Detectando Malwares com Técnicas Baseadas em Machine Learning

Gian Giovanni Rodrigues da Silva[1], Jessyca Jordanna Barroso de Moraes[1], Tammy Hikari Yanai Gusmão[1], Thalita Naiara Andre Alves [1]

[1]Escola Superior de Tecnologia
Universidade Estadual do Amazonas (UEA) – Manaus, AM – Brazil

{ggrds.cid20, jjbdm.cid20, thyg.cid20, tnaa.cid20}@uea.edu.br

Abstract. *The increasing dependence of companies on technologies connected to the Internet have made them constant targets of cyber attacks that aim at data theft and/or destruction. That highlights the importance of malware detection in order to maintain the business systems security. This paper compares three techniques based on Machine Learning, capable of detecting malware. Results show a greater performance from Random Forest, with 99.713% of correct prediction rate, followed by Decision Tree with 99.643% and KNN with 99.023%.*

Resumo. *O aumento da dependência das empresas sobre tecnologias ligadas à Internet os têm tornado alvos constantes de ataques cibernéticos que visam roubo e/ou destruição de dados. Isto evidencia a importância da detecção de malwares para que seja mantida a segurança dos sistemas empresariais. Este trabalho compara três técnicas, baseadas em Machine Learning, capazes de detectar malwares. Os resultados obtidos com a aplicação dos modelos mostram que Random Forest teve desempenho maior, com taxa de predição correta de 99.713%, seguido do Decision Tree com 99.643% e KNN com 99.023%.*

1. Introdução

Apesar do advento de mecanismos de cibersegurança (e sua contínua evolução), malwares ainda permanecem entre as ameaças mais eficazes na área de cibersegurança. Com o aumento da dependência na tecnologia, os *hackers* se vêem incentivados a melhorar, isto é, modificar e aumentar a complexidade de códigos maliciosos objetivando explorar falhas na segurança de sistemas.

Por definição, malware é um software malicioso desenvolvido com a intenção de violar uma política de segurança de sistemas de computador no que diz respeito à confidencialidade, integridade e disponibilidade de dados (Sethi et al., 2017). Este tipo de software pode ser categorizado em diferentes classes dependendo da forma como se comporta. Exemplos de categorias englobam vírus, *Trojan Horse*, *Spyware*, *Worm*, etc. (Shhadat et al., 2020). Malwares são comumente utilizados para desencadear um amplo escopo de ataques à segurança, por exemplo, derrubar sistemas, adquirir dados confidenciais, envio de mensagens de spam, incapacitar infraestruturas e penetrar redes. Esses ataques frequentemente levam a danos extremos e grandes perdas financeiras (Choudhary and Sharma, 2021).

Como foi explicitado, a necessidade de métodos de detecção de malware é urgente. Para tanto, técnicas baseadas em Machine Learning (ML) podem ser de grande valor para a segurança de sistemas. Com modelos de ML, é possível automatizar a

identificação e classificação de malwares. Assim, a proteção de usuários é providenciada ao manter o sistema seguro e ao impedir ataques cibernéticos mais rapidamente. Diversas técnicas já foram estudadas por outros pesquisadores. As que foram observadas como as mais eficientes são as seguintes: Random Forest (RF), Support Vector Machine (SVM), Naive Bayes (NB), Logistic Regression (LR), Principal Component Analysis (PCA), K-Means e K-Nearest Neighbors (KNN) (Shhadat et al., 2020; Choudhary and Sharma, 2021). Para este trabalho, foram utilizadas as técnicas Decision Tree, Random Forest e KNN.

Este trabalho está organizado da seguinte forma. Seção 2 o objetivo geral e os objetivos específicos do trabalho. A Seção 3 apresenta a metodologia empregada para atingir os objetivos descritos anteriormente. Os resultados obtidos são explicitados na Seção 4. Na Seção 5, os resultados são discutidos. Finalmente, concluímos na Seção 6.

2. Objetivo

O objetivo geral deste trabalho é o desenvolvimento de uma análise comparativa entre diferentes técnicas, baseadas em ML, para detectar malwares a partir de um determinado dataset.

Os objetivos específicos constam abaixo:

- Selecionar técnicas para serem utilizadas no contexto do trabalho;
- Comparar e analisar os resultados obtidos das técnicas utilizadas.

3. Metodologia

Para atingir os objetivos definidos na seção anterior, primeiramente foi selecionado um dataset relativo a detecção de malwares, cujo resumo se encontra na subseção 3.1. Após a definição do dataset, foi realizada uma seleção simples de *features*, que se baseou nos dados de correlação de Pearson. Esta atividade está mais detalhada na subseção 3.2. A definição dos modelos desenvolvidos compreende a subseção 3.3 e as métricas utilizadas para analisar suas performances se encontram na subseção 3.4.

3.1. Dataset

Com base nas características das observações, o conjunto de dados foi criado em uma máquina virtual, baseado em Unix / Linux, para fins de classificação de malwares. Este conjunto de dados consiste em 100.000 dados de observação e 35 atributos, sendo 50.000 classificados como *malware* e o restante como *benign* (benigno). O mesmo foi disponibilizado na plataforma Kaggle (Saravana, 2018). Abaixo está uma tabela de especificações e descrições.

Tabela 1. Tabela de especificações e descrições

Atributos	Descrição
hash	APK/ SHA256 file name
milisecond	Time
classification	Malware/Benign

state	Sinalização de tarefas não executáveis / executáveis / interrompidas
usage_counter	Contador de uso da estrutura da tarefa
prio	Mantém a prioridade dinâmica de um processo
static_prio	Prioridade estática de um processo
normal_prio	Prioridade sem levar em consideração a herança RT
policy	Política de planejamento do processo
vm_pgoff	O deslocamento da área no arquivo, em páginas
vm_truncante_count	Usado para marcar uma vma como agora tratada
task_size	Tamanho da tarefa atual
cached_hole_size	Tamanho do orifício do espaço de endereço livre
free_area_cache	Primeiro buraco no espaço de endereço
mm_users	Usuários do espaço de endereço
map_count	Número de áreas de memória
hiwater rss	Pico do tamanho do conjunto residente
total_vm	Número total de páginas
shared_vm	Número de páginas compartilhadas
exec_vm	Número de páginas executáveis
reserved_vm	Número de páginas reservadas
nr_ptes	Número de entradas da tabela de página
end_data	Endereço final do componente de código
last_interval	Último intervalo de tempo antes de thrashing
nvcsw	Número de trocas de contexto voluntárias
nivcsw	Número de mudanças de contexto voluntárias
min_flt	Falhas mínimas de página
maj_flt	Falhas máximas de página
fs_excl_counter	Contém recursos exclusivos do sistema de arquivos
lock	Bloqueio de sincronização de leitura e gravação usado para acesso ao sistema de arquivos
utime	Tempo do usuário

stime	Hora do sistema
gtime	Tempo de convidado
cgtime	Tempo cumulativo do grupo
signal_nvcsw	Usado como contador de recurso cumulativo

3.2. Seleção de Features

A estratégia utilizada para selecionar os atributos mais importantes foi calcular a correlação entre todas as variáveis do dataset. Foi criado um mapa de calor de correlação de Pearson, no qual é possível ver a relação das variáveis independentes com a variável de saída *classification*. O coeficiente de correlação tem valores entre -1 e 1. Quanto mais próximo de 0 o valor, mais fraca a correlação e quanto mais próximo de 1 ou -1 mais forte a correlação (positiva ou negativa). A Figura 1 mostra o mapa de calor gerado:

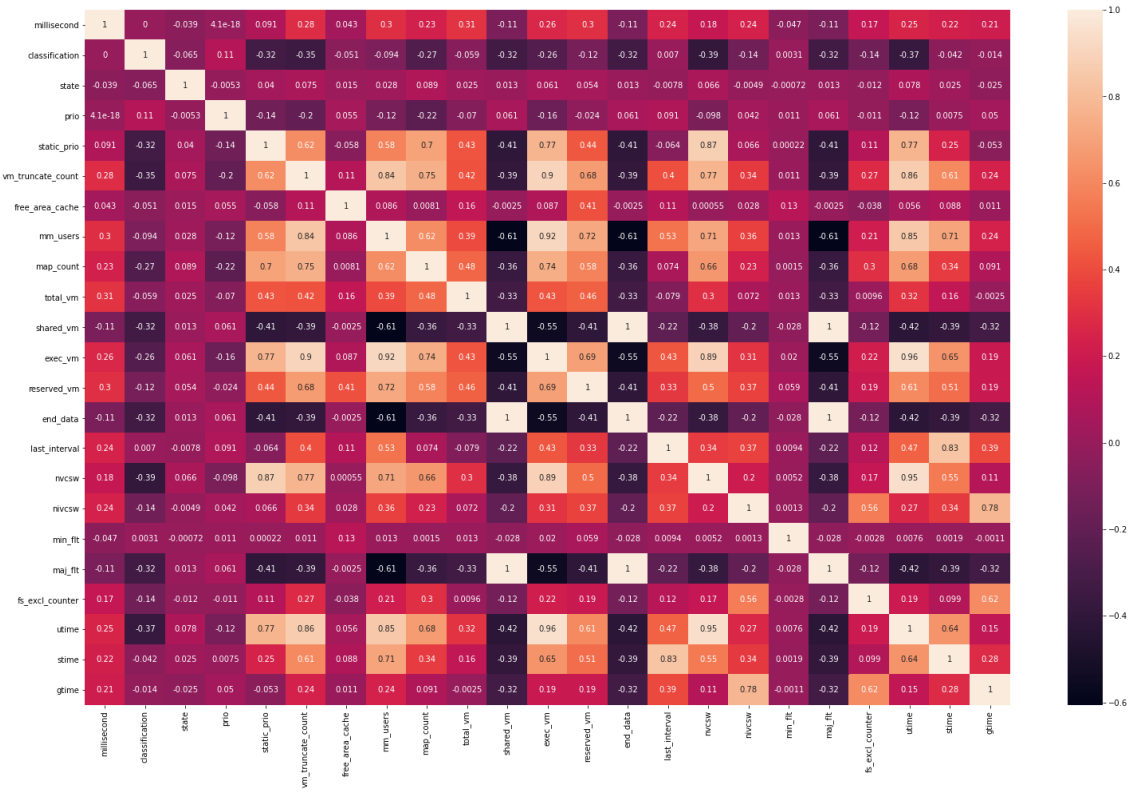

Figura 1. Mapa de Calor de Correlação

Trinta atributos apresentaram correlação negativa, portanto somente as que apresentaram correlação positiva foram selecionadas. Abaixo estão especificados os valores de correlação para cada uma dessas variáveis:

- prio: 0.110036
- last_interval: 0.006952
- min_flt: 0.003070
- millisecond: 0.000000

3.3. Técnicas de Machine Learning

Os algoritmos de ML adotados para a realização do trabalho foram Decision Tree, Random Forest e KNN. Estes três foram escolhidos devido à popularidade de emprego em trabalhos relacionados ao tema escolhido.

Em computação, árvores são estruturas de dados que são formadas por um conjunto de elementos os quais armazenam informações (nós). Toda árvore possui um ponto de partida (raiz) e ligações para outros elementos (filhos). Além disso, esses filhos podem possuir seus próprios filhos ou não (nó folha ou terminal). Desta forma, uma árvore de decisão é uma árvore que armazena regras em seus nós, tendo como decisão a ser tomada os nós-folhas (Campos, 2017).

A Decision Tree, ou árvore de decisão na língua portuguesa, é um algoritmo de aprendizagem de máquina supervisionado baseado na divisão dos dados em grupos similares. Esse algoritmo pode ser usado tanto em problemas de classificação quanto de regressão. Não requer conhecimento estatístico para sua interpretação e minimiza a necessidade de tratar os dados já que aceita tanto dados numéricos quanto categóricos. No entanto, a árvore de decisão tende a sofrer sobreajuste e pequenas alterações nos dados de treino podem produzir novas árvores (Santana, 2020).

Random Forest, ou florestas aleatórias no português, é uma forma de resolver o problema de sobreajuste das árvores de decisão. Este modelo define essencialmente uma coleção de árvores de decisão, onde cada árvore é ligeiramente diferente das outras. A ideia por trás deste modelo é que cada árvore pode fazer um bom trabalho de previsão, mas provavelmente superestimará parte dos dados. Para reduzir o sobreajuste, é calculada a média dos seus resultados, mantendo o poder preditivo das árvores (Müller and Guido, 2016; Grus, 2016).

O KNN (ou K-Vizinhos Mais Próximos) é um dos algoritmos mais simples para aprendizado supervisionado de máquina. Seu modelo requer somente o conjunto de dados para treinamento e a definição do valor k, que delimita a quantidade de pontos de dados mais próximos no conjunto para treinamento (Müller and Guido, 2016). Com estes itens, o KNN traz a predição como output. Neste trabalho, foi definido um valor ímpar para evitar empates em potencial (Mohaisen, 2015). Uma rápida verificação da acurácia de outros valores k foi realizada previamente com o intuito de avaliar o melhor número de vizinhos. Para calcular distâncias dos pontos neste trabalho, o método de similaridade utilizado foi Distância Euclidiana. Esta distância é calculada com a raiz quadrada da soma das diferenças (elevada ao quadrado) dos pontos de interesse.

O framework principal para o desenvolvimento e análise de desempenho dos modelos de ML foi o Scikit-Learn, que contém funções necessárias para modelagem e métricas para realizar a análise comparativa. O API de métricas “classification_report” gera um relatório evidenciando as principais métricas de classificação: *precision*, *recall* e *f1-score*. Além destes, o relatório exibe a acurácia do modelo, a quantidade total de registros de cada classe e médias macro e de peso (*weighted*) de cada uma das métricas.

3.4. Métricas de classificação

As métricas de classificação utilizadas neste trabalho foram acurácia, precisão (do inglês, *precision*), sensibilidade (do inglês, *recall*) e *f1-score*. Abaixo se encontra uma breve definição do que cada uma representa.

A acurácia é uma das métricas mais simples, sendo representada pelo cálculo da divisão do total de acertos (positivos e negativos). No entanto, sua aplicação é mais indicada para *datasets* balanceados - caso contrário, pode favorecer a classe que mais possui registros e, por fim, causando uma impressão errônea sobre o seu desempenho (Miguel, 2021).

A precisão é definida pela divisão entre o número de exemplos classificados corretamente como pertencentes a uma classe (verdadeiros positivos) e a soma desse número com o total de exemplos classificados erroneamente nesta classe (falsos positivos). Ou seja, ao se utilizar a precisão pretende-se mostrar, dentre todas as classificações da classe Positivo que o modelo fez, quantas estão corretas (Chen et al., 2020).

A precisão pode ser usada em problemas nos quais se considera os falsos positivos como sendo mais prejudiciais que os falsos negativos. No caso deste trabalho, ainda que o modelo acabe considerando um bom software como malware no processo, é necessário que o mesmo esteja correto, pois, se o modelo classificar um malware como um bom software pode haver um grande problema de segurança.

Sensibilidade é uma métrica que representa a proporção ou porcentagem de classificações positivas que foram identificadas corretamente, ou seja: quantidade de verdadeiros positivos dividido pela soma das quantidades de verdadeiros positivos e de falsos negativos. É mais indicada para uma situação em que os falsos negativos são considerados mais prejudiciais que os falsos positivos. Isto é, o modelo deve classificar corretamente todos os registros da classe que se deseja prever, mesmo classificando erroneamente os que não pertencem a classe em foco (Miguel, 2018; Rodrigues, 2018; Rodrigues, 2019).

F1-Score é uma métrica que faz uso dos valores de precisão e sensibilidade com o intuito de determinar um número que definirá a qualidade do modelo de ML. Sendo uma média harmônica entre precisão e sensibilidade, um F1-Score baixo é um indicativo de que ou a precisão ou a sensibilidade está baixa. Portanto, quanto maior o *score*, melhor. Assim como a acurácia, o *f1-score* é mais indicado para *datasets* com classes proporcionais e para um modelo que não emite probabilidades (Miguel, 2018; Rodrigues, 2018; Rodrigues, 2019).

Para este trabalho, chegou-se à conclusão de que não há impedimento no uso destas métricas, apesar das indicações e contra-indicações, pois, como será visto na seção seguinte, as classes presentes no *dataset* não possuem grande diferença em termos de quantidade de registros.

4. Resultados

O primeiro passo foi discretizar a variável target em 0 (*benign*) e 1 (*malware*). Dividindo o dataset em 70% de observações (70.000) para aprendizado e 30% para teste (30.000) e aplicando os mesmos nos modelos KNN, Decision Tree e Random Forest, foram obtidos os relatórios exibidos pela Figura 2, Figura 3 e Figura 4. Como pode ser observado nos primeiros dois valores, da coluna “support”, de qualquer uma das figuras, temos um conjunto desbalanceado de dados para teste. A quantidade de malwares

contabiliza 14.950 observações, enquanto os software benignos totalizam 15.050. Há uma diferença de 100 observações entre as classes.

4.1. KNN

KNN:

	precision	recall	f1-score	support
benign	0.99180	0.98870	0.99025	15050
malware	0.98866	0.99177	0.99022	14950
accuracy			0.99023	30000
macro avg	0.99023	0.99024	0.99023	30000
weighted avg	0.99024	0.99023	0.99023	30000

Figura 2. Métricas do modelo KNN

O algoritmo KNN requer a definição de da quantidade de vizinhos mais próximos , a fim de realizar a predição. O valor k = 5 foi escolhido após uma rápida verificação da acurácia do modelo quando este recebe k ímpar (prevenção contra possíveis empates). De acordo com a Figura 2, a acurácia do KNN corresponde a 99.023%. Também pode ser observado que a sensibilidade do KNN é maior para a identificação de malwares, enquanto, para a precisão e f1-score, a taxa de acerto é maior para os softwares benignos (*benign*). No entanto, mesmo dando peso ao desbalanceamento ou não, a métrica calculada é a mesma para o f1-score. Porém, esta afirmação não pode ser aplicada à sensibilidade e precisão.

4.2. Decision Tree

Decision Tree:

	precision	recall	f1-score	support
benign	0.99615	0.99674	0.99645	15050
malware	0.99672	0.99612	0.99642	14950
accuracy			0.99643	30000
macro avg	0.99643	0.99643	0.99643	30000
weighted avg	0.99643	0.99643	0.99643	30000

Figura 3. Métricas do modelo Decision Tree

O algoritmo Decision Tree apresentou uma acurácia levemente superior ao KNN (99.643%). A métrica sensibilidade foi maior para detectar softwares benignos. Já a precisão e o f1-score foram maiores para descobrir malwares. Seja considerando ou não a diferença entre a quantidade de observações das classes, os valores das métricas são os mesmos.

4.3. Random Forest

Random Forest:

	precision	recall	f1-score	support
benign	0.99728	0.99701	0.99714	15050
malware	0.99699	0.99726	0.99712	14950
accuracy			0.99713	30000
macro avg	0.99713	0.99713	0.99713	30000
weighted avg	0.99713	0.99713	0.99713	30000

Figura 4. Métricas do modelo Random Forest

Dentre os três modelos, o Random Forest alcançou a melhor acurácia (99.713%). Para as métricas de precisão e f1-score a classe benign obteve um desempenho mais alto que a malware. E o recall foi o contrário, a classe malware resultou em valores melhores.

5. Discussão

Em termos de acurácia, o Random Forest apresentou melhor desempenho dentre os modelos utilizados. Isso pode ser explicado pelo fato do Random Forest combinar o resultado de diversas árvores de decisão para realizar predições mais assertivas.

Random Forest e KNN, para as métricas apresentadas, têm o mesmo comportamento, em termos de qual classe foi melhor para classificar. Para a classe *benign,* a métrica f1-score obteve resultados mais satisfatórios em todos os modelos.

O *macro* e *weighted average* apresentaram comportamentos iguais, todas as métricas receberam o mesmo valor em seus respectivos modelos: Decision Tree com 99.643% e Random Forest com 99.713%.

6. Considerações Finais

Com a finalidade de detectar malwares, foi realizada uma análise comparativa entre métricas de classificação, aplicada sobre as técnicas KNN, Decision Tree e Random Forest. O dataset utilizado contém 35 atributos, sendo apenas uma o target: (*classification)*. Este atributo possui dois valores possíveis, *benign* e *malware*, que foi discretizado em 0 e 1, respectivamente.

Com o relatório disponibilizado pelo framework Scikit-Learn, foi observado que o modelo Random Forest teve o melhor desempenho em todas as métricas, se comparado aos demais modelos. Também foi observado similaridade de comportamentos entre algumas métricas dos modelos, como também, em seus parâmetros *macro* e *weighted average*.

Referências Bibliográfica

Campos, R. (2017) “Árvores de Decisão”. Medium, https://medium.com/machine-learning-beyond-deep-learning/%C3%A1rvores-de-decis%C3%A3o-3f52f6420b69, 09 Maio 2021.

Chen, Dehua et al. (2020). “Métricas de Avaliação em Machine Learning: Classificação”. Medium, https://medium.com/kunumi/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-em-machine-learning-classifica%C3%A7%C3%A3o-49340dcdb198, 09 Maio 2021.

Choudhary, S. and Sharma, A. (2021). Data Science Approach for Malware Detection. In: *Journal of Physics: Conference Series* 1804 012196.

Grus, J. (2016), Data Science do Zero: Primeiras Regras com o Python, Alta Books, 1st Edition.

Miguel, T. (2021) “Métricas de erro usadas na ciência de dados”. Aprenda Data Science, https://aprenderdatascience.com/metricas-de-erro-usadas-na-ciencia-de-dados, 12 Maio 2021.

Mohaisen, D., Alrawi, O. and Mohaisen, M. (2015). AMAL: High-Fidelity, Behavior-Based Automated Malware Analysis and Classification. In *Computers & Security*, volume 52, pages 251-266. Publishing Press.

Müller, A. C. and Guido, S. (2016), Introduction to Machine Learning with Python, O’Reilly Media, 1st Edition.

Rodrigues, V. (2018) “Entenda o que é AUC e ROC nos modelos de Machine Learning”. Medium, https://medium.com/bio-data-blog/entenda-o-que-%C3%A9-auc-e-roc-nos-modelos-de-machine-learning-8191fb4df772, 12 Maio 2021.

Rodrigues, V. (2019) “Métricas de Avaliação: acurácia, precisão, recall… quais as diferenças?”. Medium, https://medium.com/@vitorborbarodrigues/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c, 12 Maio 2021.

Santana, F. (2020) “Árvores de Decisão (Projeto passo a passo)”. Minerando Dados, https://minerandodados.com.br/arvores-de-decisao-conceitos-e-aplicacoes/, 09 Maio 2021.

Saravana, N. (2018) “Malware Detection”. Kaggle, https://www.kaggle.com/nsaravana/malware-detection, 24 Abril 2021.

Sethi, K., Chaudhary, S., Tripathy, Bata and Bera, P. (2017). A novel malware analysis for malware detection and classification using machine learning algorithms. In: Proceedings of the 10th International Conference on Security of Information and Networks, pages 107-113.

Shhadat, I., Bataineh, B., Hayajneh, A. and Al-Sharif, Z. (2020). The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware. In *Procedia Computer Science*, volume 170, pages 917-922. Publishing Press.