Análise do impacto de viés nos conjuntos de dados para detecção de Malwares Android (versão estendida)*

Lucas Vilanova[1], Renato Sayyed[1], Taina Soares[1], Guilherme Siqueira[1]
Gustavo Rodrigues[1,3], Jonas Pontes[2], Eduardo Feitosa[2], Diego Kreutz[1]

[1]Laboratório de Estudos Avançados em Computação (LEA)
Programa de Pós-Graduação em Engenharia de Software (PPGES)
Universidade Federal do Pampa (Unipampa)
[2]Grupo de Pesquisa em Tecnologias Emergentes e Segurança de Sistemas (ETSS)
Instituto de Computação (IComp)
Universidade Federal do Amazonas (UFAM)
[3]Combate à Fraude

{NomeSobrenome}.aluno@unipampa.edu.br
gustavo.rodrigues@combateafraude.com
{pontes,efeitosa}@icomp.ufam.br
diegokreutz@unipampa.edu.br

Resumo. *Atualmente, a detecção de malwares Android é realizada, majoritariamente, através de modelos de aprendizado de máquina. O problema é que a maioria dos modelos desenvolvidos têm sido treinados com conjuntos de dados defasados (e.g., de 2012). Nosso objetivo é coletar e apresentar evidências para demonstrar o impacto de diferentes datasets no desempenho de modelos preditivos. Para isto, utilizamos conjuntos de dados de diferentes períodos temporais, isto é, de 2008 a 2021.*

1. Introdução

O sistema Android é atualmente um alvo frequente de aplicações maliciosas (*malwares*). Estudos apontam que modelos de aprendizado de máquina (*machine learning*) conseguem detectar *malwares* com acurácia acima de 90% [Sharma and Rattan, 2021]. Contudo, os *malwares* aumentam em número e sofisticação, exigindo atualização e desenvolvimento de novos modelos [Sahay et al., 2020].

Como modelos de *machine learning* aprendem com os dados de entrada, o desempenho (i.e., capacidade preditiva) destes é impactado fortemente pelos *datasets* (conjunto de dados) utilizados para treino [Allix et al., 2015]. Ainda assim, a maioria das soluções aplicadas à detecção de *malwares* utiliza *datasets* antigos (e.g., The Drebin Dataset, de 2012) ou defasados (e.g., características de APIs antigas) [Soares et al., 2021b].

Levantamos a hipótese de que ao treinarmos modelos de *machine learning* com um *dataset* defasado, eles não irão apresentar o mesmo poder preditivo para conjuntos de dados atuais. A premissa é simples: o perfil e o comportamento dos *malwares* Android

*Este artigo é uma versão estendida do paper [Vilanova et al., 2021], de 6 páginas, originalmente publicado no WRSeg 2021 e convidado para publicação na ReABTIC.

muda com o passar do tempo. Aplicativos mais recentes podem possuir mais caraterísticas ou ainda explorar outras características quando comparados com aplicativos maliciosos antigos. Dentre os diversos indícios que corroboram a hipótese e a premissa, podemos destacar: a evolução das APIs dos aplicativos Android (da versão 1 em 2008 a versão 31 em 2021); a constante evolução e sofisticação do comportamento dos aplicativos maliciosos; e o aumento expressivo na quantidade de *malwares*, conforme demonstrado recorrentemente por relatórios especializados [Beckert-Plewka et al., 2019, Kaspersky, 2020, AV-TEST Institute, 2021, Islam, 2021].

Com o objetivo de verificar a hipótese levantada, neste trabalho propomos três experimentos para a análise do impacto no desempenho de um modelo de aprendizagem de máquina utilizando conjuntos de dados de diferentes períodos de tempo (i.e., mais antigos e mais atuais). Para a construção do modelo, escolhemos o algoritmo Random Forest [Breiman, 2001] por ser um dos mais utilizados na detecção de *malwares* Android [Sharma and Rattan, 2021].

As contribuições do trabalho podem ser resumidas em:

1. caracterização do impacto da utilização de uma mesma configuração de um modelo, adequado a um conjunto de dados defasados, em *datasets* recentes;
2. identificação de dados duplicados em *datasets* amplamente utilizados na literatura, bem como demonstração dos efeitos desses dados nos modelos de aprendizado de máquina; e
3. a criação de um *dataset* com permissões de aplicativos Android atuais (i.e., 2018 a 2021).

O trabalho está organizado como segue. Nas Seções 2, 3 e 4 apresentamos os *datasets*, o modelo e os resultados, respectivamente. Na Seção 5 discutimos a atualidade dos *datasets* e, finalmente, na Seção 6 apresentamos as considerações finais.

2. Datasets

Dos 6 (seis) *datasets* utilizados, 1 (um) foi criado pela nossa equipe, e 5 (cinco) foram selecionados a partir do levantamento apresentado em [Soares et al., 2021b]. Eles são apresentados na Tabela 1. Para a seleção dos *datasets*, utilizamos dois critérios: (1) o mais popular na literatura, naquele ano, e (2) os que estavam publicamente disponíveis. Selecionamos então, conjuntos de dados com o intervalo de 3 anos (2012, 2015, 2018 e 2021).

Conjuntos de dados como o Drebin-215 e Androcrawl contêm características estáticas como permissões, chamadas de API e intenções. O KronoDroid *dataset* também possui características dinâmicas. Entretanto, como o objetivo é utilizar apenas permissões, pois são as características mais relevantes e utilizadas [Sharma and Rattan, 2021], removemos as demais características. Ao final desse processo e da limpeza de dados, os *datasets* Drebin-215, Androcrawl, AndroidMalwareNormal, KronoDroid, DefenseDroid e MotoDroid_22K ficaram com 113, 58, 173, 166, 134 e 144 permissões, respectivamente.

O MotoDroid_22K é um conjunto de dados de permissões Android, contendo amostras de aplicativos atuais (i.e., 2018 a 2021). Esse *dataset* foi criado pelo fato de nenhum dos outros *datasets* conter um número representativo de amostras de *malwares*

Tabela 1. Datasets

Datasets	Data	Com Duplicatas		Sem Duplicatas		Permissões	Permissões após limpeza
		Benignos	Malignos	Benignos	Malignos		
Drebin-215	2012	9.476	5.560	5.539	1.720	114	113
Androcrawl	2014-2015	139.670	23.313	138.867	23.247	61	58
AndroidMalwareNormal	2018	18.850	9.999	16.698	7.026	173	173
KronoDroid	2008-2021	17.560	16.755	6.747	6.533	166	166
DefenseDroid	2021	5.975	6000	3.146	4.579	1.490	134
MotoDroid_22K	2018-2021	11.172	11.654	11.032	11.144	148	144

atuais. Para a construção do *dataset*, utilizamos o repositório de aplicativos Android AndroZoo[1]. Além dos aplicativos, o repositório disponibiliza também metadados sobre cada aplicativo (e.g., classificação do aplicativo como *malware* segundo diversos scanners online).

2.1. Preparação dos dados

A preparação dos dados é um conjunto de etapas fundamentais para melhorar e adequar o conjunto de dados ao algoritmo de aprendizado de máquina [Zheng and Casari, 2018]. Por exemplo, a preparação pode evitar ruídos nos dados (como no caso do Androcrawl *dataset*, que continha todos os valores com formato textual e apresentava valores inválidos) que poderiam causar erros nos cálculos do modelo por não estarem em um formato numérico ou por apresentarem valores faltantes [Qi et al., 2018]. A preparação dos 6 *datasets* incluiu a exclusão de características (e.g., a permissão `SET_WALLPAPER`, duplicada no Drebin-215) de forma a auxiliar o modelo a realizar seus cálculos.

No *dataset* Androcrawl identificamos três permissões com apenas valores "0" (`RECEIVE`, `MESSAGE`, `SEND`) e com 10.262 exemplos de aplicativos Android com valores inválidos (i.e., "?"). Para que o *dataset* não sofresse uma redução significativa nos exemplos de aplicativos caso removêssemos cada registro com valores faltantes, essas três permissões foram removidas. O KronoDroid possuía dois conjuntos de dados separados, um benigno e um maligno. Concatenamos os dois conjuntos e embaralhamos as amostras para consolidar o *dataset* de 384 características. Em seguida, removemos as características dinâmicas, deixando apenas as estáticas (i.e., permissões), resultando em um conjunto final de 167 características.

No DefenseDroid removemos 1.356 características, incluindo aquelas duplicadas e permissões que não fazem parte da lista de permissões do Android[2], resultando em um *dataset* de 134 permissões. No caso do AndroidMalwareNormal e MotoDroid_22K, foram removidas apenas características como "Category" e "Package".

[1] `https://androzoo.uni.lu`

[2] `https://developer.android.com/reference/android/Manifest.permission`

O que mais nos chamou atenção na etapa de preparação dos dados foram os exemplos de aplicativos Android duplicados. Há 21.035, 7.777, 5.125, 4.250, 869 e 659 duplicatas nos *datasets* KronoDroid, Drebin-215, AndroidMalwareNormal, DefenseDroid, Androcrawl e MotoDroid_22K, respectivamente. Os exemplos duplicados em *datasets* podem impactar no desempenho dos algoritmos de aprendizado de máquina, podendo levar ao *overfitting* e enviesamento do modelo [Zhao et al., 2021]. O conjunto de dados utilizado para teste e validação de um modelo de *machine learning* precisa ser diferente daquele outrora usado na fase de treino, caso contrário o modelo saberá predizer corretamente que aquele aplicativo específico é benigno ou malicioso, resultando em uma acurácia falsa. Por essa razão, removemos todas as duplicatas, mantendo apenas a primeira ocorrência.

2.2. *Datasets* dos experimentos

A partir do processo de limpeza e redução dos *datasets*, criamos novos *datasets* para cada experimento em particular. No primeiro experimento utilizamos todos os *datasets* e todas as permissões contidas em cada um deles. No segundo experimento utilizamos apenas quatro conjuntos de dados (Drebin-215, Androcrawl, AndroidMalwareNormal e DefenseDroid) e mantivemos apenas as 14 características comuns entre eles. Finalmente, no terceiro experimento selecionamos um segundo conjunto de 4 *datasets* (Drebin-215, KronoDroid, DefenseDroid e MotoDroid_22K), cujo critério foi maximizar o número de características comuns, passando de 14 para 56 características.

É importante destacar também que tomamos cuidado com relação às amostras idênticas de aplicativos. Diferentemente das duplicatas, as amostras idênticas, potencializadas pela redução das características para dois dos experimentos, devem ser mantidas no *dataset* uma vez que representam diferentes aplicativos.

Os dados originais, que passaram pelo processo de limpeza e correção, bem como os conjuntos de dados resultantes e suas análises exploratórias, utilizados neste trabalho, estão disponíveis em `https://github.com/Malware-Hunter/Dataset-Matters`.

3. Modelo RandomForest

O algoritmo RandomForest é baseado em diversas árvores de decisão utilizadas em amostras de dados selecionadas aleatoriamente [Breiman, 2001]. A partir das previsões de cada árvore é selecionada a melhor solução por meio de votação.

Os dados de cada um dos seis *datasets* em cada experimento foram separados em três partes, sendo 40% para treino, 30% para validação e 30% para teste. Além disso, devido à diferença da amostragem dentre *datasets*, padronizamos para o menor conjunto, o que reduz a probabilidade de que um *dataset* aprenda mais que outro por conta da quantidade de amostras de aplicativos. Para isso, selecionamos a menor quantidade de cada classe entre todos os *datasets* sem duplicatas (3.146 benignos e 1.720 malignos) e com duplicatas (5.975 benignos e 5.560 malignos), e a padronizamos como quantidade total de exemplos fornecidos ao conjunto de treino nos seis conjuntos de dados, em todos os experimentos. Assim, o impacto da diferença da quantidade de exemplos treinados em cada *dataset* no desempenho do modelo é anulado. Utilizamos o parâmetro `random_state=42` para selecionar os mesmos conjuntos de dados em cada partição e não modificar os resultados a cada treino.

Como o nosso objetivo é classificar aplicativos entre benignos e malignos, considerando conjuntos de dados desbalanceados, utilizamos a métrica ROC-AUC (ou curva ROC) [Fawcett, 2006, Halimu et al., 2019]. A curva ROC é uma métrica de avaliação que descreve a capacidade do modelo de distinguir entre duas classes (benignos e malignos), na qual é necessário calcular as probabilidades de cada observação (exemplo de um aplicativo) pertencer a uma classe [Muschelli, 2020]. O valor de ROC-AUC varia de 0,0 até 1,0, em que o 0,0 representa um modelo que erra todas as predições e 1,0 indica um teste perfeito de predição.

3.1. Otimização do modelo

O trabalho consiste em analisar o desempenho de um modelo, configurado para um *dataset* defasado, em conjuntos de dados mais recentes. O *dataset* de referência selecionado foi o Drebin-215 (de 2012). Para cada um dos três experimentos detalhados a seguir, iniciamos com a adqueção do modelo ao *dataset* Drebin-215.

Experimento 1: validamos o modelo em conjuntos de dados mais recentes, utilizando todas permissões de cada *dataset*;

Experimento 2: utilizando apenas 14 características comuns entre os *datasets* Drebin-215, Androcrawl, AndroidMalwareNormal e DefenseDroid, validamos o modelo em conjuntos de dados mais recentes;

Experimento 3: utilizando *datasets* com 56 características em comum (Drebin-215, KronoDroid, DefenseDroid e MotoDroid_22K), validamos o modelo em conjuntos de dados mais recentes.

Primeiramente, foi necessário balancearmos os dados de treino a fim de nivelar a proporção entre exemplos de aplicações benignas e malignas, pois como a classe majoritária (benignos para os *datasets* com duplicatas e malignos para os *datasets* sem duplicatas) está presente em excesso, isso pode ocasionar em *overfitting* no algoritmo, fazendo com que o modelo acabe errando muitos exemplos da classe minoritária.

Com o conjunto de dados de treino balanceado, treinamos o classificador RandomForest com seus parâmetros *default* (i.e., parâmetros padrão fornecidos pela biblioteca do *scikit-learn* [3]) e o validamos com dados desbalanceados para aproximar o teste de um caso real. Com isto, obtivemos um resultado de 88,59% de ROC-AUC, e com uma garantia maior de não incluir viés no modelo.

O passo seguinte no desenvolvimento de um modelo é ajustar os hiperparâmetros do algoritmo, pois eles impactam nas taxas de detecção [Probst et al., 2019]. Utilizamos a função `RandomizedSearchCV` da biblioteca `scikit-learn` [4]. O espaço de busca dos hiperparâmetros, bem como a combinação ótima de valores encontrada para os experimentos sem duplicatas estão descritos na Tabela 2. O resultado do modelo otimizado no *dataset baseline* Drebin-215, utilizando todas as permissões, realizando o treino balanceado, foi de 90,61% de ROC-AUC, ganhando uma melhoria no desempenho de 2,02%.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html`

Tabela 2. Hiperparâmetros do modelo RandomForest

Hiperparâmetro	Padrão	Intervalo de busca	Experimento 1	Experimento 2	Experimento 3
n_estimators	100	200 : 2000	1.800	1400	1800
min_samples_split	2	2, 5 e 10	2	2	2
min_samples_leaf	1	1, 2 e 4	1	1	1
max_features	auto	auto e sqrt	auto	auto	auto
max_depth	None	10 : 110 e None	20	100	20
bootstrap	True	True e False	False	True	False

4. Resultados

4.1. Experimento 1

A partir dos resultados apresentados na Tabela 3, podemos concluir que o modelo configurado para o Drebin-215 obteve resultados muito baixos nos *datasets* Androcrawl e AndroidMalwareNormal. Isso se deve ao fato desses dois conjuntos de dados conterem muitas características não relevantes para os cálculos do algoritmo, prejudicando a capacidade preditiva do modelo. Os resultados da aplicação do SigPID[Li et al., 2018, Assolin et al., 2021], um método para seleção de características relevantes, cujos resultados podem ser visto na Tabela 4, comprovam a existência de um número significativo de características pouco relevantes para detecção de *malwares*.

..

Tabela 3. Resultados do modelo no Experimento 1

Dataset	ROC-AUC	
	Com duplicata	Sem duplicata
Drebin-215 (2010-2012)	93,93%	89,23%
Androcrawl (2014-2015)	54,41%	52,44%
AndroidMalwareNormal (2018)	57,30%	51,50%
KronoDroid (2008-2021)	91,03%	84,17%
DefenseDroid (2021)	85,91%	82,42%
MotoDroid_22K (2018-2021)	80,78%	78,51%

Tanto o Androcrawl quanto o AndroidMalwareNormal, contendo 58 e 173 permissões, respectivamente, contém apenas 6 permissões consideradas significativas para o modelo segundo o SigPID. Isto representa uma quantidade extremamente baixa de características com alguma relevância, resultando em uma acurácia muito baixa para esses conjuntos de dados.

Observando também os resultados do modelo para os *datasets* Drebin-215, KronoDroid, DefenseDroid e MotoDroid_22K, podemos concluir que a nossa hipótese se confirma, isto é, um modelo configurado para um *dataset* defasado não irá manter a mesma capacidade preditiva para conjuntos de dados mais atuais. A capacidade do modelo em predizer um APK adequado para amostras de 2010 até 2012, é de 89,23% de ROC_AUC, e essa acurácia decai para amostras mais atuais, como no caso dos *datasets* DefenseDroid e MotoDroid_22K. Os resultados do KronoDroid foram um pouco melhores pelo fato de

..

Tabela 4. Resultados do SigPID nos datasets

Dataset	Características		
	Antes do SigPID	**Depois do SigPID**	**Redução (%)**
Drebin-215 (2010-2012)	114	87	23,68%
Androcrawl (2014-2015)	59	7	88,13%
AndroidMalwareNormal (2018)	174	7	95,97%
KronoDroid (2008-2021)	167	124	25,74%
DefenseDroid (2021)	135	84	37,77%
MotoDroid_22K (2018-2021)	145	109	24,30%

o *dataset* conter muitas amostras antigas e uma quantidade maior de permissões relevantes. Mesmo assim, houve uma queda na capacidade de detecção quando comparado ao Drebin-215.

Adicionalmente, as duplicatas também podem causar impacto negativo no desempenho do algoritmo [Zhao et al., 2021]. Analisando os resultados da Tabela 3, observamos também que os registros duplicados levaram o modelo a exibir uma falsa acurácia. Por exemplo, o desempenho do modelo decaiu em 4,70% para o *dataset* Drebin-215 após a remoção das duplicatas. Como o Drebin-215 continha mais de 50% de exemplos duplicados, o modelo poderia estar enviesado. Com a remoção das duplicatas, temos uma garantia maior de termos um modelo que não sofre de enviesamento.

No caso do AndroidMalwareNormal, o impacto dos registros duplicados foi ainda mais significativo, reduzindo a ROC_AUC do modelo em 5,8%. As duplicatas no KronoDroid foram as que mais impactaram o modelo, reduzindo seu desempenho em 6,86%. Para os *datasets* KronoDroid, Drebin-215 e AndroidMalwareNormal, a redução foi causada principalmente pela redução significativa da quantidade de amostrar após a remoção das duplicatas.

No caso do *dataset* DefenseDroid, apesar de possuir 35% de seus dados duplicados, o impacto no resultado do modelo não foi tão significativo pelo fato de uma boa parcela das amostras de aplicativos do *dataset* utilizarem 84 permissões relevantes para a detecção de *malwares* (ver Tabela 4). Finalmente, o impacto no desempenho de detecção (apenas 2%) foi bem menor no MotoDroid_22K e no Androcrawl pelo fato de possuirem poucos registros duplicados. Em um *dataset* ideal, o número de amostras que utilizem características relevantes deveria ser significativo e não apresentar registros duplicados, resultando em um conjunto de dados de qualidade para o modelo aprender [Wei et al., 2017].

4.2. Experimentos 2 e 3

Realizamos também outros dois experimentos (numerados como 2 e 3) com *datasets* menores, derivados dos *datasets* do experimento 1, contendo a interseção de características comuns entre dois grupos de *datasets*. O experimento 2 foi realizado utilizando os 4 *datasets* com as mesmas 14 características, como detalhado na Tabela 5. Já no experimento 3 utilizamos conjuntos de dados com as mesmas 56 características, conforme detalhado na Tabela 6. Observando os dados dessas duas tabelas, podemos concluir que os resultados de ambos experimentos também ratificam a nossa hipótese. Mesmo utilizando as

mesmas permissões de aplicativos Android, o modelo não mantém a mesma performance em conjuntos de dados mais atuais. Como é sabido, com passar dos anos, os *malwares* evoluem, isto é, ficam mais sofisticados e mudam o seu comportamento.

Tabela 5. Resultados do modelo no Experimento 2

Dataset	ROC-AUC	
	Com duplicata	Sem duplicata
Drebin-215 (2010-2012)	58,07%	61,93%
Androcrawl (2014-2015)	50,68%	52,90%
AndroidMalwareNormal (2018)	49,38%	49,63%
DefenseDroid (2021)	52,67%	50,12%

Tabela 6. Resultados do modelo no Experimento 3

Dataset	ROC-AUC	
	Com duplicata	Sem duplicata
Drebin-215 (2010-2012)	80,80%	77,49%
KronoDroid (2008-2021)	52,71%	59,25%
DefenseDroid (2021)	50,98%	56,19%
MotoDroid_22K (2018-2021)	45,65%	52,71%

5. Discussão

Existe uma questão bastante peculiar com relação à atualidade dos dados dos *datasets*. Inicialmente assumimos que a data do *dataset* é a mesma da coleta dos dados. Entretanto, durante nossas análises, identificamos que existe uma probabilidade de que os dados sejam mais antigos do que o esperado (e.g., do que o informado no *dataset*). Esse problema ocorre quando os *datasets* possuem dados de versões de APIs muito antigas e não possuem nenhum dado de versões das APIs do Android (e.g., versão da API do mesmo ano de criação do *dataset* ou do ano anterior). Isto pode ocorrer por diferentes razões, como discutimos detalhadamente em outro trabalho [Soares et al., 2021a]. Por exemplo, é comum os autores compilarem *datasets* a partir de outros *datasets* previamente existentes (e mais antigos). Consequentemente, as características dos aplicativos (e.g., permissões, chamadas de API) acabam sendo defasadas.

6. Considerações Finais

Nesse artigo, avaliamos o impacto de *datasets* de diferentes datas no desempenho de um modelo baseado no algoritmo RandomForest. Os experimentos consistiram em treinar e configurar o modelo para o *dataset* Drebin-215 e avaliar seu desempenho em *datasets* mais recentes, utilizando todas as permissões presentes nos conjuntos de dados, e, utilizando também, *datasets* com as mesmas permissões.

Os resultados obtidos indicam que há uma forte relação entre o *dataset* e o desempenho do modelo. Simplesmente utilizar um *dataset* defasado não significa que o modelo irá performar bem com *datasets* mais atuais. Primeiro, os *datasets* podem conter características distintas, o que vai levar o modelo a resultados diferentes do esperado quando

este for treinado com dados defasados. Segundo, diferentemente dos *datasets* defasados, *datasets* recentes podem incorporar conjuntos de caractetísticas das novas versões das APIs do Android, que não existiam anteriormente, impactando o desempenho dos modelos. Portanto, para detectarmos *malwares* atuais, é fundamental o treinamento dos novos modelos com dados atuais. Terceiro, os resultados aplicando métodos de seleção de características, como o SigPID, indicam que há também uma relação entre a quantidade de características relevantes e o desempenho do modelo. Os números indicam que, por via de regra, quanto maior a qualidade e a significância dos dados, melhor será o aprendizado de máquina.

Como trabalhos futuros, podemos destacar:

1. a criação de um *dataset* com dados atuais, pois os que existem na literatura não contemplam as últimas versões das APIs do Android e não incorporam amostras significativas de aplicativos maliciosos e benignos de 2020 e 2021;
2. investigar extensivamente os enviesamentos causados pelos dados de *datasets* existentes (e.g., registros duplicatas); e
3. investigar questões relacionadas a rastreabilidade das amostras dos *datasets*.

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

Allix, K., Bissyandé, T. F., Klein, J., and Le Traon, Y. (2015). Are your training datasets yet relevant? In *International Symposium on Engineering Secure Software and Systems*, pages 51–67. Springer.

Assolin, J., Rocha, V., Silveira, G., Rodrigues, G., Feitosa, E., Casola, K., and Kreutz, D. (2021). Detecção de Malwares Android: reprodução da seleção de características do SigPID. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

AV-TEST Institute (2021). Malware. `https://www.av-test.org/en/statistics/malware/`.

Beckert-Plewka, K., Gierow, H., Haake, V., and Karpenstein, S. (2019). G DATA mobile malware report 2019: New high for malicious android apps. `https://www.gdatasoftware.com/news/g-data-mobile-malware-report-2019-new-high-for-malicious-android-apps`.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Halimu, C., Kasem, A., and Newaz, S. S. (2019). Empirical comparison of area under roc curve (auc) and mathew correlation coefficient (mcc) for evaluating machine learning

algorithms on imbalanced datasets for binary classification. In *Proceedings of the 3rd international conference on machine learning and soft computing*, pages 1–6.

Islam, Z. (2021). Android malware on the rise, Google's OS is more "interesting"to cybercriminals than Apple iOS. https://www.techspot.com/news/91519-android-more-interesting-average-cybercriminal-than-ios-malware.html.

Kaspersky (2020). Mobile malware evolution 2020. https://securelist.com/mobile-malware-evolution-2020/101029/.

Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., and Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics*, 14(7):3216–3225.

Muschelli, J. (2020). Roc and auc with a binary predictor: a potentially misleading metric. *Journal of Classification*, 37(3):696–708.

Probst, P., Boulesteix, A.-L., and Bischl, B. (2019). Tunability: importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965.

Qi, Z., Wang, H., Li, J., and Gao, H. (2018). Impacts of dirty data: and experimental evaluation. *arXiv preprint arXiv:1803.06071*.

Sahay, S. K., Sharma, A., and Rathore, H. (2020). Evolution of malware and its detection techniques. In *Information and Communication Technology for Sustainable Development*, pages 139–150. Springer.

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

Soares, T., Mello, J., Barcellos, L., Sayyed, R., Siqueira, G., Casola, K., Costa, E., Gustavo, N., Feitosa, E., and Kreutz, D. (2021a). Detecção de Malwares Android: Levantamento empírico da disponibilidade e da atualização das fontes de dados. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Soares, T., Siqueira, G., Barcellos, L., Sayyed, R., Vargas, L., Rodrigues, G., Assolin, J., Pontes, J., Feitosa, E., and Kreutz, D. (2021b). Detecção de Malwares Android: datasets e reprodutibilidade. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Vilanova, L., Sayyed, R., Soares, T., Siqueira, G., Rodrigues, G., Feitosa, E., and Kreutz, D. (2021). Análise do impacto de viés nos conjuntos de dados para detecção de malwares android. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Wei, F., Li, Y., Roy, S., Ou, X., and Zhou, W. (2017). Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 252–276. Springer.

Zhao, Y., Li, L., Wang, H., Cai, H., Bissyandé, T. F., Klein, J., and Grundy, J. (2021). On the impact of sample duplication in machine-learning-based android malware detection. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(3):1–38.

Zheng, A. and Casari, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. "O'Reilly Media, Inc.".