UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DENIS HENRIQUE DOS SANTOS

APLICAÇÃO DE TÉCNICAS DE ENSEMBLE LEARNING NA DETECÇÃO ESTÁTICA E DINÂMICA DE MALWARES

BAURU
2023

DENIS HENRIQUE DOS SANTOS

APLICAÇÃO DE TÉCNICAS DE ENSEMBLE LEARNING NA DETECÇÃO ESTÁTICA E DINÂMICA DE MALWARES

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

BAURU
2023

S237a

Santos, Denis Henrique dos
Aplicação de técnicas de ensemble learning na detecção estática e dinâmica de malwares / Denis Henrique dos Santos. -- Bauru, 2023
25 f. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: Kelton Augusto Pontara da Costa

1. Inteligência artificial. 2. Aprendizado do computador. 3. Malware (Software de computador). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Denis Henrique dos Santos

Aplicação de Técnicas de Ensemble Learning na Detecção Estática e Dinâmica de Malwares

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

Banca Examinadora

Prof. Dr. Kelton Augusto Pontara da Costa
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profª. Drª. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Me. Luiz Felipe de Camargo
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, _____ de _____ de _____.

Resumo

Com o constante aumento do uso de tecnologia no dia a dia a importância da segurança da informação cresce drasticamente, levando à existência de *malwares* ser uma ameaça ao funcionamento de sistemas e dados importantes, requisitando formas de combatê-los. Modelos de *ensemble learning* são modelos de aprendizado de máquina que utilizam vários algoritmos para chegar a resultados de um determinado problema. Este trabalho propõe um sistema de detecção de *malware* utilizando modelos *ensemble learning* dos tipos *bagging*, *boosting* e *stacking*, usando o conjunto de dados *Malevis* para análise estática e o conjunto de dados *Top-1000 PE Imports* para análise dinâmica. Os modelos foram treinados e os resultados foram comparados com outro sistema de detecção de *malware*. Os resultados obtidos foram satisfatórios, especialmente os da análise dinâmica, com modelos do tipo *bagging* e *stacking* mostrando maior desempenho em ambos os cenários, enquanto que os modelos do tipo *boosting* apresentaram maior dificuldade em chegar a resultados mais eficazes com os conjuntos de dados usados.

Palavras-chave: *Malware*; Aprendizado de máquina; *Ensemble learning*.

Abstract

With the constant increase of technology use in everyday life the importance of information security drastically rises, leading to the existence of malware being a threat to the operation of systems and important data, requiring ways to combat them. Ensemble learning models are machine learning models that use many algorithms to reach results in a geiven problem. This work proposes a malware detection system using bagging, boosting and stacking ensemble learning models, using the Malevis dataset for static analysis and the Top-1000 PE Imports dataset for dynamic analysis. The models were trained and the results compared with another malware detection system. The obtained results were satisfactory, especially the ones from the dynamic analysis, with bagging and stacking models showing better performance in both scenarios, while the boosting models presented greater difficulty in arriving at more effective results with the datasets used.

Keywords: Malware; Machine learning; Ensemble learning.

Lista de figuras

Lista de tabelas

Sumário

1 Introdução

Malware, termo oriundo da aglutinação das palavras *malicious* e *software*, se refere a qualquer *software* de cunho malicioso que busca entrar em um sistema sem a autorização de seu usuário (VINOD et al., 2009). A quantidade de *malware* existente aumenta mais rapidamente a cada ano, representando uma ameaça à segurança em computadores, o que consequentemente torna a detecção de *malwares* um tópico de suma importância (SANTOS et al., 2013).

A detecção de *malware* pode ser fundamentalmente categorizada em diferentes métodos baseados em: assinaturas, rápidas porém incapazes de detectar ameaças desconhecidas e necessitam de alto esforço para registrar novas assinaturas; comportamento, capaz de detectar novas ameaças e possíveis mutações porém não possuem uma taxa promisora de falsos positivos e consomem mais tempo de análise; heurística, proposta para superar as desvantagens dos outros métodos através de técnicas de *machine learning* e mineração de dados (BAZRAFSHAN et al., 2013).

Este trabalho foi baseado na implementação de redes neurais sem peso na detecção estática e dinâmica de *malware* feita por Ramos et al. (2020), substituindo a rede neural sem pesos por modelos baseados em técnicas *ensemble learning* no ambiente recriado e foi feita comparações entre os resultados das implementações, com o intuito de determinar qual delas é mais eficaz.

1.1 Problema

Durante a era digital ataques utilizando *malware* constituem um grande problema dentro da área de segurança, responsáveis por impactar um alto número de dispositivos, sendo capazes de comprometê-los, furtar informações confidenciais assim como penetrar redes (RATHORE et al., 2018).

Devido ao risco existente de infecção por *malware*, sua detecção se torna algo de importância para a cibersegurança, porém a proteção que programas antivírus providenciam se tornaram menos efetivas com o tempo, o que levou ao uso de técnicas de *machine learning* para obter maior eficácia contra *malware*, porém ainda existem desafios significantes (RAFF et al., 2018).

Embora vários métodos de detecção de *malware* com diferentes abordagens tenham sido propostos, não existe um método capaz de detectar todos os *malware* novos e sofisticados, o que torna a construção de um método efetivo uma tarefa desafiadora (ASLAN; SAMET, 2020).

1.2 Justificativa

O contínuo crescimento da quantidade de programas maliciosos existentes leva à necessidade de garantir a segurança da informação através de métodos cada vez melhores, instigando a necessidade de desenvolver e implementar métodos defensivos novos para garantir a cibersegurança.

Assim justifica-se o estudo e busca de métodos de detecção de *malware*, buscando soluções relativamente rápidas e com maior exatidão possível para alcançar resultados precisos com um tempo gasto razoável.

1.3 Objetivos

O objetivo geral e específicos pertinentes a este trabalho estão apresentados a seguir.

1.3.1 Objetivo Geral

Este trabalho teve como objetivo desenvolver um sistema de detecção estática e dinâmica de *malware* utilizando modelos *ensemble learning* baseado no trabalho de Ramos et al. (2020) e analisar os resultados de seu funcionamento.

1.3.2 Objetivos Específicos

- Estudar conceitos referentes a *malware* e *ensemble learning*;
- Recriar o ambiente proposto por Ramos et al. (2020);
- Processar o conjunto de dados utilizando modelos *ensemble learning*;
- Analisar os resultados e compará-los com o modelo original.

1.4 Organização da Monografia

Este trabalho está estruturado em cinco capítulos. O capítulo 1 contextualiza o trabalho com o problema, justificativa e objetivos. No capítulo 2 é apresentada a fundamentação teórica que serve como base do trabalho. O capítulo 3 apresenta as ferramentas e metodologia utilizadas neste trabalho. O capítulo 4 discute os resultados obtidos e o capítulo 5 apresenta as conclusões obtidas do trabalho.

2 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica em que este trabalho foi baseado, abordando os temas estudados.

2.1 Aprendizado de Máquina

Aprendizado de máquina, ou *Machine Learning*, se trata de um campo da Inteligência Artificial que visa estudar e construir métodos e algoritmos, criando sistemas que adquirem informações automaticamente, tomando decisões baseadas em suas experiências (MONARD; BARANAUSKAS, 2003), o que possibilita, por exemplo, reconhecimento de imagens e traduções automatizadas.

Modelos de aprendizado de máquina aprendem a partir de conjuntos de dados, composto por elementos denominados de amostras ou *samples*, contendo valores que serão estudados chamados de características ou *features*, podendo ter um valor de rótulo que determina uma classe para tal elemento ou não.

O aprendizado de um modelo pode ser classificado de várias maneiras, entre elas:

- **Aprendizado supervisionado**: é apresentado ao algoritmo um conjunto com todos os dados rotulados para que o seja construído um modelo que aprenda a mapear as características para os rótulos.
- **Aprendizado não supervisionado**: o conjunto de dados usado não possui nenhum rótulo, pode ser usado para encontrar padrões ou valores no conjunto.
- **Aprendizado semi-supervisionado**: existem dados rotulados e não rotulados no conjunto, tendo o mesmo objetivo do aprendizado supervisionado.
- **Aprendizado por reforço**: o algoritmo se encontra interagindo com um ambiente dinâmico, onde o algoritmo é punido ou recompensado de acordo com suas ações, levando ao algoritmo procurar o melhor conjunto de ações possível.

Os modelos de aprendizado de máquina também podem ser classificados de acordo com o tipo de saída desejada:

- **Classificação**: os dados são divididos em diferentes classes discretas e o modelo deve determinar qual classe cada dado pertence.

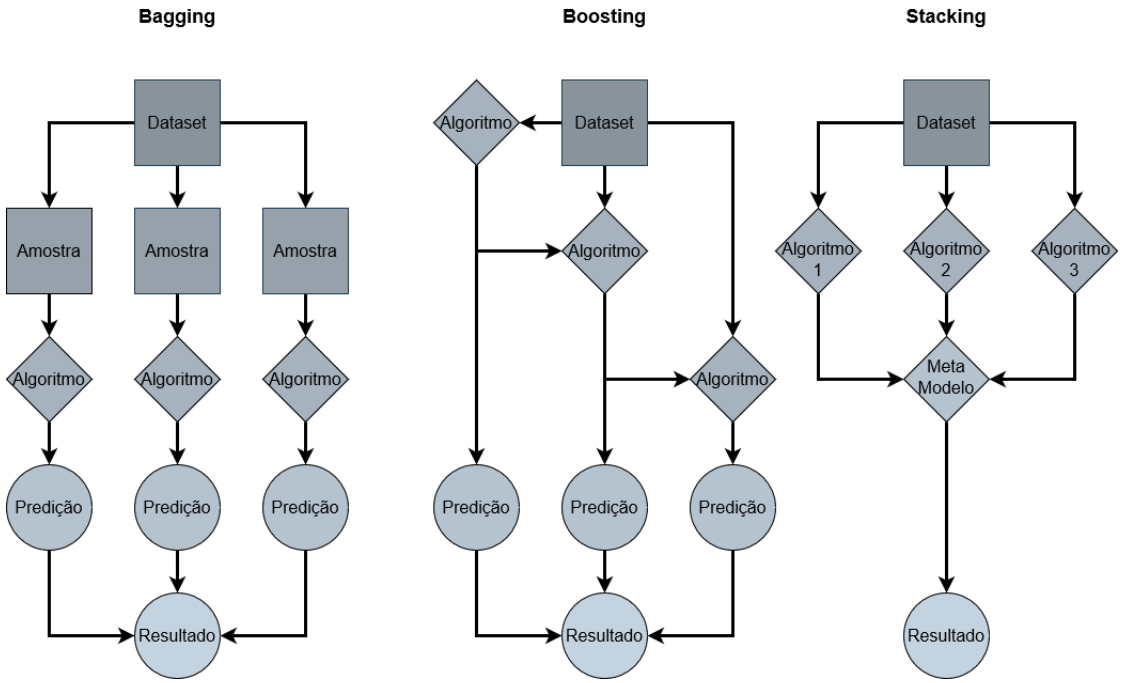
- **Regressão**: na regressão o modelo deve predizer um valor numérico contínuo para cada dado.
- **Clusterização**: o modelo procura agrupar amostras em grupos semelhantes.

2.2 *Ensemble Learning*

Modelos de aprendizado de máquina do tipo *ensemble learning* se tratam de modelos que usam do processo em que vários modelos são gerados e combinados estrategicamente para solucionar um problema (MAHESH, 2020).

Algoritmos de aprendizado de máquina comuns funcionam procurando por um espaço de funções chamada hipótese a fim de encontrar a função *h* que seja a melhor aproximação de certa função desconhecida *f*, o algoritmo pode então determinar a melhor função *h* testando com os dados disponíveis. Já modelos *ensemble* utilizam um conjunto de hipóteses que fazem uma votação para decidir o resultado final, aumentando a probabilidade de estar correto (DIETTERICH, 2002). Logo a ideia de *ensemble learning* é empregar múltiplos modelos para combinar suas predições (SEWELL, 2008).

Figura 1 – Representação de classificação de modelos *ensemble*



Fonte: Elaborado pelo autor.

Modelos *ensemble* podem ser classificados de várias maneiras, como as mais comuns que estão representadas na figura 1, sendo elas:

- ***Bagging***: os dados do conjunto são escolhidos aleatoriamente e distribuídos para várias instâncias de um algoritmo, de modo que cada modelo de algoritmo seja treinado paralelamente com dados diferentes. O resultado final é obtido através de uma votação ou média das predições dos modelos paralelos.
- ***Boosting***: são criados modelos fracos que são treinados com o conjunto de dados sucessivamente, recebendo informações dos modelos anteriores cujos erros influenciam no aprendizado de modelos futuros. No final todos os modelos juntam seus resultados em uma votação ou média para chegar na predição final.
- ***Stacking***: são treinados vários modelos de algoritmos diferentes que são então usados como entrada para um metamodelo que procura combinar as predições dos modelos base, podendo empilhar modelos em diferentes níveis com mais instâncias de *stacking*.

2.3 Detecção estática e dinâmica

As principais formas de análise para detecção de *malware* se tratam da análise estática, baseada no arquivo binário, e da análise dinâmica, baseada no comportamento de execução (NATARAJ et al., 2011).

Em métodos estáticos as informações são obtidas através da análise do código desmontado do programa, enquanto que métodos dinâmicos extraem dados a partir da execução do malware em um ambiente controlado (ISLAM et al., 2013).

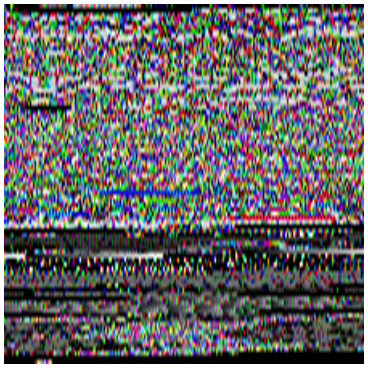
Cada método pode encontrar desafios referentes ao seu tipo de análise, técnicas de ofuscação binária podem dificultar a análise de métodos estáticos e os métodos dinâmicos encontram maiores dificuldades quando trabalhando com quantidades grandes de dados (NATARAJ et al., 2011).

2.4 Conjunto de dados *Malevis* e *Top-1000 PE Imports*

O conjunto de dados *Malevis* se trata de um conjunto de imagens em três canais de cor extraídas do arquivo binário de vários programas divididos em classes que representam *malware* ou não. O conjunto de dados possui duas versões de acordo com a resolução de cada imagem, podendo ser 224x224 ou 300x300 pixels (BOZKIR; CANKAYA; AYDOS, 2019).

Na figura 2 é apresentada uma das imagens presentes no conjunto de dados *Malevis*, esta em particular sendo de um *malware* que injeta anúncios nos navegadores de Internet que infecta.

Figura 2 – Imagem extraída do *malware BrowseFox*



Fonte: Conjunto de dados *Malevis*[1].

O conjunto de dados *Top-1000 PE Imports* contém as chamadas ao sistema operacional de vários programas que foram executados dentro do ambiente de análise de comportamento *Cuckoo Sandbox*, dividindo seus dados em *malwares* ou não (OLIVEIRA, 2019).

[1] Disponível em: <https://web.cs.hacettepe.edu.tr/~selman/malevis/>. Acesso em 01 de ago. de 2022.

3 Metodologia

Neste capítulo são apresentadas as ferramentas e métodos utilizados para desenvolver o trabalho.

3.1 Ferramentas

Estão apresentadas a seguir as ferramentas usadas durante o desenvolvimento deste trabalho.

3.1.1 *Python*

Python[2] é uma linguagem de programação orientada a objetos que possui uma sintaxe simples. O *Python* é uma das linguagens mais relevantes no campo de aprendizado de máquina, permitindo acesso a uma alta seleção de bibliotecas da área, como *scikit-learn*, *pandas* e *matplotlib*.

3.1.2 *Scikit-learn*

Scikit-learn[3] é uma biblioteca da linguagem *Python* que possui várias ferramentas na área de aprendizado de máquina, contendo ferramentas de pré-processamento de dados, implementações de modelos de aprendizado de máquina e funções de cálculo de métricas, como acurácia e precisão (PEDREGOSA et al., 2011).

3.1.3 *Imbalanced-learn*

O *imbalanced-learn*[4] é uma biblioteca compatível com o *scikit-learn* que providencia ferramentas para lidar com problemas de classificação com grande desequilíbrio de classes (LEMAîTRE; NOGUEIRA; ARIDAS, 2017).

3.1.4 *Pandas*

Pandas[5] é uma biblioteca de análise e manipulação de dados que pode ser utilizada no *Python*, tendo ferramentas para ler, armazenar e escrever dados mais facilmente (MCKINNEY, 2010).

---

[2] Disponível em: <https://www.python.org/>.
[3] Disponível em: <https://scikit-learn.org/stable/>.
[4] Disponível em: <https://imbalanced-learn.org/stable/>.
[5] Disponível em: <https://pandas.pydata.org/>.

3.1.5 *Matplotlib*

A biblioteca *matplotlib*[6] é uma biblioteca gráfica do *Python* que auxilia na visualização de dados de um modo simples.

3.1.6 *Scikit-image*

Scikit-image[7] é uma biblioteca da linguagem *Python* que possui ferramentas de processamento de imagens (WALT et al., 2014).

3.2 Pré-processamento de dados

Antes de utilizar os conjuntos de dados foi necessário fazer um pré-processamento para que os modelos a serem treinados consigam interpretá-los e para aumentar a qualidade dos resultados obtidos.

O conjunto de dados *Top-1000 PE Imports*, usado na detecção dinâmica, necessitou de menos passos de pré-processamento, apenas retirando a característica *hash* por ser apenas usada para identificar amostras.

O conjunto de dados da análise estática, *Malevis*, recebeu um pré-processamento seguindo o utilizado em Ramos et al. (2020) para que o ambiente seja recriado o mais semelhante possível.

Primeiramente, utilizando o *scikit-image*, as imagens de resolução 224x224 foram transformadas em escala de cinza, redimensionadas para o tamanho 128x128 pixels e os valores foram vetorizados em um vetor de tamanho 16384.

Utilizando os métodos de pré-processamento do *scikit-learn* o vetor sofreu uma binarização onde os valores entre 0 e 127 receberam valor 0 e os valores entre 128 e 255 receberam valor 1. Cada posição do vetor então passou a representar uma característica do banco de dados e cada imagem transformada passou a representar uma amostra.

Os dados de ambos os conjuntos foram então balanceados utilizando o *imbalanced-learn*, pois nos dois conjuntos de dados o número de amostras da classe *malware* era muito maior que o número de amostras da classe benigna, o que pode impactar negativamente os resultados gerados, criando um modelo enviesado.

3.3 Treinamento

Os dois conjuntos de dados, após prontos para serem interpretados, foram então treinados em três tipos diferentes de modelo cada, totalizando seis modelos. Os modelos

[6] Disponível em: <https://matplotlib.org/>.
[7] Disponível em: <https://scikit-image.org/>.

treinados com o conjunto *Malevis* foram denominados de detecção estática e os que foram treinados com o conjunto *Top-1000 PE Imports* foram denominados de detecção dinâmica.

Cada tipo de modelo foi escolhido de acordo com as classificações mais comuns de modelos de *ensemble learning*: *bagging*, *boosting* e *stacking*. Para cada modelo foi escolhido um algoritmo de classificação presente no *scikit-learn*, sendo utilizado o *RandomForestClassifier* para o *bagging*, o *AdaBoostClassifier* para o *boosting* e o *StackingClassifier* para o *stacking*.

O *RandomForestClassifier* implementa um algoritmo de floresta aleatória, onde várias árvores de decisão trabalham juntas, cada uma focando em características escolhidas aleatoriamente. O *AdaBoostClassifier* foca nos erros de seus modelos fracos, mudando o peso dos valores dos dados para tentar corrigir tais erros. O *StackingClassifier* junta vários modelos, como os próprios *RandomForestClassifier* e *AdaBoostClassifier*, e os combina em um modelo novo.

4 Resultados

Neste capítulo são apresentadas as métricas utilizadas para analisar os resultados, assim como os próprios resultados obtidos e sua comparação com os resultados obtidos em Ramos et al. (2020).

4.1 Métricas

Para analisar os resultados obtidos foram utilizadas certas métricas baseadas nas taxas:

- Verdadeiro Positivo VP, quando a predição que a amostra é *malware* está correta;
- Verdadeiro Negativo VN, quando a predição que a amostra não é *malware* está correta;
- Falso Positivo FP, quando a predição que a amostra é *malware* está errada;
- Falso Negativo FN, quando a predição que a amostra não é *malware* está errada.

A equação 4.1 representa a acurácia que se trata da razão das predições certas sobre todos os casos examinados. A precisão, definida pela equação 4.2, representa a porcentagem de predições de *malware* que foram classificadas corretamente. A taxa de revocação é a porcentagem de *malwares* que foram classificados corretamente, visto na equação 4.3. Na equação 4.4 se encontra a *F1-Score*, que se trata de uma média harmônica entre a precisão e revocação.

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN} \tag{4.1}$$

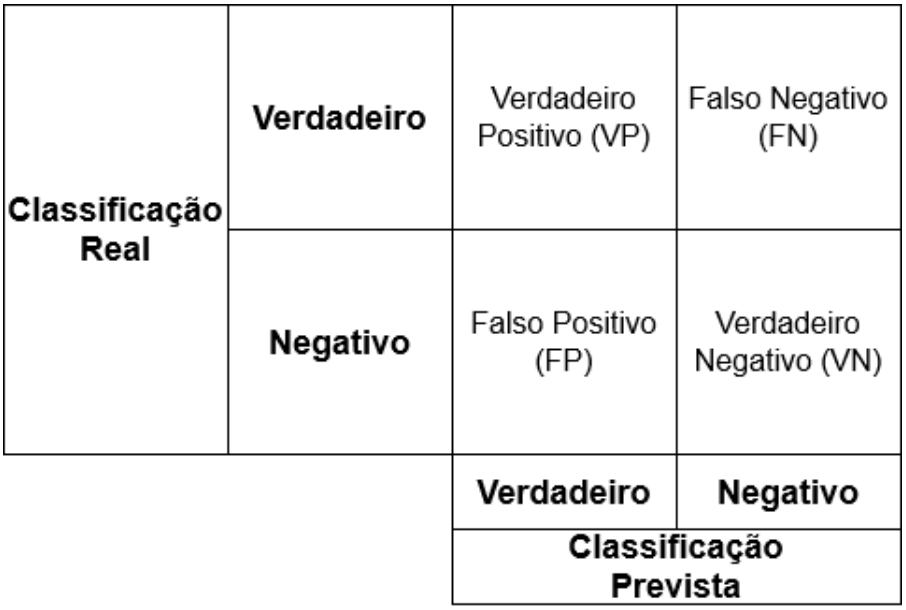
$$Precisão = \frac{VP}{VP + FP} \tag{4.2}$$

$$Revocação = \frac{VP}{VP + FN} \tag{4.3}$$

$$F1 - Score = \frac{2VP}{2VP + FP + FN} \tag{4.4}$$

Também foram usadas matrizes de confusão para facilitar a visualização das taxas usadas. Uma matriz de confusão é uma tabela que permite a visualização do desempenho de um algoritmo, um de seus eixos representa as classes reais dos dados e o outro eixo representa as classes previstas pelo algoritmo, como exemplificado na figura 3. Desse modo é possível visualizer a quantidade de predições corretas ou não, de acordo com a classe.

Figura 3 – Representação de uma matriz de confusão



Fonte: Elaborado pelo autor.

4.2 Análise dos Resultados

Abaixo nas tabelas 1 e 2 são apresentadas as métricas dos modelos de detecção dinâmica e estática, respectivamente, com os valores aproximados a 2 casas decimais. Em ambos os casos os modelos do tipo *boosting* foram os que mostraram menor eficiência em seus resultados, isso provavelmente se deve ao alto número de características do conjunto de dados impactando o desempenho do modelo. Abaixo também se encontram as matrizes de confusão dos modelos, apresentadas nas figuras 4, 5 e 6.

Tabela 1 – Resultados para detecção dinâmica

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
Bagging	95,94	97,15	95,36	96,25
Boosting	91,86	91,86	93,33	92,59
Stacking	96,14	97,05	95,83	96,44

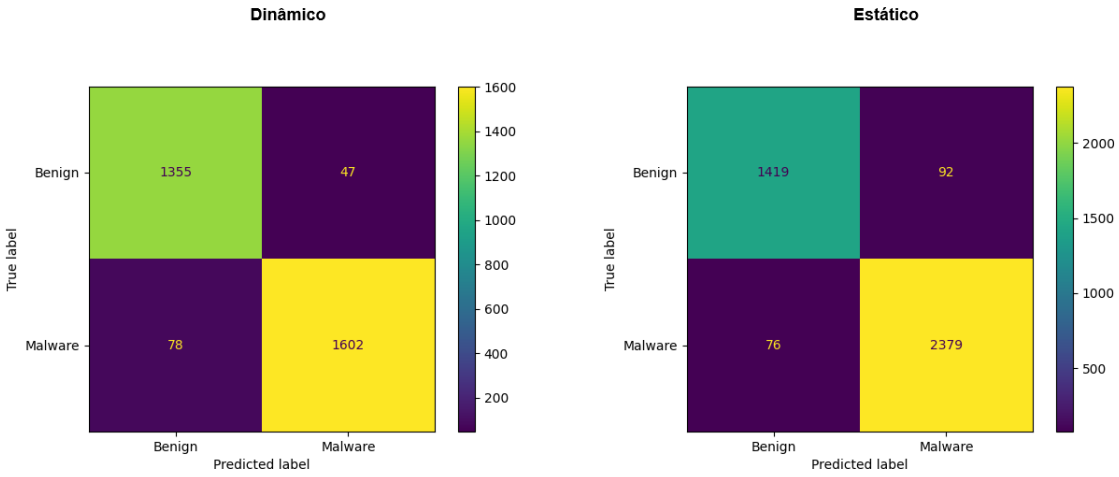
Fonte: Elaborado pelo autor.

Tabela 2 – Resultados para detecção estática

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
Bagging	95,76	96,28	96,90	96,59
Boosting	86,84	88,86	90,02	89,44
Stacking	95,84	95,58	97,80	96,68

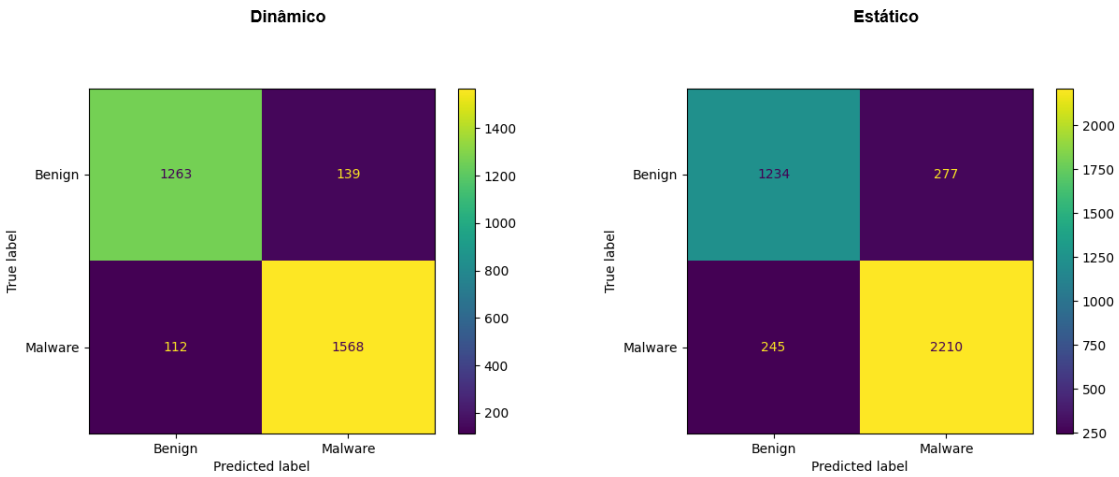
Fonte: Elaborado pelo autor.

Figura 4 – Matriz de confusão dos modelos tipo *bagging*



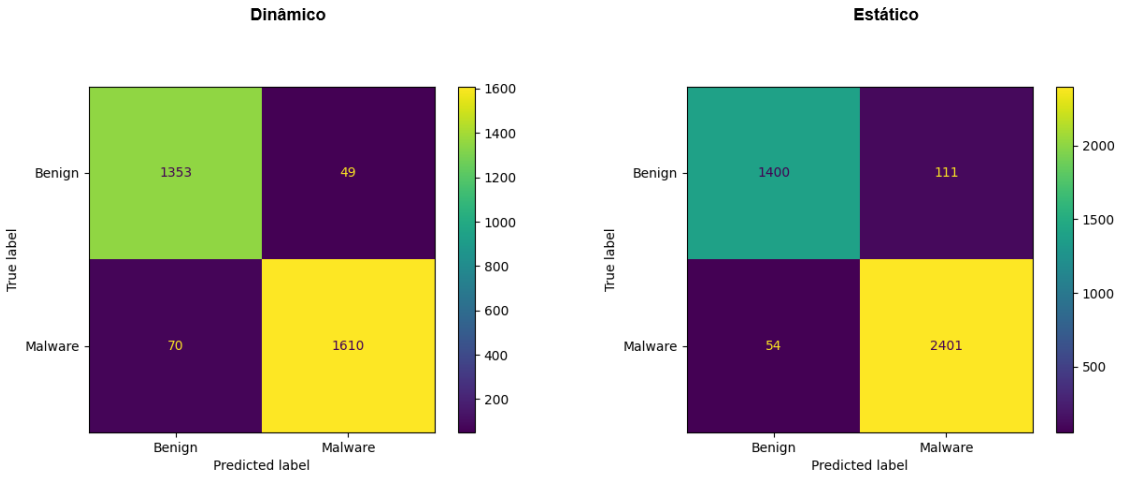
Fonte: Elaborado pelo autor.

Figura 5 – Matriz de confusão dos modelos tipo *boosting*



Fonte: Elaborado pelo autor.

Figura 6 – Matriz de confusão dos modelos tipo *stacking*



Fonte: Elaborado pelo autor.

Abaixo na tabela 3 são apresentados os valores dos resultados obtidos em Ramos et al. (2020) pelo modelo denominado *WiSARD*. O *WiSARD* se trata de uma rede neural, que é um tipo de modelo de aprendizado de máquina inspirado pelo sistema nervoso de animais, que armazena as informações aprendidas em memórias RAM.

Tabela 3 – Resultados do modelo WiSARD

Análise	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
Dinâmica	91,36	93,48	88,57	90,95
Estática	96,44	96,58	99,84	98,18

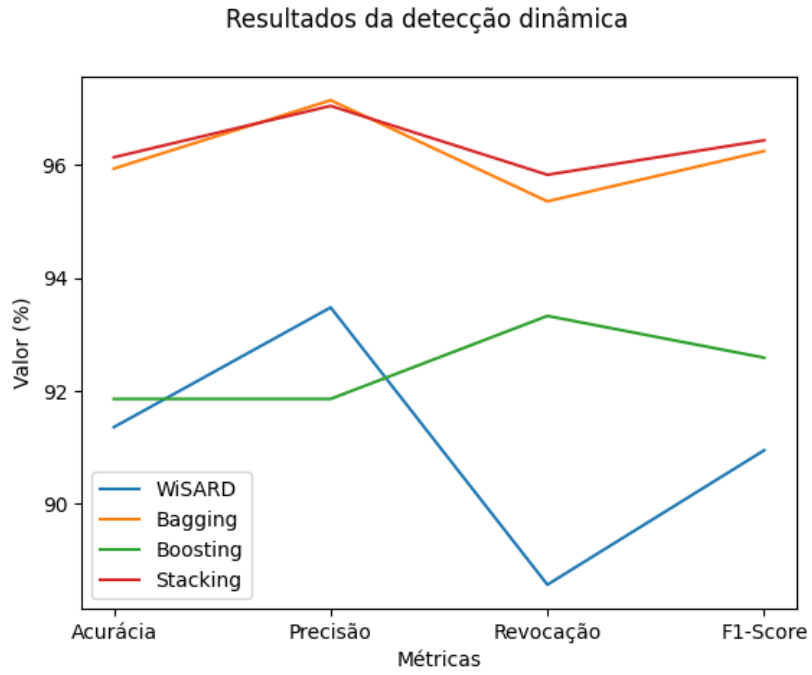
Fonte: Adaptado de Ramos et al. (2020).

Analisando os resultados obtidos percebe-se que os cenários de detecção dinâmica e estática produziram situações diferentes.

Na detecção dinâmica, melhor visualizada no gráfico da figura 7, todos os modelos *ensemble* alcançaram resultados melhores que o modelo WiSARD de Ramos et al. (2020), incluindo o modelo *boosting* que obteve menor eficácia dos modelos *ensemble*, porém com menor precisão que o modelo WiSARD.

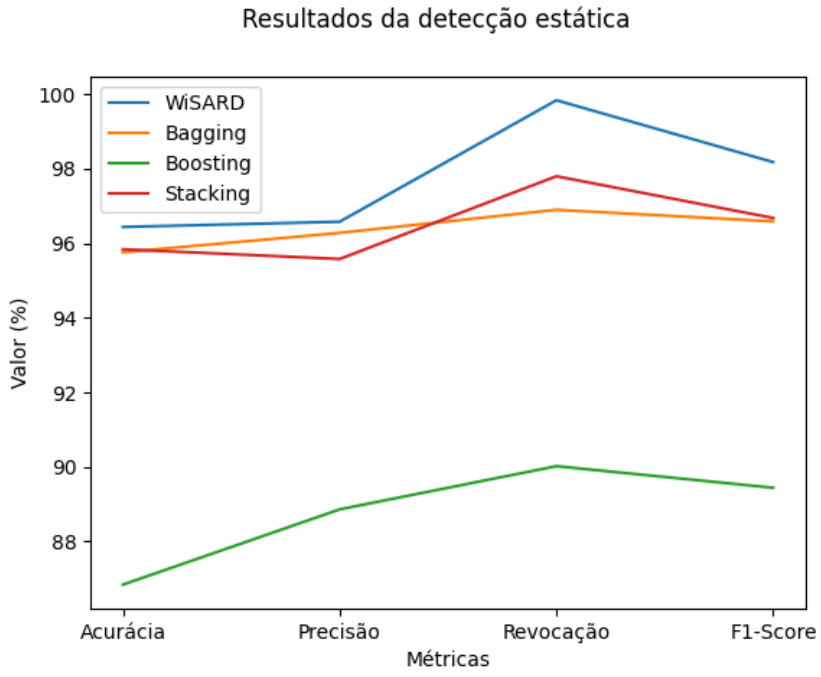
Já na detecção estática, representada na figura 8, os modelos *ensemble* todos mostraram desempenho inferior ao modelo WiSARD, com os modelos de *bagging* e *stacking* demonstrando resultados próximos aos de Ramos et al. (2020) e o modelo tipo *boosting* tendo resultados consideravelmente piores em comparação aos outros.

Figura 7 – Gráfico dos resultados dos modelos de detecção dinâmica



Fonte: Elaborado pelo autor.

Figura 8 – Gráfico dos resultados dos modelos de detecção estática



Fonte: Elaborado pelo autor.

5 Conclusão

Neste trabalho foram utilizadas técnicas de *ensemble learning* para desenvolver um sistema de detecção de *malware* baseado no trabalho de Ramos et al. (2020), que treinou um modelo de detecção estática e dinâmica de *malware*.

Os resultados obtidos foram parcialmente satisfatórios, com os modelos de detecção dinâmica apresentando resultados melhores que os vistos em Ramos et al. (2020), entretanto os modelos de detecção estática demonstraram resultados de menor qualidade, porém bastante próximos em valores, exceto pelos modelos tipo *boosting*, que demonstraram o pior desempenho dos modelos *ensemble* em ambos os tipos de detecção.

Ainda existe possibilidade de melhorar os métodos utilizados, como utilização de parâmetros ou algoritmos diferentes e pré-processar os conjuntos de dados de maneiras diferentes, logo os resultados de menor desempenho obtidos podem ser considerados satisfatórios por poderem contribuir no estudo da área.

Trabalhos futuros podem incluir a integração em um sistema real para medir sua eficiência com diferentes dados ou desenvolver uma análise utilizando outras técnicas de detecção, com a possibilidade de fazer todos os modelos trabalharem juntos para alcançar melhores resultados.

Referências

ASLAN, Ö. A.; SAMET, R. A comprehensive review on malware detection approaches. *IEEE Access*, IEEE, v. 8, p. 6249–6271, 2020.

BAZRAFSHAN, Z.; HASHEMI, H.; FARD, S. M. H.; HAMZEH, A. A survey on heuristic malware detection techniques. In: IEEE. *The 5th Conference on Information and Knowledge Technology*. [S.l.], 2013. p. 113–120.

BOZKIR, A. S.; CANKAYA, A. O.; AYDOS, M. Utilization and comparision of convolutional neural networks in malware recognition. In: IEEE. *2019 27th Signal Processing and Communications Applications Conference (SIU)*. [S.l.], 2019. p. 1–4.

DIETTERICH, T. G. Ensemble learning. *The handbook of brain theory and neural networks*, MIT press Cambridge, MA, USA, v. 2, n. 1, p. 110–125, 2002.

ISLAM, R.; TIAN, R.; BATTEN, L. M.; VERSTEEG, S. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications*, Elsevier, v. 36, n. 2, p. 646–656, 2013.

LEMAîTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, v. 18, n. 17, p. 1–5, 2017. Disponível em: http://jmlr.org/papers/v18/16-365.html. Acesso em: 2 de jan. de 2023.

MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, p. 381–386, 2020.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 56 – 61.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole, v. 1, n. 1, p. 32, 2003.

NATARAJ, L.; YEGNESWARAN, V.; PORRAS, P.; ZHANG, J. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. [S.l.: s.n.], 2011. p. 21–30.

OLIVEIRA, A. *Malware Analysis Datasets: Top-1000 PE Imports*. IEEE Dataport, 2019. Disponível em: https://dx.doi.org/10.21227/004e-v304. Acesso em: 16 de ago. de 2022.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

RAFF, E.; BARKER, J.; SYLVESTER, J.; BRANDON, R.; CATANZARO, B.; NICHOLAS, C. K. Malware detection by eating a whole exe. In: *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018.

RAMOS, L. C.; FILHO, L. A. L.; FRANÇA, F. M.; LIMA, P. M. Detecção estática e dinâmica de malwares usando redes neurais sem peso. In: SBC. *Anais do XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2020. p. 369–381.

RATHORE, H.; AGARWAL, S.; SAHAY, S. K.; SEWAK, M. Malware detection using machine learning and deep learning. In: SPRINGER. *International Conference on Big Data Analytics*. [S.l.], 2018. p. 402–411.

SANTOS, I.; BREZO, F.; UGARTE-PEDRERO, X.; BRINGAS, P. G. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, Elsevier, v. 231, p. 64–82, 2013.

SEWELL, M. Ensemble learning. *RN*, v. 11, n. 02, p. 1–34, 2008.

VINOD, P.; JAIPUR, R.; LAXMI, V.; GAUR, M. Survey on malware detection methods. In: *Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09)*. [S.l.: s.n.], 2009. p. 74–79.

WALT, S. van der; SCHÖNBERGER, J. L.; Nunez-Iglesias, J.; BOULOGNE, F.; WARNER, J. D.; YAGER, N.; GOUILLART, E.; YU, T.; CONTRIBUTORS the scikit-image. scikit-image: image processing in Python. *PeerJ*, v. 2, p. e453, 6 2014. ISSN 2167-8359. Disponível em: https://doi.org/10.7717/peerj.453. Acesso em: 2 de jan. de 2023.