Aplicação de Algoritmos de Classificação para Detecção de Malware Utilizando Dados Balanceados com SMOTE

Luis Eduardo Silva Brito[1], **Victor Macêdo Carvalho**[2], **Francisco Aparicio N Coelho**[3]

[1]Departamento de Sistemas de Informação,
Universidade Federal do Piauí - Picos, Brasil

{duardos36, faparicionc}@gmail.com, victmacc@ufpi.edu.br

Resumo. *Este artigo explora a aplicação de algoritmos de classificação para a detecção de malware, utilizando dados balanceados com SMOTE. A detecção de malware é desafiadora devido à sofisticação das ameaças e à alta dimensionalidade dos dados. Foram avaliados os modelos KNN, Random Forest, Regressão Logística, Naive Bayes, SVM e MLP. Utilizou-se a base TUANDROMD, composta por características extraídas de binários de aplicativos Android. A avaliação foi realizada com validação cruzada K-Fold. Os resultados mostraram que o Random Forest obteve o melhor desempenho em diversas métricas. O estudo destaca a importância do balanceamento de dados e do aprendizado supervisionado em relação a causa do software malicioso - malware.*

1. Introdução

Nos últimos anos, a crescente quantidade de dispositivos móveis e o aumento da utilização de sistemas baseados em Android têm gerado uma preocupação crescente com a segurança cibernética, especialmente no que se refere à detecção de malware. O uso indevido desses dispositivos pode resultar em sérios danos aos usuários, com a disseminação de vírus e softwares maliciosos que afetam tanto informações pessoais quanto corporativas. De acordo com estudos recentes, a detecção de malware tem sido uma área desafiadora devido à sofisticação dos métodos de ocultação empregados pelos malwares modernos [Hasan et al. 2025].

A identificação de malware é um processo essencial para a proteção de sistemas, e diversas técnicas têm sido investigadas para melhorar a precisão dessa detecção. Entre as abordagens mais eficazes, destacam-se os métodos de aprendizado de máquina, que têm se mostrado promissores na classificação de softwares maliciosos, com base em atributos extraídos dos próprios binários [Baburaj et al. 2024]. O uso dessas técnicas permite a criação de modelos capazes de aprender padrões complexos, utilizando um grande volume de dados para prever a natureza do software. No entanto, um dos maiores desafios é lidar com a alta dimensionalidade dos dados, o que pode afetar a performance dos modelos [Chandran et al. 2025].

Apesar dos avanços na detecção de malware, a complexidade das ameaças cibernéticas continua a ser um grande obstáculo. Um dos principais desafios enfrentados pelos pesquisadores é a constante evolução das técnicas de obfuscação usadas pelos malwares, o que dificulta a identificação por métodos tradicionais [Chandran et al. 2025]. Além disso, a implementação de modelos de aprendizado de máquina requer um grande

volume de dados rotulados, o que nem sempre está disponível, especialmente em contextos dinâmicos, como o de malware. Esse problema é agravado pelo fato de que malwares podem se adaptar rapidamente a novos métodos de detecção, exigindo que os modelos sejam continuamente atualizados para manter sua eficácia [Ambekar et al. 2024].

Os algoritmos de Aprendizado de Máquina desempenham um papel essencial ao permitir que modelos automatizados identifiquem padrões e façam previsões precisas com base em grandes volumes de dados. Técnicas como K-Nearest Neighbors (KNN), Random Forest e MultiLayer Perceptron (MLP) são utilizadas devido à sua capacidade de lidar com dados complexos e de alta dimensionalidade. Esses algoritmos se destacam em diversas aplicações, como a detecção de padrões em imagens e a classificação de dados, sendo particularmente eficazes na análise de dados em áreas como a cibersegurança, onde são usados para identificar malware por meio de atributos extraídos de binários. A habilidade desses algoritmos de aprender com dados e realizar classificações com alta precisão os torna fundamentais para resolver problemas complexos em uma ampla gama de setores [Köhler et al. 2024].

2. Metodologia

Nesta seção, será apresentada a metodologia abordada para a realização deste trabalho, na qual foi utilizada uma abordagem supervisionada com a implementação de diversos algoritmos de aprendizado de máquina, incluindo o K-Nearest Neighbors (KNN), Random Forest, Logistic Regression, Naive Bayes, SVM e MLP. A análise foi feita no conjunto de dados TUANDROMD, na qual os modelos foram avaliados por meio de validação cruzada K-Fold (com 5 divisões) e o balanceamento das classes foi realizado utilizando a técnica SMOTE, com o objetivo de melhorar o desempenho na classificação binária entre malware e software legítimo.

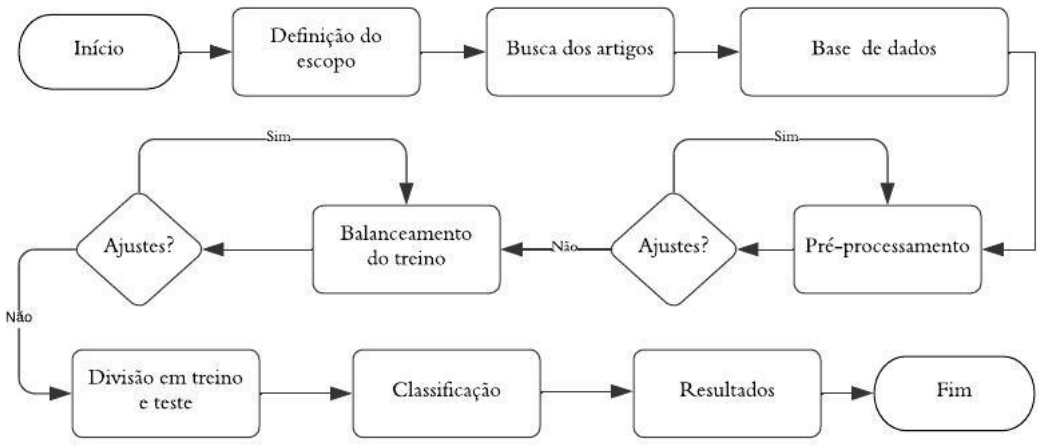

Figure 1. Desempenho Comparativo dos Modelos de Classificação (K-Fold + SMOTE)

2.1. Base de Dados

Neste estudo, utilizou-se a base de dados TUANDROMD (Tunisian Android Malware Dataset), para a análise e classificação de malwares em aplicativos Android desenvolvido

para dar suporte à pesquisa em detecção de malware e avaliar a eficácia de métodos de classificação aplicados à segurança cibernética. Neste sentido, este dataset é composto por 4.465 instâncias e 241 atributos, todos de tipo inteiro. O atributo alvo, denominado Label, representa a categoria do software analisado, sendo 1.0 para malware e 0.0 para software legítimo.

As instâncias que estão presentes no TUANDROMD representam as características extraídas de binários dos aplicativos Android, incluindo permissões, chamadas de API e propriedades comportamentais. Diante disso, essas características foram extraídas para simular cenários reais voltados para à análise de segurança, tornando assim o conjunto de dados extremamente representativo para tarefas de classificação supervisionada na área de Ciência da Computação, mais especialmente para a análise de malware e segurança digital. A Tabela 1 a seguir contém todas as características da base de dados.

Table 1. Resumo da estrutura do conjunto de dados TUANDROMD

Propriedade	**Descrição**
Tipo de objeto	DataFrame
Total de instâncias	4.464
Total de atributos	242
Atributos de entrada	241 (características binárias inteiras)
Atributo alvo	`Label` (0.0: software bom, 1.0: malware)
Tipos de dados	float64 (todos os atributos)
Uso de memória	8.3 MB
Índice	0 a 4464
Formato dos dados	Tabular
Área de aplicação	Ciência da Computação (Segurança Digital)
Tipo de tarefa	Classificação supervisionada

2.2. Pré-processamento

Nesta etapa, foi realizado o carregamento do dataset utilizando a biblioteca pandas. Em seguida, foi realizada a limpeza dos dados por meio da exclusão de todas as instâncias que apresentavam valores ausentes no atributo-alvo (Label). Após isso, para lidar com possíveis valores ausentes nas demais colunas, foi aplicada a técnica SimpleImputer, com a estratégia "constant" e valor de preenchimento zero (fill_value=0). Essa abordagem foi escolhida levando em consideração que os atributos possuem natureza binária (0 ou 1), e a substituição por zero preserva a coerência dos dados e evita enviesamento no processo de classificação. Dessa forma, o resultado obtido foi um conjunto tabular completo e consistente. Além disso, foi utilizada a técnica de *SMOTE* (Synthetic Minority Over-sampling Technique) para balancear os dados, garantindo que o modelo não fosse enviesado para a classe majoritária.

2.3. Divisão em Treino e Teste

Nesta etapa, após os dados estarem devidamente imputados e balanceados, foi realizada a separação entre os subconjuntos de treinamento e teste. Entretanto, ao invés de realizar uma simples divisão fixa, foi adotada a validação cruzada utilizando a técnica de K-Fold. A função StratifiedKFold da biblioteca *sklearn.model_selection* foi utilizada para dividir

os dados em 5 partes (*n_splits=5*), de forma que a proporção entre as classes (0.0 para software legítimo e 1.0 para malware) fosse mantida em todas as divisões. Durante o processo de validação, o modelo foi treinado em 4 das partes e avaliado na parte restante, repetindo esse processo 5 vezes, de modo que todas as instâncias dos dados fossem usadas para treino e teste. Esse procedimento permite uma avaliação mais robusta e generalizada do modelo, já que ele é testado em diferentes subconjuntos dos dados a cada iteração.

2.4. Balanceamento do Conjunto de Treino

A distribuição original da variável Label é extremamente desbalanceada. Tendo em vista isso, há uma predominância da classe 1.0 (malware) em relação a outra classe. Diante disso, para amenizar esse problema, foi então aplicada a técnica de oversampling sintético *SMOTE (Synthetic Minority Over-sampling Technique)* apenas sobre o conjunto de treino, utilizando a implementação da biblioteca imblearn. O SMOTE atua gerando instâncias sintéticas da classe minoritária (0.0) com base na interpolação entre os exemplos reais, sem replicação direta, o que aumenta a generalização do modelo. A aplicação foi restrita ao treino para evitar vazamento de dados e garantir uma avaliação justa e realista.

2.5. Algoritmos de Classificação

A etapa de classificação foi realizada utilizando algoritmos de aprendizado supervisionado, com o objetivo de avaliar o desempenho de cada modelo em relação às métricas de *Acurácia*, *Precisão*, *Recall* e *F1-Score*. Os algoritmos de classificação selecionados foram:

- **KNN (K-Nearest Neighbors)**: Este algoritmo funciona identificando os K vizinhos mais próximos de uma nova amostra, com base na distância Euclidiana. Neste sentido, para cada instância do conjunto de teste, as distâncias foram calculadas em relação a todas as instâncias do conjunto de treino balanceado, e a predição foi feita por meio da votação majoritária entre as classes dos vizinhos mais próximos. Neste trabalho, foi definido $k = 5$, ou seja, os cinco vizinhos mais próximos foram considerados para determinar a classe da amostra.
- **Random Forest**: Este modelo é baseado em um conjunto de árvores de decisão, na qual cada árvore é treinada com uma amostra aleatória do conjunto de dados. O modelo realiza a classificação por meio da votação das árvores, sendo a classe escolhida aquela que for mais votada. O *Random Forest* é eficaz em lidar com dados de alta dimensionalidade e é menos suscetível ao *overfitting* em comparação com modelos de árvores de decisão individuais.
- **Regressão Logística**: Utilizada para modelagem preditiva binária, a *Regressão Logística* estima a probabilidade de uma amostra pertencer a uma classe específica. Dessa forma, este modelo utiliza uma função logística para modelar a relação entre as variáveis independentes e a probabilidade da classe, oferecendo uma abordagem simples e eficiente para problemas de classificação binária.
- **Naive Bayes**: Este é um modelo probabilístico baseado no teorema de Bayes, que assume que as características são independentes entre si. Entretanto, apesar da simplicidade de seu modelo, o *Naive Bayes* pode ser muito eficaz, especialmente quando as suposições de independência são razoáveis, o que o torna popular para classificações de texto e problemas com muitas variáveis.

- **SVM (Support Vector Machine)**: O *SVM* é um modelo de classificação supervisionada que procura encontrar o melhor hiperplano que separa as classes no espaço de características. Com isso, esse modelo é eficaz, principalmente quando a separação entre as classes é clara, e é altamente eficaz em problemas de classificação de alta dimensão.
- **MLP (Multi-layer Perceptron)**: O *MLP* é uma rede neural com múltiplas camadas de neurônios que aprende padrões não lineares no conjunto de dados. Ele é composto por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. O modelo é treinado utilizando o algoritmo de retropropagação, o que permite ajustar os pesos das conexões de forma iterativa.

Cada um desses algoritmos foi avaliado utilizando a técnica de *K-Fold Cross-Validation*, com 5 divisões (*folds*), para garantir que o desempenho dos modelos fosse robusto e não dependente de uma única divisão do conjunto de dados. Além disso, o *SMOTE* (Synthetic Minority Over-sampling Technique) foi aplicado ao conjunto de treino para balanceamento das classes, gerando amostras sintéticas da classe minoritária e garantindo que os modelos fossem treinados em dados balanceados.

2.6. Avaliação do Modelo

A avaliação dos modelos foi conduzida sobre a base de teste original e desbalanceada. A Tabela 2 apresenta as métricas utilizadas para avaliar o desempenho dos modelos de classificação adotados neste trabalho. Essas métricas são amplamente utilizadas na literatura de aprendizado de máquina e fornecem uma análise detalhada da capacidade do modelo em identificar corretamente as classes.

Table 2. Métricas para avaliação de classificação

Métrica	Descrição
Acurácia (*Accuracy*)	Proporção de previsões corretas em relação ao total
Precisão (*Precision*)	Proporção de verdadeiros positivos entre os positivos previstos
Recall	Proporção de verdadeiros positivos entre os positivos reais
F1-Score (*F1*)	Média harmônica entre precisão e *recall*, balanceando os dois

3. Resultados

Nesta seção, serão abordados os resultados obtidos a partir da avaliação dos modelos de classificação, utilizando métricas de desempenho como Acurácia, Precisão, Recall e F1-Score. A Tabela 3 apresenta os resultados de desempenho dos modelos de classificação avaliados, com as médias e desvios padrão correspondentes. Para cada métrica, os melhores resultados foram destacados em negrito.

Com base nos resultados, podemos observar que o modelo Random Forest se destacou em quase todas as métricas avaliadas, apresentando os valores para Acurácia (0.9944 ± 0.0007), Precisão (0.9966 ± 0.0017), Recall (0.9964 ± 0.0019) e F1-Score (0.9965 ± 0.0004). O SVM também obteve bons resultados, especialmente em Precisão (0.9986 ± 0.0013) e F1-Score (0.9932 ± 0.0012), embora tenha apresentado um Recall ligeiramente inferior (0.9879 ± 0.0031) quando comparado ao Random Forest.

Table 3. Tabela de Resultados dos Modelos de Classificação

Modelo	Acurácia	Precisão	Recall	F1-Score
KNN	0.9796 ± 0.0034	0.9796 ± 0.0045	0.9952 ± 0.0021	0.9873 ± 0.0021
Random Forest	**0.9944 ± 0.0007**	0.9966 ± 0.0017	**0.9964 ± 0.0019**	**0.9965 ± 0.0004**
Logistic Regression	0.9845 ± 0.0030	0.9943 ± 0.0020	0.9863 ± 0.0027	0.9903 ± 0.0019
Naive Bayes	0.2475 ± 0.0084	0.9073 ± 0.0603	0.0645 ± 0.0083	0.1204 ± 0.0147
SVM	0.9892 ± 0.0020	**0.9986** ± 0.0013	0.9879 ± 0.0031	0.9932 ± 0.0012
MLP	0.9904 ± 0.0021	0.9936 ± **0.0011**	0.9944 ± 0.0031	0.9940 ± 0.0013

O KNN e o Logistic Regression mostraram desempenho consistente, com a Acurácia do KNN sendo 0.9796 ± 0.0034 e o F1-Score do Logistic Regression atingindo 0.9903 ± 0.0019. Entretanto, o Naive Bayes teve o desempenho mais baixo em todas as métricas, com a Acurácia de apenas 0.2475 ± 0.0084 e um Recall de 0.0645 ± 0.0083, indicando que não foi adequado para a tarefa de classificação abordada neste trabalho.

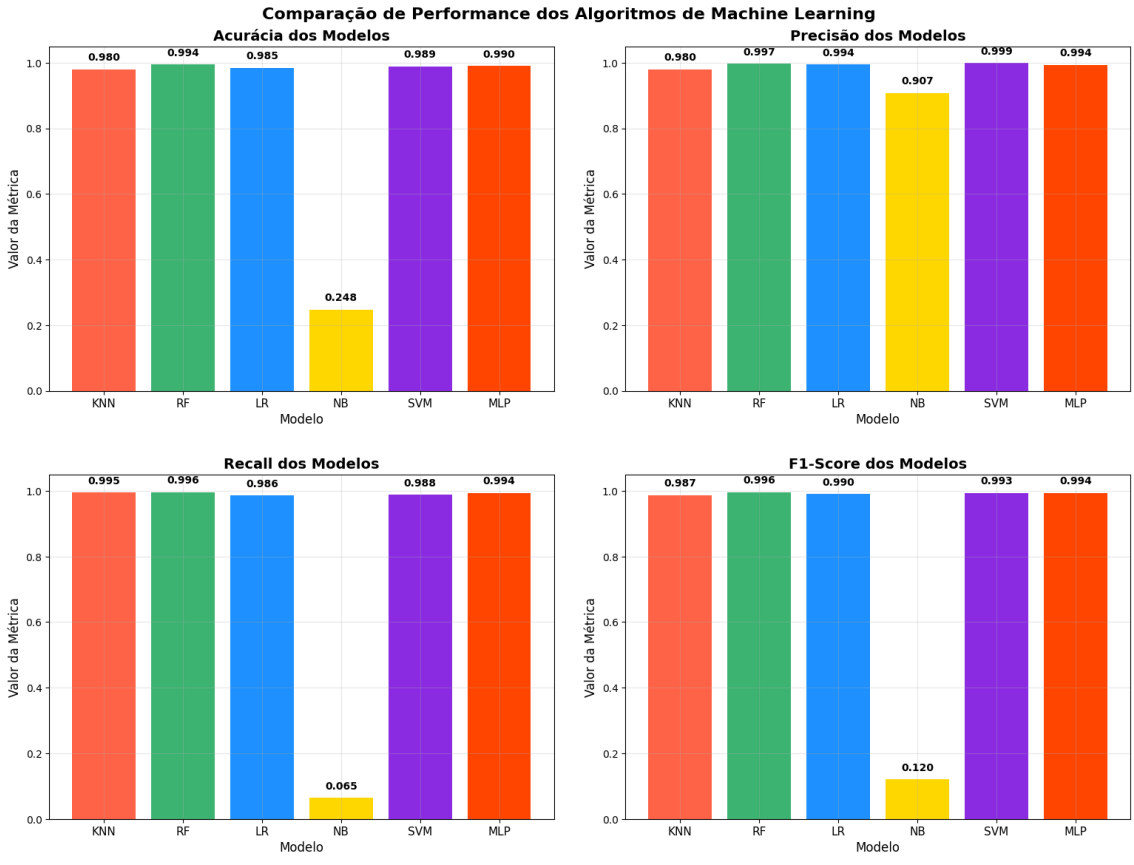

Figure 2. Desempenho Comparativo dos Modelos de Classificação (K-Fold + SMOTE)

Os gráfico apresentados na Figura 2 e Figura 3 mostram o desempenho comparativo dos modelos de classificação com base nas métricas de Acurácia, Precisão, Recall e F1-Score. Neste sentido, o modelo Random Forest se destacou como o melhor em várias métricas, como visto na tabela e no gráfico, enquanto o Naive Bayes teve um desempenho

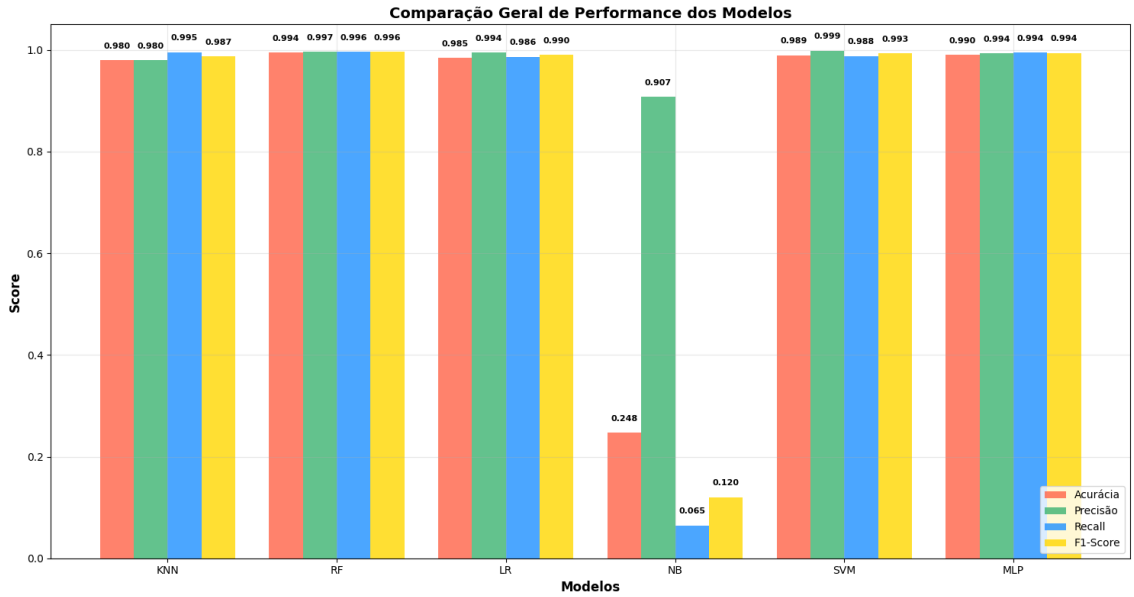

Figure 3. Comparação Geral de Performance dos Modelos

significativamente inferior. O SVM apresentou uma precisão bastante alta, embora com um desempenho um pouco inferior em Recall.

4. Conclusão

Este trabalho provocou um avanço significativo na forma como lidamos com a detecção de malware, especialmente em dispositivos móveis, ao aplicar algoritmos de classificação e técnicas de balanceamento de dados. A implementação do SMOTE para balanceamento de classes teve um efeito direto na melhoria da eficácia da detecção, permitindo que os modelos classificadores, como o Random Forest, lidassem melhor com a distribuição desbalanceada dos dados. Isso gerou um impacto notável na capacidade de identificar ameaças em um contexto digital em constante mudança, oferecendo um potencial de aplicação prática em cenários de segurança cibernética real. Ao possibilitar uma detecção mais precisa e robusta de malware, este trabalho não apenas aumenta a segurança de sistemas e dispositivos, mas também contribui para a redução de riscos associados a ataques cibernéticos. Tendo em vista o exposto, o efeito geral desse estudo é uma maior capacidade de resposta a ameaças emergentes, protegendo dados pessoais e corporativos e, consequentemente, criando um ambiente digital mais seguro e confiável para os indivíduos.

References

Ambekar, N. G., Thokchom, S., and Moulik, S. (2024). Tc-amd: Android malware detection through transfomer-cnn hybrid architecture. In *2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE.

Baburaj, D., Gupta, D., et al. (2024). A comprehensive exploration of machine learning and explainable ai techniques for malware classification. In *2024 2nd World Conference on Communication & Computing (WCONF)*, pages 1–7. IEEE.

Chandran, S., Syam, S. R., Sankaran, S., Pandey, T., and Achuthan, K. (2025). From static to ai-driven detection: A comprehensive review of obfuscated malware techniques. *IEEE Access*.

Hasan, R., Biswas, B., Samiun, M., Saleh, M. A., Prabha, M., Akter, J., Joya, F. H., and Abdullah, M. (2025). Enhancing malware detection with feature selection and scaling techniques using machine learning models. *Scientific Reports*, 15(1):9122.

Köhler, R. K., de Gouveia, L. T., and Senger, L. J. (2024). Processamento digital de imagens na classificação de agregados minerais quanto a alteração para aplicações em pavimentação. *OBSERVATÓRIO DE LA ECONOMÍA LATINOAMERICANA*, 22(8):e6130–e6130.