Universidade Federal do Amazonas
Instituto de Computação
Programa de Pós-Graduação em Informática

Gabriel Sousa Canto

Jaccard-ABC e Cor-ABC para Seleção de Características em Datasets de *Malware* Android

Manaus - Amazonas
Setembro 2024

Gabriel Sousa Canto

Jaccard-ABC e Cor-ABC para Seleção de Características em Datasets de *Malware* Android

Defesa de Mestrado apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Dr. Eduardo Luzeiro Feitosa

Manaus - Amazonas
Setembro 2024

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Canto, Gabriel Sousa
C232j Jaccard-ABC e Cor-ABC para Seleção de Características em Datasets de Malware Android / Gabriel Sousa Canto . 2024
63 f.: il. color; 31 cm.

Orientador: Eduardo Luzeiro Feitosa
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Android. 2. Detecção de Malwares. 3. Seleção de Características. 4. artificial bee colony. 5. Meta-heurísticas. I. Feitosa, Eduardo Luzeiro. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"JACCARD-ABC E COR-ABC PARA SELEÇÃO DE CARACTERÍSTICAS EM DATASETS DE MALWARE ANDROID"

GABRIEL SOUSA CANTO

DISSERTAÇÃO DE MESTRADO DEFENDIDA E APROVADA PELA BANCA EXAMINADORA CONSTITUÍDA PELOS PROFESSORES:

Prof. Dr. Eduardo Luzeiro Feitosa - PRESIDENTE

Profa. Dra. Eulanda Miranda dos Santos - MEMBRO INTERNO

Prof. Dr. Diego Luis Kreutz - MEMBRO EXTERNO

MANAUS, 08 de novembro de 2024.

Documento assinado eletronicamente por **Eduardo Luzeiro Feitosa**, **Professor do Magistério Superior**, em 07/11/2024, às 15:02, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Diego Luis Kreutz**, **Usuário Externo**, em 07/11/2024, às 17:06, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eulanda Miranda dos Santos**, **Professor do Magistério Superior**, em 07/11/2024, às 18:00, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Maria do Perpétuo Socorro Vasconcelos Palheta**, **Secretária**, em 08/11/2024, às 17:46, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2314236** e o código CRC **699C34CD**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.046756/2024-16 SEI nº 2314236

AGRADECIMENTOS

Agradeço profundamente à minha família, meu porto seguro, especialmente aos meus pais, Jeane e Denilson, por todo o apoio, amor e incentivo que me acompanharam em cada etapa da minha vida. Sem vocês, nada disso seria possível.

À minha namorada, Aline, meu agradecimento por estar sempre ao meu lado, me apoiando e incentivando nos momentos em que mais precisei. Sua presença foi essencial durante essa jornada.

Aos meus amigos da UFAM, Ana, Bianca, Edwin e Samiles, minha gratidão por estarem sempre comigo, compartilhando momentos e me ajudando sempre que precisei. Vocês tornaram essa caminhada muito mais leve e especial.

Ao professor Dr. Eduardo Luzeiro Feitosa, meu orientador, agradeço por me aceitar como orientando e por toda a dedicação, paciência e apoio ao longo do período do meu mestrado. Estendo também meus agradecimentos aos professores Dra. Eulanda Miranda dos Santos e Dr. Diego Luis Kreutz, cujos ensinamentos e feedbacks foram indispensáveis para o aprimoramento deste trabalho.

Aos colegas do ETSS, meu muito obrigado por me ajudarem a esclarecer dúvidas sobre as disciplinas e as diferentes etapas desse percurso.

Ao meu primo Mauro, agradeço pelo suporte na obtenção de recursos de computação em nuvem, que foram fundamentais para a realização deste trabalho.

Por fim, a todas as pessoas que, direta ou indiretamente, contribuíram para que esse projeto se tornasse realidade, meu sincero obrigado!

Resumo

Esta dissertação tem como objetivo aprimorar a seleção de características em conjuntos de dados voltados para a detecção de malware Android, propondo dois novos métodos de exploração de vizinhança baseados em medidas de similaridade para o algoritmo Colônia Artificial de Abelhas (ABC). Os métodos propostos, Jaccard-ABC e Cor-ABC, demonstraram eficácia na redução do número de características em diversos cenários, frequentemente superando 30% de redução, com Jaccard-ABC excedendo os resultados obtidos pelo ABC tradicional em todos os cenários. Além disso, os métodos mantiveram um desempenho competitivo em termos de acurácia, precisão, revocação e F1, com variações geralmente inferiores a 1% em relação ao ABC tradicional. O método Cor-ABC, em particular, apresentou maior estabilidade, reduzindo a aleatoriedade interna do processo de seleção. Conclui-se que os métodos propostos oferecem contribuições valiosas para a seleção eficiente de características, especialmente no contexto de detecção de *malware* em dispositivos Android.

Palavras-chave: Android, Detecção de *Malwares*, Seleção de Características, ABC, Correlação, Similaridade, Meta-heurísticas, Inteligência de Enxames

Abstract

This dissertation aims to improve feature selection in datasets focused on Android malware detection by proposing two new neighborhood exploration methods based on similarity measures for the Artificial Bee Colony (ABC) algorithm. The proposed methods, Jaccard-ABC and Cor-ABC, demonstrated effectiveness in reducing the number of features across various scenarios, often surpassing 30% reduction, with Jaccard-ABC outperforming the traditional ABC in all scenarios. Furthermore, the methods maintained competitive performance in terms of accuracy, precision, recall, and F1, with variations generally less than 1% compared to the traditional ABC. The Cor-ABC method, in particular, exhibited greater stability by reducing internal randomness in the selection process. In conclusion, the proposed methods provide valuable contributions to efficient feature selection, especially in the context of malware detection on Android devices.

Keywords: Android, Malware Detection, Feature Selection, ABC, Correlation, Similarity, Metaheuristics, Swarm Intelligence

Lista de Figuras

Lista de Tabelas

Nomenclatura

ABC (*Artificial Bee Colony*)

API Interfaces de Programação de Aplicativos (*Application Program Interface*)

APK (*Android Package Kit*)

BALO (*Binary Ant Lion Optimizer*)

BBOA-S (*Binary Butterfly Optimization Algorithm (S variant)*)

BBOA-V (*Binary Butterfly Optimization Algorithm (V variant)*)

BOA (*Butterfly Optimization Algorithm*)

BWPA (*Binary Waterwheel Plant Algorithm*)

GA (*Genetic Algoritm*)

GOA (*Grasshopper Optimization Algorithm*)

LASSO (*Least Absolute Shrinkage and Selection Operator*)

MR (*Modification Rate*)

NLP Processamento de Linguagem Natural (*Natural Language Processing*)

PCA (*Principal Component Analysis*)

PSO (*Particle Swarm Optimization*)

RFE (*Recursive Feature Elimination*)

SRBM (*Subspace based Restricted Boltzmann Machines*)

SVM Máquina de Vetores de Suporte (*Support Vector Machine*)

TF-IDF (*Term Frequency-Inverse Document Frequency*)

WOA (*Whale Optimization Algorithm*)

Sumário

Capítulo 1

Introdução

Neste capítulo, realizamos a contextualização da pesquisa, fornecendo uma visão geral das realizações contidas. Inicialmente, discorremos sobre o contexto, incluindo as razões que motivaram a seleção deste tema como objeto de estudo. Em seguida, apresentamos o problema a ser resolvido. Depois, os objetivos a serem alcançados. Por fim, a organização do texto.

1.1 Contexto

Esta pesquisa é parte integrante do projeto Malware Hunter, financiado pela Motorola, cujo objetivo é desenvolver duas técnicas para a detecção de *malware* em dispositivos Android, combinando características[1] estáticas e dinâmicas. De acordo com Wang et al. (2019), as características extraídas de aplicativos Android podem ser: (1) estáticas, informações sobre o aplicativo como nome, tamanho, permissões requeridas, código-fonte e padrão de programação, obtidas sem a necessidade de instalá-lo e executá-lo; ou (2) dinâmicas, aquelas que refletem os comportamentos do aplicativo na interação com o sistema operacional ou na conectividade de rede, e para serem obtidas requerem a instalação e execução do aplicativo (APK) e a coleta de processos derivados da execução do aplicativo, seja através de um emulador ou dispositivo físico.

De volta ao projeto, seu escopo consiste na implementação de duas soluções: uma baseada em regras de associação, onde limiares estatísticos pré-especificados são utilizados; e (2) outra baseada em classificadores de aprendizado de máquina supervisionada. Vale destacar que para ambas implementações, o fluxo de funcionamento é o mesmo:

[1] informação de algum aspecto da aplicação (*e.g.*, permissão), que permite determinar seu comportamento.

coleta das características dos aplicativos (APK), extração das características relevantes, análise e apresentação dos resultados (Figura 1.1).

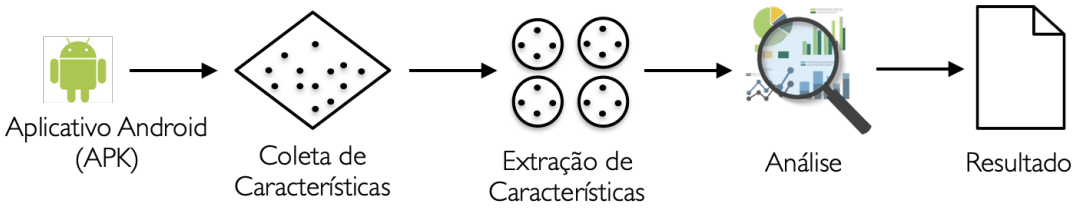

Figura 1.1: Típico Processo de Detecção de *Malware* Android.

Como representado na Figura 1.1, o processo de detecção envolve a manipulação (coleta e extração) das características. Na prática, ambas as soluções serão testadas e validadas com conjuntos de dados que contêm amostras rotuladas, como benignas ou malignas, de aplicativos Android. Tipicamente, amostras de aplicativos Android vêm acompanhadas de conjuntos de características, como permissões, intenções, chamadas de API, chamadas de sistema, tráfego de rede, componentes de *hardware*, componentes do aplicativo, comandos de sistema, *bytecodes* e *strings* semânticas (Damodaran et al., 2017; Qiu et al., 2020; Wang et al., 2019).

Entretanto, esse processo de teste e validação das soluções possui um desafio: a quantidade de dados que um *dataset* pode conter. Por exemplo, há *datasets* que chegam a conter mais de 1 milhão de amostras e mais de 1 milhão de características (Roy et al., 2015). Em termos computacionais, é demasiadamente custoso treinar e validar modelos com *datasets* dessa magnitude (e.g., matriz de 1 milhão por 1 milhão). Na prática, um *dataset* contendo apenas 6 mil amostras e 643 características pode consumir até 5,8 horas de computação em um computador moderno (Cai et al., 2021). Além disso, é provável que nem todas as características contribuam igualmente para a classificação. Algumas características podem até ser redundantes, irrelevantes ou até mesmo prejudiciais para o processo de classificação. Em linhas gerais, trabalhar com *datasets* grandes torna-se inviável em termos de custo financeiro e tempo computacional, impedindo até a detecção de *malwares* em tempo real.

Assim, reduzir a dimensionalidade do *dataset*, através da seleção das características mais significativas, é parte de um conjunto de otimizações possíveis e necessárias para viabilizar e dar qualidade ao processo de teste e validação das soluções. A seleção de características, também conhecida como *feature selection*, é um componente essencial em muitos problemas que envolvem classificação, especialmente os que abrangem aprendizado de máquina e mineração de dados. Ela se refere ao processo de escolher um

subconjunto relevante e informativo de características de um conjunto de dados mais amplo (Guyon e Elisseeff, 2003). Uma característica é considerada relevante quando é condicionalmente dependente dos rótulos (Chandrashekar e Sahin, 2014). Essa etapa é crucial porque a qualidade e a relevância das características têm um impacto significativo no desempenho das soluções. A seleção de características não apenas simplifica o processo de classificação, mas também melhora a capacidade de generalização da classificação, reduzindo o risco de superajuste (*overfitting*) e aumentando a interpretabilidade dos modelos de aprendizado. Além disso, em casos onde a coleta de dados é cara ou demorada, a seleção de características pode economizar recursos valiosos, uma vez que se concentra nas características mais importantes.

No contexto do projeto Malware Hunter foram explorados métodos de seleção de características que utilizavam: regressão linear (Şahin et al., 2021); teste qui-quadrado e a análise de variância (ANOVA) (Alazab, 2020); a combinação de técnicas, incluindo *Correlation Best Feature Selection*, *Classifier Subset Evaluation*, *Filtered Subset Evaluation*, *Rough Set Analysis* (RSA), teste do qui-quadrado, análise de regressão logística e Análise de Componentes Principais (PCA) (Mahindru e Sangal, 2021a,b); ganho de informação e ponderação de pesos (Cai et al., 2021); entre outros.

Um ponto observado durante a execução e análise desses métodos foi o uso da combinação de diferentes técnicas para escolher as características, mas sem consider a relevância das características em conjunto. Os métodos implementados e avaliados só avaliam o desempenho do classificador com o conjunto de características selecionados. Por esse motivo, decidimos implementar o método de seleção, proposto por (Schiezaro e Pedrini, 2013b), uma adaptação do algoritmo de "Colônia Artificial de Abelhas" (*Artificial Bee Colony*) , ou ABC, para seleção de caraterísticas binárias.

1.2 Definição do Problema

Como mencionado, o método proposto por (Schiezaro e Pedrini, 2013b) é baseado no ABC, que pode ser classificado como um método de metaheuristíca inspirado no comportamento das abelhas ao procurarem alimento. O ABC é utilizado para resolver problemas de otimização, onde se busca encontrar a melhor solução possível em um espaço de busca complexo. No ABC, um conjunto de "abelhas artificiais" é usado para explorar o espaço de busca em busca da solução ótima. Cada abelha artificial mantém uma solução candidata e compartilha informações sobre a qualidade das soluções com outras abelhas. Essas informações são utilizadas para guiar a busca, de forma que as abelhas tendem a se concentrar nas áreas do espaço de busca que têm maior

probabilidade de conter soluções ótimas.

O algoritmo ABC foi originalmente desenvolvido para lidar com problemas de otimização contínua. Para atuar na seleção de características, é necessário adaptá-lo. Assim, trabalhos como (Beheshti, 2021; Pampará e Engelbrecht, 2011; Yavuz e Aydin, 2016) implementaram estratégias que envolvem a aplicação de funções de transferência e modulação angular, adaptando o ABC para o seleção de características binárias. Vale ressaltar que, embora as soluções candidatas sejam representadas como vetores binários, o processo de busca ocorre predominantemente em um espaço de busca contínuo.

De acordo com Nguyen et al. (2020), projetar um mecanismo de busca adequado para a representação binária geralmente resulta em melhor desempenho. Assim, Schiezaro e Pedrini (2013a) propuseram uma abordagem modificada do algoritmo ABC para a seleção de características. Nela, as soluções continuam sendo representadas como vetores binários. A população inicial de soluções é gerada aleatoriamente e para explorar os arredores das soluções, foi adotada uma substituição no mecanismo de busca padrão do ABC ao se utilizar o operador de mutação do algoritmo genético (GA), que é mais adequado para representações binárias.

Contudo, embora a abordagem de Schiezaro e Pedrini (2013a) seja útil para evitar mínimos locais e enriquecer a diversidade das características selecionadas, ela apresenta desvantagens. A principal delas é a possibilidade de eliminar características cruciais para o desempenho dos modelos. Isso ocorre porque, para que uma característica seja selecionada, o valor aleatório escolhido pelo operador de mutação deve ser menor que a taxa de modificação (MR). Sendo assim, o único parâmetro de controle para selecionar uma característica é a taxa de modificação (MR), que estabelece uma espécie de limiar para a seleção. O restante do processo de seleção de características do algoritmo é totalmente aleatório e não depende de nenhuma métrica. Portanto, características relevantes podem não ser selecionadas, enquanto características redundantes e irrelevantes podem ser mantidas.

Por este motivo, tentando resolver esse problema, surgiu a ideia de empregar medidas de similaridade para orientar a busca no espaço de características, permitindo uma seleção mais informada e precisa das características mais relevantes para a detecção de *malware*.

1.3 Objetivos

O objetivo desta dissertação é identificar, implementar e validar novas funções de busca, explorando medidas de similaridade, para métodos de seleção de características inspirado

no algoritmo ABC, visando aprimorar a qualidade das características escolhidas e, consequentemente, aumentar a eficácia dos sistemas de detecção de *malware* Android.

Para esse objetivo, pretende-se realizar os seguintes objetivos específicos:

- Analisar métodos de seleção de características inspirados no algoritmo ABC ou similares, com o propósito de aplicá-los ao cenário de *malwares* Android.
- Realizar uma estimativa minuciosa dos parâmetros (função *fitness*, taxa de modificação, por exemplo) inerentes ao método proposto, identificando aqueles que otimizam os resultados de maneira mais eficiente e eficaz.

1.4 Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma:

- O Capítulo 2 aborda os conceitos teóricos utilizados na pesquisa, bem como os fundamentos que norteiam os métodos de seleção de características. Além disso, apresenta diversos estudos anteriores que trabalharam na seleção de características, tanto métodos gerais de seleção de características, como específicos do domínio de detecção de *malware* Android.
- O Capítulo 3 descreve o método proposto e sua implementação.
- O Capítulo 4 apresenta as conjuntos de dados e métricas avaliados, os parâmetros utilizados, como e em quais condições ocorreram as avaliações e os resultados obtidos.

Capítulo 2

Conceitos Básicos

Neste capítulo, exploraremos conceitos fundamentais que constituem as bases desta pesquisa. Abordaremos a definição de seleção de características, categorizando os tipos de métodos relevantes. Por fim, discutiremos o conceito de medidas de similaridade, apresentando algumas medidas de similaridade específicas que serão empregadas ao longo deste trabalho.

2.1 Seleção de Características

Na área de aprendizado de máquina, uma característica (*feature*) é uma propriedade mensurável de um objeto que está sendo observado (Dhal e Azad, 2022). Por exemplo, em uma imagem, as características podem ser os valores dos pixels. Já em um texto, as características podem ser as palavras, o número de letras, entre outras. As características são usadas para capturar informações relevantes dos dados e permitir que os algoritmos de aprendizado de máquina entendam os padrões e relacionamentos presentes nos dados (Chandrashekar e Sahin, 2014; Dhal e Azad, 2022).

Neste contexto, a seleção de características desempenha um papel crucial no pré-processamento de dados, permitindo-nos eliminar características irrelevantes, redundantes e ruidosas. Isso não apenas ajuda a evitar a "maldição da dimensionalidade", mas também reduz o custo computacional associado (Dhal e Azad, 2022).

A etapa de seleção de características pode ser conceituada como a busca pelo subconjunto ideal de características que maximize uma determinada métrica. No entanto, realizar uma busca completa em um conjunto de características com n elementos implica em avaliar $2^n - 1$ subconjuntos, o que se torna impraticável quando lidamos com um grande número de características. Portanto, a solução encontrada para a tarefa de

selecionar o melhor subconjunto é geralmente subótima, uma vez que alcançar a solução ótima se torna inviável na maioria dos conjuntos de dados atuais.

2.1.1 Métodos de Seleção

Existem várias abordagens para a seleção de características, cada uma com suas vantagens e limitações. Alguns métodos utilizam critérios estatísticos para avaliar a relevância das características em relação à variável de saída. Outros recorrem a algoritmos de aprendizado de máquina para determinar quais características contribuem mais para a capacidade preditiva do modelo.

De acordo com Agrawal et al. (2021) e Dhal e Azad (2022), os métodos de seleção de características podem ser classificados em três categorias principais: baseados em filtro, baseados em *wrapper* e baseados em *embedded*. Cada categoria aborda o problema de seleção de características de maneira única, considerando aspectos como eficiência computacional e desempenho preditivo.

Baseados em Filtros

A seleção de características baseada em filtros é a abordagem mais comum e amplamente utilizada, uma vez que foca em avaliar e classificar as características com base em medidas de relevância ou importância, independentemente de um algoritmo de classificação específico (Bommert et al., 2022; Lazar et al., 2012). A ideia é atribuir uma pontuação ou *ranking* a cada característica, geralmente com base em alguma métrica de relevância, que pode variar de acordo com a natureza dos dados e do problema em questão. Dentre os exemplos comuns de medidas de relevância estão: correlação, ganho de informação (*info gain*), coeficientes estatísticos como o qui-quadrado e análises de variância (Lazar et al., 2012).

Métodos de filtro são computacionalmente eficientes, uma vez que a avaliação das características ocorre independentemente do treinamento de um modelo de aprendizado de máquina, tornando-os ideais para lidar com grandes conjuntos de dados (Nssibi et al., 2023). Além disso, como as medidas de relevância são calculadas antes do treinamento do modelo, os métodos de filtro não estão sujeitos a problemas de sobreajuste. Por outro lado, métodos de filtro não levam em consideração as interações entre as características ou como elas podem ser usadas em conjunto para melhorar o desempenho do modelo. Isso significa que eles podem selecionar características individualmente relevantes, mas que podem não contribuir tanto quando combinadas em um contexto mais amplo.

As medidas empregadas pelos métodos baseados em filtro podem ser categorizadas

em diversas classes, tais como medidas de informação, distância, consistência, similaridade e medidas estatísticas.

Baseados em *Wrapper*

A seleção de características baseada em *wrapper* é uma abordagem mais complexa e intensiva em termos computacionais, caracterizando-se por usar algum algoritmo de aprendizado de máquina específico em torno do processo de seleção de características (Chandrashekar e Sahin, 2014; Jia et al., 2022). Diferentemente dos métodos de filtro, os métodos baseados em *wrapper* avaliam as características levando em consideração o desempenho real de um modelo de aprendizado de máquina, ou seja, treinando e avaliando o modelo usando diferentes subconjuntos de características. Cada subconjunto é uma combinação diferente de características e o objetivo é encontrar o subconjunto que resulta no melhor desempenho do modelo. Isso requer iterativamente treinar o modelo com diferentes conjuntos de características e avaliar seu desempenho usando alguma métrica de avaliação, como acurácia, precisão, *recall* ou F1-score (Chandrashekar e Sahin, 2014).

Dentre as vantagens do uso de métodos de seleção baseados em *wrapper*, destacamos a capacidade de capturar interações complexas entre as características. Como o desempenho do modelo é avaliado diretamente, esses métodos consideram como as características contribuem para a capacidade de generalização do modelo (Dhal e Azad, 2022). Isso pode ser especialmente útil em cenários em que as interações entre as características são importantes para a tarefa de aprendizado de máquina. Por outro lado, eles tendem a ser computacionalmente mais intensivos, pois requerem o treinamento e a avaliação repetida de um modelo de aprendizado de máquina. Além disso, podem ser mais suscetíveis ao sobreajuste, uma vez que o desempenho do modelo é avaliado usando os mesmos dados usados para treiná-lo.

Os métodos de *wrapper* podem adotar diferentes estratégias de busca (Jović et al., 2015):

- **Busca Exponencial**: Envolve examinar todas as possíveis opções de maneira sistemática. Isso significa que, à medida que o espaço de busca aumenta, o esforço computacional necessário cresce exponencialmente. Tem como vantagem a precisão, pois considera todas as possibilidades, mas é frequentemente impraticável devido ao alto custo computacional.
- **Busca Sequencial**: Progride passo a passo, considerando ou removendo características uma de cada vez. Isso evita a explosão combinatória associada à busca

exponencial, tornando-a mais eficiente em termos de tempo e recursos. No entanto, a busca sequencial pode ficar presa em ótimos locais, uma vez que as decisões tomadas em estágios anteriores são fixas.

- **Busca Aleatória**: Adiciona um toque de aleatoriedade à exploração do espaço de busca. Isso permite que o algoritmo evite ficar preso em ótimos locais e explore soluções alternativas. As estratégias de busca aleatória são comumente conhecidas como abordagens baseadas em população.

Baseados em *Embedded*

A seleção de características baseada em *embedded* é uma abordagem intermediária entre o uso de filtro e métodos baseados em *wrapper* (Jović et al., 2015). Métodos *embedded* incorporam a etapa de seleção de características diretamente no processo de treinamento de um modelo de aprendizado de máquina, tornando-a parte integrante do próprio modelo (Dhal e Azad, 2022; Jović et al., 2015). Desta forma, a seleção de características ocorre durante o processo de otimização do modelo de aprendizado de máquina. Isso significa que o algoritmo de aprendizado de máquina é ajustado para escolher automaticamente as características mais relevantes e descartar as menos relevantes enquanto aprende a mapear os dados para as saídas desejadas.

Existem diferentes abordagens para a incorporação da seleção de características no processo de treinamento de um modelo. Um exemplo é a Regressão Logística L1 (Lasso), que adiciona um termo de penalização ao processo de otimização, levando automaticamente a um conjunto de coeficientes (e, portanto, características) mais esparso (Tang et al., 2014). Isso faz com que características irrelevantes tenham coeficientes próximos a zero e, portanto, sejam efetivamente eliminadas do modelo. Outra abordagem é a utilização de modelos baseados em árvores, como *Random Forest* e *Gradient Boosting*, que naturalmente têm a capacidade de medir a importância de cada característica durante o treinamento. As características menos importantes podem ser removidas ou consideradas menos relevantes na construção do modelo.

Uma vantagem dos métodos de seleção baseados em *embedded* é que eles aproveitam a própria tarefa de aprendizado para determinar quais características são mais relevantes. Isso reduz a necessidade de treinamento e avaliação repetida do modelo, tornando-os menos computacionalmente intensivos do que os métodos de seleção baseados em *wrapper*. No entanto, os métodos de seleção baseados em *embedded* podem ser sensíveis à escolha do algoritmo de aprendizado de máquina e aos hiperparâmetros usados. Além disso, eles podem não capturar interações complexas entre as características da mesma

forma que os métodos baseados em *wrapper*.

2.1.2 Meta-Heurísticas

Apesar dos diferentes métodos de seleção de características, vários algoritmos meta-heurísticos também têm sido efetivamente usados para obter um conjunto ideal de características. De acordo com Sharma e Kaur (2021), meta-heurísticas são abordagens computacionais que visam resolver problemas de otimização que têm se mostrado úteis e eficazes na seleção de características.

As meta-heurísticas são excelentes ferramentas para explorar o espaço de soluções em busca do subconjunto de características ideal. Em vez de examinar todas as combinações possíveis, essas técnicas usam estratégias inteligentes para guiar a busca em direção a soluções promissoras. Uma característica importante das meta-heurísticas é a capacidade de equilibrar a busca de novas soluções e a melhoria das soluções atuais. Isso ajuda a evitar ficar preso em mínimos locais e a encontrar soluções mais próximas do ótimo global.

A escolha da meta-heurística depende do problema em questão e das características do conjunto de dados. Muitas vezes, as meta-heurísticas podem ser personalizadas para se adaptar às características específicas de um problema ou conjunto de dados. Isso inclui a definição de operadores de mutação, cruzamento e seleção que se adequam ao contexto da seleção de características.

Os algoritmos meta-heurísticos podem ser divididos em quatro tipos (Agrawal et al., 2021):

- **Baseados em evolução**: algoritmos que se inspiram nos processos de evolução natural para encontrar soluções. Dentre os principais algoritmos, destacam-se: o algoritmo Genético (Holland, 1992), baseado na teoria da evolução de Darwin; o algoritmo de Busca Tabu (Glover, 1986) e o algoritmo de Evolução Diferencial (Storn e Price, 1997).

- **Baseados em inteligência de enxames**: algoritmos que utilizam princípios inspirados no comportamento coletivo de organismos sociais, como pássaros, abelhas ou peixes, para resolver problemas de seleção de características em conjuntos de dados. A ideia por trás desses algoritmos é que, assim como os indivíduos, um enxame pode coordenar suas ações para alcançar objetivos comuns, ou seja, as características de um conjunto de dados podem ser selecionadas de maneira eficaz para otimizar a precisão de um modelo de aprendizado de máquina. Dentre os principais algoritmos, destacam-se: o algoritmo de Colônia de Formigas (ACO, *Ant*

Colony Optimization) (Dorigo et al., 2006), o algoritmo de Enxame de Partículas (PSO, *Particle Swarm Optimization*) (Kennedy e Eberhart, 1995) e o algoritmo de Colônia Artificial de Abelhas (ABC, *Artificial Bee Colony*) (Karaboga, 2005).

- **Baseados na Física**: são uma categoria de técnicas de otimização que se inspiram em princípios e leis físicas, como o comportamento das partículas, campos de energia e outras interações físicas. Dentre os principais algoritmos, destacam-se: o algoritmo de Busca Gravitacional (GSA, *Gravitational Search Algorithm*) (Rashedi et al., 2009), o algoritmo Busca de Sistema Carregado (CSS, *Charged System Search*) (Kaveh e Talatahari, 2010) e o algoritmo de Busca baseado em Galáxia (GbSA, *Galaxy-based Search Algorithm*) (Shah-Hosseini, 2011).

- **Baseados no comportamento humano**: são uma categoria de técnicas de otimização que se inspiram no comportamento de seres humanos, imitando processos de tomada de decisão humanos, interações sociais ou abordagens cognitivas. Entre os principais algoritmos, destacam-se: o algoritmo do Campeonato da Liga (LCA, *League Championship Algorithm*) (Kashan, 2009), o algoritmo de Inspiração Humana (HIA, *Human-Inspired Algorithm*) (Zhang et al., 2009), o algoritmo de Otimização Socioemocional (SEOA, *Social emotional optimization algorithm*) (Xu et al., 2010).

2.2 Estabilidade de Métodos de Seleção de Características

No contexto da seleção de características, **estabilidade** refere-se à consistência das características selecionadas por um método de seleção quando aplicado a diferentes amostras de dados ou sob variações nos dados (Nogueira e Brown, 2016). Em outras palavras, um método de seleção de características é considerado estável se ele seleciona um conjunto semelhante de características, independentemente das mudanças nos dados ou em diferentes execuções do método.

A importância da estabilidade pode ser descrita em termos de confiabilidade, facilidade de interpretação e garantia de reprodutibilidade. A estabilidade aumenta a **confiança** de que o modelo baseado nas características selecionadas terá um desempenho confiável em novos dados. Se um método é instável, o conjunto de características selecionadas pode mudar drasticamente com pequenas variações nos dados, o que pode afetar negativamente a performance do modelo.

Já sobre a **facilidade de interpretação**, as características selecionadas de forma consistente são mais fáceis de interpretar e compreender, pois são robustas contra

variações nos dados. Por fim, a **garantia de reprodutibilidade** é crucial em pesquisa e desenvolvimento. Métodos estáveis garantem que os resultados obtidos podem ser replicados em diferentes experimentos e conjuntos de dados, o que contribui para a validade dos achados.

Imagine que temos $d = 5$ características para escolher. Podemos representar o conjunto de características de saída do método de seleção por um vetor binário s de comprimento 5, onde um 1 na i-ésima posição significa que a i-ésima característica foi selecionada e um 0 significa que não foi selecionada. Por exemplo, o vetor [11100] significa que as características 1,2 e 3 foram selecionadas e as características 4 e 5 não foram selecionadas. Agora imagine que aplicamos dois métodos de seleção de características distintos, P_1 e P_2, a $M = 3$ amostras diferentes dos dados e obtemos a seguinte saída:

$$A1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$A2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

onde as linhas de A_1 e A_2 representam os conjuntos de características retornados por P_1 e P_2, respectivamente. Todos os conjuntos de características em A_1 são idênticos, portanto, não há variação na saída do método. Cada coluna da matriz A_1 representa a seleção de cada uma das 5 características. A frequência observada das três primeiras características é igual a 1, enquanto a das duas últimas características é igual a 0. Esta situação corresponde a uma seleção totalmente estável.

Agora vejamos A_2. Nessa situação, podemos ver que há alguma variação na saída do método de seleção, pois os conjuntos de características em A_2 são diferentes. Se olharmos para a segunda e quarta colunas de A_2, que correspondem à seleção da segunda e quarta característica nos 3 conjuntos de características, podemos ver que elas são selecionadas com uma frequência igual a $p_2 = p_4 = \frac{1}{3}$, o que mostra alguma instabilidade no método de seleção de características.

Uma boa métrica de estabilidade deve satisfazer as seguintes propriedades (Nogueira e Brown, 2016):

- **Totalmente Definido**: Às vezes, o método de seleção de características produz conjuntos de características selecionadas de tamanhos diferentes quando iteramos

o procedimento n vezes.

- **Limites Superior e Inferior**: Para uma melhor compreensão da medida de estabilidade, o valor da medida de estabilidade deve estar em um intervalo limitado. Suponha que o intervalo definido para a medida de estabilidade seja $[-\infty, +\infty]$; então, um valor de saída de 0,9 será sem significado.

- **Seleção Determinística $\leftrightarrow$ Máxima Estabilidade**: O método de seleção P seleciona as mesmas k características todas as vezes, independentemente dos dados fornecidos. Este é um método completamente estável, portanto, a medida de estabilidade deve retornar seu valor máximo. O oposto também deve ser verdadeiro. Se uma medida tem um valor máximo possível C, ela deve retornar esse valor apenas quando o método de seleção for determinístico.

- **Correção por Chance**: Notado pela primeira vez por Kuncheva (2007), garante que, quando a seleção de características é aleatória, o valor esperado da estimativa de estabilidade seja constante, o qual definimos aqui como 0 por convenção. Imagine que um método de seleção P1 seleciona aleatoriamente 5 características e que um método P2 seleciona aleatoriamente 6 características; o valor de estabilidade deve ser o mesmo.

- **Monotonicidade**: Quanto maior a interseção entre os subconjuntos de características, maior a estabilidade (Nogueira, 2018).

A Tabela 2.1 apresenta as propriedades de diferentes métricas de estabilidade.

Métrica de estabilidade	**Totalmente definido**	**Limites**	**Máximo**	**Correção**	**Monotonicidade**
Jaccard	✓	✓	✓	x	✓
Tanimoto	✓	✓	x	✓	✓
Symmetrical	✓	✓	x	✓	x
Canberra	✓	✓	x	✓	x
Spearman	✓	✓	x	✓	✓
Pearson	✓	✓	x	✓	✓

Tabela 2.1: Propriedades de métricas de estabilidade

2.3 Medidas de Similaridade

Medidas de similaridade são funções matemáticas que quantificam o grau de semelhança ou correspondência entre objetos, conjuntos de dados, características ou elementos em

um determinado contexto. São empregadas para avaliar o grau de proximidade entre itens, permitindo comparar e contrastar suas características, estruturas ou propriedades. Elas têm sido amplamente utilizadas para resolver diversos problemas, como identificação de biometria com impressões digitais (Willett, 2003), no diagnóstico de doenças do transtorno bipolar (Abdel-Basset et al., 2019) e também na detecção de *malwares* Android (Taheri et al., 2020). Dependendo do domínio e do tipo de dados, diferentes métricas de similaridade podem ser aplicadas.

Dentre exemplos de medidas de similaridade/distância, destacamos:

- **Distância euclidiana**, métrica usada para calcular a distância entre dois pontos em um espaço euclidiano. É frequentemente usada em geometria e análise de dados. A fórmula para calcular a medida euclidiana entre dois pontos $P(x_1, y_1)$ e $Q(x_2, y_2)$ em um plano bidimensional é dada por:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

  Nessa fórmula, $d(P, Q)$ representa a distância euclidiana entre os pontos P e Q. A distância euclidiana é muito aplicada em diversas áreas, como na identificação gênica (Ghosh e Barman, 2016), para cálculo de distância entre duas séries temporais (Hu et al., 2023; Keogh e Pazzani, 2000) e também seleção de características (Patel e Upadhyay, 2020), entre outras áreas em que a distância entre pontos é importante para a avaliação de similaridade.

- **Distância de Hausdorff**, métrica usada para medir a semelhança entre dois conjuntos de pontos em um espaço métrico. É frequentemente utilizada na avaliação de métodos de segmentação de imagens médicas (Karimi e Salcudean, 2019), reconhecimento de objetos a partir de imagens (Kumar et al., 2020) e em análise de trajetória de veículos (Sousa et al., 2020). A fórmula para calcular a distância de Hausdorff entre dois conjuntos A e B em um espaço métrico é dada por:

$$H(A, B) = \max\left(\sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(b, a)\right)$$

  Nesta fórmula, $H(A, B)$ representa a distância de Hausdorff entre os conjuntos A e B. O termo $d(a, b)$ representa a métrica (distância) entre dois pontos a e b no espaço métrico considerado. A distância de Hausdorff mede a maior distância possível entre um ponto em um conjunto e seu ponto correspondente mais próximo no outro conjunto, considerando ambos os conjuntos de forma simétrica. Isso

significa que a métrica é sensível a pontos extremos e *outliers*.

- **Distância de Hamming**, usada para medir a diferença entre duas sequências de igual comprimento que consistem em símbolos de um conjunto finito. Algumas aplicações para a distância de Hamming incluem: detecção de *malware* Android (Taheri et al., 2020), detecção e correção de erros (EDAC) (Hillier et al., 2019). A fórmula para calcular a distância de Hamming entre duas sequências A e B de comprimento n é dada por:

$$H(A,B) = \sum_{i=1}^{n} \delta(a_i, b_i)$$

Nesta fórmula, $H(A,B)$ representa a distância de Hamming entre as sequências A e B, e $\delta(a_i, b_i)$ é uma função que retorna 0 se os símbolos a_i e b_i forem iguais, e 1 se forem diferentes. A distância de Hamming é particularmente útil quando se trata de comparar sequências binárias. Ela mede o número de posições em que as sequências diferem. Quanto menor a distância de Hamming, mais semelhantes são as duas sequências. Por exemplo, considere as sequências binárias $A = 101101$ e $B = 100111$. A Distância de Hamming entre elas é calculada da seguinte forma:

$$H(A,B) = \delta(1,1)+\delta(0,0)+\delta(1,0)+\delta(1,1)+\delta(0,1)+\delta(1,1) = 0+0+1+0+1+0 = 2$$

Portanto, a distância de Hamming entre A e B é igual a 2.

No contexto da seleção de características, as medidas de similaridade podem ser usadas para identificar quais características são mais relevantes, redundantes ou irrelevantes para um determinado objetivo. Isso ajuda a eliminar o "ruído" nos dados, melhorando a qualidade da análise e dos modelos construídos.

Coeficiente de Correlação de Pearson

O coeficiente de correlação de Pearson, também conhecido como r de Pearson ou correlação linear de Pearson, é usado para medir a relação linear entre duas variáveis contínuas X e Y. A fórmula é dada por:

$$\rho = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum (X_i - \overline{X})^2 \sum (Y_i - \overline{Y})^2}} \tag{2.1}$$

Onde:

- ρ é o coeficiente de correlação de Pearson.
- X_i e Y_i são os valores individuais nas amostras de X e Y, respectivamente.
- $\overline{X}$ é a média de X.
- $\overline{Y}$ é a média de Y.

O coeficiente de correlação de Pearson mede a força e a direção da relação linear entre X e Y. Se r for próximo de 1, isso indica uma forte correlação positiva (à medida que X aumenta, Y também aumenta linearmente). Se r for próximo de -1, isso indica uma forte correlação negativa (à medida que X aumenta, Y diminui linearmente). Se r estiver próximo de 0, isso indica uma correlação fraca ou inexistente entre as variáveis.

O coeficiente de correlação de Pearson, no caso em que as duas variáveis são binárias, se reduz ao coeficiente de correlação de Matthews (Matthews, 1975), uma medida estatística usada para avaliar a associação ou relação entre duas variáveis categóricas, geralmente dispostas em uma tabela de contingência (Chicco e Jurman, 2020). É especialmente útil quando as duas variáveis são binárias (cada uma tem duas categorias possíveis).

Tabela 2.2: Tabela de Contingência para o Coenficiente de Correlação de Matthews

	y = 1	y = 0
x = 1	a	b
x = 0	c	d

A fórmula para calcular o coeficiente de Matthews é dada por:

$$MCC = \frac{(ad - bc)}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \tag{2.2}$$

Onde:

- a é o número de observações em que ambas as variáveis são iguais à primeira categoria.
- b é o número de observações em que a primeira variável é igual à primeira categoria, e a segunda variável é igual à segunda categoria.
- c é o número de observações em que a primeira variável é igual à segunda categoria, e a segunda variável é igual à primeira categoria.

- d é o número de observações em que ambas as variáveis são iguais à segunda categoria.

Como o coeficiente de Matthews varia de -1 a 1, podemos interpretar da seguinte forma:

- Se o valor do coeficiente for próximo de 1, as duas variáveis binárias são consideradas positivamente associadas
- Se o valor do coeficiente for próximo de 0, não há associação entre as duas variáveis.
- Se o valor do coeficiente for próximo de -1, as duas variáveis binárias são consideradas negativamente associadas

Exemplo

Considerando as duas sequências binárias $A = (1, 0, 1, 1)$ e $B = (1, 0, 0, 1)$, vamos ter a seguinte tabela de contigência para o coeficiente de Matthews:

Tabela 2.3: Tabela de contingência para as sequências A e B

	$A_i = 1$	$A_i = 0$
$B_i = 1$	2	0
$B_i = 0$	1	1

Temos, então:

$$MCC = \frac{(2 \cdot 1 - 0 \cdot 1)}{\sqrt{(2+0)(1+1)(2+1)(0+1)}} = 0{,}17$$

Isso mostra que as sequêcias binárias A e B têm uma associação alta.

O coeficiente de Matthews é especialmente útil para medir a associação em tabelas de contingência 2x2, onde ambas as variáveis têm duas categorias. Ele é muito utilizado em aprendizado de máquina, como função de perda (Abhishek e Hamarneh, 2021) e principalmente como métrica de avaliação (Chicco e Jurman, 2020; Liu et al., 2015).

Coeficiente de Jaccard

O coeficiente de similaridade de Jaccard é uma estatística usada para mensurar a similaridade e a diversidade de conjuntos de amostras. É uma das principais medidas de similaridade, utilizada em diversos contextos, por exemplo, sistema de recomendação (Bag et al., 2019) e medição de similaridade entre palavras-chave e termos de índice

para facilitar o acesso rápido aos resultados por sistemas de buscas (Niwattanakul et al., 2013).

Matematicamente, dados dois vetores, v_1 e v_2, com n coordenadas binárias cada um, o coeficiente Jaccard é uma medida útil da sobreposição que v_1 e v_2 compartilham através de suas coordenadas. Cada coordenada de v_1 e v_2 pode assumir o valor 0 ou 1. O número total de cada combinação de coordenadas, de ambos v_1 e v_2, é dado pelo seguinte:

- M_{11} representa o número total de coordenadas em que v_1 e v_2 têm o valor 1.
- M_{01} representa o número total de coordenadas onde a coordenada de v_1 é 0 e a coordenada de v_2 é 1.
- M_{10} representa o número total de coordenadas onde a coordenada de v_1 é 1 e a coordenada de v_2 é 0.
- M_{00} representa o número total de coordenadas onde v_1 e v_2 têm o valor 0.

Cada coordenada deve pertencer a uma dessas categorias, ou seja:

$$M_{11} + M_{01} + M_{10} + M_{00} = n. \tag{2.3}$$

O coeficiente de similaridade de Jaccard é dado por:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{2.4}$$

O resultado do coeficiente de similaridade de Jaccard varia entre 0 e 1, onde:

- 0 indica que os conjuntos são completamente disjuntos (nenhum elemento em comum).
- 1 indica que os conjuntos são idênticos (todos os elementos em comum).

Exemplo

Podemos calcular o coeficiente de Jaccard para o exemplo apresentado no coeficiente de Matthews - sequências $A = (1, 0, 1, 1)$ e $B = (1, 0, 0, 1)$. De acordo com a Tabela 2.3 e usando a Equação 2.4, temos:

$$J = \frac{2}{1 + 0 + 2} = 0{,}67$$

Isso mostra que essas duas sequências têm alta similaridade, como podemos ver facilmente, já que as sequências divergem em apenas uma coordenada.

2.4 Comportamento das Abelhas e ABC

Os enxames de abelhas exibem uma organização fascinante e uma adaptabilidade coletivamente inteligente em resposta às mudanças ambientais. Esses insetos possuem habilidades notáveis, incluindo memória fotográfica, sistemas sensoriais avançados, navegação de alta tecnologia e até mesmo indícios de processos de tomada de decisão em grupo ao selecionar novos locais para seus ninhos. Suas tarefas variam desde cuidar da rainha e da prole, armazenar, recuperar e distribuir mel e pólen até comunicação e coleta de alimentos (Seeley, 2014). Essas características têm atraído a atenção dos pesquisadores que buscam modelar o comportamento inteligente das abelhas.

O principal exemplo é o algoritmo ABC (*Artificial Bee Colony algorithm*). Proposto por Karaboga (2005), é um algoritmo de otimização baseado no comportamento inteligente de forrageamento de enxame de abelhas. Nele, o espaço de busca é representado como o ambiente de colônias de abelhas, onde cada ponto no espaço de busca corresponde a uma fonte potencial de alimento. Cada fonte de alimento contém uma quantidade diferente de néctar e esses néctares representam o valor de aptidão da fonte de alimento. O ABC trabalha com três tipos de abelhas (Karaboga, 2005):

- **Empregadas**, que estão vinculadas a fontes de alimento específicas;
- **Observadoras**, responsáveis por observar as danças das abelhas empregadas de dentro da colmeia para escolher uma fonte de alimento;
- **Exploradoras**, que buscam aleatoriamente novas fontes de alimento.

Tanto as abelhas observadoras quanto as abelhas exploradoras também são conhecidas como abelhas desempregadas. A colônia tem a mesma quantidade de abelhas artificiais empregadas e observadoras.

No estágio inicial, todas as posições das fontes de alimento são identificadas pelas abelhas exploradoras. Em seguida, as abelhas empregadas e as abelhas observadoras exploram o néctar das fontes de alimento, um processo que, com o tempo, leva à exaustão dessas fontes. Quando isso acontece, a abelha empregada que estava explorando a fonte de alimento esgotada se transforma em uma abelha exploradora em busca de novas fontes de alimento. Em resumo, a abelha empregada cuja fonte de alimento se esgotou assume o papel de abelha exploradora.

Dentro do contexto do ABC, a posição de uma fonte de alimento representa uma possível solução para o problema em questão, e a quantidade de néctar presente em uma fonte de alimento corresponde à qualidade (ou aptidão) da solução associada. Em sua forma mais básica, o número de abelhas empregadas é igual ao número de fontes de alimento (ou soluções), já que cada abelha empregada está designada a uma e apenas uma fonte de alimento.

O algoritmo ABC consiste em cincos fases principais, confome mostrado na Figura 2.1.

Figura 2.1: Diagrama de Fluxo das Principais Etapas do Algoritmo ABC.

Fase de inicialização

Cria as fontes de alimentos aleatórias, onde cada uma delas corresponde a uma possível solução do problema da seguinte forma:

$$x_{ij} = x_{min,j} + \alpha \cdot (x_{max,j} - x_{min,j}) \tag{2.5}$$

onde, i,$j = 0, 1, 2, \cdots, D$, onde D é o número de parâmetros a serem otimizados;

$x_{max,j}, x_{min,j}$ são o limite superior e inferior na dimensão j, respectivamente; e α representa um número real aleatório no intervalo de [0,1].

Fase das abelhas empregadas

Explora toda a fonte de alimento para reconhecer as possíveis boas características, calculando a precisão de cada alimento usando abelhas empregadas. As abelhas empregadas começam pegando cada alimento da fonte de alimento e calculando o néctar (qualidade) usando uma função de custo. Em seguida, exploram a vizinhança das fontes de alimentos. A exploração da vizinhança de uma fonte de alimento é definida como:

$$v_{ij} = x_{min,j} + \phi \cdot (x_{i,j} - x_{k,j}) \tag{2.6}$$

onde, i,j,$k = 0,1,2,\cdots, D, k \neq i$, onde D é o número de parâmetros a serem otimizados; ϕ representa um número real aleatório no intervalo de $[-1,1]$.

Uma vez que v_i é produzido, a qualidade da fonte de alimento é obtida. Defini-se

$$fitness_i = \begin{cases} \frac{1}{1+f_i} & \text{se } f_i \geq 0 \\ 1 + |f_i| & \text{se} f_i < 0 \end{cases} \tag{2.7}$$

em que f_i é a função de custo.

Fase das abelhas observadoras

Depois que a qualidade é obtida, a probabilidade de uma abelha observadora escolher uma fonte de alimento para ser explorada é calculada pela seguinte fórmula:

$$p_i = \frac{fitness_i}{\sum_n fitness_i} \tag{2.8}$$

Levando-se em conta os valores de probabilidade de exploração, as fontes de alimento são selecionadas pelas abelhas observadoras. A partir deste momento, as abelhas observadoras tornam-se abelhas empregadas.

Fase da abelhas exploradoras

Uma vez que a qualidade da fonte de alimento recente não tenha sido aprimorada pelo número predefinido de iterações, referido como limite, a fonte de alimento será abandonada. Depois disso, as abelhas exploradoras tentarão, aleatoriamente, identificar

uma nova posição de fonte de alimento na dimensão original da característica calculada pela Equação 2.6.

Condição de término: As fases acima mencionadas do algoritmo ABC serão executadas repetidamente, até que o número de execuções atinja a contagem máxima de iteração.

2.5 Trabalhos Relacionados

A Tabela 2.4 apresenta uma visão abrangente dos trabalhos mais recentes relacionados à seleção de características na detecção de *malware* em Android.

Tabela 2.4: Trabalhos Relacionados

Referência	Ano	Algoritmo Proposto	Categoria	N. Algoritmos	Estabilidade	Domínio	N. Datasets	Métricas
Alhussan et al. (2023)	2023	BWPA	Wrapper	9	Não	Diversos	30	Avg. fit., Best fit., Worst fit., Avg. fit. size, Avg. err., Std. dev.,
Bommert et al. (2020)	2020	-	Filter	22	Sim	Diversos	16	Acurácia
Bommert et al. (2022)	2022	-	Filter	14	Sim	Gene expression	11	Brier score
Chen et al. (2020)	2020	-	Filter, Wrapper, Embedded	3	Não	Médico	11	Acurácia
Bhattacharya et al. (2019)	2019	PSORS-FS	Filter, Wrapper	10	Não	Malware Android	2	Acurácia, ROC, F1
Chaudhuri e Sahu (2021b)	2021	TOPSIS	Híbrido	4	Não	Gene expression	10	Acurácia
Arora e Anand (2019)	2019	BBOA-S, BBOA-V	Wrapper	8	Não	Diversos	21	Acurácia
Salah et al. (2020)	2020	FF AF	Filter	2	Não	Malware Android	1	Acurácia, Precisão, Recall, F1
Este trabalho	2024	ABC-Corr e ABC-Jaccard	Filter, Wrapper, Embedded	6	Sim	Malware Android	8	Acurácia, Precisão, Recall, F1

A Tabela inclui informações detalhadas sobre o ano de publicação, o algoritmo proposto (quando aplicável), a categoria dos métodos, o número de algoritmos avaliados, a investigação da estabilidade dos métodos testados, o domínio de aplicação, o número de datasets utilizados e as métricas de avaliação empregadas.

Os métodos de seleção de características baseados em *wrapper* são amplamente explorados. Recentemente, Alhussan et al. (2023) propuseram o algoritmo BWPA (*binary waterwheel plant algorithm*), uma abordagem baseada em *wrapper* que foi comparada com nove algoritmos em 30 datasets diversos. As métricas utilizadas incluíram várias medidas de ajuste e erro, como a melhor e pior adequação, entre outras.

Chaudhuri e Sahu (2021b) propuseram o TOPSIS uma abordagem híbrida que combina cinco métodos de filtro e um método *wrapper* para seleção de características

que superou o desempenho de métodos de filtros individuais. Os autores utilizaram uma função de transferência variável no tempo com o algoritmo Jaya Binário. Testada em 10 conjuntos de dados de micro-arranjos, a técnica obteve melhor precisão de classificação que as técnicas existentes. Assim, a técnica é eficiente para seleção de características em dados de micro-arranjos.

Boa parte das soluções que envolvem meta-heurísticas foram originalmente desenvolvidas para solucionar problemas de otimização contínua. Portanto, para aplicar esses algoritmos a problemas binários, como a seleção de características, é necessário efetuar algumas modificações. Existem várias abordagens para essa adaptação, sendo a principal delas o uso de uma função de transferência, que converte o espaço de busca contínuo para binário, como a função de transferência sigmóide. Assim, é possível aplicar meta-heurísticas para seleção de características. Entre os trabalhos que utilizaram essa abordagem está (Arora e Anand, 2019) que introduziu os algoritmos BBOA-S e BBOA-V, adaptações que utilizam funções de transferências em formas de S e V, testados em 21 datasets diversos. A principal métrica de avaliação foi a acurácia.

Seguindo a mesma ideia, os autores em (Mafarja et al., 2019) propuseram variantes binárias do recente algoritmo de otimização Grasshopper (*Grasshopper Optimization Algorithm* ou GOA). Hussien et al. (2019) utilizam essa abordagem para adaptar o algoritmo de otimização da baleia (WOA) para seleção de características fazendo uso de uma função de transferência sigmóide. Emary et al. (2016) utilizam o algoritmo de formigas-leão binário (BALO).

Entre os trabalhos que avaliaram métodos baseados em filtro estão o de Bommert et al. (2020), que avaliaram 22 métodos para seleção de características em datasets de diversos domínios. Os resultados mostraram que nenhum método superou todos os outros, mas os métodos de filtro de importância de floresta aleatória, permutação e impureza, os métodos de filtro de informação teórica sym.uncert e JMIM, e o método de filtro de teste estatístico univariado limma tiveram um bom desempenho na maioria dos conjuntos de dados. Em Bommert et al. (2022) foram avaliados 14 métodos de filtro para seleção em conjuntos de expressão gênica. Os resultados indicaram que os filtros de variância e carss se destacam em termos de desempenho preditivo, especialmente quando combinados com modelos de riscos proporcionais de Cox regularizados com L2. Esses filtros melhoram significativamente a precisão preditiva ao selecionarem apenas um pequeno conjunto de características relevantes. A análise mostrou também que o filtro de variância oferece maior estabilidade na seleção de características, tornando-o a escolha mais consistente entre os métodos avaliados.

Chen et al. (2020) exploraram a combinação de métodos de seleção de característi-

cas. Foram utilizados métodos de união, interseção e multi-interseção. Técnicas como PCA (análise de componentes principais), algoritmos genéticos e a árvore de decisão C4.5 foram usadas para representar as abordagens de seleção de características de filtro, *wrapper* e *embedded*, respectivamente. Os resultados indicam que a seleção de características em conjunto, realizada principalmente pelos métodos de união e multi-interseção, permite ao classificador SVM alcançar uma precisão de classificação maior do que as técnicas individuais. A combinação das técnicas de filtro (PCA) e *wrapper* (GA) com o método de união mostrou o maior aumento na precisão média de classificação, mantendo uma boa taxa de redução de características.

Poucos trabalhos na literatura se preocupam com o estudo da estabilidade dos métodos testados. Entre os trabalhos que avaliam a estabilidade dos métodos estão Bommert et al. (2020), que utilizaram o coeficiente de correlação de Pearson para o estudo dos métodos de seleção testados, e Bommert et al. (2022), que utilizaram a métrica de estabilidade SMA-Count.

Entre os trabalhos focados na detecção de *malware* Android, destacam-se Bhattacharya et al. (2019), que adaptaram o algoritmo PSO clássico utilizado em problemas de otimização contínua para o domínio discreto para a seleção de características utilizando teoria dos conjuntos aproximados. Salah et al. (2020), que propuseram um novo método baseado no TF-IDF. O método proposto utiliza uma variação do TF-IDF chamada FF-AF para a seleção de características em detecção de *malwares* Android. A abordagem considera cada aplicação Android como um documento e as características extraídas como termos. O TF binário é empregado, onde o valor TF é 1 se uma característica está presente em uma aplicação, e 0 caso contrário. A pontuação FF-AF substitui o IDF, eliminando a necessidade do conceito inverso, e reflete a frequência informativa de uma característica em todas as aplicações. Características com pontuações FF-AF próximas de zero ou um são consideradas não-informativas e excluídas do espaço de características. Além disso, o método propõe métricas de peso de frequência (FW) para avaliar a distribuição das frequências das características, determinando quantas delas possuem a mesma frequência e como isso impacta na redução do espaço de características. O objetivo é selecionar características comuns dentro de uma classe de aplicativos para facilitar a detecção de *malware*.

As principais limitações dos trabalhos para o domínio de *malware* Android são o número limitado de *datasets* utilizados para testar os métodos, a ausência de um estudo sobre a estabilidade dos métodos propostos e o número reduzido de métodos comparativos.

Este trabalho introduz uma variante para o algoritmo ABC binário, diferenciando-se

das soluções comuns na literatura que utilizam funções de transferência, como visto em Alhussan et al. (2023), para aplicar o algoritmo de otimização para seleção de características. A abordagem proposta neste estudo concentra-se na redução de características altamente correlacionadas, guiando a seleção de características por meio de métricas de similaridade, algo pouco encontrado na literatura. Um destaque deste trabalho é o estudo da estabilidade dos métodos propostos, algo com o qual os autores geralmente não se preocupam, mas que impacta diretamente a qualidade e a confiabilidade do método.

Capítulo 3

Métodos

Neste capítulo apresentamos o algoritmo de Colônia Artificial de Abelhas (*Artificial Bee Colony algorithm* ou simplesmente ABC). Introduzimos, também, dois novos métodos de busca para o ABC, desenvolvidos com o objetivo de realizar a seleção de características em *datasets* de detecção de *malwares*. O foco principal desses métodos é remover características redundantes e desnecessárias, contribuindo, assim, para um processo de análise mais eficaz. Eles incorporam duas medidas de similaridade entre vetores binários: a similaridade de Jaccard e o coeficiente de correlação de Pearson.

3.1 bABC - *Binary Artificial Bee Colony algorithm*

Para abordar o desafio da seleção de características, Schiezaro e Pedrini (2013a) introduziram uma abordagem inovadora no contexto do algoritmo ABC. Essa abordagem incorpora várias modificações destinadas a lidar com vetores binários que representam o subconjunto de características selecionadas. Nesta seção, iremos apresentar as modificações realizadas para adequar o algoritmo ao problema de seleção de características.

Fase de inicialização

Durante a fase de inicialização, são geradas N fontes de alimento, correspondendo ao número total de características a serem avaliadas. Cada fonte de alimento é representada por um vetor binário de tamanho N, com apenas uma única posição contendo o valor 1, enquanto todas as outras posições contêm o valor 0, como mostra a Figura 3.1. Essa estratégia é empregada com o intuito de assegurar a obtenção da solução ótima com o menor conjunto possível de características.

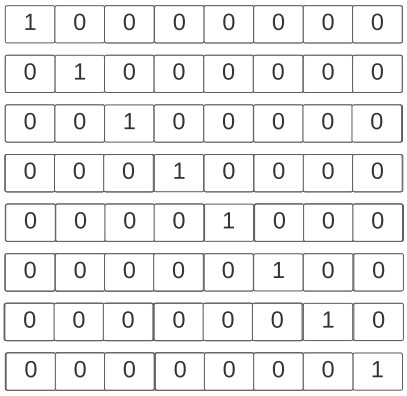

Figura 3.1: Exemplo de fontes de alimentos criadas no processo de inicialização.

Após a geração das fontes de alimento, estas são submetidas ao classificador e o valor da métrica que avalia a qualidade da fonte é registrado.

Fase das abelhas empregadas

Nesta fase, cada abelha empregada visita uma fonte de alimento e explora sua vizinhança. No método proposto, os autores introduziram duas alterações significativas. A primeira delas substituiu o mecanismo de exploração da vizinhança pelo operador de mutação do algoritmo genético para abordar o problema da seleção de características. A segunda mudança importante envolveu a aplicação desse mecanismo para todas as coordenadas do vetor binário, em contraste com a abordagem original que apenas modificava um parâmetro, resultando em uma convergência mais lenta.

Nesse novo mecanismo de exploração, um número aleatório uniformemente distribuído, representado por R_i e variando de 0 a 1, é gerado. Se o valor de R_i for menor que o parâmetro MR (Taxa de Modificação), também conhecido como Frequência de Perturbação, a característica correspondente é adicionada ao subconjunto. Caso contrário, o valor do vetor binário permanece inalterado, como ilustrado na Equação 3.1 e na Figura 3.2.

$$x_i = \begin{cases} 1 & \text{se } R_i < MR \\ x_i & \text{caso contrário} \end{cases} \tag{3.1}$$

O subconjunto de características gerado para cada vizinhança é submetido ao classificador e o valor da métrica de avaliação é registrado como a medida de aptidão da

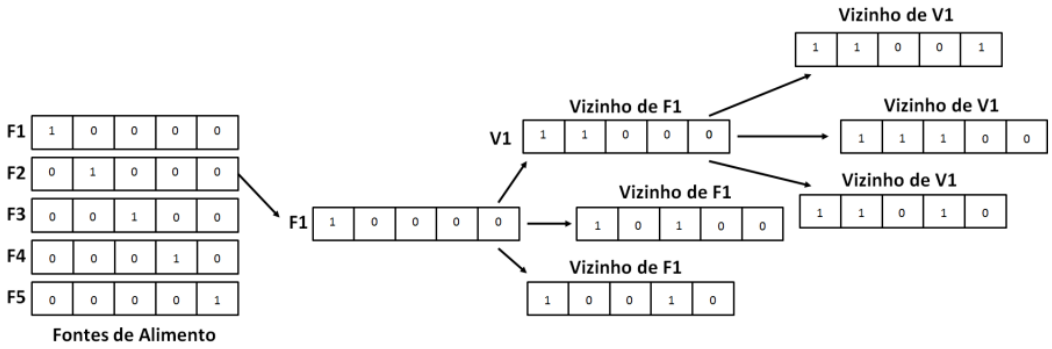

Figura 3.2: Mecanismo de busca e exploração da vizinhança [Fonte: Schiezaro e Pedrini (2013a)].

respectiva vizinhança. Quando a qualidade da fonte de alimento da vizinhança recém-criada supera a qualidade da fonte de alimento atualmente em exploração, a fonte de alimento da vizinhança é considerada uma nova. Caso contrário, a variável LIMITE, associada à fonte de alimento que está sendo explorada na vizinhança, é incrementada. Se o valor de LIMITE exceder o valor de LIMITE MÁXIMO, a fonte de alimento é abandonada, ou seja, esgotada. Para cada fonte de alimento abandonada, o método cria uma abelha exploradora para realizar uma busca aleatória em busca de uma nova fonte de alimento.

Fase das abelhas observadoras

Após a criação das fontes de alimento durante a fase de inicialização e após as abelhas empregadas terem visitado essas fontes e explorado suas vizinhanças, novas fontes de alimento, tanto de alta quanto de baixa qualidade, foram geradas. A probabilidade de uma fonte de alimento ser visitada e sua vizinhança ser explorada está diretamente relacionada à sua qualidade. Portanto, nesta etapa, as abelhas observadoras escolhem as fontes de melhor qualidade para explorar.

Para fazer essa escolha, uma variável aleatória é gerada e comparada com o valor de aptidão da fonte. Se a variável aleatória for menor do que o valor de aptidão, ela será selecionada para exploração; caso contrário, será descartada. Quando uma abelha observadora escolhe uma fonte de alimento para exploração, ela se transforma em uma abelha empregada e prossegue com a fase das abelhas empregadas.

Fase das abelhas exploradoras

Durante a fase das abelhas exploradoras, verifica-se a presença de fontes de alimento que tenham sido abandonadas na fase anterior das abelhas empregadas. Para cada fonte de alimento abandonada, uma nova abelha exploradora é criada e uma nova fonte de alimento gerada. Nessa nova fonte, um vetor binário com tamanho N de características é criado aleatoriamente e submetido ao classificador, registrando sua aptidão. A recém-criada fonte de alimento é então atribuída às abelhas exploradoras, que, a partir desse ponto, se transformam em abelhas empregadas e prosseguem com a fase das abelhas empregadas.

3.2 Novas Funções de Exploração Utilizando Medidas de Similaridade para o Algoritmo ABC

Nas abordagens propostas, fizemos uma modificação exclusivamente no mecanismo de exploração, mantendo inalteradas todas as outras etapas conforme definidas no algoritmo original proposto por (Schiezaro e Pedrini, 2013a). Para orientar de forma mais eficiente a exploração do algoritmo ABC e evitar uma busca completamente aleatória, propomos a utilização de medidas de similaridade, tais como coeficiente de correlação de Pearson (2.1) e o coeficiente de similaridade de Jaccard (2.3), para direcionar a exploração nas proximidades das fontes de alimento.

Em vez de tomar decisões de seleção ou exclusão de características com base em números aleatórios R_i e eliminar características cruciais para o desempenho dos modelos, o método proposto avalia a similaridade entre a posição i do vetor binário e uma posição aleatória j diferente de i. Se o valor da medida de similaridade entre essas duas características for menor do que o parâmetro MR, a característica é adicionada ao subconjunto; caso contrário, ela é removida. Essa abordagem foi desenvolvida com o objetivo de eliminar características que apresentam um alto grau de similaridade, uma vez que características altamente correlacionadas podem introduzir viés no classificador e reduzir a precisão dos resultados (John et al., 1994; Tolo̧si e Lengauer, 2011; Yan e Zhang, 2015).

A Equação 3.2 ilustra o mecanismo de exploração proposto com a utilização do coeficiente de correlação de Pearson:

$$s_{ij} = \begin{cases} 1 & \text{se } |\rho(X_j, X_k)| < MR \\ 0 & \text{caso contrário} \end{cases} \tag{3.2}$$

onde s_{ij} é a j-ésima coordenada da solução i do conjunto de soluções $S = \{s_1, \cdots, s_D\}$, cada $s_i \in S$ é um vetor binário onde as coordenadas representam características do conjunto de dados, com 1 indicando que a característica é selecionada e 0 indicando que não é; X_j e X_k são, respectivamente, os dados das colunas j e k do conjunto de dados X, com $j \neq k$.

Além do coeficiente de correlação de Pearson, foi utilizada também a medida de similaridade de Jaccard, medida escolhida por ser aplicada com sucesso na literatura (Aberni et al., 2020; Chaudhuri e Sahu, 2021a; Hancer et al., 2015). A Equação 3.3 ilustra o mecanismo de exploração proposto com a utilização do coeficiente de similaridade de Jaccard:

$$s_{ij} = \begin{cases} 1 & \text{se } J(X_j, X_k) < MR \\ 0 & \text{caso contrário} \end{cases} \tag{3.3}$$

onde s_{ij} é a j-ésima coordenada da solução i do conjunto de soluções $S = \{s_1, \cdots, s_D\}$, cada $s_i \in S$ é um vetor binário onde as coordenadas representam características do conjunto de dados, com 1 indicando que a característica é selecionada e 0 indicando que não é; $J(X_j, X_k)$ representa o coeficiente de similaridade de Jaccard entre as colunas j e k do conjunto de dados X.

3.3 Implementação do Algoritmo ABC

Toda implementação dos algoritmos foi realizada usando a linguagem de programação Python (versão 3.10.12) e as bibliotecas NumPy (versão 1.23.5), Pandas (versão 1.23.5) e Scikit-learn (versão 1.1.1).

Capítulo 4

Experimentação e Resultados

Este capítulo apresenta a metodologia aplicada para testar e validar os métodos propostos, bem como compará-los aos outros métodos. Para tanto, descrevemos todo o processo de experimentação (Seção 4.1), incluindo as características e os conjuntos de dados utilizados as métricas empregadas na avaliação, os parâmetros empregados na configuração dos algoritmos, as técnicas aplicadas na validação e o ambiente de experimentação. Por fim, os resultados são apresentados.

4.1 Experimentação

4.1.1 Características

Na detecção de *malwares* em Android, a característica é definida como uma informação de algum aspecto da aplicação (por exemplo, permissão), que permite determinar se tal aplicação possui indícios de comportamento malicioso. Os trabalhos na literatura categorizam, tipicamente, as características extraídas de aplicativos Android de duas formas: estáticas e dinâmicas. A Figura 4.1 apresenta os tipos de características e os subtipos de cada categoria.

Por definição, características estáticas são informações sobre o aplicativo como nome, tamanho, permissões requeridas, código-fonte e padrão de programação, obtidas sem a necessidade de instalá-lo e executá-lo. Já as características dinâmicas são aquelas que refletem os comportamentos do aplicativo na interação com o sistema operacional ou na conectividade de rede, e para serem obtidas requerem a instalação e execução do APK e a coleta de processos de execução do aplicativo, seja através de um emulador ou dispositivo físico.

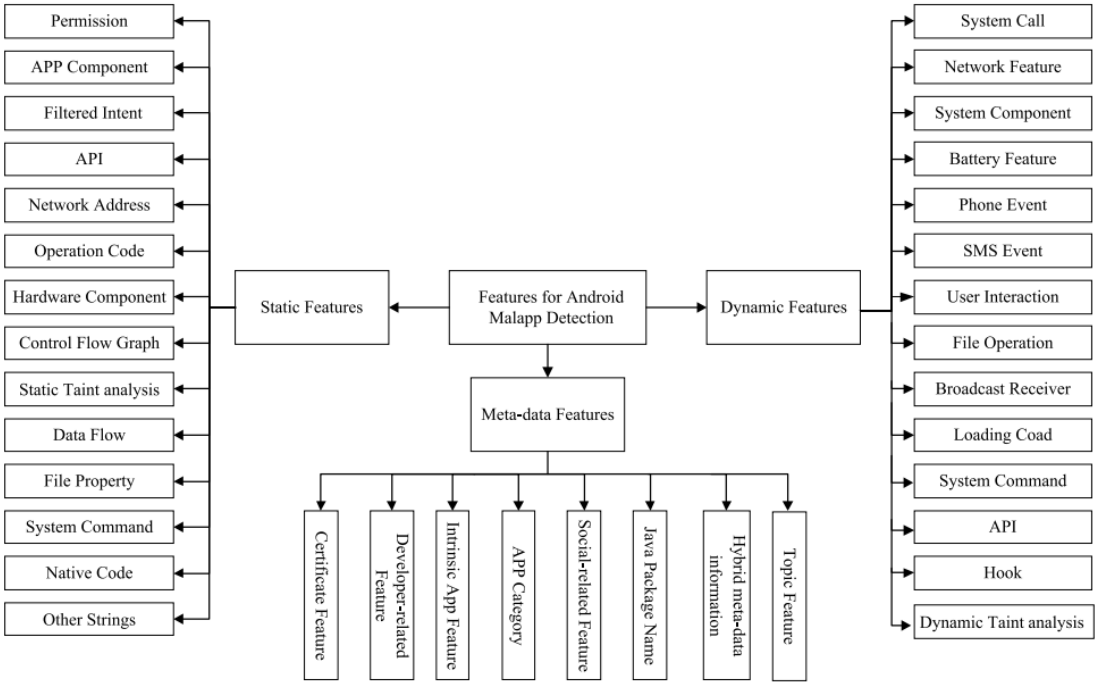

Figura 4.1: Tipos de características e subtipos de cada categoria [Fonte: Wang et al. (2019)].

Resumimos as principais características estáticas presentes nos datasets utilizados para os experimentos a seguir:

- **Permissões**: São privilégios especiais que os aplicativos devem pedir e ter aprovação para usar funcionalidades e recursos do sistema operacional e do dispositivo. Por definição, todas as permissões que um aplicativo precisa ter para acessar partes protegidas do sistema ou de outros aplicativos, ou as permissões que outros aplicativos precisam ter para acessar conteúdo desse aplicativo, são declaradas no arquivo de manifesto. O Android possui três tipos de permissões (de instalação, de execução e especiais), a fim de refletir o escopo de dados restritos que o aplicativo pode acessar e o escopo de ações restritas que o aplicativo pode realizar.

- **Chamadas de API (*API Calls*)**: Pode ser entendida como uma forma de comunicação entre sistemas, em que um deles fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade do conhecimento de detalhes da implementação.

- **Intenções (*Intents*)**: São mensagens assíncronas que permitem aos componentes

de um aplicativo solicitar funcionalidades de outros componentes do Android (do mesmo aplicativo ou não), enviando objetos de intenção. É considerado um mecanismo de segurança para impedir que aplicativos tenham acesso direto a outros aplicativos. Por isso, os aplicativos devem ter permissões específicas para usar *Intents*.

- **Componentes do Aplicativo (*App Components*)**: São os blocos de construção básicos de um aplicativo Android. Eles são os pontos de entrada para o sistema Android acessar aplicativos. Cada componente existe como uma entidade distinta e desempenha uma função específica. Esses componentes são vinculados pelo arquivo de manifesto do aplicativo.
- **Comandos do Sistema (*System Commands*)**: É uma diretiva de um programa para executar uma tarefa específica e podem ser extraídos dos arquivos `.smali` dos APKs.
- **Códigos de Operação (*OpCodes*)**: É uma instrução única (atômica) que pode ser executada pela CPU. No caso do Android, *OpCodes* são as instruções da máquina virtual para execução do aplicativo.

4.1.2 Conjuntos de Dados (*Datasets*)

Para avaliar as soluções propostas e compará-las com o método original e outros métodos de seleção, esta pesquisa utilizou *datasets* empregados para treino de modelos de detecção de *malwares* Android em outras pesquisas acadêmicas e disponibilizados publicamente. A Tabela 4.1 apresenta a composição dos *datasets* utilizados.

4.1.3 Métricas de Avaliação

Tradicionalmente, métodos de classificação são avaliados com base em: **Verdadeiros Positivos (TP)**, amostras positivas reais que são corretamente previstas como positivas; **Falsos Positivos (FP)**, amostras negativas reais que são incorretamente previstas como positivas; **Verdadeiros Negativos (TN)**, amostras negativas reais que são corretamente previstas como negativas; e **Falsos Negativos (FN)**, amostras positivas reais que são incorretamente previstas como negativas.

Os valores de TP, FP, TN e FN são usados em uma matriz de confusão, ferramenta para analisar o quão bem um método de classificação pode reconhecer as amostras de diferentes classes. Os valores TP e TN nos indicam quando um método está classificando corretamente, enquanto os valores FP e FN quando está classificando erroneamente.

Tabela 4.1: *Resumo de informações dos Datasets.*

Dataset	**Características**		**Amostras**		
	Qtde.	**Tipos**	**Maliciosas**	**Benignas**	**Total**
ADROIT	166	P	3418	8058	11476
AndroCrawl	141	A(26), I(8), P(84), O(23)	10170	86574	96744
Android Permissions	151	P	17787	9077	26864
DefenseDroid PRS	2877	P(1489), I(1388)	6000	5975	11975
DREBIN-215	215	A(73), P(113), S(6), I(23)	5555	9476	15031
KronoDroid Emulator	276	P(145), A(123), O(8)	28745	35246	63991
KronoDroid Real Devices	286	P(146), A(100), O(40)	41382	36755	78137
MH-100K	24837	P(166), A(24417), I(250), O(4)	9800	92175	101975

[P] Permissões, [A] Chamadas de API, [I] Intenções, [S] Comandos do Sistema, [O] Outros

É com base nos valores TP, FP, TN e FN que as métricas de avaliação podem ser calculadas, a partir da matriz de confusão, da seguinte forma:

- **Acurácia (ACC)**: proporção de amostras que são corretamente classificadas. Quanto maior o valor de ACC, melhor será o efeito da classificação.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

- **Precisão (PREC)**: proporção das amostras classificadas como *malwares* e que de fato são maliciosas.

$$PREC = \frac{TP}{TP + FP} \tag{4.2}$$

- **Revocação (REC)**: proporção de amostras maliciosas que são corretamente classificadas. Quanto mais alto for o valor de *recall*, melhor será o efeito da classificação.

$$REC = \frac{TP}{TP + FN} \tag{4.3}$$

- **F1 Score (F1)**: representa a média harmônica de *recall* e precisão.

$$F1 = 2 \times \frac{PREC \times REC}{PREC + REC} \tag{4.4}$$

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

4.1.4 Métodos e Parâmetros

Para avaliar o desempenho dos métodos propostos, foram selecionados algoritmos representativos de cada uma das principais categorias de seleção de características (filtro, *embedded* e *wrapper*). Esses algoritmos, amplamente utilizados na literatura, incluem: LASSO, RFE, teste qui-quadrado, Algoritmo Genético (GA) e o *Artificial Bee Colony* (ABC).

Para o método Algoritmo Genético (GA) foram utilizados os seguintes parâmetros na sua execução:

- **Tamanho do Indivíduo:** Número de características no conjunto de dados de treino que o indivíduo no algoritmo genético representa. Cada característica pode ser incluída ou excluída (representada por 1 ou 0) no modelo.
- **Função de Avaliação:** Métrica utilizada para avaliar a qualidade dos indivíduos na população. Foi utilizada a acurácia do classificador.
- **Cruzamento:** Operação que combina dois indivíduos para gerar novos indivíduos. Foi utilizado o ‘cxTwoPoint‘ (Cruzamento de dois pontos).
- **cxpb**: Probabilidade de cruzamento. Define a chance de dois indivíduos serem cruzados. Foi utilizado o valor de 0,5 (50%).
- **Mutação:** Operação que altera aleatoriamente bits em um indivíduo para manter a diversidade na população. Foi utilizado ‘mutFlipBit‘ (Inversão de bits individuais).
- **mutpb**: Probabilidade de mutação. Foi utilizado o valor de 0,2 (20%).
- **indpb**: Probabilidade de um gene ser mutado. Foi utilizado o valor de 0,05 (5%).
- **Seleção:** Método de seleção de indivíduos para a próxima geração. Foi utilizado o ‘selTournament‘ (Torneio).

- **Tamanho do Torneio (tournsize):** Número de indivíduos competindo em cada torneio para seleção. Foi utilizado o valor 3.
- **Número de gerações (ngen):** Define quantas vezes o algoritmo genético deve iterar sobre a população para evoluir soluções. Foi utilizado o valor 10.
- **Tamanho da População:** Número total de indivíduos na população em cada geração. Foi utilizado o valor 10.
- **Classificador:** Classificador utilizado para avaliar o desempenho das características selecionadas. Foram utilizados o Random Forest e XGBClassifier.
- **Métrica:** Medida utilizada para avaliar o desempenho do modelo. Foi utilizada a acurácia.

Para o método LASSO foi utilizado o parâmetro **alpha**, que controla a força da penalização para a magnitude dos coeficientes das características. Um valor menor significa menos regularização, e um valor maior aumenta a regularização. Foi utilizado o valor de 0,005. Para o método RFE foi utilizado o parâmetro **classificadores**, os modelos usados para treinar e avaliar o desempenho das características selecionadas. Foram utilizados RandomForestClassifier, XGBClassifier. Já para o método qui-quadrado foi utilizado o parâmetro **k**, que representa o número de melhores características a serem selecionadas. Foram selecionadas 50% do total de características.

Para a execução do algoritmo ABC binário e dos métodos propostos, foram utilizados os seguintes parâmetros:

- **Iteração**: Define o número de iterações que o algoritmo ABC irá realizar. Foi utilizado o número de 30 iterações.
- **Classificador**: Classificador utilizado para avaliar o desempenho das características selecionadas. Foram utilizados o Random Forest e XGBClassifier.
- **Métrica**: Medida utilizada para avaliar o desempenho do modelo. Foi utilizada a F1-score.
- **Limite máximo de tentativas**: Este parâmetro define o número máximo de tentativas permitidas para melhorar uma solução antes que uma solução seja considerada “esgotada” e seja trocada por outra. O valor adotado para esse parâmetro foi de 10 iterações.

- **MR (modification rate)**: A taxa de modificação define a probabilidade de uma abelha modificar uma característica ao tentar melhorar uma solução. Os valores adotados foram de 0,1 e 0,01.

4.1.5 Validação do Método

Para validação do método, utilizamos da técnica de validação cruzada, que consiste em dividir o conjunto de dados em K partições (*folds*). A função de predição é aprendida usando $K-1$ partições, e a partição deixada de fora é usada para teste. Como o número de amostras nos conjuntos de dados geralmente é desbalanceado, optamos por utilizar sua variação estratificada, pois mantém a proporção original de cada classe na etapa de teste. Geralmente, realiza-se a validação cruzada usando $K = 5$ ou $K = 10$, uma vez que estes valores produzem estimativas de taxa de erro de teste que não sofrem de viés ou variância excessivamente altos (James et al., 2013). Por padrão, a biblioteca scikit-learn (Pedregosa et al., 2011) utiliza $K = 5$, e por isso decidimos manter este valor.

Qualquer conjunto de dados que apresente uma distribuição desigual entre suas classes pode ser considerado desbalanceado (He e Garcia, 2009). No entanto, a percepção comum é que conjuntos de dados desbalanceados exibem desequilíbrios significativos e, em alguns casos, extremos, não sendo incomuns os desequilíbrios de 100:1, 1.000:1 e 10.000:1.

4.2 Avaliação de Estabilidade

Para avaliar a estabilidade dos métodos de seleção de características, foram considerados dois cenários distintos. Em ambos, utilizou-se o dataset Adroit, escolhido por ser o menor entre os disponíveis, dado que os testes requerem uma elevada demanda de recursos computacionais.

No primeiro cenário, os métodos foram aplicados ao *dataset* Adroit sem qualquer alteração, com o objetivo de medir a aleatoriedade interna dos métodos, ou seja, a consistência dos resultados ao longo de várias execuções.

No segundo cenário, os métodos foram executados no *dataset* Adroit utilizando a técnica de *Bootstrap Sampling*, que envolve a reamostragem com reposição a partir do conjunto de dados original. Nesse caso, a reamostragem foi utilizada para introduzir uma perturbação no dataset e observar como essa variação impacta a estabilidade dos métodos.

Em cada um dos cenários, os métodos foram executados 20 vezes para assegurar resultados mais confiáveis. Para o cálculo da estabilidade do algoritmo ABC e das variantes propostas, foi utilizado o coeficiente de Sorensen-Dice.

O Coeficiente de Sorensen-Dice, também conhecido como Índice de Dice, é uma métrica usada para medir a similaridade entre dois conjuntos e é amplamente utilizado para medir a estabilidade de métodos de seleção de características. Ele é definido pela fórmula:

$$D(S_i, S_j) = \frac{2|S_i \cap S_j|}{|S_i| + |S_j|}$$

onde:

- $|S_i \cap S_j|$ é o número de características comuns entre os subconjuntos S_i e S_j,
- $|S_i|$ e $|S_j|$ são os tamanhos dos subconjuntos S_i e S_j, respectivamente.

O Coeficiente de Sorensen-Dice varia de 0 a 1, onde:

- 0 indica que os subconjuntos são completamente dissimilares (nenhuma característica em comum),
- 1 indica que os subconjuntos são idênticos (todas as características são comuns).

Para obter uma medida geral, calculamos a média dos coeficientes de Sorensen-Dice para todos os pares de subconjuntos:

1. Dado um conjunto de subconjuntos $\{S_1, S_2, \ldots, S_n\}$, calcule o Coeficiente de Sorensen-Dice para cada par de subconjuntos (S_i, S_j), onde $i < j$. A fórmula para cada par é:

$$D(S_i, S_j) = \frac{2|S_i \cap S_j|}{|S_i| + |S_j|}$$

2. Calcule a média dos coeficientes para todos os pares de subconjuntos para obter uma medida geral de estabilidade:

$$D_{\text{médio}} = \frac{2}{\binom{n}{2}} \sum_{1 \leq i < j \leq n} D(S_i, S_j)$$

onde $\binom{n}{2} = \frac{n(n-1)}{2}$ é o número total de pares possíveis de subconjuntos.

4.2.1 Ambiente de Avaliação

A execução do experimento foi conduzida em um servidor com processador Intel Xeon E5649 @ 2.53GHz (2x6 núcleos) e 94GB RAM e HD. O sistema operacional utilizado foi o Linux Mint 20.3 Una e o Kernel 5.4.0-136-generic x86 64. Para a implementação e automação da execução dos métodos, utilizamos a linguagem Python (versão 3.10.12) e as bibliotecas NumPy (versão 1.23.5), Pandas (versão 1.5.3) e scikit-learn (versão 1.1.1).

4.3 Resultados

Com o objetivo de avaliar os métodos propostos e compará-los com o algoritmo ABC binário, conduzimos uma série de testes variando o parâmetro MR. Além disso, para avaliar o conjunto de dados resultante da seleção de características, empregamos dois classificadores distintos: *Random Forest* e XGBoost.

Os dois métodos em análise foram nomeados como "COR-ABC" para referenciar o algoritmo ABC aplicado com o coeficiente de correlação de Pearson e "Jaccard-ABC" para o algoritmo ABC utilizando a medida de similaridade de Jaccard.

4.3.1 Adroit

A Tabela 4.2 exibe os resultados obtidos para o conjunto de dados Adroit.

Tabela 4.2: Resultados para o dataset Adroit

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,9107	0,9039	0,7836	0,8395	0 %
	XGBoost	0,9116	0,9242	0,7661	0,8377	0 %
Cor-ABC, MR = 0,01	Random Forest	0,9055	0,9284	0,7398	0,8234	51,81 %
	XGBoost	0,9133	0,9307	0,7661	0,8404	38,55 %
Jaccard-ABC, MR = 0,01	Random Forest	0,9068	0,8815	0,7939	0,8354	50,60 %
	XGBoost	0,9085	0,9263	0,7529	0,8306	42,77 %
ABC, MR = 0,1	Random Forest	0,9125	0,9215	0,7719	0,8401	36,14 %
	XGBoost	0,9133	0,9292	0,7675	0,8407	15,66 %
RFE	Random Forest	0,8907	0,8676	0,7471	0,8028	93,98 %
	XGBoost	0,9245	0,9221	0,915	0,9183	93,98 %
LASSO	Random Forest	0,9011	0,9255	0,7266	0,8141	91,57%
	XGBoost	0,9007	0,9238	0,7266	0,8134	91,57%
GA	Random Forest	0,9103	0,933	0,7529	0,8333	46,99 %
	XGBoost	0,9107	0,9224	0,7646	0,8361	48,19 %
qui-quadrado	Random Forest	0,9046	0,9029	0,7617	0,8263	50,00 %
	XGBoost	0,9081	0,9324	0,7456	0,8286	50,00 %

Ao analisar os dados de execução dos métodos de seleção de características, várias tendências e comparações surgem, destacando o desempenho em termos de Acurácia, Precisão, *Recall*, F1-Score, e redução dimensional.

Os métodos Cor-ABC e ABC com XGBoost alcançaram as maiores acurácias. O Cor-ABC apresentou uma acurácia elevada tanto com Random Forest (0,9055) quanto com XGBoost (0,9133). Além disso, a redução dimensional foi significativa (51,81% com RF e 38,55% com XGBoost), indicando que o método é eficiente em reduzir características mantendo altas métricas de desempenho. O método ABC também obteve acurácias similares com ambos os classificadores, com diferenças pouco significativas.

O RFE foi o método com a menor acurácia, ligeiramente acima de 0,89. Os demais métodos apresentaram acurácias próximas, em torno de 0,90, mostrando um desempenho consistente.

Na métrica de Precisão, o método GA obteve o maior valor (0,933), seguido pelo qui-quadrado (0,9324). O RFE com Random Forest registrou o menor valor de precisão (0,8676). De modo geral, os métodos propostos mantiveram uma precisão competitiva, acima de 0,9, com exceção do Jaccard-ABC com Random Forest (0,8815). Fora esse caso, não houve diferenças significativas entre os métodos propostos e o ABC na métrica de precisão.

Para a métrica *Recall*, o Jaccard-ABC destacou-se com o maior valor (0,7939), mostrando um desempenho muito superior aos outros métodos. Com exceção dos casos Cor-ABC com Random Forest e Jaccard-ABC com XGBoost, os métodos propostos tiveram desempenho igual ou superior ao ABC. O pior desempenho em recall foi observado com o RFE e XGBoost (0,7018). Nessa métrica, houve uma maior variação nos valores, variando entre 0,70 e 0,79.

Na métrica F1-Score, o maior valor foi obtido pelo método ABC (0,8407), seguido de perto pelo Cor-ABC (0,8404). O pior desempenho foi do RFE (0,7993). A variação dos valores de F1 foi menor, situando-se no intervalo entre 0,79 e 0,84.

Em termos de redução do número de características, os métodos LASSO e RFE alcançaram as maiores reduções, acima de 90%, enquanto mantinham acurácia competitiva. Os métodos GA, Cor-ABC, Jaccard-ABC, e qui-quadrado apresentaram reduções mais moderadas (50%), oferecendo um bom equilíbrio entre redução de características e desempenho. O método ABC apresentou as menores reduções, com 36,14% para Random Forest e 15,66% para XGBoost.

Em geral, os resultados com a aplicação dos métodos de seleção de características são superiores aos resultados do classificadores treinados sem seleção.

4.3.2 Androcrawl

A Tabela 4.3 apresenta os resultados obtidos para o conjunto de dados Androcrawl.

Tabela 4.3: Resultados para o *dataset* Androcrawl

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,9854	0,9497	0,909	0,9289	-
	XGBoost	0,9861	0,9415	0,9253	0,9333	-
ABC, MR = 0,01	Random Forest	0,9856	0,9479	0,913	0,9301	27,66 %
	XGBoost	0,9865	0,9488	0,9208	0,9346	13,48 %
Cor-ABC, MR = 0,01	Random Forest	0,9838	0,9512	0,8918	0,9206	14,18 %
	XGBoost	0,9859	0,9467	0,9174	0,9318	13,48 %
Jaccard-ABC, MR = 0,01	Random Forest	0,9841	0,9417	0,9046	0,9228	45,39 %
	XGBoost	0,9849	0,9386	0,9164	0,9274	51,77 %
RFE	Random Forest	0,9833	0,9349	0,9036	0,919	92,91 %
	XGBoost	0,9814	0,9687	0,9304	0,9485	92,91%
LASSO	Random Forest	0,9801	0,9464	0,8594	0,9008	96,45%
	XGBoost	0,9801	0,9464	0,8594	0,9008	96,45%
GA	Random Forest	0,9865	0,9539	0,9154	0,9343	46,81%
	XGBoost	0,9849	0,9385	0,9159	0,9271	51,77 %
qui-quadrado	RF	0,985	0,9435	0,9115	0,9272	50,35%
	XGBoost	0,9861	0,9482	0,9179	0,9328	50,35 %

O método Cor-ABC demonstrou alto desempenho em termos de acurácia, com 0,9838 usando Random Forest e 0,9859 com XGBoost. A precisão e *recall* foram balanceados, mas com uma ligeira variação, destacando um *recall* um pouco inferior com Random Forest (0,8918) em comparação ao XGBoost (0,9174). A redução de características foi significativa, especialmente com Random Forest (14,18%).

Aplicando a técnica LASSO, tanto o Random Forest quanto o XGBoost alcançaram a mesma acurácia de 0,9801. No entanto, a grande vantagem desse método foi a redução extrema das características, com 96,45%, mantendo uma boa precisão (0,9464) e *recall* (0,8594), mas com um F1 de 0,9008, indicando uma ligeira perda em comparação a métodos como ABC.

O RFE, ao selecionar apenas 10 características, mostrou uma acurácia levemente superior com XGBoost (0,9833) em comparação com Random Forest (0,9813). A precisão foi maior com Random Forest (0,9529), mas o XGBoost teve um recall ligeiramente superior (0,9036), o que se refletiu em um F1 mais alto para Random Forest (0,907).

O método GA se destacou com a maior acurácia (0,9865) e uma excelente precisão (0,9539) e *recall* (0,9154) com Random Forest, conseguindo uma redução de 46,81% nas características, o que reflete um bom equilíbrio entre redução dimensional e desempenho preditivo.

O método qui-quadrado apresentou resultados consistentes com uma acurácia de 0,985 para Random Forest e 0,9861 para XGBoost. A redução de 50,35% das características não comprometeu a precisão e *recall*, resultando em valores F1 acima de 0,93 para ambos os classificadores.

O método Jaccard-ABC mostrou uma acurácia ligeiramente inferior comparada ao Cor-ABC, com 0,9841 para Random Forest e 0,9849 para XGBoost. A redução de características foi substancial, especialmente para XGBoost (51,773%), mantendo um equilíbrio razoável entre precisão e *recall*.

Por fim, o ABC apresentou resultados notáveis com uma acurácia de 0,9856 para Random Forest e 0,9865 para XGBoost, aliado a uma redução moderada das características (27,6596% para RF e 13,4752% para XGBoost). O F1 foi um dos mais altos, especialmente com XGBoost (0,9346), indicando um bom equilíbrio entre precisão, *recall* e eficiência na redução de dimensionalidade.

4.3.3 Android Permissions

A Tabela 4.4 apresenta os resultados obtidos para o conjunto de dados Android Permissions.

Tabela 4.4: Resultados para o dataset Android Permissions

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,6551	0,6829	0,8946	0,7745	-
	XGBoost	0,6696	0,6815	0,9407	0,7904	-
ABC, MR = 0,01	Random Forest	0,6691	0,6753	0,9637	0,7941	47,02 %
	XGBoost	0,6657	0,674	0,9592	0,7917	40,4 %
Cor-ABC, MR = 0,01	Random Forest	0,6657	0,6802	0,9348	0,7874	21,85 %
	XGBoost	0,6693	0,678	0,9533	0,7924	17,9 %
Jaccard-ABC, MR = 0,01	Random Forest	0,6685	0,6758	0,9601	0,7932	54,30 %
	XGBoost	0,6717	0,6764	0,9668	0,7959	49,67 %
RFE	Random Forest	0,66	0,672	0,9503	0,7873	93,38 %
	XGBoost	0,6642	0,6675	0,9823	0,7949	93,38 %
LASSO	Random Forest	0,6607	0,6631	0,9913	0,7946	96,03%
	XGBoost	0,6598	0,6629	0,9893	0,7939	96,03%
GA	Random Forest	0,6687	0,6765	0,9578	0,7929	50,99%
	XGBoost	0,6709	0,6808	0,9472	0,7922	43,05 %
qui-quadrado	Random Forest	0,6613	0,6837	0,9089	0,7804	50,33%
	XGBoost	0,6687	0,6801	0,9435	0,7904	50,33 %

O método Cor-ABC apresentou uma acurácia moderada com 0,6657 usando RF e 0,6693 com XGBoost. Notavelmente, o *recall* foi extremamente alto, especialmente com XGBoost (0,9533), indicando que a maioria das instâncias relevantes foi corretamente

identificada. A redução de características foi de 21,85 % para RF e 17,88 % para XGBoost, mostrando um equilíbrio razoável entre a preservação de características relevantes e a eficácia preditiva.

Aplicando a técnica LASSO, ambos os classificadores alcançaram resultados similares, com uma acurácia ligeiramente inferior em comparação a outros métodos (0,6607 para RF e 0,6598 para XGBoost). Entretanto, o *recall* foi excepcionalmente alto (0,9913 para RF e 0,9893 para XGBoost), sugerindo uma alta sensibilidade do modelo. A redução foi a mais drástica, com 96,03% das características eliminadas, o que pode justificar a leve queda na acurácia.

Com o método RFE, o XGBoost alcançou uma acurácia de 0,6642, ligeiramente superior à do RF (0,66). Ambos os classificadores mantiveram um *recall* elevado, acima de 0,9503, enquanto a redução de características foi significativa (93,38%), evidenciando a capacidade do RFE de selecionar um conjunto mínimo de características sem comprometer muito o desempenho.

O método GA se destacou por equilibrar bem a acurácia (0,6687), precisão (0,6765) e *recall* (0,9578) com RF. Com uma redução de 50,99% das características, esse método se mostrou eficaz tanto em termos de desempenho preditivo quanto na simplificação do modelo.

O método qui-quadrado demonstrou resultados consistentes, com uma acurácia de 0,6613 para RF e 0,6687 para XGBoost. O *recall* foi alto, especialmente com XGBoost (0,9435), e a redução de características foi significativa em 50,33%. Este método se mostrou uma boa opção quando se busca um compromisso entre redução e desempenho.

Utilizando o Jaccard-ABC, o classificadores apresentaram uma acurácia sólida, com 0,6685 para RF e 0,6717 para XGBoost. O *recall* foi extremamente elevado, superando 0,9601 para RF e 0,9668 para XGBoost, o que se refletiu em um bom F1. A redução de características foi notável, com mais de 49% em ambos os casos, sugerindo uma eficiente eliminação de redundâncias.

O método ABC obteve uma acurácia de 0,6691 com RF e 0,6657 com XGBoost, mostrando resultados consistentes. O *recall* permaneceu alto, acima de 0,9592, e a redução de características variou entre 47,02% para RF e 40,4% para XGBoost, demonstrando uma boa capacidade de manter a eficácia preditiva enquanto reduz a dimensionalidade do conjunto de dados.

Em suma, os resultados indicam que, embora a acurácia não varie amplamente entre os métodos, o *recall* e a redução de características são fortemente influenciados pela técnica de seleção aplicada. O LASSO, apesar de proporcionar uma enorme redução de características, manteve um muito elevado, enquanto métodos como ABC-Jaccard e GA

mostraram um bom equilíbrio entre acurácia e eficiência na redução de características.

4.3.4 Drebin

A Tabela 4.5 apresenta os resultados obtidos para o conjunto de dados Drebin.

Tabela 4.5: Resultados para o dataset Drebin

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,9874	0,9891	0,9766	0,9828	-
	XGBoost	0,99	0,9909	0,982	0,9864	-
ABC, MR = 0,01	Random Forest	0,9907	0,9918	0,9829	0,9873	35,81 %
	XGBoost	0,99	0,9909	0,982	0,9864	11,63 %
Cor-ABC, MR = 0,01	Random Forest	0,979	0,9781	0,9649	0,9715	44,65 %
	XGBoost	0,9681	0,9721	0,9406	0,9561	50,7 %
Jaccard-ABC, MR = 0,01	Random Forest	0,9857	0,989	0,9721	0,9805	53,02 %
	XGBoost	0,9924	0,991	0,9883	0,9896	4,65 %
RFE	Random Forest	0,9245	0,9146	0,8776	0,8957	95,35 %
	XGBoost	0,9445	0,9338	0,9145	0,9241	95,35 %
LASSO	Random Forest	0,984	0,9793	0,9775	0,9784	75,81%
	XGBoost	0,9844	0,9828	0,9748	0,9788	75,81%
GA	Random Forest	0,9894	0,9936	0,9775	0,9855	44,65%
	XGBoost	0,987	0,9855	0,9793	0,9824	45,12 %
qui-quadrado	Random Forest	0,984	0,9836	0,973	0,9783	50,23%
	XGBoost	0,982	0,9844	0,9667	0,9755	50,23 %

Para o *dataset* Drebin, observa-se um bom desempenho dos classificadores sem seleção de características. No entanto, os classificadores treinados com métodos de seleção de características apresentaram desempenhos comparáveis e, em alguns casos, superiores aos modelos sem seleção.

Dentre os melhores desempenhos, destacam-se os métodos GA, Jaccard-ABC e ABC, que obtiveram métricas superiores aos demais métodos e apresentaram resultados muito próximos entre si, demonstrando um bom equilíbrio entre as métricas de desempenho e a redução no número de características.

O método Jaccard-ABC com o classificador XGBoost alcançou a melhor acurácia, com um valor de 0,9924. Desempenhos bastante próximos foram observados para os métodos ABC com Random Forest (0,9907), ABC com XGBoost (0,99) e GA com Random Forest (0,9894). O pior desempenho foi do método RFE, que obteve apenas 0,9445 de acurácia com XGBoost e 0,9245 com Random Forest. O método LASSO, por sua vez, manteve um bom desempenho, atingindo 0,984 de acurácia.

Na métrica de precisão, os métodos ABC, Jaccard-ABC e GA obtiveram os melhores resultados, com valores muito próximos entre si e diferenças pouco significativas. O

método Jaccard-ABC com XGBoost se destacou, alcançando uma precisão de 0,991. Em contrapartida, o método RFE apresentou o pior desempenho em precisão, com um valor de 0,9146 para o classificador Random Forest. Os métodos qui-quadrado e LASSO também tiveram bons resultados, embora ligeiramente inferiores aos métodos ABC, Jaccard-ABC e GA.

Para as métricas de *recall* e F1, novamente os melhores resultados foram obtidos pelos métodos ABC, Jaccard-ABC e GA. O método Jaccard-ABC com XGBoost alcançou os melhores resultados em ambas as métricas, com valores de 0,9883 para *recall* e 0,9896 para F1.

O método que mais reduziu o número de características foi o RFE, com uma redução de 95,35%. No entanto, essa alta redução impactou negativamente os resultados das métricas dos classificadores. Por outro lado, o método LASSO conseguiu manter um bom equilíbrio entre alta redução e boas métricas, reduzindo o número de características em 75,81%. Os métodos GA, Jaccard-ABC e Cor-ABC também apresentaram um bom equilíbrio entre redução e métricas, com o Jaccard-ABC se destacando ao alcançar os maiores valores nas métricas. Embora o método ABC tenha obtido bons resultados nas métricas, foi o que menos reduziu o número de características, com a menor redução sendo de apenas 11,62%.

4.3.5 DefenseDroid PRS

A Tabela 4.6 apresenta os resultados obtidos para o conjunto de dados DefenseDroid PRS.

O método Cor-ABC apresentou uma alta acurácia tanto para RF (0,9294) quanto para XGBoost (0,924), com uma leve vantagem para RF. A precisão e o *recall* foram bem balanceados, resultando em valores de F1 consistentes (0,9295 para RF e 0,9226 para XGBoost). A redução de características foi moderada, em torno de 15%, o que indica que o método consegue manter um bom desempenho preditivo enquanto elimina redundâncias.

O método LASSO, conseguiu uma redução de 99,17%. No entanto, essa drástica redução parece ter um impacto nos resultados preditivos, especialmente na acurácia (0,9019 para RF e 0,9102 para XGBoost). Ainda assim, a precisão e o recall foram razoavelmente altos.

O método RFE, ao selecionar apenas 10 características, resultou na maior redução entre os métodos analisados (99,65%). Contudo, essa drástica redução resultou em uma acurácia inferior (0,8806 para ambos os classificadores), destacando um compromisso

Tabela 4.6: Resultados para o dataset DefenseDroid PRS

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,9303	0,9337	0,9267	0,9302	-
	XGBoost	0,9232	0,9394	0,905	0,9219	-
ABC, MR = 0,01	Random Forest	0,9315	0,9309	0,9325	0,9317	16,96 %
	XGBoost	0,9311	0,9473	0,9133	0,93	4,69 %
Cor-ABC, MR = 0,01	Random Forest	0,9294	0,9307	0,9283	0,9295	15,61 %
	XGBoost	0,924	0,9426	0,9033	0,9226	15,43 %
Jaccard-ABC, MR = 0,01	Random Forest	0,9273	0,9311	0,9233	0,9272	48,77 %
	XGBoost	0,9269	0,9445	0,9075	0,9256	50,09 %
RFE	Random Forest	0,8806	0,8926	0,8658	0,879	99,65 %
	XGBoost	0,8806	0,8662	0,9008	0,8832	99,65 %
LASSO	Random Forest	0,9019	0,9092	0,8933	0,9012	99,17%
	XGBoost	0,9102	0,9257	0,8925	0,9088	99,17%
GA	Random Forest	0,9269	0,9422	0,91	0,9258	49,88%
	XGBoost	0,9211	0,9415	0,8983	0,9194	50,96%
qui-quadrado	Random Forest	0,9253	0,9428	0,9058	0,9239	50,02%
	XGBoost	0,9157	0,9332	0,8958	0,9141	50,02 %

significativo entre a simplicidade do modelo e a acurácia. Apesar disso, o F1 manteve-se relativamente elevado, indicando que o RFE ainda preserva uma boa capacidade de generalização, mesmo com um conjunto de características extremamente reduzido.

O método GA demonstrou um bom equilíbrio entre desempenho e redução, com uma acurácia de 0,9269 e uma redução de cerca de 50%. A precisão foi especialmente alta com RF (0,9422), o que se traduziu em um valor F1 sólido de 0,9258. Este método é uma boa escolha quando se busca um equilíbrio entre a redução de características e a manutenção de uma alta acurácia.

O qui-quadrado apresentou uma acurácia de 0,9253 com RF e 0,9157 com XGBoost, junto com uma redução de 50%. A precisão e o recall foram consistentes, resultando em valores F1 acima de 0,9141. Esse método é eficaz na manutenção de um alto desempenho preditivo enquanto elimina metade das características, sendo uma opção robusta para cenários onde a redução de dimensionalidade é importante sem sacrificar muito a acurácia.

O Jaccard-ABC conseguiu uma acurácia de 0,9273 com RF e 0,9269 com XGBoost, com uma redução de cerca de 50%. A precisão foi particularmente elevada com XGBoost (0,9445), resultando em um F1 de 0,9256. Este método mostrou-se eficaz em preservar a qualidade preditiva, mesmo com uma redução substancial de características.

O método ABC se destacou com a maior acurácia entre todos os métodos, alcançando 0,9315 com RF e 0,9311 com XGBoost. Além disso, a precisão e o recall foram altos, resultando em F1s de 0,9317 (RF) e 0,93 (XGBoost). A redução de características foi

modesta com RF (16,9621%) e muito baixa com XGBoost (4,6924%), sugerindo que este método é particularmente eficaz em maximizar o desempenho preditivo quando a redução dimensional não é a principal prioridade.

4.3.6 KronoDroid Emulator

A Tabela 4.7 apresenta os resultados obtidos para o conjunto de dados KronoDroid Emulator.

Tabela 4.7: Resultados para o dataset KronoDroid Emulator

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,9719	0,9726	0,9645	0,9686	-
	XGBoost	0,9626	0,9637	0,9525	0,9581	-
ABC, MR = 0,01	Random Forest	0,9691	0,9688	0,9621	0,9654	26,45 %
	XGBoost	0,9615	0,9628	0,9509	0,9569	10,87 %
Cor-ABC, MR = 0,01	Random Forest	0,9674	0,9671	0,9602	0,9636	22,46 %
	XGBoost	0,9493	0,9528	0,9334	0,943	21 %
Jaccard-ABC, MR = 0,01	Random Forest	0,9624	0,9605	0,9556	0,9581	52,17 %
	XGBoost	0,9571	0,9541	0,9503	0,9522	44,56 %
RFE	Random Forest	0,9154	0,8954	0,9189	0,907	96,38%
	XGBoost	0,9029	0,858	0,9393	0,8968	96,38%
LASSO	Random Forest	0,9577	0,9556	0,9501	0,9528	90,22%
	XGBoost	0,9516	0,9511	0,9405	0,9458	90,22%
GA	Random Forest	0,964	0,966	0,9534	0,9596	48,91 %
	XGBoost	0,9548	0,958	0,9405	0,9492	47,46 %
qui-quadrado	Random Forest	0,9718	0,9706	0,9664	0,9685	50,00%
	XGBoost	0,9641	0,9658	0,9539	0,9598	50,00 %

O método Cor-ABC alcançou uma acurácia de 0,9674 e uma redução de 22,46% nas características com Random Forest. No entanto, ao aplicar o mesmo método com o XGBoost, observou-se uma ligeira queda na acurácia para 0,9493, com uma redução de 21% das características. Quando a seleção não foi aplicada, o Random Forest obteve uma acurácia maior, de 0,9719.

O LASSO se destacou pela sua capacidade de reduzir drasticamente o número de características (90,22%), mantendo uma acurácia relativamente alta com Random Forest (0,9577) e XGBoost (0,9516).

Por outro lado, o método RFE, ao selecionar apenas 10 características, mostrou-se menos eficaz em termos de acurácia, particularmente com o XGBoost (0,9029), apesar da redução expressiva de 96,38%. Essa observação sugere que, embora o RFE seja eficiente na redução do número de características, essa simplificação extrema pode levar à perda de informações cruciais, afetando negativamente o desempenho do modelo. Em

contrapartida, o método GA e o método Qui-Quadrado apresentaram um equilíbrio interessante entre redução e desempenho. O GA, com uma redução de 48,91%, conseguiu manter uma acurácia elevada de 0,964 com RF, comparável aos resultados obtidos com o Qui-Quadrado (0,9718 com RF e 50% de redução).

O método Jaccard-ABC também demonstrou boa performance, especialmente com Random Forest (acurácia de 0,9624 e redução de 52,17%), enquanto o ABC obteve uma acurácia de 0,9691 com Random Forest e uma redução mais modesta de 26,45

O método Cor-ABC, utilizando o Random Forest, obteve uma precisão de 0,9671, *recall* de 0,9602 e F1 de 0,9636, indicando um modelo equilibrado que não apenas classifica corretamente a maioria das amostras, mas também mantém uma baixa taxa de falsos positivos e falsos negativos. Entretanto, com o XGBoost, a precisão aumentou ligeiramente para 0,9528, mas com uma queda no *recall* para 0,9334 e F1 para 0,943, sugerindo que, embora o modelo seja preciso, ele perde alguma capacidade de detectar todos os verdadeiros positivos.

O LASSO apresentou resultados consistentes entre Random Forest e XGBoost. Com Random Forest, a precisão foi de 0,9556, o *recall* de 0,9501 e o F1 de 0,9528, enquanto com XGBoost, os valores foram um pouco inferiores, com uma precisão de 0,9511, recall de 0,9405 e F1 de 0,9458. Estes resultados indicam que o LASSO, apesar da alta redução de características, consegue manter um bom equilíbrio entre as métricas, tornando-o uma escolha robusta para modelos que necessitam de simplificação sem comprometer significativamente a performance.

O método RFE, que selecionou apenas 10 características, apresentou uma queda notável na precisão com o XGBoost (0,858), embora tenha mantido um *recall* alto (0,9393). Isso resultou em um F1 de 0,8968, o que sugere que o modelo estava tendendo a subestimar algumas classes, possivelmente devido à extrema redução das características. Com RF, a precisão foi melhor (0,8954), mas ainda assim, o *recall* e F1 foram mais baixos em comparação com outros métodos que mantiveram mais características, reforçando a ideia de que uma redução extrema pode levar à perda de informações críticas.

O GA demonstrou um desempenho equilibrado, com uma precisão de 0,966, *recall* de 0,9534 e F1 de 0,9596 utilizando Random Forest. Isso sugere que o GA é eficaz em manter tanto a precisão quanto a sensibilidade, mesmo após uma redução considerável nas características. Similarmente, o método Qui-Quadrado obteve uma precisão de 0,9706, *recall* de 0,9664 e F1 de 0,9685 com RF, confirmando sua capacidade de manter um desempenho robusto em termos de todas as métricas.

O método Jaccard-ABC apresentou uma precisão de 0,9605, *recall* de 0,9556 e F1 de 0,9581 com RF, resultados que mostram um bom equilíbrio entre a capacidade do

modelo de prever corretamente as classes e manter a sensibilidade. No entanto, com o XGBoost, as métricas foram ligeiramente inferiores, com uma precisão de 0,9541, *recall* de 0,9503 e F1 de 0,9522, ainda indicando um desempenho sólido, mas com uma leve tendência a reduzir a sensibilidade.

Finalmente, o método ABC com Random Forest alcançou uma precisão de 0,9688, recall de 0,9621 e F1 de 0,9654, enquanto com XGBoost, a precisão foi de 0,9628, recall de 0,9509 e F1 de 0,9569. Esses resultados demonstram que o ABC 0,1 oferece uma boa combinação de alta precisão e recall, o que resulta em uma pontuação F1 equilibrada, especialmente com RF.

4.3.7 KronoDroid Real Devices

A Tabela 4.8 apresenta os resultados obtidos para o conjunto de dados KronoDroid Real Devices.

Tabela 4.8: Resultados para o dataset KronoDroid Real Devices

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,976	0,9824	0,9721	0,9772	-
	XGBoost	0,9699	0,9763	0,9667	0,9715	-
ABC, MR=0,01	Random Forest	0,9741	0,9806	0,9704	0,9755	32,87 %
	XGBoost	0,9702	0,9769	0,9667	0,9718	14,68 %
Cor-ABC, MR=0,01	Random Forest	0,9575	0,9645	0,9549	0,9597	27,27 %
	XGBoost	0,9659	0,973	0,9623	0,9676	24,47 %
Jaccard-ABC, MR=0,01	Random Forest	0,9727	0,9767	0,9717	0,9742	45,45 %
	XGBoost	0,9685	0,9753	0,9648	0,9701	8,39 %
RFE	Random Forest	0,9249	0,9289	0,9294	0,9292	96,50 %
	XGBoost	0,9188	0,9314	0,914	0,9226	96,50 %
LASSO	Random Forest	0,9618	0,9691	0,9584	0,9637	89,86%
	XGBoost	0,9599	0,9659	0,9581	0,962	89,86%
GA	Random Forest	0,9725	0,9783	0,9696	0,9739	45,80%
	XGBoost	0,9663	0,9736	0,9624	0,968	44,76 %
Qui-Quadrado	Random Forest	0,9758	0,9809	0,9733	0,9771	50,00%
	XGBoost	0,9718	0,9775	0,9691	0,9732	50,00 %

O método Cor-ABC com Random Forest apresentou a acurácia de 0,9575, com uma precisão de 0,9645, recall de 0,9549 e F1 de 0,9597, juntamente com uma redução de 27,27% nas características. Utilizando XGBoost, o método superou ligeiramente os resultados do Random Forest, alcançando uma acurácia de 0,9659, precisão de 0,973, recall de 0,9623 e F1 de 0,9676, com uma redução de 24,48%. Sem seleção de características, os resultados foram melhores, especialmente com o Random Forest, onde a acurácia atingiu 0,976, precisão de 0,9824, recall de 0,9721 e F1 de 0,9772, embora a

redução de características não tenha sido especificada.

O método LASSO destacou-se por sua alta capacidade de redução de características, atingindo 89,86% de redução, ao mesmo tempo em que manteve uma acurácia de 0,9618 com Random Forest e 0,9599 com XGBoost. As demais métricas também foram consistentes, com F1 de 0,9637 e 0,962, respectivamente, sugerindo que o LASSO é eficaz em simplificar modelos sem sacrificar significativamente o desempenho.

O RFE, que selecionou apenas 10 características, apresentou resultados mais modestos. Com XGBoost, a acurácia foi de 0,9188, com precisão de 0,9314, recall de 0,914 e F1 de 0,9226. Já com Random Forest, a acurácia foi ligeiramente melhor, alcançando 0,9249, com precisão de 0,9289, recall de 0,9294 e F1 de 0,9292. Apesar da alta redução de 96,50%, esses resultados indicam que o método fez uma seleção extrema e eliminou informações importantes.

O GA demonstrou um bom equilíbrio entre redução e desempenho. Com Random Forest, obteve uma acurácia de 0,9725, precisão de 0,9783, recall de 0,9696 e F1 de 0,9739, com uma redução de 45,80% nas características. Estes resultados são comparáveis aos obtidos pelo método qui-quadrado, que alcançou uma acurácia de 0,9758 com Random Forest e 0,9718 com XGBoost, com uma redução de 50% em ambos os casos.

O método Jaccard-ABC obteve uma acurácia de 0,9727 com RF e 0,9685 com XGBoost. Com RF, as métricas foram bastante equilibradas, com precisão de 0,9767, recall de 0,9717 e F1 de 0,9742, e uma redução de 45,45%. No caso do XGBoost, embora a acurácia tenha sido ligeiramente inferior, a redução foi muito menor (8,39%).

Por fim, o método ABC alcançou uma acurácia de 0,9741 com Random Forest e 0,9702 com XGBoost, com reduções de 32,87% e 14,69%, respectivamente. As métricas de precisão, recall e F1 foram altas em ambos os casos, com destaque para a precisão de 0,9806 e F1 de 0,9755 com Random Forest.

4.3.8 MH-100k

A Tabela 4.9 apresenta os resultados obtidos para o conjunto de dados MH-100k. Devido a limitações computacionais e à pouca diferença entre os resultados obtidos, os métodos de seleção de características baseados em modelos (ABC, Jaccard-ABC, Cor-ABC e GA) foram executados exclusivamente com o classificador Random Forest. Além disso, o método RFE não foi incluído nas execuções devido ao seu elevado custo computacional.

Os resultados da execução dos diferentes métodos de seleção de características no conjunto de dados analisado revelam variações significativas tanto no desempenho dos classificadores quanto na redução de características. O melhor desempenho em termos

Tabela 4.9: Resultados para o dataset MH-100k

Método	**Classificador**	**Acurácia**	**Precisão**	***Recall***	**F1 Score**	**Redução (%)**
Sem Seleção	Random Forest	0,9758	0,8631	0,8963	0,8794	0%
	XGBoost	0,9774	0,8737	0,9007	0,887	0%
Cor-ABC, MR=0,01	Random Forest	0,9743	0,8636	0,8778	0,8706	54,79%
Jaccard-ABC, MR=0,01	Random Forest	0,9762	0,8704	0,8908	0,8805	49,84 %
ABC, MR=0,1	Random Forest	0,9763	0,8644	0,8998	0,8817	42,58%
LASSO	Random Forest	0,9728	0,8586	0,8658	0,8622	99,88%
	XGBoost	0,9727	0,8606	0,8623	0,8615	99,88%
GA	Random Forest	0,976	0,8741	0,8833	0,8787	49,56%
Qui-Quadrado	Random Forest	0,9757	0,8629	0,8948	0,8786	50,00%
	XGBoost	0,977	0,8722	0,8983	0,885	50,00%

de acurácia, recall e F1-score foi obtido pelo classificador XGBoost sem seleção de características, com valores de 0,9774 para acurácia, 0,9007 para recall e 0,887 para F1. Já a maior precisão foi alcançada pelo método GA com Random Forest, registrando 0,8741. Em relação à redução de características, o método LASSO se destacou, reduzindo o conjunto de características em 99,88%.

O método ABC, utilizando Random Forest (RF) como estimador, apresentou uma acurácia de 0,9763, com precisão de 0,8644, recall de 0,8998, F1-score de 0,8817 e uma redução significativa de 42,58% nas características. Esse método se destacou por equilibrar uma alta taxa de redução com métricas de desempenho próximas ao cenário sem seleção de características.

O Jaccard-ABC, uma variante do ABC, obteve uma acurácia de 0,9762, com uma precisão ligeiramente superior (0,8704), recall de 0,8908, F1-score de 0,8805 e uma redução ainda maior, de 49,849%, nas características. Isso sugere que a variante Jaccard consegue manter o desempenho enquanto aumenta a eliminação de características irrelevantes.

Já a variante Corr-ABC obteve uma acurácia de 0,9743, com valores um pouco menores de precisão (0,8636), recall (0,8778) e F1 (0,8706), mas com uma redução de 54,794% nas características, evidenciando que, apesar da maior eliminação de características, houve uma pequena perda de desempenho.

O método LASSO, ao utilizar o RF, alcançou uma acurácia de 0,9728, com uma redução substancial de 99,88% nas características, embora isso tenha resultado em métricas de precisão (0,8586), recall (0,8658) e F1 (0,8622) um pouco inferiores. Com XGBoost, os resultados foram similares, com acurácia de 0,9727 e uma ligeira melhora na precisão (0,8606) e recall (0,8623).

O método baseado no qui-quadrado, utilizando RF e XGBoost, apresentou bons

resultados de acurácia, sendo de 0,9757 e 0,977, respectivamente, com uma redução de 50% nas características. O XGBoost mostrou um leve aumento no desempenho das métricas de precisão, recall e F1 em relação ao RF.

Finalmente, a execução sem seleção de características demonstrou que, embora a acurácia seja comparável (0,9758 com RF e 0,9774 com XGBoost), a falta de redução resulta em modelos mais complexos e com maiores demandas computacionais.

4.4 Análise da Estabilidade

A Tabela 4.10 mostra os resultados da estabilidade dos métodos de seleção de características.

Tabela 4.10: Resultados para estabilidade

Método	**Sorensen-Dice** (Sem pertubação)	**Sorensen-Dice** (Com pertubação)
ABC	0,5194	0,5100
Cor-ABC	0,6751	0,7022
Jaccard-ABC	0,5148	0,5191
GA	0,5162	0,5132
LASSO	1	1
RFE	1	1
qui-quadrado	1	1

Ao analisar os resultados do estudo de estabilidade dos métodos de seleção de características utilizando a métrica Sorensen-Dice, podemos observar que, em geral, a estabilidade dos métodos se mantém quase inalterada nos dois cenários considerados, com exceção do método Cor-ABC, que demonstrou maior estabilidade no caso com perturbação. Os outros métodos apresentaram apenas pequenas variações.

Como esperado para métodos baseados em filtros e métodos embedded, LASSO, RFE e qui-quadrado mostraram uma estabilidade perfeita, evidenciando que esses métodos não são afetados por perturbações nas amostras.

No cenário sem perturbação, o método Cor-ABC alcançou a maior estabilidade (0,6751) entre os métodos baseados em *wrappers*. Outros métodos, como ABC, Jaccard-ABC e GA, exibiram uma estabilidade similar, em torno de 0,51 a 0,52.

Com a introdução de perturbação, o Cor-ABC manteve um bom desempenho, apresentando uma estabilidade de 0,7022, o que sugere que este método é relativamente robusto a mudanças na amostra entre os métodos baseados em *wrappers*. Em contraste, Jaccard-ABC, GA e ABC continuaram com uma estabilidade ligeiramente inferior, em torno de 0,52, similar aos resultados observados sem perturbação.

A leve diminuição na estabilidade observada para GA e ABC com perturbação sugere que esses métodos podem ser mais sensíveis a variações na amostra.

Essas observações indicam que o método Cor-ABC é o mais robusto entre os métodos baseados em *wrappers* avaliados, especialmente quando submetido a perturbações de amostra, enquanto outros métodos podem apresentar menor consistência.

4.5 Discussão dos Resultados Finais

Os resultados mostram que, de maneira geral, a seleção de características é uma etapa vantajosa, pois em vários cenários o desempenho dos classificadores com seleção superou aquele sem seleção, como nos datasets Adroit, Drebin, KronoDroid Real Device, Androcrawl, Android Permissions e DefenseDroid. Além disso, a maioria dos métodos obteve uma redução superior a 30% no número de características. O método RFE foi o que mais reduziu o número de características, exceto no dataset MH-100k, onde não foi executado, e nos datasets Androcrawl e Android Permissions, onde ficou atrás do LASSO.

É importante destacar que todos os métodos podem alcançar reduções extremas se os parâmetros forem ajustados corretamente. No entanto, os resultados indicam que o LASSO é o método que melhor equilibra redução e desempenho, mantendo métricas competitivas, ao contrário do RFE, que apresentou os piores resultados em termos de desempenho.

Os métodos propostos (Jaccard-ABC e Cor-ABC) superaram o ABC em termos de redução de características em vários cenários. O Jaccard-ABC, com Random Forest, reduziu mais características que o ABC em todos os datasets, enquanto o Cor-ABC superou o ABC em três datasets. No entanto, o Cor-ABC teve uma menor redução em alguns casos, pois sua seleção é baseada na correlação entre as características. Em datasets com menor correlação entre características ou com muitas variáveis, deve-se utilizar um valor de MR menor para obter uma redução mais significativa.

Quanto às métricas de desempenho, não há diferença significativa entre os métodos propostos e o ABC, com as métricas permanecendo muito próximas, com variações geralmente inferiores a 1%.

Em termos de estabilidade, o método Cor-ABC se destacou entre os métodos baseados em wrapper, mostrando-se o mais estável, com uma diferença significativa em relação ao ABC, o que indica uma menor aleatoriedade interna. Jaccard-ABC e ABC apresentaram níveis de estabilidade praticamente idênticos, junto com o GA. Esse resultado sugere que o mecanismo de exploração do Cor-ABC reduziu a aleatoriedade ao adotar o coeficiente de correlação de Pearson para a seleção de características, uma métrica mais estável. Embora o Jaccard-ABC também utilize uma métrica para correlação, o coeficiente de Jaccard é mais instável em comparação com o coeficiente de correlação de Pearson.

Uma observação importante é o impacto que o classificador escolhido pode ter nos resultados. Por exemplo, no dataset Adroit, os métodos ABC, Jaccard-ABC e Cor-ABC, quando combinados com XGBoost, apresentaram uma redução no número de características menor em comparação com o uso do Random Forest. O mesmo padrão foi observado no dataset Drebin para os métodos Jaccard-ABC e ABC, e essa tendência se repetiu em todos os datasets analisados. Isso evidencia que a escolha do classificador pode influenciar diretamente a eficácia da redução de características, mesmo utilizando os mesmos métodos de seleção.

Capítulo 5

Conclusões

A pesquisa de métodos de seleção de características desempenha um papel crucial no campo de detecção de *malware* Android, dada a vasta quantidade de características presentes nos conjuntos de dados. Esta dissertação abordou os conceitos fundamentais relacionados à seleção de características, obtendo informações relevantes e atualizadas de fontes confiáveis, incluindo artigos científicos e literatura especializada.

Adicionalmente, este trabalho propôs melhorias no algoritmo ABC, incorporando medidas de similaridade com o intuito de aprimorar a eficácia no contexto da detecção de *malware* Android.

Em conclusão, os resultados desta dissertação confirmam a importância da seleção de características para melhorar o desempenho dos classificadores em datasets de detecção de *Malware* Android. Os métodos analisados, em sua maioria, alcançaram uma redução significativa no número de características, com destaque para o LASSO e o RFE, que obtiveram as reduções mais altas no número de características.

Os resultados mostram que os métodos Cor-ABC, Jaccard-ABC, ABC, GA e qui-quadrado obtiveram os melhores resultados nas métricas em todos os cenários com diferença pouco significativas entre si, mostrando que esses métodos se destacam na conservação das métricas com uma redução consideravel (acima de 30%) na maioria dos casos.

Os métodos propostos, Jaccard-ABC e Cor-ABC, se mostraram eficazes, superando o método ABC em diversos cenários, principalmente em termos de redução de características. No entanto, a eficácia do Cor-ABC foi limitada em datasets com menor correlação entre características, sugerindo a necessidade de ajustes nos parâmetros para alcançar melhores resultados.

No quesito estabilidade, o Cor-ABC se destacou, exibindo menor aleatoriedade

interna, especialmente em comparação com o ABC, devido ao uso do coeficiente de correlação de Pearson, uma métrica mais estável. Cumprindo o que foi proposto no inicio deste trabalho.

Por fim, a escolha do classificador mostrou-se um fator determinante nos resultados de redução de características, com o Random Forest alcançando melhores resultados que o XGBoost na maioria dos casos. Isso reforça a importância de ajustar tanto o método de seleção quanto o classificador para maximizar a eficiência do processo de seleção de características em métodos baseados em modelo. Assim, os métodos propostos oferecem contribuições valiosas para o aprimoramento da seleção de características.

Referências Bibliográficas

Abdel-Basset, M., Mohamed, M., Elhoseny, M., Chiclana, F., Zaied, A. E.-N. H., et al. (2019). Cosine similarity measures of bipolar neutrosophic set for diagnosis of bipolar disorder diseases. *Artificial Intelligence in Medicine*, 101:101735.

Aberni, Y., Boubchir, L., e Daachi, B. (2020). Palm vein recognition based on competitive coding scheme using multi-scale local binary pattern with ant colony optimization. *Pattern Recognition Letters*, 136:101–110.

Abhishek, K. e Hamarneh, G. (2021). Matthews correlation coefficient loss for deep convolutional networks: Application to skin lesion segmentation. Em *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pgs. 225–229. IEEE.

Agrawal, P., Abutarboush, H. F., Ganesh, T., e Mohamed, A. W. (2021). Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *Ieee Access*, 9:26766–26791.

Alazab, M. (2020). Automated malware detection in mobile app stores based on robust feature generation. *Electronics*, 9(3):435.

Alhussan, A. A., Abdelhamid, A. A., El-Kenawy, E.-S. M., Ibrahim, A., Eid, M. M., Khafaga, D. S., e Ahmed, A. E. (2023). A binary waterwheel plant optimization algorithm for feature selection. *IEEE Access*.

Arora, S. e Anand, P. (2019). Binary butterfly optimization approaches for feature selection. *Expert Systems with Applications*, 116:147–160.

Bag, S., Kumar, S. K., e Tiwari, M. K. (2019). An efficient recommendation generation using relevant jaccard similarity. *Information Sciences*, 483:53–64.

Beheshti, Z. (2021). Utf: Upgrade transfer function for binary meta-heuristic algorithms. *Applied Soft Computing*, 106:107346.

Bhattacharya, A., Goswami, R. T., e Mukherjee, K. (2019). A feature selection technique based on rough set and improvised pso algorithm (psors-fs) for permission based detection of android malwares. *International journal of machine learning and cybernetics*, 10:1893–1907.

Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., e Lang, M. (2020). Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis*, 143:106839.

Bommert, A., Welchowski, T., Schmid, M., e Rahnenführer, J. (2022). Benchmark of filter methods for feature selection in high-dimensional gene expression survival data. *Briefings in Bioinformatics*, 23(1):bbab354.

Cai, L., Li, Y., e Xiong, Z. (2021). Jowmdroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. *Computers & Security*, 100:102086.

Chandrashekar, G. e Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Chaudhuri, A. e Sahu, T. P. (2021a). Binary jaya algorithm based on binary similarity measure for feature selection. *Journal of Ambient Intelligence and Humanized Computing*, pgs. 1–18.

Chaudhuri, A. e Sahu, T. P. (2021b). A hybrid feature selection method based on binary jaya algorithm for micro-array data classification. *Computers & Electrical Engineering*, 90:106963.

Chen, C.-W., Tsai, Y.-H., Chang, F.-R., e Lin, W.-C. (2020). Ensemble feature selection in medical datasets: Combining filter, wrapper, and embedded feature selection results. *Expert Systems*, 37(5):e12553.

Chicco, D. e Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

Damodaran, A., Troia, F. D., Visaggio, C. A., Austin, T. H., e Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13:1–12.

Dhal, P. e Azad, C. (2022). A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence*, pgs. 1–39.

Dorigo, M., Birattari, M., e Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

Emary, E., Zawbaa, H. M., e Hassanien, A. E. (2016). Binary ant lion approaches for feature selection. *Neurocomputing*, 213:54–65.

Ghosh, A. e Barman, S. (2016). Application of euclidean distance measurement and principal component analysis for gene identification. *Gene*, 583(2):112–120.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

Guyon, I. e Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Hancer, E., Xue, B., Karaboga, D., e Zhang, M. (2015). A binary abc algorithm based on advanced similarity scheme for feature selection. *Applied Soft Computing*, 36:334–348.

He, H. e Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

Hillier, C., Balyan, V., et al. (2019). Error detection and correction on-board nanosatellites using hamming codes. *Journal of Electrical and Computer Engineering*, 2019.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

Hu, D., Chen, L., Fang, H., Fang, Z., Li, T., e Gao, Y. (2023). Spatio-temporal trajectory similarity measures: A comprehensive survey and quantitative study. *arXiv preprint arXiv:2303.05012*.

Hussien, A. G., Hassanien, A. E., Houssein, E. H., Bhattacharyya, S., e Amin, M. (2019). S-shaped binary whale optimization algorithm for feature selection. Em *Recent Trends in Signal and Image Processing: ISSIP 2017*, pgs. 79–87. Springer.

James, G., Witten, D., Hastie, T., e Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

Jia, W., Sun, M., Lian, J., e Hou, S. (2022). Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693.

John, G. H., Kohavi, R., e Pfleger, K. (1994). Irrelevant features and the subset selection problem. Em *Machine learning proceedings 1994*, pgs. 121–129. Elsevier.

Jović, A., Brkić, K., e Bogunović, N. (2015). A review of feature selection methods with applications. Em *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pgs. 1200–1205. Ieee.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes university, engineering faculty.

Karimi, D. e Salcudean, S. E. (2019). Reducing the hausdorff distance in medical image segmentation with convolutional neural networks. *IEEE Transactions on medical imaging*, 39(2):499–513.

Kashan, A. H. (2009). League championship algorithm: a new algorithm for numerical function optimization. Em *2009 international conference of soft computing and pattern recognition*, pgs. 43–48. IEEE.

Kaveh, A. e Talatahari, S. (2010). A novel heuristic optimization method: charged system search. *Acta mechanica*, 213(3-4):267–289.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. Em *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pgs. 1942–1948. IEEE.

Keogh, E. J. e Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. Em *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pgs. 285–289.

Kumar, K. S., Manigandan, T., Chitra, D., e Murali, L. (2020). Object recognition using hausdorff distance for multimedia applications. *Multimedia Tools and Applications*, 79:4099–4114.

Kuncheva, L. I. (2007). A stability index for feature selection. Em *Artificial intelligence and applications*, pgs. 421–427. Citeseer.

Lazar, C., Taminau, J., Meganck, S., Steenhoff, D., Coletta, A., Molter, C., de Schaetzen, V., Duque, R., Bersini, H., e Nowe, A. (2012). A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(4):1106–1119.

Liu, Y., Cheng, J., Yan, C., Wu, X., e Chen, F. (2015). Research on the matthews correlation coefficients metrics of personalized recommendation algorithm evaluation. *International Journal of Hybrid Information Technology*, 8(1):163–172.

Mafarja, M., Aljarah, I., Faris, H., Hammouri, A. I., Ala'M, A.-Z., e Mirjalili, S. (2019). Binary grasshopper optimisation algorithm approaches for feature selection problems. *Expert Systems with Applications*, 117:267–286.

Mahindru, A. e Sangal, A. (2021a). Fsdroid:-a feature selection technique to detect malware from android using machine learning techniques: Fsdroid. *Multimedia Tools and Applications*, 80:13271–13323.

Mahindru, A. e Sangal, A. (2021b). Semidroid: a behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches. *International Journal of Machine Learning and Cybernetics*, 12:1369–1411.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Nguyen, B. H., Xue, B., e Zhang, M. (2020). A survey on swarm intelligence approaches to feature selection in data mining. *Swarm and Evolutionary Computation*, 54:100663.

Niwattanakul, S., Singthongchai, J., Naenudorn, E., e Wanapu, S. (2013). Using of jaccard coefficient for keywords similarity. Em *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pgs. 380–384.

Nogueira, S. (2018). *Quantifying the stability of feature selection*. The University of Manchester (United Kingdom).

Nogueira, S. e Brown, G. (2016). Measuring the stability of feature selection. Em *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*, pgs. 442–457. Springer.

Nssibi, M., Manita, G., e Korbaa, O. (2023). Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey. *Computer Science Review*, 49:100559.

Pampará, G. e Engelbrecht, A. P. (2011). Binary artificial bee colony optimization. Em *2011 IEEE Symposium on Swarm Intelligence*, pgs. 1–8. IEEE.

Patel, S. P. e Upadhyay, S. H. (2020). Euclidean distance based feature ranking and subset selection for bearing fault diagnosis. *Expert Systems with Applications*, 154:113400.

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., e Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., e Xiang, Y. (2020). A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, 53(6):1–36.

Rashedi, E., Nezamabadi-Pour, H., e Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248.

Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., Ranganath, V. P., Li, H., e Guevara, N. (2015). Experimental study with real-world data for Android app security analysis using machine learning. Em *31st ACSAC*, pg. 81–90. ACM.

Şahin, D. Ö., Kural, O. E., Akleylek, S., e Kılıç, E. (2021). A novel permission-based android malware detection system using feature selection based on linear regression. *Neural Computing and Applications*, pgs. 1–16.

Salah, A., Shalabi, E., e Khedr, W. (2020). A lightweight android malware classifier using novel feature selection methods. *Symmetry*, 12(5):858.

Schiezaro, M. e Pedrini, H. (2013a). Data feature selection based on artificial bee colony algorithm. *EURASIP Journal on Image and Video processing*, 2013:1–8.

Schiezaro, M. e Pedrini, H. (2013b). SData feature selection based on Artificial Bee Colony algorithm. *EURASIP Journal on Image and Video Processing*, 47.

Seeley, T. D. (2014). *Honeybee ecology: a study of adaptation in social life*, volume 36. Princeton University Press.

Shah-Hosseini, H. (2011). Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. *International journal of computational science and engineering*, 6(1-2):132–140.

Sharma, M. e Kaur, P. (2021). A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. *Archives of Computational Methods in Engineering*, 28:1103–1127.

Sousa, R. S. D., Boukerche, A., e Loureiro, A. A. F. (2020). Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.*, 53(5).

Storn, R. e Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359.

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z., e Conti, M. (2020). Similarity-based android malware detection using hamming distance of static binary features. *Future Generation Computer Systems*, 105:230–247.

Tang, J., Alelyani, S., e Liu, H. (2014). Feature selection for classification: A review. *Data classification: Algorithms and applications*, pg. 37.

Toloşi, L. e Lengauer, T. (2011). Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994.

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

Willett, P. (2003). Similarity-based approaches to virtual screening.

Xu, Y., Cui, Z., e Zeng, J. (2010). Social emotional optimization algorithm for nonlinear constrained optimization problems. Em *Swarm, Evolutionary, and Memetic Computing: First International Conference on Swarm, Evolutionary, and Memetic Computing, SEMCCO 2010, Chennai, India, December 16-18, 2010. Proceedings 1*, pgs. 583–590. Springer.

Yan, K. e Zhang, D. (2015). Feature selection and analysis on correlated gas sensor data with recursive feature elimination. *Sensors and Actuators B: Chemical*, 212:353–363.

Yavuz, G. e Aydin, D. (2016). Angle modulated artificial bee colony algorithms for feature selection. *Applied Computational Intelligence and Soft Computing*, 2016:7–7.

Zhang, L. M., Dahlmann, C., e Zhang, Y. (2009). Human-inspired algorithms for continuous function optimization. Em *2009 IEEE international conference on intelligent computing and intelligent systems*, volume 1, pgs. 318–321. IEEE.