Avaliação de Métodos de Classificação baseados em Regras de Associação para Detecção de Malwares Android

Vanderson da Silva Rocha[1], **Diego Kreutz**[2], **Jonas Pontes**[1], **Eduardo Feitosa**[1]

[1]Instituto de Computação (IComp)
Universidade Federal do Amazonas (UFAM)
vanderson@ufam.edu.br,(pontes,efeitosa)@icomp.ufam.edu.br

[2]Laboratório de Estudos Avançados em Computação (LEA)
Universidade Federal do Pampa (UNIPAMPA)
diegokreutz@unipampa.edu.br

Abstract. *We present an exploratory analysis of the performance and feasibility of three classification models based on association rules (CBA, CMAR, CPAR) for Android malware detection. We also propose and implement a new classification model based on association rules and rule quality, named EQAR, which extends the classic ECLAT algorithm. To evaluate and compare our four models, we selected three datasets frequently used for training Android malware detection models: DREBIN-215, KronoDroid Emulator and KronoDroid Physical Device. Our findings show that classification methods based on association rules can achieve good results, but not as good as those achieved by machine learning models, such as RandomForest and SVM, for Android malware detection.*

Resumo. *O nosso principal objetivo é apresentar uma análise exploratória do desempenho e da viabilidade de três modelos de regras de associação existentes na literatura (CBA, CMAR, CPAR) no contexto de classificação de malwares Android. Além disso, desenvolvemos também um novo modelo de classificação baseado em regras de associação e qualidade de regras, denominado EQAR, que estende o algoritmo clássico ECLAT. Para fins de comparação dos quatro modelos, utilizamos três datasets frequentemente utilizados para o treino de modelos de detecção de malwares Android: DREBIN-215, KronoDroid Emulador e KronoDroid Dispositivo Real. Os resultados indicam que os métodos de classificação baseados em regras de associação apresentam bons resultados, entretanto, os métodos avaliados dificilmente conseguem atingir a estabilidade de métricas e os resultados numéricos alcançados por modelos de aprendizado de máquina, como RandomForest e SVM, no domínio de detecção de malwares Android.*

1. Introdução

A mineração de regras de associação é um procedimento que visa observar padrões, correlações ou associações que ocorrem com frequência em conjuntos de dados encontrados em bancos de dados ou outras formas de repositórios. Genericamente, a mineração de regras de associação pode ser encarada como simples instruções `If ... Then` que ajudam a descobrir relacionamentos entre os dados.

Proposta inicialmente para resolver o problema da cesta de mercado em conjuntos de dados transacionais [Agrawal et al. 1993], a mineração de regras de associação tem sido utilizada em problemas do mundo real, como gestão de relacionamento com o cliente [Kaur and Kang 2016], diagnóstico médico [Ali et al. 2019], e na área de segurança da informação (e.g., no combate à fraudes) [Jeeva and Rajsingh 2016, Sadgali et al. 2021].

No contexto de mineração de dados por regras de associação, podemos destacar os algoritmos clássicos e mais utilizados na literatura [Zhang and He 2010, Li and Sheu 2021]: **Apriori** [Agrawal and Srikant 1994], **FP-Growth** [Han et al. 2000] e **ECLAT** [Zaki 2000]. A partir destes algoritmos base, surgiram diversos métodos especializados em classificação baseada em regras de associação, como CBA [Liu et al. 1998], CMAR [Li et al. 2001], CPAR [Yin and Han 2003], MCAR [Thabtah et al. 2005] e ARCID [Abdellatif et al. 2018]. Esses métodos incorporam e evoluem os algoritmos clássicos. Por exemplo, o CMAR incrementa o FP-Growth para minerar grandes conjuntos de dados e assim melhorar a precisão e a eficiência da classificação.

Apesar de serem aplicadas em diferentes domínios, como medicina [Ali et al. 2019] e combate à fraudes [Sadgali et al. 2021], desconhecemos pesquisas que investigam a aplicação desses métodos no contexto específico de detecção de *malwares* Android. Portanto, o nosso objetivo é apresentar uma análise exploratória de diferentes métodos de classificação baseados em regras de associação para o contexto de classificação de aplicativos maliciosos Android, utilizando um conjunto de *datasets* conhecidos. Para realizar a avaliação, selecionamos os métodos CBA, CMAR e CPAR pelo fato de possuírem código fonte e ferramental disponível, bem como contemplarem os processos de geração e classificação de regras. Como esses três métodos utilizam apenas os algoritmos clássicos Apriori e FP-Growth, desenvolvemos também um quarto método, denominado EQAR, que incorpora o algoritmo clássico ECLAT, para melhorar a interpretabilidade, e funções de qualidade de regras para melhorar a eficácia e eficiência em problemas de classificação.

Como contribuições do trabalho podemos destacar: (a) uma análise empírica e compreensiva de quatro métodos de classificação baseados em regras de associação no contexto específico de detecção de *malwares* Android; (b) a proposta e implementação do método EQAR para problemas de classificação baseados em regras de associação; (c) uma comparação dos resultados dos quatro métodos com os modelos de aprendizado de máquina *Random Forest* (RF) e *Support Vector Machine* (SVM), os dois modelos mais frequentemente utilizados no domínio de detecção de *malwares* Android [Sharma and Rattan 2021], para caracterizar e quantificar, com dados numéricos, o potencial dos métodos de classificação baseados em regras de associação nesse domínio.

Este trabalho está organizado como segue. Nas Seções 2 e 3 apresentamos o contexto de mineração de regras de associação e o método EQAR, respectivamente. Na sequência, apresentamos a metodologia na Seção 4 e os resultados e uma discussão na Seção 5. Finalmente, apresentamos as considerações finais na Seção 6.

2. Mineração de Regras de Associação

Uma regra de associação baseia-se no princípio de que há uma grande chance de existir uma relação entre itens de um conjunto de dados quando eles aparecem frequentemente juntos. Na prática, essas regras são implementadas por algoritmos de mineração de regras

de associação, ou seja, procedimentos que quantificam as correlações existentes, como a frequência dos itens A e B juntos. Esses algoritmos são empregados no processamento de grandes volumes de dados [Osisanwo et al. 2017] e são capazes de lidar com a complexidade de dados de diferentes origens e com um número expressivo de variáveis.

A descoberta de regras de associação pode ser formalizada como [Agrawal et al. 1993]: seja $I = \{I_1, I_2, I_3, \cdot \cdot \cdot, I_m\}$ um conjunto de m atributos distintos, denominado **itens**; t uma transação que contém um conjunto de itens ($t \subseteq I$); e D uma base de dados com diferentes transações t, representadas por um vetor binário, com $t[k] = 1$ se t indica a presença do item k e $t[k] = 0$, caso contrário. Uma regra de associação é uma implicação da forma $X \implies Y$, onde X (antecedente) e Y (consequente) são subconjuntos de itens em I ($X \subset I$ e $Y \subset I$), onde X e Y não possuem itens em comum ($X \cap Y = \varnothing$).

As principais métricas para avaliar a importância de determinada regra de associação são o suporte e a confiança [Agrawal et al. 1993]. O **suporte** é utilizado para medir a frequência, algumas vezes interpretada como significância ou importância, de um conjunto de itens em um conjunto de dados. Se o suporte de um conjunto de itens for maior do que um limite de suporte mínimo especificado, este será referido como um **conjunto de itens frequentes**. Matematicamente, o **suporte** de uma regra $X \implies Y$ é definido como a fração de transações t em D que satisfaz a união dos itens no antecedente e consequente da regra, como pode ser visto na Equação 1. Como representado na Equação 2, a **confiança** de uma regra $X \implies Y$ é a probabilidade de ver o consequente Y em uma transação, dado que ela também contém o antecedente X.

$$Suporte(X \implies Y) = \frac{Frequência(X \cup Y)}{Total\ de\ transações\ t\ em\ D},\ [0,\ 1] \tag{1}$$

$$Confiança(X \implies Y) = \frac{Suporte(X \implies Y)}{Suporte(X)},\ [0,\ 1] \tag{2}$$

É importante ressaltar que a métrica não é simétrica ou direcionada. Por exemplo, a confiança para $X \implies Y$ é diferente da confiança para $Y \implies X$. A confiança é 1 (máxima) para uma regra $X \implies Y$ se o consequente e o antecedente ocorrerem juntos.

O problema de descobrir regras de associação pode ser decomposto em duas partes: (i) encontrar os conjuntos de itens que têm **suporte** maior ou igual a um limite mínimo (i.e., conjuntos de itens frequentes); e (ii) gerar as regras de associação a partir dos conjuntos de itens frequentes considerando o limite mínimo para a **confiança**.

2.1. Algoritmos

Os algoritmos clássicos para mineração de regras de associação, como Apriori, FP-Growth e ECLAT, são recorrentemente referenciados e utilizados na literatura [Zhang and He 2010, Li and Sheu 2021]. O **Apriori** [Agrawal and Srikant 1994] procura identificar os conjuntos de itens frequentes candidatos de k elementos a partir de conjuntos de itens de $k-1$. Toda a base de dados é rastreada e os conjuntos de itens frequentes são obtidos a partir dos conjuntos de itens candidatos. Sua principal vantagem vem do baixo consumo de memória, pois apenas os conjuntos de itens frequentes (L_{k-1}) e os conjuntos

de itens frequentes candidatos (C_k) precisam ser armazenados na memória. Entretanto, isto demanda tempo de computação e reduz a eficiência [Zhang and He 2010].

O algoritmo **FP-Growth** [Han et al. 2000] é composto por duas etapas de processamento. Na primeira etapa é construída uma representação altamente condensada da base de dados, denominada *FP-Tree*. A geração da *FP-Tree* é realizada através da estratégia de busca em profundidade e da contagem de ocorrências dos itens para encontrar os conjuntos de itens frequentes. Na segunda etapa, a *FP-Tree* é utilizada para determinar os valores de suporte para todos os itens frequentes. Embora mais escalável do que o Apriori, o FP-Growth sofre de limitações de memória, pois é custoso percorrer a *FP-Tree*.

Finalmente, o algoritmo **ECLAT** [Zaki 2000] utiliza uma estratégia que combina a busca em profundidade com interseções entre conjuntos. A ideia é manter em memória apenas a lista dos identificadores (*TIDlist*) dos itens frequentes em análise, sem precisar dividir o conjunto de dados. Quando uma intersecção entre duas *TIDlist* é realizada, a *TIDList* resultante é composta pelos itens frequentes de maior suporte. Além disso, quando o suporte mínimo não é atingido, o processo de intersecção é interrompido.

2.2. Qualidade de Regras

O termo "qualidade de regras" significa encontrar um conjunto simples de regras capaz de explicar os dados do conjunto, inclusive os dados invisíveis. A qualidade de regras pode ser alcançada através da otimização de dois critérios simultaneamente [Janssen and Fürnkranz 2010]: a **cobertura**, que faz com que o número de amostras positivas abrangidas pela regra seja maximizado; e a **consistência**, que faz com que número de amostras negativas cobertos pela regra seja minimizado. No escopo da detecção de *malwares*, a amostra positiva é maliciosa, enquanto que a amostra negativa é benigna.

Tabela 1. Tabela de Contingência

	Maliciosos	Benignos	
Cobertas Pela Regra R	p	n	$p+n$
Não Cobertas Pela Regra R	$\overline{p} = P - p$	$\overline{n} = N - n$	$\overline{p} + \overline{n}$
	P	N	t

As métricas de qualidade de regras são derivadas da análise da relação entre uma regra R e uma classe C, ou seja, aplicativos maliciosos no nosso caso de estudo. Esta relação é representada na Tabela 1, onde cada regra pode ser caracterizada por: (a) p e n: quantidade de amostras malignas (p) e benignas (n) abrangidas pela regra R; (b) P e N: número total de amostras malignas (P) e benignas (N) no conjunto de treino.

A maioria das métricas de qualidade de regras dependem de p, n, P e N, e combinam esses valores de maneiras diferentes. Como exemplos de métricas de qualidade de regra podemos citar [Lenca et al. 2007, Janssen and Fürnkranz 2010, Wróbel et al. 2016]: Kappa (KAP), Acurácia (AAC), Cobertura (COV), Precisão (PREC), Confirmação Bayesiana (BC), C1, C2 e Correlação (CORR).

3. Método EQAR

O EQAR, disponível no GitHub[1], cujas etapas são ilustradas na Figura 1, tem como principal objetivo gerar conjuntos de regras utilizando o ECLAT, algoritmo clássico não coberto pelos outros métodos, e qualidade de regras para problemas de classificação baseada em regras de associação. Na primeira etapa, o modelo gera os relacionamentos mais significativos entre as características que atendam aos valores mínimos de **suporte** e **confiança**. Na etapa seguinte, esses conjuntos de características são identificados para obter aqueles que caracterizam unicamente as amostras malignas.

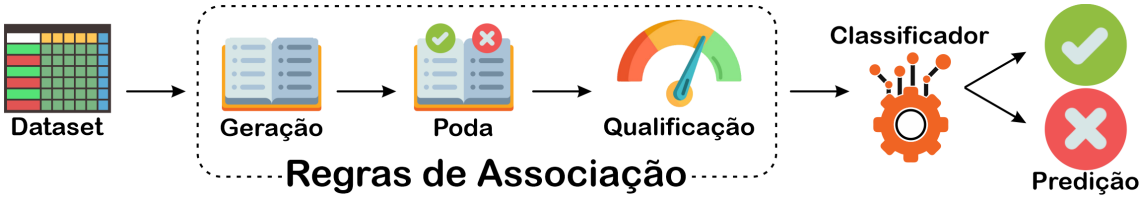


Figura 1. Etapas do método EQAR

3.1. Geração, Poda e Qualificação

A partir do *dataset* de entrada, separamos as amostras malignas e benignas para gerar as regras de ambos os conjuntos isoladamente. O Algoritmo 1 resume o processo de geração das regras de associação do EQAR.

Inicialmente, o *dataset* é dividido entre amostras malignas e benignas (linhas 1 e 2). As regras de associação são então geradas separadamente para obter as combinações de características que demonstrem os comportamentos das amostras (linhas 3 e 4).

Algoritmo 1: Geração e Poda de Regras de Associação

Entrada:
D: *dataset* de treino
minSup: suporte mínimo
minConf: confiança mínima

1 $\text{dataset}_{\text{malignas}} \leftarrow t \in D \wedge t[class] = 1$; `// amostras malignas`
2 $\text{dataset}_{\text{benignas}} \leftarrow t \in D \wedge t[class] = 0$; `// amostras benignas`
3 $\text{regras}_{\text{malignas}} \leftarrow$ **gerar_regras**($\text{dataset}_{\text{malignas}}$, minSup, minConf)
4 $\text{regras}_{\text{benignas}} \leftarrow$ **gerar_regras**($\text{dataset}_{\text{benignas}}$, minSup, minConf)
5 $\text{regras}_{\text{dif}} \leftarrow \text{regras}_{\text{malignas}} \setminus \text{regras}_{\text{benignas}}$
6 $\text{regras}_{\text{sub}} \leftarrow X \in \text{regras}_{\text{dif}} \wedge X \not\subset Y, \forall\, Y \in \text{regras}_{\text{malignas}}$
7 $\text{regras} \leftarrow X \in \text{regras}_{\text{sub}} \wedge X \not\supset Y, \forall\, Y \in \text{regras}_{\text{sub}}$
8 **retorna** *regras*

Assim que os conjuntos de regras malignas e benignas estiverem separados, o antecedente e o consequente de cada regra são unidos para formar as regras de classificação da respectiva classe. Resumidamente, o processo de geração de regras das amostras malignas produz $X \implies Y$ e o classificador interpreta a regra como $X \cup Y \implies Classe$.

A partir da diferença entre o conjunto de regras das amostras malignas ($regras_{malignas}$) e o conjunto de regras das benignas ($regras_{benignas}$), geramos um novo conjuntos de regras ($regras_{dif}$), como pode ser observado na linha 5. O intuito é remover aquelas contidas tanto nas regras geradas pelas amostras malignas quanto pelas benignas.

[1] `https://github.com/Malware-Hunter/sbseg22-eqar`

Se um conjunto de características pode classificar uma amostra, então os subconjuntos derivados também levarão à mesma classificação. Portanto, verificamos se as regras presentes no conjunto $regras_{dif}$ são subconjuntos de alguma regra do conjunto de regras das amostras benignas ($regras_{benignas}$) (linha 6) e se alguma regra deste novo conjunto ($regras_{sub}$) não é superconjunto de outras regras pertencente ao grupo (linha 7). A primeira verificação é necessária para excluir regras que possam levar a classificação da amostras tanto como malignas quanto como benignas. A segunda irá podar as regras que tenham um número maior de características e que possam produzir a mesma classificação.

Algoritmo 2: Qualificação de Regras de Associação

Entrada:
D: *dataset* de treino
regras: conjunto de regras (Algoritmo 1)
métrica: função de métrica de qualidade a ser utilizada

```
1 P ← quantidade de malignas em D
2 N ← quantidade de benignas em D
3 regras_qualidade ← ∅;                    // lista (regra, qualidade)
4 para cada regra r ∈ regras faça
5     p, n ← encontrar_cobertura(r)
6     q ← métrica(p, n, P, N)
7     regras_qualidade ← regras_qualidade ∪ (r, q)
8 retorna regras_qualidade
```

Na próxima etapa é calculado o valor da métrica de regra selecionada para o conjunto de regras remanescentes, como detalhado no Algoritmo 2. Para cada regra (linha 4), os valores de p e n são computados (linha 5) e encaminhados para calcular a métrica de qualidade de regra especificada (linha 6).

3.2. Selecionador de Regras

O conjunto derivado de regras é disposto em ordem crescente, de acordo com o valor da métrica de qualidade de regra. Em seguida, as melhores regras qualificadas são selecionadas de acordo com a porcentagem estabelecida pelo *threshold*, que atua como um limitador às regras que serão utilizadas para a avaliação das amostras.

A seleção utilizando regras de maior métrica de qualidade, a partir de um *threshold*, reduz o *overfitting*, pois garante que regras com alta cobertura recebam prioridade, o que resulta em um número menor de regras a serem avaliadas.

3.3. Predição

A avaliação das amostras é detalhada no Algoritmo 3. Para uma predição de amostras malignas (linha 8), é necessário que todas as características da regra estejam presentes na amostra em análise (linha 5) e garantir também que isto ocorra para pelo menos uma regra (linha 7).

4. Metodologia

Nesta seção apresentamos os três métodos da literatura (CBA, CMAR e CPAR) e um resumo dos quatro métodos que serão avaliados (Seção 4.1), os três *datasets* (Seção 4.2), os parâmetros e métricas (Seção 4.3) e o ambiente de execução (Seção 4.4).

4.1. Métodos de Classificação

Os métodos CBA, CMAR e CPAR possuem particularidades. Num primeiro estágio, o método CBA [Liu et al. 1998] emprega a mineração vertical, utilizando o algoritmo Apriori para gerar as regras de associação de cada classe. Na segunda etapa o método aplica uma poda, baseando-se nos métodos M1 ou M2, para produzir regras preditivas.

Algoritmo 3: Avaliação das Amostras

Entrada:
D: *dataset* de teste
regras: conjunto de regras qualificadas

```
1  predição_lista ← ∅;        // lista com as previsões do modelo
2  para cada amostra t ∈ D faça
3      regras_contador ← 0
4      para cada regra r ∈ regras faça
5          se t[r_1] = 1 ∧ t[r_2] = 1 ∧ ... ∧ t[r_n] = 1 então
6              regras_contador++
7      se regras_contador ≠ 0 então
8          predição ← 1;                          // malicioso
9      senão
10         predição ← 0;                          // benigno
11     predição_lista ← predição_lista ∪ predição
12 retorna predição_lista
```

O CMAR [Li et al. 2001] é considerado um método de classificação associativa, que emprega regras de associação múltiplas para classificação. Utilizando como base o FP-Growth para minerar grandes conjuntos de dados, o CMAR adiciona uma nova estrutura de dados, denominada *CR-Tree*, para melhorar a precisão e a eficiência. A função principal dessa estrutura é armazenar e recuperar um grande número de regras de forma compacta e eficiente.

O método de classificação CPAR [Yin and Han 2003] incorpora um algoritmo guloso para gerar regras diretamente do conjunto de dados de treino, evitando a geração de regras grandes candidatas a partir de conjuntos de itens frequentes, como ocorre nos outros métodos de classificação. Adicionalmente, o CPAR gera e testa uma quantidade maior de regras que classificadores tradicionais, evitando assim a perda de regras importantes. Para evitar o *overfitting*, o CPAR utiliza a acurácia de Laplace para avaliar cada regra e selecionar as melhores regras na previsão.

Na Tabela 2 apresentamos um resumo das principais características dos quatro métodos selecionados, CBA, CMAR, CPAR e EQAR. Como podemos observar, há uma diversidade de algoritmos de mineração utilizados pelos métodos para geração das regras, incluindo os três clássicos (i.e., Apriori/CBA, FP-Growth/CMAR e ECLAT/EQAR) e o PRM [Yin and Han 2003], empregado pelo CPAR.

Para classificação de regras, CBA, CMAR e EQAR utilizam essencialmente confiança e suporte. Já o CPAR emprega a acurácia de Laplace [Clark and Boswell 1991] para estimar o erro esperado de uma regra. No CBA, quando duas regras têm suporte e confiança idênticos, a regra escolhida é que foi gerada primeiro. Diferentemente, o CMAR utiliza a cardinalidade, isto é, a quantidade de itens que compõem a regra.

Com relação à poda, CBA e CMAR utilizam essencialmente cobertura, enquanto que o CPAR utiliza a seleção das regras de acordo com os K valores da acurácia de

Laplace mais altos e o EQAR utiliza a diferença de conjuntos e qualidade de regras, podando e priorizando as regras geradas que possuem maior cobertura. É importante ressaltar que, diferentemente dos outros métodos, o EQAR mantém todas as regras que caracterizam unicamente os comportamentos maliciosos até a etapa de predição.

Tabela 2. Métodos de Classificação baseados em Regras de Associação

Método	Geração de Regras	Classificação das Regras	Poda	Predição
CBA	Apriori	Confiança, Suporte, Geradas Primeiro	Cobertura (Método M1)	Máxima Probabilidade
CMAR	FP-Growth	Confiança, Suporte, Cardinalidade	Cobertura (Método M1), Regras Redundantes	Múltiplas Regras
CPAR	PRM	Acurácia de Laplace	Seleção dos K Melhores	Múltiplas Regras
EQAR	ECLAT	Confiança e Suporte	Qualidade de Regra, Diferença de Conjuntos	Correspondência Exata com Alguma Regra

Na predição, enquanto que o CPAR e CMAR utilizam múltiplas regras, o CBA baseia-se em máxima probabilidade e o EQAR em correspondência exata. Especificamente, o CMAR seleciona múltiplas regras de alta confiança aplicáveis a uma amostra de teste e analisa a correlação entre elas, medida através de um chi-quadrado ponderado. No caso do CPAR, o método utiliza as melhores regras de cada classe para predição, escolhendo a classe com a maior acurácia de Laplace. Já o CBA, adotando a máxima probabilidade, utiliza um classificador em um conjunto de regras R e uma amostra de teste t, onde apenas a regra de precedência mais alta em R, que corresponde à amostra de teste, é considerada. Nos casos em que não há regra aplicável em R para cobrir t, então t é associado à classe padrão, neste caso a classe dominante no conjunto de dados de treino [Thabtah 2007]. Finalmente, o EQAR selecionada para o classificador a regra que é capaz de classificar pelo menos uma amostra de treino.

4.2. *Datasets*

Utilizamos três *datasets*, conforme resumido na Tabela 3, disponíveis publicamente para avaliar os quatro métodos, sendo duas instâncias do KronoDroid[2] e uma do DREBIN-215[3], que é um sub-conjunto do *dataset* Drebin [Arp et al. 2014]. É importante destacar que ambos os *datasets* são utilizados com frequência por diversos trabalhos atuais de detecção de *malwares* Android, como [Guerra-Manzanares et al. 2021, Islam et al. 2021].

[2] `https://github.com/aleguma/kronodroid`

[3] `https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653`

Tabela 3. Resumo dos *datasets*

Dataset	Características	Amostras		
		Malignas	Benignas	Total
KronoDroid Emulador	383	28745	35246	63991
KronoDroid Dispositivo Real	383	41382	36755	78137
DREBIN-215	215	5560	9476	15036

O KronoDroid possui características estáticas e dinâmicas de aplicativos Android, extraídas de emulador e de dispositivo real, em dois conjuntos de dados distintos. Selecionamos as permissões e as chamadas de sistema, pois são consideradas as características mais representativas desses dois *datasets* para a detecção de *malwares* Android [Wang et al. 2019]. Com isso, o primeiro conjunto de dados, de emulador, ficou com 63.991 amostras (28.745 malignas e 35.246 benignas) e o segundo conjunto, de dispositivos reais, possui 78.137 amostras, sendo 41.382 malignas e 36.755 benignas. Ambos os *datasets* contém 383 características, sendo 166 permissões e 217 chamadas de sistema.

Diferentemente do KronoDroid, o DREBIN-215 apresenta um maior desbalanceamento entre as amostras, isto é, o número de amostras benignas é 70% maior do que o de malignas. Ao total, o *dataset* contém 15.036 amostras, sendo 5.560 malignas e 9.476 benignas. Em termos de características, o DREBIN-215 contém 113 permissões, 73 chamadas de API, 23 intenções e 6 comandos de sistema.

4.3. Parâmetros e Métricas

Para geração das regras, fixamos o **tamanho máximo da regra** em **5**, valor definido através de experimentos iniciais. Os números desses experimentos demonstraram que utilizar regras de associação compostas por mais de 5 características acrescenta muito pouco aos métodos e aumentava significativamente o custo computacional. Fixamos ainda a **confiança** em 95%, um valor adequado [Sun et al. 2016] para selecionarmos as características que ocorrem simultaneamente na maioria das amostras dos *datasets*.

Com relação aos modelos de aprendizado de máquina RF e SVM, utilizamos a execução padrão apresentada na biblioteca *scikit-learn*, sem qualquer alteração de parâmetro ou hiper-parâmetro. O RF é executado com 100 árvores na floresta e considera a raiz quadrada da quantidade de características ao procurar a melhor divisão entre as classes. Já o SVM utiliza o kernel *Radial Basis Function* (RBF) e uma heurística de redução para diminuir o tempo de treino.

Como métricas de avaliação, além das tradicionais (acurácia, precisão, *recall* e F1 Score), utilizamos também o Coeficiente de Correlação de Matthews (MCC) [Chicco and Jurman 2020], particularmente útil quando as duas classes (malignas e benignas) estão desequilibradas, ou seja, uma classe aparece muito mais que a outra. Avaliando verdadeiro positivo (TP), verdadeiro negativo (TN), falso positivo (FP) e falso negativo (FN), conforme Equação 3, o MCC realiza uma correlação entre o valor observado e o valor previsto na classificação, retornando um valor entre -1 e +1, onde +1 representa a predição perfeita, 0 representa uma predição mediana e -1 uma predição inversa.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \quad (3)$$

É importante destacar que, no contexto de detecção de *malwares* em Android, o *recall* é uma das métricas mais relevante, pois ela representa a porcentagem de aplicativos reconhecidamente maliciosos e detectados como tal.

Utilizamos também a técnica de **validação cruzada estratificada** para a avaliação, dividindo e alternando os conjuntos de treino e teste. O número de partições (*folds*) foi fixado em 5 (k = 5), o que significa que em cada iteração 80% das amostras serão utilizadas para treino e 20% para teste.

4.4. Ambiente

Utilizamos um computador com processador Intel Xeon E5-4617 Octa-core de 2.90GHz, com 32GB RAM e 150GB de HD para a execução dos experimentos. O sistema operacional utilizado foi o Linux Ubuntu 20.04.3 LTS.

Para a implementação e automação da execução do EQAR, utilizamos Python 3.8 e as bibliotecas numpy (versão 1.21.4), pandas (versão 1.3.4), scikit-learn (versão 1.0.1) e pyFIM[4] (versão 6.28). No caso do CBA, utilizamos a implementação Python pyARC[5] do método. Finalmente, utilizamos a implementação arulesCBA[6], em linguagem R para os métodos CMAR e o CPAR.

5. Resultados

Nesta seção apresentamos os resultados do ajuste do suporte e do *threshold* do EQAR (Seção 5.1), necessários à etapa de classificação. Na sequência, Seção 5.2, apresentamos desempenho de classificação dos métodos CBA, CPAR, CMAR e EQAR, e comparamos os resultados com um modelo de aprendizado de máquina supervisionado baseado no SVM.

5.1. Suporte e *Threshold*

O valor de suporte desempenha um papel importante na predição geral de classificadores derivados de técnicas de classificação associativa. Se utilizarmos um valor de suporte muito alto, corremos o risco de ignorar regras de boa qualidade. Já com um suporte muito baixo, potencializamos o problema de *overfitting*, levando a regras redundantes que poderão consumir mais tempo de processamento [Thabtah et al. 2005]. Portanto, o primeiro passo é definirmos os valores de **suporte**, para todos os métodos, e ***threshold***, apenas para o EQAR, mais adequados para os processos de classificação.

Com o objetivo de qualificar os classificadores, avaliamos o suporte e o *threshold* com os valores de 10%, 20% e 30%, um intervalo compatível com a literatura. No gráfico da Figura 2 apresentamos os valores de acurácia para os três valores de suporte e *threshold*. Como podemos observar, um suporte e *threshold* de 10% leva aos melhores

[4] https://borgelt.net/pyfim.html
[5] https://github.com/jirifilip/pyARC
[6] https://github.com/ianjjohnson/arulesCBA

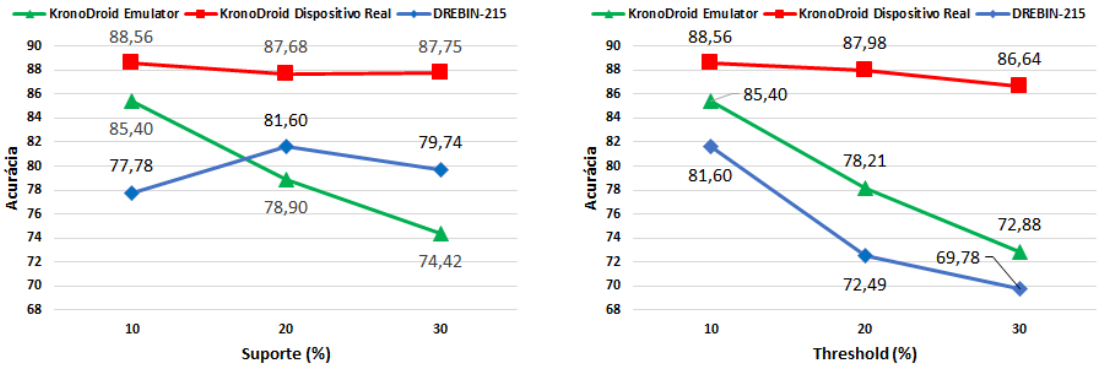


Figura 2. Suporte e *threshold* em 10%, 20% e 30%

resultados para *datasets* como o KronoDroid. Entretanto, um suporte de 20% e um *threshold* de 10% levam a melhores resultados para *datasets* como o DREBIN-215. O principal motivo dessa divergência entre os *datasets* é o fato de o DREBIN-215 conter menos características e amostras, demandando um suporte maior, isto é, que seja capaz de gerar regras que generalizem melhor os dados.

5.2. Desempenho dos Métodos de Classificação

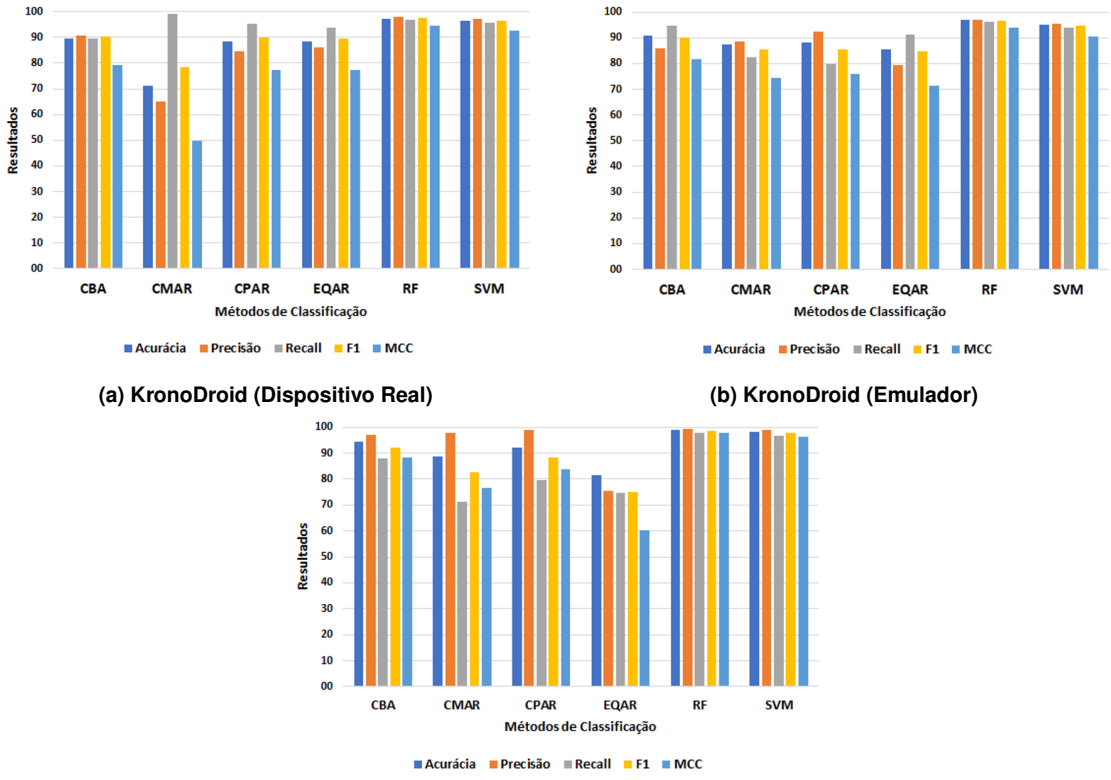
Na Figura 3 apresentamos os resultados dos métodos CBA, CMAR, CPAR e EQAR para os três *datasets*. Como pode ser observado, adicionamos também dois modelos baseados no RF e SVM para fins de comparação e discussão.

Para o *dataset* KronoDroid (Dispositivo Real), cujo gráfico é apresentado na Figura 3a, podemos observar que os métodos CBA, CPAR e EQAR apresentaram métricas como acurácia e F1 *Score* semelhantes, enquanto o CMAR não ultrapassou 75% de acurácia, por exemplo. Entretanto, é importante observarmos também que o CPAR e EQAR apresentaram um *recall* ligeiramente superior ao CBA, o que é crucial para a detecção de *malwares*. Com relação ao MCC, o CBA obteve os melhores resultados, pois é mais estável entre as diferentes métricas. Já o CMAR obteve o pior resultado de MCC, pois possui um desequilíbrio acentuado entre as métricas.

Observando as métricas de *recall* e MCC, o CBA manteve-se melhor que os demais métodos para os outros dois *datasets*, como pode ser visto nas Figuras 3b e 3c, o que demonstra uma maior estabilidade do método frente a diversidade incorporada por diferentes conjuntos de dados. É interessante observar também que houve uma piora significativa do método EQAR para o *dataset* DREBIN-215.

O EQAR utiliza as regras de associação geradas para as amostras malignas apenas para excluir as regras que são redundantes em relação às benignas. Como a qualificação é aplica somente nas regras de amostras malignas remanescentes, a quantidade de benignas impacta a predição. Analisando os *datasets* avaliados, podemos observar que a proporcionalidade entre amostras malignas e benignas é de +12,59%, -22,62% e -70,43% para o KronoDroid Disposito Real, KronoDroid Emulador e DREBIN-215, respectivamente. Essa diferença explica o comportamento do EQAR (i.e., degradação de desempenho) no gráfico do *dataset* DREBIN-215, na Figura 3c.

Por fim, podemos observar também que os modelos baseados em RF e SVM supe-



(a) KronoDroid (Dispositivo Real)

(b) KronoDroid (Emulador)

(c) DREBIN-215

Figura 3. Resultados dos métodos para acurácia, precisão, *recall*, F1 e MCC

raram os métodos de classificação baseados em regras de associação em todas as métricas para os três *datasets*. Isto nos permite concluir que modelos de aprendizado de máquina são, de fato, mais eficazes para a detecção de *malwares* Android. É importante observarmos também que o classificador SVM, assumido como sensível ao alto desequilíbrio de classe [Akbani et al. 2004], obteve resultados muitos bons, mesmo para *datasets* significativamente desbalanceados, como é o caso do DREBIN-215 (5.560 amostras malignas e 9.476 amostras benignas).

6. Considerações Finais

Neste trabalho apresentamos uma primeira avaliação de métodos de classificação baseados em regras de associação para o contexto de detecção de *malwares* Android. Além de utilizarmos três métodos existentes, CBA, CPAR e CMAR, propomos, implementamos e avaliamos também um quarto método, o EQAR, baseado no algoritmo clássico ECLAT, não coberto pelos outros métodos.

Utilizando três *datasets* publicamente disponíveis (KronoDroid Dispositivos, KronoDroid Emulador e DREBIN-215), demonstramos a qualidade de detecção dos métodos, comparando-os com dois modelos de aprendizado de máquina, o RF e o SVM. Os resultados indicam que os métodos de classificação baseados em regras de associação, que utilizam os algoritmos clássicos Apriori, FP-Growth, ECLAT e PRM, dificilmente conseguem atingir a estabilidade de métricas e os resultados numéricos alcançados por modelos de aprendizado de máquina no domínio de detecção de *malwares* Android.

Como trabalhos futuros, podemos mencionar:

1. a investigação de outras abordagens para geração de itens frequentes relacionados às regras de associação, expandindo e aprimorando modelos como o EQAR para a utilização de atributos contínuos e classificação multi-classes;
2. análise e proposição de técnicas para redução de falsos positivos e negativos;
3. avaliação dos métodos de classificação com outros *datasets* de detecção de *malwares* Android, aumentando a diversidade e representatividade do domínio de detecção de *malwares* Android; e
4. comparação dos métodos com outros modelos de aprendizado de máquina, como árvores de decisão por exemplo.

Agradecimentos

Esta pesquisa foi parcialmente financiada, conforme previsto nos Arts. 21 e 22 do decreto no. 10.521/2020, nos termos da Lei Federal no. 8.387/1991, através do convênio no. 003/2021, firmado entre ICOMP/UFAM, Flextronics da Amazônia Ltda e Motorola Mobility Comércio de Produtos Eletrônicos Ltda. O presente trabalho foi realizado também com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Abdellatif, S., Ben Hassine, M. A., Ben Yahia, S., and Bouzeghoub, A. (2018). ARCID: a new approach to deal with imbalanced datasets classification. In *SOFSEM*.

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. In *ACM SIGMOD*, page 207–216. ACM.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Cery large Data Bases, VLDB*, volume 1215, pages 487–499. Citeseer.

Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50.

Ali, Y., Farooq, A., Alam, T. M., Farooq, M. S., Awan, M. J., and Baig, T. I. (2019). Detection of schistosomiasis factors using association rule mining. *IEEE Access*, 7:18618.

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., and Siemens, C. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*, volume 14, pages 23–26.

Chicco, D. and Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *European Working Session on Learning*, pages 151–163. Springer.

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.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12.

Islam, F. Z., Jamil, A., and Momen, S. (2021). Evaluation of machine learning methods for android malware detection using static features. In *IEEE IICAIET*, pages 1–6.

Janssen, F. and Fürnkranz, J. (2010). On the quest for optimal rule learning heuristics. *Machine Learning*, 78(3):343–379.

Jeeva, S. C. and Rajsingh, E. B. (2016). Intelligent phishing url detection using association rule mining. *Human-centric Computing and Information Sciences*, 6(1):1–19.

Kaur, M. and Kang, S. (2016). Market basket analysis: Identify the changing trends of market data using association rule mining. *Procedia Computer Science*, 85:78–85.

Lenca, P., Vaillant, B., Meyer, P., and Lallich, S. (2007). Association rule interestingness measures: Experimental and theoretical studies. In *Quality Measures in Data Mining*.

Li, H. and Sheu, P. C.-Y. (2021). A scalable association rule learning heuristic for large datasets. *Journal of Big Data*, 8(1):1–32.

Li, W., Han, J., and Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. In *IEEE ICDM*, pages 369–376.

Liu, B., Hsu, W., Ma, Y., et al. (1998). Integrating classification and association rule mining. In *Kdd*, volume 98, pages 80–86.

Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, J., Olakanmi, O., and Akinjobi, J. (2017). Supervised machine learning algorithms: classification and comparison. *IJCTT*, 48(3):128.

Sadgali, I., Sael, N., and Benabbou, F. (2021). Human behavior scoring in credit card fraud detection. *IAES International Journal of Artificial Intelligence*, 10:698–706.

Sharma, T. and Rattan, D. (2021). Malicious application detection in android—a systematic literature review. *Computer Science Review*, 40:100373.

Sun, L., Li, Z., Yan, Q., Srisa-an, W., and Pan, Y. (2016). SigPID: significant permission identification for android malware detection. In *11th MALWARE*, pages 1–8. IEEE.

Thabtah, F. (2007). A review of associative classification mining. *The Knowledge Engineering Review*, 22(1):37–65.

Thabtah, F., Cowling, P., and Peng, Y. (2005). MCAR: multi-class classification based on association rule. In *3rd ACS/IEEE AICCSA*, pages 33–.

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.

Wróbel, Ł., Sikora, M., and Michalak, M. (2016). Rule quality measures settings in classification, regression and survival rule induction—an empirical approach. *Fundamenta Inf.*, 149(4):419.

Yin, X. and Han, J. (2003). CPAR: Classification based on predictive association rules. In *Proceedings of the SIAM international conference on data mining*, pages 331–335.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*, 12(3):372–390.

Zhang, M. and He, C. (2010). Survey on association rules mining algorithms. In *Advancing Computing, Communication, Control and Management*, pages 111–118. Springer.