempo real. Utilizaram o algoritmo de Floresta Aleatória, atingindo quase 98% de precisão entre os 8.000 arquivos utilizados.

USHA et al. (2021) desenvolveram um classificador usando o algoritmo da Floresta aleatória, e perceberam que a técnica de KNN é menos custosa, mais leve e exige um conjunto de dados menor, mantendo uma porcentagem alta de acertos.

HSU et al. (2021) desenvolveram um método de identificar 4 tipos de *ransomware*. Usando um SVM (*kernel trick*), 1000 arquivos corrompidos de 22 tipos, além de arquivos de criptografia comum para reduzir falsos positivos, foi alcançada a marca de 92% de precisão.

SMITH et al. (2022) documentaram diferentes tipos de *frameworks* e algoritmos de detecção de *ransomware*, suas abordagens, desafios de implementação, pontos positivos e negativos. Usaram diferentes técnicas de aprendizado de máquina e *deep learning*.

ASAJU et al. (2021) utilizaram um algoritmo de decisão em árvore (J48) para classificar arquivos em benignos ou infectados. Possibilitando ainda identificar *ransomwares* que passaram despercebidos pelo sistema, auxiliando na criação de antivírus mais eficientes.

## 3 Metodologia

A seguinte seção tem o propósito de apresentar todo o processo de criação deste projeto, iniciando pelo protótipo do *ransomware* e seguido pelo modelo de detecção.

### 3.1 Materiais e ambiente

Para a realização deste trabalho, foram utilizados os seguintes equipamentos e *softwares*:

**Equipamentos:** Desktop com processador AMD Ryzen 5 3600 6-Core de 3.59 GHz e uma Memória RAM de 8 GB e 2666 MHz ddr4 (não foi feito o uso de qualquer placa de vídeo).

**softwares:** Windows 11 Home, VS Code 1.94.2, Python 3.12.5 64-bit, Bibliotecas Scikit-Learn, Crypto, Matplotlib, Pandas, Pefile e SHAP.

#### 3.1.1 Descrição das bibliotecas Python

- **Scikit-Learn:** A biblioteca **Scikit-Learn** é uma das mais populares para aprendizado de máquina em Python. Ela fornece ferramentas eficientes para modelagem preditiva e análise de dados. Seus principais recursos incluem algoritmos de classificação, regressão, *clustering* e redução de dimensionalidade, além de funções para pré-processamento, validação cruzada e *pipelines*. A Scikit-Learn é projetada para trabalhar bem com outras bibliotecas como NumPy e pandas, tornando-a uma escolha versátil para cientistas de dados e engenheiros de aprendizado de máquina.
- **Crypto:** A biblioteca **Crypto** (ou PyCrypto) é utilizada para implementações de algoritmos criptográficos, como AES, RSA e SHA. Ela fornece ferramentas essenciais para criptografia e descryptografia de dados, geração de chaves e criação de assinaturas digitais. É amplamente usada em aplicações que requerem segurança de dados, incluindo proteção de informações confidenciais e autenticação de usuários.
- **Matplotlib:** A biblioteca **Matplotlib** é uma ferramenta poderosa para a criação de visualizações em Python. Com ela, é possível gerar gráficos 2D e 3D, incluindo linhas, barras, dispersão, histogramas e gráficos de pizza. O Matplotlib é altamente customizável, permitindo ajustar cores, estilos e tamanhos, além de exportar gráficos em diversos formatos. Ele é amplamente usado em conjunto com pandas e NumPy para visualizações baseadas em dados.
- **Pandas:** A biblioteca **Pandas** é uma ferramenta essencial para manipulação e análise de dados. Ela fornece estruturas como **DataFrame** e **Series**, que permitem trabalhar com dados tabulares e temporais de forma eficiente. Suas funcionalidades incluem filtragem, agregação, tratamento de valores ausentes, *merge* de *datasets* e muito mais. Pandas é amplamente utilizada em projetos de ciência de dados, devido à sua facilidade de integração com outras bibliotecas como NumPy, Scikit-Learn e Matplotlib.



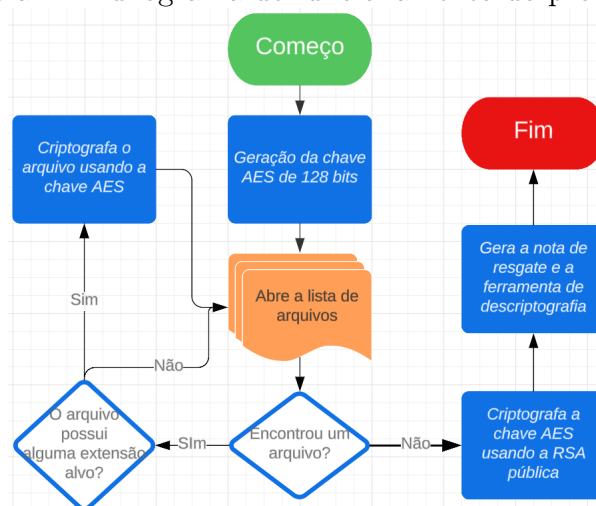
- **Pefile:** A biblioteca **Pefile** é uma ferramenta voltada para análise de arquivos executáveis no formato *Portable Executable* (PE), amplamente utilizado no sistema Windows. Ela permite extrair informações detalhadas sobre a estrutura de arquivos PE, como cabeçalhos, seções, *imports* e *exports*. Essa biblioteca é frequentemente empregada em pesquisas de segurança cibernética, especialmente na análise de *malware*.
- **SHAP:** A biblioteca **SHAP** (*SHapley Additive exPlanations*) é usada para interpretar modelos de aprendizado de máquina. Ela fornece explicações detalhadas sobre as previsões de modelos ao atribuir contribuições individuais a cada recurso de entrada. A SHAP é amplamente utilizada em contextos que exigem transparência e interpretabilidade, ajudando a entender o impacto de variáveis nos resultados preditivos de modelos complexos, como árvores de decisão e redes neurais.

## 3.2 Protótipo

Este protótipo de *ransomware* foi desenvolvido como uma ferramenta educacional para aprofundar o entendimento sobre *ransomware*, sua estrutura e seus métodos de operação. O código simula as principais etapas de um ataque *ransomware* real, abordando desde a geração de chaves de criptografia até a manipulação de arquivos para simular um cenário de extorsão digital.

O protótipo não foi criado com intenções maliciosas, mas sim para proporcionar um ambiente de estudo e prática em segurança cibernética, com foco na análise comportamental de *ransomwares*. Ao entender os mecanismos de criptografia e descriptografia, pesquisadores podem desenvolver habilidades para detectar, prevenir e mitigar ataques de *ransomware*, aplicando esse conhecimento na proteção de sistemas reais. A figura 4 exemplifica as etapas do protótipo.

Figura 4: Fluxograma de funcionamento do protótipo



Fonte: Elaborado pelo autor.

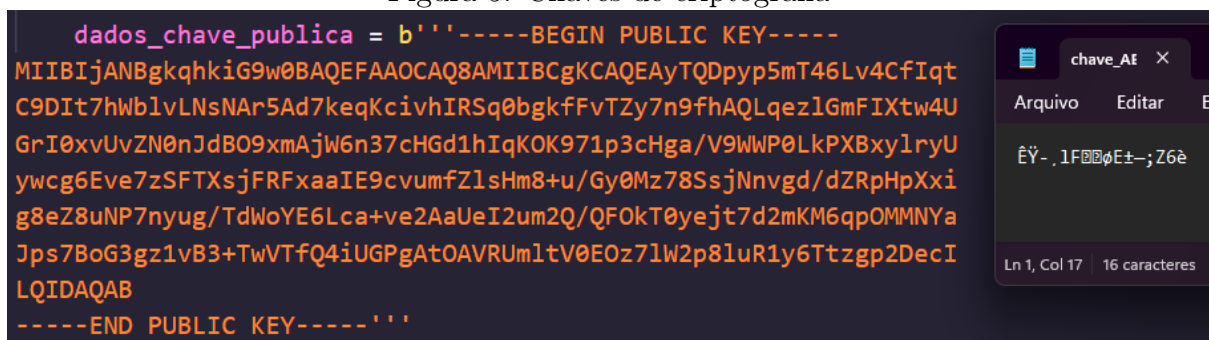
O *ransomware* começa gerando uma chave AES de 128 bits, que será usada para criptografar os arquivos no sistema da vítima. Em seguida, ele percorre uma lista de arquivos no dispositivo, identificando aqueles que possuem uma extensão de interesse, como .pdf, .txt ou .png. Quando encontra um arquivo relevante, o *ransomware* realiza a criptografia utilizando a chave AES em modo CBC (*Cipher Block Chaining*). Para cada arquivo criptografado, é gerado um vetor de inicialização (IV) aleatório, o que garante que, mesmo se dois arquivos tiverem o mesmo conteúdo, a criptografia aplicada será única.

Após criptografar todos os arquivos selecionados, o *ransomware* protege a chave AES que usou no processo. Para isso, ele utiliza criptografia RSA, empregando uma chave pública que está embutida no próprio código. Esse procedimento garante que apenas alguém com a chave privada correspondente, que está em posse dos atacantes, possa descriptografar a chave AES.

Finalmente, o *ransomware* cria uma nota de resgate, que fornece instruções para a vítima sobre como realizar o pagamento e enviar o comprovante. A nota geralmente contém detalhes sobre o valor do resgate e o método de contato, deixando claro que, ao efetuar o pagamento, a vítima receberá a chave privada RSA para descriptografar a chave AES. Com a posse dessa chave, a vítima pode, então, usar uma ferramenta de descriptografia fornecida para restaurar o acesso aos seus arquivos.

A figura 5 mostra uma imagem das chaves RSA (pública, embutida no código à esquerda) e chave AES (gerada num arquivo .key, à direita) que foram usadas para a criptografia dos arquivos:

Figura 5: Chaves de criptografia



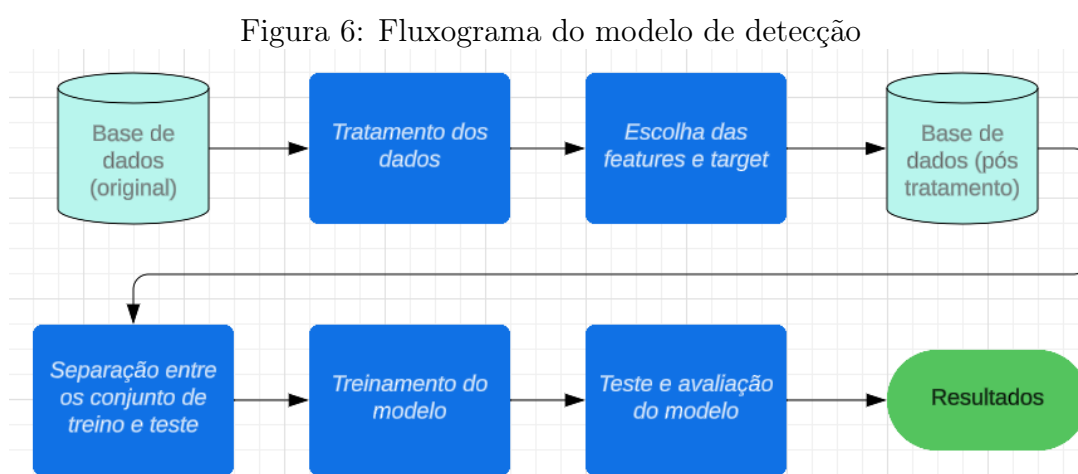
Fonte: Elaborado pelo autor.

A chave pública não contém os trechos 'BEGIN PUBLIC KEY' ou 'END PUBLIC KEY', sendo apenas formalidades para o compilador. Já a chave AES contém caracteres que possam não estar na tabela ASCII, e por isso apresenta alguns caracteres especiais.

### 3.3 Modelo

O modelo proposto utiliza o algoritmo de Floresta Aleatória para identificar e classificar arquivos como benignos ou potencialmente maliciosos, com o objetivo de detectar comportamentos associados a *ransomware*. Floresta aleatória é um método de aprendizado supervisionado baseado em técnicas de *ensemble*, onde múltiplas árvores de decisão são geradas e combinadas para aprimorar a precisão e a robustez da classificação. Esse método é amplamente reconhecido por sua eficácia em tarefas de detecção e classificação devido à sua capacidade de lidar com grandes conjuntos de dados e identificar padrões complexos.

A figura 6 descreve o processo de criação e treinamento do modelo, etapa por etapa.



Fonte: Elaborado pelo autor.

O fluxograma apresentado descreve todas as etapas para a criação e validação de um modelo de detecção de *ransomware*, desenvolvido com o objetivo de analisar dados específicos de arquivos e identificar características que possam diferenciar arquivos benignos de potenciais ameaças. A base de dados utilizada para treinar e avaliar o modelo foi adquirida no Kaggle, uma plataforma que oferece conjuntos de dados de alta qualidade e de diversas áreas de aplicação.

O processo inicia-se com a importação da base de dados original, contendo uma variedade de informações relevantes sobre arquivos, como tamanho das seções, versão do sistema operacional e tamanho da pilha reservada. Esses atributos são essenciais para treinar um modelo que possa reconhecer padrões associados a arquivos benignos e maliciosos. Em seguida, realiza-se o tratamento dos dados. Esse tratamento envolve principalmente a remoção de *outliers*, que são valores extremos que podem distorcer o comportamento do modelo, e a eliminação de informações pouco relevantes que poderiam introduzir ruído, comprometendo a precisão da classificação. Dessa forma, apenas os dados mais significativos são mantidos para análise posterior.

Após o tratamento dos dados, procede-se com a seleção das características e da variável *target* (ou variável alvo). As características escolhidas incluem atributos específicos, como o tamanho de exportação, o número de seções e o tamanho da pilha reservada, que possuem maior relevância para identificar comportamentos associados a *ransomwares*. A variável alvo, que será prevista pelo modelo, indica se o arquivo é benigno ou suspeito, servindo como base para a análise e classificação.

Com a base de dados já tratada e selecionada, realiza-se a divisão em dois subconjuntos: treino e teste. O conjunto de treino é utilizado para ajustar o modelo e encontrar padrões relevantes, enquanto o conjunto de teste é reservado para avaliar o desempenho do modelo em dados que ele ainda não viu, garantindo uma validação imparcial e um indicador real de como o modelo se comportará em situações do mundo real.

O modelo é treinado com o uso de um algoritmo de floresta aleatória, uma técnica amplamente utilizada em classificação e análise de dados complexos. A floresta aleatória cria múltiplas árvores de decisão, cada uma considerando subconjuntos aleatórios dos dados e das características, e, ao final, realiza uma votação para determinar a classificação final. Essa abordagem de *ensemble* aumenta a precisão e reduz o risco de *overfitting*, tornando o modelo mais confiável e generalizável.

Após o treinamento, o modelo passa pelo processo de avaliação usando o conjunto de teste. Métricas como precisão, matriz de confusão e relatórios de classificação são gerados para verificar o desempenho do modelo e sua capacidade de classificar corretamente os arquivos. Essa análise permite verificar a eficácia do modelo em ambientes simulados e ajustar qualquer parâmetro necessário para melhorar sua acurácia. A última etapa do fluxo é a exibição dos resultados finais, incluindo as métricas de desempenho e a importância das características para a decisão do modelo. Através desta análise, é possível entender quais características têm maior impacto na classificação, o que é útil para refinar o modelo e para estudos futuros sobre as propriedades dos *ransomwares*.

## 4 Resultados do *ransomware*

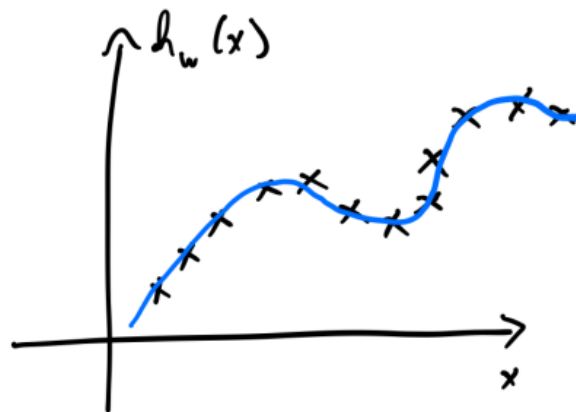
### 4.1 Arquivos criptografados

Após a execução do *ransomware*, ele passa a examinar todos os arquivos do diretório em que está instalado e em diante, ignorando arquivos críticos do sistema (como arquivos .exe e .dll) e atacando arquivos pessoais que, no caso deste projeto, foram escolhidos os arquivos com as extensões .pdf, .txt e .png.

A seguir, alguns exemplos de arquivos alvo (antes e depois da criptografia).

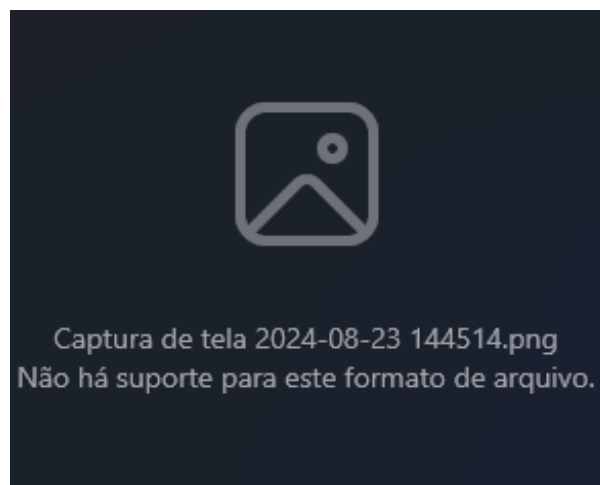
A figura 7 mostra um arquivo .png antes da criptografia, e a figura 8 mostra o mesmo arquivo, agora inacessível.

Figura 7: Arquivo .png antes da criptografia



Fonte: Elaborado pelo autor.

Figura 8: Arquivo .png após criptografia



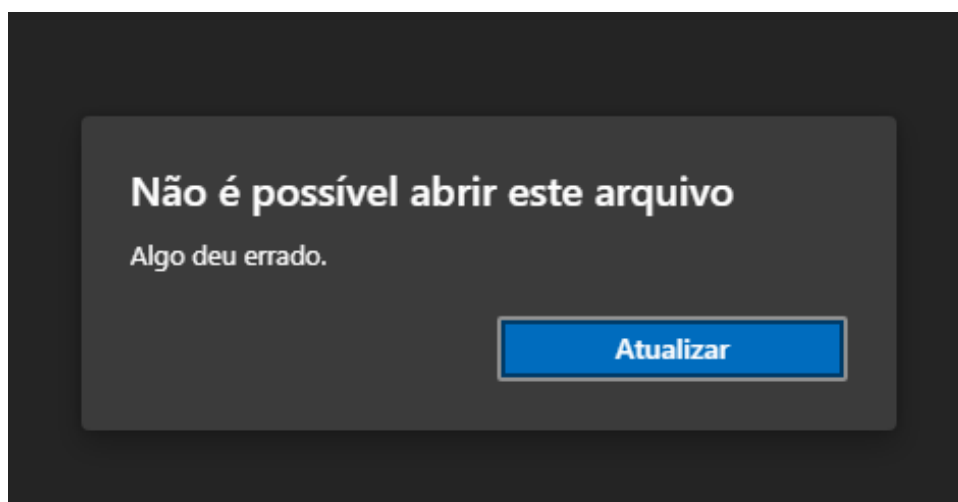
Fonte: Elaborado pelo autor.

A figura 9 mostra um arquivo .pdf antes da criptografia, e a figura 10 mostra o mesmo arquivo, agora inacessível.

Figura 9: Arquivo .pdf antes da criptografia



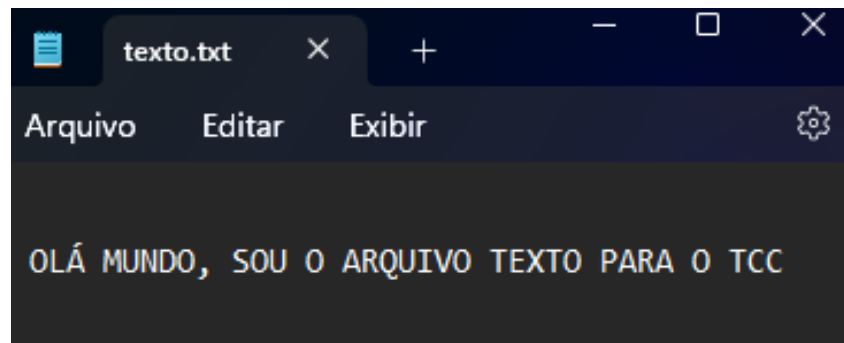
Figura 10: Arquivo .pdf após criptografia



Fonte: Elaborado pelo autor.

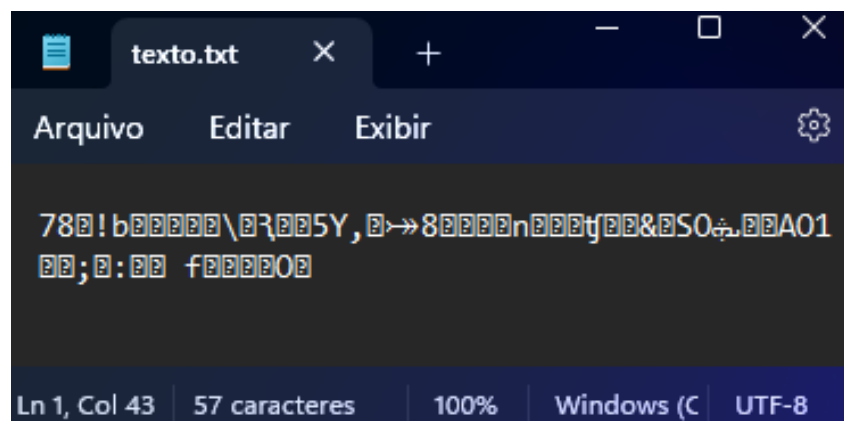
A figura 11 mostra um arquivo .txt antes da criptografia, e a figura 12 mostra o mesmo arquivo, agora inacessível.

Figura 11: Arquivo .txt antes da criptografia



Fonte: Elaborado pelo autor.

Figura 12: Arquivo .txt após criptografia

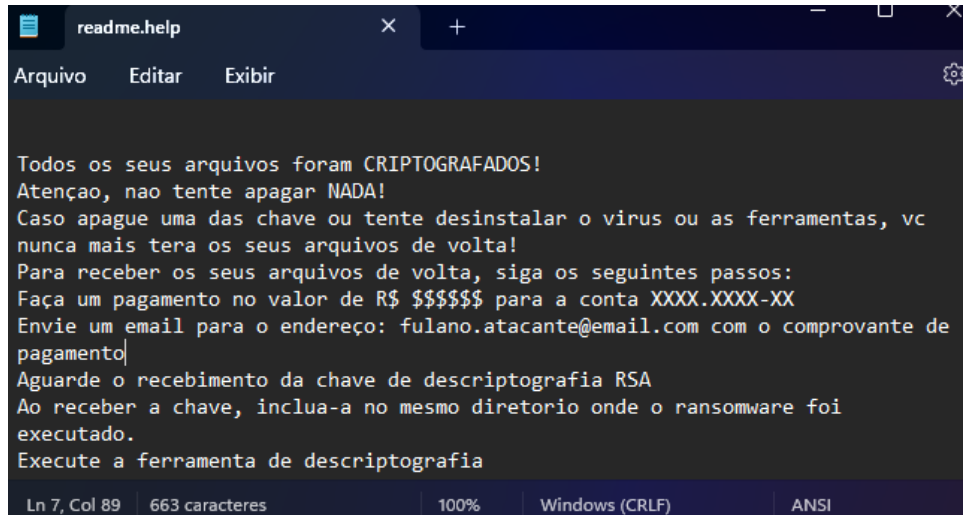


Fonte: Elaborado pelo autor.

## 4.2 Arquivos gerados pelo *ransomware*

Após sua execução, o protótipo também gerou uma nota de resgate, conforme a figura 13, e uma ferramenta de descriptografia, para auxiliar o usuário após o pagamento do resgate.

Figura 13: Nota de resgate



Fonte: Elaborado pelo autor.

A ferramenta verifica a existência de um arquivo contendo a chave AES no diretório especificado. Caso a chave esteja presente, ela é lida e carregada.

Na etapa seguinte, a ferramenta percorre o diretório e suas subpastas em busca de arquivos com as extensões específicas, que estejam criptografados. Ao localizar um arquivo compatível, a ferramenta extrai os primeiros 16 bytes para obter o vetor de inicialização (IV), e então lê o restante do conteúdo do arquivo. Usando o IV e a chave AES previamente carregada, a ferramenta descriptografa o conteúdo e remove o preenchimento adicional inserido durante a criptografia. Por fim, o conteúdo descriptografado é reescrito no próprio arquivo, restaurando-o ao seu estado original.

Para garantir o acesso à chave AES, a ferramenta verifica a existência de uma chave privada RSA, necessária para descriptografar o arquivo que contém a chave AES. Caso a chave privada RSA esteja disponível, a ferramenta a utiliza para descriptografar o conteúdo da chave AES, restaurando-a ao seu formato original e sobrescrevendo o arquivo no diretório especificado. Esse processo assegura que a chave AES utilizada na descriptografia dos arquivos está correta e íntegra. Se a chave privada não estiver presente, a ferramenta exibe uma mensagem informando que a chave RSA ainda não foi recebida, sugerindo a necessidade do pagamento de um resgate.

Ao final do processo, a ferramenta obtém o caminho do diretório onde está sendo executada, certifica-se de que a chave AES está descriptografada e carregada, e realiza a descriptografia dos arquivos presentes no diretório especificado. Dessa forma, a ferramenta cumpre seu objetivo de restaurar os arquivos afetados por meio de um processo controlado e seguro.



## 5 Resultados do modelo

Na seguinte seção, serão discutidos os resultados do modelo, as métricas utilizadas em sua avaliação e sua execução.

### 5.1 Métricas utilizadas

#### 5.1.1 Acurácia

Mede a proporção de previsões corretas em relação ao total de previsões feitas. É calculada pela fórmula da equação 1:

$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

onde:

- **TP** = Verdadeiros Positivos (previsões corretas de positivos),
- **TN** = Verdadeiros Negativos (previsões corretas de negativos),
- **FP** = Falsos Positivos (previsões incorretas de positivos),
- **FN** = Falsos Negativos (previsões incorretas de negativos).

Uma métrica simples e útil para verificar a proporção de acertos, mas pode ser enganosa quando a base de dados é desbalanceada, ou seja, quando uma classe é muito mais frequente que a outra.

#### 5.1.2 Validação Cruzada

Técnica que divide os dados em várias partes (ou “*folds*”) e executa o treinamento do modelo várias vezes, cada vez com um “*fold*” diferente sendo utilizado para teste e os restantes para treino. A média das acurácias obtidas em cada rodada é usada como a acurácia final. Para este modelo, em específico, a base de dados de teste foi dividida em 5, usando uma das partes para o treino e o restante para o teste.

#### 5.1.3 Precisão

Indica a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas. Calculado conforme a equação 2:

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

Esta métrica é importante em casos onde é necessário minimizar falsos positivos, ou seja, quando um erro ao classificar algo como positivo tem um custo alto.

#### 5.1.4 Recall

Indica a proporção de previsões positivas corretas em relação ao total de verdadeiros positivos presentes nos dados. Calculado conforme a equação 3:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Esta métrica é útil em situações onde é importante reduzir falsos negativos, ou seja, onde é crítico identificar todos os casos positivos.

#### 5.1.5 F1-Score

Média harmônica entre a precisão e o recall, buscando um equilíbrio entre ambas as métricas. Calculado conforme a equação 4:

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

O F1-Score é particularmente útil quando há um desbalanceamento entre classes e é necessário equilibrar precisão e revocação.

#### 5.1.6 Matriz de Confusão

Apresenta uma visão detalhada das previsões do modelo, separando-as em verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. A figura 14 ilustra a matriz de confusão:

Figura 14: Matriz de confusão

	Previsão: Sim	Previsão: Não
Realidade: Sim	Positivo Verdadeiro	Falso Negativo
Realidade: Não	Falso Positivo	Negativo Verdadeiro

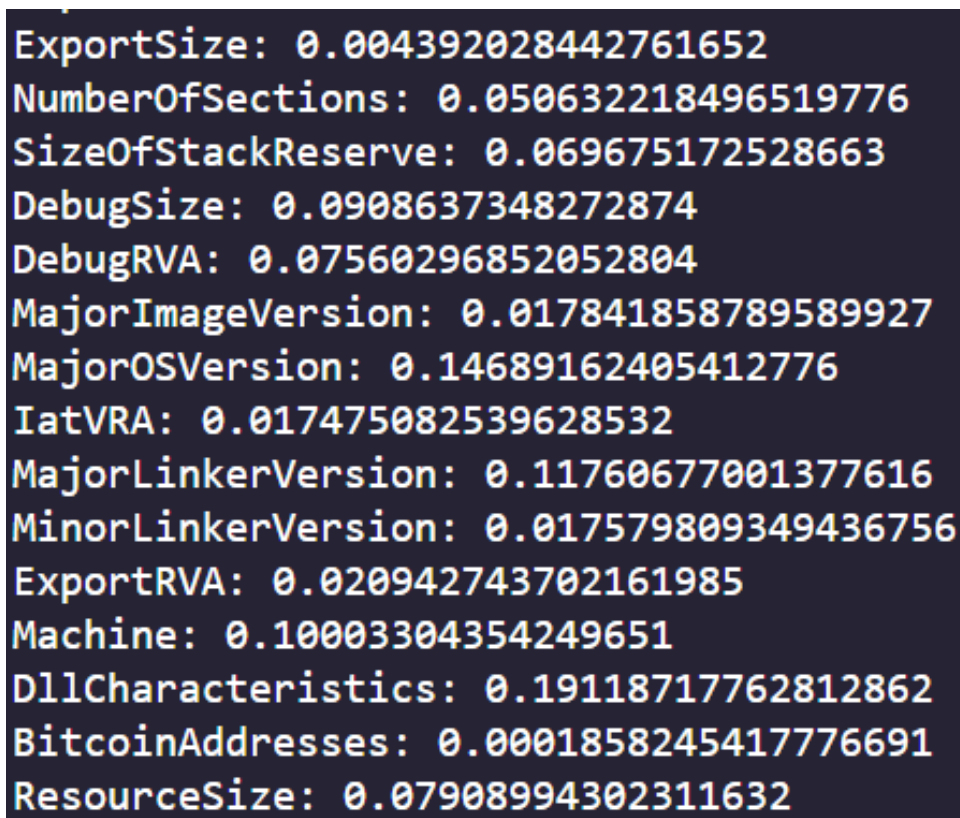
Fonte: Elaborado pelo autor.

Essa matriz ajuda a visualizar o desempenho do modelo em cada classe, sendo especialmente útil para detectar padrões de erro específicos.

## 5.2 Resultados obtidos

Para os primeiros resultados obtidos, o modelo apresentou uma precisão extremamente elevada, acima de 99%, sugerindo fortes indícios de *overfitting*. Para contornar essa situação, foi realizada uma análise detalhada de cada característica utilizada no treinamento do modelo. Como primeira tentativa de analisar as características, foi feito um simples levantamento de quanto cada uma contribuía (em porcentagem) para o treinamento do modelo. A figura 15 mostra os resultados obtidos.

Figura 15: Importância por característica (em porcentagem)



```
ExportSize: 0.004392028442761652
NumberOfSections: 0.050632218496519776
SizeOfStackReserve: 0.069675172528663
DebugSize: 0.0908637348272874
DebugRVA: 0.07560296852052804
MajorImageVersion: 0.017841858789589927
MajorOSVersion: 0.14689162405412776
IatVRA: 0.017475082539628532
MajorLinkerVersion: 0.11760677001377616
MinorLinkerVersion: 0.017579809349436756
ExportRVA: 0.020942743702161985
Machine: 0.10003304354249651
DllCharacteristics: 0.19118717762812862
BitcoinAddresses: 0.0001858245417776691
ResourceSize: 0.07908994302311632
```

Fonte: Elaborado pelo autor.

O conjunto de dados possui um total de 18 campos, porém, o campo "Benign" não é utilizado dentro das características (por ser o alvo), e os campos "FileName" e "md5Hash" também não foram utilizados, por se tratarem de campos alfanuméricos.

Após essa simples análise inicial, fica claro que nem todas as características são significativas para o treinamento do modelo. Foi feito então a remoção de características com uma contribuição menor ou igual a 5% (0.05), removendo então os campos: "ExportSize", "MajorImageVersion", "IatVRA", "MinorLinkerVersion", "ExportRVA" e "BitcoinAddresses".

Apesar dessas mudanças, o modelo continuava apresentando sintomas de *overfitting*. E foi decidido uma análise mais minuciosa no conjunto de dados. Foi possível então encontrar que a característica "Machine" estava fortemente relacionada ao alvo "Benign", influenciando e enviesando as decisões do modelo. A figura 16 prova a influência da característica 'Machine':

Figura 16: Análise da característica 'Machine'

```
Valores únicos para o campo Machine: [ 332 34404 452 43620 0 870]
Valores únicos para o campo Benign: [1 0]

Ocorrências do valor 0 em Machine: 1
Ocorrências do valor 332 em Machine: 50624
Ocorrências do valor 34404 em Machine: 11685
Ocorrências do valor 452 em Machine: 98
Ocorrências do valor 43620 em Machine: 76
Ocorrências do valor 870 em Machine: 1

Machine = 332 e Benign = 1: 15263
Machine = 332 e Benign = 0: 35361
Machine = 34404 e Benign = 1: 11681
Machine = 34404 e Benign = 0: 4
```

Fonte: Elaborado pelo autor.

Esta imagem mostra claramente que, majoritariamente, a característica "Machine" é composta pelos valores '332' e '34404' (desprezando os outros valores, devido a quantidade insignificante de ocorrências). Mostra-se no dataset que a ocorrência de "*ransomware*" (Benign = 0) está fortemente presente quando o campo "Machine" é '332' (35.361 ocorrências), e ao mesmo tempo, é praticamente nula quando o campo "Machine" é '34404' (apenas 4 ocorrências). Isso justifica o porquê da característica "Machine" ser descartada, pois sempre que ela apresenta o valor '332' o modelo tende a dizer que se trata de um *ransomware*, e quando apresenta o valor '34404' o modelo tende a dizer que se trata de um arquivo benigno. Portanto, após análises, as características selecionadas por possuir alguma relevância são:

**NumberOfSections** indica quantas seções existem em um arquivo executável, refletindo sua complexidade e funcionalidade. **SizeOfStackReserve** é a quantidade de memória reservada para a pilha do programa, sugerindo o uso de variáveis locais e complexidade nas operações.

**DebugSize** representa o tamanho dos dados de depuração, importantes para identificar erros durante o desenvolvimento; um tamanho maior pode indicar um *software* mais complexo. **DebugRVA** é o endereço onde esses dados de depuração estão localizados, facilitando seu acesso durante a depuração.

**MajorOSVersion** informa a versão principal do sistema operacional para o qual o executável foi projetado, essencial para compatibilidade. **MajorLinkerVersion** refere-se à versão do vinculador utilizado na compilação, impactando as características do arquivo.

A característica **ResourceSize** indica o tamanho dos recursos incorporados em um arquivo executável, como ícones, imagens ou outros dados necessários para o funcionamento. Em modelos de detecção, um tamanho incomum pode identificar arquivos maliciosos.

Por fim, **DllCharacteristics** oferece informações sobre as características da DLL, como segurança em ambientes *multithread* e funcionalidades especiais, ajudando a entender sua interação com outros componentes.

A seguir, os resultados do modelo após o seu treinamento com as características indicadas anteriormente, exibindo suas métricas e acurácia, conforme a figura 17, e as novas características e suas importâncias, conforme a figura 18.

Figura 17: Métricas do modelo

```
Precisão do Modelo: 0.9793022511469114
Validação Cruzada: [0.99183804 0.9937585 0.99383852 0.99567896 0.99495879]
Média da validação cruzada: 0.9940145634952389

Report das Classificações:
      precision    recall  f1-score   support

     0       0.97       0.99       0.98       10661
     1       0.99       0.96       0.98        8085

 accuracy          0.98
 macro avg          0.98
weighted avg          0.98

Matriz de Confusão:
[[10567   94]
 [  294 7791]]
```

Fonte: Elaborado pelo autor.

Figura 18: Resultados após a nova escolha das características

```
Importância das Features:
NumberOfSections: 0.03803961604566686
SizeOfStackReserve: 0.07779992711544548
DebugSize: 0.20312268513790158
DebugRVA: 0.10857346353226245
MajorOSVersion: 0.09557941823823797
MajorLinkerVersion: 0.08988067773132569
DllCharacteristics: 0.28998706402273033
ResourceSize: 0.09701714817642973
```

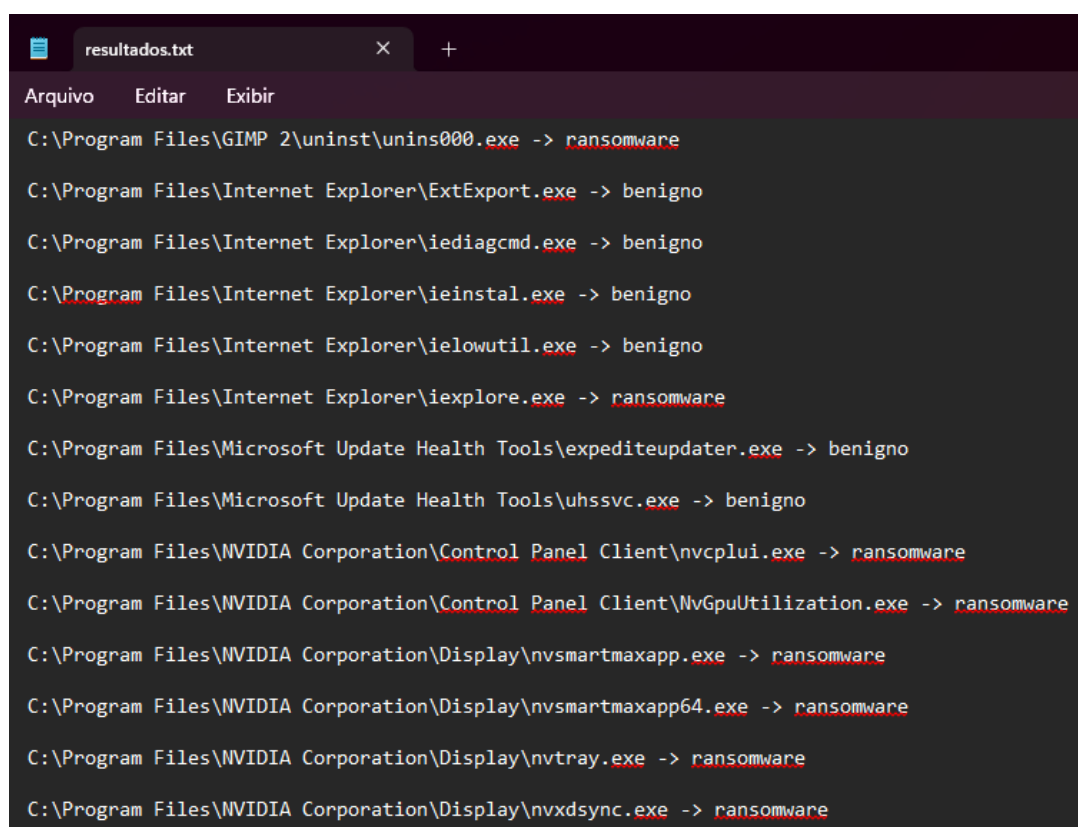
Fonte: Elaborado pelo autor.

### 5.3 Testando novos dados

Após uma seleção mais rigorosa das características e com o modelo completamente treinado, novos dados foram extraídos para serem testados e rotulados pelo modelo. Os novos dados são compostos por arquivos executáveis (.exe) retirados da pasta "C:\Arquivos de Programas (x86)", e suas características foram extraídas através da biblioteca "pefile" do Python. A biblioteca pefile é uma ferramenta em Python para análise de arquivos executáveis no formato *Portable Executable* (PE), usado em sistemas Windows. Ela permite extrair diversas informações estruturais e de metadados dos arquivos .exe, como seções, tabelas de importação e exportação, recursos e características gerais do PE. Essas informações são úteis para análise estática de *malware*, pois permitem acessar detalhes internos do arquivo executável sem executá-lo, auxiliando na criação de modelos preditivos para detecção de *malware*.

A figura 19 mostra parte dos resultados obtidos pelo modelo, que são salvos num arquivo de texto simples.

Figura 19: Resultados gerados pelo modelo



```
Arquivo  Editar  Exibir

C:\Program Files\GIMP 2\uninst\unins000.exe -> ransomware
C:\Program Files\Internet Explorer\ExtExport.exe -> benigno
C:\Program Files\Internet Explorer\ieddiagcmd.exe -> benigno
C:\Program Files\Internet Explorer\ieinstal.exe -> benigno
C:\Program Files\Internet Explorer\ielowutil.exe -> benigno
C:\Program Files\Internet Explorer\iexplore.exe -> ransomware
C:\Program Files\Microsoft Update Health Tools\expediteupdater.exe -> benigno
C:\Program Files\Microsoft Update Health Tools\uhssvc.exe -> benigno
C:\Program Files\NVIDIA Corporation\Control Panel Client\nvcplui.exe -> ransomware
C:\Program Files\NVIDIA Corporation\Control Panel Client\NvGpuUtilization.exe -> ransomware
C:\Program Files\NVIDIA Corporation\Display\nvsmartmaxapp.exe -> ransomware
C:\Program Files\NVIDIA Corporation\Display\nvsmartmaxapp64.exe -> ransomware
C:\Program Files\NVIDIA Corporation\Display\nvtray.exe -> ransomware
C:\Program Files\NVIDIA Corporation\Display\nvxdsync.exe -> ransomware
```

Fonte: Elaborado pelo autor.

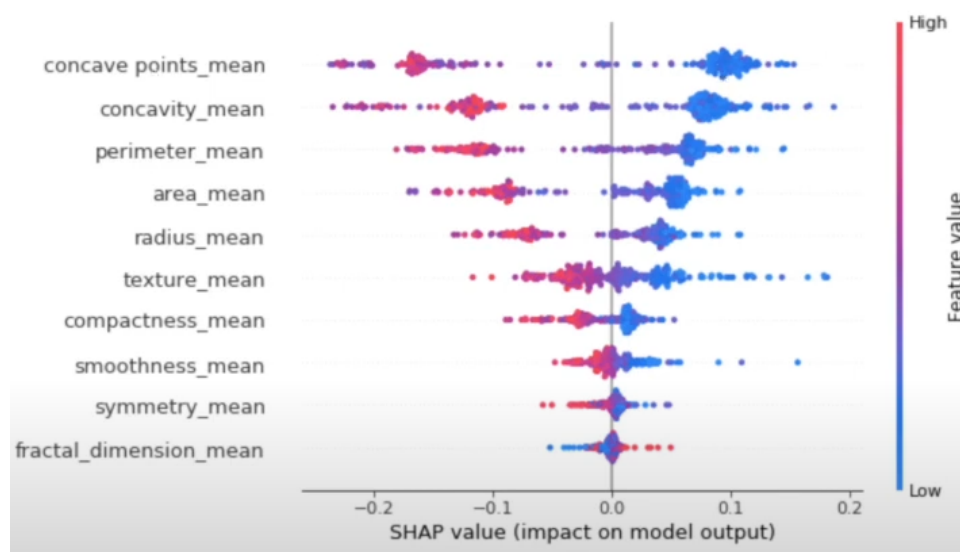
Por se tratarem de arquivos executáveis legítimos, que fazem parte do próprio sistema windows, era de se esperar que a quantidade de arquivos rotulados como "*ransomware*" fosse mínima ou até mesmo nula. Entretanto, ao analisar o arquivo de texto, pode se notar que cerca de 60% dos resultados foram rotulados como sendo maliciosos, apesar dos resultados satisfatórios durante os testes.

Para identificar a causa do problema, foi aderida a sugestão do uso de técnicas XAI. Segundo a IBM, inteligência artificial explicável (XAI) é um conjunto de processos e métodos que permite aos usuários humanos entenderem e confiarem nos resultados e saídas criadas por algoritmos de aprendizado de máquina. IA explicável é usada para descrever um modelo de IA, seu impacto esperado e potenciais vieses. Para isso, foi utilizado a biblioteca Python SHAP.

A biblioteca SHAP (*SHapley Additive exPlanations*) é usada para interpretar modelos de machine learning, explicando a contribuição de cada característica nas previsões. Baseada no conceito de valores de Shapley da teoria dos jogos, SHAP calcula a influência de cada característica de forma justa e consistente, atribuindo a elas uma pontuação que indica seu impacto nas decisões do modelo. Isso é especialmente útil para modelos complexos, como árvores de decisão e redes neurais, ajudando a aumentar a interpretabilidade e transparência das previsões feitas por esses modelos, podendo nos dar mais clareza sobre os resultados e saídas do modelo.

A figura 20 é meramente ilustrativa, e não reflete o modelo, apenas mostra uma representação de como deve ser um gráfico normal do SHAP .

Figura 20: Ilustração do SHAP Beeswam (Ilustrativo)

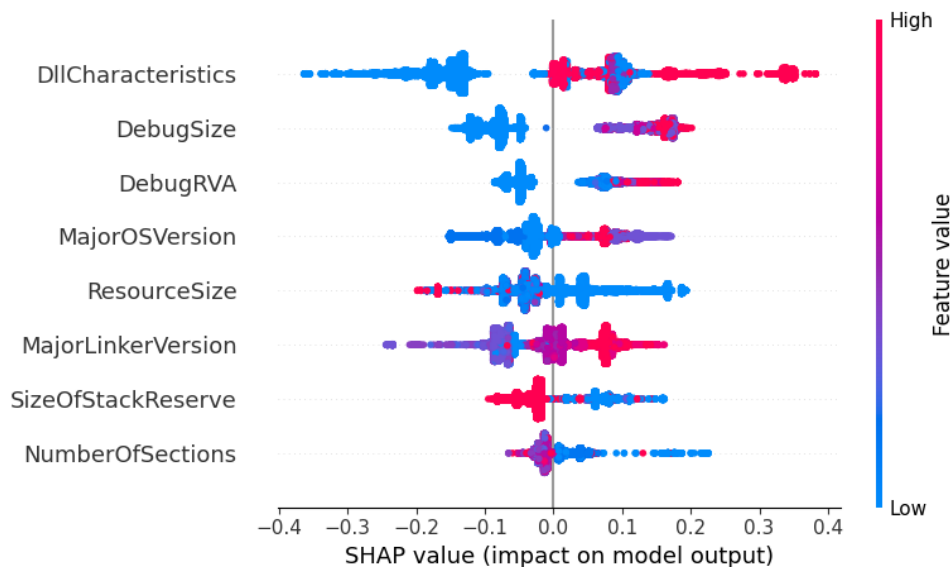


Fonte: Elaborado pelo autor.

O gráfico de valores SHAP mostra como cada característica influencia as previsões do modelo de aprendizado de máquina. No eixo vertical, temos as características, e no eixo horizontal, os valores SHAP, que representam o impacto de cada característica no resultado. Cada ponto indica um dado, com a cor variando do azul (valor baixo da característica) ao vermelho (valor alto da característica). Valores SHAP positivos indicam que a característica aumenta a previsão, enquanto valores negativos indicam o contrário.

Note que a distribuição dos pontos (pelo menos nas primeiras características) são bem definidas, tanto em cor quanto em espaçamento, isso indica que o modelo sabe bem como tratar cada dado e característica para o seu treinamento. É mostrado a seguir, na figura 21, qual foi o gráfico gerado pelo modelo real.

Figura 21: Ilustração do SHAP Beeswam (Real)



Fonte: Elaborado pelo autor.

Agora os pontos (dados) estão de certa forma aglomerados e com as cores "misturadas". Isso sugere que a característica tem um impacto considerável no modelo, mas de forma não linear. Nesse caso, tanto valores altos (vermelho) quanto baixos (azul) da característica podem ter efeitos variados no resultado, dependendo do contexto dos dados. Portanto, o efeito da característica nas previsões do modelo não segue uma relação simples, onde altos valores da característica não necessariamente aumentam ou diminuem a previsão de maneira consistente, e o mesmo vale para valores baixos. Isso pode dar pistas sobre o motivo do modelo apresentar uma alta precisão no conjunto de testes, e uma baixa precisão quando está lidando com dados reais. É possível então supor que o modelo está "confuso" e incerto sobre como utilizar as características, ou então, que o conjunto de dados é "ruim" e pode não refletir dados da vida real.



## 6 Conclusão

A presente pesquisa explorou o desenvolvimento e aplicação de técnicas de aprendizado de máquina para a detecção de *ransomwares*, uma das ameaças cibernéticas mais persistentes e complexas da atualidade. Em um cenário onde ataques de *ransomwares* evoluem rapidamente em termos de sofisticação e abrangência, a necessidade de soluções que sejam, ao mesmo tempo, precisas e proativas torna-se imperativa. O modelo proposto neste estudo, ao identificar comportamentos anômalos e padrões suspeitos de maneira eficaz, destaca-se como uma contribuição prática, reforçando a importância do uso de algoritmos de aprendizado de máquina no campo da cibersegurança.

Os resultados obtidos reforçam a robustez do modelo em diversos aspectos, especialmente na capacidade de identificar ameaças com precisão, o que possibilita uma resposta mais ágil e eficiente em ambientes vulneráveis. A avaliação minuciosa, baseada em métricas como acurácia e taxa de falso positivo, demonstrou que o modelo é competitivo e confiável, mesmo diante de variações nos padrões de comportamento de ataques. Ainda assim, as limitações observadas, como a necessidade de dados mais amplos e variados, refletem desafios inerentes ao campo, sugerindo que a eficácia do modelo poderia ser aprimorada com o uso de datasets mais ricos e confiáveis.

Adicionalmente, a pesquisa ressalta que o campo da detecção de *ransomware* é altamente dinâmico, o que requer constantes aprimoramentos para acompanhar as novas táticas de ataque que emergem de forma cada vez mais adaptativa e sofisticada. Recomenda-se, para estudos futuros, a exploração de algoritmos mais complexos, incluindo redes neurais profundas e técnicas de aprendizado não supervisionado, que poderiam ampliar ainda mais a capacidade do sistema em identificar comportamentos atípicos e desconhecidos. A integração com sistemas de monitoramento em tempo real também é indicada como um caminho promissor, fortalecendo a capacidade de resposta imediata e aumentando a resiliência contra incidentes críticos.

Em síntese, esta pesquisa reafirma a importância de soluções baseadas em aprendizado de máquina como resposta à crescente ameaça de *ransomware*. Acredita-se que as contribuições aqui realizadas servirão não apenas como base para futuras investigações, mas também como uma etapa significativa no desenvolvimento de defesas cibernéticas mais resilientes e adaptáveis, capazes de responder de forma eficiente ao complexo e mutável cenário das ameaças digitais. Futuros trabalhos podem focar na melhoria do modelo integrando características estáticas e comportamentais. As características estáticas, como tamanho de arquivos, entropia e bibliotecas usadas, seriam complementadas por características comportamentais, como padrões de acesso a arquivos, alterações no registro e conexões de rede. Para implementar essa abordagem, técnicas de engenharia de *features* seriam usadas para combinar e normalizar os diferentes tipos de dados, garantindo eficiência no treinamento do modelo. Além disso, métodos de otimização de hiperparâmetros, como busca em grade ou bayesiana, poderiam ser aplicados para ajustar parâmetros críticos, como o número de árvores e profundidade máxima. Avaliações do modelo seriam realizadas com conjuntos de dados diversos e em diferentes cenários, validando sua generalização. Ferramentas como SHAP, utilizadas neste trabalho, poderiam ser utilizadas para interpretar as decisões do modelo, destacando quais características tiveram maior impacto.

## Referências

ABOUD, Marah A.; MARIYAPPN, K. Investigation of Modern Ransomware Key Generation Methods: A Review. In: 2021 International Conference on Computer Communication and Informatics (ICCCI). [S.l.]: IEEE, 2021. Acessado em 20/10/2024. Disponível em: <<https://ieeexplore.ieee.org/document/9402680>>.

ASAJU, Christine Bukola et al. Development of a Machine Learning Model for Detecting and Classifying Ransomware. In: INTERNATIONAL Conference on Multidisciplinary Engineering and Applied Science (ICMEAS). [S.l.: s.n.], 2021. Acessado em 20/10/2024. Disponível em: <<https://ieeexplore.ieee.org/document/9692402>>.

BIAU, Gérard; SCORNET, Erwan. A Random Forest Guided Tour. **Test**, v. 25, n. 2, p. 197–227, 2016. Acessado em 20/10/2024. Disponível em: <<https://doi.org/10.1007/s11749-016-0481-7>>.

BREIMAN, Leo. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. Acessado em 20/10/2024. Disponível em: <<https://doi.org/10.1023/A:1010933404324>>.

CICALA, Francesco; BERTINO, Elisa. Analysis of Encryption Key Generation in Modern Crypto Ransomware. **IEEE Transactions on Dependable and Secure Computing**, 2020. Acessado em 20/10/2024. Disponível em: <<https://doi.org/10.1109/TDSC.2020.2999734>>.

GANTA, Venkata Gopi et al. Ransomware Detection in Executable Files Using Machine Learning. In: 2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT). [S.l.]: IEEE, 2020. P. 282–286. Acessado em 20/11/2024. DOI: 10.1109/RTEICT49044.2020.9315672. Disponível em: <[https://www.researchgate.net/publication/349171033\\_Ransomware\\_Detection\\_in\\_Executable\\_Files\\_Using\\_Machine\\_Learning](https://www.researchgate.net/publication/349171033_Ransomware_Detection_in_Executable_Files_Using_Machine_Learning)>.

GENÇ, Zekeriya; BAJPAI, Prashant; ENBODY, Richard. Ransomware Key Generation Management in Cryptosystems. **Journal of Cybersecurity**, 2021. Acessado em 20/10/2024. Disponível em: <<https://academic.oup.com/cybersecurity/article/7/1/tyab023/6190162>>.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. [S.l.]: O'Reilly Media, 2019. Acessado em 20/10/2024. Disponível em: <<https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>>.

HSU, Chia-Ming et al. Enhancing File Entropy Analysis to Improve Machine Learning Detection Rate of Ransomware. **IEEE Access**, IEEE, v. 9, p. 138345–138351, 2021. Acessado em 20/11/2024. DOI: 10.1109/ACCESS.2021.3114148. Disponível em: <<https://www.semanticscholar.org/paper/Enhancing-File-Entropy-Analysis-to-Improve-Machine-Hsu-Yang>>.

MOUHA, Nicky. Review of the Advanced Encryption Standard. **National Institute of Standards and Technology**, NIST, v. 1210, p. 1–23, 2021. Acessado em 20/11/2024. Disponível em: <<https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8319.pdf>>.

PONTE, Caio; CAMINHA, Carlos; FURTADO, Vasco. Otimização de Florestas Aleatórias através de ponderação de folhas em árvore de regressão. In: ANAIS do XVII Encontro Nacional de Inteligência Artificial e Computacional. Evento Online: SBC, 2020. P. 698–708. Acessado em 18/11/2024. DOI: 10.5753/eniac.2020.12171. Disponível em: <<https://sol.sbc.org.br/index.php/eniac/article/view/12171>>.

RIVEST, Ronald L; SHAMIR, Adi; ADLEMAN, Leonard. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, ACM New York, NY, USA, v. 21, n. 2, p. 120–126, 1978. Acessado em 19/11/2024. Disponível em: <<https://dl.acm.org/doi/pdf/10.1145/359340.359342>>.

USHA, G. et al. Enhanced Ransomware Detection Techniques using Machine Learning Algorithms. In: 2021 4th International Conference on Computing and Communications Technologies (ICCCT). [S.l.]: IEEE, 2021. P. 52–58. Acessado em 20/11/2024. DOI: 10.1109/ICCCT53315.2021.9711906. Disponível em: <<https://ieeexplore.ieee.org/document/9711906>>.