Revista da Universidade King Saud – Ciências da Computação e Informação xxx (xxxx) xxx

Listas de conteúdo disponíveis em ScienceDirect

Jornal da Universidade King Saud –
Ciências da Computação e da Informação

página inicial da revista: www.sciencedirect.com

Um modelo de seleção de recursos de várias camadas para detecção de malware do Android com base na discriminação de recursos e no ganho de informações

Parnika Bhat a,ÿ , Kamlesh Dutta

aDepartamento de Ciência da Computação e Engenharia, Instituto Nacional de Tecnologia, Hamirpur, HP, Índia

informações do artigo

Historia do artigo:
Recebido em 18 de maio de 2021
Revisado em 13 de outubro de 2021
Aceito em 5 de novembro de 2021
Disponível online xxxx

Palavras-chave:
Android
Seleção de recursos
Detecção de malware
Aprendizado de máquina
Análise estática

abstrato

Com o aumento da popularidade e sua arquitetura de sistema aberto, o Android tornou-se vulnerável a ataques maliciosos. Existem várias abordagens de detecção de malware disponíveis para fortalecer o sistema operacional Android de tais ataques. Esses detectores de malware classificam os aplicativos de destino com base nos padrões encontrados nos recursos presentes nos aplicativos Android. À medida que os dados de análise continuam a crescer, isso afeta negativamente o mecanismo de defesa do Android. Um grande número de recursos irrelevantes tornou-se o gargalo de desempenho do mecanismo de detecção. Este artigo apresenta um modelo de seleção de recursos de várias camadas, que pode descobrir recursos relevantes e significativos para melhorar a precisão das abordagens de detecção de malware. O método proposto aplica cinco técnicas de classificação de aprendizado de máquina ao conjunto de recursos selecionado. Este trabalho apresenta o Optimal Static Feature Set (OSFS) e Most Important Features (MIFs) descobertos com cada abordagem de aprendizado de máquina. Testes e análises rigorosos mostram que a classificação Random Forest atinge a maior taxa de precisão de 96,28%.

1. Introdução

O uso onipresente de telefones celulares os tornou uma parte indispensável das atividades da vida diária. O Android, sem dúvida, ganhou popularidade em relação a todos os outros sistemas operacionais disponíveis no mercado (Statcounter, 2021). O uso generalizado do Android ganhou maior atenção dos invasores cibernéticos (Kumar, 2020).
A pesquisa Check Point, publicada em setembro de 2019, revelou uma falha de segurança em smartphones Android modernos, o que os torna vulneráveis a ataques de phishing por SMS (Artyom Skrobov, 2019). O impacto do cibercrime na plataforma Android está profundamente enraizado, e a arquitetura de sistema aberto do Android agrava ainda mais o problema.

Os invasores usam arquivos.apk para explorar vulnerabilidades conhecidas e penetrar no sistema (Cebuc, 2020). Como os arquivos.apk estão prontamente disponíveis para usuários do Android, eles se tornam uma fonte fácil de ataques. Além disso, existem alguns recursos arriscados oferecidos aos aplicativos do Google Play Services. Os invasores usam essas permissões para entrar o sistema. Para aumentar o problema, o Google Play Services fornece acesso a alguns dos recursos sem o consentimento do usuário. Se alguém tentar desativá-los manualmente, isso afetará o funcionamento do sistema, conforme mostrado na Fig. 1. Os invasores usam aplicativos de terceiros embutidos e instalados pelo usuário para atacar o sistema.
Permissões perigosas solicitadas por esses aplicativos de terceiros no momento da instalação tornam o sistema mais vulnerável a ataques. A concessão de tais Permissões coloca em risco a privacidade dos usuários.
Se os usuários negarem essas permissões, eles não poderão usar o aplicativo. Quando os invasores entram no sistema, eles podem roubar os dados ou danificá-los. De acordo com (Khandelwal, 2019), os invasores lançam ataques baseados em permissão de uma maneira em que um aplicativo com menos permissões obtém acesso aos componentes para os quais eles não obtiveram permissão para acessar. Depois de obter acesso, os invasores usam o aplicativo para detectar os dados do usuário. Até mesmo os desenvolvedores de malware precisam de permissões para invocar chamadas de API incorporadas no arquivo. apk contendo código malicioso (Wu et al., 2012). Os invasores podem obter acesso aos componentes de hardware como câmera, microfone, GPS, wifi somente se o aplicativo que eles estão usando tiver permissão para usar esses componentes (Felt et al., 2011; Mandal et al., 2019). Portanto, as permissões desempenham um papel vital entre todos os outros recursos estáticos na detecção de malware. Os ataques baseados em privilégios são o resultado da violação de permissão pelos invasores.

Várias ferramentas de detecção estão disponíveis para salvar o sistema de ataques maliciosos, que analisam os aplicativos Android e os classificam como maliciosos ou benignos. Esta análise pode ser Estática,

ÿ Autor correspondente.
E-mail: bhatparnika@gmail.com (P. Bhat).

Revisão por pares sob responsabilidade da Universidade King Saud.

Produção e hospedagem pela Elsevier

https://doi.org/10.1016/j.jksuci.2021.11.004 1319-1578/

Por favor, cite este artigo como: P. Bhat e K. Dutta, Um modelo de seleção de recursos multicamadas para detecção de malware Android com base em discriminação de recursos e ganho de informações, Journal of King Saud University – Computer and Information Sciences, https://doi. org/10.1016/j.jksuci.2021.11.004

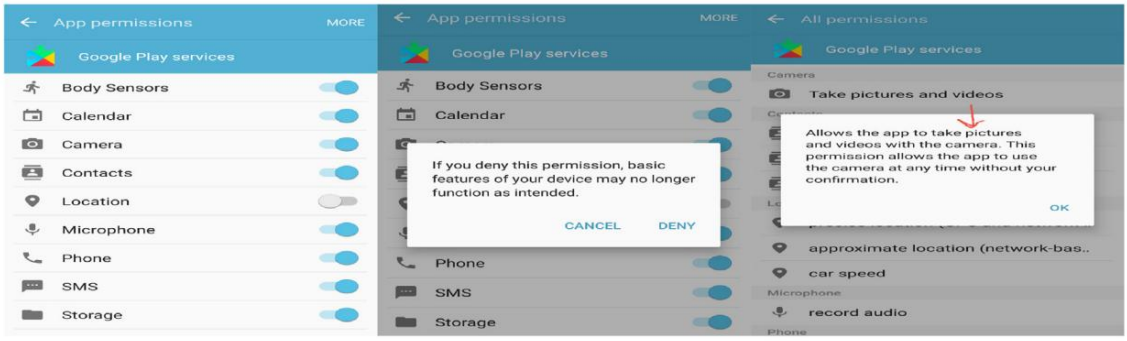

Fig. 1. Permissões de aplicativo padrão fornecidas no Android 6.0.1 (Marshmallow).

Dinâmico ou Híbrido. O pacote.apk contém manifesto, recurso, e arquivos executáveis Dalvik (dex). As ferramentas de análise estática extraem recursos como permissões, intenção, chamadas de API, componentes de hardware do manifesto e do arquivo de código java. Para análise dinâmica, o as ferramentas de detecção extraem recursos durante a execução do programa. Tráfego de rede, chamadas de sistema, CPU e uso de memória são características significativas usadas na análise dinâmica. Além disso, o híbrido A análise baseia-se em características estáticas e dinâmicas salientes (Bhat e Dutta, 2019).

Os pesquisadores preferem a detecção estática sobre a dinâmica como na estática abordagem, todas as partes estáticas do aplicativo estão disponíveis para serem examinado. Em contraste, a detecção dinâmica depende de vários fatores de tempo de execução, como uma entrada específica como um gatilho para demonstrar comportamento malicioso. Nesta situação, a abordagem de detecção estática fornece análise aprofundada e melhor precisão do que a análise dinâmica (Kapratwar et al., 2017; Li et al., 2019). Devido ao considerável aumento de dados maliciosos, a detecção manual tornou-se praticamente impossível. Além disso, muitos dos métodos de detecção de malware que os pesquisadores introduziram exigem um grande número de Recursos. Aumenta a complexidade de custo e tempo para executar o mecanismo de detecção em grandes bancos de dados. Assim, o aprendizado de máquina entrou em cena (Yerima et al., 2015). Algoritmos de aprendizado de máquina podem classificar dados com precisão em menos tempo.

1.1. Motivação

O número de recursos no arquivo Android é enorme para o número necessário de recursos para a classificação ideal. Usando também muitos recursos sobrecarregam desnecessariamente o sistema. Essa situação resulta em alto tempo de processamento e redução do desempenho geral do sistema. Os pesquisadores deram meticulosamente atenção para a criação de novos métodos de detecção de malware. No entanto, eles têm feito pouco trabalho no campo da descoberta de recursos para detecção de malware. É crucial reduzir recursos do conjunto de recursos antes de alimentá-los para qualquer máquina algoritmo de aprendizado para processamento posterior. A seleção de recursos é uma parte essencial das técnicas de detecção de malware, pois pode afetam positiva e negativamente o desempenho da técnica de detecção. Este artigo visa desenvolver um conjunto robusto de recursos contendo características relevantes. O conjunto de recursos filtrados pode fornecer análise aprofundada e resultam em um modelo de detecção altamente preciso e confiável. O trabalho proposto tenta descobrir conjuntos de recursos relevantes usando a redução automatizada de recursos e um teste manual abordagem.

1.2. Contribuição

A principal contribuição deste estudo é propor um recurso modelo de seleção que pode descobrir características relevantes de uma vasta piscina. O modelo é de duas partes, onde a primeira parte é característica redução, e o segundo é o teste e análise no selecionado recursos.

A princípio, a abordagem toma os recursos como entrada e os alimenta para uma fase de redução de recursos em várias camadas. No primeiro passo, o método proposto remove recursos com base em sua frequência, discriminação e, pontuações de Ganho de Informação. Muitos dos recursos não são úteis na classificação devido a vários motivos.

Uma razão crucial pode ser um recurso de baixa frequência, que não pode melhorar o resultado da ferramenta de classificação. Essas características são consideradas ruído, o que também pode impactar negativamente a precisão da ferramenta de classificação. Assim, a seleção de recursos torna-se um passo fundamental.

Além disso, para analisar o desempenho da abordagem, cinco abordagens de classificação de aprendizado de máquina foram usadas. Aleatório floresta apresentou a maior taxa de precisão de 96,28%. O método alimenta os recursos selecionados para várias técnicas de classificação de aprendizado de máquina usando várias combinações dos recursos.

Finalmente, a abordagem classifica conjuntos de recursos com base na precisão, Recall, Precision, F1-Score e outras métricas de desempenho. o recursos selecionados podem ser usados posteriormente junto com abordagens de classificação de detecção de malware para classificar e identificar se um aplicativo Android é um aplicativo malicioso ou não.

O restante do artigo é o seguinte: a Seção 2 discute a Trabalho relatado. A Seção 3 ilustra a abordagem proposta e a Seção 4 representa as métricas de avaliação, seguida pela Seção 5, apresentando resultados e análises experimentais. A seção 6 apresenta uma análise comparativa dos resultados. A Seção 7 discute brevemente o cenário atual e a direção de pesquisa futura e a Seção 8 conclui o trabalho.

2. Trabalho relacionado

A arquitetura do sistema operacional Android possui quatro camadas: (i) camada de hardware, (ii) camada de kernel, (iii) camada de abstração de hardware e, (iv) camada de aplicação. Cada camada se comunica por meio de chamadas de API. Pesquisadores e desenvolvedores tomaram muitas medidas de segurança para garantir a segurança dessas camadas e interfaces. Eles têm introduziu um mecanismo de sandbox para proteger os componentes. Lá são várias ferramentas antivírus e de detecção de malware disponíveis para proteger o sistema contra ataques conhecidos e desconhecidos. No entanto, os invasores continuam escavando as brechas de segurança presentes nesses camadas (Bhat e Dutta, 2019).

Pesquisadores realizaram muito trabalho em malware para Android pesquisa de detecção usando vários recursos. Sahs e Khan (2012) propôs uma abordagem de aprendizado de máquina para detecção de malware em Android em um ambiente estático. Era um modelo SVM de classe única treinados em amostras benignas. O modelo usou permissões internas e bytecode bruto apresentado como gráfico de fluxo de controle como recursos para

análise. O trabalho usa um conjunto de dados limitado e tendencioso de 2.081 arquivos benignos e 91 maliciosos. Os nós nos gráficos de fluxo de controle são rotulados com base na última instrução e são pequenos. Tais rótulos não capturam informações importantes presentes no código original.

PUMA, uma abordagem de detecção de malware baseada em permissão. O conjunto de dados inclui 1.800 aplicativos benignos coletados do Android Market não oficial e 249 amostras maliciosas da ferramenta Vir usTotal. Vários algoritmos de aprendizado de máquina são aplicados ao conjunto de dados para classificar se os aplicativos são maliciosos ou benignos. A abordagem alcançou a maior taxa de precisão de 86,41% usando a classificação Random Forest. Os resultados mostram uma taxa de falsos positivos muito alta com algoritmos de aprendizado de máquina. Além disso, o método conta com apenas um único conjunto de características para classificação (Sanz et al., 2013).

O Droid Detective, uma ferramenta de detecção de malware baseada em regras, alcançou uma taxa de precisão de 96%. Essas regras são um conjunto de combinações de Permissões extraídas usando o k-map. Qualquer desvio das regras geradas indica comportamento malicioso. A abordagem usa um conjunto de dados de 741 aplicativos benignos do Android Market e 1.260 aplicativos maliciosos do Malware Genome Project. O trabalho mostra como as combinações de permissões e comportamentos de aplicativos maliciosos estão relacionados. Os resultados mostram uma alta taxa de falso-positivos de 50,62% com (k)=1, onde k é a frequência de solicitação das Permissões (Liang e Du, 2014).

Sharma e Dash (2014) propuseram um método que usa permissão e chamadas de API como recursos. Para análise, o autor usa um conjunto de dados de 800 arquivos maliciosos do Android Malware Genome Project e 800 arquivos benignos de fontes oficiais e de terceiros.
A seleção de recursos baseada em correlação é aplicada para remover os recursos redundantes. Para avaliação, a abordagem utiliza classificadores Naive Bayes e KNN. KNN alcançou a maior taxa de precisão de 95,1%.

Verma e Muttoo (2016) propuseram uma abordagem de análise estática semelhante usando permissão e intenção como recursos estáticos. O conjunto de dados compreende 850 aplicativos benignos e 620 maliciosos, coletados do mercado de aplicativos oficiais Android, Google Play e contagio mini dump13. A próxima etapa aplica o Ganho de Informações ao conjunto de dados para seleção de recursos. Para a classificação a abordagem proposta utiliza agrupamento K-means e vários algoritmos de classificação.
A taxa de precisão mais alta para detectar arquivos maliciosos é de 94%, com um algoritmo de aprendizado de máquina J48 para classificação.

AndroDialysis usa Intenção e Permissões como recursos para detecção de malware. O conjunto de dados inclui 1.846 aplicativos benignos e 5.560 malignos coletados da loja Google Play e Dre bin. Para o método de classificação emprega-se a Rede Bayesiana com quatro algoritmos de busca LAGDHillClimber, Geneticsearch, HillClimber e K2. A abordagem alcança os melhores resultados com a combinação Bayesian Network e LAGDHillClimber. Os resultados mostram que o método atinge uma taxa de precisão de 91% com intenção, 83% com permissões e 95,5% com ambos os recursos (Feizollah et al., 2017).

Razak et al., 2018 usa o método Particle Swarm Optimization (PSO), Information Gain e Evolutionary para otimização de recursos. A abordagem usa 3.500 aplicativos benignos do Google Play e 5.000 aplicativos maliciosos de Drebin.

Além disso, o autor usou várias técnicas de aprendizado de máquina no conjunto de recursos resultante; Random Forest, AdaBoost, KNN, MLP e J48. Sua abordagem alcançou a maior taxa de Sensibilidade com o classificador AdaBoost e o Particle Swarm Optimization (PSO), 95,6%, que é menor que a nossa abordagem (97,92% com Random For est). Além disso, o classificador AdaBoost mostra uma alta taxa de falsos positivos de 32% e uma baixa pontuação F1 de 87,7%.

Outra abordagem proposta por Shang et al. (2018) usa Permissão como recursos e um algoritmo de classificação Naive Bayes para análise. Para o experimento, o método usa um conjunto de dados composto por 1725 aplicativos maliciosos e 945 benignos. Atribuição adicional weight(w) para os atributos, Permissões anteriormente desconhecidas atuam como fatores de ponderação para ponderação do algoritmo Naive Bayes. Esse peso representa a correlação entre a correlação de permissão conhecida e desconhecida. Na próxima etapa, o método calcula o coeficiente de correlação de Pearson para Permissões. A abordagem remove Permissões com um valor de coeficiente de correlação menor que um valor limite para melhorar a eficiência. O método atinge uma taxa de detecção de 86,54% com Naive Bayes e 97,59% com Naive Bayes ponderado.

Li et ai. (2019) usam permissões e chamadas de API como recursos estáticos. O conjunto de dados inclui 1.000 aplicativos maliciosos e benignos do VirusShare e da Google Store são coletados. A próxima etapa calcula a discriminação de recursos e seleciona recursos de alta discriminação e executa a mineração de padrões frequente. Usando Naive Bayes ponderados e padrões discriminantes, a abordagem detecta aplicativos maliciosos com uma taxa de precisão de 88,69%.

Guan et ai. (1634 (2020)), usa Permissões e Intenção como recursos para classificar os aplicativos. O conjunto de dados tem 1280 aplicativos maliciosos e 1200 benignos. A fonte de dados para arquivos maliciosos é AndroMalShare e MalGenome e arquivos não maliciosos são mercados de aplicativos Xiaomi. Para seleção de recursos, o método utiliza PSO e Ganho de Informação. A abordagem aplica quatro técnicas de aprendizado de máquina - J48, KNN, SVM e NB ao conjunto de dados para analisar o desempenho. Sem a seleção de recursos, o método atinge uma taxa de precisão de 95,2% com o classificador KNN e SVM usando Permissions e Intent como recursos. Usando Permissões como um conjunto de recursos e aplicando Ganho de Informação e KNN, a abordagem atinge uma taxa de Precisão de 96,1% e com PSO e SVM Precisão é de 96,6%.

Jung et ai. (2021), uma abordagem de análise estática usa chamadas de API e permissões como recursos para analisar os aplicativos maliciosos.
Os pesquisadores empregam conhecimento mínimo de domínio e métodos baseados em importância Gini para seleção de recursos. Para classificação, o trabalho aplica o algoritmo Random Forest em 26.276 aplicações maliciosas e 27.041 benignas. O conjunto de dados é coletado do AndroZoo e verificado usando a ferramenta VirusTotal. Random Forest mostra uma taxa de precisão de 96,51/%. Outra abordagem proposta por Sahin et al. (2021) aplica regressão linear para selecionar recursos baseados em permissão ideais. Vários modelos de aprendizado de máquina são usados para classificar o conjunto de dados composto por 1.000 aplicativos maliciosos e benignos cada. O modelo MLP (Multilayer Perceptron) mostra o melhor desempenho com uma pontuação F-measure de 96,1%.

Lacunas de
pesquisa No trabalho proposto, as seguintes lacunas de pesquisa são abordadas:

A pesquisa pré-existente se concentra em melhorar a precisão do sistema de detecção de malware, observando como a inclusão de todos os recursos afeta o custo computacional. As abordagens de aprendizado de máquina exigem recursos para classificar um determinado conjunto de dados (Crussell et al., 2012). Usar todos os recursos presentes no conjunto de dados levará ao desastre dimensional. Um grande número de características insignificantes e redundantes tornam o modelo difícil de interpretar.
A abordagem proposta aplica várias técnicas de redução de recursos para reduzir recursos irrelevantes e melhorar o desempenho computacional.

Será mais fácil para o desenvolvedor de malware evitar uma ferramenta de detecção que depende de um único conjunto de recursos para padrões e insights. Um conjunto de recursos composto por vários recursos será mais confiável.
O trabalho proposto considera o conjunto de recursos mais utilizado; Permissões, chamadas de API e intents para análise.
É de extrema importância descobrir os melhores conjuntos de recursos possíveis para melhorar o desempenho geral do sistema de detecção. A abordagem proposta usa as características estáticas mencionadas para o experimento. Após remover recursos irrelevantes na etapa de seleção de recursos, o método executa todas as combinações possíveis dos recursos resultantes em cinco aprendizado de máquina

abordagens. Os conjuntos de recursos são classificados com base nas métricas de desempenho. O foco principal do trabalho é obter um conjunto de recursos que forneça uma alta taxa de precisão e baixos alarmes falsos positivos.

O trabalho relacionado não discute o impacto do uso do conjunto de dados tendencioso, onde o número de arquivos maliciosos é menor ou maior que os arquivos benignos. O trabalho proposto analisa o desempenho do sistema de detecção para conjuntos de dados tendenciosos e não tendenciosos.

A seção a seguir discute o trabalho proposto em detalhes.

3. Abordagem proposta

Como pode ser visto no fluxograma da Fig. 2, o trabalho proposto compreende duas fases; redução de recursos seguida de classificação de malware usando algoritmos de aprendizado de máquina bem definidos.
A técnica de redução de recursos remove os recursos que são muito infrequentes para ter qualquer impacto na classificação. Em segundo lugar, a abordagem de discriminação de recursos reduz o número de conjuntos de recursos. Além disso, o Ganho de Informação é aplicado aos recursos reduzidos para obter um conjunto de vinte e um recursos. Posteriormente, esta seção discute esses processos em detalhes. Além disso, conforme mostra o fluxograma, na segunda fase, o método aplica cinco algoritmos de aprendizado de máquina listados abaixo em todas as combinações possíveis do conjunto de recursos obtido na etapa anterior para encontrar o melhor conjunto de recursos junto com um algoritmo de classificação.

1. Árvore de Decisão (DT)
2. Regressão Logística (LR)
3. Máquina de vetores de suporte (SVM)
4. Floresta Aleatória (RF)
5. Naive Bayes (NB)

3.1. Conjunto de dados usado

O conjunto de dados consiste em arquivos maliciosos e benignos.apk. O malware é adquirido de fontes de terceiros; Virustotal (Virustotal, 2019), VirusShare (VirusShare, 2019) e Drebin (Arp et al., 2014) e os aplicativos benignos da Play Store. Além disso, os aplicativos foram verificados usando uma ferramenta online VirusTotal. A principal ênfase durante a preparação do conjunto de dados foi garantir a diversidade de dados.

Para análise, usamos o conjunto de dados (D) composto por 11; 449 aplicações. Existem 5.279 aplicativos maliciosos e 6.170 aplicativos não maliciosos. Para o experimento, usamos quase metade das aplicações malignas (2640) e benignas (3085) para treinamento (Dt) e o restante para fins de teste. Para analisar o desempenho da abordagem para o conjunto de dados enviesado e não enviesado, dividimos-o em três subconjuntos, conforme mostrado na Tabela 1. O tamanho da amostra é selecionado de forma que haja subconjuntos tendenciosos e não enviesados do conjunto de dados. Db é um conjunto de dados tendencioso, onde o número de arquivos maliciosos (2640) é maior do que arquivos não maliciosos (1543). O número de arquivos não maliciosos é metade do número total de arquivos benignos em (Dt).
Da mesma forma, Dc consiste em um grande número de aplicativos não maliciosos. O número de arquivos maliciosos (1320) é metade do número de arquivos maliciosos em (Dt). Em Da, o número de arquivos maliciosos (1320) e não maliciosos (1543) é quase equivalente. Todos os arquivos.apk são selecionados aleatoriamente. A próxima subseção discutirá os recursos utilizados no trabalho proposto.

3.2. Recursos usados

Os aplicativos Android incluem muitos recursos, como permissão, chamadas de API, filtro de intenção, endereço de rede e componentes de hardware. De acordo com o estado da arte, as permissões do Android são as

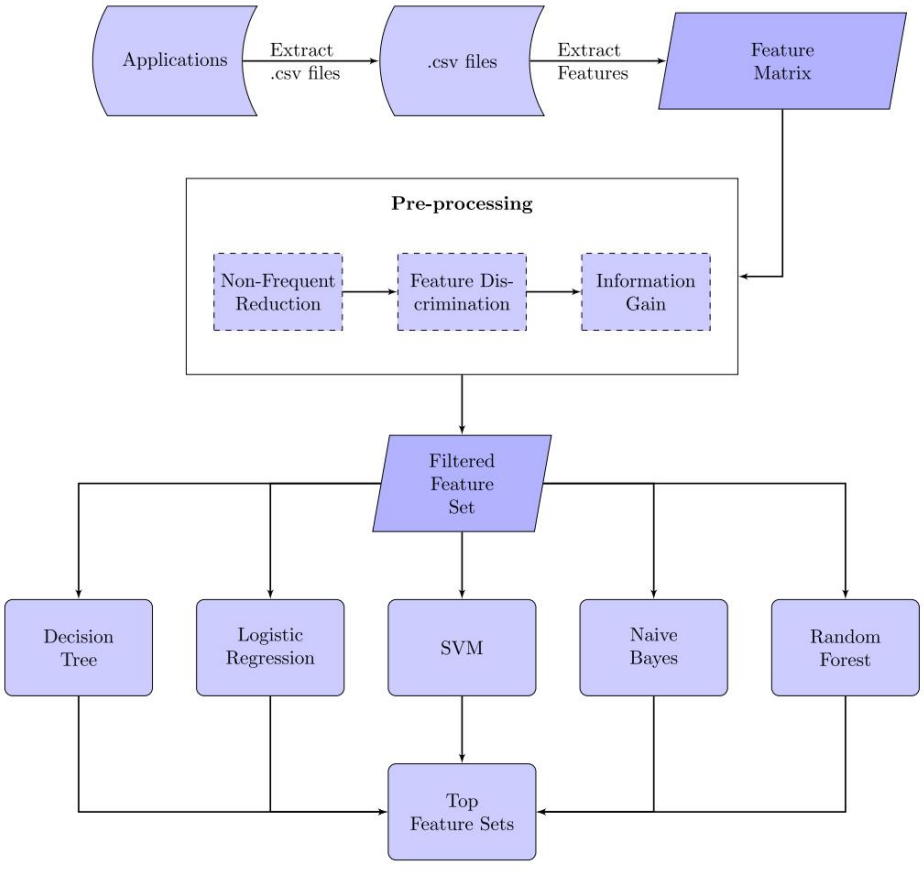

Fig. 2. Fluxograma da abordagem proposta.

tabela 1

Subconjuntos do conjunto de dados.

S. Não.	Conjunto de dados	Malicioso(M)	Benigno(B)
	Da	1320	1543
1	Db	2640	1543
2 3	CC	1320	3085

recurso mais usado entre os recursos estáticos para detecção de malware. O segundo recurso mais utilizado é o código java, que consiste em chamadas de API. Outro conjunto de recursos usado neste estudo é Intent (Feizollah et al., 2015; Feizollah et al., 2017). O método proposto usa esses três conjuntos de recursos para detecção de malware no ambiente estático meio Ambiente.

Permissões

O Android é um sistema operacional baseado em Linux, portanto, para obter acesso a qualquer coisa no sistema, os aplicativos precisam mencionar Permissão para acessar esse componente no arquivo de manifesto separadamente. Três categorias de As permissões do Android dadas aos aplicativos são tempo de instalação, tempo de execução e permissões especiais. Essas permissões definem o escopo de dados acessados e as ações permitidas para realizar nesses dados. Um aplicativo que solicita mais do que o número mínimo de Permissions necessário indica atividade maliciosa (A. Developer, 2021).

Chamadas de API

Chamadas de API são um conjunto de regras usadas para chamar funções presentes em um pacote já instalado. Os pesquisadores usam esse recurso para análise comportamental de aplicativos Android observando o padrão de chamadas da API. Esta informação funciona como uma assinatura, pois é único para cada aplicação. Qualquer desvio dos padrões regulares pode indicar atividade maliciosa. Pesquisadores com destaque usar chamadas de API como recursos para a classificação de aplicativos Android (Fernando, 2016).

Intenção

Outra característica importante é o Intent, que os pesquisadores usam para análise de malware. Desenvolvedores Android usam intents para se comunicar entre os componentes do aplicativo. Seu principal uso é lançar atividades associadas ao método. Cada Intent carrega um par de informações, a) ação, b) dados sobre os quais deve agir. AÇÃO_ VER, ACTION_ DIAL, ACTION_ EDIT são alguns exemplos de dados de ação pares (A. Team, Intents, Android Developer, 2019). Assim, as intenções pode fornecer informações sobre as ações que o aplicativo está vai realizar. Os desenvolvedores de aplicativos fornecem detalhes da intenção que o aplicativo receberia no arquivo de manifesto. Ao analisar o padrão desses Intents, é possível descobrir formulários.

3.3. Extração de recursos

Usamos uma ferramenta de código aberto (AndroPyTool et al., 2019) para converter arquivos Android em arquivos JavaScript Object Notation (JSON) e um código python extraiu recursos de arquivos JSON para CSV arquivos. O arquivo JSON representa os campos de um objeto como pares de valores-chave. Cada par mostra se um determinado recurso está presente ou não em a aplicação (JSON, 1999). A ferramenta então pega esses arquivos CSV como entrada para realizar a redução de recursos. O primeiro passo é desmontar

$$\begin{pmatrix} & F_1 & F_2 & F_3 & F_4 & F_5 & F_6... & F_N \\ Apk_1 & 11 & 15 & 19 & 6 & 0 & 8... & 20 \\ Apk_2 & 21 & 13 & 12 & 18 & 9 & 78... & 22 \\ Apk_3 & 22 & 11 & 30 & 12 & 16 & 12... & 14 \\ . & & & & & & & \\ . & & & & & & & \\ Apk_M & 0 & 9 & 22 & 15 & 12 & 34... & 13 \end{pmatrix}$$

Fig. 3. Uma instância de uma matriz de recursos.

ble os arquivos.apk para extrair recursos estáticos deles. A abordagem armazena os recursos extraídos na matriz de recursos Fig. 3. Recurso matriz consiste nos aplicativos como linhas e recursos como colunas. O valor da célula denota a ocorrência do recurso no respectiva aplicação.

3.4. Redução de recursos

A primeira fase da abordagem proposta consiste em uma abordagem de redução de recursos em três etapas para filtrar recursos não informativos conjuntos.

A primeira etapa remove os recursos cuja frequência é menor que um valor limite mínimo de 0,8 3.4.1. O valor limite é selecionado usando a abordagem de força bruta. As características menos frequentes são ruídos e afetam a classificação. O próximo passo calcula a pontuação de discriminação de característica. Esta etapa remove os recursos que são quase iguais em uma proporção de arquivos maliciosos e benignos. Esses recursos não ajudam na classificação de arquivos de aplicativos entre dois classes (benignas e malignas) 3.4.2. Para melhorar ainda mais o desempenho da classificação, a abordagem aplica Ganho de Informação para remover recursos menos informativos 3.4.3.

A redução gradual de recursos remove estrategicamente recursos redundantes e insignificantes. E também aumenta o processamento tempo e reduz a eficiência do classificador. Os recursos resultantes são usados juntamente com abordagens de aprendizado de máquina para enquadrar uma abordagem eficiente de detecção de malware.

3.4.1. Filtrando recursos não frequentes

Na primeira etapa, a abordagem filtra características presentes em um número insignificante em aplicativos maliciosos e benignos. Esses recursos são insignificantes e não terão nenhum efeito positivo na técnica de detecção devido à sua baixa frequência.

P, I e A são três conjuntos de recursos onde 1:

P ¼ Todas as Permissões no conjunto de dados D

I ¼ Todos os intents no conjunto de dados D ð1Þ

A ¼ Todas as chamadas de API no conjunto de dados D

O conjunto de dados D representa todos os aplicativos Android. P0 , e A0 são o conjunto de características resultante 2 3 consistindo de características cuja frequência é maior que o valor mínimo de limiar (Minð Þ thresh) selecionado.

P0P ; EU EU; A0 A ð2Þ

P0 [eu [A0 # P [I [A ð3Þ

F representa o conjunto de itens de recurso 4. Todos os elementos em F têm fre frequência maior que Minð Þ thresh .

F ¼ ff 1; f 2; f 3; :::f gn ; fi 2 P0 [I [A0 ð4Þ

Um valor limite ideal de 0:8 é selecionado. Esta etapa filtrada os recursos que não estão presentes em pelo menos 20% dos ou banco de dados benigno.

3.4.2. Discriminação de recursos

Esta etapa aplica a discriminação de recursos (Li et al., 2019) método para o conjunto de recursos resultante P0 , e A0 para filtrar a fea turas que são indiscriminadas entre as duas classes, ou seja, malware e benigno. Isso significa que este método irá pontuar os recursos com base em seu padrão de distribuição entre aplicativos maliciosos e benignos. Este método dá uma pontuação alta para o recurso apresentar mais ou menos em uma das classes e uma pontuação baixa para apresentar com um pouco presença igual em ambos. Uma vez que características com distribuição igual em ambas as aulas não serão muito informativas para a técnica de detecção. A fórmula para calcular a pontuação de discriminação de características para cada recurso fi é:(5), (6)

$$\text{Pontuação}\ f\ \eth\ Þ¼i\ 1\ \frac{\min fib; f\ \eth\ Þ\ im}{\text{fibra máxima}\ ; f\ \eth\ Þ\ im} \quad \eth 5Þ$$

Onde,

$$fib\ ¼\ \frac{\text{número total de arquivos benignos com recurso fi}}{\text{número total de arquivos benignos}}$$

$$fim\ ¼\ \frac{\text{número total de arquivos maliciosos com recurso fi}}{\text{número total de arquivos de malware}} \quad \eth 6Þ$$

fib representa a frequência do recurso fi em arquivos benignos e, fim representa a frequência do recurso fi em arquivos maliciosos

Escore fi ð Þ2 fg 0; 1

Quando:

Score(fi) = 0 {igual frequência de ocorrência em ambas as classes; sem discriminação}

Escore(fi) 0 {baixa frequência de ocorrência em qualquer um dos Aulas; pior característica discriminante}

Escore(fi) 1 {alta frequência de ocorrência em qualquer um dos Aulas; melhor característica discriminativa}

Calcule o grau de discriminação ou pontuação para cada recurso fi, os elementos com a pontuação mais alta de cada conjunto de recursos P0 ; eu 0, e A0 , são selecionados. Finalmente, os elementos no recurso resultante definir P00 ; eu 00 e A00 7 têm alta frequência de ocorrência em ambos das duas classes (benigna ou maliciosa).

P00 [eu 00 [A00 # P0 [I 0 [A0 ð7Þ

Essa etapa resultou em uma redução de quase 80% dos recursos em relação aos números iniciais.

3.4.3. Ganho de informações

A última técnica de redução de recursos utilizada é o Ganho de Informação. Isto é um método de filtragem que filtra as características com base nas informações de classificação que possui. O ganho de informação também é conhecido como uma medida de seleção variável. Ele fornece atributos (recursos) que melhor discrimina entre duas classes (maliciosa e benigna) em um determinado conjunto de dados de treinamento.

A informação aqui é entropia. Ganho de Informação, também conhecido como divergência de Kullback-Leibler ou entropia relativa, é a quantidade de informações estatísticas adquiridas sobre uma variável alvo de um variável preditora. Ganho de informações é um método de seleção de recursos usado para remover recursos irrelevantes. É independente do algoritmo de aprendizado de máquina usado e precisa de menos tempo para calcular os valores das variáveis. Ele desempenha um papel fundamental na melhoria do desempenho de classificação de algoritmos de aprendizado de máquina. o abordagem ajuda a filtrar o redundante, bem como irrelevante vetores de características do conjunto de dados (Quinlan, 1986).

A abordagem calcula o Ganho de Informação para os restantes ele I 00 e A00. Uma mentos no conjunto resultante P00 ; pontuação mais alta indica a característica é informativa e, uma pontuação mais baixa mostra que é menos informativa, portanto, irrelevante para a classificação.

O valor de Ganho de Informação (IG) para um recurso é calculado como (8)

IG fi ð Þ¼ E Ismalware ð Þ E Ismalware; fi ðÞ ð8Þ

Maior valor de IG(fi) indica que fi é de extrema importância para classificação.

E(Ismalware)- A entropia do Ismalware (classe) é calculada como (9):

E Ismalware ð Þ ¼ Xc i¼1 pilog2pi ð9Þ

onde c = número total de classes, neste caso c = 2 (benigno e mal ware) &

pi= Probabilidade de ocorrência da classe.

E(Ismalware,fi)- Entropia condicional do Ismalware (classe) dado recurso fi é calculado como (10):

E Ismalware; f ð Þ¼ p fi E Ismalware j fi þ p fi E Ismalware j fi ð10Þ

O valor de IG varia de 0 a 0:5.

A abordagem seleciona sete recursos com a maior pontuação de ganho de cada tipo de conjunto de recursos. A Tabela 2 fornece uma lista dos vinte e um principais recursos. Este conjunto de recursos 11 consiste nos melhores recursos de um conjunto diversificado de recursos. A combinação de vários tipos de recursos fornecerão uma análise aprofundada e confiável.

P000 [I 000 [A000 # P00 [I 00 [A00 ð11Þ

A primeira fase filtra 21 melhores características de 21; 149 recursos usando três métodos de redução de recursos. Na próxima seção análise dos resultados e o conjunto de recursos mais significativo são apresentado.

4. Métricas de Avaliação

Para avaliar o desempenho da abordagem a seguir, são usadas métricas de avaliação (Bhat e Dutta, 2021):

Precisão 12 representa a proporção de aplicativos classificados corretamente para o número total de aplicativos benignos e maliciosos presentes no conjunto de dados de teste.

$$Um\ ¼\ \frac{TP\ þ\ TN}{TP\ þ\ FP\ þ\ TN\ þ\ FN} \quad \eth 12Þ$$

Recall(Sensitivity)13 também conhecido como taxa de detecção, representa a proporção de aplicativos maliciosos classificados corretamente para o total número de aplicativos maliciosos presentes no conjunto de dados de teste (Anael Beaugnon, 2018; Shrivastav, 2020).

$$R\ ¼\ \frac{TP}{TP\ þ\ FN} \quad \eth 13Þ$$

A precisão 14 representa a proporção de arquivos maliciosos reais detectado para o número total de aplicativos previstos como malignos pela abordagem proposta.

$$P\ ¼\ \frac{TP}{TP\ þ\ FP} \quad \eth 14Þ$$

F1-Score 15 representa a média harmônica de Precisão e Lembrar.

$$FS\ ¼\ \frac{2\ PR}{P\ þ\ R} \quad \eth 15Þ$$

Especificidade (TNR) 16 representa a razão de classificados corretamente aplicações benignas.

$$\text{Especificidade TNR}\ \eth\ Þ¼\ \frac{TN}{TN\ þ\ FP} \quad \eth 16Þ$$

A taxa de falsos negativos 17 representa a proporção de aplicativos maliciosos classificados erroneamente como benignos em relação ao número total de aplicativos maliciosos aplicativos no conjunto de dados de teste.

$$\text{Taxa de Falso Negativo FNR}\ \eth\ Þ¼\ \frac{FN}{TP\ þ\ FN} \quad \eth 17Þ$$

A taxa de falsos positivos 18 representa a proporção de aplicativos benignos classificados erroneamente como maliciosos em relação ao número total de aplicativos benignos aplicativos no conjunto de dados de teste.

$$\text{Taxa de falsos positivos FPR}\ \eth\ Þ¼\ \frac{PF}{FP\ þ\ TN} \quad \eth 18Þ$$

O coeficiente de correlação de Matthews (MCC) 19 é a melhor medida para avaliar o desempenho quando o conjunto de dados de teste for desigual. O valor de MCC varia entre (þ1; 1).

mesa 2

Os 21 principais recursos com maior pontuação de Ganho de Informação (IG).

N	Características	Pontuação do IG	Abreviação
	android.permission.SEND_ SMS	0,359	P1
1 2	android.permission.RECEIVE_ SMS	0,321	P2
3	android.permission.READ_ SMS	0,297	P3
4	android.permission.READ_ PHONE_ STATE	0,267	P4
5	android.permission.WRITE_ SMS	0,233	P5
6	com.google.android.c2dm.permission.RECEIVE	0,163	P6
7	android.permission.MOUNT_ UNMOUNT com. _ SISTEMAS DE ARQUIVOS	0,155	P7
8	android.vending.INSTALL_ REFERRER	0,251	I1
9	com.google.firebase.INSTANCE _ ID _ EVENTO	0,237	I2
10	com.google.android.c2dm.intent.RECEIVE	0,201	I3
11	android.provider.Telephony.SMS _ RECEIVED	0,196	I4
12	android.intent.action.USER com.google.[illegible]ESSAGING_	0,144	I5
13	EVENT android.provider.Telephony.SMS_ DELIVER	0,116	I6
14	java.util.TimeZone.getAvailableIDs	0,106	I7
15	android.view.ViewGroup.indexOfChild	0,519	A1
16	android.content.pm.PackageManager.getPackageInstaller	0,517	A2
17	android.os.ParcelFileDescriptor java.nio.BufferOverflowException	0,507	A3
18	android.view.View.getDisplay android. support.v7.view.ActionMode.getTitle	0,498	A4
19		0,495	A5
20		0,495	A6
21		0,490	A7

Onde, þ1= previsão perfeita.

0 = previsão aleatória

1 = desacordo entre observação e previsão

(Mateus, 1975)

Coeficiente de correlação de Matthews MCC ð Þ¼

$$\frac{\delta TP\ TN\ \eth\ Þ\ FP\ FN}{p\ \delta TP\ þ\ FP\ðTP\ þ\ FN\ \eth Þ\ TN\ þ\ FP\ \eth\ Þ\ TN\ þ\ FN} \quad \eth 19Þ$$

Onde,

True Positive(TP)-Representa o número de aplicativos mal-intencionados corretamente classificados como mal-intencionados.

True Negative(TN)-Representa o número de aplicações benignas corretamente classificadas como benignas.

Falso Positivo(FP)-Representa o número de aplicativos benignos classificados incorretamente como maliciosos.

Falso Negativo(FN)-Representa o número de aplicativos maliciosos classificados incorretamente como benignos.

A Tabela 3 representa a matriz de confusão para cada aprendizado de máquina algoritmo de processamento no caso de D.

5. Resultados e Análise

Esta seção discute os resultados obtidos usando o recursos com várias abordagens de classificação de aprendizado de máquina para descobrir os principais conjuntos de recursos. O método leva todos os 21 recursos chegou na etapa anterior e executa todas as combinações possíveis de 10 recursos em cinco abordagens de aprendizado de máquina.

O hardware usado para realizar este experimento é um processador Core i5 com 8 GB de RAM e um disco rígido de 512 GB de capacidade de armazenamento. Considerando os recursos de hardware limitados e como 21C10 resulta em 3.52716 combinações, a figura de 10 características é selecionada para que o número não é nem muito baixo nem muito alto. Em experimentos futuros pode ser feito usando o framework proposto com um número diferente de recursos. As Tabelas 4–7 mostram os resultados para todos os subconjuntos.

5.1. Árvore de decisão

A Árvore de Decisão é um método de classificação supervisionado. Ele divide os dados em partições usando particionamento recursivo binário. No árvore de saída, os nós representam o teste de classificação e o ramos representam o resultado deste teste. Essas ramificações se conectam ao próximo nó ou ao nó folha que representa o resultado de classificação (Pedregosa et al., 2011; Sharma e Kumar, 2016). A Árvore de Decisão mostra uma taxa de Precisão de 96,28% e Recall de 97,40% avaliar. O valor de FNR e FPR é de 3,86% e 12,24%, respectivamente.

O tempo de execução da Árvore de Decisão para classificar e encontrar o melhor combinação possível de conjuntos de recursos foi de 90 s.

5.2. Regressão Logística

O terceiro modelo de classificação utilizado neste estudo é o Logístico. Regressão. É um método de classificação supervisionado usado para análise preditiva que usa uma equação de regressão linear para fornecer uma saída binária discreta (Karthi et al., 2015). A regressão logística abordagem de classificação resultou em 95,30% de precisão e 97,92% Taxa de recall. Além disso, o método atinge um FNR de 5,82% e 13,82% Taxa FPR. O tempo de execução da Regressão Logística foi de 734 s.

5.3. Máquina de vetor de suporte

O segundo modelo de classificação utilizado neste artigo é o SVM. SVM é um método de classificação supervisionada amplamente utilizado em malware pesquisa de detecção.

O SVM baseia-se na teoria da aprendizagem estatística (Evgeniou e Pontil, 2001). A SVM conseguiu atingir uma taxa de precisão de 95,30% e Taxa de recall de 97,08%. Além disso, o valor de FNR é de 5,30% e, FPR é 10,83%. Levou 3.111 s para o SVM classificar e encontrar o melhor conjunto de características.

5.4. Floresta Aleatória

Outro modelo de classificação utilizado neste estudo é o Random Forest est. Leo Breiman propôs a Random Forest na década de 2000 ''para construir um conjunto preditor com um conjunto de árvores de decisão" (Biau, 2012). Ele usa subespaços de dados selecionados aleatoriamente do conjunto de dados para aumentar as Árvores de Decisão, e a previsão final é obtida por agregando sobre o conjunto. A classificação Random Forest A abordagem mostra uma precisão de 96,28%, recall de 97,92%, FNR de 3,47%, e taxa de FPR de 12,74% com um número ótimo de 100 árvores. O tempo de execução foi de 454 s.

5.5. Baías ingénuas

O classificador Naive Bayes é o modelo de classificação final utilizado no estudo. É um modelo simplificado de probabilidade bayesiana baseado na "assunção de que as características são independentes de cada

Tabela 3

Matriz de confusão para cada algoritmo de aprendizado de máquina (D).

Actual

Predicted	Positive	Negative	Total
Positive	2717	367	3084
Negative	109	2632	2741
Total	2826	2999	*N*

Decision Tree

Actual

Predicted	Positive	Negative	Total
Positive	2770	314	3084
Negative	155	2586	2741
Total	2925	2900	*N*

Support Vector Machine

Actual

Predicted	Positive	Negative	Total
Positive	2671	413	3084
Negative	165	2576	2741
Total	2836	2989	*N*

Logistic Regression

Actual

Predicted	Positive	Negative	Total
Positive	2698	386	3084
Negative	97	2644	2741
Total	2795	3030	*N*

Random Forest

Actual

Predicted	Positive	Negative	Total
Positive	2715	369	3084
Negative	183	2558	2741
Total	2898	2927	*N*

Naive Bayes

Tabela 4

Comparação de desempenho de vários algoritmos de aprendizado de máquina usando D.

ML	Precisão	Lembrar	Precisão	Pontuação F1	TNR	FNR	FPR	MCC
TD	96,18%	97,40%	89,20%	93,12%	87,76%	3,86%	12,24%	0,840
LR	95,30%	97,92%	88,75%	93,11%	86,18%	5,82%	13,82%	0,804
SVM	95,30%	97,08%	89,53%	93,15%	89,17%	5,30%	10,83%	0,840
RF	96,28%	97,92%	89,40%	93,47%	87,26%	3,47%	12,74%	0,838
NB	95,30%	95,91%	90,06%	92,89%	87,39%	6,31%	12,61%	0,812

Tabela 5

Comparação de desempenho de vários algoritmos de aprendizado de máquina usando Da.

ML	Precisão	Lembrar	Precisão	Pontuação F1	TNR	FNR	FPR	MCC
TD	96,19%	97,72%	94,22%	95,94%	93,04%	4,84%	6,9%	0,882
LR	95,14%	97,95%	92,8%	95,31%	91,30%	5,32%	8,70%	0,861
SVM	95,4%	96,8%	96,8%	95,73%	93,31%	5,30%	5,30%	0,881
RF	96,15%	98,02%	93,21%	95,56%	91,32%	4,85%	6,68%	0,885
NB	95,21%	96,66%	93,53%	95,07%	92,21%	5,71%	7,09%	0,865

Tabela 6

Comparação de desempenho de vários algoritmos de aprendizado de máquina usando Db.

ML	Precisão	Lembrar	Precisão	Pontuação F1	TNR	FNR	FPR	MCC
TD	95,40%	94,23%	93,26%	93,74%	96,08%	4,11%	3,92%	0,901
LR	93,97%	94,53%	93,00%	94,18%	95,84%	5,81%	4,16%	0,871
SVM	95,45%	93,44%	94,29%	93,87%	96,64%	5,56%	3,36%	0,902
RF	94,10%	96,10%	95,10%	93,72%	96,40%	3,50%	3,84%	0,906
NB	93,53%	93,07%	94,42%	91,51%	96,60%	6,50%	3,39%	0,864

outro", implicando que a probabilidade de ocorrência de um dos características é independente da probabilidade de ocorrência de outras características (Mukherjee e Sharma, 2012). O Naive Bayes é conhecido para dar bons resultados em situações práticas. O método de classificação Naive Bayes alcançou uma taxa de Precisão e Recall de 95,30% e 95,91%, respectivamente. Além disso, as taxas de FNR e FPR são de 6,31% e 12,61%. Levou Naive Bayes 89 s para completar a execução.

Além disso, usamos um conjunto de dados de malware SMS (Dsms) (Mahdavifar et al., 2020) para analisar o desempenho do abordagem para categorias específicas de malware. É um serviço público conjunto de dados (CICMalDroid2020) com 3904 malware SMS e 1795 arquivos benignos. A Tabela 8 mostra que o método proposto alcançou a maior taxa de precisão de 95,02% e valor MCC de 0,900 com Floresta Aleatória.

Tabela 7
Comparação de desempenho de vários algoritmos de aprendizado de máquina usando Dc.

ML	Precisão	Lembrar	Precisão	Pontuação F1	TNR	FNR	FPR	MCC
TD	93,96%	97,51%	93,96%	95,70%	86,06%	2,49%	11,94%	0,857
LR	93,77%	96,81%	94,42%	95,60%	85,75%	3,19%	13,25%	0,850
SVM	95,43%	95,80%	95,25%	95,73%	88,03%	2,85%	10,20%	0,860
RF	94,10%	97,45%	94,22%	95,81%	86,56%	2,55%	13,44%	0,859
NB	93,22%	97,16%	93,25%	95,17%	86,55%	2,85%	13,45%	0,840

A taxa de Precisão, Recall, Precisão e Especificidade (TNR) alcançado em todos os casos é alto. Além disso, o falso-negativo e o a taxa de falsos positivos é baixa. O tempo de execução inclui o tempo tomadas para executar todas as combinações possíveis também. Naive bayes se saem melhor em termos de complexidade de tempo. O tempo gasto pelo SVM é alta em comparação com outras abordagens de classificação. A execução tempo de Floresta Aleatória é alto em comparação com Regressão Logística e inferior à Árvore de Decisão. A Random Forest supera tem uma taxa de precisão mais alta.

O valor de MCC é próximo de 1 em todos os casos. Implica que o abordagem funciona bem mesmo quando o conjunto de dados está desequilibrado, por exemplo, Db e Dc. Da representa um conjunto de dados quase balanceado, que é um caso ideal. Além disso, este trabalho discute os resultados da conjunto de dados D em detalhes.

A Tabela 9, representa algumas das características que são comuns para múltiplas abordagens mostrando sua importância. Os resultados também mostram que alguns recursos são onipresentes e são significativos para detecção de malware. A Tabela 10 representa o melhor conjunto de recursos com cinco abordagens; por isso, denominamos essas características como MIFs (Recursos Mais Importantes). Eles podem ser usados com outras abordagens de classificação para obter uma abordagem de detecção de malware altamente precisa e confiável.

Uma abordagem independente de plataforma será muito importante uma vez que os ataques maliciosos não se limitam ao sistema operativo Android. Os pesquisadores estão trabalhando na área de detecção de malware em diferentes sistemas operacionais. Haddadpajouh et al. (2018), Cimitile et ai. (2017) aplicaram Ganho de Informação para recurso algoritmos de seleção e aprendizado de máquina para detecção de malware para Mac OS e iOS. No futuro, os pesquisadores podem usar a proposta abordagem de seleção de recursos em várias camadas em diferentes plataformas para melhorar a detecção de malware e analisar o desempenho.

6. Análise Comparativa

As Fig. 4 e 5 mostram a análise comparativa dos resultados obtidos usando vários conjuntos de recursos para todos os subconjuntos do conjunto de dados. Melhor A precisão foi alcançada em 96,28% (D) usando a classificação Random Forest. Uma das principais razões pelas quais a Random Forest produz produção estável e resultados precisos é que é uma abordagem de conjunto que mescla as previsões de Árvores de Decisão correlacionadas. O método gera várias Árvores de Decisão correlacionadas e aplica o método de ensacamento para treinamento. Além disso, um esquema de votação por maioria amalgama o previsão de árvores individuais para gerar a saída final (Donges, 2021).

A Floresta Aleatória e a Regressão Logística forneceram os maiores Recall em 97,92% (D). Além disso, Naive Bayes alcançou o maior Pré pontuação de cisão em 90% (D). As abordagens de classificação mostram resultados para Db quando arquivos maliciosos são mais do que arquivos benignos.

As Fig. 6 e 7 representam a análise comparativa entre D e Dsms. O valor de Precisão e Recall com Random Forest é maior para o conjunto de dados D, enquanto o valor de MCC é quase igual a 1 para Dsms. Além disso, o valor do F1-Score para Dsms é de 95,21%, que é moderadamente maior que D(93,47%) com Random Forest.

Os resultados gerais mostram que a abordagem funciona bem para todos os casos, ou seja, quando há um conjunto de dados tendencioso e até mesmo para um determinado classe de malware (Dsms).

A Tabela 11 fornece uma análise comparativa do desempenho com características descobertas neste trabalho e trabalhos relacionados. O proposto O Optimal Static Feature Set (OSFS) pode detectar malware com maior Precisão. Esses resultados justificam que o OSFS descoberto usando o abordagem proposta pode ser usada na detecção de malware para melhorar seu desempenho.

7. Discussão e Trabalho Futuro

Esta seção discute brevemente as ameaças emergentes contra o Plataforma Android e direção de pesquisa futura para melhorar o desempenho do mecanismo de detecção Android existente.

A rápida evolução das famílias de malware de dia zero tornou-se um problema de segurança desafiador para Android. O mecanismo de detecção precisa analisar o código cibernético e extrair informações para proteger o sistema de ataques. Coulter et ai. (2020) propõem uma estrutura de segurança cibernética orientada a dados (DDCS) para analisar o código cibernético. O trabalho fornece uma diretriz para melhorar o aprendizado de máquina algoritmos para proteger o sistema de ataques como habilidades de vulnerabilidade de dia zero. O primeiro componente do framework é o processamento de dados. O artigo enfatiza como um conjunto de dados equilibrado e bem definido afeta o desempenho dos algoritmos de aprendizado de máquina. Nosso abordagem usa um conjunto de dados bem definido e rotulado, Drebin. A correção do conjunto de dados também é verificada usando a ferramenta VirusTotal, como sugerido no jornal. Além disso, o modelo de redução de recursos proposto elimina dados desnecessários e irrelevantes. O segundo componente é a engenharia de recursos, que se concentra no uso de recursos essenciais para classificação e representação de recursos. Em nossa abordagem, nós use três recursos, permissão, chamadas de API e intenção para análise. Uma matriz de recursos representa a ocorrência desses recursos em as aplicações.

No futuro, usaremos técnicas como Abstract Syntax Tree (AST) (Lin et al., 2017) e Extreme Programming (Holzinger et al., 2005) para representar programas. Tais métodos ajudarão para analisar código e detectar ataques de dia zero. O terceiro componente do framework é a modelagem cibernética que discute várias métricas

Tabela 8
Comparação de desempenho de vários algoritmos de aprendizado de máquina usando Dsms.

ML	Precisão	Lembrar	Precisão	Pontuação F1	TNR	FNR	FPR	MCC
TD	94,08%	95,92%	92,45%	94,15%	92,26%	7,74%	4,08%	0,882
LR	93,15%	92,58%	93,26%	93,41%	93,77%	6,23%	7,42%	0,862
SVM	94,08%	93,47%	95,17%	94,31%	94,75%	5,25%	6,53%	0,881
RF	95,02%	94,36%	96,07%	95,21%	95,74%	4,26%	5,64%	0,900
NB	92,09%	91,44%	93,15%	92,28%	92,79%	7,21%	8,56%	0,841

Tabela 9

Os 10 principais recursos com cinco algoritmos de aprendizado de máquina (D).

ML	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
TD	P2	P3	P4	P7	I8	I3	A1	A4	A6	A7
LR	P1	P2	P4	P7	I2	A2	A3	A5	A6	A7
SVM	P1	P2	P4	P7	I1	I2	I7	A1	A6	A7
RF	P2	P4	P6	P7	I1	I5	A1	A4	A6	A7
NB	P1	P2	P4	P6	I1	I3	A1	A4	A6	A7

Tabela 10

Recursos mais importantes (MIFs) (D).

SNenhum recurso		Abreviação
1	android.permission.RECEIVE_ SMS	P2
2	android.permission.READ_ PHONE_ STATE	P4
3	android.permission.MOUNT_ UNMOUNT_ FILESYSTEMS	P7
4	java.util.TimeZone.getAvailableIDs	A1
5	android.view.View.getDisplay	A6
6	android.support.v7.view.ActionMode.getTitle	A7

usado para avaliar o desempenho dos modelos. No trabalho proposto, além das métricas TP, FP, TN e FN, os valores TPR e FPR também são considerado para analisar o desempenho do aprendizado de máquina modelos usando várias métricas de desempenho. Para resolver o problema de um conjunto de dados desequilibrado, usamos subconjuntos do conjunto de dados para analisar o desempenho dos algoritmos. Além disso, o valor da avaliação MCC métricas mostram como o modelo de redução de recursos multicamadas melhora o desempenho dos algoritmos mesmo quando o conjunto de dados está desequilibrado.

Os algoritmos de aprendizado de máquina funcionam de forma eficiente em um conjunto de dados médio e rotulado. Os modelos são fáceis de interpretar e tem requisitos de recursos limitados. O processamento e treinamento tempo varia de segundos a horas. Esses modelos são adequados para aplicações leves. No entanto, a enorme quantidade de dados dificultar a proteção do sistema contra potenciais desconhecidos ataques.

Algoritmos de aprendizado profundo desempenharão um papel fundamental na análise e interpretar um grande volume de dados. Qiu et ai. (2020) discute vários algoritmos de aprendizado profundo que podem abstrair comportamentos complexos de um grande número de aplicativos Android. Algoritmos de aprendizado profundo podem proteger eficientemente o sistema de um malware desconhecido, pois esses algoritmos podem interpretar a partir de um conjunto de dados não rotulado. Qiu et ai. (2019) propõe um método A3CM (Auto matic Capability Annotation for Android Malware) que detecta e anota automaticamente malware anteriormente desconhecido. A abordagem extrai características semânticas e usa vetores para representar as feições. Para usos do método de classificação SVM, DT e DNN (Deep Neural Network), DNN mostra desempenho superior com A3CM para classificação de família de malware multiclasse. Sua abordagem se baseia em um conjunto de dados limitado de 6.899 amostras maliciosas anotadas. Além disso, ajustar os parâmetros de DNN reduz o desempenho do sistema.

A extensão do uso do Android para lidar com a segurança da IoT (Internet das Coisas) e CPS (Cyber-Physical Systems) também aumentou a taxa de ataques maliciosos sofisticados contra dispositivos Android. Para aumentar a segurança de tais dispositivos requer analisar o comportamento de aplicativos Android em um ambiente de tempo real. SideNet, um modelo de aprendizado profundo proposto por Ma et ai. (2021), analisa comportamentos sensíveis de aplicativos Android. O modelo é uma combinação de Encoder (Universal Encoder for Time

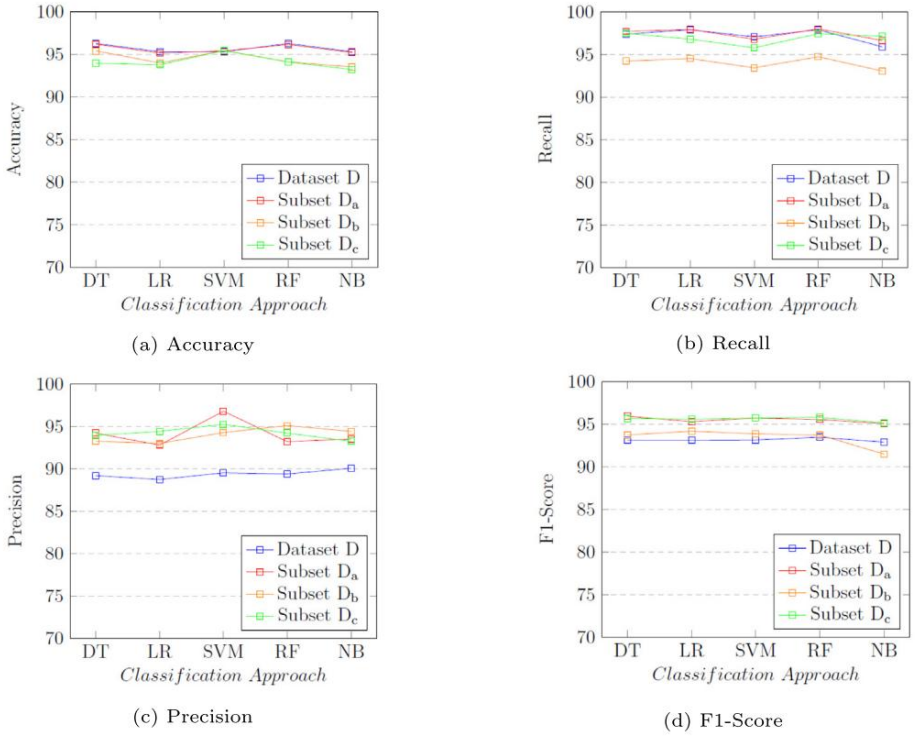

Fig. 4. Análise comparativa de Accuracy, Recall, Precision e, F1-Score de D e três subconjuntos.

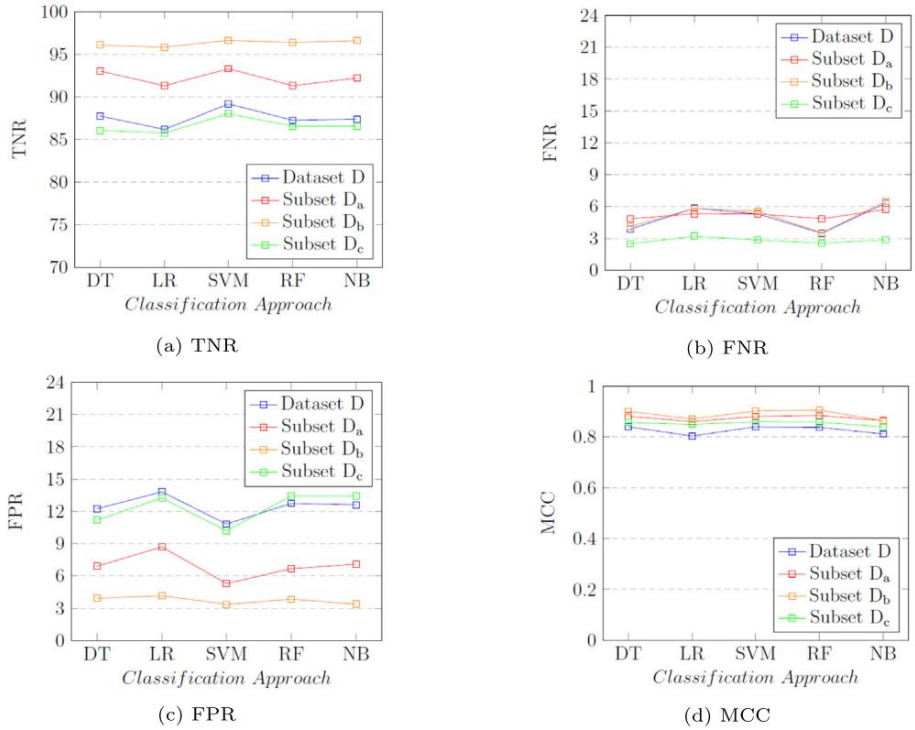

Fig. 5. Análise comparativa de Especificidade, FNR, FPR e, MCC de D e três subconjuntos.

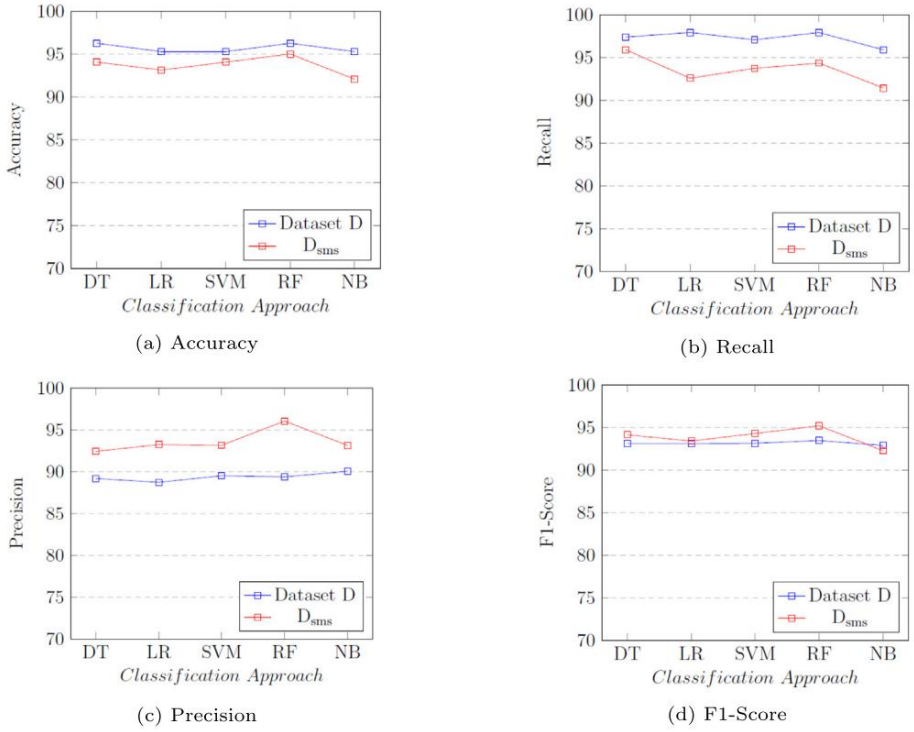

Fig. 6. Análise comparativa de Accuracy, Recall, Precision e, F1-Score de D & Dsms.

Series) (Serrà et al., 2018) e ResNet (framework de aprendizagem residual) (He et al., 2015). A abordagem de aprendizagem esparsa é empregada para reduzir o número de parâmetros e o tempo de treinamento do modelo (Dettmers e Zettlemoyer, 2019). O autor utiliza 15 aplicativos em tempo real para analisar o desempenho do SideNet. O modelo apresentou um aumento significativo na taxa de precisão e eficiência em comparação com os modelos existentes. O modelo requer dispositivos offline (observador de canal lateral) e um Android Open Source Project (AOSP) personalizado para coletar informações confidenciais para análise. O SideNet requer um número considerável de

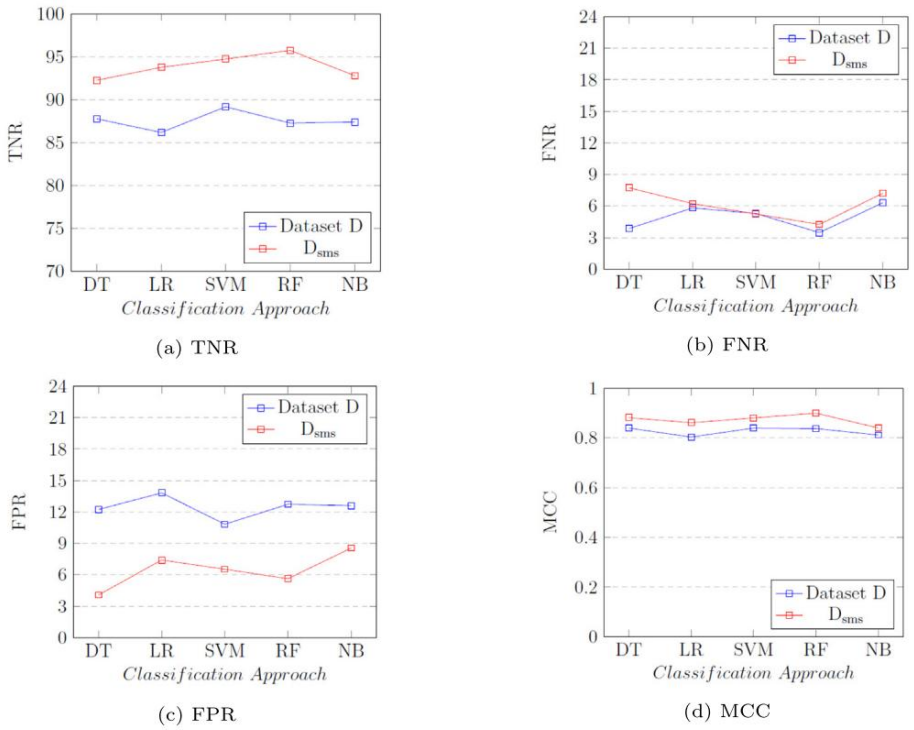

Fig. 7. Análise comparativa de Especificidade, FNR, FPR e, MCC de D & Dsms.

Tabela 11
Análise comparativa com trabalhos relacionados.

Autor	Características	atuação	Conjunto de dados	Observações
Sahs et ai. (2012)	Permissões, Bytecode Sanz et	Precisão e pontuação F1 diminuem com o aumento de arquivos benignos 86,41% Taxa de precisão	2081 benignos e 91 aplicativos maliciosos 1811 benignos e 249	Conjunto de dados tendencioso; Alta taxa de falsos negativos.
al. (2013) Permissões			aplicativos maliciosos 741 benignos e 1260 aplicativos	Conjunto de dados tendencioso; Baixa taxa de precisão em comparação com a abordagem proposta.
Liang e Du (2014)	Permissões de combinação de	Taxa de precisão de 96%; 88% de taxa de reconhecimento de arquivos benignos 95,1% de taxa	maliciosos 800 benignos e 800 aplicativos maliciosos 850	Conjunto de dados limitado; Alta taxa de falsos positivos.
Sharma et ai. (2014)	permissões, permissões de chamadas de API,	de precisão	benignos e 620 aplicativos maliciosos 1846 benignos e 5560	Conjunto de dados limitado
Verma et ai. (2016)	permissões de intenção,	Taxa de precisão de 94%	aplicativos maliciosos 3500 benignos e 8000 aplicativos	Conjunto de dados menor para uma classificação confiável.
Feizollah et ai.	(2017)	Taxa de precisão de 91% usando intenção do Android e	maliciosos 945 benignos e 1725 aplicativos maliciosos 1000	Conjunto de dados tendencioso
Razak et al. (2018) Permissões	Intenção	83% usando permissão Taxa de sensibilidade de 95,6%	benignos e 1.000 aplicativos maliciosos 1.200 aplicativos	Baixa pontuação F1_ em comparação com a abordagem proposta.
Shang F. et ai. (2018)	Permissões	Taxa de detecção de 86,54% com Naive Bayes e 97,59%	benignos e 1.280 aplicativos maliciosos 27.041 aplicativos	Conjunto de dados limitado; Baixa taxa de detecção em comparação
Li et ai. (2019)	Permissões, chamadas	com Naive Bayes ponderada Taxa de precisão de 88,69%	benignos e 26.276 aplicativos	com a abordagem proposta. Conjunto de dados limitado; Baixa taxa de precisão em comparação com a
Guan et ai. (2020) Permissões,	de API Intenção	Taxa de precisão de 96,6%	maliciosos 1.000 aplicativos benignos e 1.000 aplicativos	abordagem proposta. Conjunto de dados limitado; O uso de um grande número de recursos
Jung et ai. (2021)	Permissões, chamadas	Taxa de precisão de 96,51%	maliciosos 6.170 aplicativos benignos e 5.279 aplicativos maliciosos	com PSO resulta no aumento da complexidade do tempo. Conjunto de dados balanceado; Precisão ligeiramente alta e baixa taxa
(2021) Permissões	de API Sahin et al.	Taxa de pontuação F1 de 96,1%		de recall em comparação com a abordagem proposta. Conjunto de dados limitado
OSFS	Permissões, Intenção e chamadas de API	Taxa de precisão de 96,28% e recall de 97,92% avaliar		Alta taxa de falsos positivos

recursos para operar, o que aumenta o custo e a complexidade do sistema. Além disso, uma abordagem de aprendizado profundo para trabalhar com eficiência requer um grande conjunto de dados para análise, aumentando a complexidade de tempo do sistema. A dependência de hardware e uma grande quantidade de dados aumentam o custo geral do sistema e afetam o desempenho.

Os pesquisadores empregam modelos de aprendizado profundo para detectar uma ampla gama de ataques contra a plataforma Android, mas ao custo de maior complexidade, custo e tempo. Há uma necessidade de abordar certos fatores ao empregar algoritmos de aprendizado profundo para análise.
O conjunto de dados deve ser grande o suficiente para um algoritmo chegar a uma interpretação. O modelo de aprendizado profundo requer a otimização de sev

parâmetros gerais que retardam o processo de treinamento e aumentam o tempo de processamento do modelo de aprendizado profundo. Um grande conjunto de dados aumentará o número de recursos a serem analisados (Flair, 2018; Naumenko, 2020). No futuro, empregaremos a abordagem de redução de recursos proposta em um modelo de aprendizado profundo para remover o número de recursos insignificantes. Reduzir o número de recursos para treinar um modelo de aprendizado profundo diminuirá significativamente o tempo de treinamento e melhorará o desempenho.

8. Conclusão

Este estudo propõe um método para descobrir recursos que fornecem alta precisão na detecção de malware. O método proposto usa técnicas de redução de características para eliminar características irrelevantes.

A redução gradual garantiu que restassem apenas recursos informativos e significativos. Esses recursos principais são usados com várias abordagens de classificação para testar a eficácia dos recursos resultantes. O método atinge uma precisão tão alta quanto 96,28%, Recall 97,92% e F1-Score de 93,47% usando a abordagem de aprendizado de máquina Random Forest. A comparação com trabalhos relacionados mostra que o trabalho proposto com um conjunto reduzido de recursos fornece alta Precisão. Esses resultados supõem que Permissões e chamadas de API podem desempenhar um papel essencial nas abordagens de detecção de malware estático. Há uma variação no melhor conjunto de recursos para cada método de classificação. Os resultados mostram que as ferramentas de detecção de malware não podem focar em nenhum dos aspectos individuais de forma independente. Uma ênfase igual deve ser colocada em recursos e abordagens de classificação, pois ambos parecem dependentes um do outro.

Em trabalhos futuros, é possível experimentar mais recursos. Os pesquisadores podem usar o conjunto de recursos descobertos neste artigo para obter melhores insights sobre a eficácia dos resultados.

Financiamento

Este trabalho é o resultado da pesquisa financiada pelo Ministério do Desenvolvimento de Recursos Humanos

Declaração de Interesse Concorrente

Os autores declaram que não conhecem interesses financeiros concorrentes ou relações pessoais que possam ter influenciado o trabalho relatado neste artigo.

Referências

A. Desenvolvedores, Permissões no Android, Permissões, Disponível: URL:https:// developer.android.com/guide/topics/permissions/overview (2021).

Anael Beaugnon, AH, 2018. Pierre Collet, Métricas de desempenho de detecção, Taxa de detecção, Disponível: URL: https://... detect_perf.html#:~:text=A%20confusion%20matrix%20permite%20to,PTP% 2BFN.

AndroPyTool, Andropytool, GitHub, Disponível: URL: https://github.com/alexMyG/ AndroPyTool (2019).

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., Rieck, K., 2014. Drebin: Detecção eficaz e explicável de malware Android no seu bolso, Simpósio sobre Rede e Segurança de Sistemas Distribuídos (NDSS) (02 2014). doi:10.14722/ndss.2014.23247.

Artyom Skrobov, SM, 2019. Ataques avançados de phishing por SMS contra smartphones modernos baseados em Android, Check Point Software Technologies LTD, Disponível: URL: https:// research.checkpoint.com/advanced-sms-phishing-attacks-against modern-android -baseados em smartphones/.

A. Equipe, Intenções, Desenvolvedor Android, Disponível: URL:https://developer.android.com/ referência/android/conteúdo/intenção (2019).

Bhat, P., Dutta, K., 2019. Uma pesquisa sobre várias ameaças e estado atual de segurança na plataforma android. Computação ACM. Pesquisas 52, 1–35. https://doi.org/10.1145/ 3301285.

Bhat, P., Dutta, K., 2021. Cogramdroid-uma abordagem para detecção de malware no Android usando opcode ngrams, Simultaneidade e Computação: Prática e Experiência n/a (n/a) e6332. arXiv: https://onlinelibrary.wiley.com/doi/pdf/ 10.1002/cpe.6332, 10.1002/cpe.6332. doi: URL: https://onlinelibrary. wiley.com/doi/abs/10.1002/cpe.6332.

Biau, G., 2012. Análise de um modelo de florestas aleatórias, J. Mach. Aprender. Res. 13 (1) (2012) 1063–1095, URL:http://dl.acm.org/citation.cfm?id=2503308.2343682.

Cebuc, E., 2020. Como estamos lidando com os ataques de sobreposição do Android em 2020?, F Disponível: URL: https://thehackernews.com/2020/05/stranhogg Secure, android-vulnerability.html.

Cimitile, A., Martinelli, F., Mercaldo, F., 2017. O aprendizado de máquina atende ao malware ios: identificando aplicativos maliciosos no ambiente apple (02 2017).

Coulter, R., Han, Q.-L., Pan, L., Zhang, J., Xiang, Y., 2020. Análise de código para sistemas cibernéticos inteligentes: uma abordagem baseada em dados, Ciências da Informação 524 (03 2020) . doi:10.1016/j.ins.2020.03.036.

Crussell, J., Gibler, C., Chen, H., 2012. Ataque dos clones: Detectando aplicativos clonados em Android Markets. In: Foresti, S., Yung, M., Martinelli, F. (Eds.), Computer Security – ESORICS 2012. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 37–54.

Dettmers, T., Zettlemoyer, L., 2019. Redes esparsas do zero: treinamento mais rápido sem perda de desempenho, CoRR abs/1907.04840 (2019). arXiv:1907.04840. URL: http://arxiv.org/abs/1907.04840.

Donges, N., 2021. Um guia completo para o algoritmo de floresta aleatória, Tudo o que você precisa saber sobre o modelo de floresta aleatória., Disponível: URL:https://builtin.com/data science/ random-forest-algorithm (2021).

Evgeniou, T., Pontil, M., 2001. Support vector machines: Theory and applications, in: Machine Learning and Its Applications, Springer-VerlagBerlin, Heidelberg, 2001.

Feizollah, A., Anuar, NB, Salleh, R., Wahab, AWA, 2015. Uma revisão sobre a seleção de recursos na detecção de malware móvel. Dígito. Investigação 13 (C), 22-37. https://doi. org/10.1016/ j.diin.2015.02.001.

Feizollah, A., Anuar, N., Salleh, R., Suarez-Tangil, G., Furnell, S., 2017. Androdiálise: Análise da eficácia da intenção do Android na detecção de malware. Segurança de Computadores 65, 121–134. https://doi.org/10.1016/j.cose.2016.11.007.

Felt, AP, Chin, E., Hanna, S., Song, D., Wagner, D., 2011. Permissões do Android desmistificadas. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, https://doi.org/10.1145/ 2046707.2046779 pág. 627-638.

Fernando, J., 2016. O que é uma API? como chamar uma API do Android?, DroidMentor, Disponível: URL:https://droidmentor.com/api-call-api-android/.

Flair, D., 2018. Deep learning vs machine learning - desmistificado em palavras simples, Blog, Disponível: URL:https://data-flair.training/blogs/deep-learning-vs machine-learning/ (2018).

Guan, J., Mao, B., Jiang, X., 1634 (2020). A seleção de recursos com base em AndroidManifest.xml. J. Física: Conf. Ser. https://doi.org/10.1088/1742-6596/ 1634/1/012027 012027.

Haddadpajouh, H., Dehghantanha, A., Khayami, R., Choo, K.-KR, 2018. Detecção inteligente de ameaças de malware os x com inspeção de código, J. Computer Virology and Hacking Techniques 14 (08 2018). doi:10.1007/s11416-017-0307-5.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Aprendizagem residual profunda para reconhecimento de imagem, CoRR abs/1512.03385 (2015). arXiv:1512.03385. URL: http://arxiv.org/abs/ 1512.03385.

Holzinger, A., Errath, M., Searle, G., Thurnher, B., Slany, W., 2005. Da programação extrema e engenharia de usabilidade à usabilidade extrema no ensino de engenharia de software (xp+ue/ spl rarr/ xu) , em: 29th Annual International Computer Software and Applications Conference (COMPSAC'05), Vol. 2, 2005, pp. 169–172 Vol. 1. doi:10.1109/COMPSAC.2005.80.

JSON, ECMA-404 The JSON Data Interchange Standard, Disponível: URL: https://www.json.org/ (1999).

Jung, J., Park, J., je Cho, S., Han, S., Park, M., Cho, H.-H., 2021. https://jit.ndhu.edu.tw/ article/ view/ 2500Engenharia e avaliação de recursos para esquema de detecção de malware do Android, J. Internet Technology 22 (2) (2021) 423–440. URL: https://jit.ndhu.edu.tw/article/view/2500.

Kapratwar, A., Troia, FD, Stamp, M., 2017. Análise estática e dinâmica de malware android, em: Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ForSE, (ICISSP 2017), INSTICC , SciTePress, pp. 653-662. doi:10.5220/0006256706530662.

Mary Gladence, L., Karthi, M., Anu, M., 2015. Uma comparação estatística de regressão logística e diferentes métodos de classificação bayes para aprendizado de máquina, ARPN J. Eng. Aplic. Sci. 10 (2015) 5947-5953.

Khandelwal, S., 2019 Novo ataque permite que aplicativos Android capturem dados de alto-falantes sem qualquer permissão, The Hacker News, Disponível: URL: https://thehackernews.com/ 2019/07/android-side-channel-attacks.html.

Kumar, M., 2020. Nova falha do Android que afeta mais de 1 bilhão de telefones permite que invasores sequestrem aplicativos, The Hacker News, Disponível: URL: https:// thehackernews.com/ 2020/05/stranhogg-android-vulnerability.html.

Liang, S., Du, X., 2014. Esquema baseado em combinação de permissão para detecção de malware móvel Android, IEEE International Conference on Communications, ICC 2014, Sydney, Austrália, 10–14 de junho de 2014 (2014) 2301–2306 doi :10.1109/ICC.2014.6883666. URL: https://doi.org/10.1109/ICC.2014.6883666.

Li, J., Wu, B., Wen, W., 2019. Método de detecção de malware Android baseado em padrão frequente e baías ingênuas ponderadas. In: Yun, X., Wen, W., Lang, B., Yan, H., Ding, L., Li, J., Zhou, Y. (Eds.), Cyber Security. Springer Singapore, Cingapura, pp. 36-51.

Lin, G., Zhang, J., Luo, W., Pan, L., Xiang, Y., 2017. Pôster: Descoberta de vulnerabilidades com aprendizado de representação de funções a partir de projetos não rotulados. In: Anais da Conferência ACM SIGSAC 2017 sobre Segurança de Computadores e Comunicações. Association for Computing Machinery, Nova York, NY, EUA, pp. 2539-2541 . https://doi.org/ 10.1145/3133956.3138840.

Ma, H., Tian, J., Qiu, K., Lo, D., Gao, D., Wu, D., Jia, C., Baker, T., 2021. Vigilância de comportamento sensível a aplicativos baseada em aprendizado profundo para sistemas ciberfísicos com Android. Trans. IEEE Indústria Inf. 17 (8), 5840-5850. https://doi.org/10.1109/ TII.2020.3038745 .

Mahdavifar, S., Abdul Kadir, AF, Fatemi, R., Alhadidi, D., Ghorbani, AA, 2020. Classificação dinâmica de categoria de malware android usando deep learning semi-supervisionado (2020) doi:10.1109/DASC-PICom-CBDCom CyberSciTech49142.2020.00094.

Mandal, A., Panarotto, F., Cortesi, A., Ferrara, P., Spoto, F., 2019. Análise estática de aplicativos de infoentretenimento automático e diagnóstico a bordo ii do Android. Software: Prática e Experiência 49, 1131–1161.

Matthews, B., 1975. Comparação da estrutura secundária prevista e observada da lisozima do fago t4, Biochimica et Biophysica Acta (BBA) - Estrutura Proteica. 405 (2), 442-451. https://doi.org/10.1016/0005-2795(75)90109-9. URL: http://www.sciencedirect.com/science/article/pii/0005279575901099.

Mukherjee, S., Sharma, N., 2012. Detecção de intrusão usando classificador naive bayes com redução de recursos, Procedia Technology 4 (2012) 119–128, 2nd International Conference on Computer, Communication, Control and Information Technology (C3IT-2012) on 25 a 26 de fevereiro de 2012. doi: 10.1016/j. protcy.2012.05.017.

Naumenko, V., 2020. Guia sobre aprendizado de máquina versus aprendizado profundo versus inteligência artificial, Trends Engineering, Disponível: URL: https://jelvix.com/blog/ai-vs machine-learning-vs-deep-learning (2020).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Mach. Aprender. Res. 12, 2825-2830.

J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, Y. Wang, Y. Xiang, A3cm: Anotação de capacidade automática para malware Android, IEEE Access PP (2019) 1–1. doi:10.1109/ACCESS.2019.2946392.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., Xiang, Y., 2020. Uma pesquisa sobre detecção de malware no Android com modelos neurais profundos. Computação ACM. Sobreviv. 53 (6), (dezembro https://doi.org/10.1145/3417978.

Quinlan, JR, 1986. Indução de árvores de decisão. Mach. Aprenda 1, 81–106.

Razak, MFA, Anuar, NB, Othman, F., Firdaus, A., Afifi, F., Salleh, R., 2018. Bio inspirado para otimização de recursos e detecção de malware. Arabian J. Sci. Eng. 43, 6963-6979.

Sahin, D., Kural, O., Akleylek, S., Kilic, E., 2021. Um novo sistema de detecção de malware android baseado em permissão usando seleção de recursos com base em regressão linear. Computação Neural. Appl., 1–16 https://doi.org/10.1007/s00521-021-05875-1.

Sahs, J., Khan, L., 2012. Uma abordagem de aprendizado de máquina para detecção de malware no Android. In: Proceedings - 2012 Conferência Europeia de Inteligência e Informática de Segurança, EISIC 2012, pp. 141–147. https://doi.org/10.1109/EISIC.2012.34.

Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, PG, Álvarez, G., 2013. Puma: Uso de permissão para detectar malware no Android. In: Herrero, Á., Snášel, V., Abraham, A., Zelinka, I., Baruque, B., Quintián, H., Calvo, JL, Sedano, J., Corchado, E. (Eds.), Conferência Conjunta Internacional CISIS'12-ICEUTE'12-SOCO'12 Sessões Especiais. Springer Berlin Heidelberg, Berlim, Heidelberg, pp. 289–298.

Serrà, J., Pascual, S., Karatzoglou, A., 2018. Rumo a um codificador de rede neural universal para séries temporais, CoRR abs/1805.03908 (2018). arXiv:1805.03908. URL: http://arxiv.org/abs/1805.03908.

Shang, F., Li, Y., Deng, X., He, D., 2018. Método de detecção de malware Android baseado em naive bayes e algoritmo de correlação de permissão. Computação em Cluster 21, 1–12. https://doi.org/10.1007/s10586-017-0981-6.

Sharma, A., Dash, SK, 2014. Chamadas de API de mineração e permissões para malware Android detecção, 191-205.

Sharma, H., Kumar, S., 2016. Uma pesquisa sobre algoritmos de árvore de decisão de classificação em mineração de dados, International Journal of Science and Research (IJSR).

Shrivastav, N., 2020. Confusion matric, TPR, Disponível: URL: https:// medium.datadriveninvestor.com/confusion-matric-tpr-fpr-fnr-tnr-precision recall-f1-score-73efa162a25f.

Statcounter, participação de mercado de sistemas operacionais móveis em todo o mundo, GlobalStats, disponível: URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (2021).

Verma, S., Muttoo, S., 2016. Uma estrutura de detecção de malware para Android baseada em permissões e intenções. Defense Science J. 66, 618. https://doi.org/10.14429/dsj.66.10803 .

VirusShare, equipe do Virusshare, conjunto de dados, disponível: URL: https://virusshare.com/ (2019).

Virustotal, Virustotal equipe de antivírus, Antivírus, Disponível: URL:https://www.virustotal.com/gui/home/upload (2019).

Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., Wu, K.-P., 2012. Droidmat: detecção de malware Android por meio de manifesto e api chama rastreamento. In: Proceedings of the 2012 7th Asia Joint Conference on Information Security AsiaJCIS 2012, pp. 62-69. https://doi.org/10.1109/AsiaJCIS.2012.18.

Yerima, S., Muttik, I., Sezer, S., 2015. Detecção de malware android de alta precisão usando ensemble learning, IET Inform. Segurança (04 2015). doi:10.1049/iet ifs.2014.0099.