UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ABORDAGEM INTELIGENTE COM COMBINAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS PARA DETECÇÃO DE NOVAS FAMÍLIAS DE *RANSOMWARE*

CAIO CARVALHO MOREIRA

TD 04/2024

BELÉM-PA
2024

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CAIO CARVALHO MOREIRA

ABORDAGEM INTELIGENTE COM COMBINAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS PARA DETECÇÃO DE NOVAS FAMÍLIAS DE *RANSOMWARE*

Tese de Doutorado submetida à Banca Examinadora do Programa de Pós-Graduação em Engenharia Elétrica como requisito para obtenção do Grau de Doutor em Engenharia Elétrica na área de Computação Aplicada.

Orientador: Prof. Dr. Claudomiro de Souza de Sales Júnior

TD 04/2024

BELÉM-PA
2024

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

M835a Moreira, Caio Carvalho.
Abordagem Inteligente com Combinação de Características Estruturais para Detecção de Novas Famílias de Ransomware / Caio Carvalho Moreira. — 2024.
91 f. : il. color.

Orientador(a): Prof. Dr. Claudomiro de Souza de Sales Júnior
Tese (Doutorado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2024.

1. Segurança Cibernética. 2. Aprendizado de Máquina. I. Título.

CDD 006.31

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"ABORDAGEM INTELIGENTE COM COMBINAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS PARA DETECÇÃO DE NOVAS FAMÍLIAS DE RANSOMWARE"

AUTOR: **CAIO CARVALHO MOREIRA**

TESE DE DOUTORADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE DOUTOR EM ENGENHARIA ELÉTRICA NA ÁREA DE COMPUTAÇÃO APLICADA.

APROVADA EM: 22/03/2024

BANCA EXAMINADORA:

Documento assinado digitalmente
gov.br
CLAUDOMIRO DE SOUZA DE SALES JUNIOR
Data: 26/03/2024 09:45:06-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Claudomiro de Souza de Sales Júnior
(Orientador – PPGEE/UFPA)

Documento assinado digitalmente
gov.br
ALDEBARO BARRETO DA ROCHA KLAUTAU JUN
Data: 27/03/2024 12:43:41-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior
(Avaliador Interno - PPGEE/UFPA)

Documento assinado digitalmente
gov.br
OTAVIO NOURA TEIXEIRA
Data: 30/03/2024 14:35:28-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Otávio Noura Teixeira
(Avaliador Externo ao Programa - PPCA/NDAE/UFPA)

Documento assinado digitalmente
gov.br
REGINALDO CORDEIRO DOS SANTOS FILHO
Data: 26/03/2024 17:16:48-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Reginaldo Cordeiro dos Santos Filho
(Avaliador Externo ao Programa - PPGCC/UFPA)

Documento assinado digitalmente
gov.br
BRUNO BOGAZ ZARPELAO
Data: 28/03/2024 13:11:46-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Bruno Bogaz Zarpelão
(Avaliador Externo - UEL)

VISTO:

Prof. Dr. Diego Lisboa Cardoso
(Coordenador do PPGEE/ITEC/UFPA)

Dedico esta tese ao meu filho, Samuel, cuja presença encheu minha jornada de alegria e motivação, e à minha filha, Clarice, cuja chegada antecipo com amor e ansiedade.

AGRADECIMENTOS

Agradeço, primeiramente, à minha esposa Laís, companheira de vida e fonte inesgotável de amor, apoio e compreensão. Obrigado por estar ao meu lado por todos esses anos.

Aos meus pais, Huldaci e Pedro, e meus irmãos Pedro Igor e Davi, expresso minha gratidão. Vocês são o alicerce dos valores que carrego.

Ao Prof. Dr. Claudomiro Sales Jr., sou grato por sua orientação sábia, paciência e disponibilidade, bem como por ter confiado no potencial do meu trabalho.

A todos que contribuíram de alguma forma para a realização desta jornada acadêmica de pesquisa e descoberta, meus sinceros agradecimentos.

Este trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

RESUMO

Ransomware é um *software* malicioso que tem como objetivo criptografar os arquivos do usuário e exigir um resgate para desbloqueá-los. Trata-se de uma ameaça cibernética que pode causar significativos danos financeiros, além do comprometimento de privacidade e integridade dos dados. Embora os *scanners* de detecção baseados em assinaturas comumente combatam essa ameaça, eles falham na identificação de famílias (variantes) desconhecidas de *ransomware*. Um método para detectar novas ameaças sem a necessidade de executá-las é a análise estática, que inspeciona o código e a estrutura do *software*, juntamente com a classificação através de abordagens inteligentes. A Detecção de Novas Famílias de *Ransomware* (DNFR) pode ser avaliada em um cenário realista e desafiador pela categorização e isolamento de famílias para treinamento e teste. Desta forma, o objetivo desta tese é desenvolver um modelo eficaz de análise estática para a DNFR, que pode ser aplicado em sistemas *Windows* como uma camada adicional de segurança para verificar os arquivos executáveis no momento do recebimento ou antes de sua execução. A detecção precoce do *ransomware* é fundamental para reduzir a probabilidade de um ataque bem-sucedido. A abordagem proposta analisa abrangentemente os binários executáveis, ao extrair e combinar diversas características estruturais, e os distingue entre *ransomware* ou *software* benigno empregando um modelo de votação suave que compreende três técnicas de Aprendizado de Máquina: *Logistic Regression* (LR), *Random Forest* (RF) e *eXtreme Gradient Boosting* (XGB). Os resultados para a DNFR demonstraram médias de 97,53% de acurácia, 96,36% de precisão, 97,52% de *recall* e 96,41% de *F-measure*. Além disso, a varredura e a predição de amostras individuais levaram uma média de 0,37 segundos. Essa performance indica sucesso na identificação rápida de variantes desconhecidas de *ransomware* e na adaptabilidade do modelo ao cenário em constante evolução, o que sugere sua aplicabilidade em sistemas de proteção antivírus, mesmo em dispositivos com recursos limitados. Portanto, o método oferece vantagens significativas e pode ajudar desenvolvedores de sistemas de detecção de *ransomware* na criação de soluções mais resilientes, confiáveis e com rápido tempo de resposta.

Palavras-chaves: Detecção de *Ransomware*, Detecção de Dia Zero, Análise de *Features*, Aprendizado de Máquina, Segurança Cibernética.

ABSTRACT

Ransomware is a malicious software that aims to encrypt user files and demand a ransom to unlock them. It is a cyber threat that can cause significant financial damage, as well as compromise privacy and data integrity. Although signature-based detection scanners commonly combat this threat, they fail to identify unknown ransomware families (variants). One method to detect new threats without the need to execute them is static analysis, which inspects the code and structure of the software, along with classification through intelligent approaches. The Detection of New Ransomware Families (DNFR) can be evaluated in a realistic and challenging scenario by categorizing and isolating families for training and testing. Hence, this thesis aims to develop an effective static analysis model for DNFR, which can be applied in Windows systems as an additional security layer to check executable files upon receipt or before execution. Early ransomware detection is essential to reduce the likelihood of a successful attack. The proposed approach comprehensively analyzes executable binaries, extracting and combining various structural features, and distinguishes them between ransomware or benign software employing a soft voting model comprising three machine learning techniques: Logistic Regression (LR), Random Forest (RF), and eXtreme Gradient Boosting (XGB). Results for DNFR demonstrated an average accuracy of 97.53%, precision of 96.36%, recall of 97.52%, and F-measure of 96.41%. Additionally, scanning and predicting individual samples took an average of 0.37 seconds. This performance indicates success in quickly identifying unknown ransomware variants and adapting the model to the constantly evolving landscape, suggesting its applicability in antivirus protection systems, even on resource-limited devices. Therefore, the method offers significant advantages and can assist developers of ransomware detection systems in creating more resilient, reliable, and fast-response solutions.

Keywords: Ransomware Detection, Zero-day Detection, Feature Analysis, Machine Learning, Cybersecurity.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BD	Banco de Dados
C&C	Comando & Controle
CFG	*Control-Flow Graph*
CNN	*Convolutional Neural Network*
COFF	*Common Object File Format*
CPU	*Central Processing Unit*
CV	*Cross-Validation*
CSV	*Comma-Separated Values*
DC	Distância Crítica
DL	*Deep Learning*
DNFR	Detecção de Novas Famílias de *Ransomware*
DLL	*Dynamic-Link Library*
EL	*Ensemble Learning*
EXT	*Extra-Trees*
FN	*False Negative*
FP	*False Positive*
Grad-CAM	*Gradient-weighted Class Activation Mapping*
GUI	*Graphical User Interface*
IC	Intervalo de Confiança
KNN	*K-Nearest Neighbors*
LDA	*Linear Discriminant Analysis*
LSTM	*Long Short Term Memory*

LR	*Logistic Regression*
LOF	*Local Outlier Factor*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
PCA	*Principal Component Analysis*
PE	*Portable Executable*
PLN	Processamento de Linguagem Natural
RF	*Random Forest*
RaaS	*Ransomware-as-a-Service*
RAM	*Random Access Memory*
RGB	*Red, Green and Blue*
SA-CNN	*Self-Attention - Convolutional Neural Network*
SGD	*Stochastic Gradient Descent*
SHAP	*SHapley Additive exPlanations*
SNN	*Siamese Neural Network*
SO	Sistema Operacional
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency - Inverse Document Frequency*
TN	*True Negative*
TP	*True Positive*
VT	*Variance Threshold*
XGB	*eXtreme Gradient Boosting*

LISTA DE SÍMBOLOS

S	Entropia de Shannon
y	Evento aleatório discreto
$P(.)$	Probabilidade
u	Unidade de informação
U	Número de símbolos
N	Número de elementos em um N-grama
n	Frequência de um N-grama em uma determinada sequência
i	Índice de N-grama
j	Sequência de *opcodes* de N-grama
k	Índice de N-grama na sequência j
D	Conjunto de todas as sequências de N-gramas
$Var(.)$	Variância
X	Conjunto de valores para uma determinada *feature*
m	Tamanho da X
a	Índice da observação
x	Valor da observação
p	Probabilidade de sucesso
F_1	*F-measure* balanceada
β	Fator real positivo da relação *recall*/precisão
q_α	Valor crítico
c	Número de grupos
M	Número de métricas em c
H_0	Hipótese nula
H_1	Hipótese alternativa

$\phi(.)$	Valor de Shapley
g	Índice de jogador
G	Número de jogadores
C	Subconjunto de combinações que não incluem o jogador g
$\hat{\phi}$	Amostragem de Monte-Carlo para $\phi(.)$
B	Número de amostras Monte-Carlo
b	Índice da amostra em B
$\hat{f}$	Predição do modelo
w	Subconjunto de *features*
l	*Feature* em w

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Motivação

A ampla disseminação da Internet tem propiciado a realização remota de uma variedade de atividades, como transações bancárias, comércio eletrônico, contratação e prestação de serviços, ensino a distância, trabalho remoto, consultas de saúde, entre outras, por meio de computadores pessoais ou dispositivos móveis, como *smartphones*. Essa transformação digital tem alterado significativamente a forma como as pessoas interagem com empresas e instituições, pois possibilita um acesso conveniente a uma vasta gama de serviços e oportunidades sem a necessidade de presença física. Contudo, sistemas computacionais são diariamente ameaçados por programas especificamente desenvolvidos para executar ações danosas e atividades maliciosas. Este tipo de *software* nocivo, denominado *malware*, é normalmente usado por criminosos para lançar ataques cibernéticos contra os dispositivos alvo (OrMeir et al., 2019; Alenezi et al., 2020).

1.1.1 *Malwares*

Malware é qualquer *software* que executa intencionalmente códigos maliciosos em dispositivos ou estruturas de destino, tais como computadores, *smartphones* e redes de computadores. Estes programas maliciosos possuem uma variedade de formas, tipicamente agrupados em famílias ou tipos, como vírus, *worms*, *keyloggers*, cavalos de Tróia, *rootkits* e *ransomwares*. Uma família de *malware* é uma coleção de amostras com a mesma base de código. Consequentemente, cada família de *malware* possui propriedades distintas que as diferenciam umas das outras, o que acarreta em impactos variados nos dispositivos das vítimas. Esses efeitos podem incluir danos ao sistema, possibilidade de execução de código remoto, roubo de dados confidenciais, monitoramento das atividades do usuário, exibição de propagandas indesejadas ou utilização silenciosa dos recursos computacionais (OrMeir et al., 2019; Aslan; Samet, 2020).

Os primeiros códigos de *malware* foram inicialmente desenvolvidos com propósitos simples e específicos. No entanto, ao longo do tempo, esses códigos evoluíram, tornando-se mais sofisticados ao buscar compatibilidade com múltiplas plataformas, aprimorar habilidades para executar vários processos simultaneamente, sofrer mutações, se auto-replicar em outros dispositivos, ocultar a sua presença e persistir nos sistemas. Essas evoluções recentes, além de aumentarem a periculosidade dos ataques, também dificultam a detecção por meio de estratégias tradicionais baseadas em informações previamente conhecidas (Milošević, 2013; Alenezi et al., 2020; M.; Sethuraman, 2023; Ling et al., 2023).

Nos sistemas finais (*hosts*), os *scanners* de *malware* são amplamente utilizados como mecanismo de defesa contra ameaças de segurança, apesar de suas limitações e fragilidades

conhecidas. Tanto indivíduos quanto empresas adotam esses *softwares* antivírus em seus sistemas de *desktop* para mitigar ataques. Portanto, milhões de *hosts* executam esses *scanners* em busca de arquivos infectados e códigos maliciosos. A eficácia e a prevalência desses produtos baseiam-se, em grande parte, em sua funcionalidade simples, porém confiável: eles realizam comparações entre dados extraídos dos arquivos suspeitos e um Banco de Dados (BD) local contendo padrões de detecção conhecidos, também chamados de assinaturas. Essas assinaturas são periodicamente atualizadas pelos fornecedores de soluções de segurança para incluir proteções contra ameaças emergentes (Wressnegger et al., 2017; Alsmadi; Alqudah, 2021).

A abordagem de correspondência de padrões empregada pelos *scanners* de *malware* nos sistemas finais é altamente eficaz para identificar uma ampla variedade de ameaças, desde que as assinaturas adequadas e atualizadas estejam disponíveis. No entanto, a detecção baseada em assinaturas apresenta uma desvantagem significativa: a incapacidade de identificar ameaças desconhecidas para as quais não existem assinaturas. Essas ameaças podem passar despercebidas pelos *scanners*, o que é agravado pelo uso frequente de técnicas de ofuscação em códigos maliciosos que dificultam a correspondência baseada em assinaturas (Wressnegger et al., 2017; Alsmadi; Alqudah, 2021).

As empresas líderes no desenvolvimento de soluções antivírus enfrentam um grande desafio diante do volume significativo de novas amostras que recebem diariamente para análise. Dada a quantidade de submissões, a análise manual se torna inviável em termos de tempo e recursos. Assim, é imprescindível automatizar o processo de análise dessas amostras. Nesse sentido, as amostras submetidas são inicialmente avaliadas por meio de técnicas de correspondência de assinaturas, em que, caso haja correspondência, a amostra é identificada como maliciosa. Caso contrário, analisa-se automaticamente com *softwares* especializados que buscam identificar padrões comportamentais e estruturais maliciosos presentes nas amostras (Wang; Cong; Yu, 2020; Aslan; Samet, 2020).

Além disso, mesmo após a identificação de um novo *malware* por uma central de análises, é necessário gerar uma nova assinatura e distribuí-la para os clientes a fim de proteger os sistemas finais. Portanto, mesmo com um antivírus atualizado, caso determinado *host* seja o primeiro alvo de uma ameaça desconhecida, há uma probabilidade significativa de danos ao cliente (Moussaileb et al., 2021).

1.1.2 *Ransomwares*

Embora os diversos *malwares* sejam, por definição, prejudiciais aos sistemas computacionais, um dos seus tipos mais destrutivos é o *ransomware*. Esse *software* malicioso é usado para extorsão digital. Na maioria dos casos, ele bloqueia o acesso aos arquivos do usuário por meio de criptografia até que a vítima pague um resgate em criptomoedas para obter a chave de descriptografia válida. Esta chave é necessária para a recuperação dos arquivos, mas receber a chave do criminoso não é garantido mesmo após o pagamento do resgate (Beaman et al., 2021;

Kapoor et al., 2022).

Nos últimos anos, empresas e instituições governamentais necessitaram que os colaboradores se conectassem remotamente às redes corporativas. Como resultado, vários aspectos das práticas de segurança organizacional precisaram ser flexibilizados a requisitos de ambientes não supervisionados e não controlados (Georgiadou; Mouzakitis; Askounis, 2022). Para gerentes de segurança de Tecnologia da Informação, é uma situação alarmante ter os sistemas finais de suas redes como possíveis pontos de entrada para *ransomwares* disseminados pela Internet. A probabilidade de ocorrência de uma violação de segurança deixou de ser remota e se tornou quase certa (Lang et al., 2023).

Os criminosos de *ransomware* também passaram a adotar táticas perversas de dupla extorsão para maximizar seu retorno ao pedir resgate não apenas para a recuperação dos dados, mas também para o seu não vazamento. Os ataques de *ransomware* têm demonstrado sua natureza perigosa, causando perdas financeiras consideráveis, interrupções operacionais e comprometimento da privacidade de indivíduos e empresas (Iacono; Wojcieszek; Glass, 2022).

Devido à natureza altamente lucrativa do *ransomware*, esses ataques evoluem constantemente para contornar os mecanismos de proteção atuais, fortalecer seu processo de criptografia e melhorar suas técnicas de difusão (Kolodenker et al., 2017). Além disso, o modelo de negócio denominado *Ransomware-as-a-Service* (RaaS) possibilitou a compra e venda desses *malwares* nos mercados *darknet*, com a produção de *ransomwares* personalizados com a mesma base de código, também conhecidos como famílias, subfamílias, cepas ou variantes. Isso permitiu que criminosos cibernéticos e usuários não técnicos maliciosos participem dessa indústria antiética (Meland; Bayoumy; Sindre, 2020). Ou seja, essa ameaça se desenvolveu em um grau tão perigoso que estabeleceu um mecanismo cooperativo entre criminosos tecnicamente qualificados ou não. Portanto, é possível inferir que o aumento desse tipo de *malware* continuará uma realidade no mundo cibernético nos próximos anos.

Para examinar a estrutura e o comportamento dos *malwares* e, consequentemente, dos *ransomwares*, existem dois tipos principais de análise: dinâmica e estática. Embora ambas as abordagens tenham vantagens e limitações, concentram-se em aspectos distintos e fornecem tipos diferentes de dados. Adicionalmente, elas podem ser combinadas em abordagens híbridas, o que permite usufruir dos benefícios de ambas, por exemplo, para aprimorar a capacidade geral de detecção e reduzir o risco de um ataque bem-sucedido (Ferrante et al., 2018; Oz et al., 2022).

A análise dinâmica investiga as características (*features*) comportamentais de amostras de *software* em execução. As características comportamentais referem-se às ações e atividades realizadas pelo *software* quando ele é executado em um sistema. Na análise dinâmica, o programa em investigação é executado em um ambiente isolado e o seu comportamento é monitorado a fim de identificar padrões de ações maliciosas no Sistema Operacional (SO). Durante a execução de uma amostra, a análise dinâmica pode obter valiosas características comportamentais, por exemplo: chamadas de *Application Programming Interface* (API) e de sistema

realizadas; bibliotecas carregadas; arquivos, diretórios e registros do sistema alterados; operações de memória principal e secundária; tráfego de rede e utilização de *Central Processing Unit* (CPU) (Luckett et al., 2018; Sihwail et al., 2019; Singh; Singh, 2021; Urooj et al., 2022).

Com base nessas observações de comportamento, é possível determinar se a amostra analisada é maliciosa ou não. A análise dinâmica tem como vantagens dificultar a implementação de técnicas de evasão e permitir que pesquisadores investiguem e entendam como esses programas perigosos operam no SO. Entretanto, essa abordagem tipicamente é mais lenta e perigosa (Torres; Yoo, 2017; Kok et al., 2019; Oz et al., 2022). Além disso, esse tipo de análise não é adequada para a detecção de ameaças em tempo de execução em sistemas finais.

Por outro lado, a análise estática inspeciona as características estruturais do *software* suspeito sem executá-lo. As características estruturais referem-se aos atributos do código binário e às propriedades do *sofware*. A análise estática proporciona aos pesquisadores uma compreensão do conteúdo e da estrutura da amostra, permitindo-lhes identificar instruções potencialmente maliciosas em estágios iniciais. A detecção precoce do *malware* é crucial para minimizar seu impacto e reduzir a probabilidade de um ataque bem-sucedido. Esse método de análise é geralmente caracterizado por sua velocidade e segurança em comparação com outros métodos de análise. Todavia, essa abordagem é mais suscetível a ser enganada por técnicas de ofuscação, polimorfismo, criptografia e antidesmontagem (Oz et al., 2022; Cen et al., 2024).

Ao realizar a análise estática de um arquivo binário, é possível identificar suas características estruturais por meio de diferentes elementos como dados do cabeçalho, códigos de operação (*opcodes*), sequência de *bytes*, *Dynamic-Link Libraries* (DLLs) e funções importadas, entropia, valores de *hash*, *Control-Flow Graph* (CFG) e textos em ASCII embutidos. Diversos *softwares* de detecção de *malware* são desenvolvidos com o uso desses recursos (Ucci; Aniello; Baldoni, 2019; Singh; Singh, 2021).

Considerando que os *ransomwares* geralmente são disseminados por meio de *e-mails* de *phishing* e *downloads* de sites maliciosos, a análise estática pode ser automatizada localmente nos sistemas finais. Quando uma possível ameaça é identificada pela análise estática, ela pode ser bloqueada antes da execução e, posteriormente, submetida a uma análise mais aprofundada no servidor central da solução antivírus (Verma et al., 2018; Preuveneers; Joosen, 2021; Moussaileb et al., 2021). Portanto, a análise estática pode ser aplicada em sistemas finais para detectar *ransomwares* antes de suas execuções, consequentemente, antes que tenham a oportunidade de causar qualquer tipo de dano.

Tanto a análise estática quanto a dinâmica desempenham um papel fundamental na detecção de *ransomware*. Para a automatização dessas análises, a aplicação de abordagens inteligentes tem sido amplamente adotada. Geralmente, esse processo envolve a extração de características estruturais ou comportamentais, seguida de classificação por meio de algoritmos de *Machine Learning* (ML) e *Deep Learning* (DL). No entanto, enfrentam-se desafios significativos na detecção eficaz de *ransomwares* desconhecidos, o que resulta em taxas consideráveis

de falsos positivos e falsos negativos nas abordagens de detecção existentes (Moussaileb et al., 2021; Guo, 2023).

Uma amostra de *ransomware* desconhecida ou recém-descoberta é conhecida como de dia zero. Muitos desses *ransomwares* se originam do modelo de negócio RaaS com amostras similares, o que forma uma família de *ransomware* de dia zero. Essas famílias são particularmente perigosas porque as defesas de segurança, sejam centrais ou locais, ainda não estão preparadas para enfrentá-las. Isso ocorre porque não há atualizações de segurança disponíveis que contenham suas assinaturas para detectá-las no momento da descoberta (Meland; Bayoumy; Sindre, 2020; Moussaileb et al., 2021; Guo, 2023).

Relatórios de empresas de segurança cibernética apontam que o SO *Windows* corresponde a mais de 93% dos alvos de *ransomware*, em comparação com 2% para o Android e aproximadamente 5% para outros sistemas (VIRUSTOTAL, 2021; AAG-IT, 2024). Desta forma, soluções de proteção para sistemas *Windows* são essenciais para mitigar a imensa maioria dessa ameaça.

Diante da alta periculosidade do *ransomware*, das limitações dos métodos atuais de detecção e da rápida proliferação de novas famílias devido ao RaaS, surge uma demanda urgente por uma nova abordagem que possa detectar efetivamente famílias de *ransomware* de dia zero antes da execução, que operem localmente, a fim de garantir a proteção dos clientes, mesmo quando são os primeiros alvos de um ataque. No entanto, são escassos os estudos que se concentram na Detecção de Novas Famílias de *Ransomware* (DNFR), pois é comum na literatura incluir diferentes amostras de uma mesma família nos conjuntos de treinamento e teste.

Consequentemente, são necessários estudos que transitem dos modelos tradicionais baseados em assinaturas e das formas comuns de teste para técnicas e metodologias mais avançadas que possam identificar com precisão novas variantes de *ransomware* antes de sua execução no *host*, de forma rápida e sem comprometer o desempenho do sistema.

É possível avaliar a DNFR por meio de uma metodologia que consiste em isolar famílias de *ransomware* para fins de treinamento e teste. Essa abordagem metodológica apresenta desafios significativos para os modelos de detecção, uma vez que permite simular cenários mais realistas de aplicação. A premissa subjacente a essa abordagem é que novas variantes de *ransomware* geralmente compartilham características semelhantes com anteriores. Deste modo, pode-se desenvolver modelos de detecção focados em identificar efetivamente novas famílias de *ransomware*, mesmo na ausência de observações prévias dessas variantes.

Ainda, destaca-se que a maioria dos estudos existentes não investigou a fundamentação por trás das decisões tomadas pelos modelos de detecção de *ransomware*. Assim, novos estudos são necessários para melhor compreender as características estruturais do *ransomware*, estabelecer confiança nas decisões tomadas pelo modelo e auxiliar desenvolvedores de sistemas *antimalware* na construção de soluções mais robustas.

Esta tese propõe um modelo que aprimora a DNFR em estágio pré-execução ao empregar uma análise estática abrangente dessas ameaças. Essa análise é realizada pela combinação de diversas características dos arquivos executáveis, como campos de cabeçalho, DLLs importadas, chamadas de função e entropia de seções. A abordagem abrangente possibilita a identificação de padrões que efetivamente distinguem *ransomware* de *goodware* por meio de diversas técnicas de ML e DL.

Esse modelo é aplicável como uma camada adicional de segurança em sistemas de proteção *antimalware* que verifica cada arquivo executável quando *downloads* e anexos são recebidos ou antes de sua execução. A vantagem essencial desta abordagem sobre outros métodos de análise estática é que ela examina o *software* suspeito em vários aspectos de sua estrutura e conteúdo de código. Este nível de análise ajuda a identificar características exclusivas do *ransomware*, mesmo que a família específica seja desconhecida, como será demonstrado nos resultados desta tese.

Além disso, a análise estática abrangente é menos propensa a ser enganada por métodos de ofuscação, criptografia e antidesmontagem, porque é um desafio para desenvolvedores mal-intencionados aplicar essas técnicas de evasão em várias características estruturais ao mesmo tempo (Garcia; Hammad; Malek, 2018; Wu; Zhu; Liu, 2021). Portanto, a abordagem proposta nesta tese é mais resiliente contra famílias de *ransomware* novas e em evolução que usam essas técnicas.

1.2 Trabalhos Relacionados

Esta seção revisa o estado da arte relacionado à detecção de *malware*, especialmente de *ransomware*, ao utilizar ML e DL com análise estática para arquivos executáveis em sistemas *Windows*. Além disso, aborda estudos de análise dinâmica ou híbrida que se concentram na DNFR.

Pesquisas recentes têm demonstrado avanços na detecção de *malware* por meio da análise de características estruturais. Tipicamente, essa abordagem envolve a extração de características do código de *malware* e a classificação por meio de algoritmos de ML e DL, como observado nos estudos de Rezaei, Manavi e Hamzeh (2021), He, Yu e Song (2022), Rizvi et al. (2022), Paik, Jin e Cho (2022) e Yousuf et al. (2023). Os resultados desses trabalhos são promissores, com boas taxas de detecção. No entanto, esses estudos não se concentraram em tipos específicos de *malware*, o que pode resultar em investigações menos precisas. Tipos comuns de *malware* geralmente visam o comprometimento funcional do sistema e a exploração de vulnerabilidades para atividades de intrusão e monitoramento. Por outro lado, *ransomware* tem como alvo as informações contidas nos arquivos do usuário, as quais têm grande valor para indivíduos e organizações devido à sua natureza intangível. Portanto, soluções especializadas para a detecção de *ransomware* são essenciais, dada a significativa ameaça de perda de arquivos.

Hanqi Zhang et al. (2019) conduziram experimentos baseados em N-gramas de *opcodes* com *Term Frequency - Inverse Document Frequency* (TF-IDF) para classificações de *ransomware*. Para a formação do conjunto de dados, o estudo coletou 1787 amostras de oito famílias de *ransomware* e 100 amostras benignas. O estudo variou os vetores de N-gramas ($N = 2$, 3 e 4) e a quantidade de *features* após o cálculo do TF-IDF. No experimento de classificação binária, o melhor resultado foi uma acurácia de 99,3% e *recall* de 99,8% com o classificador *Random Forest* (RF), quando $N = 3$ e quantidade de 180 *features*. Entretanto, a pesquisa se absteve de apresentar outras métricas, o que prejudica a avaliação e comparação da eficácia do método, especialmente em conjuntos de dados desbalanceados entre as classes alvo.

Bin Zhang et al. (2020) propuseram um modelo de análise estática baseada em N-gramas de *opcodes* com DL. O conjunto de dados utilizado consistia em 1787 amostras de oito famílias de *ransomware* e 100 amostras benignas. A abordagem dos autores realiza a conversão do arquivo executável em uma sequência de *opcodes*, transformando-os em N-gramas e aplicando um filtro TF-IDF. Para lidar com diferentes comprimentos de sequências de *opcode*, eles dividiram as sequências em vários *patches* e utilizaram um mecanismo de Auto-Atenção com *Convolutional Neural Network* (CNN), denominado *Self-Attention - Convolutional Neural Network* (SA-CNN), em cada *patch*. As saídas dos SA-CNNs foram concatenadas e inseridas em uma rede de Auto-Atenção bidirecional para a classificação de *ransomware*. Os resultados atingiram 89,50% de acurácia, 87,50% de precisão, 87,60% de *recall* e 87,30% de *F-measure*.

Khammas (2020) propôs um método de detecção de *ransomware* com extração direta de características do binário em forma de sequência de bytes brutos. O estudo reuniu 840 amostras de *ransomware*, divididas em três famílias, e 840 amostras de *goodware*. Nesse método, é desnecessária a desmontagem do binário para a obtenção de suas características. O pré-processamento de todo o conjunto de amostras foi realizado em três etapas. Na primeira, foi realizada a extração de sequência de bytes brutos em forma de N-grama de 32-bits, onde $N = 4$. Na segunda, uma técnica de mineração de padrões frequentes foi utilizada para identificar itens de 4-gramas que se sobressaíram nos dados. Na terceira, um processo de normalização dos padrões de frequência obtidos foi aplicado. Em seguida, o método supervisionado *Gain Ratio* foi usado para seleção das *features* (4-gramas) mais significativas, que mostrou que 1000 características era o número ideal para o processo de detecção. A metodologia utilizada no experimento dividiu, após o pré-processamento e a seleção de *features*, o conjunto de dados em 50% para treino e 50% para teste. O estudo utilizou RF como classificador e atingiu 97,74% de acurácia, 95,87% de precisão, 99,80% de *recall* e 97,80% de *F-measure* para distinção de *ransomware* e *goodware*.

Stiawan et al. (2021) utilizaram CFG como técnica de análise estática para a extração do comportamento dos códigos de operação. O estudo utilizou 3000 amostras de *ransomware* e 3000 amostras benignas. No método proposto, a sequência de *opcodes* foi construída pela análise de CFG, em seguida, foram extraídos os N-gramas para $N = 1$, 2, 3 e 4, os quais foram

utilizados os dez de maior frequência para cada N. Os N-gramas foram convertidos em números para compor os vetores de entrada para o algoritmo classificador *K-Nearest Neighbors* (KNN). Os melhores resultados foram para $N = 1$, que atingiram uma acurácia de 98,86%, precisão de 100%, *recall* de 98,00% e *F-measure* de 98,98%.

Manavi e Hamzeh (2021) construíram uma rede *Long Short Term Memory* (LSTM) para processar a sequência de bytes que constrói o cabeçalho para distinguir as amostras de *ransomware* e goodware. Um dos conjuntos de dados utilizado foi composto de 1000 amostras de *ransomware* e 1000 de *goodware* coletadas pelos autores. Os autores sugeriram uma estratégia que usava duas unidades LSTM e uma camada totalmente conectada para treinar o melhor modelo com o menor número de camadas possível para evitar o *overfitting*. O estudo atingiu uma acurácia de 93,25%, precisão de 93,33%, *recall* de 93,25% e *F-measure* de 93,24%.

Manavi e Hamzeh (2022a) apresentaram um método baseado em incorporação de grafos para detectar *ransomware*. Um conjunto de dados que eles trabalharam foi o mesmo da pesquisa anterior dos autores (Manavi; Hamzeh, 2021). A abordagem proposta usou os cabeçalhos dos binários para formar um gráfico que foi mapeado em um auto-espaço pelo método *Power Iteration*. Por sua vez, o mapeamento resultante foi convertido em um vetor de *features* usado para treinar um classificador RF. O estudo alcançou uma acurácia de 93,30%, precisão de 93,48%, *recall* de 93,30% e *F-measure* de 93,28%.

Outro estudo de Manavi e Hamzeh (2022b) propôs um método de análise estática para detectar *ransomware* com CNN em que os principais passos eram extrair o cabeçalho dos arquivos executáveis, construir uma imagem em escala de cinza em um padrão de zigue-zague, treinar seu próprio modelo CNN com base nas imagens e testar o modelo com amostras não vistas. Um conjunto de dados que eles trabalharam foi o mesmo da pesquisa anterior dos autores (Manavi; Hamzeh, 2021). Na nova pesquisa, os autores criaram um modelo sequencial de CNN com 12 camadas de profundidade, imagens com dimensão de entrada de $32 \times 32 \times 1$ e treinamento por 100 épocas para atingir acurácia de 93,33%, precisão de 93,40%, *recall* de 93,33% e *F-measure* de 93,34%.

Zhu et al. (2022) propuseram uma *Siamese Neural Network* (SNN) baseada em meta-aprendizagem que classifica diferentes famílias de *ransomware*. O estudo coletou 1046 amostras de *ransomware*, divididas em 11 famílias. No método proposto, a entropia dos binários foi extraída e transformada em gráficos para serem utilizados como dados de entrada em cada sub-rede da SNN, que foi composta por dois modelos pré-treinados de CNN com arquitetura VGG-16. As saídas das sub-redes são direcionadas para camadas totalmente conectadas, depois uma função de perda é aplicada e, por fim, a função de ativação *softmax* projeta as probabilidades nas 11 saídas resultantes. O estudo atingiu precisão de 85,30%, *recall* de 88,70% e *F-measure* de 86,20%.

Ciaramella et al. (2023) apresentaram um modelo de detecção de *ransomware* que transforma *opcodes* em imagens coloridas e as classifica por meio de CNN. Para realizar o estudo,

foram coletadas 5000 amostras de *ransomware*, 5000 de *malwares* genéricos e 5000 de *goodwares*. Não houve categorização por família de *ransomware*. Os autores extraíram os *opcodes* das amostras com a ferramenta *objdump* e converteram esses dados em imagens quadradas. Cada *opcode* foi associado a um *pixel* com valores *Red, Green and Blue* (RGB). O conjunto de imagens resultante foi disponibilizado publicamente. Dentre as diferentes CNNs testadas, os resultados mais promissores foram obtidos com o uso da arquitetura VGG-16, onde alcançaram acurácia de 96,90%, precisão de 97,00%, *recall* de 96,90% e *F-measure* de 96,90%. Além disso, os autores aplicaram a técnica *Gradient-weighted Class Activation Mapping* (Grad-CAM) para interpretar as saídas do modelo e aprimorar a compreensão das decisões tomadas por ele.

Os estudos mencionados sobre detecção de *ransomware* investigaram tipos únicos de características. No entanto, tais abordagens têm limitações, uma vez que podem capturar apenas algumas informações para uma detecção precisa de *ransomware*. Uma abordagem combinada de características, por outro lado, pode adquirir uma ampla gama de informações relacionadas ao comportamento e à estrutura do *ransomware*. Além disso, combinar diferentes tipos de características pode aprimorar o desempenho de detecção e auxiliar no enfrentamento de técnicas de evasão (Garcia; Hammad; Malek, 2018; Wu; Zhu; Liu, 2021).

Gaur et al. (2021) propuseram um modelo de detecção baseado em diversas características estruturais de atributos numéricos, como tamanho das seções, endereços utilizados e entropia; atributos textuais, como DLL e símbolos importados, chamadas de API requisitadas e presença de empacotadores; atributos únicos, como valores de *hash* MD5 e SHA1; e códigos de operação. O estudo utilizou um conjunto de dados com cerca de 23 mil amostras de *ransomware* e 27 mil amostras benignas. O modelo proposto alcançou uma acurácia de 99,68%, precisão de 99,72%, *recall* de 99,65% e *F-measure* de 99,72% com o classificador RF.

Vidyarthi et al. (2019) conduziram um estudo para explorar as características discriminantes de arquivos executáveis que distinguem *ransomware* de outros *malwares* e executáveis benignos. Os autores utilizaram um conjunto de dados composto por 2488 amostras benignas e 2722 amostras maliciosas (incluindo *ransomware* e outros *malwares*). O estudo identificou 60 propriedades estruturais típicas de *ransomware*, como a presença de empacotadores, entropia do arquivo e da seção de dados, textos imprimíveis usuais, comandos de modificação de chave de registro e DLL de comunicação de rede. O estudo alcançou uma acurácia de 99,25% e valores de precisão, *recall* e *F-measure* de 99,30% com o classificador RF.

Ayub e Sirai (2021) investigaram indicadores suspeitos em diversas amostras de *ransomware* para auxiliar pesquisadores a entender melhor os perfis desses *malwares*. O estudo coletou 727 amostras de *ransomware* que foram agrupadas em 50 famílias. O estudo analisou várias características dos binários após sua desmontagem, como metadados do cabeçalho, funções e DLL importadas e textos em ASCII incorporados ao binário. Foi observado, por exemplo, que 62% das amostras de *ransomware* acessam funções de movimento do *mouse* ou do cursor, como TrackMouseEvent e GetCursorPos, o que pode indicar que são usadas para monitorar

se o usuário está ativo ou não. Com base no conjunto de dados obtido, aplicaram redução de dimensionalidade com a técnica de *Principal Component Analysis* (PCA), em seguida, verificaram a formação de *clusters* com o algoritmo *K-Means* e selecionaram a quantidade de três *clusters* devido a ser o ponto antes de uma diminuição aproximadamente linear na inércia. Por fim, aplicaram três classificadores unários (*one-class*) para encontrar similaridades entre os *ransomwares* através da formação dos *clusters* aprendidos. Os autores avaliaram a porcentagem de instâncias de teste que se enquadraram no cluster, e o melhor resultado do estudo foi uma acurácia de 89,96% com o classificador *Local Outlier Factor* (LOF). Contudo, o estudo apresentou apenas a métrica de acurácia, o que prejudica a avaliação e comparação da eficácia do método.

Embora as técnicas utilizadas nos estudos de Ayub e Sirai (2021) e Zhu et al. (2022) sejam inovadoras e a classificação de famílias seja relevante em situações em que já se tem conhecimento prévio de que a amostra testada é maliciosa, destaca-se que esses estudos deixaram de considerar os *softwares* benignos. Essa omissão inviabiliza a avaliação da detecção de amostras pertencentes à classe negativa (*goodwares*) e prejudica a comparação com outros métodos existentes. A inclusão de amostras benignas no conjunto de dados é fundamental para uma avaliação completa e precisa da eficácia do método proposto em distinguir entre *softwares* maliciosos e benignos.

Além disso, todos os estudos anteriormente mencionados para detecção de *ransomware* utilizaram abordagens padrão, como *Cross-Validation* (CV) ou divisão de treino-teste de 80/20. Essas abordagens incluem diferentes amostras das mesmas famílias de *ransomware* nos conjuntos de treinamento e teste. Embora esses estudos tenham alcançado bons resultados métricos, o cenário de DNFR, onde não há amostras de uma determinada família na fase de treinamento, apresenta um desafio mais significativo com aplicações práticas. Esse teste expõe uma capacidade específica de detectar ataques de dia zero, fundamental em qualquer sistema de proteção para garantir robustez e confiabilidade.

Poucos autores têm seguido metodologias de teste semelhantes de DNFR, como Sgandurra et al. (2016), Cohen e Nissim (2018), Shaukat e Ribeiro (2018) e Zahoora et al. (2022). No entanto, as análises de detecção desses estudos são dinâmicas ou híbridas, e suas limitações são bem documentadas, pois são conhecidas por serem relativamente lentas devido ao tempo necessário para capturar o comportamento da amostra em um ambiente controlado. Por outro lado, esta tese investiga a DNFR por meio de análise estática, o que proporciona desempenho rápido, adequado para detecções em tempo de execução.

No estudo conduzido por Vehabovic et al. (2023), foi introduzido um *framework* de ML para detecção precoce de *ransomware* com base em poucas características estruturais do cabeçalho. Os autores coletaram 1240 amostras de *ransomware*, provenientes de nove famílias diferentes, bem como 2000 amostras de *goodware*. No método proposto, os autores realizaram testes em um cenário de classificação multiclasse com conjuntos de 5, 7, 10 e 15 características específicas do cabeçalho do arquivo executável. Inicialmente, o estudo utilizou uma divisão de

85/15 para treinamento/teste nos conjuntos de dados, sem distinção de famílias. Os resultados mais promissores foram obtidos com o conjunto de 15 características ao aplicar os classificadores RF e *eXtreme Gradient Boosting* (XGB), ambos alcançando níveis de detecção semelhantes, com uma acurácia de 95% e um recall de 95%. Posteriormente, o estudo abordou a detecção de ameaças de dia zero por meio de uma metodologia de DNFR ao isolar uma das famílias para teste, enquanto as outras foram usadas para treino. O melhor desempenho registrado nesse cenário foi uma acurácia média de 81,04% com o classificador RF. Contudo, a limitada diversidade de famílias pode ter prejudicado a generalização do modelo, e os autores reconheceram que a escassez de *features* pode ter contribuído para um baixo desempenho em famílias específicas, por exemplo, a família *WannaCry* que obteve uma acurácia de detecção de apenas 14,86%. O estudo não apresentou outras métricas.

Em pesquisa anterior do autor desta tese (Moreira; Moreira; Sales JR., 2023), foi indroduzido o modelo *Xception ColSeq*, um método de análise estática para detectar *ransomware* pela conversão o cabeçalho em imagens coloridas em um padrão vetorial sequencial, seguido por classificação com um modelo de CNN Xception sem transferência de aprendizado. Os benefícios dessa abordagem incluem a simplificação da extração de características e a redução da carga de processamento. A eficácia do método foi avaliada para o cenário de DNFR ao utilizar um conjunto de dados com 1023 amostras de *ransomware* de 25 famílias e 1134 de *goodware*. Os resultados foram médias ponderadas de 93,84%, 97,50%, 89,77% e 92,27% para acurácia, precisão, *recall* e *F-measure*, respectivamente. Além disso, foi aplicado um método de interpretação visual para analisar regiões significativas de detecção do cabeçalho para a tomada de decisão do modelo. As descobertas indicaram que o Cabeçalho Opcional e os primeiros Cabeçalhos de Seção são os campos mais decisivos. No entanto, uma limitação da pesquisa anterior é o uso apenas um tipo de *feature* para a detecção.

Dentre todos os estudos mencionados no contexto da detecção de *ransomware*, vale ressaltar que apenas Ciaramella et al. (2023), Sgandurra et al. (2016) e Moreira, Moreira e Sales Jr. (2023) disponibilizaram publicamente seus conjuntos de características. A prática de não compartilhar conjuntos de dados é comum nessa área de pesquisa (Fernando; Komninos; Chen, 2020). Portanto, esta tese disponibiliza publicamente o conjunto de *features* extraídas para facilitar a implementação de novas técnicas de proteção contra *ransomware*.

1.3 Justificativa e Contribuições

Embora pesquisas anteriores apresentem resultados encorajadores, o desenvolvimento de um modelo de análise estática abrangente voltado para a DNFR pode resultar em maior precisão, rapidez e confiabilidade. A Tabela 1 resume as vantagens e limitações de cada trabalho relacionado. Esta tese preenche lacunas dos estudos anteriores, mantendo seus benefícios, como será evidenciado pela metodologia e pelos resultados.

Frente às lacunas identificadas, esta tese desenvolve um modelo eficaz de detecção de

Tabela 1 – Resumo das vantagens e limitações dos trabalhos relacionados.

Trabalhos	Vantagens	Limitações
(Rezaei; Manavi; Hamzeh, 2021) (He; Yu; Song, 2022) (Rizvi et al., 2022) (Paik; Jin; Cho, 2022) (Yousuf et al., 2023)	Complementa técnicas contemporâneas *antimalware*	Não foca em um *malware* específico
(Zhang, H. et al., 2019) (Zhang, B. et al., 2020) (Khammas, 2020) (Stiawan et al., 2021) (Manavi; Hamzeh, 2021) (Manavi; Hamzeh, 2022a) (Manavi; Hamzeh, 2022b) (Zhu et al., 2022) (Ciaramella et al., 2023)	Complementa técnicas contemporâneas *antimalware* Foca em um *malware* específico	Investiga único tipo de *feature*
(Gaur et al., 2021)	Combina diferentes tipos de *features*	Não avalia a DNFR
(Vidyarthi et al., 2019) (Ayub; Sirai, 2021)	Combina diferentes tipos de *features* Analisa as *features* mais importantes	Não avalia a DNFR
(Sgandurra et al., 2016)	Avalia a DNFR Cria e compartilha o conjunto de dados com as *features* extraídas	Aplica análise dinâmica/híbrida (não adequada para detecção em tempo de execução)
(Cohen; Nissim, 2018) (Shaukat; Ribeiro, 2018) (Zahoora et al., 2022)	Avalia a DNFR	Aplica análise dinâmica/híbrida (não adequada para detecção em tempo de execução)
(Vehabovic et al., 2023)	Avalia a DNFR Aplica análise estática (adequada para detecção em tempo de execução)	Investiga único tipo de *feature*
(Moreira; Moreira; Sales JR., 2023)	Avalia a DNFR Aplica análise estática (adequada para detecção em tempo de execução) Analisa as *features* mais importantes Cria e compartilha o conjunto de dados com as *features* extraídas	Investiga único tipo de *feature*

Fonte: Elaborado pelo autor

um tipo específico de *malware*, o que complementa as tecnologias *antimalware* atuais. Este modelo utiliza as combinações mais relevantes de conjuntos de características estruturais, acompanhado por uma seleção apropriada de classificadores com integração de predições.

Além disso, a tese investiga a problemática da DNFR, um desafio contínuo e pouco explorado no campo da segurança cibernética. A abordagem adotada usa análise puramente estática, que pode ser empregada em tempo de execução em dispositivos com recursos de *hardware* limitados, como computadores pessoais ou dispositivos móveis, tornando-a mais adequada para aplicações do mundo real.

Esta pesquisa também realiza a análise de *features* para identificar e avaliar as variáveis mais significativas usadas no conjunto de características combinadas para o processo de tomada de decisão do modelo. Ainda, examina e explica os erros de classificação cometidos pelo modelo.

Por fim, a tese supre carências na área ao disponibilizar um conjunto de dados com características extraídas, categorizando 40 famílias críticas de *ransomware* com base em relatórios de fornecedores globais de cibersegurança. Isso amplia oportunidades para o desenvolvimento de novos métodos de detecção dessa ameaça digital.

Desta forma, o estudo conduzido nesta tese representa uma contribuição para o avanço dessa área de pesquisa e impulsiona o desenvolvimento de soluções mais robustas e confiáveis no campo da DNFR. Esta pesquisa preenche lacunas da área, disponibiliza recursos valiosos e abre caminho para o aprimoramento de técnicas de detecção inovadoras.

1.4 Objetivos

O objetivo geral deste estudo consiste em desenvolver um modelo eficaz de análise estática para a detecção de famílias de *ransomware* de dia zero (DNFR) através de abordagens inteligentes. Para tanto, é necessário o cumprimento dos objetivos específicos a seguir:

- Coletar amostras de famílias de *ransomware* recentes e relevantes, com critérios objetivos;
- Criar um conjunto de dados contendo características estruturais extraídas das amostras de *ransomwares* e de *softwares* benignos;
- Avaliar diversos conjuntos de características, tanto de forma individual quanto combinada, assim como diferentes classificadores;
- Utilizar uma metodologia de DNFR: caso particular de detecção de ataques de dia zero;
- Implementar um modelo de análise estática baseado em técnicas de ML e DL direcionados para DNFR; e
- Avaliar a eficácia do modelo desenvolvido.

1.5 Organização da Tese

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 aborda a ameaça central investigada neste estudo, o *ransomware*, seus principais tipos, fases de ataque e as características estruturais que podem ser extraídas. O Capítulo 3 descreve os materiais e métodos utilizados neste estudo. O Capítulo 4 apresenta os resultados experimentais. O Capítulo 5 compara e discute os resultados com os trabalhos relacionados. Por fim, no Capítulo 6, são formuladas as conclusões com base nos resultados desta tese, assim como é exposta a divulgação desta pesquisa.

2 ANÁLISE DE *RANSOMWARES*

Este capítulo apresenta a ameaça investigada neste estudo, o *ransomware*, e discute seus principais tipos e fases de ataque, bem como características estruturais que podem ser extraídas dos arquivos executáveis.

2.1 Tipos de *Ransomware* e Fases do Ataque

Os *ransomwares* podem ser divididos em dois grupos: *Locker* e *Crypto*. Esses dois grupos têm características diferentes em termos de ações maliciosas. Enquanto o *Locker ransomware* normalmente bloqueia o sistema da vítima exibindo uma página de *login* para impedir que o usuário acesse os arquivos, mas sem criptografar o sistema ou os dados do usuário; o *Crypto ransomware* criptografa os arquivos da vítima e exige um pagamento de resgate para descriptografá-los. Este segundo tipo constitui a maior parte no mundo cibernético e também é o mais perigoso. A capacidade do *ransomware* de criptografar arquivos e torná-los inacessíveis representa um risco significativo, pois as vítimas são forçadas a tomar decisões difíceis ao ponderar o valor de seus dados em relação às possíveis consequências de não cumprir as exigências dos atacantes (Kok et al., 2019).

Os ataques de *ransomware* podem ser segmentados em quatro fases distintas: infecção, comunicação com servidores de Comando & Controle (C&C), destruição e extorsão. Cada fase pode empregar diferentes mecanismos para prevenção e/ou detecção da ameaça (Oz et al., 2022; Begovic; Al-Ali; Malluhi, 2023). A Figura 1 ilustra as quatro fases do *ransomware* e aponta suas principais formas de combate.

Na fase de infecção, o *ransomware* é entregue ao sistema da vítima, como computadores pessoais, servidores, dispositivos móveis, dispositivos de Internet das Coisas, entre outros. Os atacantes utilizam diversos vetores de infecção, como *e-mails* de *phishing* com anexos maliciosos ou arquivos para *downloads* em sites comprometidos. Para detectar a ameaça nessa fase, são mais adequados o uso de *scanners* locais de *malware* ou sistemas de detecção de intrusão baseados em *hosts* (Oz et al., 2022). O modelo proposto nesta tese visa combater o *ransomware* na fase de infecção, antes que ele tenha a oportunidade de ser executado e causar qualquer dano.

Na fase subsequente, ao executar o *ransomware* recebido, ele estabelece uma conexão da vítima com o servidor de C&C, onde ocorre troca de dados com o atacante, como chaves de criptografia e informações do sistema alvo. Para prevenir essas ameaças, sistemas de detecção de intrusão baseados em rede são mais apropriados (Cabaj; Gregorczyk; Mazurczyk, 2018; Arivudainambi; KA; Visu et al., 2020). Embora muitos tipos de *ransomware* realizem comunicação com servidores de C&C, existem famílias que não apresentam essa propriedade. Os ataques de *ransomware* que não comunicam com o C&C reduzem a possibildade de identificação apenas

Figura 1 – Quatro fases do ataque de *ransomware* e suas as principais formas de combate.

Fonte: Elaborado pelo autor

às capacidades de detecção do próprio *host*, evitando as medidas de detecção de rede que se concentram na comunicação entre os dois (Berrueta et al., 2019).

Em seguida, na fase de destruição, o *ransomware* executa suas ações maliciosas para impedir o acesso do usuário aos seus arquivos ou ao sistema. Essa fase de ataque pode incluir etapas de geração de chaves de criptografia, busca no sistema de arquivos e exfiltração de arquivos com extensões específicas ou de diretórios particulares, além de exclusão de *backups* (Begovic; Al-Ali; Malluhi, 2023). Nessa fase, podem ser empregadas diversas técnicas de detecção que fiscalizam o comportamento do sistema. Uma abordagem consiste no monitoramento das atividades no sistema de arquivos, em que arquivos-chamariz com extensões típicas de arquivos de usuários, como .docx, .pptx e .txt, são deliberadamente colocados no sistema de arquivos para atrair os ataques (*File System Honeypots*). Outra abordagem envolve o monitoramento da utilização de recursos, como CPU, memória *Random Access Memory* (RAM) e requisições de Entrada/Saída, com o objetivo de identificar taxas de utilização anormais. Além disso, uma abordagem comumente utilizada e eficaz é o monitoramento de eventos durante a execução, coletando dados como chamadas de sistema, chamadas de APIs, nomes e extensões de arquivos modificados e alterações no registro do sistema (Moussaileb et al., 2021).

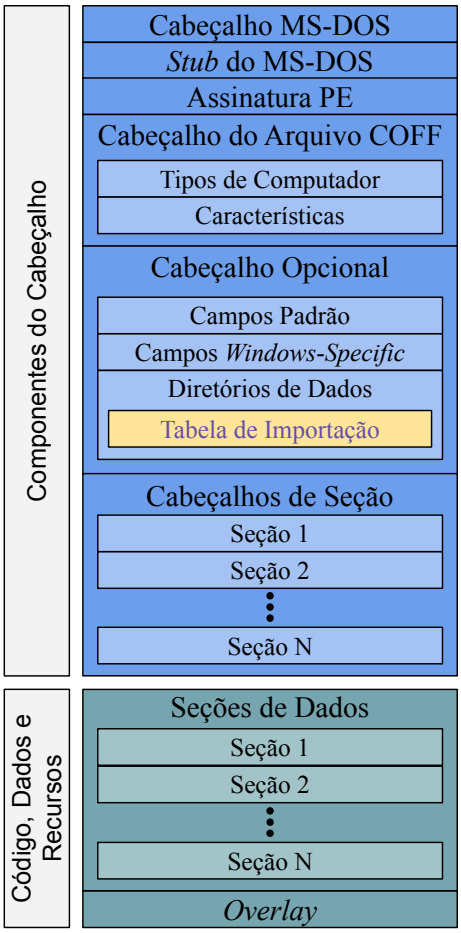
Por fim, na fase de extorsão, o *ransomware* notifica a vítima sobre o ataque ao exibir uma mensagem de resgate. Essa mensagem contém informações sobre o ataque e instruções para o pagamento do resgate. Sistemas de detecção que se baseiam na identificação da tela inicial, papel de parede ou mensagem de resgate são apropriados para essa fase, pois permitem a identificação da variante de *ransomware* envolvida no ataque, o que possibilita que os usuários procurem por ferramentas específicas de descriptografia (Atapour-Abarghouei; Bonner; Mc-

Gough, 2019), como as disponibilizadas pelo projeto *The No More Ransomware*.[1]

2.2 Características Estruturais de Arquivos Executáveis

No contexto de arquivos executáveis, o formato de arquivo *Portable Executable* (PE) é um formato padrão para executáveis baseados no SO *Windows*. Ele pode ser submetido a uma análise estática, na qual são extraídas diversas características estruturais valiosas sem a necessidade de executá-lo. O binário PE engloba os componentes que compõem o executável, como o código, os dados, os recursos e seus metadados. A Figura 2 ilustra os principais elementos da estrutura do arquivo PE. O formato de arquivo PE consiste em vários componentes, como o cabeçalho PE, que contém dados vitais sobre o binário, a Tabela de Importação, que armazena as DLLs importadas e as chamadas de funções, as seções de dados, que mantêm o código executável e os dados, e um bloco extra (*Overlay*), que pode servir como armazenamento adicional de dados (Microsoft, 2023).

Figura 2 – Visão geral dos principais elementos que compõem a estrutura do arquivo PE.



Fonte: Elaborado pelo autor

[1] https://www.nomoreransom.org/en/decryption-tools.html

2.2.1 Campos do Cabeçalho

O cabeçalho completo de um arquivo PE inclui os campos Cabeçalho do MS-DOS, *Stub* do MS-DOS, Assinatura PE (em inglês, PE *Signature*), Cabeçalho do Arquivo *Common Object File Format* (COFF), Cabeçalho Opcional (em inglês, *Optional Header*) e Cabeçalhos de Seção (em inglês, *Section Headers*), os quais contêm metadados em diferentes níveis, mostram a estrutura e fornecem dados significativos sobre o binário executável. Alguns campos não são obrigatórios, como cabeçalhos de seção, ou não têm tamanho fixo, como *ImageBase*, *SizeOfStackReserve*, *SizeOfStackCommit*, *SizeOfHeapReserve* e *SizeOfHeapCommit* de campos específicos do *Windows*.

O cabeçalho é um componente indispesável de uma amostra binária, pois contém diversos dados sobre a estrutura e características do binário. O cabeçalho PE contém campos que descrevem o tipo do arquivo, tamanho, ponto de entrada, seções e outros detalhes relevantes. Esses detalhes podem ser usados para identificar e analisar uma amostra binária, como sua plataforma e SO pretendidos, bem como peculiaridades ou anormalidades em sua estrutura.

O cabeçalho PE pode ser uma ferramenta útil para a detecção de *ransomware*. Vários campos dentro do cabeçalho podem ser aproveitados para esse fim. Por exemplo, o campo *SizeOfCode* no *Optional Header* fornece o tamanho da seção de código, o que pode ser usado para determinar se o binário é grande o suficiente para conter o código do *ransomware*. Além disso, a Tabela de Importação, que lista as DLLs e funções usadas pelo executável, pode ser usada para identificar chamadas de API maliciosas ou inesperadas feitas pelo *ransomware*. Adicionalmente, o cabeçalho PE pode ser usado para identificar se o binário está compactado ou ofuscado, que são técnicas comuns usadas pelo *ransomware* para evitar a detecção.

2.2.2 *Dynamic-Link Libraries* Importadas e Chamadas de Função

As DLLs desempenham um papel fundamental na execução de arquivos PE, uma vez que são carregadas dinamicamente pelo SO conforme necessário. Elas fornecem um conjunto de rotinas, chamadas de função e estruturas de dados que podem ser utilizadas pelo binário executável, o que reduz a quantidade de código necessária a ser incluída no próprio arquivo executável. A análise das DLLs importadas e suas funções associadas em uma amostra binária pode fornecer percepções significativas sobre seu comportamento e funcionalidade. Essa análise se torna particularmente relevante na identificação de atividades maliciosas, uma vez que as ameaças podem utilizar DLLs e funções específicas para executar seu código malicioso ou modificar as existentes para alcançar seus objetivos. Consequentemente, o monitoramento e a análise das DLLs importadas e chamadas de função possibilitam o desenvolvimento de mecanismos de defesa eficazes contra os *malwares*.

Os dados sobre as DLLs e as chamadas de função importadas em um arquivo PE podem ser encontradas principalmente na Tabela de Importação, localizada nos Diretórios de Dados

(em inglês, *Data Directories*) do Cabeçalho Opcional. A Tabela de Importação contém uma lista de DLLs importadas, juntamente com os nomes e endereços das funções que o arquivo executável usa de cada DLL. No entanto, esses dados também podem ser encontrados em outras partes do arquivo PE, como a Tabela de Endereços de Importação, que é uma estrutura de dados em tempo de execução que armazena os endereços de memória reais das funções importadas depois de serem carregadas na memória. Além disso, a Tabela de Importação com Atraso, também localizada no cabeçalho PE, contém uma lista de DLLs e funções que são carregadas apenas quando são chamadas pela primeira vez, o que proporciona uma forma de otimizar o carregamento de executáveis grandes (Microsoft, 2023). Portanto, para analisar completamente as DLLs e as funções importadas em um arquivo PE, é preciso considerar todas as localizações onde esses dados podem ser encontrados.

2.2.3 Seções de Dados

Um arquivo PE é dividido em uma série de seções, cada uma das quais contém informações específicas sobre o código executável, dados, recursos e outros metadados. Essas seções fornecem uma estrutura padronizada para o arquivo executável e permitem que o SO e outras ferramentas localizem e interpretem os diferentes componentes do arquivo de forma eficiente. As seções são tipicamente identificadas por nomes como .text, .rdata, .data, .rsrc, .reloc e outros, onde cada uma corresponde a um tipo específico de dado armazenado no arquivo. Por exemplo, a seção .text contém o código executável, enquanto a seção .rdata armazena dados somente leitura, como constantes e cadeias de caracteres. A seção .data contém dados inicializados, como variáveis globais e estáticas, e a seção .rsrc contém recursos, como ícones, imagens e caixas de diálogo. A seção .reloc armazena dados de realocação, que são usadas pelo SO para ajustar endereços quando o executável é carregado na memória (Microsoft, 2023).

Destaca-se que as seções de dados mencionadas estão tipicamente presentes na amostras binárias. Contudo, existem várias seções diferentes, sem uma convenção de nomenclatura fixa, que podem ser especificadas pelo compilador. O compilador tem a capacidade de definir o nome, tamanho e características de cada seção no arquivo objeto antes que ele seja vinculado ao executável final. Essa flexibilidade permite que os desenvolvedores personalizem a estrutura do executável para melhor atender às suas necessidades e otimizar o desempenho do programa. A capacidade de personalizar as seções também pode apresentar desafios na análise estática de binários, pois pode dificultar a determinação da função de cada seção sem dados adicionais.

Além disso, o *Overlay* é um bloco adicional que contém dados anexados ao final do arquivo PE, após a última seção. Esses dados não são carregados na memória quando o programa é executado e podem ser usados para diversos fins, como armazenar recursos adicionais do programa ou personalizar o comportamento do programa. Um *software* benigno pode usar o *Overlay* para armazenar dados adicionais, como informações de configuração necessárias durante a execução. Por outro lado, um *software* malicioso pode usar o *Overlay* para ocultar seu

código ou dados, pois ele não é analisado pelo carregador do *Windows*.

A análise das seções de dados e do *Overlay* é uma técnica relevante para investigar arquivos binários. Contudo, quando um binário é empacotado ou ofuscado, o código original é comprimido ou criptografado, o que dificulta sua análise. Ferramentas de empacotamento são comumente usadas por criadores de *malware* para evitar sua detecção por *softwares* de segurança. Uma maneira de detectar código empacotado ou ofuscado é pela análise de entropia de cada seção no arquivo binário. A entropia é uma medida matemática usada para avaliar o grau de incerteza presente em um conjunto de números (ou *bytes*), com o objetivo de quantificar o nível de dificuldade associado à previsão independente de cada elemento na série (Mantovani et al., 2020).

Especificamente, a entropia de Shannon S de um evento aleatório discreto y, definida por

$$S(y) = -\sum_{u=1}^{U} P(y_u) \log_2(P(y_u)), \tag{1}$$

é utilizada para estimar o número de bits necessários para codificar um dado, onde $P(y_u)$ denota a probabilidade da u-*ésima* unidade de informação (como um número) na série de símbolos do evento y com U símbolos. Por meio dessa fórmula, é possível determinar a quantidade de incerteza ou aleatoriedade presente na série, onde uma entropia mais elevada indica maior imprevisibilidade. Ao considerar dados binários, nos quais cada símbolo pode assumir $U = 256$ valores possíveis, o intervalo de entropia pode variar de $0,0$ a $8,0$. Na prática, é comum encontrar valores de entropia elevados, geralmente acima de $6,5$, associados a funções de compressão sem perdas e criptografia, nas quais o objetivo é criar dados de saída que sejam difíceis de prever ou comprimir (Mantovani et al., 2020).

Portanto, quando uma seção contém código compactado ou ofuscado, a entropia dessa seção será alta. Em contraste, seções que contêm código não compactado ou não ofuscado terão um valor de entropia mais baixo. A incorporação da característica de entropia permite que o aprimoramento da confiabilidade dos cálculos envolvidos na determinação das relações entre as amostras de *ransomware*. Como resultado, essa abordagem pode ajudar na identificação de variantes derivadas da base de código da mesma família de *ransomware* (Zhu et al., 2022).

2.2.4 Códigos de Operação

Os *opcodes* são os blocos de construção do código executável e representam as instruções executadas por uma CPU. Os *opcodes* podem ser extraídos por meio da desmontagem do binário por uma ferramenta apropriada, como Capstone, Distorm3, Ghidra, IDA Pro ou Radare2. Essa extração permite que os pesquisadores identifiquem padrões e assinaturas únicas que auxiliam na classificação de variantes de *malware* ou na detecção de famílias específicas de

ransomware.

É comum analisar os *opcodes* com Processamento de Linguagem Natural (PLN), que é uma área especializada da Ciência da Computação que trata da interação entre computadores e linguagem natural. Uma técnica amplamente utilizada em PLN para detectar *malware* é a análise de N-gramas dos *opcodes*, com seus pesos calculados usando o método TF-IDF. Essa abordagem consiste em dividir as sequências de *opcodes* em pequenos segmentos, chamados de N-gramas, e calcular a pontuação TF-IDF para cada N-grama (Zhang, H. et al., 2019).

Um esquema de N-grama é um modelo de linguagem probabilístico que pode, por exemplo, prever o próximo item possível em uma sequência dada. Por exemplo, no contexto dos opcodes, uma sequência {*add*, *push*, *pop*, *mov*, *int*, *mov*} é representada como uma série de N-gramas quando $N = 3$, que consiste em {(*add*, *push*, *pop*), (*push*, *pop*, *mov*), (*pop*, *mov*, *int*) e (*mov*, *int*, *mov*)}. Em cenários de classificação e mineração de dados, a seleção do valor N afeta significativamente o desempenho. Um valor de N baixo pode levar a uma diminuição no desempenho do modelo, enquanto um valor de N alto pode armazenar mais informações de contexto. No entanto, o número de *opcodes* de N-gramas únicos aumenta exponencialmente à medida que N aumenta, o que pode causar problemas com o custo computacional do modelo. Valores de N comuns e com bons resultados utilizados em pesquisas anteriores variam de 3 a 5, o que equilibra o desempenho, as informações de contexto e o custo computacional (Poudyal et al., 2019; Zhang, B. et al., 2020).

A pontuação TF-IDF mede a relevância de um N-grama em um documento, ponderando a Frequência do Termo (TF), que representa com que frequência o N-grama aparece no documento, em relação à Frequência Inversa do Documento (IDF), que representa com que frequência o N-grama aparece em todos os documentos. O cálculo do TF-IDF é obtido multiplicando-se a TF pela IDF. Matematicamente, a TF é definida como

$$TF(i,j) = \frac{n(i,j)}{\sum_k n(k,j)}, \tag{2}$$

onde $TF(i,j)$ é a frequência do N-grama i na sequência j, $n(i,j)$ denota o número de vezes que o N-grama i ocorre na sequência de N-grama j, e $\sum_k n(k,j)$ é o número total de N-gramas na sequência j. Enquanto a IDF é definida como

$$IDF(i) = \log_2 \frac{|D|}{|\{j : i \in j | j \in D\}|}, \tag{3}$$

onde $IDF(i)$ expressa se o N-grama i é pouco frequente em todas as sequências de N-gramas, D é o conjunto de todas as sequências de N-gramas e $|\{j : i \in j | j \in D\}|$ representa o número de sequências de N-gramas que contêm o N-grama i. Portanto, TF-IDF é

$$TF - IDF(i,j) = TF(i,j) \times IDF(i), \tag{4}$$

onde $TF - IDF(i, j)$ designa o valor TF-IDF do N-grama i na sequência de N-grama j. Quanto maior o valor do TF-IDF, mais significativo é o N-grama na sequência de *opcodes* em relação ao conjunto (Zhang, H. et al., 2019).

2.2.5 Texto Imprimível

A presença de texto legível por humanos, composto por caracteres imprimíveis do conjunto ASCII, em arquivos binários, pode fornecer informações valiosas sobre a função e o propósito do código. Não é incomum que o código de *ransomware* inclua cadeias de caracteres com palavras-chave e frases específicas relacionadas às suas atividades maliciosas, especialmente em suas notas de resgate. Por exemplo, a nota de resgate apresentada na amostra `63 3b9d373da7d2916f4d3b2902d4817c0f3ad5de5466ac85f34bdd37a8d3dd37` (SHA-256) da variante Conti:

```
All of your files are currently encrypted by CONTI ransomware.
If you try to use any additional recovery software - the files might be damaged or
lost.

To make sure that we REALLY CAN recover data - we offer you to decrypt samples.

You can contact us for further instructions through our website :

TOR VERSION :
(you should download and install TOR browser first https://torproject.org)

http://m232fdxbfmbrcehbrj5iayknxnggf6niqfj6x4iedrgtab4qupzjlaid.onion

HTTPS VERSION :
https://contirecovery.info

YOU SHOULD BE AWARE!
Just in case, if you try to ignore us. We've downloaded your data and are ready to
publish it on out news website if you do not respond. So it will be better for both
sides if you contact us ASAP.

---BEGIN ID---
7c85vpfYlRYHIA03SjFhX3oDfk2uTNlCQ8IROOMM33gL1FASiKPeodbG1K5YULtD
---END ID---
```

A análise dos caracteres imprimíveis em amostras binárias permite que analistas de segurança detectem cadeias de caracteres específicas e as utilizem para determinar se uma amostra é um *ransomware* ou não. Técnicas de PLN como N-gramas e TF-IDF são abordagens comumente utilizadas no tratamento desse tipo de informação.

3 MATERIAIS E MÉTODOS

Este capítulo descreve os materiais e métodos empregados para atingir os objetivos desta tese. Inicialmente, são descritos os conjuntos de dados, as características analisadas, as ferramentas, o ambiente de experimento, assim como os parâmetros e hiperparâmetros utilizados no modelo. O objetivo do modelo é identificar com precisão variantes desconhecidas de *ransomware* por meio de técnicas de Aprendizado de Máquina ao utilizar análise estática de múltiplos conjuntos de características estruturais extraídas das amostras executáveis. Essa abordagem eleva a robustez e confiabilidade das soluções de segurança, bem como permite verificações locais em tempo de execução. Dessa forma, o modelo pode atuar como uma camada adicional de segurança antes da execução de arquivos binários no SO *Windows*. Em seguida, são apresentadas as métricas de avaliação de desempenho, a metodologia de comparação dos resultados dos classificadores, a abordagem de teste para a DNFR e, por fim, o método de interpretação de resultados.

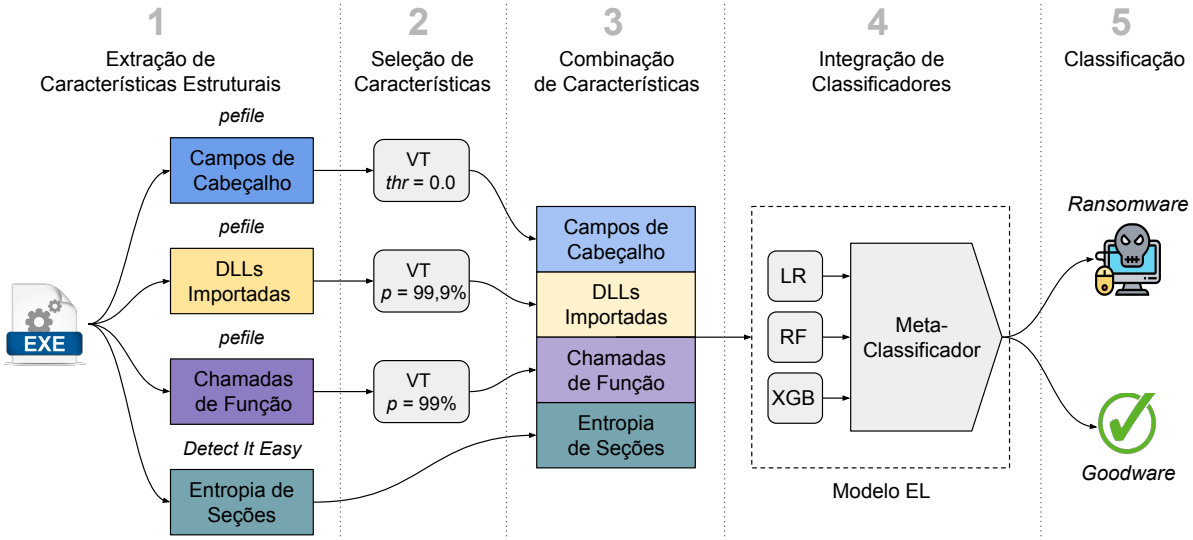
A Figura 3 ilustra o fluxo de trabalho do modelo de análise estática com combinação de características estruturais do binário executável. Inicialmente, dados estruturais são extraídos da amostra executável, como campos específicos do cabeçalho, DLLs importadas, chamadas de funções e entropia das seções (Etapa 1). Em seguida, o método de seleção de características *Variance Threshold* (VT) é aplicado a três dos quatro conjuntos de características para reduzir a dimensionalidade (Etapa 2). Posteriormente, esses quatro conjuntos resultantes são combinados em um único vetor (Etapa 3). O vetor combinado de características estruturais alimenta um modelo *Ensemble Learning* (EL) de votação ponderada composto pelos algoritmos *Logistic Regression* (LR), RF e XGB (Etapa 4), que classifica se a amostra é *ransomware* ou *goodware* (Etapa 5). Este método oferece uma abordagem abrangente para a detecção de *ransomware*, utilizando múltiplos conjuntos de características, métodos de seleção de características e técnicas de ML em conjunto para melhorar a performance métrica do modelo no desafiador cenário de DNFR.

3.1 Descrição dos Dados

O modelo é avaliado com um conjunto de dados que contém 2675 amostras binárias executáveis. O conjunto de treinamento e validação consiste em 2157 amostras (80%): 1023 *ransomwares* pertencentes a 25 famílias relevantes e 1134 *goodwares*. O conjunto de testes consiste em 518 amostras (20%): 385 *ransomwares* pertencentes a 15 famílias recentes e 133 *goodwares*. O conjunto de dados completo, com as características extraídas e seus devidos nomes, foi disponibilizado publicamente no formato *Comma-Separated Values* (CSV) por meio do repositório de dados *Mendeley Data*.[1]

[1] https://data.mendeley.com/datasets/yzhcvn7sj5

Figura 3 – Fluxo de trabalho do modelo proposto de análise estática com combinação de características estruturais.



Fonte: Elaborado pelo autor

A Tabela 2 lista os nomes das famílias, identificador (ID) de família atribuído, o primeiro ano de observação da ameaça, o número de vezes apontadas nos relatórios, o número de amostras coletadas e a respectiva distribuição percentual. A escolha das 25 famílias de treinamento foi baseada nas informações de relatórios de sete instituições globais de segurança cibernética, os quais apontam uma grande diversidade de famílias de *ransomware* perigosas (consultar a Tabela 14 no Apêndice A). Todavia, as informações desses relatórios nem sempre convergem. Portanto, foram utilizados dois critérios para selecionar as variantes mais relevantes: 1) primeira observação da família ocorreu a partir de 2020 *ou* 2) foi apontada por pelo menos dois dos sete relatórios.

Além disso, a seleção das 15 variantes de teste foi baseada em cinco relatórios mais recentes (consultar a Tabela 15 no Apêndice A), onde qualquer família relatada não pertencente ao conjunto de amostras de treinamento foi coletada. Todas as amostras de *ransomware* foram baixadas das plataformas VirusShare[2] e Hybrid-Analysis.[3] Esses arquivos são binários executáveis do SO *Windows* com tamanhos mínimo e máximo de 14,8 kB e 12,2 MB, respectivamente.

O conjunto de dados abrange um período temporal de 2016 a 2021 para treinamento e de 2019 a 2022 para teste. Esses intervalos permitem que o modelo aprenda a partir de dados históricos, ao mesmo tempo em que é testado em ameaças mais recentes para garantir sua adaptabilidade ao cenário de *ransomware* em constante evolução. As famílias de *ransomware* selecionadas abrangem um espectro diversificado que incluem grupos maliciosos de desenvolvedores RaaS pequenos e grandes. Eles exploram diversos tipos de vulnerabilidades e utilizam diferentes linguagens de programação, compiladores e algoritmos de criptografia (Begovic; Al-Ali; Malluhi, 2023). Essa diversidade contribui para treinar um modelo robusto capaz de detectar

[2] https://www.virusshare.com/
[3] https://www.hybrid-analysis.com/

Tabela 2 – Quarenta famílias de *ransomware* coletadas de acordo com os critérios definidos.

	Família	ID	1ª Observação	# Reportada	# Amostras	%
Treinamento	Avaddon	1	2020	1	49	3,48
	Babuk (Babyk)	2	2021	1	44	3,13
	BlackMatter	3	2021	1	45	3,20
	Conti	4	2020	4	48	3,41
	DarkSide	5	2020	2	50	3,55
	Dharma (CrySIS, Wadhrama)	6	2016	2	46	3,27
	DoppelPaymer	7	2019	2	23	1,63
	Exorcist	8	2020	1	19	1,35
	GandCrab	9	2018	2	50	3,55
	LockBit (ABCD)	10	2019	2	48	3,41
	Makop (Oled)	11	2020	1	35	2,49
	Maze (ChaCha)	12	2019	4	50	3,55
	MountLocker	13	2020	1	14	0,99
	Nefilim (Nephilim)	14	2020	1	39	2,77
	NetWalker (MailTo, Koko)	15	2019	3	50	3,55
	Phobos	16	2019	2	50	3,55
	Pysa (Mespinoza)	17	2019	2	38	2,70
	Ragnarok (RagnarLocker)	18	2019	2	43	3,05
	RansomeXX (Defray777, Target 777)	19	2017	2	13	0,92
	REvil (Sodinokibi)	20	2019	6	49	3,48
	Ryuk	21	2018	3	48	3,41
	Stop (Djvu)	22	2019	2	48	3,41
	Thanos (Prometeus)	23	2020	1	35	2,49
	WastedLocker	24	2020	2	40	2,84
	Zeppelin (Buran, VegaLocker)	25	2019	2	49	3,48
Teste	AvosLocker	26	2021	3	50	3,55
	BianLian	27	2021	1	11	0,78
	Black Basta	28	2022	2	32	2,27
	BlackByte	29	2021	1	13	0,92
	BlackCat (ALPHV, Noberus)	30	2021	3	50	3,55
	BlueSky	31	2022	1	34	2,41
	Clop	32	2019	1	46	3,27
	Hive	33	2021	2	50	3,55
	HolyGhost (HolyLocker, SiennaBlue)	34	2021	1	4	0,28
	Karma	35	2021	1	13	0,92
	Lorenz	36	2020	1	16	1,14
	Maui	37	2021	1	3	0,21
	Night Sky	38	2022	1	14	0,99
	PlayCrypt (Play)	39	2022	1	43	3,05
	Quantum	40	2021	1	6	0,43
	Total				1408	100

Fonte: Elaborado pelo autor

uma ampla gama de variantes de *ransomware*.

Os nomes das famílias podem ser atribuídos pelos criadores dos *ransomwares* ou pela comunidade de pesquisa dos vendedores de soluções de segurança cibernética. Determinar se

uma amostra é *ransomware* e a qual família ela pertence não é uma tarefa trivial, pois diferentes fornecedores de antivírus podem categorizar a amostra como a) não detectada, b) *ransomware* ou c) outro tipo de *malware*. Além disso, empresas de antivírus podem não especificar a família de *ransomware* ou usar nomes diferentes para a mesma família, como *REvil* e *Revilcrypt* ou *REvil* e *Sodinokibi*; ou pior, podem categorizá-los em famílias diferentes.

O site do VirusTotal fornece uma pontuação com base na análise de empresas de segurança cibernética onde, para cada amostra, é apresentada uma lista com o resultado de todas as empresas que a analisaram e suas respectivas categorizações. Desta forma, durante a coleta das amostras, a categorização das famílias dos *ransomwares* de treinamento seguiu três critérios baseados nas detecções dos sistemas de proteção apontadas pelo VirusTotal: 1) pelo menos 45 sinalizações como maliciosas, 2) pelo menos 15 sinalizações como *ransomware e* 3) pelo menos dez indicações para uma mesma variante. Esse conjunto de critérios foi balanceado para garantir uma correta categorização das famílias e obter amostras suficientes para treinamento.

Contudo, para a coleta e categorização das famílias de teste, foi necessário flexibilizar os critérios devido à escassez de amostras de algumas famílias recentes, passando a ser: 1) pelo menos 35 sinalizadas como maliciosas, 2) pelo menos dez sinalizadas como *ransomware*, *e* 3) pelo menos sete indicações para a mesma família. Essa flexibilização dos critérios permitiu uma adequada categorização das famílias e a obtenção de amostras suficientes para teste.

Para diversificar os arquivos executáveis benignos, foram coletadas amostras dos portais de *download* de *software* PortableApps[4] e Softonic[5] e de um SO *Windows* 10 com instalação padrão. Além disso, foi conduzida uma busca na internet por *softwares* que não requerem instalação. Todos os executáveis foram verificados com o antivírus ESET-NOD32[6] versão 16.0.26.0 para assegurar que são benignos.

3.2 Características Analisadas

Para uma análise abrangente dos arquivos executáveis, é necessário considerar diversas *features* que podem ser extraídas desses arquivos. Isso inclui dados dos campos do cabeçalho, DLLs importadas, chamadas de funções, entropia das seções, códigos de operação e textos imprimíveis.

Abordagens que tratam o cabeçalho de modo genérico, ao extrair sua sequência bruta de *bytes*, como as realizadas nos estudos de Khammas (2020), Manavi e Hamzeh (2021, 2022a,b) e Moreira, Moreira e Sales Jr. (2023), ocasionam a não distinção entre seus diversos campos, tamanhos e posições. Em contraste com essas abordagens, o modelo proposto nesta tese extrai campos específicos do cabeçalho e seus valores correspondentes. Isso oferece vantagens, como a redução do número de características, ao descartar campos desnecessários, enquanto mantém

[4] https://portableapps.com/
[5] https://en.softonic.com/
[6] https://www.eset.com/

significado único para cada característica independentemente de sua posição na sequência no cabeçalho. No entanto, existem desvantagens associadas, como a necessidade de utilizar ferramentas especializadas para a extração de características e a exigência de normalizar os dados, o que aumenta a complexidade dos testes para evitar vazamentos de dados (*data leakage*) (Hastie; Tibshirani; Friedman, 2009).

A biblioteca *pefile*[7] em Python foi utilizada para extrair diversas características dos arquivos PE, dentre elas, os campos dos cabeçalhos. A Tabela 3 lista as 84 *features* numéricas extraídas do cabeçalho PE das amostras coletadas com a biblioteca *pefile*. Essas características foram extraídas de vários campos específicos: 17 do DOS_HEADER, seis do FILE_HEADER, um do NT_HEADERS, 30 do OPTIONAL_HEADER e 30 do OPTIONAL_HEADER/ DATA_DIRECTORY/ IMAGE_DIRECTORY_ENTRY. Esses campos representam uma parte significativa dos dados armazenados no cabeçalho PE. Como o número de seções em um arquivo PE pode variar, os dados contidos nos cabeçalhos das seções não foram utilizados. No entanto, uma análise das seções será realizada em uma etapa posterior.

Tabela 3 – Características específicas do cabeçalho PE extraídas com biblioteca Python *pefile*.

Propriedade	Campos Específicos
DOS_HEADER	`e_magic, e_cblp, e_cp, e_crlc, e_cparhdr, e_minalloc, e_maxalloc, e_ss, e_sp, e_csum, e_ip, e_cs, e_lfarlc, e_ovno, e_oemid, e_oeminfo, e_lfanew`
FILE_HEADER	`Machine, NumberOfSections, PointerToSymbolTable, NumberOfSymbols, SizeOfOptionalHeader, Characteristics`
NT_HEADERS	`Signature`
OPTIONAL_HEADER	`Magic, MajorLinkerVersion, MinorLinkerVersion, SizeOfCode, SizeOfInitializedData, SizeOfUninitializedData, AddressOfEntryPoint, BaseOfCode, BaseOfData, ImageBase, SectionAlignment, FileAlignment, MajorOperatingSystemVersion, MinorOperatingSystemVersion, MajorImageVersion, MinorImageVersion, MajorSubsystemVersion, MinorSubsystemVersion, Reserved1, SizeOfImage, SizeOfHeaders, CheckSum, Subsystem, DllCharacteristics, SizeOfStackReserve, SizeOfStackCommit, SizeOfHeapReserve, SizeOfHeapCommit, LoaderFlags, NumberOfRvaAndSizes`
OPTIONAL_HEADER/ DATA_DIRECTORY/ IMAGE_DIRECTORY_ENTRY	`EXPORT_VA, EXPORT_Size IMPORT_VA, IMPORT_Size, RESOURCE_VA, RESOURCE_Size, EXCEPTION_VA, EXCEPTION_Size, SECURITY_VA, SECURITY_Size, BASERELOC_VA, BASERELOC_Size, DEBUG_VA, DEBUG_Size, COPYRIGHT_VA, COPYRIGHT_Size, GLOBALPTR_VA, GLOBALPTR_Size, TLS_VA, TLS_Size, LOAD_CONFIG_VA, LOAD_CONFIG_Size, BOUND_IMPORT_VA, BOUND_IMPORT_Size, IAT_VA, IAT_Size, DELAY_IMPORT_VA, DELAY_IMPORT_Size, COM_DESCRIPTOR_VA, COM_DESCRIPTOR_Size`

*VA = *VirtualAddress*.

Fonte: Elaborado pelo autor

Ao contrário de estudos anteriores na literatura (Hu; Tan, 2017; Kumar; Kuppusamy; Aghila, 2019; Wadkar; Di Troia; Stamp, 2020; Yousuf et al., 2023), que utilizaram o campo

[7] https://github.com/erocarrera/pefile

TimeDateStamp do cabeçalho OPTIONAL_HEADER como uma *feature* para classificação de *malware*, neste estudo foi optado por excluir essa característica. O valor do *TimeDateStamp* indica o número de segundos decorridos desde a meia-noite de 1º de janeiro de 1970 e é atualizado pelo compilador ou ligador (*linker)* durante o processo de compilação ou ligação. Ao considerar a metodologia DNFR utilizada neste estudo e o fato de que as famílias de teste são mais recentes, utilizar o campo *TimeDateStamp* para detectar amostras de *ransomware* de dia zero poderia introduzir viés na detecção, uma vez que é esperado que amostras mais recentes tenham um *TimeDateStamp* mais atual do que as amostras mais antigas. Além disso, se essa *feature* tiver um peso significativo na tomada de decisão do modelo, pode ser mais fácil gerar amostras que o enganem, pois a indicação de data e hora pode ser manipulada para qualquer data posterior a 1º de janeiro de 1970. Portanto, a exclusão do *TimeDateStamp* do conjunto de características resulta na criação de um modelo atemporal e mais resistente a técnicas de evasão.

A ferramenta *pefile* também foi utilizada para extrair os objetos importados das amostras coletadas (*ransomware* e *goodware*), incluindo DLLs e chamadas de função. Foram identificadas um total de 576 DLLs distintas. Em cada amostra, essas DLLs foram codificadas como características binárias, onde o valor 1 foi atribuído para a presença e o valor 0 foi atribuído para a ausência. Um procedimento similar foi utilizado para extrair chamadas de função e API, com um total de 9961 ocorrências identificadas. A identificação dos objetos importados foi restrita ao conjunto de dados de treinamento, ou seja, excluindo novas DLLs ou chamadas de função presentes no conjunto de teste. Isso proporciona um contexto de avaliação mais realista para o modelo, uma vez que reflete a situação em que o modelo é aplicado a amostras desconhecidas, sem acesso prévio a novas DLLs ou chamadas de função dessas amostras de teste.

Para extrair os dados das seções dos binários, foi empregado o *software Detect It Easy*.[8] A *feature* extraída das seções é a entropia de Shannon, formulada na Equação 1. A ferramenta *Detect It Easy* é capaz de automatizar a identificação das seções e realizar o cálculo de suas respectivas entropias. Este *software* também oferece a computação de medidas de entropia para o arquivo completo (*Total*), o cabeçalho completo (*Header*) e o bloco extra (*Overlay*). Essa abordagem resultou na identificação de 128 seções distintas do conjunto de amostras de treinamento, o que totaliza 131 características ao somar com *Total*, *Header* e *Overlay*.

Os códigos de operação foram extraídos com a biblioteca *Distorm3*[9] em Python. Em seguida, foi utilizada a biblioteca *Scikit-learn* para extrair os N-gramas dos *opcodes* do conjunto de amostras de treinamento e calcular suas pontuações TF-IDF (consultar Equação 4). Foram utilizados os valores de $N = \{3, 4, 5\}$. Devido a grande quantidade de N-gramas únicos, foram realizados experimentos para encontrar o equilíbrio entre desempenho métrico de classificação e a eficiência de tempo. Esse equilíbrio foi alcançado com a seleção dos 1000 melhores N-gramas para cada valor de N, com base nas pontuações do TF-IDF. O número de ocorrências

[8] https://github.com/horsicq/Detect-It-Easy
[9] https://pypi.org/project/distorm3/

de cada N-grama nas respectivas amostras foi utilizada como o valor final da *feature*.

Também foi realizada uma análise sobre a viabilidade de utilizar textos imprimíveis como *features* para a detecção de *ransomware* em amostras binárias. No entanto, essa abordagem foi considerada desvantajosa e, portanto, não foi incluída no modelo proposto. Isso ocorre devido à possibilidade de atacantes empregarem técnicas adversárias para evitar a detecção, como a adição de dados de cadeias de caracteres aleatórias que são comumente encontradas em *software* legítimo (Ling et al., 2023). Além disso, é possível empregar técnicas diretas para disfarçar palavras-chave nas notas de resgate (consultar a Seção 2.2.5) ao substituir letras por caracteres ou símbolos de aparência semelhante, como '*encrypted*' por '*3ncrypted*' ou '*data*' por '*d@ta*', entre outras combinações possíveis. Contornar esse método de disfarce aumenta a complexidade do conjunto de palavras para detectar *ransomware*, o que apresenta desafios na identificação de todas as formas possíveis de ofuscação.

Ainda, os métodos de extração de caracteres imprimíveis frequentemente resultam em uma grande quantidade de dados irrelevantes, o que pode prejudicar o desempenho de métodos baseados em PLN, como N-grama com TF-IDF. Para mitigar esse problema, é possível filtrar palavras que contenham apenas palavras legíveis por humanos. No entanto, a maioria das palavras retornadas não é relevante para o contexto de busca por *ransomware*, como nomes de DLLs, chamadas de funções e palavras aleatórias. Conforme mencionado, este estudo já contempla adequadamente o uso de DLLs e de chamadas de funções.

Uma solução é analisar apenas a seção de código, a qual tipicamente contém a nota de resgate; no entanto, os atacantes podem anexar a nota de resgate em outra seção de nome não definido. Ademais, alguns empacotadores, como o UPX, podem comprimir a seção .text e movê-la para outra seção, como .rsrc ou .data. Isso torna o executável menor e mais difícil de analisar, pois o código comprimido não pode ser executado diretamente até que seja descompactado. Conforme mencionado, este estudo já contempla adequadamente o uso de empacotadores por meio da análise da entropia das seções.

Por fim, é importante ressaltar que nem todas as famílias de *ransomware* possuem notas de resgate com caracteres imprimíveis. Ao analisar o conjunto de amostras de treinamento, constatou-se que apenas oito das 25 variantes de *ransomware* apresentavam notas de resgate com caracteres imprimíveis. Como resultado, essa característica foi excluída do modelo.

3.3 Redução de Dimensionalidade

Uma das técnicas comumente utilizadas para reduzir a dimensionalidade é o VT, que busca identificar e remover variáveis com baixa variância, ou seja, aquelas que têm pouca ou nenhuma variação nos valores ao longo do conjunto de dados. Essa abordagem é baseada na suposição de que variáveis com baixa variância possuem menor relevância para o problema em questão. A aplicação do VT pode levar a uma redução significativa na quantidade de variáveis, o

que simplifica o modelo e melhora sua eficiência computacional (Solorio-Fernández; Carrasco-Ochoa; Martínez-Trinidad, 2020).

O método VT é uma técnica não supervisionada do tipo filtro, que remove as *features* cuja variação não atenda a um limite pré-definido, dado por

$$Var(X) = \frac{1}{m}\sum_{a=1}^{m}(x_a - \bar{x})^2, \tag{5}$$

onde $Var(X)$ é a variância da amostra X, m é o tamanho da amostra, x_a é a a-ésima observação e $\bar{x}$ é a média da amostra (Al Fatih Abil Fida; Ahmad; Ntahobari, 2021).

Por ser um filtro não supervisionado, esse método tem a vantagem de selecionar as *features* sem a necessidade de conhecer os rótulos das classes alvo. Além disso, ao contrário dos métodos do tipo *wrapper* e híbridos, que dependem de algoritmos de agrupamento e podem apresentar o risco de *overfitting*, o VT avalia as *features* com base em suas propriedades intrínsecas, tornando-se uma escolha vantajosa para a redução de dimensionalidade e seleção de *features* em problemas de ML (Solorio-Fernández; Carrasco-Ochoa; Martínez-Trinidad, 2020).

Um caso particular da seleção de *features* do tipo VT ocorre quando é aplicado a *features* binárias. Nesse caso, baseia-se no experimento de Bernoulli, que é um experimento probabilístico que pode resultar em sucesso ($x = 1$) ou falha ($x = 0$). Desta forma, a variância para *features* binárias é dada por

$$Var(X) = p(1 - p), \tag{6}$$

onde p denota a probabilidade de sucesso, também conhecida como parâmetro de probabilidade de Bernoulli (Forbes et al., 2010). Ao expressar o parâmetro p como uma porcentagem, o valor do limite torna-se mais facilmente interpretável.

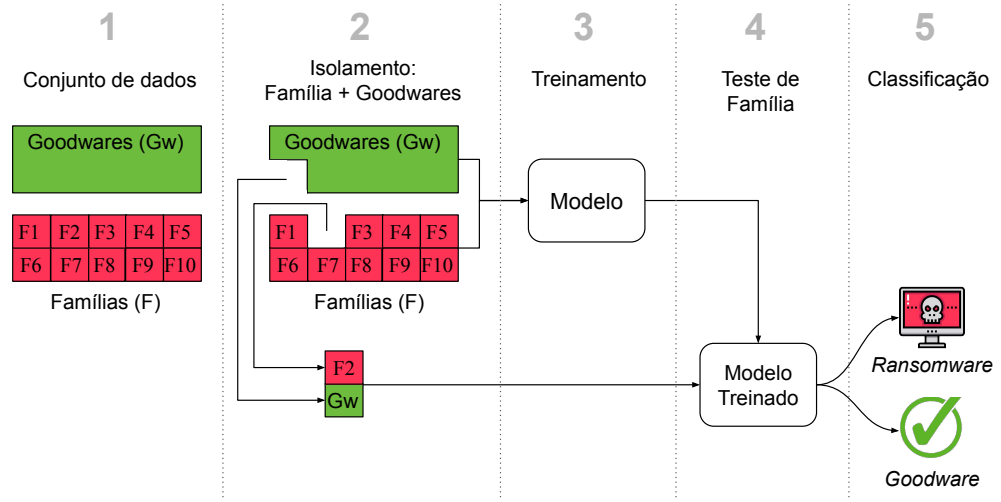
A aplicação do método VT pode promover uma representação mais compacta e eficiente dos dados, preservando as características mais discriminativas. Essa abordagem facilita a interpretação e análise dos dados, além de potencialmente melhorar o desempenho dos algoritmos de ML, por meio da redução da dimensionalidade do problema.

3.4 Metodologia de Avaliação de Detecção de Novas Famílias de *Ransomware*

Um caso particular de detecção de ataque de *ransomware* de dia zero é a DNFR. Essa capacidade é crucial para qualquer sistema de proteção. A Figura 4 esquematiza a metodologia adotada para a avaliação de DNFR nesta tese. Inicialmente, o conjunto de dados é constituído por *goodwares* e diversas famílias de *ransomware* previamente categorizadas (Passo 1). A avaliação de uma família específica envolve seu isolamento das demais para formar o conjunto de teste, ao qual são adicionadas amostras de *goodware* selecionadas aleatoriamente em igual

quantidade às amostras da família em avaliação (Passo 2). O restante do conjunto de dados é utilizado como entrada para o treinamento do modelo (Passo 3). Posteriormente, as amostras da família em avaliação são submetidas ao modelo treinado (Passo 4), o que resulta na classificação binária entre ransomware e *goodware* (Passo 5).

Figura 4 – Metodologia de Avaliação de Detecção de Novas Famílias de *Ransomware*



Fonte: Elaborado pelo autor

Devido à dificuldade de se coletar e categorizar *ransomwares* com critérios objetivos, algumas variantes podem possuir menos de 15 amostras; consequentemente, o conjunto de teste é completado com amostras de *goodware* até a quantidade de 30, desta forma, o resultado será estatisticamente relevante e pode-se utilizar o *Z-score* para calcular o Intervalo de Confiança (IC) (Aityan, 2022). O processo de avaliação é repetido dez vezes para cada família para obter os resultados médios, que correspondem ao desempenho completo do modelo para DNFR.

3.5 Avaliação e Comparação de Performance dos Modelos de Classificação

Para avaliar o desempenho de um algoritmo de ML para classificação em um determinado conjunto de dados, é preciso medir o quão bem as predições realizadas pelo modelo realmente correspondem aos dados observados. Para um classificador dicotômico, as classes alvo podem ser definidas como positivas e negativas. Nesta tese, a classe positiva é definida como a classe de *ransomware* e a classe negativa como a de *goodware*. Diferentes métricas avaliam diferentes propriedades da performance do classificador, cada uma com um propósito interpretativo dos resultados. Em geral, essas métricas de avaliação podem ser descritas como ferramentas de medição de desempenho do classificador e podem ser interpretadas como relações baseadas em (Hossin; Sulaiman, 2015; James et al., 2021):

- *True Positive* (TP): número de amostras de *ransomware* classificadas corretamente como *ransomware*;

- *True Negative* (TN): número de amostras de *goodware* classificadas corretamente como *goodware*;
- *False Positive* (FP): número de amostras de *goodware* classificadas incorretamente como *ransomware*; e
- *False Negative* (FN): número de amostras de *ransomware* classificadas incorretamente como *goodware*.

A métrica mais básica é a taxa de acurácia geral, que reflete a concordância entre as classes observadas e previstas e tem a interpretação mais direta. A acurácia é definida como o número total de amostras classificadas corretamente dividido pelo número total de amostras, dada por

$$\textit{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

Métricas como precisão e *recall* interpretam o resultado ao indicar o tipo de erro do modelo. A precisão, definida por

$$\textit{Precisão} = \frac{TP}{TP + FP}, \tag{8}$$

indica a capacidade do classificador de não rotular uma amostra positiva como negativa. Já o *recall*, dado por

$$\textit{Recall} = \frac{TP}{TP + FN}, \tag{9}$$

avalia a capacidade do classificador de encontrar todas as amostras positivas, ou seja, para esta tese, a habilidade de prever corretamente a classe *ransomware* (Hossin; Sulaiman, 2015).

Embora as métricas anteriores sejam facilmente interpretadas e indiquem os tipos de erros cometidos, também é comum a avaliação com a métrica *F-measure*, que combina os resultados de precisão e *recall*. Essa métrica é definida por

$$F_\beta = (1 + \beta^2) \times \left(\frac{\textit{Precisão} \times \textit{Recall}}{(\beta^2 \times \textit{Precisão}) + \textit{Recall}} \right), \tag{10}$$

onde β é um fator real positivo que é escolhido de tal forma que indica que o *recall* é β vezes mais significativo que a precisão na avaliação do modelo (Bekkar; Djemaa; Alitouche, 2013).

Nesta tese, a métrica *F-measure* será utilizada como a média harmônica entre *recall* e precisão, ou seja, terão o mesmo peso, logo, $\beta = 1$. Esta configuração também é conhecida como *F-measure* balanceada, simbolizada por F_1. A implicação de *F-measure* para $\beta = 1$ é dada por

$$F_{\beta=1} = 2 \times \left(\frac{\textit{Precisão} \times \textit{Recall}}{\textit{Precisão} + \textit{Recall}} \right), \tag{11}$$

e desta forma, o resultado pode ser interpretado como o desempenho geral do modelo (Bekkar; Djemaa; Alitouche, 2013).

Para uma abrangente avaliação da performance de um modelo de ML, é necessária a interpretação combinada dos resultados de diversas métricas, pois fornecem dados de diferentes ângulos sobre o desempenho do modelo (Powers, 2011; Bekkar; Djemaa; Alitouche, 2013). Além disso, nesta tese, os resultados são apresentados com um IC de 95%. Ao apresentar resultados comparativos, números em negrito indicam os resultados mais altos.

Para a comparação de métricas de desempenho, testes estatísticos são frequentemente utilizados. Na análise comparativa dos resultados dos conjuntos de características nos modelos de ML para a DNFR, assim como na comparação entre o modelo proposto e suas técnicas constituintes, são conduzidos dois testes estatísticos não paramétricos: Friedman e Nemenyi. Inicialmente, o teste de Friedman é conduzido considerando a hipótese nula H_0, que sugere que os resultados comparados são estatisticamente iguais, e a hipótese alternativa H_1, que indica que eles são diferentes. Nesta tese, é utilizado um nível de confiança de 95% ($\alpha = 0,05$), o qual indica que a relação é significativa e a hipótese nula deve ser rejeitada se o valor observado (*p-value*) for menor que $0,05$. Valores acima de $0,05$ indicam que a hipótese nula não pode ser rejeitada (Demšar, 2006).

Se a hipótese nula (H_0) for rejeitada, será realizado o teste *post-hoc* de Nemenyi para identificar diferenças significativas no desempenho dos grupos (conjuntos de características ou classificadores) entre si. O desempenho de dois grupos comparados é significativamente diferente se os *ranks* médios dos seus resultados diferirem em pelo menos a Distância Crítica (DC), dada por

$$DC = q_\alpha \sqrt{\frac{c(c+1)}{6M}} \tag{12}$$

onde q_α é o valor crítico definido segundo a estatística de *Studentized range* dividida por $\sqrt{2}$ (os valores podem ser encontrados no estudo de Demšar (2006)), c é a quantidade de grupos e M é o número de métricas em c (Demšar, 2006; Kaur; Kaur, 2021).

3.6 Modelos Experimentais

Para cumprir o objetivo desta tese, é necessário identificar os conjuntos de características mais significativas e investigar a possibilidade de melhorar o desempenho de DNFR ao combiná-las. Uma abordagem para enfrentar esse desafio é explorar técnicas tradicionais de ML e DL, de diferentes formas de aprendizado, que podem fornecer percepções valiosas sobre sua adequação para detectar novas variantes de *ransomware*. Para isso, foram selecionados nove algoritmos de aprendizado empregados em diversos estudos para detecção de *ransomware* e outros *malwares*, incluindo (Bae; Lee; Im, 2020; Liu et al., 2020), e que obtiveram bons resultados: *Extra-Trees* (EXT), KNN, *Linear Discriminant Analysis* (LDA), LR, *Multilayer Perceptron* (MLP), RF, *Stochastic Gradient Descent* (SGD), *Support Vector Machine* (SVM) e XGB.

Essas técnicas estão entre as categorias de aprendizado supervisionado mais comumente utilizadas, como árvores de decisão, regressão linear, redes neurais, métodos em conjunto e aprendizado baseado em instâncias. Elas são amplamente aplicadas para classificação binária, e cada algoritmo possui um mecanismo interno único e método de manipulação de dados, assim como vantagens e desvantagens, o que proporciona uma exploração abrangente dos dados (Ray, 2019; Bae; Lee; Im, 2020; Liu et al., 2020).

Para ajustar os hiperparâmetros dos modelos experimentais, foi empregado o método *GridSearchCV* do *Scikit-learn* com $K = 10$ e a métrica F_1 como estratégia de pontuação. Os intervalos de hiperparâmetros investigados para cada modelo experimental estão detalhados na Tabela 16 no Apêndice B. Além disso, os hiperparâmetros finais, tanto os ajustados quanto os padrões, utilizados na abordagem de DNFR com melhor desempenho para cada conjunto de características, são apresentados na Tabela 17 do Apêndice C.

3.7 Interpretação de Resultados

Métodos de interpretação para ML, especialmente para modelos EL, são essenciais para compreender suas decisões. Eles podem identificar características-chave e detectar vieses ou inconsistências. Essas percepções aumentam a confiança nesses modelos para uma implementação bem-sucedida no mundo real (Barredo Arrieta et al., 2020; Mi; Li; Zhou, 2020).

Para examinar o impacto das *features* em arquivos PE para a DNFR e estabelecer confiança no processo de tomada de decisão do modelo proposto, esta tese utiliza o método de interpretação *SHapley Additive exPlanations* (SHAP). Esse método fornece explicações compreensíveis para o processo de tomada de decisão do modelo ao considerar as variáveis de entrada.

O SHAP é baseado no conceito de 'valores de Shapley' da Teoria dos Jogos cooperativos, onde esses valores garantem uma distribuição equitativa da contribuição entre os jogadores em um jogo. O valor de Shapley para um jogador é calculado como a média de todas as contribuições marginais que o jogador faz ao jogo, considerando todas as combinações (coalizões) possíveis de jogadores. Os valores de Shapley baseiam-se na ideia de que o resultado de cada combinação possível de jogadores deve ser considerado para determinar a importância de um único jogador. Matematicamente, o valor de Shapley é dado por

$$\phi_g(val) = \sum_{C \subseteq \{1,\ldots,G\} \setminus \{g\}} \frac{|C|!(G - |C| - 1)!}{G!} [val(C \cup \{g\}) - val(C)] \tag{13}$$

onde g representa um jogador em um jogo com G jogadores, C é o subconjunto de combinações que não incluem o jogador g, ou seja, contém todas as coalizões para as quais o jogador g é capaz de fazer uma contribuição, $|C|!$ é o número de maneiras que os jogadores podem combinar em C *antes* do jogador g, $(G-|C|-1)!$ é o número de maneiras que os jogadores podem combinar em

C *depois* do jogador g, $G!$ é o número de maneiras de formar coalizões de G jogadores. Assim, $\frac{|C|!(G-|C|-1)!}{G!}$ indica o peso da contribuição marginal do jogador g. Além disso, $val(C \cup \{g\})$ é o valor da coalizão C incluindo o jogador g e $val(C)$ é o valor da coalizão de C. Desta forma, $[val(C \cup \{g\}) - val(C)]$ é a contribuição marginal do jogador g para a combinação C. Portanto, $\phi_g(val)$ é o somatório das contribuições marginais ponderadas de todas as coalizões que um jogador g pode participar (Lundberg; Lee, 2017; Lundberg; Erion; Lee, 2019; Molnar, 2022).

O SHAP utiliza os valores de Shapley na interpretação de modelos de ML ao quantificar a contribuição de cada valor de *feature* (jogador) para a predição (jogo) feita pelo modelo. No entanto, o cálculo exato dos valores de Shapley para conjuntos de dados extensos é computacionalmente oneroso devido à complexidade computacional do cálculo exaustivo para cada possível coalizão de *features*. Assim, o SHAP emprega técnicas de aproximação e amostragem para reduzir a carga computacional desses cálculos. Esse método foi implementado com a biblioteca *shap*[10] em Python, uma ferramenta versátil para compreensão e depuração de modelos de ML e DL.

Uma das estratégias empregadas pelo SHAP para reduzir a demanda computacional é a aproximação por meio de amostragem de Monte-Carlo, dada por

$$\hat{\phi}_l = \frac{1}{B} \sum_{b=1}^{B} \left(\hat{f}(w_{+l}^b) - \hat{f}(w_{-l}^b) \right) \tag{14}$$

onde B é o número de amostras Monte-Carlo, $\hat{f}(w_{+l}^b)$ é a predição do modelo quando a *feature* l está presente e $\hat{f}(w_{-l}^b)$ é a predição do modelo quando a *feature* l está ausente, logo, $\hat{\phi}_l$ é a estimativa aproximada do valor de Shapley para a *feature* l. Essa fórmula representa o cálculo da contribuição marginal média da feature l sobre B amostras. Em cada amostra, a feature l é presente ou ausente, e a diferença nas predições ($\hat{f}(w_{+l}^b) - \hat{f}(w_{-l}^b)$) é calculada para medir o impacto dessa feature específica. A média dessas diferenças, ponderada pelo inverso do número de amostras ($\frac{1}{B}$), fornece uma estimativa do valor de Shapley para a feature l. A aleatoriedade introduzida pelas amostras Monte-Carlo ajuda a capturar a variabilidade nas contribuições marginais, o que proporciona uma boa estimativa dos valores de Shapley (Molnar, 2022; Mitchell et al., 2022).

O método SHAP tem recebido significativa atenção no campo da cibersegurança em estudos relacionados à detecção de *malware* e de intrusões, como observado em (Wang; Zheng et al., 2020; Hsupeng et al., 2022; Alani; Awad, 2022; Morcos et al., 2022; Manthena et al., 2023) e, particularmente, em detecção de *malwares* ofuscados, como demonstrado em (Alani; Mashatan; Miri, 2023; Greco et al., 2023).

O SHAP possui várias propriedades notáveis que contribuem para sua ampla aplicabi-

[10] https://github.com/shap/shap

lidade. Em primeiro lugar, ele é independente do modelo, o que significa que pode ser aplicado a qualquer modelo de aprendizado, independentemente de suas características específicas. Em segundo lugar, o SHAP fornece atribuições de características individualizadas, permitindo uma compreensão detalhada da contribuição de cada característica para as previsões do modelo. Essa granularidade facilita a identificação de características influentes e seu impacto no processo de tomada de decisão dos modelos (Lundberg; Lee, 2017; Lundberg; Erion; Lee, 2019; Mi; Li; Zhou, 2020).

Além disso, o SHAP é fundamentado em princípios da teoria dos jogos, o que garante equidade e consistência nas atribuições de características em diferentes combinações. Isso garante resultados de interpretação confiáveis e reproduzíveis. Ainda, o SHAP considera os efeitos de interação entre *features*, permitindo avaliar como as interações entre elas influenciam a saída do modelo. Essa propriedade é particularmente valiosa para modelos de aprendizado que integram vários modelos individuais com graus variados de relevância das características. Portanto, a adoção do SHAP como método de interpretação para modelos de EL fornece percepções abrangentes sobre seu comportamento, melhorando a transparência, confiança e segurança na tomada de decisões (Lundberg; Lee, 2017; Lundberg; Erion; Lee, 2019; Mi; Li; Zhou, 2020).

4 RESULTADOS EXPERIMENTAIS

Este capítulo apresenta as etapas de pré-processamento e seleção de características aplicadas aos conjuntos de dados. Em seguida, compara os resultados obtidos pelas metodologias de CV e DNFR (ver Seção 3.4) para cada conjunto de características. Posteriormente, são comparados os resultados dos conjuntos de características individuais mais promissores com uma estratégia de conjuntos combinados. Para aprimorar os resultados obtidos, é empregado um modelo de EL de votação ponderada (*soft voting*), no qual são apresentados os resultados individuais de detecção das famílias. Por fim, são identificadas e analisadas as características mais impactantes que diferenciam *ransomware* de *goodware*, bem como as decisões de classificação incorretas do modelo. O ambiente utilizado foi o *Google Colab* equipado com um processador *Intel® Xeon®* de dois núcleos @2,20GHz, 12 GB de RAM e 225 GB de disco rígido.

4.1 Pré-processamento e Seleção de Características

Inicialmente, foi observado que determinadas APIs têm nomes semelhantes, diferindo apenas por certas letras no início, no fim ou em ambas as extremidades de seus nomes. De acordo com Hampton, Baig e Zeadally (2018) e Microsoft (2021a,b), um 'A' ao final do nome indica a versão da API que aceita strings de entrada codificadas em ANSI e um 'W' indica uma variante que aceita Unicode. A adição de 'Ex' no final indica uma nova versão dessa função, com um padrão de chamada diferente. A adição de 'Nt' ou 'Zw' no início do nome pode indicar uma diferença no tratamento dos valores do parâmetro que o chamador passa para a rotina. No geral, essas variações não alteram o comportamento dessas APIs. Em outras palavras, essas variações são formas diferentes de chamar a mesma funcionalidade subjacente.

Desta forma, foram mescladas as diferentes versões identificadas das mesmas APIs, as quais, se consideradas separadamente, poderiam ser mais facilmente removidas durante o processo de seleção de características. Foram encontradas 725 correspondências; portanto, o número de chamadas de funções diminuiu de 9961 para 9236.

Em seguida, as características extraídas foram normalizadas para o intervalo de 0,0 a 1,0. Essa normalização foi aplicada a três conjuntos de características: 1) Campos do Cabeçalho, utilizando seus valores mínimos e máximos correspondentes; 2) Entropia das Seções, cujos valores de entropia variam de 0,0 a 8,0; e 3) N-gramas de *opcodes*, onde a frequência dos N-gramas varia dependendo dos valores mínimos e máximos. A obtenção dos valores mínimos e máximos para cada característica foi realizada apenas com base nas amostras do conjunto de treinamento.

Depois, foi utilizado o método VT para a seleção de *features*. A Tabela 4 resume os resultados do processo de seleção aplicado a cada conjunto de *features*. O tipo de limiar aplicado

pelo método VT (número real ou porcentagem) varia de acordo com o tipo de *feature* (Ponto Flutuante ou Binária), conforme descrito na Seção 3.3 nas Equações 5 e 6.

Tabela 4 – Resultados da aplicação do método de redução de dimensionalidade VT aos conjuntos de características extraídas.

Características	Tipo	Valor VT	# Antes	# Depois	% Redução
Campos do Cabeçalho	Ponto Flutuante	$thr = 0,0$	84	77	8,33
DLLs Importadas	Binária	$p = 99,9\%$	576	275	47,74
Chamadas de Função	Binária	$p = 99\%$	9236	1084	88,26
Entropia das Seções	Ponto Flutuante	-	131	131	0,0
Opcodes N-grama	Ponto Flutuante	-	1000	1000	0,0
Total			11 027	2567	76,72

Fonte: Elaborado pelo autor

No conjunto Campos de Cabeçalho, foi aplicada uma seleção com um valor limiar VT de $thr = 0,0$ para eliminar características com valor único, o que reduziu número de 84 para 77. Para o conjunto DLLs Importadas, foi aplicado um valor limiar VT de $p = 99,9\%$, o que resultou em uma redução substancial de 47,74%. Considerando um total de 2157 amostras, ou seja, cada característica possui 2157 instâncias, essa redução foi alcançada ao descartar características que apresentavam duas ou menos ocorrências com valor 0, ou duas ou menos ocorrências com o valor 1. No conjunto Chamadas de Função, foi aplicada uma seleção com um valor limiar VT de $p = 99\%$, o que reduziu significativamente o número de 9236 para 1084, uma redução de 88,26%. Essa redução foi alcançada ao descartar características que apresentavam 21 ou menos ocorrências com o valor 0, ou 21 ou menos ocorrências com o valor 1.

Experimentos com o conjunto Entropia das Seções mostraram que os melhores resultados foram obtidos sem a aplicação do VT. Por fim, os conjuntos *Opcodes* N-grama já foram selecionados com o método TF-IDF e, portanto, não exigiram uma seleção adicional. O número total de características foi reduzido de 11 027 para 2567, ou seja, uma diminuição de 76,72%.

4.2 Comparação dos Resultados das Metodologias de CV e DNFR

Para comparar os cenários de detecção de amostras de uma variante previamente conhecida, ou seja, onde há diferentes amostras de uma mesma família nos conjuntos de treino e teste, com o cenário de detecção de amostras de variantes desconhecidas, isto é, onde não há amostras da família em teste no conjunto de treinamento, foram conduzidos testes de CV estratificado (cenário de família previamente conhecida), e testes de DNFR (cenário de família desconhecida).

A Tabela 5 apresenta o desempenho dos nove modelos de ML, pela métrica *F-measure* (F_1), aplicados a cada conjunto de *features* com a abordagem de CV estratificado. De forma geral, os resultados dos melhores modelos para cada conjunto foram considerados bons, com desempenho acima de 97%. Destaca-se que o conjunto de *features* Campos de Cabeçalho obteve o maior desempenho global, alcançando um F_1 de 99,13% quando RF foi utilizado. Além disso, os conjuntos Entropia das Seções e os de *opcodes* com N-gramas também apresentaram resultados superiores a 98% pela técnica EXT. Já os conjuntos DLLs Importadas e Chamadas de Função obtiveram os melhores desempenhos de 97,08% e 97,87% respectivamente, com XGB e SGD.

Tabela 5 – Comparação de desempenho de nove modelos de ML e DL para detecção de *ransomware* para a abordagem de CV com $K = 10$, utilizando conjuntos de características diferentes. F_1 (%) $\pm$ 0,95 IC (%).

Modelo	Campos de Cabeçalho	DLLs Importadas	Chamadas de Função	Entropia das Seções	*Opcode* 3-grama	*Opcode* 4-grama	*Opcode* 5-grama
EXT	99,06±0,09	97,01±0,15	97,55±0,14	**98,55±0,10**	**98,40±0,12**	**98,35±0,13**	**98,29±0,12**
KNN	98,60±0,47	95,87±0,63	96,92±0,18	97,60±0,58	97,26±0,21	97,49±0,18	96,59±0,29
LDA	96,03±0,85	93,57±0,60	84,23±0,37	93,75±0,82	89,06±0,35	89,08±0,41	88,92±0,43
LR	97,08±0,74	95,42±0,50	97,69±0,17	94,15±0,94	93,93±0,36	94,83±0,35	93,96±0,33
MLP	97,23±0,83	96,99±0,17	97,64±0,15	96,65±0,17	97,00±0,20	97,45±0,14	96,77±0,16
RF	**99,13±0,09**	97,04±0,16	97,76±0,14	98,13±0,13	97,93±0,13	97,97±0,13	97,58±0,15
SGD	92,03±2,53	95,73±0,45	**97,87±0,17**	93,73±0,87	94,45±0,57	95,75±0,31	96,11±0,29
SVM	96,79±0,82	96,32±0,49	97,60±0,17	94,61±0,88	94,33±0,34	95,17±0,29	94,88±0,35
XGB	99,08±0,09	**97,12±0,16**	97,69±0,14	97,57±0,14	98,05±0,14	97,8±0,13	97,95±0,15

Fonte: Elaborado pelo autor

Observa-se que os modelos EXT, RF e XGB apresentaram consistentemente bons desempenhos, superiores a 97%, em todos os conjuntos de características. Esses modelos utilizam árvores de decisão em conjunto, o que sugere que essa categoria de modelos é adequada para testar novas amostras de *ransomware* de famílias conhecidas, assim como o modelo MLP, que também apresentou um desempenho consistente de aproximadamente 97%. Em contraste, os algoritmos LDA e SGD não apresentaram desempenho consistente em todos os conjuntos de características. O resultado geral mais baixo, de 84,23%, foi obtido com o conjunto Chamadas de Função com a técnica LDA.

A Tabela 6 exibe o desempenho de nove modelos, avaliado pela métrica F_1, bem como o *rank* médio dos resultados obtidos em cada conjunto de características, com a metodologia de teste para a DNFR. Esse cenário representa o teste de detecção de amostras de famílias desconhecidas, pois não há amostras da variante em teste no conjunto de treinamento. Os resultados revelam um desempenho geral inferior em comparação com a abordagem de CV, evidenciando

a dificuldade do cenário de DNFR. No conjunto Campos de Cabeçalho, o modelo XGB obteve a maior performance de 97,24%, que é aproximadamente 2% inferior ao melhor resultado obtido por meio de CV com o mesmo modelo. Para os conjuntos DLLs Importadas e Chamadas de Função, a técnica SVM produziu os resultados mais altos, com pontuações de 95,07% e 96,13%, respectivamente, embora não tenha sido o melhor modelo para esses conjuntos na abordagem de CV. O modelo EXT superou os outros com um resultado de 94,30% ao utilizar o conjunto Entropia das Seções, embora o resultado tenha sido aproximadamente 4% inferior em comparação com a abordagem de CV.

Tabela 6 – Comparação de desempenho de nove modelos de ML e DL para a abordagem de DNFR, utilizando conjuntos de características diferentes. F_1 (%) $\pm$ 0.95 IC (%).

Modelo	Campos de Cabeçalho	DLLs Importadas	Chamadas de Função	Entropia das Seções	*Opcode* 3-grama	*Opcode* 4-grama	*Opcode* 5-grama
EXT	95,30±0,29	90,62±0,38	92,24±0,88	**94,30±0,28**	86,17±0,69	84,33±0,46	82,24±0,82
KNN	94,23±0,32	91,97±0,70	91,93±0,32	87,01±0,20	77,81±0,18	78,64±0,56	83,13±0,23
LDA	93,85±0,40	91,52±0,32	58,63±2,72	89,31±0,35	70,49±1,63	59,61±2,09	48,50±2,36
LR	89,96±0,60	91,83±0,31	95,03±0,56	90,93±0,34	86,58±0,35	84,57±0,41	71,86±0,54
MLP	87,47±0,53	92,18±0,37	93,55±0,78	88,02±0,91	83,38±1,30	79,67±1,88	81,06±1,70
RF	97,11±0,35	91,58±0,42	93,37±0,42	92,97±0,35	84,34±0,80	82,74±0,69	**87,69±0,79**
SGD	82,55±2,14	92,45±0,35	85,94±0,92	87,46±0,95	85,06±1,39	77,86±1,77	79,82±2,07
SVM	87,51±0,46	**95,07±0,29**	**96,13±0,31**	91,85±0,30	83,01±0,30	77,87±0,57	74,91±0,24
XGB	**97,24±0,17**	92,15±0,24	91,22±0,63	92,65±0,26	**89,31±0,68**	**88,81±0,91**	85,55±0,86
$\overline{Rank}$	2,56	2,44	2,67	2,78	5,00	6,22	6,33

Fonte: Elaborado pelo autor

Observa-se que as características de *opcodes* com N-gramas obtiveram resultados gerais não satisfatórios, com desempenho abaixo de 90%, embora tiveram bom desempenho na abordagem de CV (acima de 98%). Essa observação sugere que esse tipo de característica pode não contribuir significativamente para DNFR.

Além disso, foram observadas variações consideráveis no desempenho dos mesmos modelos em diferentes conjuntos de características. Por exemplo, enquanto o SVM alcançou uma alta performance de 96,13% ao utilizar o conjunto Chamadas de Função, seu desempenho foi consideravelmente inferior com Campos de Cabeçalho (87,51%), *Opcode* 3-grama (83,01%), *Opcode* 4-grama (77,87%) e *Opcode* 5-grama (74,91%).

Ao analisar a significância estatística dos resultados de cada conjunto de *features* da Tabela 6, a hipótese nula H_0 do teste de Friedman foi rejeitada com um *p-value* observado de $1,8 \times 10^{-}6$. Portanto, a hipótese alternativa H_1 sugere que houve diferença estatística signifi-

cativa entre os resultados para os conjuntos de *features*. No teste *post-hoc* de Nemenyi, a DC calculada entre as médias dos *ranks* foi de $3,00$. Isso indica que houve diferenças significativas entre os resultados dos conjuntos Campos de Cabeçalho $(2,56)$, DLLs Importadas $(2,44)$, Chamadas de Função $(2,67)$ e Entropia das Seções $(2,78)$ em relação aos conjuntos *Opcode* 4-grama $(6,22)$ e *Opcode* 5-grama $(6,33)$. Não foram observadas outras relações que diferem pela DC. Os resultados desta avaliação serão empregados na próxima seção.

4.3 Combinação dos Conjuntos de Características

Em busca de aprimorar os resultados de DNFR por meio de uma análise mais abrangente das características extraídas das amostras, foram combinados os conjuntos que apresentaram os melhores *ranks* médios dos resultados.

Entretanto, apesar de o resultado do conjunto *Opcode* 3-grama não ter revelado uma diferença estatística significativa em relação aos Campos de Cabeçalho, DLLs Importadas, Chamadas de Função e Entropia das Seções, seu *rank* médio $(5,00)$ está próximo da DC em relação aos outros conjuntos mencionados. Desta forma, para encontrar a melhor combinação de *features*, foram realizados experimentos com conjuntos combinados que incluíram e excluíram o *Opcode* 3-grama.

A Tabela 7 apresenta o desempenho em F_1 e o tempo médio de treinamento em segundos dos nove modelos experimentados ao combinar Campos de Cabeçalho, DLLs Importadas, Chamadas de Função e Entropia das Seções *com* e *sem* a inclusão do conjunto *Opcode* 3-grama. Os hiperparâmetros foram ajustados em ambas as abordagens para melhorar os resultados. Para a combinação que inclui o conjunto *Opcode* 3-grama, o modelo SVM obteve o maior resultado de 96,60% com um tempo de treinamento de 1,46s. Com a combinação que exclui o conjunto *Opcode* 3-grama, o modelo LR alcançou o desempenho geral mais alto de 97,93%, com um tempo de treinamento de 0,18s. Importante observar que o modelo LR obteve resultados insatisfatórios quando aplicado à maioria dos conjuntos de características individuais. No entanto, a combinação desses conjuntos resultou em um aumento no desempenho do modelo LR. Por fim, devido à ausência da etapa de treinamento em seu funcionamento, o algoritmo KNN apresentou o menor tempo de execução em ambas as combinações de conjuntos de características. Contudo, teve o segundo pior desempenho em termos da métrica avaliada.

Ao comparar as duas abordagens, a combinação *sem* o conjunto *Opcode* 3-grama apresentou desempenho superior em sete dos nove modelos. Essa constatação confirma que as características de *opcodes* com N-gramas não são adequadas para a DNFR. Além disso, o tempo de treinamento foi reduzido em todos os modelos devido à diminuição da dimensão do conjunto de dados e, consequentemente, da complexidade do problema. Portanto, o conjunto de características *Opcode* 3-grama foi descartado do modelo.

Ademais, ao comparar a abordagem combinada *sem* o *Opcode* 3-grama com os me-

Tabela 7 – Comparação de desempenho de nove modelos de ML e DL para a DNFR ao combinar os conjuntos de características Campos de Cabeçalho, DLLs Importadas, Chamadas de Função e Entropia das Seções *com* e *sem* a inclusão do conjunto *Opcode* 3-grama. F_1 (%) $\pm$ 0.95 IC (%).

Modelo	Combinado *com* 3-grama	Tempo Médio de Treinamento (s)	Combinado *sem* 3-grama	Tempo Médio de Treinamento (s)
EXT	92,48 $\pm$ 0,24	1,27	95,96 $\pm$ 0,23	1,18
KNN	93,56 $\pm$ 0,19	**0,06**	93,26 $\pm$ 0,27	**0,04**
LDA	65,25 $\pm$ 2,07	9,91	53,23 $\pm$ 3,31	4,83
LR	94,54 $\pm$ 0,14	1,12	**97,93 $\pm$ 0,21**	0,18
MLP	93,56 $\pm$ 0,12	23,07	95,42 $\pm$ 0,42	21,86
RF	92,04 $\pm$ 0,34	1,94	97,64 $\pm$ 0,35	0,73
SGD	92,76 $\pm$ 0,47	0,63	95,69 $\pm$ 0,75	0,45
SVM	**96,60 $\pm$ 0,27**	1,46	96,81 $\pm$ 0,31	0,82
XGB	94,67 $\pm$ 0,12	19,31	97,72 $\pm$ 0,30	4,03

Fonte: Elaborado pelo autor

lhores conjuntos de *features* individuais da Tabela 6, sete dos nove modelos aprimoraram o desempenho. Isso implica que a combinação das *features* estruturais amplifica os dados discriminativos extraídos de variantes de *ransomware* previamente conhecidas e dos *goodwares*, resultando em uma melhoria na detecção de famílias desconhecidas de *ransomware*.

4.4 Modelo de Análise Estrutural Combinada

Após a análise dos conjuntos de características, foram selecionadas três técnicas de ML que demonstraram desempenho superior para integrar um classificador conjunto de votação ponderada: LR, RF e XGB. Essas abordagens apresentaram uma diferença máxima de 0,29% em seus resultados. Assim, o modelo proposto é definido pela combinação das características estruturais dos campos de cabeçalho, DLLs importadas, chamadas de funções e entropia das seções, através de um classificador de votação ponderada composto pelos algoritmos LR, RF e XGB.

Para validar o desempenho do modelo proposto, a Tabela 8 apresenta seus resultados e dos algoritmos seus constituintes em termos de médias ponderadas de acurácia, precisão, *recall* e F_1 na avaliação do conjunto de treinamento para o cenário de DNFR.

O modelo proposto nesta tese utilizou as medidas de certeza preditiva de cada classificador individual para melhorar o desempenho em acurácia (98,41%), *recall* (97,94%) e F_1 (98,36%). A precisão também alcançou uma alta taxa de 98,89%, embora tenha apresentado uma diminuição de 0,53% em comparação ao modelo XGB. Contudo, ao analisar a significância estatística dos resultados, a hipótese nula H_0 do teste não paramétrico de Friedman não foi

Tabela 8 – Resultados dos três melhores modelos (LR, RF e XGB) e do modelo proposto utilizando os dados de *treinamento* com a metodologia de DNFR. Métrica (%) ± 0.95 IC (%).

Modelo	Acurácia	Precisão	*Recall*	F_1	Média de Tempo de Treinamento (s)
LR	97,98 ± 0,21	98,41 ± 0,39	97,57 ± 0,09	97,93 ± 0,21	0,18
RF	97,82 ± 0,30	98,81 ± 0,36	96,80 ± 0,31	97,64 ± 0,35	0,73
XGB	97,98 ± 0,27	**99,42 ± 0,23**	96,53 ± 0,36	97,72 ± 0,30	4,03
Modelo Proposto	**98,41 ± 0,20**	98,89 ± 0,32	**97,94 ± 0,08**	**98,36 ± 0,20**	5,21

Fonte: Elaborado pelo autor

rejeitada. Isso sugere que, para o conjunto de treinamento, os resultados dos classificadores são estatisticamente equivalentes. Por fim, embora o modelo proposto tenha o maior tempo médio de treinamento entre os métodos comparados, levando 5,21s, ainda é considerado um tempo baixo.

Para mitigar possíveis fontes de viés e vazamento de dados que possam ter surgido das etapas de pré-processamento de dados e ajuste de hiperparâmetros, uma avaliação foi conduzida ao treinar o modelo com as 25 famílias de *ransomware* previamente utilizadas e testá-lo com 15 famílias não observadas no treinamento. A Tabela 9 apresenta os resultados médios ponderados tanto para o modelo proposto quanto para seus algoritmos constituintes.

Tabela 9 – Resultados dos três melhores modelos (LR, RF e XGB) e do modelo proposto utilizando os dados de *teste* com a metodologia de DNFR. Métrica (%) ± 0.95 IC (%).

Modelo	Acurácia	Precisão	*Recall*	F_1	$\overline{Rank}$
LR	94,94 ± 0,48	93,8 ± 1,35	94,21 ± 0,00	93,23 ± 0,69	3,00
RF	92,21 ± 1,04	91,43 ± 2,14	86,86 ± 1,57	88,02 ± 1,88	4,00
XGB	95,33 ± 0,46	94,54 ± 1,25	94,62 ± 0,00	94,04 ± 0,67	2,00
Modelo Proposto	**97,53 ± 0,38**	**96,36 ± 1,09**	**97,52 ± 0,06**	**96,41 ± 0,58**	1,00

Fonte: Elaborado pelo autor

O modelo proposto apresentou um desempenho superior em comparação às técnicas individuais, o que destaca os benefícios na melhoria dos resultados métricos ao aproveitar as qualidades de cada método constituinte. O modelo proposto obteve acurácia de 97,53%, precisão de 96,36%, *recall* de 97,52% e F_1 de 96,41%. Ao comparar os resultados da Tabela 9 com os apresentados na Tabela 8, o modelo proposto mostrou uma diminuição de 0,88%, 2,53%, 1,42% e 1,95%, em acurácia, precisão, *recall* e F_1, respectivamente.

Ao analisar a significância estatística dos resultados, a hipótese nula H_0 do teste de

Friedman foi rejeitada com um *p-value* observado de $0,007$. Portanto, a hipótese alternativa H_1 sugere que, para o conjunto de teste, os resultados dos classificadores são estatisticamente diferentes.

No teste *post-hoc* de Nemenyi, a DC calculada entre as médias dos *ranks* foi de $2,34$. Isso indica que o modelo proposto $(1,00)$ comparado ao RF $(4,00)$ apresenta um desempenho significativamente melhor $(4,00 - 1,00 = 3,00 > 2,34)$, enquanto o LR (3,00) está próximo da DC $(3,00 - 1,00 = 2,00 \approx 2,34)$. Não foram observadas outras relações entre os modelos que diferem pela DC. Sendo assim, é possível concluir que há benefícios de desempenho estatisticamente significativos ao utilizar o modelo proposto.

As métricas alcançadas pela abordagem proposta no desafiador cenário de famílias mais recentes e desconhecidas fornecem evidências empíricas que respaldam sua eficácia. Além disso, indicam que a técnica aplicada tem o potencial de ser resiliente à evolução do *ransomware* ao longo do tempo, uma vez que a maioria das variantes de teste são mais recentes e, portanto, mais representativas das ameaças atuais.

4.4.1 Análise da Detecção de Novas Famílias de *Ransomware*: Desempenho Individual

A Tabela 10 apresenta os resultados da aplicação do modelo proposto para as famílias de treinamento na metodologia de DNFR. Conforme mencionado, a média ponderada de acurácia do método foi encontrada como 98,41%, com 21 das 25 variantes alcançando uma acurácia acima de 98%. A média ponderada de precisão foi de 98,89%, em que todas as famílias obtiveram resultados acima de 97,55%, o que indica uma baixa taxa de classificação incorreta de amostras de *goodware* como *ransomware* (FP). A média ponderada de *recall* foi de 97,94%, em que 18 das 25 variantes de *ransomware* foram 100% classificadas corretamente e 21 das 25 famílias obtiveram resultados acima de 97%, o que indica um baixo número de amostras de *ransomware* classificadas incorretamente como *goodware* (FN). A média ponderada de F_1 foi encontrada como 98,36%, com 21 das 25 variantes alcançando resultados acima de 98%, o que reflete um excelente desempenho geral do método.

Embora o modelo proposto tenha demonstrado alta eficácia geral na classificação de *ransomware* e *goodware*, houve algumas famílias com cujo desempenho do classificador ficou baixo. Especificamente, a variante Thanos apresentou uma taxa de *recall* de 81,71%, o que significa que aproximadamente seis de 35 amostras de *ransomware* foram classificadas erroneamente como *goodware* (FN). Da mesma forma, as famílias Conti, Makop e Zeppelin demonstraram *recall* de 92,08%, 91,71% e 89,80%, respectivamente. Isso significa que aproximadamente quatro de 48 (Conti), três de 35 (Makop) e cinco de 49 (Zeppelin) amostras de *ransomware* foram classificadas incorretamente como *goodware* (FN). Isso sugere que essas quatro famílias têm menos similaridades estruturais com as outras variantes e podem compartilhar características comumente encontradas em *goodware*, tornando-as desafiadoras de distinguir.

A Tabela 11 apresenta os resultados obtidos para cada família de *ransomware* no con-

Tabela 10 – Resultados do modelo proposto para as 25 famílias do conjunto de *treinamento* para a DNFR. Métrica (%) ± 0.95 IC (%).

Família	# Amostras (*Rans.-Good.*)	Acurácia	Precisão	*Recall*	F_1
Avaddon	49-49	99,49 ± 0,14	99,01 ± 0,26	100,0 ± 0,00	99,50 ± 0,13
Babuk	44-44	99,54 ± 0,16	99,12 ± 0,30	100,0 ± 0,00	99,55 ± 0,15
Blackmatter	45-45	99,33 ± 0,18	98,71 ± 0,35	100,0 ± 0,00	99,35 ± 0,18
Conti	48-48	95,73 ± 0,15	99,33 ± 0,20	92,08 ± 0,17	95,57 ± 0,15
Darkside	50-50	99,80 ± 0,08	99,61 ± 0,15	100,0 ± 0,00	99,80 ± 0,08
Dharma	46-46	99,57 ± 0,15	99,16 ± 0,28	100,0 ± 0,00	99,57 ± 0,14
Doppelpaymer	23-23	98,91 ± 0,31	97,92 ± 0,60	100,0 ± 0,00	98,94 ± 0,31
Exorcist	19-19	98,68 ± 0,56	97,55 ± 1,03	100,0 ± 0,00	98,73 ± 0,54
Gandcrab	50-50	99,20 ± 0,15	98,45 ± 0,28	100,0 ± 0,00	99,21 ± 0,14
Lockbit	48-48	99,69 ± 0,13	99,40 ± 0,26	100,0 ± 0,00	99,69 ± 0,13
Makop	35-35	95,14 ± 0,48	98,46 ± 0,36	91,71 ± 0,70	94,95 ± 0,51
Maze	50-50	99,50 ± 0,18	99,04 ± 0,34	100,0 ± 0,00	99,51 ± 0,18
Mountlocker	14-16	99,67 ± 0,36	99,33 ± 0,72	100,0 ± 0,00	99,66 ± 0,37
Nefilim	39-39	98,21 ± 0,19	98,99 ± 0,37	97,44 ± 0,00	98,20 ± 0,18
Netwalker	50-50	99,30 ± 0,20	98,66 ± 0,37	100,0 ± 0,00	99,32 ± 0,19
Phobos	50-50	98,30 ± 0,18	98,62 ± 0,34	98,00 ± 0,00	98,30 ± 0,17
Pysa	38-38	99,61 ± 0,14	99,23 ± 0,26	100,0 ± 0,00	99,61 ± 0,13
Ragnarok	43-43	99,19 ± 0,16	98,42 ± 0,30	100,0 ± 0,00	99,20 ± 0,15
Ransomexx	13-17	100,0 ± 0,00	100,0 ± 0,00	100,0 ± 0,00	100,0 ± 0,00
Revil	49-49	99,39 ± 0,10	98,80 ± 0,19	100,0 ± 0,00	99,39 ± 0,10
Ryuk	48-48	99,48 ± 0,17	98,99 ± 0,32	100,0 ± 0,00	99,49 ± 0,16
Stop	48-48	98,44 ± 0,19	98,98 ± 0,37	97,92 ± 0,00	98,44 ± 0,19
Thanos	35-35	90,00 ± 0,70	98,00 ± 0,52	81,71 ± 1,38	88,99 ± 0,82
Wastedlocker	40-40	99,50 ± 0,18	99,04 ± 0,35	100,0 ± 0,00	99,51 ± 0,18
Zeppelin	49-49	94,39 ± 0,10	98,89 ± 0,22	89,80 ± 0,00	94,12 ± 0,10
Média Ponderada		98,41 ± 0,20	98,89 ± 0,32	97,94 ± 0,08	98,36 ± 0,20

Fonte: Elaborado pelo autor

junto de teste. O modelo proposto demonstrou acurácia superior a 97% em 14 das 15 famílias e precisão acima de 97% em 10 famílias. Destaca-se o *recall* com 100% em 13 famílias e, por fim, F_1 acima de 97% em 11 famílias.

A métrica de precisão revelou resultados insatisfatórios nas famílias HolyGhost, Maui e Quantum. Embora essas famílias tenham sido corretamente classificadas com 100% (*recall*), a escassez de amostras de *ransomware* no conjunto de teste exigiu a inclusão de várias amostras de *goodware* para avaliação, o que aumenta a probabilidade de um resultado FP e seu respectivo

Tabela 11 – Resultados do modelo proposto para as 15 famílias do conjunto de *teste* para a DNFR. Métrica (%) ± 0.95 IC (%).

Família	# Amostras (*Rans.-Good.*)	Acurácia	Precisão	*Recall*	F_1
AvosLocker	50-50	98,78 ± 0,18	97,65 ± 0,34	100,0 ± 0,00	98,80 ± 0,17
BianLian	11-19	98,33 ± 0,78	95,95 ± 1,86	100,0 ± 0,00	97,86 ± 0,99
Black Basta	32-32	98,53 ± 0,34	97,22 ± 0,63	100,0 ± 0,00	98,57 ± 0,33
BlackByte	13-17	72,00 ± 0,81	93,38 ± 3,46	38,62 ± 0,55	54,50 ± 0,85
BlackCat	50-50	98,86 ± 0,18	97,80 ± 0,35	100,0 ± 0,00	98,88 ± 0,18
BlueSky	34-34	98,82 ± 0,28	97,75 ± 0,52	100,0 ± 0,00	98,85 ± 0,27
Clop	46-46	97,33 ± 0,26	97,64 ± 0,35	97,02 ± 0,36	97,32 ± 0,27
Hive	50-50	98,79 ± 0,18	97,67 ± 0,34	100,0 ± 0,00	98,81 ± 0,18
HolyGhost	4-26	97,87 ± 0,88	88,21 ± 4,54	100,0 ± 0,00	93,23 ± 2,68
Karma	13-17	98,80 ± 0,64	97,46 ± 1,35	100,0 ± 0,00	98,67 ± 0,71
Lorenz	16-16	98,81 ± 0,57	97,78 ± 1,06	100,0 ± 0,00	98,85 ± 0,55
Maui	3-27	97,60 ± 0,94	84,00 ± 5,76	100,0 ± 0,00	90,44 ± 3,54
NightSky	14-16	98,67 ± 0,75	97,40 ± 1,44	100,0 ± 0,00	98,64 ± 0,76
PlayCrypt	43-43	98,55 ± 0,22	97,22 ± 0,42	100,0 ± 0,00	98,58 ± 0,22
Quantum	6-24	98,00 ± 0,81	91,81 ± 3,15	100,0 ± 0,00	95,50 ± 1,77
Média Ponderada		97,53 ± 0,38	96,36 ± 1,09	97,52 ± 0,06	96,41 ± 0,58

Fonte: Elaborado pelo autor

impacto na métrica de precisão. Por exemplo, no caso do teste da família HolyGhost, que possui apenas quatro amostras de *ransomware* e 26 amostras de *goodware*, a primeira classificação errada de *goodware* como *ransomware* (FP) reduz a precisão em 20%. A precisão de 88,21% para a família HolyGhost indica uma média de ≈0,53 amostras de *goodware* classificadas erroneamente.

A métrica de *recall* demonstrou consistentemente um alto desempenho, o que enfatiza que é improvável que o *ransomware* seja classificado erroneamente como *goodware*, o que representa o pior cenário. No entanto, a família BlackByte apresentou uma taxa de detecção baixa, com uma média de aproximadamente oito das 13 amostras de *ransomware* sendo classificadas erroneamente. Será realizada uma análise mais aprofundada dessa variante em particular para entender as características que a tornam semelhante ao *goodware*. A família Clop foi classificada erroneamente em uma média de aproximadamente de apenas uma das 46 amostras, resultando em um total de aproximadamente nove classificações incorretas de *ransomware* em 385 amostras do conjunto de teste.

Adicionalmente, foram realizadas medições de tempo para extração de características, pré-processamento e predição de amostras, de forma individual, com o modelo proposto já

treinado. Essas medições simulam um *host* em que um único arquivo PE é verificado ao ser recebido ou antes da execução. O processo de verificação (*scanning*) e predição foi concluído em uma média de 0,31s e 0,06s, respectivamente, totalizando 0,37s por amostra. Esse resultado indica que o modelo proposto pode ser aplicado a dispositivos de *hardware* com recursos limitados para aplicações em tempo de execução.

Ainda, o modelo treinado é compacto, pois ocupa apenas 2,3 MB. Essa eficiência é especialmente relevante para implantações em grande escala, o que garante que o modelo proposto possa ser implementado de maneira rápida, pois facilita a distribuição de atualizações pelos servidores centrais de antivírus.

4.4.2 Análise de Impacto das Características

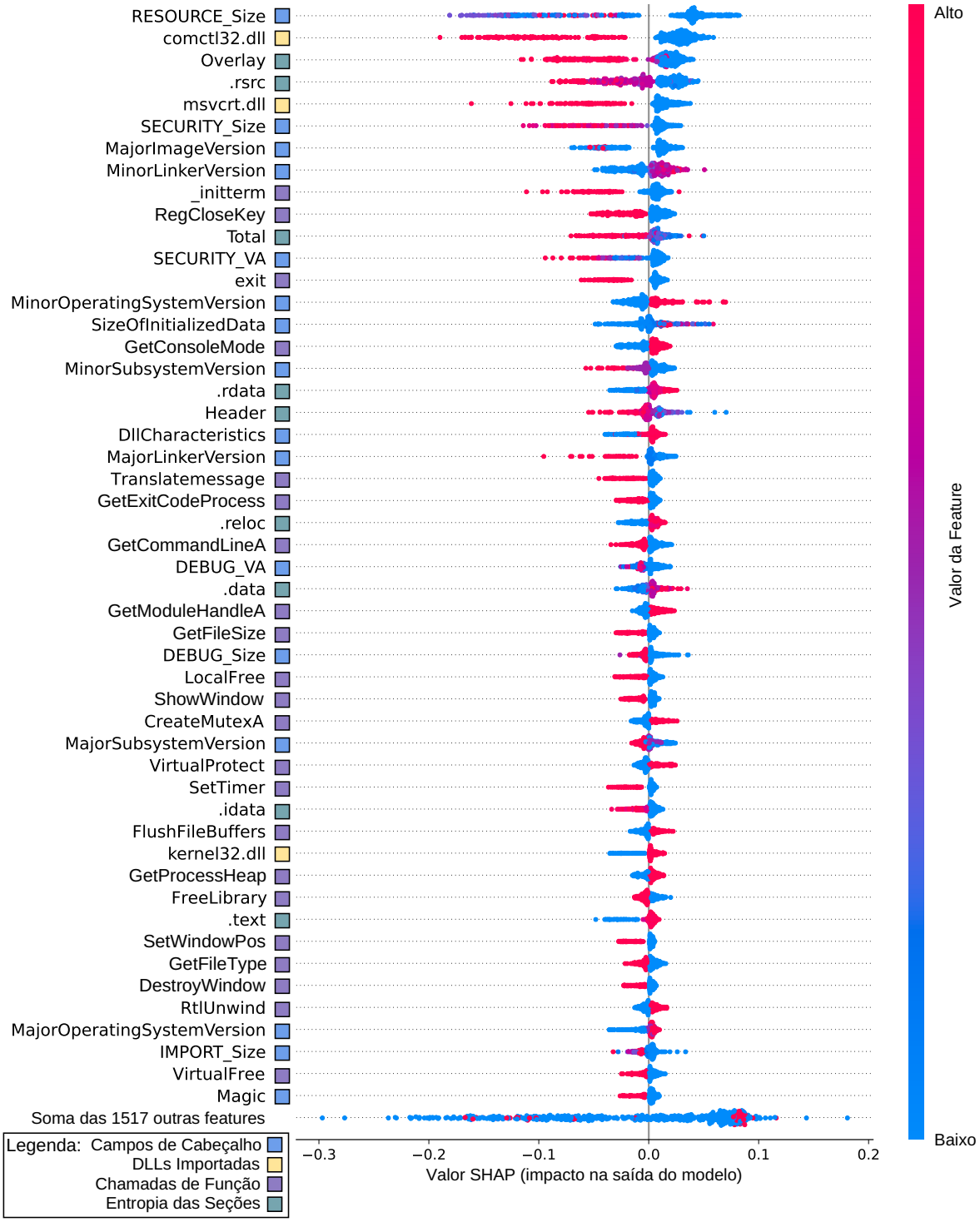
Para compreender melhor e estabelecer confiança no processo de tomada de decisão do modelo proposto, o modelo foi treinado com o conjunto de dados de treinamento e, em seguida, foi aplicada a técnica de interpretação SHAP para todas as amostras do conjunto de testes.

A Figura 5 ilustra um gráfico SHAP do tipo *beeswarm* apresentando as 50 principais características, organizadas em ordem decrescente de acordo com seu impacto no processo de tomada de decisão, juntamente com o impacto cumulativo das outras 1517 características restantes. A predição do modelo atribui o valor 1 para *ransomware* e 0 para *goodware*. Os valores à esquerda do eixo vertical influenciam a predição em direção ao *goodware*, enquanto os valores à direita a empurram em direção ao *ransomware*. Os pontos vermelhos indicam valores altos das características, enquanto os pontos azuis indicam valores baixos. Os conjuntos DLLs Importadas e Chamadas de Função consistem em características binárias, representadas pelas cores vermelho e azul, respectivamente. Por outro lado, Campos de Cabeçalho e Entropia das Seções consistem em características de ponto flutuante, o que resulta em uma transição de cores. Para cada característica, cada ponto no gráfico representa uma amostra do conjunto de testes. Em regiões de alta densidade de pontos, eles são empilhados verticalmente.

Dentre as 50 características mais impactantes no processo de tomada de decisão do modelo, o conjunto de Campos de Cabeçalho apresentou 16, enquanto as DLLs Importadas apresentaram três, Chamadas de Função 22 e a Entropia de Seções nove. Ao realizar uma inspeção individual, pode-se notar que a característica RESOURCE_Size apresenta o maior nível de influência no processo de tomada de decisão. Essa característica mostrou um padrão consistente, em que valores mais baixos estão predominantemente associados à classe positiva, indicando que amostras de *ransomware* geralmente possuem um tamanho menor (ou zero) no diretório de recursos do Cabeçalho Opcional. Por outro lado, a classe negativa exibiu uma distribuição de valores mais variada.

A DLL comctl32.dll foi identificada como a segunda característica mais impactante. Essa DLL contém uma coleção de funções e recursos que fornecem controles comuns para *Graphical User Interface* (GUI) em aplicativos do *Windows*. A ausência dessa DLL tende a estar

Figura 5 – Gráfico SHAP do tipo *beeswarm* das 50 principais características, organizadas em ordem decrescente de acordo com seu impacto no processo de tomada de decisão, juntamente com o impacto cumulativo das 1517 características restantes.



Fonte: Elaborado pelo autor

associada à classe positiva, o que indica que seu uso é improvável em amostras de *ransomware* e implica uma preferência dos *ransomwares* em operar em um ambiente de modo texto. Por outro lado, a presença de comctl32.dll tende para a classe negativa, o que sugere o uso de GUI em amostras de *goodware*.

A *feature Overlay*, que pertence ao conjunto de entropia das seções, mostrou-se como a terceira mais significativa. Este bloco extra serve a aplicativos que exigem o armazenamento de recursos adicionais ou o carregamento dinâmico de código em tempo de execução. Além disso, quando um arquivo executável é empacotado, há uma maior probabilidade de uma seção *Overlay* estar presente no arquivo. Embora a presença ou um valor alto de entropia geralmente esteja associada a *goodware*, e a ausência ou um valor baixo de entropia esteja mais relacionada a *ransomware*, a Figura 5 indica que há amostras maliciosas que usam esses artifícios, possivelmente para ocultar sua natureza maliciosa dos mecanismos de defesa através de criptografia ou compressão.

A quarta característica mais impactante é a entropia da seção .rsrc, que corresponde à seção de recursos. Esta seção é responsável por armazenar vários recursos relacionados ao arquivo executável, como ícones, bitmaps, caixas de diálogo, menus, *strings* e outros dados necessários para o correto funcionamento e apresentação visual da aplicação. Os valores da seção .rsrc seguem um padrão similar ao a seção de *Overlay*. Observa-se que as características RESOURCE_Size, Overlay e .rsrc estão relacionadas em termos de utilização de recursos. Portanto, esses fatores desempenham um papel significativo na distinção entre amostras de *ransomware* e *goodware*.

A quinta *feature* mais impactante, msvcrt.dll, contém uma coleção de funções e recursos usados por programas compilados com o *Microsoft Visual C++*, fornecendo funções padrão da biblioteca C para alocação de memória, entrada/saída de arquivo, manipulação de *strings* e operações matemáticas. A presença da *feature* msvcrt.dll indica o uso frequente do *Microsoft Visual C++* em aplicativos benignos, pois é indispensável para o seu correto funcionamento. Por outro lado, a ausência de msvcrt.dll em amostras de *ransomware* sugere que é menos provável que elas sejam compiladas com o *Microsoft Visual C++*.

A Figura 5 também mostra que a presença ou altos valores estão associados a *goodwares* em 32 das 50 *features*, enquanto que ausência ou baixos valores indicativos de *ransomware* ocorrem em 18 das 50. Isto é, existe uma propensão na associação de presença/altos valores de *features* com *goodware* e ausência/baixos valores com *ransomware*. Essa propensão pode ser atribuída à enorme variação de utilizações dos *softwares* benignos, em contraste ao escopo limitado das ações realizadas pelo *ransomware*, que envolvem tipicamente as tarefas de coleta de informações do sistema, mapeamento do ambiente-alvo, criptografia de arquivos e estabelecimento de conexões com servidores de C&C.

A característica kernel32.dll é uma das 18 que apresentam um padrão que desvia dessa propensão. Essa DLL é frequentemente importada pela classe positiva (*ransomware*), enquanto sua ausência está mais associada ao *goodware*. A biblioteca kernel32.dll fornece uma ampla gama de funções para interagir com o SO. Entre essas funções, várias APIs nativas, como GetConsoleMode, GetModuleHandleA, CreateMutexA, VirtualProtect, FlushFileBuffers, GetProcessHeap e RltUnwind, foram identificadas como algumas das 50 características mais impac-

tantes. Essas APIs têm diversos propósitos, como operações de console, trabalho com módulos, sincronização de *threads* ou processos, gerenciamento de proteção de memória, operações de arquivos, gerenciamento do *heap* de processos e tratamento de exceções. *Malwares* frequentemente importam e utilizam essas funções para realizar atividades maliciosas e explorar vulnerabilidades. Essas atividades podem incluir acesso a dados sensíveis, modificação de configurações do sistema ou obtenção de privilégios elevados. Ao utilizar essas APIs, os *ransomwares* podem executar suas ações maliciosas pretendidas e potencialmente evitar a detecção ao aproveitar funções legítimas do sistema.

Por fim, também foram observados que valores mais altos de entropia nas seções .rdata, .reloc, .data e .text estão mais comumente associados ao *ransomware*. Isso sugere que essas seções podem ter sido submetidas à compressão ou criptografia por empacotadores, o que reduz o tamanho do arquivo ou obscurece seu conteúdo.

4.4.3 Análise das Classificações Incorretas do Modelo

Para explorar visualmente, compreender e discutir as características que contribuem para as classificações incorretas do modelo, a técnica SHAP foi aplicada à família com pior desempenho de detecção (BlackByte) e às amostras de *goodware*. Ao focar na análise de FP e FN, é possível destacar as competências do modelo e as áreas para melhoria.

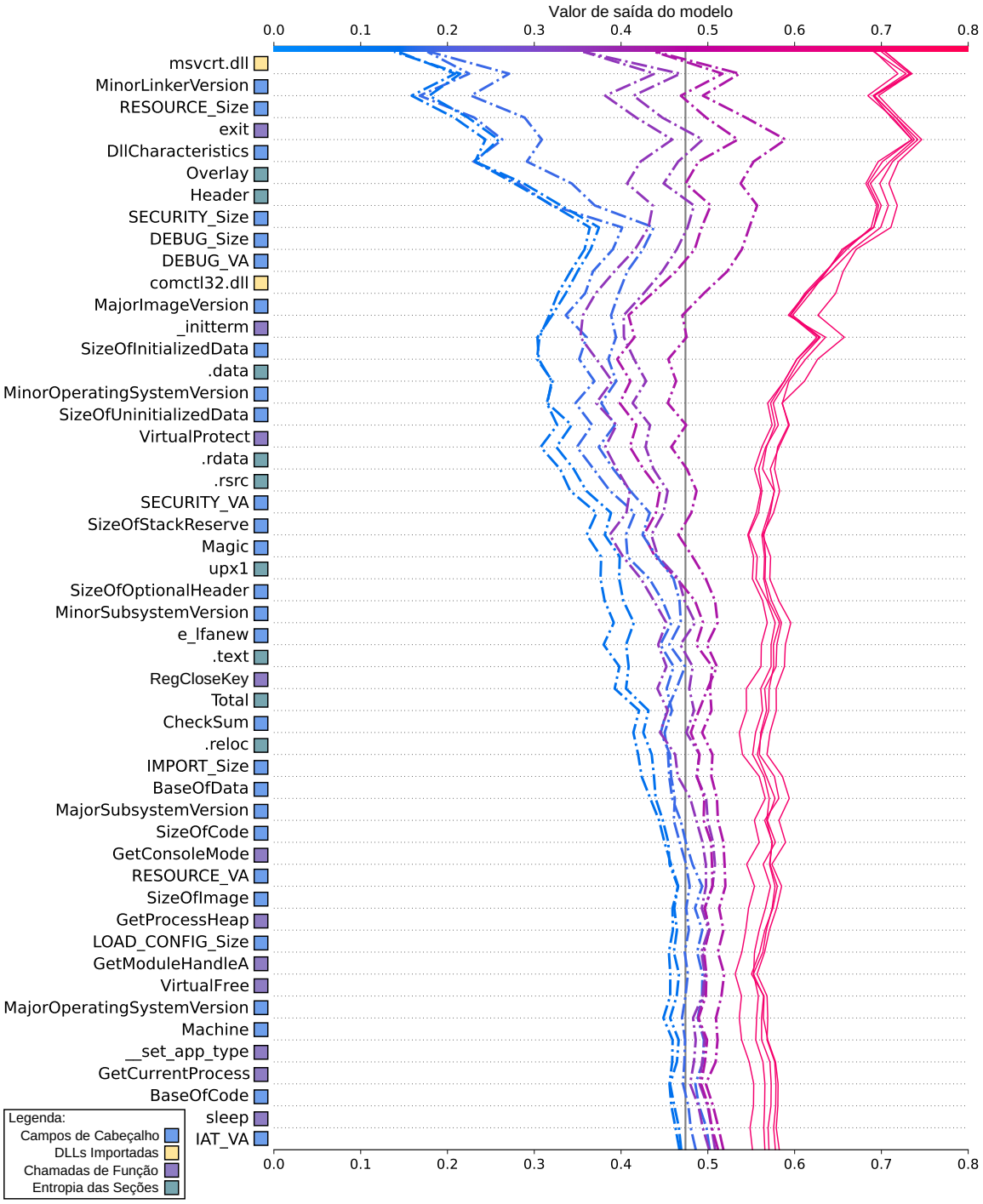
A Figura 6 ilustra o gráfico de decisão SHAP para as 13 amostras pertencentes à família BlackByte, ordenadas de acordo com seu impacto na saída do modelo. Dentre as 50 *features* principais, a maioria é composta por campos de cabeçalho, totalizando 28, seguido por chamadas de função com 11 e entropia de seções com nove; as DLLs importadas, por sua vez, somam apenas duas.

O gráfico representa os caminhos de decisão do modelo com base nos valores SHAP de cada característica. Cada linha representa uma amostra e sua jornada para uma predição. Da parte inferior do gráfico, a linha de predição demonstra o acúmulo dos valores SHAP, refletindo os efeitos das características, desde o valor base até a saída do modelo na parte superior. Linhas vermelhas indicam predição de *ransomware* (acima de $0,5$), enquanto linhas azuis indicam predição de *goodware* (abaixo de $0,5$). Linhas tracejadas representam classificações incorretas. A linha vertical cinza é o valor base ($0,474$) de onde começam as contribuições das características, sendo a média de todas as predições de treinamento.

Dentro das 13 linhas na Figura 6, as oito tracejadas representam classificações incorretas de *ransomware* (FN). Três principais caminhos de decisão são distinguíveis, nas cores azul, roxo e vermelho. O modelo proposto demonstra uma consideração equilibrada de *features* influentes, pois não há uma em particular com impacto drasticamente maior do que as outras. No entanto, grupos de *features* podem impactar a saída do modelo.

Os três principais caminhos exibem comportamento semelhante nas cinco principais

Figura 6 – Gráfico SHAP do tipo decisão das 50 principais características para as 13 amostras da família BlackByte, ordenadas de acordo com seu impacto na saída do modelo.



Fonte: Elaborado pelo autor

características, mas a magnitude das linhas inclinadas em direção a zero aumenta nos caminhos azul e roxo. Além disso, no caminho azul, o grupo Overlay, Header e SECURITY_Size mostra a direção ou magnitude mais contrastante em comparação com os outros caminhos. Vale destacar os grupos SizeOfInitializedData e .data, bem como .rdata, .rsrc e SECURITY_VA, que também contribuem para se associar à classe negativa, influenciando gradualmente a divergência do

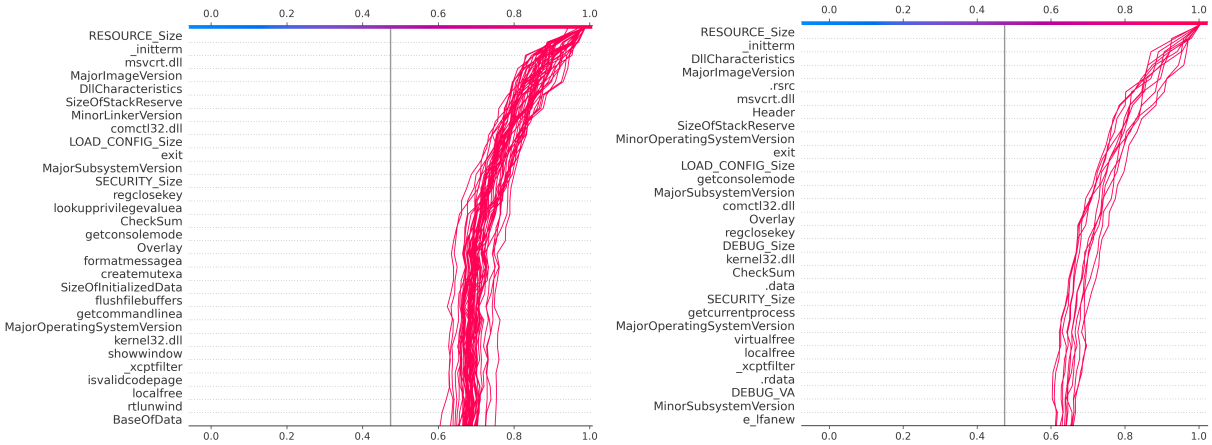
caminho vermelho.

As *features* mencionadas, com valores baixos ou ausentes, estão predominantemente associadas a *ransomware*, exceto SizeOfInitializedData, .data e .rdata, onde valores baixos são observados tipicamente em *goodware* (consultar a Figura 5). A direção contrastante das linhas tracejadas em comparação com as linhas vermelhas sugere valores de características opostos, como observado nas *features* Overlay, SECURITY_Size, SizeOfInitializedData, .data, .rdata e outras. Portanto, a família BlackByte exibe propriedades internamente distintas, evidenciadas nos três principais caminhos. Isso sugere a existência de cepas dentro dessa família, duas das quais demonstraram semelhanças com *goodwares*, o que levou à classificação incorreta pelo modelo proposto.

Para verificar a presença de possíveis cepas nas outras famílias, a Figura 7 apresenta gráficos SHAP do tipo decisão para as 14 famílias que obtiveram resultados satisfatórios de detecção. Enquanto dez das famílias analisadas apresentaram caminhos de decisão uniformes, as famílias Black Basta, Clop, Hive e Night Sky mostraram ramificações dentro dos caminhos de decisão, o que indica a possibilidade de existência de cepas dentro dessas variantes. No entanto, o modelo proposto foi capaz de classificá-las corretamente, o que demonstra sua eficácia mesmo diante de variações internas dentro das famílias de *ransomware*.

Figura 7 – Gráficos SHAP do tipo decisão das 30 principais características para as 14 famílias de *ransomware* que obtiveram resultados satisfatórios de detecção.

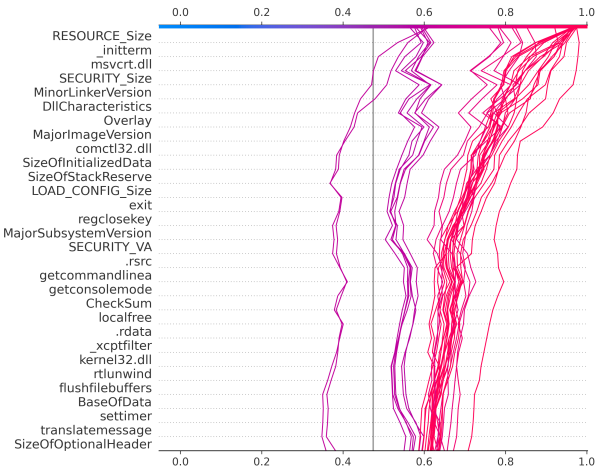
(continua)



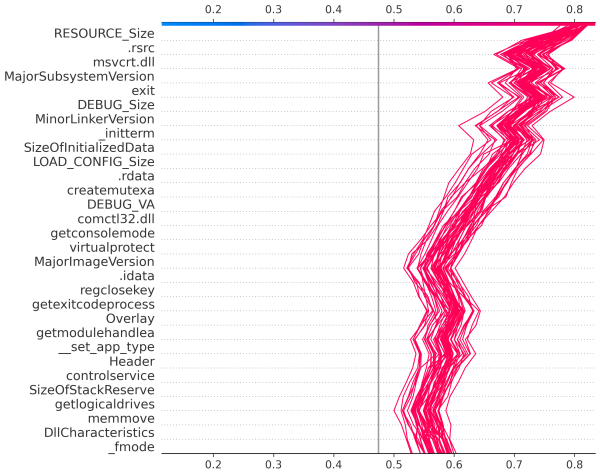
(a) AvosLocker (b) BianLian

Figura 7 – Gráficos SHAP do tipo decisão das 30 principais características para a 14 famílias de *ransomware* que obtiveram resultados satisfatórios de detecção.

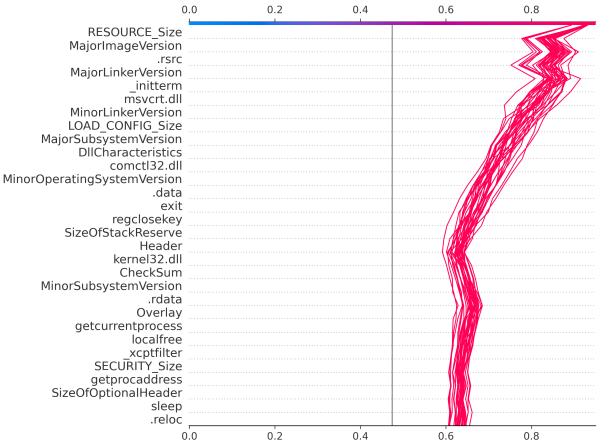
(continuação)



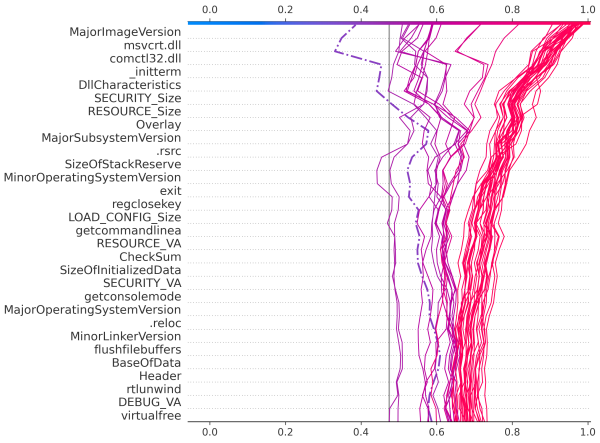
(c) Black Basta



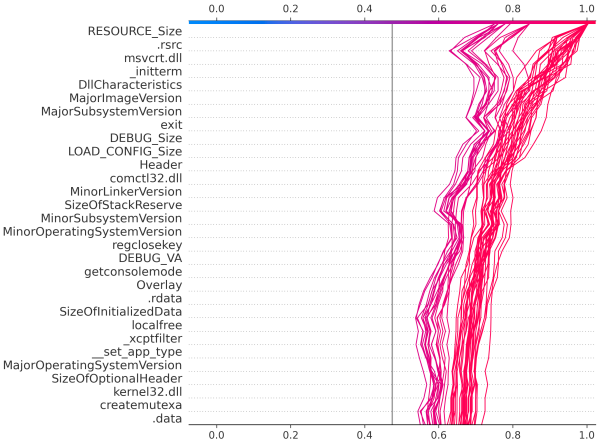
(d) BlackCat



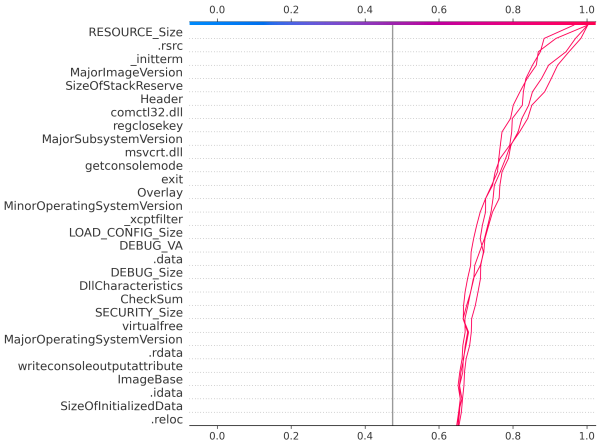
(e) BlueSky



(f) Clop



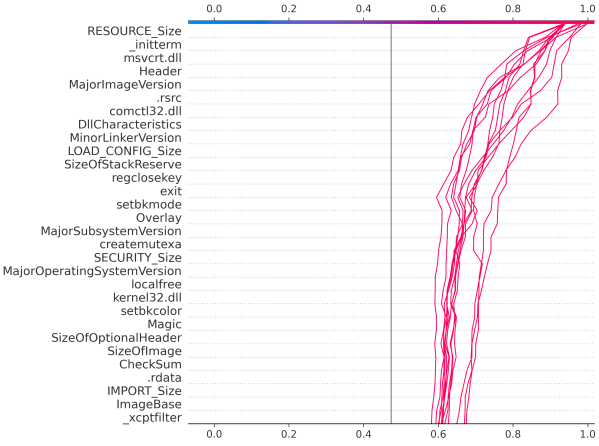
(g) Hive



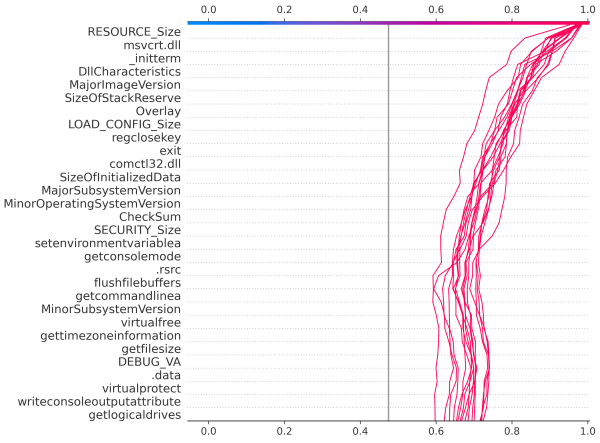
(h) HolyGhost

Figura 7 – Gráficos SHAP do tipo decisão das 30 principais características para a 14 famílias de *ransomware* que obtiveram resultados satisfatórios de detecção.

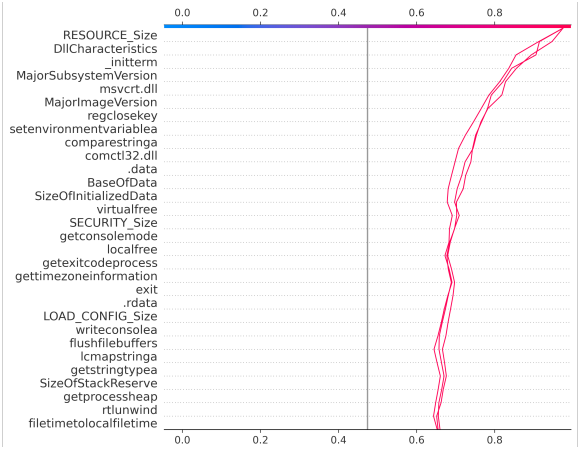
(conclusão)



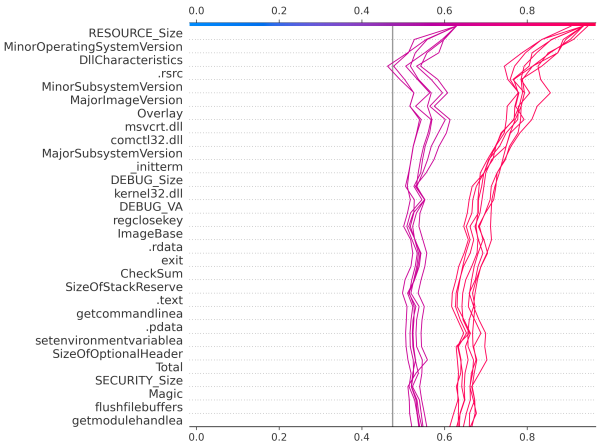
(i) Karma



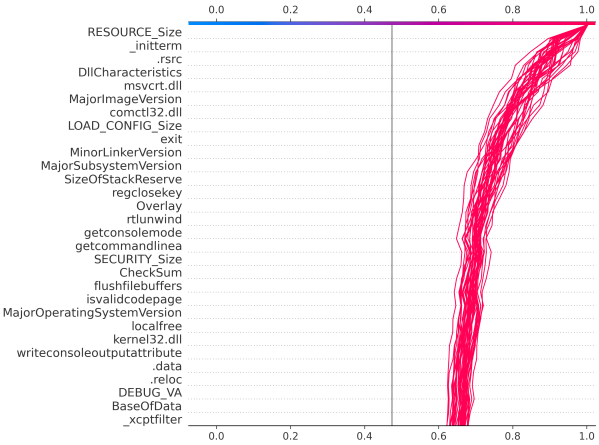
(j) Lorenz



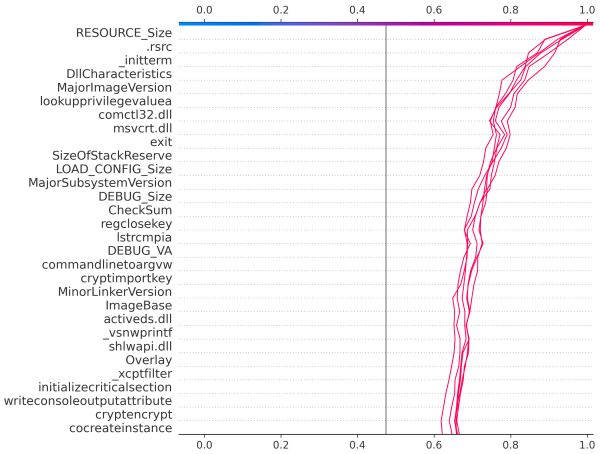
(k) Maui



(l) Night Sky



(m) PlayCrypt



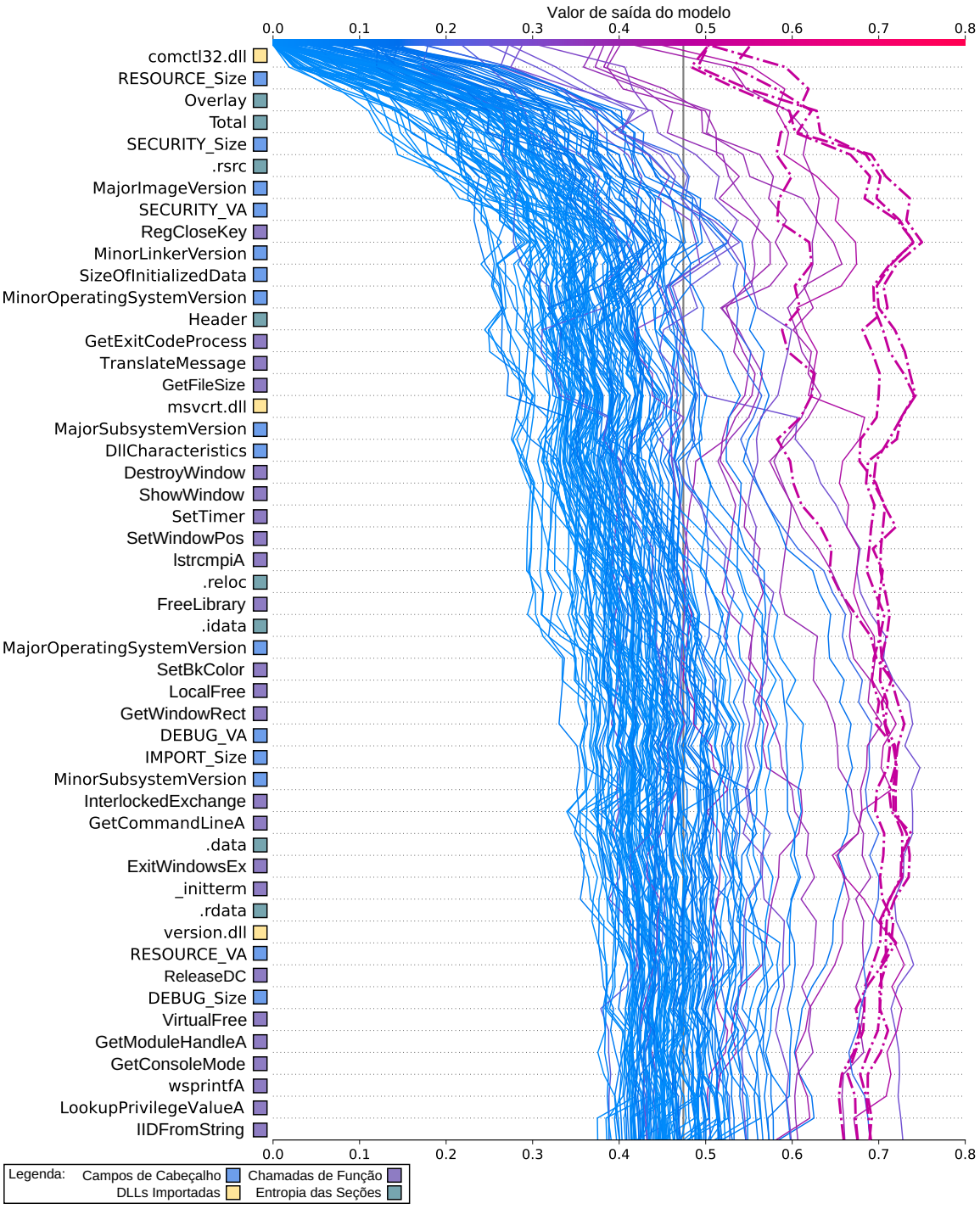
(n) Quantum

Fonte: Elaborado pelo autor

O SHAP também foi aplicado às 133 amostras de *goodware*, e a Figura 8 ilustra o gráfico de decisão das 50 principais características, ordenadas de acordo com seu impacto na

saída do modelo. Dentre as 50 características principais, a maioria consiste em chamadas de função, totalizando 24, seguido por campos de cabeçalho com 15 e entropia de seções com oito; enquanto DLLs importadas somam apenas três. O modelo proposto mantém uma consideração equilibrada de características influentes.

Figura 8 – Gráfico SHAP do tipo decisão das 50 principais características para as 133 amostras de *goodware*, ordenadas de acordo com seu impacto na saída do modelo.



Fonte: Elaborado pelo autor

É possível distinguir o principal caminho de decisão em azul, onde as três principais

características contribuem significativamente para a classificação correta (TN) com um valor de saída do modelo abaixo de $0,2$. Ainda, há algumas amostras corretamente classificadas dentro de caminhos roxos e quatro instâncias de *goodware* classificadas incorretamente (FP), representadas por linhas tracejadas roxas.

As linhas dos FPs, inicialmente próximas ao ponto $0,7$ na parte inferior do gráfico, estendem-se até o topo com valores de saída próximos a $0,5$, o que representa um ajuste de $\approx 0,2$ em direção à decisão correta. Isso indica que as 50 principais *features* não foram as principais contribuintes para as saídas incorretas; em vez disso, as outras 1517 características, acumuladas do valor base para a classe positiva, desempenharam esse papel. Nota-se ainda que as linhas mantêm pouca variação do 50º ao décimo lugar para a maioria dos FPs e, embora insuficiente para classificá-los corretamente como *goodwares*, as nove *features* mais importantes influenciaram significativamente o resultado do modelo para uma saída de valor mais baixo.

A Figura 8 também exibe muitas *features* diretamente relacionadas à GUI, como DestroyWindow, ShowWindow, SetWindowPos, SetBkColor, GetWindowRect e ExitWindowsEx, confirmando a preferência do *goodware* por esse modo operacional. Portanto, é válido considerar o potencial dos *ransomwares* para burlar o modelo ao utilizar essas *features*.

5 DISCUSSÃO

Esta seção discute e compara os resultados desta tese com trabalhos relacionados. A abordagem proposta, detalhada na Seção 3.6, concentrou-se em identificar conjuntos de características significativas e explorar o potencial de melhoria no desempenho da DNFR ao combiná-las. Foram examinadas várias técnicas de ML, integrando três delas em um modelo de votação suave em conjunto. Além disso, a utilização do SHAP nas Seções 4.4.2 e 4.4.3 promoveu a interpretabilidade dos resultados, oferecendo percepções sobre as influências das características e possibilitando uma análise aprofundada de falsos positivos e negativos.

Para discutir e analisar comparativamente os resultados desta tese com os trabalhos relacionados, o modelo proposto foi avaliado na abordagem de CV estratificada com $K = 10$ com o conjunto de dados completo e obteve uma acurácia de 99,39%, precisão de 99,27%, *recall* de 99,58% e F_1 de 99,42%. Como apresentado na Tabela 9 da Seção 4.4, a avaliação pela abordagem de DNFR obteve métricas ligeiramente mais baixas de 97,53%, 96,36%, 97,52% e 96,41%, o que demonstra ainda um desempenho robusto no desafiador cenário de detecção de famílias de dia zero.

A Tabela 12 compara o modelo proposto com trabalhos relacionados usando abordagens de teste padrão, todos empregando análise estática. Embora outros trabalhos tenham alcançado altas performances de detecção, comparáveis ao modelo proposto, o modelo desta tese os supera no tempo de teste, com apenas 0,37s. O competidor mais próximo em tempo de detecção foi (Zhu et al., 2022) com 0,5s, que utilizou apenas características de Entropia. O desempenho superior em tempo de detecção do modelo proposto nesta tese pode ser creditado à combinação hábil de características leves, juntamente com a exclusão de *opcodes* com N-gramas, o que eliminou a demora associada a esse tipo de característica. Essa observação é consistente com os resultados de (Zhang, H. et al., 2019), que utilizou somente *opcodes* com 3-gramas e relatou um tempo de teste de 4,26s.

A Tabela 12 revela variações no número de instâncias para *ransomware* e *goodware*, com valores de 727 a 50 mil. Além disso, apenas esta tese e os estudos (Ciaramella et al., 2023; Vidyarthi et al., 2019; Ayub; Sirai, 2021) conduziram análise de importância de *features*, um passo fundamental para garantir confiança nas decisões do modelo e identificar possíveis viéses.

A Tabela 12 também destaca a superioridade das abordagens de características combinadas em relação às individuais em métricas de desempenho. As abordagens combinadas têm uma média de 97%, enquanto estudos que usam características individuais têm uma média de 94%. Ademais, a técnica RF se destaca em seis dos 13 trabalhos, o que reforça a eficácia de árvores de decisão em conjunto como EXT, RF e XGB em cenários de testes com novas amostras de *ransomware* de famílias conhecidas, como evidenciado na Tabela 5 na Seção 4.2.

Tabela 12 – Comparação do modelo proposto com trabalhos relacionados utilizando abordagens de teste padrão.

Trabalho	Classificador	*Features*	Instâncias *Rans.-Good.*	Análise Imp. Ft.	Acur. (%)	Prec. (%)	Rec. (%)	*F-m.* (%)	Tempo Teste
(Zhang, H. et al., 2019)	RF	*Opcode*	1787-100	✗	99,30	-	99,80	-	4,26s
(Zhang, B. et al., 2020)	CNN	*Opcode*	1787-100	✗	89,50	87,50	87,60	87,30	-
(Khammas, 2020)	RF	*Byte* Bruto	840-840	✗	97,74	95,87	99,80	97,80	1,37s
(Stiawan et al., 2021)	KNN	*Opcode*	3000-3000	✗	98,86	100	98,00	98,98	-
(Manavi; Hamzeh, 2021)	LSTM	Cabeçalho	1000-1000	✗	93,25	93,33	93,25	93,24	-
(Manavi; Hamzeh, 2022a)	RF	Cabeçalho	1000-1000	✗	93,30	93,48	93,30	93,28	-
(Manavi; Hamzeh, 2022b)	CNN	Cabeçalho	1000-1000	✗	93,33	93,40	93,33	93,34	-
(Zhu et al., 2022)	SNN	Entropia	1046-0	✗	-	85,30	88,70	86,20	0,5s
(Ciaramella et al., 2023)	CNN	*Opcode*	5000-5000	✓	96,90	97,00	96,90	96,90	-
(Gaur et al., 2021)	RF	Cabeçalho, API, DLL, Entropia, *Opcode*, *Packer*, *Hash*	23k-27k	✗	99,68	99,72	99,65	99,72	-
(Vidyarthi et al., 2019)	RF	Cabeçalho, API, DLL, Entropia, *Packer*, Texto	2722-2488	✓	99,25	99,30	99,30	99,30	-
(Ayub; Sirai, 2021)	LOF	Cabeçalho, API, DLL, Texto	727-0	✓	89,96	-	-	-	-
Modelo Proposto	*Ensemble*: LR,RF,XGB	Cabeçalho, API, DLL, Entropia	1408-1267	✓	99,39	99,27	99,58	99,42	0,37s

Fonte: Elaborado pelo autor

Apesar de alguns estudos alcançarem taxas de detecção em torno de 99% na Tabela 12, o desempenho geral diminui em um cenário mais desafiador de detecção de famílias dia zero, como observado ao contrastar as Tabelas 5 e 6 na Seção 4.2. A Tabela 13 compara o modelo proposto com trabalhos relacionados que aplicaram a metodologia de DNFR, ou seja, nenhuma amostra de uma determinada família de teste incluída na fase de treinamento.

A maioria dos estudos que abordam o cenário de DNFR depende de análises dinâmicas ou híbridas. Notavelmente, o modelo proposto e os trabalhos (Vehabovic et al., 2023; Moreira; Moreira; Sales JR., 2023) exploram de forma única a DNFR por meio de análise estática. Além disso, apenas esta tese e os trabalhos (Sgandurra et al., 2016; Shaukat; Ribeiro, 2018; Moreira; Moreira; Sales JR., 2023) analisaram a importância das características para tornar as decisões de modelo transparentes e interpretáveis.

Na Tabela 13, é evidente a discrepância no tempo de teste entre análises dinâmicas/híbridas e estáticas. As abordagens dinâmicas/híbridas requerem mais de 30s para a execução da amostra, enquanto as análises estáticas são concluídas em menos de 1s. Esse contraste destaca as limitações dos estudos dinâmicos/híbridos para detecção em tempo de execução, considerando

Tabela 13 – Comparação do modelo proposto com trabalhos relacionados que usaram abordagem de DNFR.

Trabalho	Anál.	Classif.	*Features*	Instâncias *Rans.-Good.*	Fam.	Análise Imp. Ft.	Acur. (%)	Prec. (%)	Rec. (%)	*F-m.* (%)	Tempo Teste
(Sgandurra et al., 2016)	Din.	LR	API, Arq./Dir., Registro, Texto	582 - 942	11	✓	93,30	-	-	-	30s
(Cohen; Nissim, 2018)	Din.	RF	*Dump* de Memória	5 - 4	5	✗	-	100	95,70	97,50	10min
(Shaukat; Ribeiro, 2018)	Híb.	*Gradient Tree Boosting*	Cabeçalho, *Packer*, Entropia, Arq./Dir., Texto	574 - 442	12	✓	-	99,55	98,25	98,89	60min
(Zahoora et al., 2022)	Din.	*Ensemble Zero-shot Learning*	API, Arq./Dir., Registro, Texto	582 - 942	11	✗	92,88	90,78	95,52	93,09	30s
(Vehabovic et al., 2023)	Est.	RF	Cabeçalho	1240 - 2000	9	✗	81,04	-	-	-	0,97s*
(Moreira; Moreira; Sales JR., 2023)	Est.	CNN	Cabeçalho	1023 - 1134	25	✓	93,84	97,50	89,77	92,27	0,66s
Modelo Proposto	Est.	*Ensemble*: LR,RF,XGB	Cabeçalho, API, DLL, Entropia	1408 - 1267	40	✓	97,53	96,36	97,52	96,41	0,37s

*Média ponderada de *ransomwares* e *goodwares*.

Fonte: Elaborado pelo autor

que certas famílias de *ransomware* podem criptografar dados a taxas superiores a 250 MB/s.[1]

Entre os trabalhos de análise estática delineados na Tabela 13, esta pesquisa os supera em taxas de detecção e tempo de teste. Ao comparar as taxas de detecção do modelo proposto com as melhores abordagens de análise dinâmica/híbrida, os estudos (Cohen; Nissim, 2018; Shaukat; Ribeiro, 2018) mostraram resultados comparáveis. No entanto, (Cohen; Nissim, 2018) utilizou apenas cinco instâncias de *ransomware*, insuficientes para cenários do mundo real. Por outro lado, o estudo (Shaukat; Ribeiro, 2018) empregou um conjunto de dados maior, composto de 574 instâncias de *ransomware* de 12 famílias. Notavelmente, esta tese supera os demais ao investigar 1408 instâncias de *ransomware* de 40 famílias, o que aumenta a representatividade das ameaças atuais para um estudo mais abrangente.

[1] https://www.splunk.com/en_us/blog/security/gone-in-52-seconds-and-42-minutes-a-comparative-analysis-of-ransomware-encryption-speed.html

6 CONCLUSÕES

Este estudo contribui ao desenvolver um modelo abrangente de análise estática que detecta com sucesso famílias de *ransomware* desconhecidas. O modelo combina conjuntos de características extraídas de arquivos PE e utiliza um classificador de votação ponderada do tipo *ensemble* composto pelos algoritmos LR, RF e XGB. A abordagem mantém tamanho compacto e tempos de teste praticáveis, tornando-a adequada para implantação em larga escala e funcionamento em dispositivos de *hardware* com recursos limitados em aplicações em tempo real. Portanto, o modelo treinado pode ser empregado como uma camada adicional de proteção do *Windows*, examinando cada arquivo executável ao receber *downloads* e anexos ou antes da execução sem comprometer o desempenho do sistema.

A análise das características extraídas levou à exclusão do conjunto de *features* de texto imprimível devido à sua suscetibilidade à variabilidade causada por caracteres substitutos. Essa variabilidade pode prejudicar a detecção. Além disso, os textos imprimíveis contidos nos binários podem facilitar a exploração por técnicas adversárias que clonam textos de amostras benignas em amostras de *ransomware*. Os conjuntos de *opcodes* com frequências N-grama exibiram baixo desempenho na DNFR e, portanto, foram também descartados do modelo proposto. Desta forma, a melhor combinação de características compreendeu campos de cabeçalho, DLLs importadas, chamadas de funções e entropia das seções. Essa combinação superou as outras e todos conjuntos de características individuais em termos de desempenho métrico.

Em comparação com a abordagem padrão de CV, o cenário desafiador de DNFR demonstrou um desempenho *menor* na maioria dos modelos e conjuntos de características avaliados, o que destaca a dificuldade do teste. No entanto, mesmo para a DNFR, foi observado um notável nível de eficácia no modelo proposto, que identificou com precisão novas amostras de *ransomware*, alcançando uma taxa de *recall* de 100% para 13 das 15 famílias desconhecidas analisadas. Como resultado, esse desempenho demonstra a capacidade do modelo de se adaptar à natureza em constante evolução do *ransomware* ao longo do tempo.

Além disso, o método de interpretação SHAP revelou as características decisivas e seus respectivos valores no processo de tomada de decisão do modelo. Essas características incluíam baixos valores para RESOURCE_Size no campo de cabeçalho, a ausência das DLLs comctl32.dll e msvcrt.dll, e a não utilização ou baixos valores de entropia para as seções Overlay e .rsrc. Essas descobertas destacam a preferência dos *ransomwares* pelo modo de *console* em vez do modo GUI, o uso de compiladores diferentes do *Microsoft Visual C++* e a importância de considerar a baixa utilização de recursos nas características estruturais exibidas pelos *ransomwares*.

A segurança cibernética está imersa numa batalha constante entre técnicas de ataque

cibernético e *softwares antimalware*. O *ransomware*, uma ameaça altamente perigosa, continua a se proliferar e apresenta desafios substanciais à segurança cibernética. É fundamental desenvolver continuamente técnicas de segurança inovadoras e eficazes para combater essa ameaça. Os resultados experimentais desta tese demonstram a eficácia do modelo em identificar com precisão novas famílias de *ransomware*. Portanto, pode ajudar os desenvolvedores de sistemas de detecção de *ransomware* a construir soluções mais robustas e confiáveis.

Embora o foco principal desta tese tenha se centrado na detecção de *ransomware* no SO *Windows*, uma exploração da adaptabilidade do modelo proposto a outros ambientes, como *Android* ou Internet das Coisas, tem o potencial de fornecer resultados valiosos. Além disso, a avaliação da aplicabilidade do modelo em defesas baseadas em análise de tráfego de arquivos para prevenção de *ransomware* em estágios iniciais ampliaria o alcance deste estudo para fortalecer as medidas de segurança cibernética em um nível de rede. A metodologia aplicada nesta tese estabelece uma base para pesquisas futuras desenvolverem soluções adaptadas a diferentes ambientes operacionais e setores industriais. Essa linha de investigação é promissora para a versatilidade e a eficácia de soluções de detecção de *ransomware* em um cenário tecnológico mais amplo.

6.1 Divulgação da Pesquisa

Três artigos científicos foram publicados em periódicos a partir dos resultados desta pesquisa:

- MOREIRA, Caio; SALES JR., Claudomiro; MOREIRA, Davi. Understanding Ransomware Actions Through Behavioral Feature Analysis. **Journal of Communication and Information Systems**, v. 37, n. 1, p. 61-76, mar. 2022. ISSN 1980-6604. DOI: 10.14209/jcis.2022.7. Qualis CAPES A4 (2017-2020).

- MOREIRA, Caio; MOREIRA, Davi; SALES JR., Claudomiro. Improving Ransomware Detection based on Portable Executable Header using Xception Convolutional Neural Network. **Computers & Security**, v. 130, p. 103265, jul. 2023. ISSN 0167-4048. DOI: 10.1016/j.cose.2023.103265. Qualis CAPES A1 (2017-2020).

- MOREIRA, Caio; MOREIRA, Davi; SALES JR., Claudomiro. A Comprehensive Analysis Combining Structural Features for Detection of New Ransomware Families. **Journal of Information Security and Applications**, v. 81, p. 103716, mar. 2024. ISSN 2214-2126. DOI: 10.1016/j.jisa.2024.103716. Qualis CAPES A3 (2017-2020).

REFERÊNCIAS

AAG-IT. **The Latest 2024 Ransomware Statistics**. Fev. 2024. Disponível em: <https://aag-it.com/the-latest-ransomware-statistics/>. Acesso em: 13 fev. 2024. Citado na p. 21.

AITYAN, Sergey K. Confidence Intervals. In: BUSINESS Research Methodology: Research Process and Methods. Cham: Springer International Publishing, 2022. P. 233–277. ISBN 978-3-030-76857-7. DOI: 10.1007/978-3-030-76857-7_13. Citado na p. 46.

AL FATIH ABIL FIDA, Muhammad; AHMAD, Tohari; NTAHOBARI, Maurice. Variance Threshold as Early Screening to Boruta Feature Selection for Intrusion Detection System. In: 2021 13th International Conference on Information & Communication Technology and System (ICTS). 2021. P. 46–50. DOI: 10.1109/ICTS52701.2021.9608852. Citado na p. 45.

ALANI, Mohammed M.; AWAD, Ali Ismail. PAIRED: An Explainable Lightweight Android Malware Detection System. **IEEE Access**, v. 10, p. 73214–73228, 2022. DOI: 10.1109/ACCESS.2022.3189645. Citado na p. 50.

ALANI, Mohammed M.; MASHATAN, Atefeh; MIRI, Ali. XMal: A lightweight memory-based explainable obfuscated-malware detector. **Computers & Security**, v. 133, p. 103409, 2023. ISSN 0167-4048. DOI: 10.1016/j.cose.2023.103409. Citado na p. 50.

ALENEZI, Mohammed N; ALABDULRAZZAQ, Haneen; ALSHAHER, Abdullah A; ALKHARANG, Mubarak M. Evolution of Malware Threats and Techniques: A Review. **International Journal of Communication Networks and Information Security**, Kohat University of Science e Technology (KUST), v. 12, n. 3, p. 326–337, 2020. DOI: 10.17762/ijcnis.v12i3.4723. Citado na p. 17.

ALSMADI, Tibra; ALQUDAH, Nour. A Survey on malware detection techniques. In: 2021 International Conference on Information Technology (ICIT). 2021. P. 371–376. DOI: 10.1109/ICIT52682.2021.9491765. Citado na p. 18.

ARIVUDAINAMBI, D; KA, Varun Kumar; VISU, P et al. Ransomware Traffic Classification Using Deep Learning Models: Ransomware Traffic Classification. **International Journal of Web Portals (IJWP)**, IGI Global, v. 12, n. 1, p. 1–11, 2020. DOI: 10.4018/IJWP.2020010101. Citado na p. 30.

ASLAN, Ömer Aslan; SAMET, Refik. A Comprehensive Review on Malware Detection Approaches. **IEEE Access**, v. 8, p. 6249–6271, 2020. DOI: 10.1109/ACCESS.2019.2963724. Citado nas pp. 17, 18.

ATAPOUR-ABARGHOUEI, Amir; BONNER, Stephen; MCGOUGH, Andrew Stephen. A King's Ransom for Encryption: Ransomware Classification using Augmented One-Shot Learning and Bayesian Approximation. In: 2019 IEEE Int. Conf. Big Data. 2019. P. 1601–1606. DOI: 10.1109/BigData47090.2019.9005540. Citado na p. 31.

AYUB, Md. Ahsan; SIRAI, Ambareen. Similarity Analysis of Ransomware based on Portable Executable (PE) File Metadata. In: 2021 IEEE Symposium Series on Computational Intelligence (SSCI). 2021. P. 1–6. DOI: 10.1109/SSCI50451.2021.9660019. Citado nas pp. 25, 26, 28, 72, 73.

BAE, Seong Il; LEE, Gyu Bin; IM, Eul Gyu. Ransomware detection using machine learning algorithms. **Concurrency and Computation: Practice and Experience**, John Wiley e Sons Ltd, v. 32, n. 18, e5422, abr. 2020. ISSN 1532-0626. DOI: 10.1002/cpe.5422. Citado nas pp. 48, 49.

BARREDO ARRIETA, Alejandro; DÍAZ-RODRÍGUEZ, Natalia; DEL SER, Javier; BENNETOT, Adrien; TABIK, Siham; BARBADO, Alberto; GARCIA, Salvador; GIL-LOPEZ, Sergio; MOLINA, Daniel; BENJAMINS, Richard; CHATILA, Raja; HERRERA, Francisco. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. **Information Fusion**, v. 58, p. 82–115, 2020. ISSN 1566-2535. DOI: 10.1016/j.inffus.2019.12.012. Citado na p. 49.

BEAMAN, Craig; BARKWORTH, Ashley; AKANDE, Toluwalope David; HAKAK, Saqib; KHAN, Muhammad Khurram. Ransomware: Recent advances, analysis, challenges and future research directions. **Computers & Security**, v. 111, p. 102490, 2021. ISSN 0167-4048. DOI: 10.1016/j.cose.2021.102490. Citado na p. 18.

BEGOVIC, Kenan; AL-ALI, Abdulaziz; MALLUHI, Qutaibah. Cryptographic ransomware encryption detection: Survey. **Computers & Security**, v. 132, p. 103349, 2023. ISSN 0167-4048. DOI: 10.1016/j.cose.2023.103349. Citado nas pp. 30, 31, 39.

BEKKAR, Mohamed; DJEMAA, Hassiba Kheliouane; ALITOUCHE, Taklit Akrouf. Evaluation Measures for Models Assessment over Imbalanced Data Sets. **Journal of Information Engineering and Applications**, v. 3, n. 10, 2013. Disponível em: <https://www.iiste.org/Journals/index.php/JIEA/article/view/7633>. Citado nas pp. 47, 48.

BERRUETA, Eduardo; MORATO, Daniel; MAGAÑA, Eduardo; IZAL, Mikel. A Survey on Detection Techniques for Cryptographic Ransomware. **IEEE Access**, v. 7, p. 144925–144944, 2019. DOI: 10.1109/ACCESS.2019.2945839. Citado na p. 31.

CABAJ, Krzysztof; GREGORCZYK, Marcin; MAZURCZYK, Wojciech. Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. **Compu-**

ters & Electrical Engineering, v. 66, p. 353–368, 2018. ISSN 0045-7906. DOI: 10.1016/j.compeleceng.2017.10.012. Citado na p. 30.

CEN, Mingcan; JIANG, Frank; QIN, Xingsheng; JIANG, Qinghong; DOSS, Robin. Ransomware early detection: A survey. **Computer Networks**, v. 239, p. 110138, 2024. ISSN 1389-1286. DOI: 10.1016/j.comnet.2023.110138. Citado na p. 20.

CIARAMELLA, Giovanni; IADAROLA, Giacomo; MARTINELLI, Fabio; MERCALDO, Francesco; SANTONE, Antonella. Explainable Ransomware Detection with Deep Learning Techniques. **Journal of Computer Virology and Hacking Techniques**, set. 2023. ISSN 2263-8733. DOI: 10.1007/s11416-023-00501-1. Citado nas pp. 24, 27, 28, 72, 73.

COHEN, Aviad; NISSIM, Nir. Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. **Expert Systems with Applications**, v. 102, p. 158–178, 2018. ISSN 0957-4174. DOI: 10.1016/j.eswa.2018.02.039. Citado nas pp. 26, 28, 74.

DEMŠAR, Janez. Statistical Comparisons of Classifiers over Multiple Data Sets. **Journal of Machine Learning Research**, v. 7, n. 1, p. 1–30, 2006. Disponível em: <http://jmlr.org/papers/v7/demsar06a.html>. Citado na p. 48.

FERNANDO, Damien Warren; KOMNINOS, Nikos; CHEN, Thomas. A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques. **IoT**, v. 1, n. 2, p. 551–604, 2020. ISSN 2624-831X. DOI: 10.3390/iot1020030. Citado na p. 27.

FERRANTE, Alberto; MALEK, Miroslaw; MARTINELLI, Fabio; MERCALDO, Francesco; MILOSEVIC, Jelena. Extinguishing Ransomware - A Hybrid Approach to Android Ransomware Detection. In: IMINE, Abdessamad; FERNANDEZ, José M.; MARION, Jean-Yves; LOGRIPPO, Luigi; GARCIA-ALFARO, Joaquin (Ed.). **Foundations and Practice of Security**. Cham: Springer International Publishing, 2018. P. 242–258. ISBN 978-3-319-75650-9. DOI: 10.1007/978-3-319-75650-9_16. Citado na p. 19.

FORBES, Catherine; EVANS, Merran; HASTINGS, Nicholas; PEACOCK, Brian. Statistical Distributions. In: John Wiley & Sons, Inc., dez. 2010. P. 53–54. doi: 10.1002/9780470627242.ch7. Citado na p. 45.

GARCIA, Joshua; HAMMAD, Mahmoud; MALEK, Sam. Lightweight, Obfuscation-Resilient Detection and Family Identification of Android Malware. **ACM Trans. Softw. Eng. Methodol.**, Association for Computing Machinery, New York, NY, USA, v. 26, n. 3, p. 11–29, jan. 2018. ISSN 1049-331X. DOI: 10.1145/3162625. Citado nas pp. 22, 25.

GAUR, Kartikeya; KUMAR, Nitesh; HANDA, Anand; SHUKLA, Sandeep K. Static Ransomware Analysis Using Machine Learning and Deep Learning Models. In: ANBAR, Mohammed; ABDULLAH, Nibras; MANICKAM, Selvakumar (Ed.). **Advances in Cyber Security**. Singapore: Springer Singapore, 2021. P. 450–467. ISBN 978-981-33-6835-4. DOI: 10.1007/978-981-33-6835-4_30. Citado nas pp. 25, 28, 73.

GEORGIADOU, Anna; MOUZAKITIS, Spiros; ASKOUNIS, Dimitris. Working from home during COVID-19 crisis: a cyber security culture assessment survey. **Security Journal**, v. 35, n. 2, p. 486–505, jun. 2022. ISSN 1743-4645. DOI: 10.1057/s41284-021-00286-2. Citado na p. 19.

GRECO, Claudia; IANNI, Michele; GUZZO, Antonella; FORTINO, Giancarlo. Explaining Binary Obfuscation. In: 2023 IEEE International Conference on Cyber Security and Resilience (CSR). 2023. P. 22–27. DOI: 10.1109/CSR57506.2023.10224825. Citado na p. 50.

GUO, Yang. A review of Machine Learning-based zero-day attack detection: Challenges and future directions. **Computer Communications**, v. 198, p. 175–185, 2023. ISSN 0140-3664. DOI: 10.1016/j.comcom.2022.11.001. Citado na p. 21.

HAMPTON, Nikolai; BAIG, Zubair; ZEADALLY, Sherali. Ransomware behavioural analysis on windows platforms. **Journal of Information Security and Applications**, Elsevier Ltd, v. 40, p. 44–51, jun. 2018. ISSN 22142126. DOI: 10.1016/j.jisa.2018.02.008. Citado na p. 52.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. The elements of statistical learning: data mining, inference, and prediction. In: 2. ed.: New York: Springer, 2009. P. 245–247. ISBN 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7. Citado na p. 42.

HE, Jikai; YU, Jianguo; SONG, Zheng. A static detection method for malware with low false positive rate for packed benign software. In: TIWARI, Rajeev (Ed.). **International Conference on Neural Networks, Information, and Communication Engineering (NNICE 2022)**. Qingdao, China: SPIE, 2022. v. 12258, 122581a. DOI: 10.1117/12.2639229. Citado nas pp. 22, 28.

HOSSIN, Mohammad; SULAIMAN, M. N. A Review on Evaluation Metrics for Data Classification Evaluations. **International Journal of Data Mining & Knowledge Management Process**, v. 5, p. 01–11, mar. 2015. DOI: 10.5121/ijdkp.2015.5201. Citado nas pp. 46, 47.

HSUPENG, Boryau; LEE, Kun-Wei; WEI, Te-En; WANG, Shih-Hao. Explainable Malware Detection Using Predefined Network Flow. In: 2022 24th International Conference on Advanced Communication Technology (ICACT). 2022. P. 27–33. DOI: 10.23919/ICACT53585.2022.9728897. Citado na p. 50.

HU, Weiwei; TAN, Ying. On the robustness of machine learning based malware detection algorithms. In: 2017 International Joint Conference on Neural Networks (IJCNN). 2017. P. 1435–1441. DOI: 10.1109/IJCNN.2017.7966021. Citado na p. 42.

IACONO, Laurie; WOJCIESZEK, Keith; GLASS, George. **Q2 2022 Threat Landscape: Ransomware Returns, Healthcare Hit**. Kroll, ago. 2022. Disponível em: <https://www.kroll.com/-/media/kroll-images/pdfs/q2-2022-threat-landscape-report.pdf>. Citado na p. 19.

JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. **An introduction to statistical learning: with applications in R**. 2. ed.: Spinger New York, NY, 2021. ISBN 978-1-0716-1418-1. DOI: 10.1007/978-1-0716-1418-1. Citado na p. 46.

KAPOOR, Adhirath; GUPTA, Ankur; GUPTA, Rajesh; TANWAR, Sudeep; SHARMA, Gulshan; DAVIDSON, Innocent E. Ransomware Detection, Avoidance, and Mitigation Scheme: A Review and Future Directions. **Sustainability**, v. 14, n. 1, p. 8, 2022. ISSN 2071-1050. DOI: 10.3390/su14010008. Citado na p. 18.

KAUR, Inderpreet; KAUR, Arvinder. Comparative analysis of software fault prediction using various categories of classifiers. **International Journal of System Assurance Engineering and Management**, v. 12, n. 3, p. 520–535, jun. 2021. ISSN 0976-4348. DOI: 10.1007/s13198-021-01110-1. Citado na p. 48.

KHAMMAS, Ban Mohammed. Ransomware Detection using Random Forest Technique. **ICT Express**, v. 6, n. 4, p. 325–331, 2020. ISSN 2405-9595. DOI: 10.1016/j.icte.2020.11.001. Citado nas pp. 23, 28, 41, 73.

KOK, Sim; ABDULLAH, Azween; JHANJHI, Noor; SUPRAMANIAM, Mahadevan. Ransomware, Threat and Detection Techniques: A Review. **IJCSNS - International Journal of Computer Science and Network Security**, v. 19, n. 2, p. 136–146, 2019. Disponível em: <http://paper.ijcsns.org/07_book/201902/20190217.pdf>. Citado nas pp. 20, 30.

KOLODENKER, Eugene; KOCH, William; STRINGHINI, Gianluca; EGELE, Manuel. PayBreak: Defense Against Cryptographic Ransomware. In: PROCEEDINGS of the 2017 ACM on Asia Conference on Computer and Communications Security. Abu Dhabi, United Arab Emirates: Association for Computing Machinery, 2017. (ASIA CCS '17), p. 599–611. ISBN 9781450349444. DOI: 10.1145/3052973.3053035. Citado na p. 19.

KUMAR, Ajit; KUPPUSAMY, K.S.; AGHILA, G. A learning model to detect maliciousness of portable executable using integrated feature set. **Journal of King Saud University - Computer and Information Sciences**, v. 31, n. 2, p. 252–265, 2019. ISSN 1319-1578. DOI: 10.1016/j.jksuci.2017.01.003. Citado na p. 42.

LANG, Michael; CONNOLLY, Lena; TAYLOR, Paul; CORNER, Phillip J. The Evolving Menace of Ransomware: A Comparative Analysis of Pre-pandemic and Mid-pandemic Attacks. **Digital Threats**, Association for Computing Machinery, New York, NY, USA, v. 4, n. 4, out. 2023. DOI: 10.1145/3558006. Citado na p. 19.

LING, Xiang; WU, Lingfei; ZHANG, Jiangyu; QU, Zhenqing; DENG, Wei; CHEN, Xiang; QIAN, Yaguan; WU, Chunming; JI, Shouling; LUO, Tianyue; WU, Jingzheng; WU, Yanjun. Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art. **Computers & Security**, v. 128, p. 103134, 2023. ISSN 0167-4048. DOI: 10.1016/j.cose.2023.103134. Citado nas pp. 17, 44.

LIU, Kaijun; XU, Shengwei; XU, Guoai; ZHANG, Miao; SUN, Dawei; LIU, Haifeng. A Review of Android Malware Detection Approaches Based on Machine Learning. **IEEE Access**, v. 8, p. 124579–124607, 2020. DOI: 10.1109/ACCESS.2020.3006143. Citado nas pp. 48, 49.

LUCKETT, Patrick; MCDONALD, J. Todd; GLISSON, William B.; BENTON, Ryan; DAWSON, Joel; DOYLE, Blair A. Identifying stealth malware using CPU power consumption and learning algorithms. **Journal of Computer Security**, IOS Press, v. 26, n. 5, p. 589–613, 2018. ISSN 1875-8924. DOI: 10.3233/JCS-171060. Citado na p. 20.

LUNDBERG, Scott M.; ERION, Gabriel G.; LEE, Su-In. **Consistent Individualized Feature Attribution for Tree Ensembles**. 2019. DOI: 10.48550/arXiv.1802.03888. Citado nas pp. 50, 51.

LUNDBERG, Scott M.; LEE, Su-In. A Unified Approach to Interpreting Model Predictions. In: PROCEEDINGS of the 31st International Conference on Neural Information Processing Systems. Long Beach, California, USA: Curran Associates Inc., 2017. (NIPS'17), p. 4768–4777. ISBN 9781510860964. DOI: 10.48550/arXiv.1705.07874. Citado nas pp. 50, 51.

M., Gopinath; SETHURAMAN, Sibi Chakkaravarthy. A comprehensive survey on deep learning based malware detection techniques. **Computer Science Review**, v. 47, p. 100529, 2023. ISSN 1574-0137. DOI: 10.1016/j.cosrev.2022.100529. Citado na p. 17.

MANAVI, Farnoush; HAMZEH, Ali. A novel approach for ransomware detection based on PE header using graph embedding. **Journal of Computer Virology and Hacking Techniques**, jan. 2022. (2022). ISSN 2263-8733. DOI: 10.1007/s11416-021-00414-x. Citado nas pp. 24, 28, 41, 73.

MANAVI, Farnoush; HAMZEH, Ali. Ransomware Detection Based on PE Header Using Convolutional Neural Networks. **The ISC International Journal of Information Security**, Iranian Society of Cryptology, v. 14, n. 2, p. 181–192, 2022. ISSN 2008-2045. DOI: 10.22042/isecure.2021.262846.595. Citado nas pp. 24, 28, 41, 73.

MANAVI, Farnoush; HAMZEH, Ali. Static Detection of Ransomware Using LSTM Network and PE Header. In: 2021 26th International Computer Conference, Computer Society of Iran (CSICC). 2021. P. 1–5. DOI: 10.1109/CSICC52343.2021.9420580. Citado nas pp. 24, 28, 41, 73.

MANTHENA, Harikha; KIMMEL, Jeffrey C.; ABDELSALAM, Mahmoud; GUPTA, Maanak. Analyzing and Explaining Black-Box Models for Online Malware Detection. **IEEE Access**, v. 11, p. 25237–25252, 2023. DOI: 10.1109/ACCESS.2023.3255176. Citado na p. 50.

MANTOVANI, Alessandro; AONZO, Simone; UGARTE-PEDRERO, Xabier; MERLO, Alessio; BALZAROTTI, Davide. Prevalence and Impact of Low-Entropy Packing Schemes in the Malware Ecosystem. In: NETWORK and Distributed Systems Security (NDSS) Symposium 2020. San Diego, CA, USA, fev. 2020. ISBN 1-891562-61-4. DOI: 10.14722/ndss.2020.24297. Citado na p. 35.

MELAND, Per Håkon; BAYOUMY, Yara Fareed Fahmy; SINDRE, Guttorm. The Ransomware-as-a-Service economy within the darknet. **Computers & Security**, v. 92, p. 101762, 2020. ISSN 0167-4048. DOI: 10.1016/j.cose.2020.101762. Citado nas pp. 19, 21.

MI, Jian-Xun; LI, An-Di; ZHOU, Li-Fang. Review Study of Interpretation Methods for Future Interpretable Machine Learning. **IEEE Access**, v. 8, p. 191969–191985, 2020. DOI: 10.1109/ACCESS.2020.3032756. Citado nas pp. 49, 51.

MICROSOFT. **PE Format**. Mar. 2023. Disponível em: <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>. Acesso em: 20 ago. 2023. Citado nas pp. 32, 34.

MICROSOFT. **Unicode in the Windows API**. Jul. 2021. Disponível em: <https://learn.microsoft.com/en-us/windows/win32/intl/unicode-in-the-windows-api>. Acesso em: 20 ago. 2023. Citado na p. 52.

MICROSOFT. **Using Nt and Zw Versions of the Native System Services Routines**. Dez. 2021. Disponível em: <https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/using-nt-and-zw-versions-of-the-native-system-services-routines>. Acesso em: 20 ago. 2023. Citado na p. 52.

MILOŠEVIĆ, Nikola. **History of malware**. arXiv, 2013. DOI: 10.48550/ARXIV.1302.5392. Citado na p. 17.

MITCHELL, Rory; COOPER, Joshua; FRANK, Eibe; HOLMES, Geoffrey. Sampling Permutations for Shapley Value Estimation. **Journal of Machine Learning Research**, v. 23, n. 43, p. 1–46, 2022. Disponível em: <http://jmlr.org/papers/v23/21-0439.html>. Citado na p. 50.

MOLNAR, Christoph. **Interpretable machine learning**. 2. ed., 2022. Disponível em: <https://christophm.github.io/interpretable-ml-book/>. Citado na p. 50.

MORCOS, Martina; AL HAMADI, Hussam; DAMIANI, Ernesto; NANDYALA, Sivaprasad; MCGILLION, Brian. A Surrogate-Based Technique for Android Malware Detectors' Explainability. In: 2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). 2022. P. 112–117. DOI: 10.1109/WiMob55322.2022.9941515. Citado na p. 50.

MOREIRA, Caio; MOREIRA, Davi; SALES JR., Claudomiro. Improving ransomware detection based on portable executable header using xception convolutional neural network. **Computers & Security**, v. 130, p. 103265, jul. 2023. ISSN 0167-4048. DOI: 10.1016/j.cose.2023.103265. Citado nas pp. 27, 28, 41, 73, 74.

MOUSSAILEB, Routa; CUPPENS, Nora; LANET, Jean-Louis; BOUDER, Hélène Le. A Survey on Windows-Based Ransomware Taxonomy and Detection Mechanisms. **ACM Computing Surveys**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 6, p. 117, jul. 2021. ISSN 0360-0300. DOI: 10.1145/3453153. Citado nas pp. 18, 20, 21, 31.

ORMEIR, Ori; NISSIM, Nir; ELOVICI, Yuval; ROKACH, Lior. Dynamic Malware Analysis in the Modern Era—A State of the Art Survey. **ACM Computing Surveys**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 5, set. 2019. ISSN 0360-0300. DOI: 10.1145/3329786. Citado na p. 17.

OZ, Harun; ARIS, Ahmet; LEVI, Albert; ULUAGAC, A. Selcuk. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. **ACM Computing Surveys**, Association for Computing Machinery, New York, NY, USA, v. 54, 11s, set. 2022. ISSN 0360-0300. DOI: 10.1145/3514229. Citado nas pp. 19, 20, 30.

PAIK, Joon-Young; JIN, Rize; CHO, Eun-Sun. Malware classification using a byte-granularity feature based on structural entropy. **Computational Intelligence**, v. 38, n. 4, p. 1536–1558, 2022. DOI: 10.1111/coin.12521. Citado nas pp. 22, 28.

POUDYAL, Subash; DASGUPTA, Dipankar; AKHTAR, Zahid; GUPTA, Kishor. A multi-level ransomware detection framework using natural language processing and machine learning. In: 14TH International Conference on Malicious and Unwanted Software" MALCON. 2019. Citado na p. 36.

POWERS, David M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, 2011. ISSN 2229-3981. DOI: 10.48550/arXiv.2010.16061. Citado na p. 48.

PREUVENEERS, Davy; JOOSEN, Wouter. Sharing Machine Learning Models as Indicators of Compromise for Cyber Threat Intelligence. **Journal of Cybersecurity and Privacy**, v. 1, n. 1, p. 140–163, 2021. ISSN 2624-800X. DOI: 10.3390/jcp1010008. Citado na p. 20.

RAY, Susmita. A Quick Review of Machine Learning Algorithms. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). 2019. P. 35–39. DOI: 10.1109/COMITCon.2019.8862451. Citado na p. 49.

REZAEI, Tina; MANAVI, Farnoush; HAMZEH, Ali. A PE header-based method for malware detection using clustering and deep embedding techniques. **Journal of Information Security and Applications**, v. 60, p. 102876, 2021. ISSN 2214-2126. DOI: 10.1016/j.jisa.2021.102876. Citado nas pp. 22, 28.

RIZVI, Syed Khurram Jah; ASLAM, Warda; SHAHZAD, Muhammad; SALEEM, Shahzad; FRAZ, Muhammad Moazam. PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable. **Complex & Intelligent Systems**, v. 8, n. 1, p. 673–685, fev. 2022. ISSN 2198-6053. DOI: 10.1007/s40747-021-00560-1. Citado nas pp. 22, 28.

SGANDURRA, Daniele; MUÑOZ-GONZÁLEZ, Luis; MOHSEN, Rabih; LUPU, Emil C. **Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection**. 2016. DOI: 10.48550/arXiv.1609.03020. Citado nas pp. 26–28, 73, 74.

SHAUKAT, Saiyed Kashif; RIBEIRO, Vinay J. RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In: 2018 10th International Conference on Communication Systems & Networks (COMSNETS). 2018. P. 356–363. DOI: 10.1109/COMSNETS.2018.8328219. Citado nas pp. 26, 28, 73, 74.

SIHWAIL, Rami; OMAR, Khairuddin; ZAINOL ARIFFIN, Khairul Akram; AL AFGHANI, Sanad. Malware Detection Approach Based on Artifacts in Memory Image and Dynamic Analysis. **Applied Sciences**, v. 9, n. 18, 2019. ISSN 2076-3417. DOI: 10.3390/app9183680. Citado na p. 20.

SINGH, Jagsir; SINGH, Jaswinder. A survey on machine learning-based malware detection in executable files. **Journal of Systems Architecture**, v. 112, p. 101861, 2021. ISSN 1383-7621. DOI: 10.1016/j.sysarc.2020.101861. Citado na p. 20.

SOLORIO-FERNÁNDEZ, Saúl; CARRASCO-OCHOA, J. Ariel; MARTÍNEZ-TRINIDAD, José Fco. A review of unsupervised feature selection methods. **Artificial Intelligence Review**, v. 53, n. 2, p. 907–948, fev. 2020. ISSN 1573-7462. DOI: 10.1007/s10462-019-09682-y. Citado na p. 45.

STIAWAN, Deris; DAELY, Somame Morianus; HERYANTO, Ahmad; AFIFAH, Nurul; IDRIS, Mohd Yazid; BUDIARTO, Rahmat. Ransomware Detection Based On Opcode Behavior Using K-Nearest Neighbors Algorithm. **Information Technology and Control**, v. 50, n. 3, p. 495–506, 2021. DOI: 10.5755/j01.itc.50.3.25816. Citado nas pp. 23, 28, 73.

TORRES, Edgar; YOO, Sang Guun. Detecting and Neutralizing Encrypting Ransomware Attacks by Using Machine-Learning Techniques: A Literature Review. **International Journal of Applied Engineering Research**, v. 12, n. 18, p. 7902–7911, mai. 2017. ISSN 0973-4562. Disponível em: <https://www.ripublication.com/ijaer17/ijaerv12n18_105.pdf>. Citado na p. 20.

UCCI, Daniele; ANIELLO, Leonardo; BALDONI, Roberto. Survey of machine learning techniques for malware analysis. **Computers & Security**, v. 81, p. 123–147, 2019. ISSN 0167-4048. DOI: 10.1016/j.cose.2018.11.001. Citado na p. 20.

UROOJ, Umara; AL-RIMY, Bander Ali Saleh; ZAINAL, Anazida; GHALEB, Fuad A.; RASSAM, Murad A. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. **Applied Sciences**, v. 12, n. 1, 2022. ISSN 2076-3417. DOI: 10.3390/app12010172. Citado na p. 20.

VEHABOVIC, A.; ZANDDIZARI, H.; GHANI, N.; SHAIKH, F.; BOU-HARB, E.; POUR, M. Safaei; CRICHIGNO, J. Data-Centric Machine Learning Approach for Early Ransomware Detection and Attribution. In: NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium. 2023. P. 1–6. DOI: 10.1109/NOMS56928.2023.10154378. Citado nas pp. 26, 28, 73, 74.

VERMA, Mayank; KUMARGURU, Ponnurangam; BRATA DEB, Shuva; GUPTA, Anuradha. Analysing Indicator of Compromises for Ransomware: Leveraging IOCs with Machine Learning Techniques. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI). 2018. P. 154–159. DOI: 10.1109/ISI.2018.8587409. Citado na p. 20.

VIDYARTHI, Deepti; KUMAR, CRS; RAKSHIT, Subrata; CHANSARKAR, Shailesh. Static Malware Analysis to Identify Ransomware Properties. **International Journal of Computer Science Issues**, v. 16, n. 3, p. 10–17, mai. 2019. DOI: 10.5281/zenodo.3252963. Citado nas pp. 25, 28, 72, 73.

VIRUSTOTAL. **Ransomware in a global context**. Out. 2021. Disponível em: <https://assets.virustotal.com/reports/ransomware-in-a-global-context-2021>. Acesso em: 13 fev. 2024. Citado na p. 21.

WADKAR, Mayuri; DI TROIA, Fabio; STAMP, Mark. Detecting malware evolution using support vector machines. **Expert Systems with Applications**, v. 143, p. 113022, 2020. ISSN 0957-4174. DOI: 10.1016/j.eswa.2019.113022. Citado na p. 42.

WANG, Maonan; ZHENG, Kangfeng; YANG, Yanqing; WANG, Xiujuan. An Explainable Machine Learning Framework for Intrusion Detection Systems. **IEEE Access**, v. 8, p. 73127–73141, 2020. DOI: 10.1109/ACCESS.2020.2988359. Citado na p. 50.

WANG, Zhongru; CONG, Peixin; YU, Weiqiang. Malicious Code Detection Technology Based on Metadata Machine Learning. In: 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC). 2020. P. 394–399. DOI: 10.1109/DSC50466.2020.00068. Citado na p. 18.

WRESSNEGGER, Christian; FREEMAN, Kevin; YAMAGUCHI, Fabian; RIECK, Konrad. Automatically Inferring Malware Signatures for Anti-Virus Assisted Attacks. In: PROCEEDINGS of the 2017 ACM on Asia Conference on Computer and Communications Security. Abu Dhabi, United Arab Emirates: Association for Computing Machinery, 2017. (ASIA CCS '17), p. 587–598. ISBN 9781450349444. DOI: 10.1145/3052973.3053002. Citado na p. 18.

WU, Qing; ZHU, Xueling; LIU, Bo. A Survey of Android Malware Static Detection Technology Based on Machine Learning. **Mobile Information Systems**, Hindawi, v. 2021, p. 8896013, mar. 2021. ISSN 1574-017X. DOI: 10.1155/2021/8896013. Citado nas pp. 22, 25.

YOUSUF, Muhammad Irfan; ANWER, Izza; RIASAT, Ayesha; ZIA, Khawaja Tahir; KIM, Suhyun. Windows malware detection based on static analysis with multiple features. **PeerJ Computer Science**, PeerJ Inc., v. 9, e1319, 2023. DOI: 10.7717/peerj-cs.1319. Citado nas pp. 22, 28, 42.

ZAHOORA, Umme; RAJARAJAN, Muttukrishnan; PAN, Zahoqing; KHAN, Asifullah. Zero-day Ransomware Attack Detection using Deep Contractive Autoencoder and Voting based Ensemble Classifier. **Applied Intelligence**, mar. 2022. (2022). ISSN 1573-7497. DOI: 10.1007/s10489-022-03244-6. Citado nas pp. 26, 28, 74.

ZHANG, Bin; XIAO, Wentao; XIAO, Xi; SANGAIAH, Arun Kumar; ZHANG, Weizhe; ZHANG, Jiajia. Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes. **Future Generation Computer Systems**, Elsevier B.V., v. 110, p. 708–720, set. 2020. ISSN 0167739X. DOI: 10.1016/j.future.2019.09.025. Citado nas pp. 23, 28, 36, 73.

ZHANG, Hanqi; XIAO, Xi; MERCALDO, Francesco; NI, Shiguang; MARTINELLI, Fabio; SANGAIAH, Arun Kumar. Classification of ransomware families with machine learning based

on N-gram of opcodes. **Future Generation Computer Systems**, v. 90, p. 211–221, 2019. ISSN 0167-739X. DOI: 10.1016/j.future.2018.07.052. Citado nas pp. 23, 28, 36, 37, 72, 73.

ZHU, Jinting; JANG-JACCARD, Julian; SINGH, Amardeep; WELCH, Ian; AI-SAHAF, Harith; CAMTEPE, Seyit. A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. **Computers & Security**, v. 117, p. 102691, 2022. ISSN 0167-4048. DOI: 10.1016/j.cose.2022.102691. Citado nas pp. 24, 26, 28, 35, 72, 73.

APÊNDICE A — RELATÓRIOS DE SEGURANÇA CIBERNÉTICA

Tabela 14 – Sete relatórios de instituições globais de segurança cibernética sobre *ransomware*.

Instituição	Título do Relatório	Data
Cyber Security Networks	*Ransomware: Through the Lens of Threat and Vulnerability Management*	Fev-2022
EmsiSoft	*Ransomware statictis for 2021: Year in summary*	Jan-2022
Sophos	*Sophos 2022 Threat Report: Interrelated threats target an interdependent world*	Nov-2021
Trellix	*Advanced Threat Research Report*	Out-2021
VirusTotal	*Ransomware in a Global Context*	Out-2021
Palo Alto Networks	*Unit 42 - Ransomware Threat Report 2021*	Abr-2021
Group-IB	*Ransomware Uncovered 2020-2021*	Mar-2021

Fonte: Elaborado pelo autor

Tabela 15 – Cinco relatórios recentes de instituições globais de segurança cibernética sobre *ransomware*.

Instituição	Título do Relatório	Data
Sophos	Sophos 2023 Threat Report: Maturing criminal marketplaces present new challenges to defenders	Nov-2022
Cyber Security Networks	Ransomware: Through the Lens of Threat and Vulnerability Management Index Update Q2-Q3 2022	Ago-2022
Kroll	Q2 2022 Threat Landscape: Ransomware Returns, Healthcare Hit	Ago-2022
Zscaler	2022 ThreatLabz State of Ransomware Report	Jun-2022
Cyber Security Networks	Ransomware: Through the Lens of Threat and Vulnerability Management Index Update Q1 2022	Mai-2022

Fonte: Elaborado pelo autor

APÊNDICE B — INTERVALOS DE HIPERPARÂMETROS

Tabela 16 – Intervalos de hiperparâmetros investigados para cada modelo experimental.

Modelo	Hiperparâmetro	Intervalo
EXT	criterion	[‘gini’, ‘entropy’]
	max_depth	[None, 50, 100, 150, 200, 250, 300, 350]
	max_features	[‘log2’, ‘sqrt’, 5, 10, 15, 20, 30]
	n_estimators	[100, 125, 150, 175, 200, 225, 250]
KNN	metric	[‘minkowski’, ‘euclidean’, ‘manhattan’]
	n_neighbors	[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	weights	[‘uniform’,‘distance’]
LDA	n_components	[None, 1, 2, 3]
	solver	[‘svd’, ‘lsqr’, ‘eigen’]
	tol	[5.0e-05, 1.0e-4, 1.5e-04, 2.0e-04, 2.5e-04, 3.0e-04]
LR	C	[0.5, 1, 2, 3, 4, 5, 10, 20, 25, 30]
	max_iter	[60, 80, 100, 120, 140, 160, 180, 200]
	penalty	[‘l1’, ‘l2’, ‘elasticnet’]
	solver	[‘newton-cg’, ‘lbfgs’, ‘liblinear’]
MLP	activation	[‘identity’, ‘logistic’, ‘tanh’, ‘relu’]
	early_stopping	[True, False]
	learning_rate	[‘constant’, ‘invscaling’, ‘adaptive’]
	solver	[‘lbfgs’, ‘sgd’, ‘adam’]
RF	criterion	[‘gini’, ‘entropy’]
	max_depth	[None, 50, 100, 150, 200, 250, 300, 350]
	max_features	[‘log2’, ‘sqrt’, 5, 10, 15, 20, 30]
	n_estimators	[100, 125, 150, 175, 200, 225, 250]
SGD	alpha	[0.1, 0.001, 0.0001, 0.00001]
	early_stopping	[True, False]
	loss	[‘hinge’, ‘log’, ‘perceptron’]
	penalty	[‘l2’, ‘l1’, ‘elasticnet’]
	tol	[5.0e-05, 1.0e-4, 1.5e-04, 2.0e-04, 2.5e-04, 3.0e-04]
SVM	C	[0.1, 0.5, 1, 3, 5, 10, 15, 20]
	kernel	[‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’]
	gamma	[0.001, 0.005, 0.01, 0.1, 0.5]
XGB	booster	[‘gbtree’]
	colsample_bytree	[0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
	learning_rate	[0.05, 0.1, 0.3, 0.5, 0.8, 1.0]
	max_depth	[3, 4, 5, 6, 7, 8, 9]
	n_estimators	[100, 125, 150, 175, 200, 225, 250]
	tree_method	[‘exact’, ‘approx’, ’hist’]

Fonte: Elaborado pelo autor

APÊNDICE C — HIPERPARÂMETROS AJUSTADOS

Tabela 17 – Hiperparâmetros utilizados na DNFR de melhor desempenho para cada conjunto de características. Valores ajustados em **'negrito'** e valores padrão em texto 'regular'.

Característica	Modelo	Hiperparâmetros e valores
Campos do Cabeçalho	XGB	**booster='gbtree'**, **colsample_bytree=0.6**, **learning_rate=0.1**, **max_depth=7**, **n_estimators=250**, **tree_method='hist'**, objective=binary:logistic, use_label_encoder=None, base_score=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, gpu_id=None, grow_policy=None, importance_type=None, interaction_constraints=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, n_jobs=None, num_parallel_tree=None, predictor=None, random_state=None, reg_alpha=None, reg_lambda=None, sampling_method=None, scale_pos_weight=None, subsample=None, validate_parameters=None, verbosity=None
DLLs Importadas	SVM	**C=15**, **gamma=0.1**, **kernel='poly'**, degree=3, coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None
Chamadas de Função	SVM	**C=5**, **gamma=0.01**, **kernel='rbf'**, degree=3, coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None
Entropia das Seções	EXT	**criterion='gini'**, **max_depth=None**, **max_features=15**, **n_estimators=200**, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=False, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
3-grama	XGB	**booster='gbtree'**, **colsample_bytree=0.6**, **learning_rate=0.8**, **max_depth=5**, **n_estimators=150**, **tree_method='hist'**, **
Combinação sem 3-grama	LR	**C=20**, **max_iter=180**, **penalty='l2'**, **solver='liblinear'**, dual=False, tol=0.0001, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None
	RF	**criterion='gini'**, **max_depth=None**, **max_features='sqrt'**, **n_estimators=125**, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
	XGB	**booster='gbtree'**, **colsample_bytree=0.7**, **learning_rate=0.1**, **max_depth=6**, **n_estimators=200**, **tree_method='hist'**, **

**Hiperparâmetros padrão iguais aos do XGB nos Campos de Cabeçalho.

Fonte: Elaborado pelo autor