Bioj48: Adaptando o Método J48 para Classificação de Dados Biológicos Desbalanceados

Thacyo Euqueres De Villa

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2025

Thacyo Euqueres De Villa

Bioj48: Adaptando o Método J48 para Classificação de Dados Biológicos Desbalanceados

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Laurence Rodrigues do Amaral

Uberlândia
2025

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

V712t
2025

Villa, Thacyo Euqueres De, 1994-
Bioj48 [recurso eletrônico]: adaptando o método J48 para classificação de dados biológicos desbalanceados / Thacyo Euqueres De Villa. - 2025.

Orientador: Laurence Rodrigues do Amaral.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Programa de Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2025.5552
Inclui bibliografia.
Inclui ilustrações.

1. Computação. 2. Inteligência artificial. 3. Aprendizado do computador. 4. Algoritmos computacionais. I. Amaral, Laurence Rodrigues do, 1978-, (Orient.). II. Universidade Federal de Uberlândia. Programa de Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

Rejâne Maria da Silva
Bibliotecária-Documentalista – CRB6/1925

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação, 20/2025, PPGCO				
Data:	10 de Julho de 2025	Hora de início:	14:00	Hora de encerramento:	16:53
Matrícula do Discente:	12312CCP032				
Nome do Discente:	Thacyo Euqueres De Villa				
Título do Trabalho:	bioJ48: Adaptando o Método J48 para Classificação de Dados Biológicos Desbalanceados				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	----------				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Marcelo Tavares - IME/UFU, Leticia da Conceição Braga - NEP/Instituto Mário Penna e Laurence Rodrigues do Amaral - FACOM/UFU, orientador do candidato.

Os examinadores participaram das seguintes localidades: Laurence Rodrigues do Amaral - Patos de Minas/MG, Leticia da Conceição Braga - Belo Horizonte/MG e Marcelo Tavares de Uberlândia/MG. O aluno Thacyo Euqueres De Villa participou de Tupaciguara/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Laurence Rodrigues do Amaral , apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação da Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir ao candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Marcelo Tavares**, **Professor(a) do Magistério Superior**, em 11/07/2025, às 18:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Letícia da Conceição Braga**, **Usuário Externo**, em 11/07/2025, às 21:00, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Laurence Rodrigues do Amaral**, **Professor(a) do Magistério Superior**, em 15/07/2025, às 11:22, conforme horário oficial de Brasília, 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://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6488738** e o código CRC **7CC81086**.

Referência: Processo nº 23117.046338/2025-62 SEI nº 6488738

Dedico este trabalho aos meus pais, Jerson e Elaine, por terem sido minha base e meu exemplo de esforço e integridade.

Ao meu irmão Thyago, pelo companheirismo em todas as fases da vida.

À minha esposa Maria Tereza, pela parceria, apoio e amor incondicional.

E, principalmente, ao meu filho Heitor, razão maior da minha dedicação e inspiração constante para construir um futuro melhor.

Agradecimentos

A realização desta pesquisa não seria possível sem o apoio e incentivo de pessoas especiais que caminharam ao meu lado ao longo desta jornada.

Agradeço primeiramente aos meus pais, Jerson e Elaine, pelo amor incondicional, pela educação e pelos valores que me foram ensinados desde sempre. Ao meu irmão Thyago, pelo companheirismo e apoio constante. À minha esposa Maria Tereza, pela paciência, compreensão e incentivo nos momentos de cansaço e incerteza.

Um agradecimento mais do que especial ao meu filho Heitor, que com seu sorriso e presença trouxe ainda mais sentido a essa conquista. Ele é, sem dúvida, a maior motivação para que eu busque sempre ser alguém melhor.

Agradeço também ao meu orientador Laurence Rodrigues do Amaral, por toda a orientação, disponibilidade e confiança no desenvolvimento deste trabalho. Sua contribuição foi fundamental para a evolução desta pesquisa.

A todos, meu sincero muito obrigado.

Epígrafe

"Life is very short, and there's no time
For fussing and fighting, my friend"
– Lennon/McCartney

Resumo

Modelos de classificação baseados em árvores de decisão são amplamente utilizados em Machine Learning (ML) devido à sua alta interpretabilidade e precisão. No entanto, algoritmos tradicionais de árvores de decisão, como o J48 (uma implementação do C4.5), enfrentam desafios quando aplicados a conjuntos de dados biológicos desbalanceados, nos quais a quantidade de registros por classe varia significativamente. Esse desbalanceamento pode levar a métricas de desempenho enganosas, pois os modelos tendem a favorecer a classe majoritária, negligenciando a classe minoritária, que muitas vezes é crucial em aplicações médicas e biológicas. Este estudo propõe modificações no algoritmo J48 para melhorar sua sensibilidade e especificidade na classificação de dados biológicos desbalanceados. A pesquisa explora ajustes no cálculo de ganho de informação, visando aprimorar o desempenho do algoritmo sem comprometer sua interpretabilidade. Diversas métricas de avaliação são analisadas para garantir uma abordagem de classificação mais equilibrada. A metodologia envolve a implementação e teste do J48 modificado, chamado bioJ48, em conjuntos de dados biológicos reais com diferentes graus de desbalanceamento. Experimentos comparativos entre o J48 tradicional e o bioJ48 são conduzidos, avaliando a eficácia preditiva com base em métricas como acurácia, precisão, recall e F1-score. Os resultados esperados incluem um modelo de classificação aprimorado, capaz de identificar melhor as classes minoritárias em dados biológicos, proporcionando uma avaliação de desempenho mais confiável por meio de métricas diversificadas. As modificações propostas visam contribuir para o campo de ML, oferecendo uma solução prática para o problema do desbalanceamento de classes na análise de dados biológicos.

Palavras-chave: BioJ48, J48, C4.5, Classificação, Desbalanceamento de Classes, Aprendizado de Máquina, Dados Biológicos, Inteligência Artificial.

Abstract

Decision tree-based classification models are widely used in Machine Learning (ML) due to their high interpretability and accuracy. However, traditional decision tree algorithms, such as J48 (an implementation of C4.5), face challenges when applied to imbalanced biological datasets, where the number of records per class varies significantly. This imbalance can lead to misleading performance metrics, as models tend to favor the majority class while neglecting the minority class, which is often crucial in medical and biological applications. This study proposes modifications to the J48 algorithm to improve its sensitivity and specificity when classifying imbalanced biological data. The research explores adjustments in information gain calculation, aiming to enhance the algorithm's performance without compromising interpretability. Various evaluation metrics are analyzed to ensure a more balanced classification approach. The research methodology involves the implementation and testing of the modified J48, named bioJ48, on real biological datasets with different degrees of class imbalance. Comparative experiments between traditional J48 and bioJ48 are conducted, evaluating their predictive effectiveness based on key performance metrics such as accuracy, precision, recall, and F1-score. The expected results include an improved classification model that better identifies minority classes in biological datasets, providing a more reliable evaluation of performance through diversified metrics. The proposed modifications aim to contribute to the field of ML by offering a practical solution to class imbalance issues in biological data analysis.

Palavras-chave: BioJ48, J48, C4.5, Classification, Class Imbalance, Machine Learning, Biological Data, Artificial Intelligence.

Lista de ilustrações

Figura 1 – Distribuição de Classes 25
Figura 2 – Distribuição das Variáveis por Classe e Geral (Boxplots) 26
Figura 3 – Matriz de Correlação das Variáveis Independentes 27
Figura 4 – Curva ROC 30

Lista de tabelas

Tabela 1 – Comparação entre métodos de alteração do J48 no contexto de dados desbalanceados. 21
Tabela 2 – Comparação entre os algoritmos J48 e BioJ48 quanto às métricas de desempenho em cenário de treinamento completo (full training). 35
Tabela 3 – Comparação entre os algoritmos J48 e BioJ48 quanto às métricas de desempenho com seus respectivos intervalos de confiança de 95% e a quantidade de métricas com resultado zerado (Cenário de validação cruzada com todas as 66 sementes possíveis). 36

Sumário

Agradecimentos **6**

Epígrafe **7**

Lista de ilustrações **10**

Lista de tabelas **11**

1 INTRODUÇÃO **14**
1.1 Motivação **15**
1.2 Objetivos **15**
1.3 Contribuições **15**
1.4 Desafios da Pesquisa e Hipóteses **16**
1.5 Organização da Dissertação **16**

2 TRABALHOS RELACIONADOS: ESTADO DA ARTE **18**
2.1 Uso do J48 em tarefas de classificação. **18**
2.2 Uso do J48 em conjuntos de dados biológicos **19**
2.3 Alterações no J48 para melhoria de desempenho **20**
2.4 Considerações Finais **21**

3 FUNDAMENTAÇÃO TEÓRICA **22**
3.1 Hanseníase **22**
3.2 Aprendizado de Máquina **23**
3.2.1 Aprendizado Supervisionado 23
3.2.2 Aprendizado Não-Supervisionado 23
3.2.3 Aprendizado Semi-supervisionado 23
3.2.4 Modelo de Classificação 23

4 METODOLOGIA **25**
4.1 Conjunto de Dados **25**

4.2 Métricas de avaliação 27
4.3 Alterações Propostas no J48: O Algoritmo BioJ48 31

5 EXPERIMENTOS E ANÁLISE DOS RESULTADOS 34
5.1 Contextualização do Experimento 34
5.1.1 Procedimentos de Avaliação 35
5.2 Resultados Obtidos 35

6 CONCLUSÃO 37
6.1 Trabalhos Futuros 38
6.2 Contribuições em Produção Bibliográfica 38

REFERÊNCIAS 40

APÊNDICE 43

Capítulo 1

Introdução

O Aprendizado de Máquina (*Machine Learning*) é uma subárea da Inteligência Artificial focada no desenvolvimento de algoritmos que permitem aos computadores aprender a partir de dados e tomar decisões baseadas nesses dados. Uma das principais tarefas do Aprendizado de Máquina é a classificação, onde o objetivo é prever a classe de novos exemplos com base em um conjunto de dados rotulados. Entre os modelos de classificação, as árvores de decisão são amplamente estudadas e utilizadas devido à sua alta acurácia e estrutura de fácil entendimento. Esses modelos são especialmente populares em domínios onde a interpretabilidade é crucial, como na área da Biologia e Medicina.

Pesquisadores e profissionais em modelagem de dados utilizam modelos de *Machine Learning* (ML) a fim de criar soluções para determinados problemas de diversas áreas. Tanto no processo de modelagem quanto no monitoramento ou uso do modelo otimizado, é necessário a utilização de métricas de desempenho e avaliação de forma a garantir a eficácia da solução proposta. Nenhuma métrica única é capaz de retornar todas as propriedades e informações necessárias para avaliação completa de um modelo de ML, o que justifica a utilização de diversas métricas distintas para prever o desempenho de um único modelo otimizado. A procura do melhor modelo de ML é um grande desafio e normalmente consiste na comparação entre outros modelos ajustados por meio de métricas de desempenho (HICKS et al., 2022).

O J48 é uma implementação do algoritmo C4.5 no software Weka (WITTEN; FRANK; HALL, 2011), é um algoritmo de aprendizado de máquina amplamente utilizado. O C4.5, desenvolvido por Ross Quinlan (QUINLAN, 1993), é um algoritmo de indução de árvores de decisão que gera classificadores expressos como árvores de decisão. Embora eficaz, o J48 possui limitações, especialmente ao lidar com dados biológicos, que frequentemente apresentam desbalanceamento severo entre as classes (RODRIGUES; AMARAL, 2012).

A classificação de dados desbalanceados é um desafio recorrente em ML. A eficácia dos algoritmos de classificação pode ser comprometida quando os dados estão desbalanceados, resultando em métricas de desempenho enganosas. Devido a isso, esse projeto de pesquisa visa explorar a aplicação do algoritmo J48 em conjuntos de dados desbalanceados, propondo modificações no algoritmo J48, avaliando diversas métricas de desempenho para identificar e solucionar problemas de sensibilidade e especificidade encontrados.

1.1 Motivação

Muitos profissionais tem dificuldade em compreender medidas de avaliação em ML tornando desafiadora a tarefa de avaliação e monitoramento dos modelos. Além disso, os artigos científicos que abordam tais conceitos de avaliação ainda são escassos na comunidade acadêmica. O interesse em melhorar o desempenho de modelos de classificação em conjuntos de dados desbalanceados surge da necessidade de obter previsões mais precisas e confiáveis em áreas críticas como em diagnósticos médicos. Em muitos casos, a alta acurácia de um modelo pode mascarar sua incapacidade de identificar corretamente a classe minoritária, o que é especialmente problemático quando a detecção da classe minoritária é crucial.

Dados biológicos possuem características únicas que os diferenciam de outros tipos de dados. Com frequência, esses dados possuem um baixo número de registros e poucos atributos, além de um desbalanceamento severo entre as classes. Esse cenário é ainda mais desafiador em casos de doenças raras, como o câncer de ovário, onde os dados são escassos e altamente desbalanceados. A maioria dos métodos de classificação, incluindo o J48, não foram originalmente desenvolvidos para lidar com essas peculiaridades dos dados biológicos (CRUZ; WISHART, 2006).

Esse projeto de pesquisa é motivado pela necessidade de abordar essa limitação, propondo e avaliando modificações e estratégias no algoritmo J48 para melhorar a sensibilidade e a especificidade em cenários de desbalanceamento entre as classes.

1.2 Objetivos

O objetivo é melhorar a classificação de *datasets* desbalanceados, focando na identificação precisa de classes com baixo número de registros, sem priorizar unicamente a maior acurácia.

Objetivos específicos:

- Avaliar o desempenho e identificar as limitações do algoritmo J48 em conjunto de dados balanceado e desbalanceado;
- Identificar métricas de avaliação que sejam apropriadas para medir o desempenho de modelos de classificação em diferentes contextos e avaliar sua relevância;
- Propor e testar uma solução para melhorar a sensibilidade e a especificidade do J48, sem comprometer significativamente outras métricas de desempenho.

1.3 Contribuições

Com o desenvolvimento dessa pesquisa realizou-se as seguintes contribuições:

- Uma análise detalhada do desempenho do algoritmo J48 em cenários de dados desbalanceados, destacando as métricas mais afetadas;
- Redução de viés em avaliações de modelos de ML;

- Uma solução prática para aumentar a sensibilidade do modelo em dados desbalanceados, validada por experimentos empíricos;
- Proposto um novo modelo chamado bioJ48 para classificação de dados biológicos desbalanceados.

1.4 Desafios da Pesquisa e Hipóteses

A hipótese central desse trabalho é a seguinte: "as modificações propostas no algotimo J48 resultarão em um modelo mais adequado para conjuntos de dados biológicos desbalanceados", "a alteração poderá ser feita na fórmula de cálculo da entropia ou do índice Gini, com o objetivo de melhorar a sensibilidade sem comprometer a acurácia global do modelo".

Desafios esperados:

- Balanceamento entre Sensibilidade e Especificidade: Modificar o J48 para dar mais peso às classes minoritárias pode diminuir a especificidade para as classes majoritárias, criando um novo tipo de desbalanceamento;
- Generalização do Modelo: Garantir que o modelo modificado generalize bem para diferentes tipos de conjuntos de dados biológicos, sem ser excessivamente ajustado para um *dataset* específico;
- Diversidade de métricas: A grande variedade de métricas de avaliação disponíveis torna desafiador identificar as mais apropriadas para cada cenário;
- Interpretação dos Resultados: As mudanças na fórmula de entropia ou índice Gini devem ser transparentes e facilmente compreendidas por especialistas da área biológica.

1.5 Organização da Dissertação

Este texto foi organizado em 6 capítulos para apresentar o estudo de maneira clara e compreensível. Cada capítulo aborda os conceitos e métodos empregados, os experimentos realizados e os resultados obtidos.

- ❑ **Capítulo 1:** inclui as considerações iniciais e o contexto em que está inserido pesquisa, motivação para seu desenvolvimento e objetivos a atingir.
- ❑ **Capítulo 2:** revisa trabalhos correlatos que utilizam o algoritmo J48 em dados biologicos e que exploram modificações no algoritmo. A análise destaca as contribuições e limitações desses estudos, proporcionando uma base teórica para o desenvolvimento deste trabalho.
- ❑ **Capítulo 3:** aborda os conceitos fundamentais necessários para o entendimento do estudo. São discutidos tópicos sobre hanseníase e sua relevância como problema de saúde pública. Também são introduzidos métodos de machine learning com foco em algoritmos de classificação.

- **Capítulo 4:** detalha os procedimentos metodológicos empregados no estudo. Abrange a preparação do dataset, implementação do BioJ48, simulações, calculo de métricas de desempenho, e avaliação do BioJ48. Este capítulo fornece uma descrição completa das etapas experimentais e das técnicas aplicadas para alcançar os resultados propostos.

- **Capítulo 5:** apresenta os experimentos realizados com diferentes datasets sinteticos usando J48 e BioJ48. O capítulo avalia o desempenho desses algoritmos utilizando métricas de avaliação.

- **Capítulo 6:** este capítulo oferece uma visão geral dos principais resultados obtidos e explora a aplicação dos métodos propostos, com ênfase em suas contribuições e limitações dentro do dataset biológicos com desbalanceamento de classes.

Capítulo 2

Trabalhos Relacionados: Estado da Arte

2.1 Uso do J48 em tarefas de classificação.

O J48 é amplamente utilizado em diversas tarefas de classificação devido à sua capacidade de gerar árvores de decisão de forma eficiente e com alta acurácia. Esse algoritmo funciona dividindo o conjunto de dados com base em atributos que proporcionam a maior informação ganha, formando uma árvore de decisão que pode ser usada para classificar novos dados. Estudos demonstram que o J48 é eficaz em várias áreas, incluindo a análise de dados financeiros, detecção de fraudes e diagnóstico médico, onde a acurácia e a interpretabilidade são cruciais (QUINLAN, 1993).

Apesar da sua popularidade, o J48 enfrenta desafios quando aplicado a dados desbalanceados. Em muitos casos, o algoritmo tende a favorecer as classes majoritárias, o que pode resultar em uma classificação injusta para as classes minoritárias. Diversas abordagens tem sido propostas para mitigar esse problema, como o uso de técnicas de reamostragem e a introdução de novos critérios de divisão que levem em consideração o balanceamento das classes (FERNÁNDEZ et al., 2018). Essas modificações tem mostrado melhorias na performance do J48 em cenários de desbalanceamento, mas ainda há espaço para aprimoramentos, especialmente em contextos específicos como os dados biológicos.

Publicado por (SAHU; MEHTRE, 2015), esse trabalho desenvolve uma versão aprimorada do algoritmo J48 para melhorar a precisão na detecção de intrusões em redes. O estudo utiliza o J48 para detectar possíveis ataques que poderiam comprometer a confidencialidade da rede, demonstrando uma melhoria significativa na precisão da detecção em comparação com métodos tradicionais.

O estudo conduzido por (SCHAUERHUBER et al., 2021), investiga a otimização de hiperparâmetros em algoritmos de indução de árvores de decisão, incluindo o J48. O estudo avaliou o desempenho do J48 em 18 conjuntos de dados de classificação do repositório UCI *Machine Learning Repository*, ajustando parâmetros como a confiança de poda e o número mínimo de instâncias por folha. A pesquisa destacou a importância da otimização adequada de hiperparâmetros para melhorar a precisão dos modelos de classificação baseados em J48.

No estudo (MENDES et al., 2021) a aplicação do J48, combinada com Algoritmos Genéticos (AGs), resultou em uma arquitetura híbrida que demonstrou alta precisão em tarefas de classificação de imagens, superando outros classificadores em benchmarks como MNIST e Fashion-MNIST. A abordagem híbrida otimizou não apenas a estrutura da rede, mas também os parâmetros de treinamento, resultando em um modelo altamente eficiente e preciso.

2.2 Uso do J48 em conjuntos de dados biológicos

Um caso de utilização do J48 em que apresentou grande importância foi no estudo de (AIRES et al., 2022) para diferenciar pacientes com COVID-19 graves e não graves com base em parâmetros de coagulação. O algoritmo mostrou alta precisão na classificação, destacando seu potencial como ferramenta de suporte para intervenções terapêuticas em pacientes com COVID-19.

Outro exemplo de utilização é o estudo de (MARTINS et al., 2023), que investigou a expressão do papilomavírus humano (HPV), p16, p53 e p63 em carcinomas de células escamosas da bexiga não associados à esquistossomose. Nesse estudo, o algoritmo J48 foi utilizado para construir árvores de decisão a fim de prever a classificação histopatológica com base em características clinicopatológicas.

Mais recentemente, o estudo de (FRADICO et al., 2023) explorou o uso de redes de mediadores solúveis no soro para identificar biomarcadores prognósticos em pacientes com febre amarela. Nesse estudo, o algoritmo J48 foi empregado para construir árvores de decisão, identificando mediadores eficazes na classificação de desfechos da doença com alta precisão.

No estudo (ALVES et al., 2021) os experimentos conduzidos mostraram que a metodologia IG-CEE com a utilização do J48 apresentou uma melhora significativa no desempenho dos Algoritmos Genéticos, em comparação com outras abordagens tradicionais. A combinação de seleção de atributos com ganho de informação e evolução de regras resultou em classificações mais precisas e eficientes.

No estudo (AMARAL et al., 2021) os resultados indicaram que a integração do J48 no *framework* NEL aprimorou significativamente a eficiência e a eficácia do biocurador de Ontologia de Genes, proporcionando uma ferramenta robusta para a análise e a classificação de grandes volumes de dados biológicos.

No estudo (AMARAL, 2012) a integração do J48 no *framework* do GANEL resultou em uma maior precisão e interpretabilidade das regras de classificação. O sistema demonstrou ser capaz de extrair conhecimento de alto nível e útil dos dados biológicos, superando métodos tradicionais como árvores de decisão simples, One R e Single Conjunctive Rule Learner.

No estudo (ALIE; NEGESSE, 2023) utilizou-se dados demográficos e de serviços de saúde de nível nacional para desenvolver um modelo de previsão para os serviços de teste de HIV em adolescentes na Etiópia. Entre os oito algoritmos de ML testados, o modelo J48 de árvore de decisão demonstrou a maior precisão e exatidão na classificação. O estudo destaca a eficácia do J48 na otimização da alocação de recursos limitados e na identificação precoce de barreiras para

o teste de HIV, especialmente em ambientes de recursos limitados. Este trabalho mostra como o J48 pode ser utilizado para melhorar os resultados de saúde e a gestão de recursos em países em desenvolvimento.

2.3 Alterações no J48 para melhoria de desempenho

No artigo de (KATZ ASAF SHABTAI; OFEK, 2014), as árvores de decisão apresentam três desvantagens: desempenho reduzido quando o conjunto de treinamento é pequeno; critérios de decisão rígidos; e a possibilidade de que um único atributo "não característico"possa "descarrilar"o processo de classificação. Para mitigar esses problemas, eles apresentam o ConfDTree (Árvore de Decisão Baseada em Confiança), um método de pós-processamento que permite que as árvores de decisão classifiquem melhor instâncias atípicas. O estudo experimental demonstra que o método de pós-processamento proposto melhora consistentemente e de forma significativa o desempenho preditivo das árvores de decisão, especialmente em conjuntos de dados pequenos, desbalanceados ou de múltiplas classes, com uma melhoria média de 5% a 9% na performance da curva ROC. No artigo, foram apresentadas e avaliadas duas variações do mesmo método para aprimorar as árvores de decisão tanto para atributos nominais quanto contínuos. O método ConfDTree pode ser utilizado para lidar com três problemas importantes que afetam as árvores de decisão: desempenho reduzido em conjuntos de treinamento pequenos; rigidez do processo de classificação; e valores de atributos atípicos que interferem na correta classificação de uma instância. A capacidade do método de se integrar com qualquer tipo de algoritmo de árvore de decisão é um ponto relevante, permitindo a seleção do algoritmo mais adequado para um conjunto de dados específico, enquanto se mantém os benefícios dos intervalos de confiança. No entanto, o algoritmo proposto possui duas desvantagens: aumenta ligeiramente o custo computacional para classificar uma nova instância e reduz a compreensibilidade do modelo. Em particular, algumas instâncias são afetadas pelo método e eventualmente recebem uma distribuição de classe diferente.

No artigo de (KAPOOR; RANI, 2015) busca-se melhorar o desempenho preditivo por meio de um novo algoritmo de árvore de decisão baseado no J48 e na poda de erro reduzido. A poda reduz a complexidade do classificador final, melhorando a precisão preditiva ao diminuir o sobreajuste resultando em um aprendizado de árvore de decisão rápido, baseado no ganho de informação ou na redução da variância. Eles apresentam o REPTree um método de pós-processamento que reduz o tamanho da árvore, a média da entropia final e a entropia absoluta. O estudo experimental demonstra que o erro absoluto relativo e o erro absoluto relativo da raiz são diminuídos, resultando em uma classificação mais precisa em comparação ao J48. Assim, o REPTree classifica os itens com base em seus atributos de forma mais precisa. Na árvore classificada, a entropia total da árvore de decisão e a aleatoriedade no J48 são muito maiores do que no REPTree. Além disso, o tamanho da árvore para o conjunto de dados utilizado é muito menor, indicando que o REPTree tem um desempenho muito melhor em comparação ao J48.

2.4 Considerações Finais

Tabela 1 – Comparação entre métodos de alteração do J48 no contexto de dados desbalanceados.

Estudo	**Objetivo**	**Métodos Utilizados**	**Resultados Principais**
(KATZ ASAF SHABTAI; OFEK, 2014)	Mitigar problema de desempenho reduzido em árvores de decisão com conjuntos de treinamento pequenos	Calculo de intervalos de confiança e testes de proporção, para identificar instâncias de difícil classificação e sugerir rotas alternativas dentro da árvore de decisão funcionando como uma etapa de pós-processamento	Aumento médio de 5% a 9% na métrica AUC (Área Sob a Curva) de desempenho
(KAPOOR; RANI, 2015)	Mitigar o problema de Sobreajuste (Overfitting) em árvores de decisão	Técnica de poda por redução de erro no algoritmo J48 baseando-se em medidas estatísticas para identificar e eliminar ramos menos confiáveis	Redução da complexidade do modelo contribuindo para a diminuição do sobreajuste além de redução do custo computacional no processo de classificação

Os estudos comparados (Tabela 1) mostram uma variedade de abordagens e técnicas para a mitigar as desvantagens do J48. Cada método oferece vantagens específicas e contribui de maneira significativa para o estado da arte em suas respectivas áreas. As técnicas estatísticas empregadas por (KATZ ASAF SHABTAI; OFEK, 2014) e (KAPOOR; RANI, 2015), demonstram a melhora no desempenho preditivo das árvores de decisão, especialmente em conjuntos de dados pequenos, desequilibrados ou com múltiplas classes, indicando o potencial de aplicação prática. Em conclusão, a alteração de calculo do algoritmo J48 oferece uma oportunidade inexplorada que pode complementar e aprimorar os métodos atuais. Essa técnica tem o potencial de contribuir significativamente para o estado da arte, fornecendo uma ferramenta adicional para pesquisadores na análise e diagnóstico de classificação em dados biológicos.

Capítulo 3

Fundamentação Teórica

3.1 Hanseníase

A hanseníase, também conhecida como lepra, é uma doença infecciosa crônica causada pelo bacilo *Mycobacterium leprae*, que apresenta predileção por células da pele e nervos periféricos. Sua transmissão se dá principalmente por vias aéreas superiores, através do convívio prolongado com uma pessoa infectada e não tratada. Apesar disso, a maioria das pessoas possui resistência natural à infecção, o que dificulta a disseminação em larga escala da doença na população geral (Brasil. Ministério da Saúde, 2022; World Health Organization, 2021).

O Brasil é um dos países com maior número de casos de hanseníase no mundo, ficando atrás apenas da Índia em número absoluto de notificações. Segundo o Ministério da Saúde, a hanseníase ainda representa um sério problema de saúde pública, com forte impacto social, devido ao estigma associado e às incapacidades físicas permanentes que pode provocar quando não tratada adequadamente (Brasil. Ministério da Saúde, 2022).

A doença possui um longo período de incubação — que pode variar de meses a mais de uma década — e manifesta-se com sinais e sintomas como manchas na pele com perda de sensibilidade, formigamento, dor ou espessamento de nervos periféricos e, em casos avançados, atrofia muscular e deformidades (PEREIRA et al., 2020). A hanseníase é classificada clinicamente em formas paucibacilares (PB) e multibacilares (MB), de acordo com o número de lesões e a carga bacilar do paciente.

O diagnóstico ainda é, majoritariamente, clínico-epidemiológico, embora testes laboratoriais imunológicos venham sendo cada vez mais estudados para auxiliar na detecção precoce e no monitoramento de casos. Entre os marcadores imunológicos investigados, destacam-se os níveis de imunoglobulinas, como a IgG1, que podem indicar a presença de resposta imune ativa e servir como variável de interesse para algoritmos de classificação assistida por computador (NERY et al., 2014; MOURA; CALADO; OLIVEIRA, 2023).

3.2 Aprendizado de Máquina

Machine learning (Aprendizado de Máquina) é um subcampo da Inteligência Artificial (IA) focado no desenvolvimento de algoritmos e técnicas que permitem aos computadores aprender e fazer previsões ou decisões baseadas em dados. Ao invés de serem explicitamente programados para realizar uma tarefa específica, os sistemas de aprendizado de máquina usam padrões e inferências estatísticas para melhorar seu desempenho em tarefas específicas à medida que são expostos a mais dados (MITCHELL, 1997). Os tipos de Aprendizado de Máquina são:

3.2.1 Aprendizado Supervisionado

O aprendizado supervisionado é uma abordagem onde os algoritmos são treinados com um conjunto de dados rotulados, isso é, os dados de entrada são associados a suas saídas corretas. O objetivo desses algoritmos é aprender uma função que possa prever a saída correta para novas entradas baseadas nos exemplos de treinamento. Algoritmos supervisionados, como árvores de decisão (ex.: J48), redes neurais artificiais e máquinas de vetor de suporte (SVM), tem sido amplamente aplicados em tarefas de classificação (CHAPELLE; SCHOLKOPF; ZIEN EDS., 2009).

3.2.2 Aprendizado Não-Supervisionado

Diferente do aprendizado supervisionado, o aprendizado não-supervisionado lida com dados que não possuem rótulos. O objetivo é descobrir padrões ou estruturas intrínsecas nos dados. Técnicas como clustering (ex.: k-means) e redução de dimensionalidade (ex.: PCA) são comuns nesse tipo de aprendizado. Essas técnicas são úteis para reduzir a complexidade dos dados de forma a facilitar a visualização e interpretação (CHAPELLE; SCHOLKOPF; ZIEN EDS., 2009).

3.2.3 Aprendizado Semi-supervisionado

O aprendizado semi-supervisionado é uma combinação dos métodos supervisionado e não-supervisionado, utilizando uma pequena quantidade de dados rotulados junto com uma grande quantidade de dados não-rotulados. Essa abordagem é particularmente valiosa quando a rotulação dos dados é cara ou demorada, como frequentemente ocorre em pesquisas biológicas. Modelos semi-supervisionados podem aprender mais eficientemente, aproveitando o grande volume de dados não-rotulados para melhorar a precisão e a robustez das previsões (CHAPELLE; SCHOLKOPF; ZIEN EDS., 2009).

3.2.4 Modelo de Classificação

Os modelos de classificação foram desenvolvidos para uma ampla gama de aplicações em diversas áreas, com o objetivo principal de automatizar a tarefa de atribuir categorias ou rótulos a dados com base em suas características ou atributos. São uma parte fundamental do

Aprendizado de Máquina e desempenham um papel importante em uma variedade de domínios auxiliando na tomada de decisão.

Um modelo de classificação binária é um tipo de modelo projetado para resolver problemas em que existem apenas duas classes, já um modelo de classificação multiclasse é utilizado quando existem três ou mais classes possíveis para prever (BISHOP, 2006).

Capítulo 4

Metodologia

A pesquisa consiste na modificação do algoritmo J48, ajustando a função de seleção de atributos que maximiza a separação entre as classes. Foi explorada a alteração do cálculo do ganho de informação para dar maior peso as classes minoritárias. Realizou-se experimentos comparando o J48 tradicional com a versão modificada, utilizando conjuntos de dados biológicos com diferentes graus de desbalanceamento. Foram considerados dados com classificação binária, em que há apenas duas classes possíveis para predizer.

O trabalho também explora a aplicação de diversos métodos para avaliar o desempenho. As etapas descritas fornecem uma base sólida para compreender as técnicas e procedimentos empregados no desenvolvimento e análise dos modelos propostos.

4.1 Conjunto de Dados

Devido à natureza sensível dos dados e à sua aplicação em contexto médico, todas as informações foram tratadas de forma anonimizada, garantindo a integridade ética da análise.

O conjunto de dados utilizado neste estudo é composto por informações clínicas e laboratoriais de pacientes avaliados para diagnóstico de hanseníase. A base de dados possui um total de 66 amostras, das quais 16 (aproximadamente 24,2%) correspondem a pacientes diagnosticados com hanseníase (classe positiva), e 50 (75,8%) a pacientes não diagnosticados com a doença (classe negativa). Isso caracteriza um cenário de desbalanceamento de classes, bastante comum em contextos biomédicos.

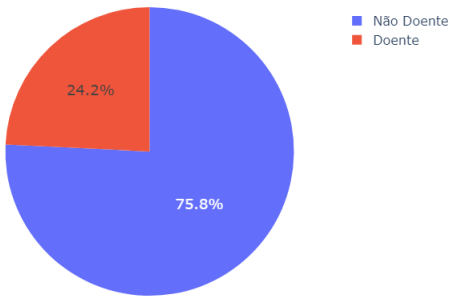

Figura 1 – Distribuição de Classes

As variáveis independentes consistem em marcadores imunológicos de IgG1 específicos, medidos em diferentes antígenos: 2055, Ag85B, 0405, 2331, LID, NDO-LID e NDO-HSA. Todas são variáveis numéricas contínuas e representam concentrações sorológicas relevantes para o diagnóstico da hanseníase. Cada registro corresponde a um indivíduo, e a variável-alvo (classe) é categórica binária, com os rótulos "Doente"e "Não Doente".

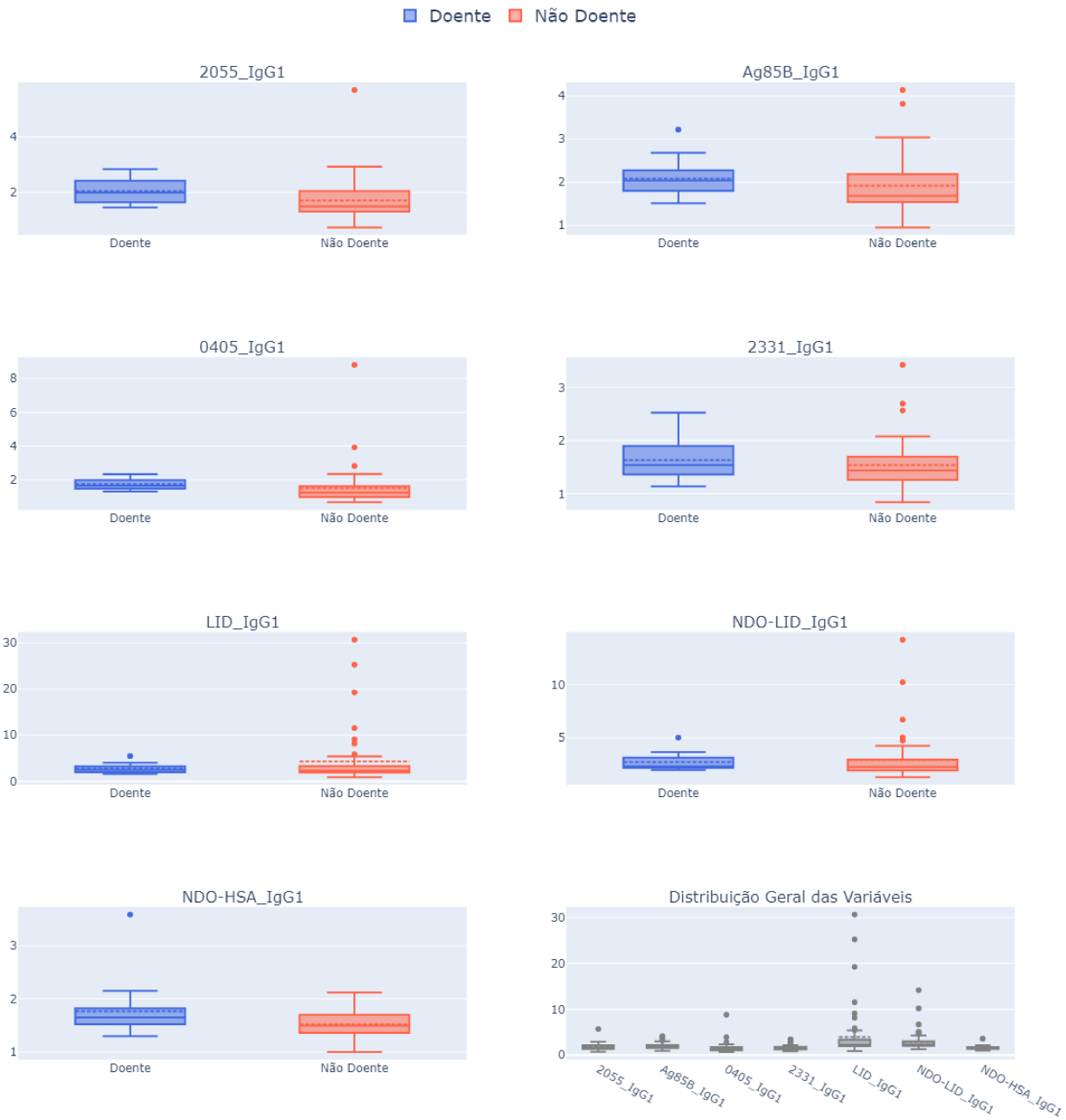

Figura 2 – Distribuição das Variáveis por Classe e Geral (Boxplots)

Para investigar relações entre as variáveis numéricas, foi construída uma matriz de correlação apresentada na Figura 3. A análise revelou correlações moderadas entre alguns marcadores, mas sem sinais de multicolinearidade severa.

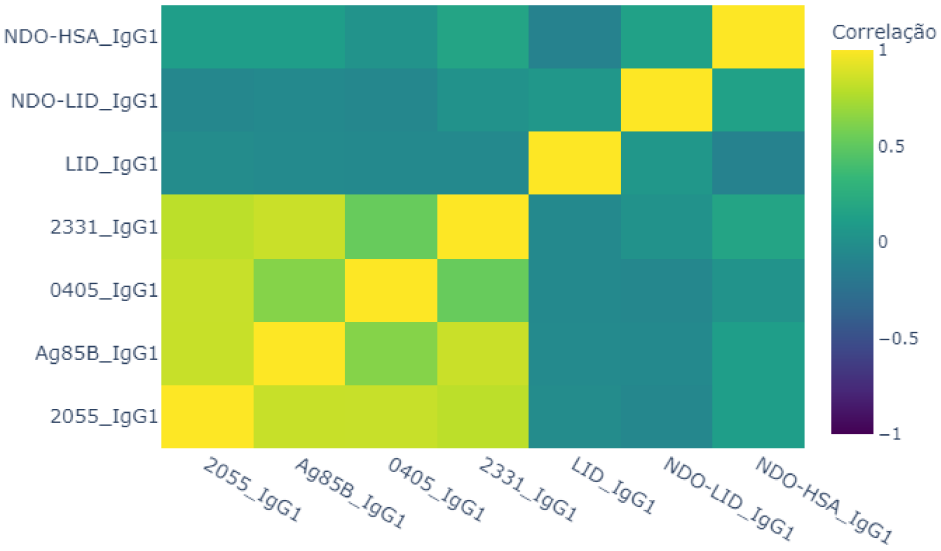

Figura 3 – Matriz de Correlação das Variáveis Independentes

Esse conjunto de dados foi essencial para avaliar a capacidade dos algoritmos J48 e BioJ48 em lidar com desbalanceamento de classes, sendo utilizado em experimentos com validação cruzada estratificada e em execução full training para análise comparativa das métricas de desempenho.

4.2 Métricas de avaliação

As métricas de avaliação de desempenho em Machine Learning são essenciais para medir, comparar e melhorar o desempenho dos modelos, garantindo que eles atendam aos requisitos e expectativas de suas aplicações.

Segundo (HICKS et al., 2022), quantidades relevantes para o cálculo das métricas de um classificador são as quatro entradas na matriz de confusão.

$$M = \begin{pmatrix} VP & FN \\ FP & VN \end{pmatrix}, \tag{1}$$

Aqui são apresentadas algumas métricas tradicionais:

- Verdadeiro positivo (VP): denota o número de amostras positivas corretamente classificadas (HICKS et al., 2022);
- Verdadeiro negativo (VN): denota o número de amostras negativas corretamente classificadas (HICKS et al., 2022);
- Falso positivo (FP): denota o número de amostras incorretamente classificadas como positivas (HICKS et al., 2022);

- Falso negativo (FN): denota o número de amostras incorretamente classificadas como negativas (HICKS et al., 2022).

- Acurácia (ACC): É a proporção entre as amostras corretamente classificadas e o número total de amostras no conjunto de dados de avaliação sendo umas das mais utilizadas na comunidade. Ela é limitada em um intervalo de 0 a 1, sendo 1 a representação da previsão correta de todos os casos amostrais e 0 a representação da previsão incorreta de todos os casos amostrais (HICKS et al., 2022).

$$ACC = \frac{VP + VN}{VP + FP + VN + FN} \tag{2}$$

- Recall (REC): Também conhecida como sensibilidade ou Taxa de Verdadeiros Positivos (TVP), denota a taxa de amostras positivas corretamente classificadas e é calculada como a proporção entre as amostras positivas corretamente classificadas e todas as amostras atribuídas à classe positiva. A recall está limitada a um intervalo de [0, 1], onde 1 representa a previsão perfeita da classe positiva e 0 representa a previsão incorreta de todas as amostras da classe positiva (HICKS et al., 2022).

$$REC = \frac{VP}{VP + FN} \tag{3}$$

- Especificidade (SPEC): É a versão da classe negativa da sensibilidade e denota a taxa de amostras negativas corretamente classificadas. Ela é calculada como a proporção entre as amostras negativas corretamente classificadas e todas as amostras classificadas como negativas. A especificidade está limitada a um intervalo de [0, 1], onde 1 representa a previsão perfeita da classe negativa e 0 representa a previsão incorreta de todas as amostras da classe negativa (HICKS et al., 2022).

$$SPEC = \frac{VN}{VN + FP} \tag{4}$$

- Precisão (PREC): Denota a proporção das amostras recuperadas que são relevantes e é calculada como a proporção entre as amostras corretamente classificadas e todas as amostras atribuídas a essa classe. A precisão está limitada a um intervalo de [0, 1], onde 1 representa todas as amostras da classe corretamente previstas e 0 representa nenhuma previsão correta na classe (HICKS et al., 2022).

$$PREC = \frac{VC}{VC + FC} \tag{5}$$

onde C denota "classe"e, em classificação binária, pode ser tanto positiva (P) quanto negativa (N). O caso positivo da precisão é frequentemente referido como Valor Preditivo Positivo (PPV), e o caso negativo é frequentemente referido como Valor Preditivo Negativo (NPV). Especificamente, o PPV é a razão entre as amostras positivas corretamente classificadas e todas as amostras classificadas como positivas e é igual à precisão para a classe positiva (HICKS et al., 2022).

$$PPV = \frac{VP}{VP + FP} \tag{6}$$

- NPV: É a razão entre as amostras negativas corretamente classificadas e todas as amostras classificadas como negativas e é igual à precisão para a classe negativa (HICKS et al., 2022).

$$NPV = \frac{VN}{VN + FN} \tag{7}$$

- O escore F1 (F1): O escore F1 é a média harmônica da precisão e da recall, o que significa que penaliza valores extremos de qualquer uma delas. Esta métrica não é simétrica entre as classes, ou seja, depende de qual classe é definida como positiva e negativa. Por exemplo, no caso de uma grande classe positiva e um classificador enviesado para essa maioria, o escore F1, sendo proporcional a VP (verdadeiros positivos), seria alto. Redefinir os rótulos de classe para que a classe negativa seja a maioria e o classificador seja enviesado para a classe negativa resultaria em um baixo escore F1, embora os dados e a distribuição relativa das classes não tenham mudado. O escore F1 está limitado a um intervalo de [0, 1], onde 1 representa os valores máximos de precisão e recall, e 0 representa zero precisão e/ou recall (HICKS et al., 2022).

$$F1 = 2 \times \frac{PREC \times REC}{PREC + REC} = \frac{2 \times VP}{2 \times VP + FP + FN} \tag{8}$$

❏ Positive Likelihood Ratio: É razão entre a probabilidade de um VP e a probabilidade de um FP (BOLIN; LAM, 2013).

$$LR+ = \frac{VP}{FP} = \frac{REC}{1 - SPEC} \tag{9}$$

❏ Negative Likelihood Ratio: É razão entre a probabilidade de um FN e a probabilidade de um VN (BOLIN; LAM, 2013).

$$LR- = \frac{FN}{VN} = \frac{1 - REC}{SPEC} \tag{10}$$

❏ Curva ROC: A Curva ROC (*Receiver Operating Characteristic*) consiste na representação gráfica da performance por meio da taxa de sensibilidade e a fração dos falsos positivos (1 - SPEC). O gráfico é produzido traçando a sensibilidade (taxa de verdadeiro positivo) no eixo y contra a 1-SPEC (taxa de falso positivo) no eixo x. Uma curva ROC que segue a linha diagonal y=x produz resultados falsos positivos na mesma proporção que resultados verdadeiros positivos. Espera-se que um teste diagnóstico com precisão razoável tenha uma curva ROC no triângulo superior esquerdo acima da linha y=x, quanto maior for a área sob a curva da curva ROC, maior será a acurácia do teste, conforme mostrado na Figura 4 (HOO; CANDLISH; TEARE, 2017).

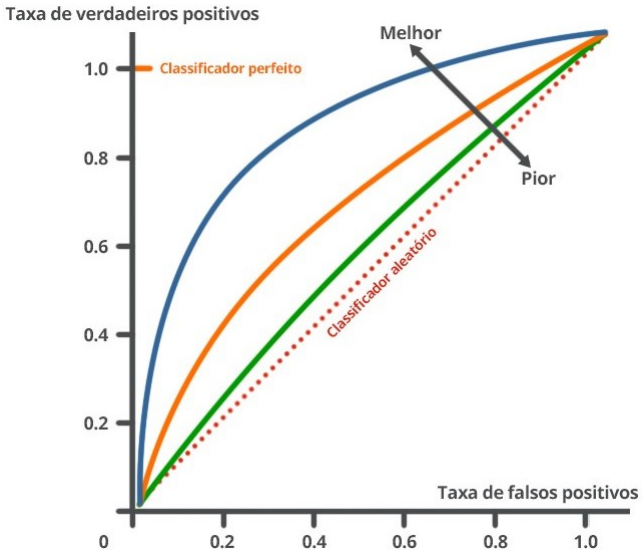

Figura 4 – Curva ROC

4.3 Alterações Propostas no J48: O Algoritmo BioJ48

O algoritmo BioJ48 foi desenvolvido com o objetivo de aprimorar a capacidade do J48 tradicional em lidar com conjuntos de dados desbalanceados, especialmente em aplicações biomédicas. As alterações propostas focam em compensar a tendência do J48 original de favorecer a classe majoritária durante a construção da árvore, o que é particularmente problemático em domínios como o diagnóstico de doenças, onde a identificação da classe minoritária é prioritária.

O BioJ48 mantém a estrutura recursiva de indução de árvores de decisão característica do C4.5 (e, por consequência, do J48), mas altera significativamente o critério de divisão dos nós internos e o comportamento dos nós folha, introduzindo medidas de correção para aumentar a sensibilidade e o equilíbrio da árvore.

As modificações podem ser agrupadas em três frentes principais:

1. **Valorização da Classe Minoritária no Cálculo do Ganho de Informação:**

   No J48 tradicional, a escolha do atributo a ser dividido em cada nó da árvore é baseada no ganho de informação, calculado a partir da entropia das partições resultantes. No entanto, esse critério tende a favorecer atributos que melhor separam a classe majoritária, ignorando a representatividade da minoria.

   No BioJ48, foi introduzido um fator de ajuste no ganho de informação, ponderado por uma constante chamada `minor_class_weight`. Esse fator é proporcional à proporção de instâncias da classe minoritária nas duas partições geradas após o split. A fórmula modificada do ganho de informação passa a incorporar esse valor:

   $$G_{\text{mod}} = G_{\text{orig}} + \alpha \cdot p_{\text{min}} \tag{11}$$

   onde:

   - G_{mod}: ganho de informação modificado;
   - α: hiperparâmetro de ajuste (`minor_class_weight`);
   - p_{min}: proporção da classe minoritária presente nas partições esquerda e direita após o split.

   Esse ajuste permite que divisões que mantêm a representatividade da classe minoritária recebam maior valor, mesmo que o ganho original de informação não seja o maior possível. O objetivo é balancear o critério de pureza com a preservação da diversidade de classes.

2. **Restrição a Splits que Eliminem a Classe Minoritária:**
   Em algoritmos tradicionais de árvores de decisão, como o J48, é comum que o processo de divisão (ou split) dos dados priorize unicamente o maior ganho de informação. No entanto, essa escolha pode resultar em partições que excluem completamente a classe minoritária

de um dos lados da divisão. Embora esse tipo de split possa maximizar a pureza local (ou seja, tornar uma partição mais homogênea), ele tem um efeito colateral grave: a classe minoritária deixa de ser representada em determinadas regiões da árvore, dificultando — ou até impossibilitando — que ela seja corretamente classificada nos nós seguintes.

Para contornar esse problema, o BioJ48 impõe uma regra de validação adicional: Uma divisão só é aceita se a classe minoritária estiver presente em ambas as partições resultantes. Caso contrário, a divisão é descartada, mesmo que o valor do ganho de informação seja alto. Essa verificação é implementada diretamente dentro da função `find_best_split`. Essa decisão foi tomada com base no princípio de que preservar a representatividade da classe minoritária ao longo da árvore é mais importante do que maximizar o ganho informacional em um único passo.

Essa abordagem garante que a classe de interesse (geralmente a mais difícil de identificar) permaneça visível para o algoritmo durante a construção de toda a árvore. Isso evita o "apagamento" da minoria e incentiva a seleção de atributos que produzam divisões mais equilibradas entre as classes. Essa restrição é especialmente importante em contextos clínicos, onde a identificação de um caso positivo (como um paciente com determinada doença) não pode ser comprometida em nome de uma acurácia global aparentemente maior, mas enviesada.

Em resumo, a restrição a splits que eliminam a classe minoritária força o modelo a construir caminhos que mantenham a diversidade de classes em todas as ramificações, promovendo um aprendizado mais justo e sensível aos desequilíbrios do conjunto de dados.

3. **Ajuste Probabilístico nos Nós Folha:**
   Ao final da indução, nos nós folha, o J48 tradicional utiliza a maioria simples para determinar a classe predita. Essa abordagem é problemática quando o conjunto de treinamento já é desbalanceado, pois o modelo tende a prever sistematicamente a classe majoritária.

   Para corrigir isso, o BioJ48 aplica um reforço probabilístico à distribuição da classe minoritária nas folhas, baseado na razão de desbalanceamento global. Esse reforço é calculado com uma função logarítmica suavizada:

$$Refinamento = \log\left(1 + \frac{n_{\text{maioria}}}{n_{\text{minoria}}}\right) \tag{12}$$

   Esse valor é incorporado à probabilidade da classe minoritária antes da decisão final no nó folha, garantindo que mesmo em regiões da árvore onde a minoria está sub-representada, sua probabilidade de predição não seja desprezível.

Essas modificações tornam o algoritmo mais sensível a padrões relacionados à classe com menor representatividade, sem comprometer significativamente outras métricas como acurácia e especificidade.

O BioJ48 foi inteiramente implementado em Python puro, utilizando apenas bibliotecas fundamentais como numpy. A construção da árvore segue a lógica recursiva, com controle de profundidade (`max_depth`) e tratamento dos casos-base (nós homogêneos ou com número de amostras insuficiente). A implementação está disponível no Apêndice 6.2 para reprodutibilidade.

Capítulo 5

Experimentos e Análise dos Resultados

Neste capítulo são apresentados os experimentos conduzidos para avaliar o desempenho do algoritmo proposto, BioJ48, em comparação com o J48 tradicional, frente a um problema real de classificação binária com dados desbalanceados. O foco principal está em avaliar a eficácia da modificação em diferentes configurações de validação, utilizando métricas que permitam capturar a capacidade preditiva em relação à classe minoritária. Foram utilizadas métricas clássicas de avaliação como Acurácia, Especificidade, F1-Score e Sensibilidade, com ênfase na última, dada a sua relevância em contextos clínicos, onde a identificação de casos positivos de hanseníase (classe minoritária) é crucial.

5.1 Contextualização do Experimento

O conjunto de dados utilizado contém informações clínicas sorológicas de pacientes analisados para diagnóstico de hanseníase. São 66 registros no total, com 16 amostras da classe positiva ("Doente") e 50 da classe negativa ("Não Doente"), resultando em um desbalanceamento de aproximadamente 1:3. O desafio proposto está justamente na capacidade dos modelos de classificação detectarem corretamente a classe com menor representatividade, algo de alta relevância no contexto médico. Trata-se de um cenário típico de desbalanceamento.

O objetivo da tarefa de classificação é prever corretamente a condição de saúde dos indivíduos com base em sete variáveis numéricas correspondentes a marcadores imunológicos de subclasses IgG1, com especial atenção à correta identificação dos casos positivos, que representam a classe minoritária.

A severidade do desbalanceamento impõe ao classificador o desafio de não ignorar a classe minoritária. Modelos tradicionais como o J48, ao priorizarem acurácia global, frequentemente falham em capturar os casos positivos, resultando em alta taxa de falsos negativos — um resultado crítico em cenários médicos.

Para garantir rigor e reprodutibilidade, a execução dos algoritmos foi automatizada por meio de scripts em Python. Foi desenvolvida uma rotina utilizando a biblioteca subprocess que executa chamadas ao Weka para o J48 original e desenvolvido um código nativo do BioJ48 (im-

plementado totalmente Python puro). Ambas as versões foram avaliadas nos mesmos conjuntos de dados e sob as mesmas condições experimentais garantindo rastreabilidade dos parâmetros e dos resultados. Os scripts completos estão disponíveis no Apêndice 6.2.

5.1.1 Procedimentos de Avaliação

Foram consideradas duas abordagens principais para os experimentos:

- **Treinamento completo (full training):** Todo o conjunto de dados foi utilizado para treinamento, e as métricas foram calculadas com base na própria predição sobre os dados vistos. Embora essa configuração não reflita um cenário real de generalização, essa técnica permite identificar o potencial máximo de aprendizado do algoritmo, especialmente útil em conjuntos pequenos. Este cenário é proposto visando maior foco no desempenho da métrica de sensibilidade entre os algoritmos.

- **Validação cruzada com todas as 66 sementes possíveis:** Para cada uma das 66 sementes, foi realizada uma separação estratificada de 70% para treino e 30% para teste, repetindo o processo para cobrir todas as possíveis combinações. As métricas foram calculadas em cada execução e, posteriormente, agregadas com estatísticas descritivas, como média, desvio padrão e intervalo de confiança de 95%;

5.2 Resultados Obtidos

A Tabela 2 apresenta os resultados obtidos com os dois algoritmos na abordagem de Treinamento Completo (full training), cenário em que todo o conjunto de dados foi utilizado para treinamento do algoritmo.

Tabela 2 – Comparação entre os algoritmos J48 e BioJ48 quanto às métricas de desempenho em cenário de treinamento completo (full training).

Métrica	J48	BioJ48
Acurácia	**0,757**	0,591
Especificidade	**1,000**	0,580
F1-Score	0,000	**0,426**
Sensibilidade	0,000	**0,625**

Conforme a Tabela 2, na execução de full training, o BioJ48 alcançou 62,5% de sensibilidade e 42,6% de F1-score, enquanto o J48 permaneceu 0% em ambas métricas, reforçando a tendência do algoritmo tradicional em ignorar completamente a classe minoritária no treinamento com dados altamente desbalanceados.

A Tabela 3 resume os principais resultados obtidos com os dois algoritmos, destacando o número de execuções em que a métrica resultou em valor zero (indicando falha completa na detecção), a média e o intervalo de confiança para cada métrica analisada.

Tabela 3 – Comparação entre os algoritmos J48 e BioJ48 quanto às métricas de desempenho com seus respectivos intervalos de confiança de 95% e a quantidade de métricas com resultado zerado (Cenário de validação cruzada com todas as 66 sementes possíveis).

Métrica	**J48**			**BioJ48**		
	Qtd zerados	**Média**	**IC (95%)**	**Qtd zerados**	**Média**	**IC (95%)**
Acurácia	25	0,414	[0,329; 0,499]	**0**	**0,508**	[0,482; 0,534]
Especificidade	25	**0,504**	[0,398; 0,612]	**0**	0,494	[0,465; 0,524]
F1-Score	46	0,098	[0,058; 0,137]	**4**	**0,330**	**[0,290; 0,370]**
Sensibilidade	46	0,156	[0,086; 0,225]	**4**	**0,540**	**[0,477; 0,603]**

O algoritmo J48 tradicional apresentou comportamento problemático: zerou a sensibilidade em 46 execuções (quase 70%) indicando a completa incapacidade do modelo em detectar a classe "Doente"na maioria das execuções, o que é crítico em contextos clínicos. Esse resultado é sintomático de classificadores que maximizam acurácia às custas da classe de interesse. Em contrapartida, o BioJ48 resultou apenas 4 execuções com sensibilidade zerada (6%), elevando o valor médio dessa métrica de 0,156 para 0,540, com ganho também no F1-Score que passou de 0,098 no J48 para 0,330, ambas métricas apresentaram aumentos estatisticamente significantes comprovados por meio do Intervalo de Confiança (IC) ao nível de 95% de confiança. Isso demonstra que o BioJ48 consegue detectar corretamente mais pacientes doentes sem comprometer significativamente a especificidade, o que indica um equilíbrio mais saudável entre precisão e sensibilidade.

O aumento na sensibilidade do BioJ48 decorre principalmente das modificações na função de seleção de atributos, que passou a considerar um fator de valorização da classe minoritária no ganho de informação. Além disso, a poda balanceada e o reforço probabilístico nos nós folha asseguram que a árvore mantenha representatividade da classe minoritária mesmo em regiões pouco povoadas do espaço amostral.

Esses resultados confirmam que o BioJ48 consegue equilibrar o foco entre as classes, entregando ganhos significativos em sensibilidade e F1-score, mesmo com moderado sacrifício em especificidade, o que é aceitável em diagnósticos médicos onde o objetivo principal é evitar falsos negativos.

A modificação realizada no algoritmo original — que consiste em ajustes no cálculo de ganho de informação para valorizar a classe minoritária — mostrou-se eficaz. A inclusão de um fator de correção baseado na razão de desbalanceamento e na presença da classe minoritária em ambas as partições após o split garantiu uma árvore mais justa e sensível aos casos positivos.

Embora a especificidade do BioJ48 tenha apresentado uma redução foi considerada aceitável, pois o impacto foi mínimo se comparado ao aumento expressivo de sensibilidade e também dado o contexto médico, em que é preferível cometer falsos positivos (exames complementares desnecessários) a deixar de diagnosticar casos reais (falsos negativos). A melhora no F1-Score evidencia o equilíbrio entre precisão e sensibilidade obtido com o novo algoritmo.

Capítulo 6

Conclusão

Este trabalho apresentou o BioJ48, uma versão modificada do algoritmo de árvore de decisão J48, com o objetivo de melhorar a detecção de classes minoritárias em contextos biomédicos, onde o desbalanceamento de classes é um problema recorrente e de grande impacto clínico.

A modificação proposta concentrou-se em três pilares:

- Valorização da classe minoritária no cálculo do ganho de informação, utilizando uma função de ajuste proporcional à representatividade da classe de interesse;
- Prevenção de divisões que eliminem completamente a classe minoritária, promovendo partições mais inclusivas e justas;
- Ajuste probabilístico nos nós folha, corrigindo a distribuição de probabilidade final para mitigar o viés de amostras desbalanceadas.

Os experimentos comprovaram que o BioJ48 supera significativamente o J48 tradicional em termos de sensibilidade e F1-score, sem comprometer drasticamente outras métricas como especificidade e acurácia. O modelo proposto consegue manter a interpretabilidade e, ao mesmo tempo, melhora a performance na detecção da classe minoritária — o que é altamente desejável em aplicações clínicas.

Além disso, o BioJ48 demonstrou maior robustez em execuções repetidas, frente a diferentes particionamentos do conjunto de dados, evidenciado pela baixa variação entre as métricas e a drástica redução no número de execuções com sensibilidade igual a zero. Esse comportamento é particularmente relevante em aplicações clínicas, onde a falha na detecção de um caso positivo pode representar riscos à saúde do paciente.

A principal inovação do BioJ48 está na reformulação do critério de divisão e na introdução de um reforço probabilístico adaptativo nos nós folha. Tais modificações demonstraram que é possível preservar a interpretabilidade da árvore — característica crucial em ambientes regulados como a medicina — enquanto se melhora substancialmente a capacidade preditiva para a classe de menor ocorrência.

Comparado ao estado da arte, a quantidade de pesquisas que abordam soluções para pequenos datasets com desbalanceamentos de classes é notavelmente escassa. Essa lacuna na literatura evidencia a inovação e a forte contribuição desta abordagem dentro do campo. A aplicação do BioJ48 pode auxiliar significativamente os profissionais de saúde na tomada de decisões, contribuindo com ferramentas mais eficazes para o diagnóstico clínico.

6.1 Trabalhos Futuros

Como continuidade deste trabalho, propõe-se o desenvolvimento de uma ferramenta prática baseada no algoritmo BioJ48, com o objetivo de torná-lo acessível a profissionais da área da saúde ou pesquisadores que lidam com conjuntos de dados desbalanceados.

Uma das etapas previstas é a implementação do BioJ48 como uma biblioteca Python de código aberto, de forma a facilitar sua integração em pipelines de ciência de dados e aplicações clínicas. A biblioteca deverá incluir funcionalidades como visualização da árvore, análise de sensibilidade por classe, explicabilidade das decisões e ajuste dinâmico do peso atribuído à classe minoritária.

Além disso, pretende-se realizar uma avaliação sistemática do desempenho do BioJ48 sob diferentes cenários de desbalanceamento, incluindo:

- Casos de desbalanceamento severo, como os enfrentados neste estudo;
- Casos de desbalanceamento moderado;
- E também cenários com classes balanceadas, a fim de verificar se o algoritmo mantém sua estabilidade e não introduz viés desnecessário quando não há desbalanceamento relevante.

Também se pretende ampliar os experimentos comparativos com outros algoritmos de classificação mais modernos e complexos, como XGBoost, LightGBM e redes neurais profundas. O objetivo é avaliar se o BioJ48, mesmo sendo um modelo relativamente simples, permanece competitivo em termos de desempenho, especialmente em métricas sensíveis à classe minoritária.

Essas investigações permitirá avaliar a robustez e a adaptabilidade do algoritmo em contextos mais amplos, e poderá contribuir para uma configuração automática do parâmetro de valorização da classe minoritária, de acordo com a natureza dos dados.

Espera-se que essa iniciativa contribua para a disseminação e usabilidade do algoritmo, além de promover colaborações e validações em diferentes domínios biomédicos ou conjuntos de dados reais.

6.2 Contribuições em Produção Bibliográfica

A presente dissertação também gerou uma contribuição relevante para a produção científica da área, com a submissão de um artigo derivado à revista internacional Bioinformatics (Oxford

Academic), uma das publicações de maior prestígio no campo da bioinformática e da inteligência artificial aplicada à biologia.

Referências

AIRES, R. B. et al. Thromboelastometry demonstrates endogenous coagulation activation in nonsevere and severe covid-19 patients and has applicability as a decision algorithm for intervention. **PLOS ONE**, Public Library of Science, v. 17, n. 1, p. 1–19, 01 2022. Disponível em: <https://doi.org/10.1371/journal.pone.0262600>.

ALIE, M. S.; NEGESSE, Y. Machine learning prediction of adolescent hiv testing services in ethiopia. **Frontiers in Public Health**, 2023. Disponível em: <https://doi.org/10.3389/fpubh.2024.1341279>.

ALVES, A. H. da S. et al. Ig-cee: An embedded information gain approach to genetic algorithms. In: IEEE. **2021 IEEE Congress on Evolutionary Computation (CEC)**. 2021. p. 1086–1092. Disponível em: <https://doi.org/10.1109/CEC45853.2021.9504776>.

AMARAL, E. R. H. J. Laurence Rodrigues do. Never-ending learning principles in gene ontology classification using genetic algorithms. In: IEEE. **2012 IEEE Congress on Evolutionary Computation (CEC)**. 2012. p. 1–8. Disponível em: <https://doi.org/10.1109/CEC.2012.6256637>.

AMARAL, L. R. do et al. Applying never-ending learning (nel) principles to build a gene ontology (go) biocurator. In: IEEE. **2021 IEEE Congress on Evolutionary Computation (CEC)**. 2021. p. 458–465. Disponível em: <https://doi.org/10.1109/CEC45853.2021.9504981>.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Springer, 2006. Disponível em: <https://dl.acm.org/doi/10.5555/1162264>.

BOLIN, E.; LAM, W. A review of sensitivity, specificity, and likelihood ratios: evaluating the utility of the electrocardiogram as a screening tool in hypertrophic cardiomyopathy. **Congenital Heart Disease**, v. 8, n. 5, p. 406–410, may. 2013. ISSN 1176-9351. Disponível em: <https://doi.org/10.1111/chd.12083>.

Brasil. Ministério da Saúde. **Diretrizes para Vigilância, Atenção e Eliminação da Hanseníase como Problema de Saúde Pública: manual técnico-operacional**. Brasília: Ministério da Saúde, 2022. Disponível em: <https://biblioteca.cofen.gov.br/wp-content/uploads/2022/01/diretrizes-vigilancia-atencao-eliminacao-hanseniase.pdf>.

CHAPELLE, O.; SCHOLKOPF, B.; ZIEN EDS., A. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. **IEEE Transactions on Neural Networks**, v. 20, n. 3, p. 542–542, 2009. Disponível em: <https://doi.org/10.1109/TNN.2009.2015974>.

CRUZ, J. A.; WISHART, D. S. Applications of machine learning in cancer prediction and prognosis. **Cancer Informatics**, n. 2, p. 59–77, feb. 2006. ISSN 1176-9351. Disponível em: <https://doi.org/10.1177/117693510600200030>.

FERNÁNDEZ, A. et al. **Foundations on Imbalanced Classification**. Cham: Springer International Publishing, 2018. 19–46 p. ISBN 978-3-319-98074-4. Disponível em: <https://doi.org/10.1007/978-3-319-98074-4_2>.

FRADICO, J. R. B. et al. Serum soluble mediators as prognostic biomarkers for morbidity, disease outcome, and late-relapsing hepatitis in yellow fever patients. **Clinical Immunology**, v. 251, p. 109321, 2023. ISSN 1521-6616. Disponível em: <https://doi.org/10.1016/j.clim.2023.109321>.

HICKS, S. A. et al. On evaluation metrics for medical applications of artificial intelligence. **Scientific Reports**, v. 12, n. 5979, apr. 2022. Disponível em: <https://doi.org/10.1038/s41598-022-09954-8>.

HOO, Z. H.; CANDLISH, J.; TEARE, D. What is an roc curve? **Emergency Medicine Journal**, British Association for Accident and Emergency Medicine, v. 34, n. 6, p. 357–359, 2017. ISSN 1472-0205. Disponível em: <https://doi.org/10.1136/emermed-2017-206735>.

KAPOOR, P.; RANI, R. Efficient decision tree algorithm using j48 and reduced error pruning. **International Journal of Engineering Research and General Science**, v. 3, n. 3, may. 2015. ISSN 2091-2730. Disponível em: <http://pnrsolution.org/Datacenter/Vol3/Issue3/211.pdf>.

KATZ ASAF SHABTAI, L. R. G.; OFEK, N. Confdtree: A statistical method for improving decision trees. **Journal of Computer Science and Technology**, v. 29, n. 3, p. 392–407, may. 2014. Disponível em: <https://doi.org/10.1007/s11390-014-1438-5>.

MARTINS, P. R. et al. Association of human papillomavirus (hpv), p16, p53 and p63 expression with non-bilharzia-associated squamous cell carcinoma of the bladder and algorithm construction for histopathological grading prediction. **einstein (São Paulo)**, v. 21, 2023. ISSN 1679-4508. Disponível em: <https://doi.org/10.31744/einstein_journal/2023AO0109>.

MENDES, R. de L. et al. gacnn: Composing cnns and gas to build an optimized hybrid classification architecture. In: IEEE. **2021 IEEE Congress on Evolutionary Computation (CEC)**. 2021. p. 79–86. Disponível em: <https://doi.org/10.1109/CEC45853.2021.9504850>.

MITCHELL, T. **Machine Learning**. McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673. Disponível em: <https://books.google.com.br/books?id=EoYBngEACAAJ>.

MOURA, R. S.; CALADO, K. L.; OLIVEIRA, M. L. W. Leprosy serology using pgl-i: a systematic review. **Transactions of the Royal Society of Tropical Medicine and Hygiene**, v. 117, n. 1, p. 10–18, 2023. Disponível em: <https://doi.org/10.1590/S0037-86822008000700004>.

NERY, J. A. C. et al. Advances in the understanding of leprosy: contribution of molecular biology and immunology. **Anais Brasileiros de Dermatologia**, v. 89, n. 6, p. 849–860, 2014.

PEREIRA, M. H. et al. Hanseníase: diagnóstico, tratamento e complicações neurológicas. **Revista Brasileira de Neurologia**, v. 56, n. 1, p. 33–41, 2020.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

RODRIGUES, F. A.; AMARAL, L. R. D. Aplicação de métodos computacionais de mineração de dados na classificação e seleção de oncogenes medidos por microarray. **Revista Brasileira de Cancerologia**, v. 58, p. 241–249, 2012. Disponível em: <https://doi.org/10.32635/2176-9745.RBC.2012v58n2.625>.

SAHU, S.; MEHTRE, B. An enhanced j48 classification algorithm for the anomaly intrusion detection systems. **Cluster Computing**, 2015. Disponível em: <https://doi.org/10.1007/s10586-017-1109-8>.

SCHAUERHUBER, A. et al. Better trees: an empirical study on hyperparameter tuning of classification decision tree induction algorithms. **Data Mining and Knowledge Discovery**, 2021. Disponível em: <https://doi.org/10.1007/s10618-024-01002-5>.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.

World Health Organization. Global leprosy update 2020: Impact of covid-19 on the global leprosy control. **Weekly Epidemiological Record**, v. 96, n. 36, p. 421–444, 2021.

APÊNDICE

A seguir é apresentado o código da análise exploratória e o código completo do algoritmo BioJ48, desenvolvido em linguagem Python. O código implementa as alterações descritas no Capítulo 4.3, como o ajuste dinâmico de ganho de informação e o reforço probabilístico para classes minoritárias.

Este código foi utilizado nos experimentos descritos no Capítulo 5 e está disponível também em repositório digital para reprodutibilidade da pesquisa.

Listing 1 – Implementação de chamada do Algoritmo J48 do Weka em Python

```
import os
import re
import subprocess
from sklearn.model_selection import train_test_split
import pandas as pd

def encontrar_weka_jar(drive='C:\\'):
    for root, dirs, files in os.walk(drive):
        if 'weka.jar' in files:
            return os.path.join(root, 'weka.jar')
    return None

weka_path = encontrar_weka_jar()
print("weka.jar encontrado em:", weka_path)

def clean_csv(df):
    df = df.fillna('?')
    for col in df.columns:
        df[col] = df[col].map(lambda x: str(x).replace('\n', ' ')
            .replace('\r', ' ').strip())
    return df
```

```
21
22 def run_j48_weka_metrics(
23     weka_jar_path,
24     dataset_path,
25     class_index="last",
26     mode="cv",  # opcoes: "cv", "loocv", "split", "train"
27     seed=1,
28     split_ratio=70,
29     folds=10,
30     params="-C␣0.25␣-M␣2"
31 ):
32     train_csv = "train_temp.csv"
33     test_csv = "test_temp.csv"
34
35     if mode == "split":
36         df = pd.read_csv(dataset_path)
37         df = clean_csv(df)
38
39         class_col = df.columns[-1]
40         unique_classes = df[class_col].unique()
41
42         # Linhas dummy para garantir as classes
43         dummy_rows = []
44         for cls in unique_classes:
45             dummy = df.iloc[0].copy()
46             dummy[:] = "?"
47             dummy[class_col] = cls
48             dummy_rows.append(dummy)
49
50         df = pd.concat([pd.DataFrame(dummy_rows), df],
               ignore_index=True)
51
52         # Split
53         train_df, test_df = train_test_split(df, test_size=(100 -
               split_ratio) / 100, random_state=seed, shuffle=True)
54
55         # Remove dummies
56         train_df = train_df[~train_df.index.isin(range(len(
               dummy_rows)))].reset_index(drop=True)
57         test_df = test_df[~test_df.index.isin(range(len(
               dummy_rows)))].reset_index(drop=True)
58
```

```
59          # Salva os arquivos
60          train_df.to_csv(train_csv, index=False)
61          test_df.to_csv(test_csv, index=False)
62
63          command = [
64              "java", "-cp", weka_jar_path,
65              "weka.classifiers.trees.J48",
66              "-t", train_csv,
67              "-T", test_csv,
68              "-c", str(class_index)
69          ]
70      else:
71          command = [
72              "java", "-cp", weka_jar_path,
73              "weka.classifiers.trees.J48",
74              "-t", dataset_path,
75              "-c", str(class_index)
76          ]
77
78          if mode == "loocv":
79              num_instances = len(pd.read_csv(dataset_path))
80              command += ["-x", str(num_instances), "-s", str(seed)
                    ]
81          elif mode == "cv":
82              command += ["-x", str(folds), "-s", str(seed)]
83          elif mode == "train":
84              command += ["-no-cv"]
85
86      if params:
87          command += params.split()
88
89      # Executa o Weka
90      result = subprocess.run(command, stdout=subprocess.PIPE,
           stderr=subprocess.PIPE, text=True)
91      stdout = result.stdout.replace(',', '.')
92
93      #       Se for split, isolamos o bloco de teste
94      if mode == "split":
95          parts = re.split(r"(?i)Time␣taken␣to␣test␣model␣on␣test␣
               data:", stdout)
96          if len(parts) > 1:
97              stdout = parts[1]  # so pega o bloco de avaliacao no
```

```
            test.csv
98
99      # === Acuracia ===
100     acc_match = re.search(r"Correctly␣Classified␣Instances\s+\d+\
           s+([\d.]+)", stdout)
101     acuracia_j48 = float(acc_match.group(1))/100 if acc_match
           else 0.0
102
103     # === Confusion Matrix ===
104     cm = re.search(r"Confusion␣Matrix␣===\n\n(.*?)(\n\n|\Z)",
           stdout, re.DOTALL)
105     TP = TN = FP = FN = 0
106
107     if cm:
108         lines = cm.group(1).strip().split('\n')
109         numeric_rows = []
110         class_labels = []
111
112         for line in lines:
113             if "|" in line:
114                 parts = line.strip().split('|')
115                 try:
116                     nums = list(map(int, parts[0].split()))
117                     label = parts[1].split('=')[-1].strip()
118                     numeric_rows.append(nums)
119                     class_labels.append(label)
120                 except:
121                     continue
122
123         if "PB_" in class_labels:
124             pos_index = class_labels.index("PB_")
125             neg_index = 1 - pos_index
126             TP = numeric_rows[pos_index][pos_index]
127             FN = numeric_rows[pos_index][neg_index]
128             FP = numeric_rows[neg_index][pos_index]
129             TN = numeric_rows[neg_index][neg_index]
130
131     # === Metricas ===
132     recall_j48 = TP / (TP + FN) if (TP + FN) > 0 else 0.0
133     spec_j48 = TN / (TN + FP) if (TN + FP) > 0 else 0.0
134     sens_j48 = recall_j48
135     precision = TP / (TP + FP) if (TP + FP) > 0 else 0.0
```

```
136     Fscore_j48 = 2 * (precision * recall_j48) / (precision +
            recall_j48) if (precision + recall_j48) > 0 else 0.0
137
138     return acuracia_j48, sens_j48, spec_j48, recall_j48,
            Fscore_j48, TP, TN, FP, FN, stdout, result.stderr
```

Listing 2 – Análise exploratória dos dados

```
1  import os
2  import pandas as pd
3  import numpy as np
4  import plotly.express as px
5  import plotly.graph_objects as go
6  import plotly.io as pio
7  from plotly.subplots import make_subplots
8
9  # Carregar o dataset
10 df = pd.read_csv('7_1_PB_NDoentes_IgG1.csv')
11 df['Class'] = np.where(df['Class']=='PB_', 'Doente','Nao Doente')
12
13 # Informacoes gerais
14 print("\nInformacoes do dataset:")
15 print(df.info())
16
17 # Estat sticas descritivas
18 print("\nEstat sticas descritivas:")
19 print(df.describe())
20
21 # Verificacao de valores ausentes
22 print("\nValores ausentes por coluna:")
23 print(df.isnull().sum())
24
25 # Seleciona apenas as colunas numericas
26 numeric_columns = df.select_dtypes(include='number').columns
27
28 # Analise de outliers usando IQR
29 print("\nOutliers detectados por variavel (metodo IQR):")
30 for col in numeric_columns:
31     Q1 = df[col].quantile(0.25)
32     Q3 = df[col].quantile(0.75)
33     IQR = Q3 - Q1
34     lower = Q1 - 1.5 * IQR
```

```
35      upper = Q3 + 1.5 * IQR
36      outliers = df[(df[col] < lower) | (df[col] > upper)]
37      print(f"{col}: {len(outliers)} outlier(s)")
38
39  ## Plots
40  # Criar diretorio para exportacao das imagens
41  output_dir = "EDA_Plots"
42  os.makedirs(output_dir, exist_ok=True)
43
44  # Grafico 1: Pizza da distribuicao da variavel alvo com legendas
        e rotulos grandes
45  fig_class_dist = px.pie(
46      df,
47      names='Class',
48      title='',
49      hole=0.0
50  )
51
52  # Ajustes de layout e fonte
53  fig_class_dist.update_traces(
54      textposition='inside',
55      textfont=dict(size=18)
56  )
57  fig_class_dist.update_layout(
58      legend=dict(font=dict(size=16)),
59      title_font=dict(size=20)
60  )
61
62  fig_class_dist.show()
63  # Salvar (se tiver kaleido instalado)
64  pio.write_image(fig_class_dist, f"{output_dir}/
        distribuicao_classes.png")
65
66  # 2. Boxplots em subplots
67  from plotly.subplots import make_subplots
68  import plotly.graph_objects as go
69
70  # Lista de variaveis numericas
71  numeric_cols = df.select_dtypes(include='number').columns.tolist
        ()
72  n_vars = len(numeric_cols)
73
```

```
74 # Subplots: +1 para o boxplot geral
75 total_plots = n_vars + 1
76 cols = 2
77 rows = (total_plots + cols - 1) // cols
78
79 # Criar grid
80 fig_box = make_subplots(
81     rows=rows,
82     cols=cols,
83     subplot_titles=numeric_cols + ["Distribuicao Geral das
       Variaveis"]
84 )
85
86 # Adicionar boxplots separados por classe
87 cores = {
88     'Doente': 'royalblue',
89     'Nao Doente': 'tomato'
90 }
91
92 for i, col in enumerate(numeric_cols):
93     row = (i // cols) + 1
94     col_idx = (i % cols) + 1
95     for cls in ['Doente', 'Nao Doente']:
96         fig_box.add_trace(
97             go.Box(
98                 y=df[df['Class'] == cls][col],
99                 name=cls,
100                marker=dict(color=cores[cls]),
101                boxmean=True,
102                showlegend=(i == 0)
103            ),
104            row=row,
105            col=col_idx
106        )
107
108 # Adicionar boxplot geral (sem separar classe)
109 row = (n_vars // cols) + 1
110 col_idx = (n_vars % cols) + 1
111
112 for col in numeric_cols:
113     fig_box.add_trace(
114         go.Box(
```

```
            y=df[col],
            name=col,
            marker=dict(color='gray'),
            boxmean=True,
            showlegend=False
        ),
        row=row,
        col=col_idx
    )

# Ajustar layout
fig_box.update_layout(
    height=300 * rows,
    width=1200,
    title_text="",
    title_font=dict(size=20),
    legend=dict(font=dict(size=14)),
    font=dict(size=12)
)

fig_box.show()
pio.write_image(fig_box, f"{output_dir}/boxplots_subplots.png",
    width=1200, height=300 * rows)

# Grafico 3: Heatmap da correlacao
correlation = df.select_dtypes(include='number').corr()
heatmap = go.Figure(data=go.Heatmap(
    z=correlation.values,
    x=correlation.columns,
    y=correlation.index,
    colorscale='Viridis',
    zmin=-1, zmax=1,
    colorbar=dict(title='Correlacao')
))
heatmap.update_layout(
    title='',
    title_font=dict(size=20),
    xaxis=dict(title_font=dict(size=16), tickfont=dict(size=14)),
    yaxis=dict(title_font=dict(size=16), tickfont=dict(size=14))
)
heatmap.show()
pio.write_image(heatmap, f"{output_dir}/heatmap_correlacao.png")
```

Listing 3 – Implementação do Algoritmo BioJ48 em Python

```
import numpy as np

class BioJ48Classifier:
    """Classe para representar um no da arvore de decisao."""
    def __init__(self, attribute=None, threshold=None, left=None,
        right=None, value=None, class_prob=None):
        self.attribute = attribute
        self.threshold = threshold
        self.left = left
        self.right = right
        self.value = value  # Classe escolhida por votacao
           ponderada
        self.class_prob = class_prob  # Probabilidades das
           classes no no folha

def bioj48(X, y, max_depth=None, minor_class_weight=2.0):
    """Funcao principal que encapsula a construcao e predicao da
       arvore BioJ48."""

    def calculate_entropy(y):
        """Calcula a entropia de um conjunto de classes."""
        if len(y) == 0:
            return 0
        _, counts = np.unique(y, return_counts=True)
        probabilities = counts / counts.sum()
        return -np.sum(probabilities * np.log2(probabilities))

    def split_dataset(X, y, attribute, threshold):
        """Divide o conjunto de dados em duas partes baseado no
           atributo e limiar."""
        left_mask = X[:, attribute] < threshold
        right_mask = ~left_mask
        return X[left_mask], X[right_mask], y[left_mask], y[
           right_mask]

    def find_best_split(X, y):
        """Encontra a melhor divisao de atributo e valor de
           limiar para o conjunto de dados."""
        best_gain = -np.inf
        best_attribute = None
        best_threshold = None
```

```
35
36          unique_classes, counts = np.unique(y, return_counts=True)
37          minor_class = unique_classes[np.argmin(counts)]
38          major_class = unique_classes[np.argmax(counts)]
39
40          for attribute in range(X.shape[1]):
41              unique_values = np.unique(X[:, attribute])
42              for threshold in unique_values:
43                  X_left, X_right, y_left, y_right = split_dataset(
                        X, y, attribute, threshold)
44
45                  # Evita splits que eliminam a classe minoritaria
46                  if minor_class in y and (minor_class not in
                        y_left or minor_class not in y_right):
47                      continue
48
49                  parent_entropy = calculate_entropy(y)
50                  left_entropy = calculate_entropy(y_left)
51                  right_entropy = calculate_entropy(y_right)
52
53                  n = len(y)
54                  left_weight = len(y_left) / n
55                  right_weight = len(y_right) / n
56
57                  # Ajuste para favorecer a classe minoritaria
58                  minor_class_ratio_left = (y_left == minor_class).
                        sum() / len(y_left) if len(y_left) > 0 else 0
59                  minor_class_ratio_right = (y_right == minor_class
                        ).sum() / len(y_right) if len(y_right) > 0 else
                         0
60
61                  # Aumenta a importancia da classe minoritaria no
                        ganho de informacao
62                  minor_adjustment = (minor_class_ratio_left +
                        minor_class_ratio_right) * minor_class_weight
63
64                  child_entropy = left_weight * left_entropy +
                        right_weight * right_entropy
65                  gain = parent_entropy - child_entropy +
                        minor_adjustment
66
67                  if gain > best_gain:
```

```
68                      best_gain, best_attribute, best_threshold =
                            gain, attribute, threshold
69
70          return best_attribute, best_threshold
71
72      def build_tree_bio(X, y, max_depth):
73          """Constroi a arvore de decisao recursivamente."""
74          unique_classes, class_counts = np.unique(y, return_counts
                =True)
75          minor_class = unique_classes[np.argmin(class_counts)]
76          major_class = unique_classes[np.argmax(class_counts)]
77
78          # Novo Criterio de Parada
79          # Se a classe minoritaria foi completamente eliminada,
                reequilibra os pesos no no folha
80          if len(unique_classes) == 1:
81              return BioJ48Classifier(value=y[0])
82
83          if max_depth == 0 or X.shape[0] == 0:
84              class_prob = class_counts / class_counts.sum()
85              return BioJ48Classifier(value=np.random.choice(
                    unique_classes, p=class_prob), class_prob=dict(zip(
                    unique_classes, class_prob)))
86
87          best_attribute, best_threshold = find_best_split(X, y)
88
89          if best_attribute is None:
90              class_prob = class_counts / class_counts.sum()
91
92              # Ajuste Dinamico do Peso da Classe Minoritaria
93              imbalance_ratio = class_counts.max() / class_counts.
                    min()  # Razao de desbalanceamento
94
95              # Usando logaritmo para crescimento ainda mais lento
96              minor_prob_boost = min(0.9, np.log(1 +
                    imbalance_ratio) / (1 + np.log(1 + imbalance_ratio)
                    ))
97
98              if minor_class in unique_classes:
99                  minor_index = unique_classes.tolist().index(
                        minor_class)
100                 class_prob[minor_index] += minor_prob_boost  #
```

```
                    Aumenta a chance da classe minoritaria
101
102             # Correcao: Normalizar probabilidades para evitar
                    erro de soma     1
103             class_prob /= class_prob.sum()
104
105             return BioJ48Classifier(value=np.random.choice(
                    unique_classes, p=class_prob), class_prob=dict(zip(
                    unique_classes, class_prob)))
106
107         X_left, X_right, y_left, y_right = split_dataset(X, y,
                best_attribute, best_threshold)
108
109         # Poda Balanceada
110         if len(y_left) == 0 or len(y_right) == 0:
111             class_prob = class_counts / class_counts.sum()
112             return BioJ48Classifier(value=np.random.choice(
                    unique_classes, p=class_prob), class_prob=dict(zip(
                    unique_classes, class_prob)))
113
114         left_child = build_tree_bio(X_left, y_left, max_depth - 1
                if max_depth else None)
115         right_child = build_tree_bio(X_right, y_right, max_depth
                - 1 if max_depth else None)
116
117         return BioJ48Classifier(attribute=best_attribute,
                threshold=best_threshold, left=left_child, right=
                right_child)
118
119     def predict_tree(tree, x):
120         """Classifica um exemplo 'x' usando a arvore treinada."""
121         if tree.class_prob is not None:
122             return np.random.choice(list(tree.class_prob.keys()),
                        p=list(tree.class_prob.values()))
123         if tree.attribute is None or tree.threshold is None:
124             return tree.value
125         if x[tree.attribute] < tree.threshold:
126             return predict_tree(tree.left, x)
127         else:
128             return predict_tree(tree.right, x)
129
130     # Construcao da arvore
```

```
131     tree = build_tree_bio(X, y, max_depth)
132 
133     # Funcao de predicao
134     def predict(X_test):
135         return np.array([predict_tree(tree, x) for x in X_test])
136 
137     return tree, predict
```