# Análise Preditiva e Interpretação da Classificação de Malwares em Sistemas Android Usando Aprendizado de Máquina

**Geovani da S. do Amaral**[1], **Heitor S. R. S. Pinto**[1], **Caio C. Moreira**[1]

[1]Faculdade de Engenharia de Computação
Universidade Federal do Pará (UFPA) – Tucuruí-PA – Brasil

{geovani.amaral, heitor.pinto}@tucurui.ufpa.br, caiomoreira@ufpa.br

***Abstract.*** *This paper presents a predictive analysis for detecting malwares in Android devices using machine learning and to interpret the results with explainable methods. After preprocessing, the dataset was reduced to 34,076 samples and 179 features of system calls and permissions. Among the 13 classifiers evaluated, eXtreme Gradient Boosting (XGBoost) proved to be the most efficient, with accuracy, precision, recall, and F1-Score metrics of approximately 94% and a training time of 1.48s. The SHapley Additive exPlanations (SHAP) method was used to explain the model's decisions, revealing system calls and sensitive permissions such as READ_PHONE_STATE, SYSTEM_ALERT_WINDOW, SEND_SMS, ACCESS_WIFI_STATE, getpriority, and getrlimit strongly associated with malwares.*

***Resumo.*** *Este trabalho apresenta uma análise preditiva para detecção de malwares em dispositivos Android usando Aprendizado de Máquina e a interpretação dos resultados com métodos de explicabilidade. Após o pré-processamento, o conjunto de dados foi reduzido para 34.076 amostras e 179 características de chamadas de sistema e permissões. Entre 13 classificadores avaliados, o eXtreme Gradient Boosting (XGBoost) mostrou-se o mais eficiente, com métricas de acurácia, precisão, recall e F1-Score de aproximadamente 94%, e Tempo de Treinamento de 1,48s. O método SHapley Additive exPlanations (SHAP) foi utilizado para explicar as decisões do modelo, revelando chamadas de sistema e permissões sensíveis, como READ_PHONE_STATE,SYSTEM_ALERT_WINDOW, SEND_SMS, ACCESS_WIFI_STATE, getpriority e getrlimit, fortemente associados a malwares.*

## 1. Introdução

Devido à sua popularidade, o Android se tornou o principal alvo de ataques e fraudes. Um estudo recente da Kaspersky revelou que, em 2023, houve mais de 600 milhões de *downloads* de aplicativos mal-intencionados (*malwares*) [Kaspersky 2023].

Para enfrentar o desafio dos *malwares*, diversas estratégias de detecção têm sido desenvolvidas. Muitas delas se concentram na análise de características dos aplicativos, como chamadas de sistema e permissões, para identificar ameaças potenciais. Essas características ajudam a retratar o comportamento dos aplicativos e a detectar padrões de *malwares* [Wang et al. 2019].

As soluções geralmente empregam técnicas de Aprendizado de Máquina (AM) e detecção de anomalias para reconhecer comportamentos atípicos em relação ao uso normal do dispositivo [Kouliaridis and Kambourakis 2021]. No entanto, muitos estudos não analisam a interpretação das características mais impactantes nas decisões dos modelos, o que pode levar a uma falta de identificação de inconsistências ou vieses [Liu et al. 2020].

Portanto, este trabalho tem como objetivo analisar e interpretar as características de chamadas de sistema e permissões em aplicativos Android para entender o que distingue *malwares* de aplicativos benignos (*goodwares*) utilizando técnicas de AM.

## 2. Trabalhos relacionados

Diversos estudos têm explorado a detecção de malwares em dispositivos Android, focando em diferentes abordagens de análise. No trabalho [Akbar et al. 2022], foi desenvolvido um sistema chamado PerDRaML, que detecta aplicativos android maliciosos analisando permissões suspeitas. Usando AM, o sistema classifica aplicativos como *goodwares* ou *malwares*.

Outro estudo, [Abuthawabeh and Mahmoud 2019], propôs um modelo para detectar e classificar *malware* utilizando dados de tráfego de rede, extraindo características com a ferramenta Peer Shark e aplicando técnicas de AM, como Random Forest (RF) e Light Gradient Boosting Machine (LightGBM).

Por fim, em [Kouliaridis et al. 2020], foi desenvolvido o Androtomist, uma ferramenta que combina análise estática e dinâmica para avaliar o comportamento de aplicativos Android, mostrando que essa abordagem híbrida melhora significativamente a detecção de *malwares*.

## 3. Metodologia

O conjunto de dados utilizado, conforme descrito em [Guerra-Manzanares et al. 2021], é de acesso público contendo 78.137 amostras, das quais 41.382 são *malwares*, distribuídos em 240 famílias, e 36.755 são *goodwares*. Esse conjunto de dados possui 383 características binárias, sendo 217 do tipo chamadas de sistema e 166 do tipo permissões. No entanto, foi identificado que o conjunto de dados continha registros nulos e duplicados, além de características com baixa informação. Portanto, foi realizado um pré-processamento para reduzir o viés nos algoritmos de AM e diminuir a dimensionalidade.

Aplicou-se a técnica de Variance Threshold (VT), um filtro não supervisionado que remove variáveis com baixa variância, simplificando o modelo e melhorando sua eficiência computacional [Solorio-Fernández et al. 2020]. Com a aplicação do método VT, utilizando um limite de 99,9% — determinado experimentalmente como o melhor valor para manter a menor quantidade de features e obter os melhores resultados de classificação —, 204 características foram eliminadas, resultando em uma redução de 53,1% nas variáveis. Além disso foram removidos 14 registros compostos apenas por valores 0 e 44.047 registros duplicados, mantendo-se a primeira ocorrência de cada, o que reduziu o conjunto de dados em aproximadamente 56,4%.

Após esse processo, o conjunto de dados resultante contém 34.076 amostras, sendo 16.612 *malwares* e 17.464 *goodwares*, e 179 características, das quais 73 são chamadas de sistema e 106 são permissões.

Para avaliar a performance, foram utilizados 13 classificadores com funcionalidades distintas: Adaptive Boosting (AdaBoost), Decision Tree (DT), Extra-Trees (EXT), eXtreme Gradient Boosting (XGBoost), Gradient Boosting (GB), K-Nearest Neighbors (KNN), LightGBM, Logistic Regression (LR), Linear Discriminant Analysis (LDA), Naive Bayes (NB), RF, Ridge Classifier (RC) e Support Vector Machine (SVM), os modelos foram avaliados utilizando Validação Cruzada com 10 *folds* ($K = 10$) e medidos em Acurácia, Precisão, *Recall*, *F1-Score*, e Tempo de Treinamento (TT). O melhor classificador será utilizado na interpretação dos resultados.

Métodos de explicabilidade para AM são essenciais para entender as decisões dos modelos, identificar características-chave e detectar vieses. Para avaliar o impacto das características e aumentar a confiança no algoritmo, utilizou-se o método SHapley Additive exPlanations (SHAP), que, baseado nos valores de Shapley da teoria dos jogos, fornece explicações detalhadas sobre quais chamadas de sistema e permissões foram mais relevantes para classificar um aplicativo como malicioso ou benigno [Mosca et al. 2022].

Por fim, devido ao grande volume de amostras, o conjunto foi dividido em 95% para treino e 5% para teste, a fim de facilitar a visualização e interpretação dos gráficos.

## 4. Resultados

A Tabela 1 apresenta os resultados dos 13 classificadores utilizados. Observa-se que os algoritmos baseados em árvores de decisão em conjunto (*ensemble*), como EXT, LightGBM, RF e XGBoost, obtiveram os melhores desempenhos gerais de classificação, sem diferença significativa entre eles, alcançando acurácia, *recall*, precisão e *F1-Score* de aproximadamente 94%.

No entanto, o XGBoost destacou-se como uma alternativa mais eficiente, apresentando um TT de 1,48 segundos. Assim, o XGBoost foi selecionado como o modelo de classificação para a interpretação dos resultados.

**Tabela 1. Resultados dos classificadores utilizando Validação Cruzada ($K = 10$).**

| Modelo | Acurácia(%) | *Recall*(%) | Precisão(%) | *F1-Score*(%) | TT (s) |
|---|---|---|---|---|---|
| EXT | 94,73 | 93,98 | 95,15 | 94,56 | 6,77 |
| RF | 94,57 | 93,64 | 95,15 | 94,39 | 4,73 |
| XGBoost | 94,34 | 93,41 | 94,91 | 94,15 | 1,48 |
| LightGBM | 94,16 | 93,18 | 94,76 | 93,96 | 3,11 |
| KNN | 92,76 | 90,90 | 94,06 | 92,45 | 1,91 |
| GB | 92,28 | 91,43 | 92,64 | 92,03 | 7,14 |
| LR | 92,08 | 90,46 | 93,10 | 91,76 | 1,45 |
| SVM | 91,40 | 90,97 | 91,56 | 91,13 | 5,72 |
| LDA | 91,35 | 89,43 | 92,58 | 90,98 | 7,36 |
| RC | 91,34 | 89,41 | 92,58 | 90,97 | 2,22 |
| DT | 91,05 | 91,66 | 90,14 | 90,89 | 8,32 |
| AdaBoost | 91,05 | 89,62 | 91,82 | 90,70 | 2,23 |
| NB | 82,97 | 83,55 | 81,93 | 82,71 | 2,23 |

A Figura 1 apresenta um gráfico SHAP do tipo *beeswarm*, mostrando as 30 principais características organizadas em ordem decrescente de impacto na saída do modelo. A predição do classificador atribui o valor 1 para *malware* e 0 para *goodware*. Os valores à esquerda do eixo vertical influenciam a predição em direção ao *goodware*, enquanto os

à direita a influenciam em direção ao *malware*. Pontos vermelhos indicam valores altos (1) das características binárias, e pontos azuis indicam valores baixos (0). Cada ponto no gráfico corresponde a uma amostra do conjunto de testes, sendo empilhados verticalmente em regiões de alta densidade.

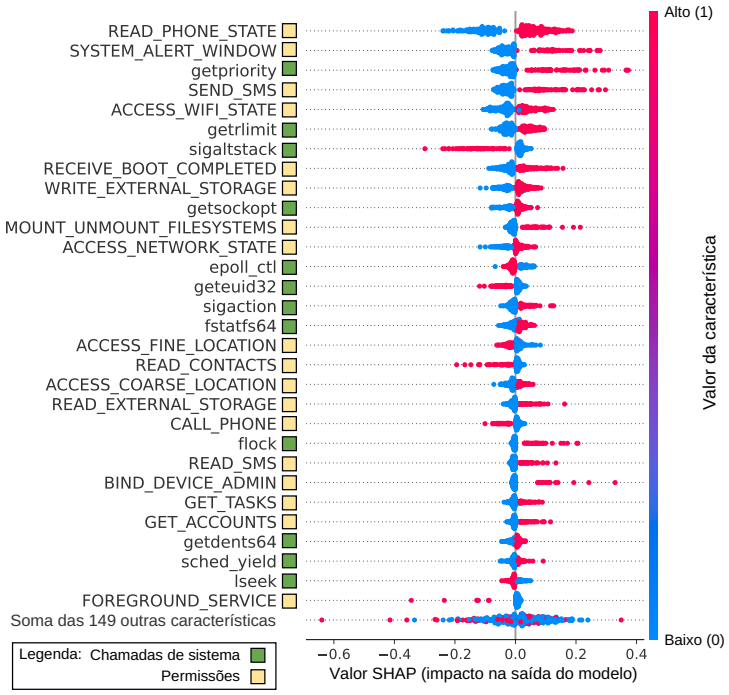


**Figura 1. Gráfico SHAP do tipo *beeswarm* das 30 principais características, em ordem decrescente de acordo com seu impacto na saída do modelo.**

Nota-se que as seis características mais impactantes na saída do modelo — as permissões *READ_PHONE_STATE*, *SYSTEM_ALERT_WINDOW*, *SEND_SMS* e *ACCESS_WIFI_STATE*, e as chamadas de sistema *getpriority* e *getrlimit* —, ao apresentarem valores altos (1), indicam que sua utilização em um aplicativo Android está fortemente associada à classificação como *malware*, enquanto a não utilização dessas características está associada à classificação oposta.

Por outro lado, a utilização de características como as chamadas de sistema *sigaltstack*, *epoll_ctl*, *geteuid32* e *lseek* e as permissões *ACCESS_FINE_LOCATION*, *READ_CONTACTS* e *CALL_PHONE*, está associada aos *goodwares*, indicando que os *malwares* normalmente não utilizam essas características.

A Figura 2 apresenta um gráfico de decisão SHAP com as 30 características mais impactantes na saída do modelo. Cada linha representa uma amostra e sua trajetória de decisão, acumulando os valores SHAP desde a base até a saída do modelo. Linhas vermelhas indicam predições de *malware* (acima de $0,5$), enquanto as azuis predizem *goodware* (abaixo de $0,5$). Linhas tracejadas representam classificações incorretas.

No gráfico de decisão SHAP, dois fluxos principais, representados pelas cores azul e vermelha, mostram como as características impactam a predição. Na parte inferior, os fluxos estão mais mesclados, indicando sobreposição nas contribuições para ambas as classes em características de menor impacto. À medida que se deslocam para a parte

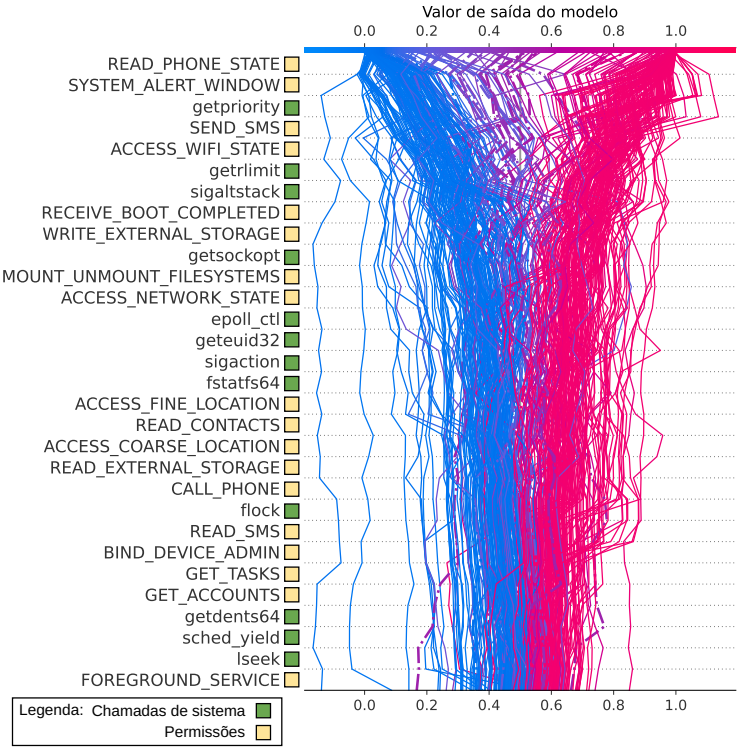


**Figura 2. Gráfico SHAP do tipo decisão das 30 principais características, em ordem decrescente de acordo com seu impacto na saída do modelo.**

superior do gráfico, a separação entre os fluxos se torna mais clara, especialmente nas dez características mais impactantes. Essa separação destaca a influência decisiva dessas características na distinção entre as classes e mostra que o modelo confia fortemente nelas para identificar padrões específicos associados a comportamentos maliciosos ou benignos.

Analisando as características mais impactantes associadas aos *malwares*, pode-se observar que a permissão *READ_PHONE_STATE* fornece acesso a informações sensíveis, como número de telefone e IMEI, que podem ser usadas para clonagem e roubo de dados. A permissão *SYSTEM_ALERT_WINDOW* permite a sobreposição de tela, facilitando ataques como *phishing* e *tapjacking*. Já a chamada de sistema *getpriority* pode ser usada com *setpriority* para manter o *malware* ativo, tornando-o mais difícil de ser interrompido.

A permissão *SEND_SMS* pode ser explorada para enviar *spam* ou mensagens fraudulentas, resultando em cobranças indesejadas. A permissão *ACCESS_WIFI_STATE* fornece informações sobre redes *Wi-Fi*, úteis para geolocalização e rastreamento. A chamada de sistema *getrlimit*, em conjunto com *setrlimit*, permite ajustar limites de recursos para evitar interrupções do processo e dificultar a detecção.

A permissão *RECEIVE_BOOT_COMPLETED* permite que o *malware* se reinicie automaticamente após o *boot* do dispositivo, garantindo sua persistência. Finalmente, *WRITE_EXTERNAL_STORAGE* concede acesso ao armazenamento externo, onde o *malware* pode armazenar dados roubados e ocultar suas atividades. Cada uma dessas chamadas de sistema e permissões pode ser explorada para realizar atividades prejudiciais e comprometer a segurança e privacidade do usuário.

## 5. Considerações Finais

Com base na análise preditiva e na interpretação das características mais impactantes de chamadas de sistema e permissões de aplicativos Android usando AM, conclui-se que o modelo desenvolvido é eficaz na distinção entre *malwares* e *goodwares*. O classificador XGBoost mostrou-se particularmente eficiente, e o método SHAP possibilitou uma compreensão aprofundada das características que mais influenciam as predições. A interpretação dos resultados revelou como os *malwares* exploram chamadas de sistema e *permissões* específicas para criar vulnerabilidades e realizar atividades que comprometam a segurança e a privacidade do usuário.

Como próximos passos, recomenda-se investigar outros conjuntos de dados e aplicar algoritmos de regras de associação para analisar os padrões de *malware* em aplicativos Android sob diferentes perspectivas.

## Referências


Abuthawabeh, M. K. A. and Mahmoud, K. W. (2019). Android malware detection and categorization based on conversation-level network traffic features. In *2019 International Arab Conference on Information Technology (ACIT)*, volume 6, page 42–47. IEEE.

Akbar, F., Hussain, M., Mumtaz, R., Riaz, Q., Wahab, A. W. A., and Jung, K.-H. (2022). Permissions-based detection of android malware using machine learning. *Symmetry*, 14(4):718.

Guerra-Manzanares, A., Bahsi, H., and Nõmm, S. (2021). Kronodroid: Time-based hybrid-featured dataset for effective android malware detection and characterization. *Computers & Security*, 110:102399.

Kaspersky (2023). Malwares do google play atingem mais de 600 milhões de downloads em 2023. Disponível em: `https://www.kaspersky.com.br/blog/malware-in-google-play-2023/21985/`. Acesso em: 12 de março de 2024.

Kouliaridis, V. and Kambourakis, G. (2021). A comprehensive survey on machine learning techniques for android malware detection. *Information*, 12(5).

Kouliaridis, V., Kambourakis, G., Geneiatakis, D., and Potha, N. (2020). Two anatomists are better than one—dual-level android malware detection. *Symmetry*, 12(7):1128.

Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., and Liu, H. (2020). A review of android malware detection approaches based on machine learning. *IEEE Access*, 8:124579–124607.

Mosca, E., Szigeti, F., Tragianni, S., Gallagher, D., and Groh, G. (2022). SHAP-based explanation methods: A review for NLP interpretability. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4593–4603, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Solorio-Fernández, S., Carrasco-Ochoa, J. A., and Martínez-Trinidad, J. F. (2020). A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53(2):907–948.

Wang, W., Zhao, M., Gao, Z., Xu, G., Xian, H., Li, Y., and Zhang, X. (2019). Constructing features for detecting android malicious applications: Issues, taxonomy and directions. *IEEE Access*, 7:67602–67631.