Jader Duque Figueredo

Modelos de Aprendizado de Máquina para Árvore de Decisão Interpretável: Otimização vs Heurística

Brasil

Dezembro de 2024

Jader Duque Figueredo

Modelos de Aprendizado de Máquina para Árvore de Decisão Interpretável: Otimização vs Heurística

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Universidade Federal de Itajubá – UNIFEI

Programa de Pós-Graduação em Ciência e Tecnologia Computação

Orientador: Prof. Dr. Carlos Henrique da Silveira

Coorientador: Prof. Dr. Pedro Henrique Del Bianco Hokama

Brasil

Dezembro de 2024

Dedico este trabalho aos meus pais, em nome de todas as pessoas que trabalham insistentemente para transformar as suas vidas e das pessoas que amam.

Agradecimentos

Este não é o fim de um caminho, mas é a conclusão de uma etapa desafiadora, motivo de realização pessoal e felicidade compartilhada. Agradeço aos meus pais Itamar e Leila por tudo que fizeram por mim e por tudo que me ensinaram, pelo amor infinito e pela dedicação incansável.

Agradeço a todos meus familiares, desde avós (em memória), tios, primos, padrinhos e madrinhas e todos que compartilhamos momentos juntos, pelo carinho e apoio; Às minhas irmãs pelo carinho, amor e parceria. Agradeço de coração aos meus tios Efraim e Lúcia pelo incentivo e apoio contínuo na minha formação e por acreditarem em mim. Todos estes, saibam que eu lembro e agradeço por cada palavra de incetivo, cada ajuda financeira, cada carona, cada estadia, cada sugestão de oportunidade, cada favor e cada palavra de esperança.

Agradeço a todos os meus professores, meu orientador, Prof. Dr. Carlos Henrique da Silveira, pela confiança, apoio e pelos ensinamentos passados; Ao Prof. Dr. Carlos Henrique Valério de Moraes pelas importantes contribuições a este trabalho, pelo apoio e pela amizade. Me faltam palavras para agradecer meu coorientador Prof. Dr. Pedro Henrique Del Bianco Hokama pelo imenso apoio, por todo aprendizado, pela calma, paciência, parceria, amizade e principalmente por persistir e acreditar em mim. Serei eternamente grato!

Agradeço também aos meus amigos Diego, Carlos, Sávio, André, Afrânio, Maico e outros pela amizade, incentivo e por todos os bons momentos compartilhados. Agradeço à Liziane pela parceria, companhia e compreensão. Sua presença melhora meus dias. Obrigado também pelo apoio, incentivo, pelas contribuições e conhecimentos compartilhados.

Agradeço à Universidade Federal de Itajubá - UNIFEI e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo Ensino Público do Brasil, assim como todos que trabalham diariamente para mantê-lo. Enfim, à todos que apoiaram, incentivaram, contribuíram, comemoraram e estiveram presentes na minha vida ou durante essa jornada até aqui, muito obrigado!

"É o desconforto que provoca a mudança."
(Leandro Karnal)

"... A cada passo há 360º de caminhos possíveis e cada caminho escolhido pressupõe uma infinidade de vidas preteridas [...]. A cada segundo a escolha será de uma vida concretamente vivida em detrimento de tantas outras ..."
(Clóvis de Barros Filho)

Resumo

Cada dia mais a inteligência artificial é encontrada em diferentes atividades do cotidiano, gerando ferramentas e soluções cada vez mais robustas, melhorando resultados e potencializando habilidades humanas. Os modelos de aprendizado supervisionado mais complexos, chamados de "caixa preta", tais como Redes Neurais, são poderosos, mas deixam a desejar em interpretabilidade para soluções que tratam de dados sensíveis em contextos como finanças, saúde, jurídico ou mesmo acadêmico. Neste sentido, modelos de "caixa branca", como Árvores de Decisão se mostram soluções robustas e mais adequadas devido ao seu alto grau de interpretabilidade. Além de modelos de aprendizado de máquina já consolidados, como Árvore de Classificação e Regressão - CART, estudos recentes também trouxeram novos modelos como Árvore de Classificação Ótima usando Classificação Inteira Mista - OCT-MIO, capaz de se ajustar ainda mais aos dados de treinamento e alcançar maior acurácia em alguns casos. Este trabalho traz a modelagem, implementação e comparação destes dois modelos, tanto em treinamento, quanto em teste usando validação cruzada (*K-Fold)*, além de uma análise da interpretabilidade das árvores e da utilização do OCT-MIO como heurística. Os experimentos utilizam dados reais e sensíveis como para diagnóstico de nível de estresse, predição para aprovação de crédito e predição de sucesso acadêmico. Apesar do CART ser um bom modelo de classificação, foi possível observar que o modelo OCT-MIO é uma alternativa capaz de obter resultados próximos, iguais ou ainda melhores, especialmente para árvores de classificação de menor altura, ideais em cenários onde a interpretabilidade é necessária. Desta forma, o modelo OCT-MIO é capaz de classificar dados mais corretamente que o CART em árvores de altura mínima o suficiente para classificar todas as classes de um problema, sem abrir mão da interpretabilidade.

Abstract

Artificial intelligence is becoming increasingly integrated into various daily activities, producing more robust tools and solutions, improving results, and enhancing human capabilities. The more complex supervised learning models, known as "black boxes" such as Neural Networks, are powerful but fall short in interpretability for solutions dealing with sensitive data in contexts like finance, healthcare, legal, or academia. In this regard, "white box" models, such as Decision Trees, prove to be robust and more suitable solutions due to their high level of interpretability. In addition to well-established machine learning models, such as Classification and Regression Trees (CART), recent studies have introduced new models like Optimal Classification Tree using Mixed-Integer Optimization (OCT-MIO), which is capable of fitting the training data even better and achieving higher accuracy in some cases. This work presents the modeling, implementation, and comparison of these two models, both in training and testing using cross-validation (K-Fold). It also includes a interpretability analysis of the obtained classification trees and the use of OCT-MIO as a heuristic. The experiments utilize real and sensitive data, such as for stress level diagnosis, credit approval prediction, and academic success prediction. Although CART is a good classification model, it was observed that the OCT-MIO model is a viable alternative capable of achieving results that are comparable, equal, or even better, especially for classification trees with a smaller height, which are ideal in scenarios where interpretability is required. Thus, the OCT-MIO model can classify data more accurately than CART in trees with a minimal height sufficient to classify all classes of a problem, while maintaining interpretability.

Keywords: optimal decision tree. optimal classification tree. mixed integer programing. mixed integer optimization. machine learning. explicability. interpretability. auditability. odt. oct-mio. cart. k-fold cross validation. artificial intelligence. IA. XAI.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

elemento	é cada observação em um conjunto de dados que corresponde a uma linha na tabela de dados.
$\mathbf{X}$	é o vetor de todos os elementos de um problema.
X	é o conjunto de todos os elementos de um problema.
$\mathbf{y}$	é o vetor de todas as classes do elementos de $\mathbf{X}$.
i	é o índice de cada elemento do vetor de dados e é utilizado para iterar através dos elementos.
P	é o número de atributos dos elementos de um conjunto de dados.
p	é cada atributo do conjunto $\{1,2,\ldots,P\}$. p é usado para iterar através do conjunto de atributos.
$\boldsymbol{x}_i$	é um elemento $\mathbf{x}$ de índice i.
K	é o número de classes do problema.
k	é cada classe do conjunto de classes do problema, sendo $k \in \{1,\ldots,K\}$.
k^*	é a classe escolhida para rotular um nó folha.
j	é o índice de cada classe, é usado para iterar sobre o conjunto de classes do problema $\{1,k2,\ldots,K\}$.
y_i	é a classe de um elemento $\mathbf{x}_i$, onde $y_i \in \{1,2,\ldots,K\}$.
V	é o conjunto de vértices (nós) da uma árvore.
V_f	é o conjunto de vértices folha.
V_b	é o conjunto de vértices galho.

v	é um vértice (nó) qualquer da árvore.
v'	é um dado nó pai em um subconjunto de nós da árvore
v''	é um dado nó filho em um subconjunto de nós da árvore
v_L	é o nó filho esquerdo de um dado nó da árvore.
v_R	é nó filho direito de um dado nó da árvore.
b_v	é o limiar b de divisão do nó v para todo $v \in X_b$.
$\mathbf{X}_L$	é o subconjunto de $\mathbf{X}$ que vai para o nó filho esquerdo.
$\mathbf{X}_R$	é o subconjunto de $\mathbf{X}$ que vai para o nó filho direito.
s	é um valor numérico ou categórico que compõe uma condição que divide em um nó galho.
S	é o conjunto de todos os valores limiares s.
$i(v)$	é uma função que dá a impureza de um nó v.
$p_{v''}$	é a proporção relativa de elementos em um nó filho v'' em relação ao número de elementos em seu nó pai v' antes de aplicar uma divisão.
$p(v)$	é a uma função que dá a proporção absoluta de elementos em um nó v em relação a todo conjunto de dados de treinamento.
$I(v)$	é uma função que dá a impureza absoluta de um nó v.
$\Delta i(s,v')$	é uma função que dá a variação da impureza relativa através da diferença da impureza relativa de um nó pai $i(v')$ para seus nós filhos v_L e v_R.
$\Delta I(s,v')$	é uma função que dá a variação da impureza absoluta através da diferença da impureza relativa de um nó pai $i(v')$ para seus nós filhos v_L e v_R.

s^*	é a condição $s \in S$ aplicada em um nó v' que gera a máxima variação de impureza absoluta.
$p(k^*\|v)$	é uma função que dá a classe de maior proporção em um nó v.
T	é uma árvore de classificação, dada por um arranjo específico de nós $v \in V$.
T'	é uma nova árvore de classificação, dada por um arranjo específico de nós $v \in V$ após uma nova ramificação de nó.
N	é a quantidade de nós de uma árvore.
N_{V_f}	é a quantidade de nós folha de uma árvore.
N_{V_b}	é a quantidade de nós galhos de uma árvore.
n_{min}	é o número mínimo de elementos que deve haver em um nó folha.
$R_{xy}(T)$	é o número de classificações incorretas (perdas) em uma árvore T.
D	é a altura máxima para crescimento da árvore.
N_{max}	é o número máximo de nós possíveis em uma árvore.
α	é uma constante fornecida por parâmetro ao OCT-MIO que pondera o equilíbrio entre acurácia e número de nós da árvore.
$A(v)$	é o conjunto de todos os nós ancestrais tangíveis pelo caminho feito da raiz da árvore, v_1, até v.
$A_L(v)$	é o conjunto de todos os nós ancestrais tangíveis pelo caminho feito da raiz da árvore, v_1, até v, a partir dos quis segue-se pela aresta esquerda.
$A_R(v)$	é o conjunto de todos os nós ancestrais tangíveis pelo caminho feito da raiz da árvore, v_1, até v, a partir dos quis segue-se pela aresta direita.

$\boldsymbol{d}$	é um vetor de binário $d = \{1,0\}$ de tamanho N que determina se um nó v faz ou não divisão.
$\boldsymbol{a}$	é um vetor cujo o índice que contém 1 determina qual atributo será usado em uma determinada divisão
$\mathbf{Z}$	é uma matriz de valores binários que mapeia onde cada elemento x em um nó v.
z_i	refere-se a um elemento $\boldsymbol{x}_i$ mapeado pelas linhas em Z.
z_v	refere-se a um nó v mapeado pelas colunas em Z.
M	é uma constante de valor maior que 1 utilizada para afrouxar restrições da árvore de classificação.
ϵ	é uma constante de valor extremamente pequeno, utilizado para possibilitar transformar comparações $<$ em $\leq$.
Y	é uma matriz $i \times k$ que define que cada elemento $\boldsymbol{x}_i$ possui uma classe k.
n_{kv}	é o número de elementos de uma classe k em um nó v.
c_{kv}	é uma matriz de valores binários que mapeia a qual classe k o nó v é rotulado.
c_v	é a classe k que rotula o nó v.
$R(v)$	é o número de classificações incorretas em um nó v, dada pela diferença entre número total de elementos em v e o número de elementos da classe de maior proporção em v.
$\hat{R}$	número de classificações incorretas normalizadas sobre a linha de base da acurácia.
I.A.	Inteligência Artificial.
CART	*Classification and Regression Tree* - Árvore de Classificação e Regressão.

OCT	*Optimal Classification Tree* - Árvore de Classificação e Regressão.
MIO	*Mixed-Integer Optimization* - Otimização Inteira Mista.
OCT-MIO	*Optimal Classification Treee using Mixed-Integer Optimization* - Árvore de Classificação Ótima usando Otimização Inteira Mista.
ODT	*Optimal Decision Tree* - Árvore de Decisão Ótima.
XAI	*Explainable Artificial Intelligence* - Inteligência Artificial Explicável

Sumário

1 Introdução

1.1 Motivação

No cenário atual, onde a inteligência artificial permeia cada vez mais o cotidiano das pessoas, diversas atividades humanas têm tido suporte ou mesmo têm sido substituídas por ferramentas e recursos automatizados. Entretanto, o aprimoramento vertiginoso destes modelos traz consigo o aumento da complexidade dos mesmos, tornando cada vez mais difícil entender a relação entre os dados de entrada e as saídas dadas por esses modelos, bem como a lógica por trás de cada decisão tomada por eles (COSTA; PEDREIRA, 2023; LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020), dificultando sua utilização em atividades em áreas sensíveis ou de risco como saúde, finanças, trânsito, jurídica, governamental ou mesmo acadêmica (BERTSIMAS; DUNN, 2017).

Neste contexto, as árvores de decisão são um modelo de aprendizado de máquina supervisionado que tem ganhado destaque não somente pela sua robustez e aplicabilidade, mas também por sua característica interpretável, isto é, a facilidade e intuitividade que ele proporciona a seres humanos para compreender quais decisões são tomadas pelo modelo. (COSTA; PEDREIRA, 2023; BERTSIMAS; DUNN, 2017).

Muitos estudos ainda estão surgindo sobre Árvores de Decisão. Eles variam em relação à forma de divisão, à forma de predição, a conjuntos de dados com características específicas, métodos utilizados, entre outros (COSTA; PEDREIRA, 2023). Dentre eles destacam-se:

- Árvore Ótimas (*Optimal Trees*) é um dos tópicos mais pesquisados atualmente em Árvores de Decisão, especialmente porque contornam sua principal limitação: as divisões sub-ótimas. Por outro lado, seu principal desafio é a escalabilidade, o qual não é uma preocupação para árvores tradicionais. (COSTA; PEDREIRA, 2023).

- Árvores Baseadas em Gradiente (*Gradient-Based Trees*) têm proporcionado resultados promissores, especialmente para interpretabilidade de modelos "caixa-pret" como redes neurais, não para superá-las, mas como modelos baseados em gradiente, que continuam preservando sua interpretabilidade, já que a maioria dos métodos baseados em gradiente não são tradicionalmente interpretáveis (COSTA; PEDREIRA, 2023).

Há uma relação inversa entre acurácia e interpretabilidade, não é possível afirmar que essa relação seja inevitável. Existem estudos apontando que é possível melhorar a acurácia sem afetar comprometer a interpretabilidade frequentemente elogiada dos modelos de Árvore de Decisão. Nesse sentido, há estudos que exploram tanto novos critérios de divisão, quanto novas formas de interpretabilidade das árvores. Estudos recentes também exploram ferramentas de visualização, técnicas de rotulagem *post hoc* e representação para divisões multivariadas (COSTA; PEDREIRA, 2023).

Por fim, a literatura ainda carece de pesquisas em larga escala, divulgação de implementações e abordagens variadas, por exemplo, que compare tanto mais modelos de Árvore de Decisão, quanto suas aplicações a mais conjuntos de dados. (COSTA; PEDREIRA, 2023).

Árvores de decisão tem sido estudadas há mais de 60 anos e, desde então, elas continuam se aprimorando continuamente. Ainda assim, existem lacunas que têm sido exploradas nas últimas décadas. Recentemente, com o crescimento da I.A., as árvores de decisão voltaram a receber bastante atenção especialmente por sua interpretabilidade, a qual é uma de suas principais vantagens sobre modelos de aprendizado de máquina mais complexos (COSTA; PEDREIRA, 2023).

1.2 Objetivos

A maioria dos modelos de árvore de classificação, como o CART, utilizam uma estratégia heurística, que é um dos motivos pelo qual, por vezes, esses modelos não são tão acurados quanto outros. Contudo, com o surgimento de modelos de árvores de classificação ótimas que utilizam estratégias exatas ao invés de heurísticas,

estas podem alcançar desempenho próximo ou ainda superior, também sem abrir mão da interpretabilidade (BERTSIMAS; DUNN, 2017).

Dentre estes estudos, a Árvore de Classificação Ótima usando Otimização Inteira Mista (*Optimal Classification Trees using Mixed-Integer Optimization* - OCT-MIO) de Bertsimas e Dunn (2017) tem se mostrado muito promissora, principalmente por sua capacidade de produzir árvores de decisão ótimas pequenas, geradas a partir apenas de decisões estritamente binárias, tornando-as soluções ainda mais interpretáveis.

Diferente de uma árvore de classificação gananciosa, uma árvore de classificação ótima objetiva encontrar um conjunto de divisões que gerem o maior número de classificações corretas. A principal diferença é que estas combinações não são de forma hierarquicamente encadeada, do tipo topo-base (*top-down*), impedindo que a árvore fique presa a uma solução ótima local.

Este trabalho propõe-se a estudar tanto o modelo CART, quanto o OCT-MIO, entender seus mecanismos, aplicá-los a conjuntos de dados reais, medir seus desempenhos e analisar empiricamente suas interpretabilidades baseada no tamanho das suas árvores geradas, a fim de verificar se o OCT-MIO pode ser uma solução alternativa que seja tão interpretável quanto o CART e ainda mais precisa para conjuntos de dados reais. Para isso, os seguintes objetivos específicos precisam ser respondidos:

1.2.1 Objetivos específicos

Objetivo específico 1: Implementar os modelos de árvore de classificação CART e OCT-MIO.

Objetivo específico 2: Analisar o desempenho de ambos os modelos nas principais métricas utilizadas para problemas de classificação.

Objetivo específico 3: Analisar o desempenho de ambos os modelos sobre tempo gasto na etapa de treinamento.

Objetivo específico 4: Analisar a capacidade de generalização de ambos os modelos.

Objetivo específico 5: Analisar empiricamente a interpretabilidade de ambos os modelos baseado no tamanho da árvore gerada por cada um após a etapa de treinamento.

1.3 Justificativa

Para entender melhor diversos aspectos das árvores de classificação enquanto modelos de aprendizado de máquina interpretáveis, é necessário entendimento profundo de seus mecanismos, potenciais, limitações, aplicabilidade, desempenho e, claro, sua interpretabilidade.

Dado o contexto atual de crescente introdução de inteligência artificial em sistemas e recursos utilizados nas mais diversas atividades humanas, como também uma crescente demanda e necessidade de modelos interpretáveis para garantir a confiança e aceitação do uso de I.A. em áreas sensíveis, este trabalho busca explorar diversos aspectos de modelos de árvore de decisão interpretáveis através de experimentos comparando dois modelos de referência de árvores de decisão: o CART, que utiliza heurística, e OCT-MIO, que utiliza otimização.

A análise de ambos os modelos em diversos aspectos como praticidade, eficiência, generalização e interpretabilidade, busca fornecer contribuições relevantes sobre eles em contextos reais, onde os efeitos podem ser críticos, sensíveis ou precisem ser auditáveis, através de experimentos práticos e teoricamente embasados, contribuindo também para a ampliação do conhecimento nesta área.

1.4 Objeto

O presente trabalho trata-se de uma análise comparativa entre dois modelos de árvores de decisão: CART e OCT-MIO. Esta análise busca avaliar a eficiência, adequação e interpretabilidade de ambos os modelos em problemas de classificação usando dados reais sensíveis, através das seguintes etapas e aspectos:

- Modelagem Matemática: Detalhar as modelagens matemáticas de ambos os modelos com o objetivo de estudá-los e entendê-los completamente;

- Implementação: Implementar ambos os modelos com o objetivo de utilizá-los na classificação de diferentes conjuntos de dados reais sensíveis;
- Desempenho: Medir o desempenho de ambos os modelos tanto na etapa de treinamento quanto teste, sobre diferentes porções dos conjuntos de dados e diferentes alturas de árvore, utilizando diferentes métricas;
- Análise: Analisar e comparar métricas de acurácia, F1-Score e também tempo de treinamento dos dois modelos diferentes variações de experimentos;
- Capacidade de Generalização: Analisar e identificar tendências dos modelos na capacidade de predição de conjuntos de mesmo domínio, mas sobre os quais o modelo não foi treinado;
- Interpretabilidade: Gerar e analisar diagramas de diferentes árvores resultantes de ambos os modelos a fim de compará-las, analisar suas interpretabilidades e inferir tendências, vantagens, desvantagens e eventuais ajustes;

Espera-se fornecer conclusões construtivas acerca do estudo e aplicação de árvores de classificação interpretáveis para conjuntos de dados sensíveis. Os estudos e experimentos deste trabalho trarão uma visão bastante detalhada sobre as estratégias de cada modelo, suas vantagens e limitações além de perspectivas de aplicações práticas.

1.5 Estrutura do Trabalho

O Capítulo 1 - Introdução delimita áreas gerais, específicas e traça os objetivos deste trabalho. A seguir, o Capítulo 2 - Fundamentação Teórica apresenta uma visão geral dos conceitos, métodos e técnicas que serão abordadas; O Capítulo 3 - Metodologia detalha de forma extensiva as bases conceituais de métodos e técnicas utilizados ao longo do trabalho; O Capítulo 4 - Experimentos e Resultados primeiro apresenta os resultados obtidos nos experimentos e em seguida faz análises e comparações sobre eles, onde também apresenta e analisa diferentes diagramas de árvores de decisão. Por fim, discute questões gerais e limitações dos experimentos;

O Capítulo 5 - Conclusão fornece as conclusões obtidas a partir do experimentos e o Capítulo 6 - Trabalhos Futuros traça uma direção da continuidade deste trabalho.

2 Fundamentação Teórica

2.1 Aprendizado

Aprendizado é o processo de aquisição de conhecimentos declarativos, desenvolvimento motor e habilidades cognitivas através da observação, da instrução ou da prática. Desde a concepção dos computadores já havia um esforço em se implantar a capacidade de aprendizado neles. Ao estudo e modelagem do processo de aprendizado a computadores dá-se o nome de **Aprendizado de Máquina**, do inglês, *Machine Learning* (CARBONELL; MICHALSKI; MITCHELL, 1983).

2.2 Classificação

Classificação é uma das tarefas de tomada de decisão mais comuns da atividade humana e é caracterizada pela necessidade de atribuir a um objeto, um rótulo, dentre um grupo definido de rótulos, baseando-se na análise de um número de atributos deste objeto (BERTSIMAS; DUNN, 2017). A tarefa de classificação automatizada pode ser aplicada na predição de mercado de ações, previsão do clima, predição de falência, diagnostico médico, reconhecimento de fala, entre outros (SATHYA; ABRAHAM et al., 2013).

2.3 Aprendizado de Máquina: Supervisionado vs Não-supervisionado

Aprendizado supervisionado é o aprendizado feito a partir de uma fonte de dados que contém um rótulo (classe, categoria) já atribuído a cada elemento ou indivíduo de um conjunto, permitindo a validação e ajustes da resposta dada pelo algoritmo de aprendizado. Na outra mão, **aprendizado não-supervisionado** é o aprendizado feito a partir de uma fonte de dados que não possui rótulos atribuídos a cada elemento do conjunto. Consequentemente, este método não conta com um

indicador que permite validar objetivamente os padrões encontrados nos dados. Todavia, esta ausência de validação também é útil, pois permite que o algoritmo encontre outros padrões ou organize os dados sem considerações anteriores, possibilitando soluções potencialmente diferentes de soluções eventualmente esperadas (SATHYA; ABRAHAM et al., 2013).

2.4 Aprendizado Supervisionado

O **aprendizado supervisionado** acontece em 2 etapas principais: treinamento e teste, nesta ordem. No treinamento, um conjunto de dados é utilizado para ajustar o modelo. Este conjunto de dados é composto por uma matriz de atributos (características) e um vetor de classes (rótulos), derivados dos elementos do conjunto. Ambos são submetidos como entrada do modelo de aprendizado, que utiliza técnicas matemáticas e estatísticas para analisar os atributos e suas respectivas classes associadas, com o objetivo de ajustar os seus parâmetros internos, como pesos e coeficientes, de forma a minimizar uma função de erros ou maximizar uma métrica de desempenho específica. Os parâmetros internos são ajustados com base em algoritmos que implementam expressões matemáticas, onde os parâmetros internos são incógnitas dessas expressões. O próprio modelo treinado é a saída resultante da etapa de treinamento e que será utilizada na etapa de teste. (NASTESKI, 2017).

Na etapa de teste, um novo conjunto, diferente do conjunto de treinamento, é submetido como entrada do modelo já treinado. O conjunto de teste é composto apenas por uma matriz, onde cada linha é um vetor de atributos do elemento correspondente ao índice da linha. Baseado apenas nestes atributos, o modelo, agora já ajustado, irá predizer uma classe para cada vetor de atributos. O modelo novamente irá aplicar técnicas matemáticas e estatísticas sobre os atributos com o objetivo de estimar uma classe associada para cada um deles. O objetivo é estimar (predizer) corretamente a classe de cada elemento. Por fim, a saída do modelo é um vetor de classes, onde cada linha contém um rótulo de classe para cada vetor de atributos (elemento) de entrada de índice correspondente. Este vetor de classes preditas, quando comparado ao vetor de classes originais dos atributos (o qual foi

omitido como entrada do modelo), fornece métricas importantes sobre o modelo de aprendizado.

2.5 Árvores de Classificação

De acordo com Bertsimas e Dunn (2017), o problema de árvore de classificação começa ser descrito a partir de um **conjunto de dados** $(\mathbf{X}, \mathbf{y})$ de n elementos $(\mathbf{x}_i, y_i)$ onde $i = \{1, \ldots, n\}$, sendo cada um deles composto por P atributos representados no vetor $\mathbf{x}_i \in \mathbb{R}^P$ e por uma das K possíveis classes $y_i \in \{1, \ldots, K\}$ onde $i = \{1, \ldots, n\}$. Não é estritamente necessário que os dados estejam normalizados no intervalo de 0 até 1 para a implementação de uma árvore de classificação, ou seja, $\mathbf{x}_i \in [0, 1]^P$, mas isso simplifica a implementação, como feito por Bertsimas e Dunn (2017), melhora a padronização e visualização gráfica do problema. Além disso é esperado que alguns atributos sejam categóricos, porém é possível substituir tais dados por valores numéricos, tal como assumido neste trabalho.

Definição 2.1. *Um classificador é uma função* $c : [0, 1]^P \to \{1, 2, \ldots, K\}$ *que mapeia qualquer elemento* $\mathbf{x} \in [0, 1]^P$ *a uma das* K *possíveis classes* 1, 2, ..., K *(BERTSIMAS; DUNN, 2017).*

Uma **árvore de classificação**, do inglês *classification tree*, ou ainda "classificadores estruturados em árvores binárias" (BREIMAN et al., 2017), é uma estrutura de dados do tipo árvore estritamente binária, ou seja, possui 0 ou 2 filhos, composta por um conjunto V de nós, que são unidades da árvore ligados por arestas, pelas quais é possível acessá-los, e está organizada de forma que seus nós possuam uma relação hierárquica, onde qualquer aresta sempre parte de um nó ascendente pai v e chega em um nó descendente esquerdo v_L ou direito v_R.

Um **nó** ou **vértice** v é uma unidade da árvore, geralmente composto por informações características e zero ou duas ligações para seus filhos. Há dois tipos de nós em uma árvore, os nós galho (*branch*) V_b e os nós folha (*leaf*) V_f, tal que $V_b \cap V_f = \emptyset$ e $V_b \cup V_f = V$:

- **Nós galho** ou **internos** são aqueles que possuem filhos, ou seja, que há uma aresta partindo dele e chegando diretamente até cada um de seus nós filhos.

Cada nó galho da árvore de classificação representa uma pergunta que se faz sobre os dados e cuja resposta é booleana, ou seja, a resposta pode ser apenas sim (verdadeiro) ou não (falso), particionando, assim, estes dados em dois subconjuntos disjuntos, onde uma parte corresponde aos dados cuja resposta à pergunta é "sim" e outra parte correspondente aos dados cuja resposta à pergunta é "não". A pergunta aplicada à cada nó galho v segue o formato $\boldsymbol{a}_v\boldsymbol{x}_i < b_v$, melhor definido posteriormente.

- **Nós folha**, também chamados de **terminais** ou **externos**, são nós v da árvore que não possuem nós filhos, ou seja, que não há arestas começando nele e terminando em outro. Um nó folha em uma árvore de classificação significa uma conclusão sobre o subconjunto dos dados contida nele.

Os dados do problema são movidos ao longo da árvore, começando pelo nó raiz v_1 (definido posteriormente) no qual eles são particionados em dois conjuntos, e cada um é direcionado a um dos seus nós filhos; O processo se repete recursivamente até alcançar um nó folha. O objetivo em um problema de classificação é atribuir corretamente rótulos de classes à cada nó folha da árvore resultante, para isso o maior desafio não está em apenas determinar os rótulos de classe dos nós folha, mas sim em encontrar quantas e quais divisões fazer para que a árvore resultante forneça a melhor predição de classes, uma vez que todos os dados contidos em um nó folha receberão o mesmo rótulo de classe predita. Portanto, uma **boa árvore** de classificação é aquela cujas divisões aplicadas sobre os dados geram nós folhas cujos dados contidos neles sejam, em maior número possível, de uma única classe, isto é, com o menor número de predições incorretas e, ao mesmo tempo, com o menor número de nós e menor altura possível.

Para fins práticos, no contexto deste trabalho, uma árvore de classificação também poderá ser chamada simplesmente de "árvore", nós galho de "galho" e nós folha de "folha". Para aprofundar na definição de árvore de classificação, é importante definir alguns de seus componentes e conceitos.

O primeiro nó da árvore, é chamado de **nó raiz** (v_1) ou apenas **raiz**. Este é o único nó da árvore que não possui arestas chegando até ele. O nó raiz pode ser um nó galho, se ele tiver nós filhos, ou nó folha, no caso da árvore ter apenas um nó.

Uma **árvore vazia** é um conjunto que não possui nenhum nó e, consequentemente, nenhuma aresta, isto é, não há sequer o nó raiz, portanto uma árvore vazia é um conjunto vazio de nós e não pode ser uma árvore de classificação.

O **limiar** b_v é um ponto divisório em uma escala. Em árvores de classificação, o limiar é um valor que divide um conjunto de dados. Um nó galho, na árvore de classificação, possui um limiar associado a ele que permite impor uma **condição** aos dados, neste caso, na forma de uma divisão do tipo $\boldsymbol{a}_v^\intercal \boldsymbol{x}_i < b_v$ que define se cada indivíduo i do conjunto de dados seguirá para a esquerda ou para a direita. Neste trabalho, $\boldsymbol{a}_v$ é um vetor coluna, com P valores, composto de zeros, exceto em uma posição p, ou seja, $a_{pv} = 1$. Isso equivale a dizer que, em cada nó galho, o conjunto de dados é dividido baseado apenas em 1 atributo.

Dito isto, a divisão associada à condição (pergunta) envia todo $\boldsymbol{x}$ que responde "sim" à pergunta para v_L e todo $\boldsymbol{x}$ que responde "não" para v_R. Fica convencionado que uma condição envolve um limiar, um critério de comparação e um atributo específico dos elementos do conjunto de dados. Então é feita a seguinte comparação: Se o valor de um dado atributo do elemento é menor que o limiar associado a um nó galho, o elemento segue para o nó filho esquerdo, mas se o valor do atributo é maior ou igual ao limiar, o elemento segue para a direita.

Diferentemente do nó galho, um nó folha não particiona os dados, ou seja, todos os dados que chegam até ele, permanecem nele. Um nó folha é definido como um nó que não faz divisões dos dados.

Segundo Breiman et al. (2017), a construção de uma árvore de classificação envolve sempre três fatores:

1. A seleção das divisões;

2. A decisão de quando declarar um nó terminal ou continuar dividindo o nó;

3. A associação de cada nó terminal a uma classe.

A Figura 1 ilustra o conceito de árvore de classificação composta de nós galho e nós folha, onde S = Sim e N = Não.

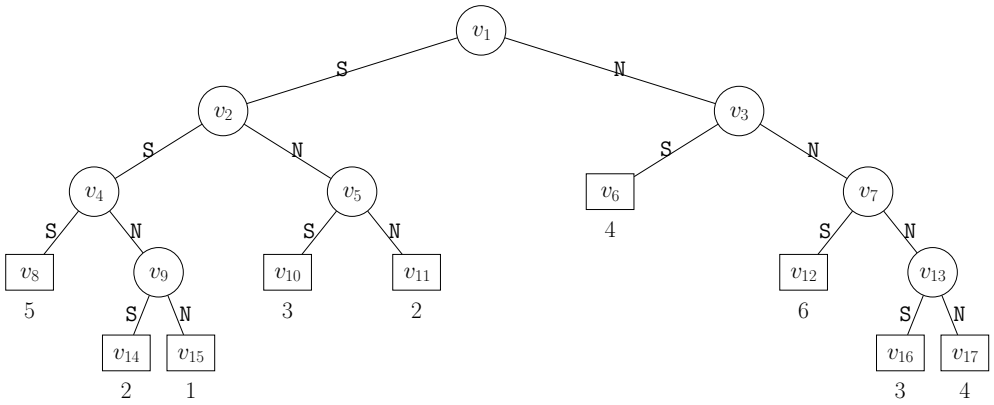

Figura 1 – Exemplo de árvore binária com 6 classes

2.6 Estudos Recentes em Árvore de Decisão

Nos últimos anos, novos estudos trouxeram aprimoramentos sobre árvores de decisão, sobretudo desde as árvores de decisão ótimas.

As Árvores de Classificação Ótimas Fortes (*Strong Optimal Classification Trees*) buscam aprimorar as árvores ótimas através de relaxação linear mais forte, udo de método de decomposição de benders, aumento de performance computacional e acurácia e também gerando árvores mais justas e interpretáveis. Essas melhorias juntas proporcionam uma aplicação mais poderosa e prática das árvores de classificação (AGHAEI; GÓMEZ; VAYANOS, 2024).

Já as Árvores de Classificação Ótima usando Programação Linear Binária (BinOCT) simplifica o processo de otimização reduzindo a dependência aos dados de treinamento, aumentando significativamente o tempo de treinamento, usando limiares binários mais eficientes e menor número de restrições. Isso confere ao modelo melhor escalabilidade e resultados melhores que as Árvores de Classificação Ótimas usando Otimização Inteira Mista tanto em performance quanto em tempo de computação. (VERWER; ZHANG, 2019)

Linden, Weerdt e Demirović (2023) propuseram uma Árvore de Decisão Ótima usando Programação Dinâmica através da decomposição recursiva do pro-

blema, Componentes de Framework, princípio de otimalidade, preservação da ordem e anti-monotonicidade e otimização generalizada. Sua árvore pode lidar com uma grande variedade de restrições e objetivos além de superar linhas de base de eficiência e escalabilidade de abordagens baseadas em Otimização Inteira Mista.

2.7 Inteligência Artificial Explicável - XAI

Inteligência Artificial Explicável ou ***Explainable Artificial Intelligence (XAI)***, em inglês, é o campo de estudo de métodos de interpretabilidade e explicabilidade de modelos de de inteligência artificial, em especial, modelos de aprendizagem de máquina, o qual tem crescido aceleradamente desde meados do ano de 2016 em resposta ao aprimoramento e crescimento dos modelos de aprendizado de máquina, especialmente os modelos de aprendizado profundo (*Deep Learning*). Em geral, à medida que os modelos de aprendizado se tornaram mais acurados, eles também se tornaram mais complexos, sendo cada vez mais difícil entender seus mecanismos internos (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020).

O aprimoramento dos modelos de aprendizado de máquina tem trazido muitos avanços na ciência e tecnologia, entretanto há cenários onde a aplicação destes modelos pode levantar questões sobre sua confiabilidade, imparcialidade, transparência e auditabilidade, principalmente em setores como saúde, financeiro, jurídico, automação e acadêmico, por exemplo. Por esse motivo, estudos sobre interpretabilidade e explicabilidade dos modelos também têm crescido com o objetivo de atender estas necessidades (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020).

Interpretabilidade e Explicabilidade são conceitos amplamente utilizados na literatura, sendo possível encontrar tanto trabalhos que tratam-os como equivalentes, como trabalhos que fazem uma distinção explícita entre eles (MOHSENI; ZAREI; RAGAN, 2021). **Interpretabilidade** descreve a capacidade humana na compreensão da relação de causa e efeito em um modelo de aprendizado de máquina, onde as causas são as observações de entrada, seus atributos e relações, e efeitos seriam os resultados e predições do modelo. Já **Explicabilidade** descreve a

capacidade humana na compreensão da lógica e mecanismo interno de um modelo de aprendizado de máquina, isto é, na capacidade de compreender como um modelo é treinado e toma suas decisões (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020).

A partir destas definições, interpretabilidade é um termo mais amplo que explicabilidade, pois a interpretabilidade de um modelo não requer o entendimento do seu mecanismo interno, apenas da relação de causa e efeito nas suas entradas e saídas, respectivamente. Já explicabilidade requer um certo grau de entendimento do funcionamento interno de um modelo, o que, por consequência, fornece entendimento das relações de causa e efeito sobre suas entradas e saídas (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020).

Os modelos de aprendizagem de máquina podem ser divididos em 2 grupos. Os modelos categorizados como "caixa preta" (black-box) são modelos mais complexos, cujo o entendimento do seu funcionamento não é claro ou fácil de ser entendimento por um ser humano, tais como modelos de aprendizagem profunda ou modelos de conjuntos (*ensembles*). Já os modelos categorizados como "caixa branca" (*white-box*) ou "caixa de vidro" (*glass-box*) produzem resultados facilmente explicáveis, como é o caso de modelos baseados em vetores ou em árvores de decisão (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020).

Quanto mais transparente e compreensível os mecanismos de um modelo de aprendizado de máquina, mais confiabilidade e segurança ele proporciona às pessoas que os utilizam, viabilizando sua aplicação em áreas como saúde, finanças, agricultura, jurídica, governamentais etc. (DAS et al., 2020).

Segundo Das et al. (2020), existem alguns requisitos para se ter um modelo de aprendizado de máquina interpretável:

Grau de modelagem de caixa-branca: sugere que quanto maior o grau de "caixa branca" do modelo, mais o modelo se torna compreensível a humanos e mais ele tende a ser utilizado em tomadas de decisão de alto risco e em contexto onde a justiça e segurança são requisitos importantes.

Visualização de Dados: é um requisito de modelos de aprendizado de máquina interpretáveis que permite que pessoas consigam analisar o modelo e tirar

conclusões sobre os dados do problema de maneira visual. Permite também que modelos sejam auditados ou mesmo que erros sejam identificados mais facilmente.

Importância de Variáveis: é a medida do quanto cada variável de entrada contribui na predição feita pelo modelo. Quanto mais importante uma variável, mais ela é utilizada para explicar o modelo.

Justiça: refere-se à análise de desigualdades nas predições causadas por viés ou discriminação racial, étnica, etária, de gênero, entre outros. Um modelo que incorpora justiça permite verificar se as predições ou erros estão distribuídos de forma justa entre diferentes grupos demográficos, garantindo que as previsões não causem impactos desiguais ou prejuízos.

Sensibilidade: é a capacidade do modelo de ser estável, isto é, de fazer predições consistentes, mesmo quando os dados de entrada são modificados sutilmente.

Residualidade: é a capacidade de analisar como erros ou diferenças entre os valores previstos por um modelo e os valores reais estão distribuídos. O modelo que permite analisar suas residualidades, permite determinar se o padrão de distribuição dos resíduos está distribuído de forma aleatória (modelo bem ajustado) ou se o padrão de distribuição dos resíduos apresenta um tipo de tendência (modelo pode ter algum erro ou viés).

2.8 Interpretabilidade de Árvores de Classificação

A partir dos conceitos básicos sobre árvore de classificação, tais como descritos na Seção 2.5, uma árvore de classificação pode ser interpretada como um conjunto de nós galho que descrevem quais divisões o modelo aplicou para separar os dados em subconjuntos cada vez mais puros, começando pelo nó raiz e seguindo de forma subsequente até um de seus nós folhas, cujos os rótulos classificam todos os elementos contidos neles (MOLNAR, 2020).

É possível analisar cada valor de cada divisão que compõem uma árvore de classificação, como é possível entender o que levou o modelo à definir tais valores, conforme detalhado na Seção 3.1 e Seção 3.2. Os atributos que geram subconjunto

de dados mais puros, são também aqueles de maior importância para a classificação destes dados (MOLNAR, 2020).

Molnar (2020) também atribui vantagens às árvores de classificação como sendo "ideal para captura de interações entre atributos de dados", uma vez que os dados "terminam em grupos distintos", geralmente mais fáceis de entender do que dados dispersos em um espaço multidimensional, como ocorre na regressão linear.

Além disso, a estrutura da árvore de classificação pode ser desenhada de forma simples e natural usando nós e arestas, que pode ser lida de forma intuitiva através dos mesmos caminhos que os dados fazem no modelo instanciado, onde cada decisão é contrafactual, isto é, se o valor de atributo de um elemento é menor que um dado limiar, consequentemente ele não pode ser maior ou igual a este limiar, e vice-versa, de tal forma que, atendendo a condição, o elemento só pode ir para a esquerda ao invés de ir para a direita ou, caso contrário o elemento só pode ir para a direita ao invés de ir para a esquerda, respectivamente (MOLNAR, 2020). Cada caminho possível da árvore leva a um nó folha, que é rotulado com a classe da maioria dos elementos que chegam até ele e, então, todos os elementos neste nó são preditos como sendo da classe majoritária.

A interpretabilidade da árvore também depende da altura dela. Uma árvore com altura 2 precisa de, no máximo, 3 atributos para fazer a predição de classes de um problema e portanto requer apenas 3 atributos para formar a explicação para a classificação de um dado elemento. Desta forma uma árvore de classificação de altura pequena são geralmente simples, pois cada elemento contido em um único nó que o classifica pode ser explicado através de poucas decisões simples que envolvem apenas um atributo e um valor limiar (MOLNAR, 2020).

Entretanto, a quantidade de nós de uma árvore cresce exponencialmente à medida que a altura da árvore aumenta, portanto, quanto mais alta uma árvore, menos interpretável ela se torna. Além disso a confiabilidade de uma árvore depende do seu desempenho (MOLNAR, 2020) que pode ser avaliado usando técnicas como, por exemplo, *K-Fold* (Seção 4.2.3).

Por outro lado, Molnar (2020) declara que árvores (de decisão) falham ao lidar com relações lineares, isto porque qualquer relação linear entre entradas

e saídas acaba sendo feita pelas árvores através de aproximações, gerando uma função de etapas cujas mudanças entre uma saída e outra é feita de forma abrupta, afetando negativamente a eficiência do modelo. Outro ponto negativo das árvores de decisão é sua altura, quando estas precisam ser maiores, pois quanto mais altas, mais difíceis de serem interpretadas (MOLNAR, 2020; MOHSENI; ZAREI; RAGAN, 2021).

Portanto, em geral, há um consenso na bibliografia sobre a boa interpretabilidade das árvores de decisão, colocando-as como modelos de aprendizado de máquina muito interessantes, sobretudo para uso em contextos críticos.

3 Metodologia

Neste capítulo, são apresentados métodos, técnicas, modelos, entre outros utilizados na abordagem do problema de pesquisa. São descritos dois modelos de árvore de classificação, sendo eles: Árvore de Classificação e Regressão (CART), baseada em heurística, e Árvore de Classificação usando Otimização Inteira Mista (OCT-MIO). Por fim, também são descritas as técnicas utilizadas para avaliação de ambos os modelos.

3.1 Árvore de Classificação e Regressão (CART)

Árvore de Classificação e Regressão (CART) é um modelo conceitual de Árvore de Classificação proposto por Breiman et al. (2017). Trata-se de um modelo heurístico cujo o objetivo é obter uma árvore de classificação com a baixa impureza, dado um conjunto de dados. O modelo CART é baseado na estrutura de dados de árvore estritamente binária e está entre os modelos de árvore de classificação de maior destaque na literatura.

Árvore de classificação definida por Breiman et al. (2017) parte da estrutura de conjuntos divisíveis recursivamente, conforme descrito na Seção 2.5, onde há um conjunto de nós ligados, organizados hierarquicamente por arestas, pelos quais os dados passam e onde são particionados até alcançarem um nó folha. Nessa estrutura, os nós galho são responsáveis por aplicar particionamento ao conjunto de dados que passam através deles até chegarem aos nós folhas, os quais são responsáveis por rotular os subconjuntos dos dados contidos neles com uma das classes do problema.

Um conjunto de dados pode ser mais homogêneo ou mais heterogêneo. No contexto de árvore de classificação,

- o conjunto de elementos em um nó têm maior **impureza** quanto mais igual é a quantidade de elementos de cada classe. A maior impureza de um nó ocorre

quando um nó possui quantidades iguais de elementos de todas as classes do problema.

- quanto mais puro é um conjunto, maior é a proporção de elementos de uma mesma classe dentro deste conjunto, ou seja, quanto mais houver elementos de classes distintas, maior a impureza. Assim, a menor impureza ocorre quando um nó possui apenas elementos de uma única classe.

A **qualidade de uma partição**[1] é medida pela sua capacidade de separar elementos de classes distintas enquanto associa elementos de uma mesma classe. Quanto mais pura as partes geradas, melhor a qualidade da separação dos dados.

Para explicar o conceito de impureza é importante entender como as proporções afetam o problema. Haverá diferentes proporções a serem calculadas ao longo do problema, pois à cada divisão, as proporções se alteram e precisam ser recalculadas.

Para entender detalhadamente o conceito de impureza, primeiro é necessário entender alguns conceitos bases. O primeiro conceito, a **proporção de elementos de cada classe em um nó** ou simplesmente **proporção das classes em um nó** é a razão entre a **quantidade n_k de elementos x associados à classe k em um dado nó v pela quantidade total n_v** de elementos no mesmo nó, expressa por

$$p(k|v) = n_k/n_v, \quad k \in \{1, \ldots, K\}, \quad v \in V \tag{3.1}$$

e portanto

$$p(1|v) + \ldots + p(K|v) = 1. \tag{3.2}$$

Seguindo Breiman et al. (2017) a **medida de impureza** de um nó v denotada por $i(v)$ é obtida usando uma função não negativa $\phi : p(1|v) \times \ldots \times p(K|v) \rightarrow \mathbb{R}^+$, essa função deve ter o maior valor quando as proporções são idênticas para todas as classes, ou seja,

$$\phi(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}) = máximo, \tag{3.3}$$

[1] partição aqui é usado como sinônimo de divisão, e não o conceito de teoria dos conjuntos.

e, deve ser 0 quando todos os elementos daquele nó são da mesma classe, ou seja,

$$\phi(1, 0, \ldots, 0) = 0, \phi(0, 1, \ldots, 0) = 0, \ldots, \phi(0, 0, \ldots, 1) = 0. \tag{3.4}$$

Dessa forma $i(v)$ mede a impureza de um nó v por uma função ϕ que utiliza as proporções das classes para determinar a impureza do nó.

$$i(v) = \phi(p(1|v), p(2|v), \ldots, p(K|v)). \tag{3.5}$$

A impureza de um nó pode ser dada por qualquer função que implemente as funções (3.3), (3.4) e (3.5). A princípio, Breiman et al. (2017) escolheram uma função $i(v)$ que lhes foram mais familiar, dada por

$$i(v) = -\sum_{k=1}^{K} p(k|v) \log p(k|v), \quad v \in V. \tag{3.6}$$

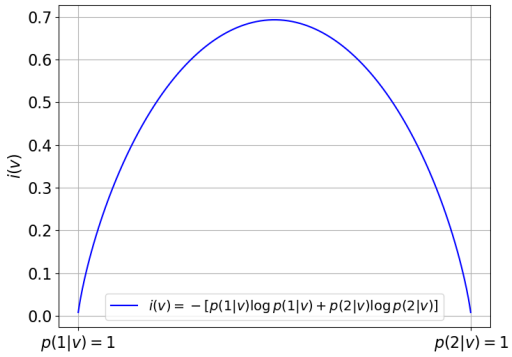

Figura 2 – Exemplo da equação (3.6) para 2 classes.

A função (3.6) mede a impureza de um nó usando dois fatores. A impureza $i(v)$ aumenta quando as proporções das classes são semelhantes e diminui quando são diferentes. O aumento de um termo $p(k|v)$ representa uma maior proporção daquela classe, tornando o nó mais homogêneo, o que parece contradizer o aumento da impureza. Para compensar, o termo $log\ p(k|v)$ diminui com o aumento de $p(k|v)$. Assim, o produto $p(k|v)\ log\ p(k|v)$ é maior quando $p(k|v)$ é mediana e menor quando $p(k|v)$ é extrema. A Figura 2 mostra o comportamento da função para 2 classes.

Para entender o conceito de impureza, foi necessário entender o conceito de proporção de cada classe em um nó. Adiante, para entender o conceito de variação de impureza, primeiro serão definidos dois conceitos relativos à proporção de elementos em um nó, independente de suas classes.

A proporção absoluta dos elementos de um nó v, ou apenas **proporção absoluta de um nó**, dada por $p(v)$, é a quantidade n_v de elementos contidos no nó $v \in V$ em relação à quantidade total n de elementos do conjunto de treinamento, ou seja:

$$p(v) = n_v/n, \tag{3.7}$$

tal que

$$p(v_L) + p(v_R) = p(v), \tag{3.8}$$

onde $p(v_L)$, $p(v_R)$ e $p(v)$ é a proporção absoluta dos nós filhos esquerdo v_L, direito v_R e do seu nó pai (v).

Para formar os conceito de **proporção relativa**, são tomadas as quantidades de elementos em alguns nós antes e depois de uma divisão. Assim a proporção relativa dos elementos de um nó filho, ou apenas **proporção relativa de um nó**, é determinada por p_v sendo a proporção absoluta $p(v), v \in V - v_1$ de elementos contidos em um nó filho v em relação à proporção absoluta do seu nó pai, dada por $p(v'), v' \in V - V_f$, o que equivale ao o número de elementos do nó filho (n_v) dividido pelo número de elementos do seu nó pai $(n_{v'})$. Portanto,

$$p_v = \frac{p(v)}{p(v')} = \frac{\frac{n_v}{n}}{\frac{n_{v'}}{n}} = \frac{n_v}{n} \times \frac{n}{n_{v'}} = \frac{n_v}{n_{v'}}, \quad v \in V - v_1, \quad v' \in V - Vf. \tag{3.9}$$

Para determinar a proporção relativa após uma divisão de um nó, tome um dado nó $v \in V_f$ a ser divido em dois nós filhos esquerdo e direito, respectivamente, v_L e v_R. Antes do particionamento, o nó pai v possui uma quantidade de elementos n_v. Suponha que, após o particionamento, uma quantidade de elementos n_{v_L} vai para seu filho esquerdo v_L e outra quantidade n_{v_R} para o filho direito v_R. As proporções relativas p_{v_L} e p_{v_R} de elementos contidos nos filhos esquerdo e direito são expressas respectivamente por

$$\begin{aligned} p_{v_L} &= p(v_L)/p(v), \\ p_{v_R} &= p(v_R)/p(v), \end{aligned} \tag{3.10}$$

tal que

$$p_{v_L} + p_{v_R} = 1. \tag{3.11}$$

Se a impureza é a medida $i(v)$ de heterogeneidade do conjunto de elementos em um nó v, a **impureza relativa** é a medida que relaciona a impureza de um nó à proporção relativa de elementos contidos nele. Portanto a impureza relativa nada mais é que o produto da proporção relativa de elementos p_v e a impureza $i(v)$ de um nó v, dada pela expressão

$$p_v i(v), v \in V. \tag{3.12}$$

Já a **impureza absoluta**, $I(v)$ é o produto da proporção absoluta $p(v)$ pela impureza $i(v)$ de um nó v, dada por

$$I(v) = p(v)i(v), v \in V. \tag{3.13}$$

Enquanto a impureza relativa $p_v i(v)$ dá uma dimensão da impureza apenas em relação à tríade nó pai - filho esquerdo - filho direito, a impureza absoluta $I(v)$ dá a dimensão da impureza em um nó v em relação à toda a árvore de classificação. Em outras palavras, para a impureza relativa $i(v)$, não importa o tamanho do conjunto sobre o qual a impureza é calculada, importa apenas as proporções das classes contidas nele. Já para a impureza absoluta $I(v)$, importa tanto as proporções das classes contidas no conjunto quanto o tamanho proporcional do conjunto. Ambos os conceitos de impureza baseiam os conceitos de variação da impureza adiante.

À medida que divisões são feitas nos nós da árvore, as impurezas dos nós folha também vão se alterando, e determinar quanto essa alteração afeta o particionamento dos dados e consequentemente a qualidade da árvore resultante é muito importante para determinar quando interromper o crescimento da árvore. A interrupção do crescimento da árvore pode ser determinada pela variação da impureza da árvore à cada divisão de nó e, portanto seria calculada inicialmente a partir da variação de impureza de cada nó individualmente, aqui chamada de variação da impureza relativa.

A **variação da impureza relativa**, Δi, é a diferença das impurezas relativas dos nós v, v_L e v_R, onde v é um dado nó da árvore, v_L seu filho esquerdo

e v_R seu filho direito, resultantes de uma divisão $s \in S$ (detalhada posteriormente) que separa os dados X_v, expressa como

$$\Delta i(s, v) = i(v) - p_{v_R} i(v_R) - p_{v_L} i(v_L), s \in S. \tag{3.14}$$

Observe que a impureza do nó pai $i(v)$ não está explicitamente multiplicada por uma proporção, já que sua proporção relativa é igual 1 (100%), isto é a soma das proporções relativas dos seus filhos p_{v_L} e p_{v_R} e, portanto, multiplicar $1 \times i(v)$ não faria diferença.

Contudo, a variação da impureza relativa $\Delta i(s, v)$ não expressa a variação da impureza em relação às variações das impurezas dos demais nós da árvore, apenas em relação à divisão de um dado nó pai em dois nós filhos. Por isso, tome como exemplo a divisão de dois nós distintos v_2 e v_3 contendo uma composição de n elementos $x \in X_{v1}, X_{v_2} \cup X_{v_3} = X_{v_1} = \{x_1, x_2, \dots x_{10}\}$ de classes $k \in \{1, 2, 3, 4\}$, sendo que os elementos $x_1, \dots, x_{10}$ têm respectivamente classes 1, 1, 1, 1, 2, 2, 2, 2, 3, 4. Seja $X_{v_2} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ e $X_{v_3} = \{x_9, x_{10}\}$, note que eles podem produzir variações de impurezas relativas iguais quando as proporções de elementos de cada classe são iguais, ainda que as quantidades não sejam. Por exemplo, suponha que uma divisão s de v_2 separe todos os elementos da classe 1 no nó filho esquerdo v_4 e todos os elementos da classe 2 no nó filho direito v_5, conforme a Figura 5. A variação da impureza relativa causada por esta divisão é dada por

$$\begin{aligned}
\Delta i(s, v_2) &= p_{v_2} i(v_2) - p_{v_4} i(v_4) - p_{v_5} i(v_5) \\
&= \mathbf{1} \times i(v_2) - \mathbf{0.5} \times i(v_4) - \mathbf{0.5} \times i(v_5) \\
&= \mathbf{1} \times 0.30 - \mathbf{0.5} \times 0 - \mathbf{0.5} \times 0 \\
&= 0.30 - 0 - 0 = \mathbf{0.30}, \\
i(v_2) &= -\sum_{k=1}^{4} p(k|v_2) \log p(k|v_2) = -(0.5 \log 0.5 + 0.5 \log 0.5 + 0 + 0) = 0.30, \\
i(v_4) &= -\sum_{k=1}^{4} p(k|v_4) \log p(k|v_4) = -(1 \log 1 + 0 + 0 + 0) = 0, \\
i(v_5) &= -\sum_{k=1}^{4} p(k|v_5) \log p(k|v_5) = -(0 + 1 \log 1 + 0 + 0) = 0,
\end{aligned}$$

e suponha uma divisão s' para v_3 de modo a separar todos os elementos da classe 3 no nó filho esquerdo v_6 e todos elementos da classe 4 no nó filho direito v_7. A variação da impureza relativa causada por esta divisão é dada por

$$\begin{aligned}
\Delta i(s', v_3) &= p_{v_3} i(v_3) - p_{v_6} i(v_6) - p_{v_7} i(v_7) \\
&= \mathbf{1} \times i(v_3) - \mathbf{0.5} \times i(v_6) - \mathbf{0.5} \times i(v_7) \\
&= \mathbf{1} \times 0.30 - \mathbf{0.5} \times 0 - \mathbf{0.5} \times 0 \\
&= 0.30 - 0 - 0 = \mathbf{0.30}, \\
i(v_3) &= -\sum_{k=1}^{4} p(k|v_3) \log p(k|v_3) = -(0 + 0 + 0.5 \log 0.5 + 0.5 \log 0.5) = 0.30, \\
i(v_6) &= -\sum_{k=1}^{4} p(k|v_6) \log p(k|v_6) = -(0 + 0 + 1 \log 1 + 0) = 0, \\
i(v_7) &= -\sum_{k=1}^{4} p(k|v_7) \log p(k|v_7) = -(0 + 0 + 0 + 1 \log 1) = 0,
\end{aligned}$$

Observe que, apesar dos nós v_2 e v_3 possuírem quantidades diferentes de elementos cujas as classes associadas também são diferentes, as proporções de elementos de cada classe em cada nó são iguais (50%) e, portanto, a variação de impureza relativa $\Delta i(s, v_2)$ e $\Delta i(s', v_3)$ são iguais. Sendo assim, apenas a impureza relativa não é suficiente para refletir a grandeza da variação da impureza em relação à quantidade de elementos separados, em outra palavras, ela não é suficiente para medir qual nó $v \in V_f$ promove maior predição de elementos ao ser divido. Para isso, também é introduzido o conceito de variação impureza absoluta.

Variação de impureza absoluta, $\Delta I(s, v)$, é a diferença entre as impurezas absolutas de um nó pai v e seus filhos esquerdo v_L e direito v_R, quando v é submetido a uma divisão s. De forma equivalente, é o produto da variação relativa da impureza pela proporção absoluta de elementos do nó v, o qual seu conjunto de elementos sofre divisão.

$$\begin{aligned}
\Delta I(s, v) &= I(v) - I(v_L) - I(v_R) \\
&= p(v) i(v) - p(v_L) i(v_L) - p(v_R) i(v_R) \\
&= [i(v) - p_{v_L} i(v_L) - p_{v_R} i(v_R)] p(v).
\end{aligned} \tag{3.15}$$

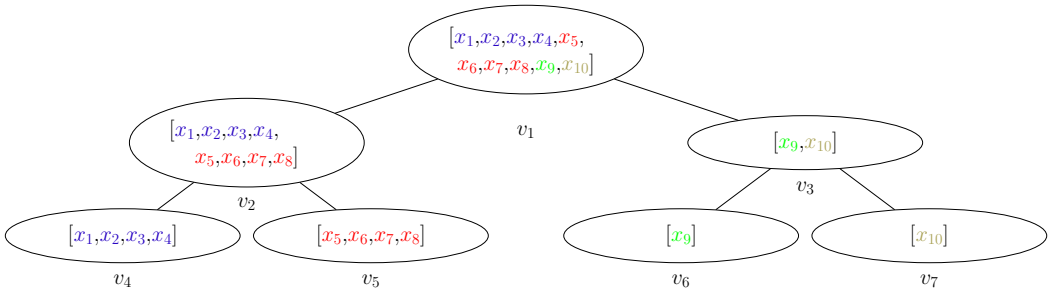

Figura 3 – Exemplo de separação de elementos de mesma classe

Enquanto $\Delta i(s, v)$ é afetada apenas pelas proporções, mas não pelo número de elementos divididos, $\Delta I(s, v)$ é afetada tanto pelas proporções quanto pelo número de elementos envolvidos no particionamento.

Tomando agora o mesmo exemplo da divisão de dois nós v_2 e v_3 contendo uma composição de n elementos $x \in X_{v_1}, X_{v_2} \cup X_{v_3} = X = \{x_1, x_2, \ldots x_{10}\}$ de classes $k \in \{1, 2, 3, 4\}$, sendo os elementos $x_1, \ldots, x_{10}$ respectivamente das classes 1,1,1,1,2,2,2,2,3,4 e sendo, $X_{v_2} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ e $X_{v_3} = \{x_9, x_{10}\}$, as variações das impurezas absolutas não são iguais, diferente do que acontece com as variações das impurezas relativas.

Por exemplo, suponha que v_2 seja dividido de modo a separar todos os elementos da classe 1 no nó filho esquerdo v_4 e todos os elementos da classe 2 no nó filho direito v_5, também tal como na Figura 5. A variação da impureza absoluta

causada por esta divisão é dada por

$$\begin{aligned}
\Delta I(s, v_2) &= I(v_2) - I(v_4) - I(v_5) \\
&= \mathbf{0.8} \times i(v_2) - \mathbf{0.4} \times i(v_4) - \mathbf{0.4} \times i(v_5) \\
&= \mathbf{0.8} \times 0.30 - \mathbf{0.4} \times 0 - \mathbf{0.4} \times 0 \\
&= \mathbf{0.24},
\end{aligned}$$

$$i(v_2) = -\sum_{k=1}^{4} p(k|v_2) \log p(k|v_2) = -(0 + 0 + 0.5 \log 0.5 + 0.5 \log 0.5) = 0.30,$$

$$i(v_4) = -\sum_{k=1}^{4} p(k|v_4) \log p(k|v_4) = -(0 + 0 + 1 \log 1 + 0) = 0,$$

$$i(v_5) = -\sum_{k=1}^{4} p(k|v_5) \log p(k|v_5) = -(0 + 0 + 0 + 1 \log 1) = 0,$$

e suponha que o v_3 seja dividido de modo a separar todos os elementos da classe 3 no nó filho esquerdo v_6 e todos elementos da classe 4 no nó filho direito v_7. A variação da impureza absoluta causada por esta divisão é dada por

$$\begin{aligned}
\Delta I(s', v_3) &= I(v_3) - I(v_6) - I(v_7) \\
&= \mathbf{0.2} \times i(v_3) - \mathbf{0.1} \times i(v_6) - \mathbf{0.1} \times i(v_7) \\
&= \mathbf{0.2} \times 0.30 - \mathbf{0.1} \times 0 - \mathbf{0.1} \times 0 \\
&= \mathbf{0.06},
\end{aligned}$$

$$i(v_3) = -\sum_{k=1}^{4} p(k|v_3) \log p(k|v_3) = -(0 + 0 + 0.5 \log 0.5 + 0.5 \log 0.5) = 0.30,$$

$$i(v_6) = -\sum_{k=1}^{4} p(k|v_6) \log p(k|v_6) = -(0 + 0 + 1 \log 1 + 0) = 0,$$

$$i(v_7) = -\sum_{k=1}^{4} p(k|v_7) \log p(k|v_7) = -(0 + 0 + 0 + 1 \log 1) = 0,$$

Ainda que as divisões s e s', que maximizam $\Delta i(s, v_2)$ e $\Delta i(s', v_3)$, também maximizem $\Delta I(s, v_2)$ e $\Delta I(s', v_3)$, a variação $\Delta I(s,v_2)$ produz uma redução mais significativa na impureza que a variação $\Delta I(s',v_3)$, pois ela separa um número maior de elementos de classes diferentes.

A tarefa de escolher o valor limiar que irá particionar os dados a cada divisão de um nó da árvore é o grande desafio do método CART, pois uma escolha ruim, principalmente no início da árvore, pode comprometer negativamente tanto a complexidade da árvore, quanto a qualidade dos conjuntos resultantes das divisões seguintes.

Uma das principais características de uma árvore de classificação do modelo CART é que ela divide os dados sempre em relação à um único valor de um único atributo de cada vez, isto é, cada divisão ocorre comparando sempre o valor de um mesmo atributo de cada um dos elementos contidos no nó com um único valor de limiar.

Breiman et al. (2017) modelam as formas de se dividir os dados tanto categóricos, quanto contínuos, através dos nós da árvore, porém este trabalho concentra-se na utilização de dados contínuos, utilizando-se da representação dos dados categóricos também como valores numéricos. Sendo assim, as **divisões** S de uma árvore de classificação são aplicadas através de condições (que podem ser interpretadas como perguntas) na forma $\{x_p < b_v\}$ aplicadas aos dados em cada nó $v \in V_b$.

Cada **divisão** $s \in S$ envolve a comparação de grandeza ($<$) de valores x_{ip} do atributo p de todos os elementos $x_i \in X_v$ a um único valor limiar b_v associado ao nó v.

O **limiar** b_v é um valor dentro da amplitude de $x_{ip} \in X_v^P$ capaz de dividir o conjunto X_v, ou seja, que $x_{ip}minimum < b_v \leq x_{ip}maximum, \quad x_{ip} \in X_v^P$.

Um possível **limiar** para o particionamento dos dados $\boldsymbol{x}$ é um valor b que está entre quaisquer dois valores ordenados consecutivos $\boldsymbol{x}_i$ e $\boldsymbol{x}_{i+1} \in X_v^P, i = [1, \ldots, n_v - 1]$ de atributos em um conjunto ordenado de dados.

Para obter os valores limiares b candidatos em cada divisão $s \in S$ a partir dos valores de atributos dos dados X_v é necessário:

1. Ordenar os valores x_{ip} restantes;

2. Descartar valores x_{ip} idênticos.

É necessário **ordenar os valores dos atributos** x_{ip} já que os valores buscados são valores intermediários entre dois pares x_{ip} e $x_{(i+1)p}$ de um mesmo atributo. Também é necessário **descartar os valores idênticos** pois não há valor intermediário entre dois valores x_{ip} e $x_{(i+1)p}$ idênticos. Observe que, ao remover os valores duplicados de cada coluna p de atributos em X, a quantidade de valores (linhas) em cada coluna de atributo p pode ser totalmente diferente. Feito isso, as divisões possíveis para particionar os dados X_v são todas compostas por valores b médios $(x_{ip} + x_{(i+1)p})/2$, onde $i \in X_v^P$.

Existem variados **critérios para definir qual o melhor limiar** b para dividir um nó. O primeiro deles é a **maior variação de impureza**[2], especificamente, o valor limiar que, ao dividir o conjunto de dados, gera a máxima variação de impureza entre um determinado conjunto de elementos de um nó v e o conjunto de elementos de seus nós filhos, v_L e v_R, isto é,

$$\Delta i(s*, v) = \max_{s \in S} \Delta i(s, v). \tag{3.16}$$

Onde $s*$ é a divisão $s \in S$ que gera a máxima variação da impureza em v. Portanto, quanto maior $\Delta i(s, v)$, melhor é a divisão $\phi(s, v)$.

Então, a estratégia de crescimento de uma Árvore de Classificação e Regressão consiste em dividir um conjunto de dados X_v que chegam em cada nó v até que o conjunto X_v tenha impureza $= 0$, em outras palavras, até que $\Delta I(s*,v) = 0$. Quando isso acontece, v é definido nó folha $v \in V_f$. Este é o critério natural de interrupção do crescimento da árvore no CART. Embora essa estratégia pareça simples e objetiva, ela tende a criar sobreajuste do modelo, sensibilidade à ruído, perda de interpretabilidade, entre outros. Para rebater tais problemas, Breiman et al. (2017) apresentaram heurísticas de interrupção do crescimento da árvore.

A primeira **heurística para interromper o crescimento da árvore** é determinar que um nó não se divida se a variação da impureza absoluta que ele gera for irrelevante, isto é, se $\Delta I(s, v)$ for menor que um valor de **mínima variação de impureza**, determinado para o CART por parâmetro. Este valor pode variar

[2] Quando se analisa a variação da impureza de um nó isoladamente, tanto a variação da impureza absoluta quanto relativa, maximizam a variação da impureza, como neste caso.

em função de alguns fatores como equilíbrio entre acurácia e interpretabilidade, tamanho do conjunto de dados, entre outros.

Diferente de Breiman et al. (2017), neste trabalho optou-se por definir a "mínima variação de impureza absoluta" = 0. Contudo, foi implementada outra heurística que permite que a árvore se divida até uma **altura máxima** $\in \mathbb{Z}^{*+}$, definida ao CART por parâmetro. Assim, mesmo que haja alguma redução de impureza absoluta em um determinado nó, ele só poderá se dividir se a altura do nó for menor que a altura máxima definida por parâmetro. Esta heurística foi proposta com o objetivo de estabelecer um limite de comparação equivalente entre o CART e outros modelos que também implementam a heurística de interrupção de crescimento da árvore por uma altura máxima. Além disso, para valores pequenos de altura máxima da árvore, problemas como sobreajuste e redução da interpretabilidade não se manifestam tão evidentemente.

Uma vez determinada a condição de parada, obtêm-se consequentemente também os critérios que determinam um **nó folha** da árvore de classificação. A estagnação da redução da impureza e a heurísticas de interrupção do crescimento da árvore proporcionam condições para estabelecimento de nós folhas contendo um subconjunto de alguns elementos X_v tal que $v \in V_f$.

Conforme já mencionado nas relações e restrições da árvore de classificação, o **rótulo de classe** associado à um nó folha é dado pela classe associada a maior quantidade de elementos contidos no nó, como, por exemplo, a regra da pluralidade expressa em

$$p(k*|v) = \max_k p(k|v), \tag{3.17}$$

onde $k*$ é a classe que rotula o nó folha v da árvore de classificação. De posse destes conceitos, é possível entender como uma árvore de classificação é determinada e quais, quantos e em qual ordem os particionamentos são aplicados aos dados do problema.

Após todos os particionamentos, o arranjo final dos nós é dado como uma *árvore binária* T, onde os nós terminais são representados por V_f. Nesse momento é importante determinar uma medida de impureza da árvore por inteira à fim determinar quão boa ela é.

A impureza total da árvore ou apenas **impureza da árvore** é o somatório das impurezas absolutas de cada nó folha $v \in V_f$ da árvore, descrita por

$$I(T) = \sum_{v \in V_f} I(v) = \sum_{v \in V_f} i(v)p(v). \tag{3.18}$$

Portanto a escolha das divisões $s \in S$ que minimizam a impureza da árvore $I(T)$ é a mesma escolha das divisões $s \in S$ que maximizam as variações da impureza absoluta $\Delta I(s, v)$, já que quanto maior a variação da impureza absoluta, menor será a soma das impurezas absolutas dos nós filhos e menor será a impureza da árvore.

Uma vez calculada a impureza da árvore $I(T)$, se houver mais uma divisão, é possível **calcular a impureza da nova árvore** T' após uma nova divisão de um determinado nó. Suponha que haja uma árvore de classificação já estabelecida com alguns nós folhas e com a impureza da árvore já calculada. Se um dado nó $v \in V_f$ for dividido em outros dois nós filhos, v_L e v_R, a nova impureza da árvore pode ser calculada pelo somatório das impurezas absolutas dos nós folhas já existentes antes da última divisão, com exceção do nó que foi dividido ($\sum_{V-\{v\}} p_v i(v)$), mais a impureza absoluta dos dois novos nós folhas, $p_L i(v_L)$ e $p_R i(v_R)$, gerados pela divisão, portanto

$$I(T') = \sum_{V_f - \{v\}} I(v) + I(v_L) + I(v_R). \tag{3.19}$$

A diferença da impureza de antes para depois da divisão é dada pela diferença da impureza da árvore antes da divisão pela impureza da árvore depois da divisão sendo

$$i(T) - i(T') = I(v') - I(v_L) - I(v_R). \tag{3.20}$$

Breiman et al. (2017) ainda avançam sobre conceitos de classificações incorretas e seus derivados na direção de propor uma segunda etapa de poda da árvore para obter melhor generalização e evitar ajustes excessivo do classificador sobre os dados de treinamento. Contudo, com os conceitos já apresentados, é possível implementar uma árvore de classificação baseada no CART.

O modelo CART implementado neste trabalho está disponível em https://github.com/jaderfigueredo/tree_cart.

3.2 Árvore de Classificação Ótima usando Otimização Inteira Mista (OCT-MIO)

Árvore de Classificação Ótima é um modelo de árvore de classificação com objetivo de se obter o melhor ajuste e a menor complexidade possível, dado um conjunto de dados e de regras para a classificação destes dados. Esta seção apresenta a modelagem matemática de uma Árvore de Classificação Ótima usando Otimização Inteira Mista (Programação Inteira Mista), conforme modelado por Bertsimas e Dunn (2017) e implementado neste trabalho.

A árvore de Classificação Ótima não implementa a estrutura convencional de nós e arestas como ponteiros que ligam um nó ao outro. Ela utiliza-se de variáveis, seus domínios juntamente com expressões e funções sobre estas variáveis para representar o mesmo mecanismo utilizado na estrutura convencional.

Assim como na definição da Fundamentação Teórica (Seção 2.5), os dados para a árvore de classificação ótima também estão estruturados como dois conjuntos $(X, \boldsymbol{y})$ de n **elementos** $(\boldsymbol{x}_i, y_i)$, $i = \{1, \ldots, n\}$ onde cada $\boldsymbol{x}_i$ é composto por P atributos $p \in \mathbb{R}^P$ e associado à uma das possíveis **classes** $k \in \{1, \ldots, K\}$ através da variável y_i.

Bertsimas e Dunn (2017) definem **árvore de classificação** como um método que divide recursivamente os dados em partições $[0, 1]^P$ obtendo regiões hierárquicas disjuntas. A árvore final é composta de nós V sendo nós folha e nós galho.

Nós galho ou internos, V_b, são unidades componentes da árvore responsáveis por particionar os dados que passam através deles. Para isso, eles utilizam dois parâmetros $\boldsymbol{a}$ e b, onde $\boldsymbol{a}$ é um vetor de valores binários que determinam qual atributo $\boldsymbol{x}_{ip}$ é utilizado para determinar o particionamento dos dados e b é o valor limiar que irá determinar se um elemento $\boldsymbol{x}_i$ seguirá para o nó filho esquerdo ou direito. Assim como no CART, no OCT-MIO apenas um atributo de $\boldsymbol{x}_i$ é considerado em cada divisão de nó. Isto significa que, a cada divisão, apenas um índice $\boldsymbol{a}_p$ terá valor 1 enquanto todos os demais terão valor 0. Matematicamente, para cada divisão em um nó galho, tem-se que $\sum_{i=1}^{P} a_i = 1$.

Nós folha ou terminais, V_f, também são unidades componentes da árvore,

mas, diferente dos nós galho, os nós folha são responsáveis por comportar um conjunto de elementos afins ao mesmo tempo que lhes atribui um rótulo de classe. Cada nó folha possui obrigatoriamente uma classe que o rotula. Este rótulo de classe é determinado pela classe da maioria dos elementos que caem dentro deste nó folha durante o treinamento.

O processo de **divisão da árvore** de classificação obedece critérios para sua interrupção. De modo geral, a qualidade da partição, dada pela impureza do conjunto de elementos, é o principal fator a ser considerado para esse propósito, contudo também há condições específicas como um **número mínimo, n_{min}, de elementos necessários para compor um nó folha**. Ou seja, se um dado particionamento gera um nó $v_i \in V$ com menos de n_{min} elementos, este nó v_i não obedece um dos critérios de interrupção do particionamento e, portanto, o particionamento não é aplicado.

Tanto o modelo CART quanto o OCT-MIO avaliam a impureza dos dados para determinar o critério de parada no crescimento da árvore, contudo, enquanto o CART leva em consideração apenas o fator da impureza do conjunto de elementos de um nó na primeira etapa, o OCT-MIO considera dois fatores em uma única etapa, sendo eles não só a impureza do conjunto de elementos mas também a penalidade sobre a complexidade da árvore como um todo. Assim, o OCT-MIO busca obter uma árvore tanto bem acurada quanto simples em uma única etapa.

Usando detalhes da metodologia do CART, o OCT-MIO se propõe resolver o mesmo problema de criação da árvore de classificação, mas como um problema formal de otimização. Para isso, são determinados dois fatores divergentes: acurácia e complexidade da árvore. À medida que a acurácia aumenta, a complexidade da árvore também aumenta e vice-versa, o que é algo indesejável. Portanto há um empasse entre os dois fatores, e o objetivo é encontrar o melhor equilíbrio entre ambos.

A **medida de complexidade**, composta pelo produto αN_{V_b}, faz com que a constante α penalize a solução em função da **quantidade de nós galho**, N_{V_b}, da árvore, enquanto a **acurácia** é dada pelo número de erros de classificação total, baseada nos nós folha V_f da árvore T e é representada por $R_{xy}(T)$. Além deste

empasse entre acurácia e complexidade, outra restrição da árvore é que a quantidade de elementos dos nós folhas, dada por N_{V_f} precisam ser no mínimo n_{min}. Dito isto, tem-se então a seguinte função objetivo e restrição do modelo parcial:

$$\begin{aligned} & min \quad R_{xy}(T) + \alpha N_{V_b} \\ s.t. \quad & n_v \geq n_{min} \quad \forall\ v \in V_f. \end{aligned} \tag{3.21}$$

O modelo completo é apresentado ao final desta seção.

Observe que, ao lidar com a acurácia e complexidade da árvore de classificação simultaneamente, não é necessário um segundo passo para fazer poda da árvore, como ocorre no CART modelado por Breiman et al. (2017). Bertsimas e Dunn (2017) também declaram que a utilização do método CART como base para o método OCT-MIO é meramente arbitrária. O método OCT-MIO também poderia ser abordado utilizando outros métodos de base, tais como C4.5 e ID3.

Então, conforme relatado, a criação de uma árvore de classificação de natureza topo-base, tal como apresentado por Breiman et al. (2017), que considera apenas a impureza dos dados durante a etapa de crescimento, pode ficar presa em ótimos locais, uma vez que particionamentos melhores podem estar escondidos atrás de particionamentos piores.

Agora, de acordo como Bertsimas e Dunn (2017), para modelar uma árvore de classificação como um problema de otimização inteira mista é necessário definir um número de **decisões discretas**, que são:

- A cada novo nó criado, é necessário decidir se ele será um nó galho ou um nó folha;
- Se for decidido interromper o particionamento em um dado nó, tornando-o um nó folha, é necessário atribuir um rótulo de classe a ele;
- Se for decidido particionar os dados em um dado nó, tornando-o um nó galho, então é necessário escolher qual atributo será considerado para o particionamento;

- Durante o processo de construção da árvore, ou seja, de classificação da amostra de treinamento, deve-se escolher a quais nós folha cada elemento do conjunto de treinamento será atribuído;

Além destas, há ainda decisões contínuas acerca do limiar:

- Quais valores de limiares serão definidos para cada nó galho.

Na modelagem da **Árvore de Classificação Ótima (OCT) usando Otimização Inteira Mista (MIO)**, são definidos alguns conceitos. O primeiro deles, a **altura máxima da árvore**, D, diz respeito ao número de níveis, que é o mesmo que a altura da árvore, o qual é usado para determinar o número máximo de nós que essa árvore pode ter. A quantidade máxima de nós v em uma árvore T é dada por $N_{max} = 2^D - 1, v \in \{1, \ldots, N_{max}\}$. Já a quantidade de nós existentes na árvore T, estes são representados apenas por N.

Para entender as definições vindas à seguir, antes é necessário entender as **referências feitas sobre a estrutura hierárquica de uma árvore de classificação**. Tome como referência um dado nó $v \in V$ de uma árvore de classificação, considere $p(v)$ seu **nó pai**, $A(v)$ **todos os seus ancestrais**, ou seja, todos os nós tangíveis pelo caminho feito da raiz da árvore, v_1, até v. Considere $A_L(v)$ um subconjunto de $A(v)$ composto apenas pelos **nós ancestrais de v, a partir dos quais segue-se pela ramificação esquerda** ao longo do caminho de v_1 até v. E considere $A_R(v)$ um subconjunto de $A(v)$ composto apenas pelos **nós ancestrais de v, a partir dos quais segue-se pela direita** ao longo do caminho de v_1 até v, de modo que $A(v) = A_L(v) \cup A_R(v)$ e $A_L(v) \cap A_R(v) = \emptyset$.

Nós galho são nós $v \in V_b = \{1, \ldots, \left\lfloor N/2 \right\rfloor\}$, que aplicam particionamento no formato $\boldsymbol{a}^\intercal \boldsymbol{x} < b$ sobre os dados que passam através deles. Os elementos $\boldsymbol{x}$ que satisfazem a condição do particionamento seguem para o o nó filho esquerdo, enquanto os elementos que não satisfazem seguem para o nó filho direito.

Nós folha são nós $v \in V_f = \{\left\lfloor N/2 \right\rfloor + 1, \ldots, N\}$ que fazem a predição de classe dos elementos que chegam até eles atribuindo-lhes um rótulo de classe.

Bertsimas e Dunn (2017) definem quais nós aplicam divisão dos dados e quais não, da seguinte maneira: Seja $\boldsymbol{d} \in \{1, 0\}$ um **vetor de valores binários, de tamanho N, que mapeia cada nó v para um de seus índices**, onde $d_v = 1$ se o nó v aplica particionamento em seus dados e $d_v = 0$, caso contrário.

Além disso, os **nós que aplicam divisão** dos dados são nós $v \in V_b$ que possuem variáveis $\boldsymbol{a}_v \in \mathbb{R}^P$ e b_v. Baseando-se ainda no método CART, Bertsimas e Dunn (2017) definiram que **cada particionamento será feito em relação a apenas um atributo** e isso é feito definindo-se que a soma de todos os valores do vetor $\boldsymbol{a}_v$ é igual à d_v.

$$\sum_{p=1}^{P} a_{pv} = d_v, \quad \forall v \in V. \tag{3.22}$$

Para modelar a **interrupção do crescimento da árvore**, eles definiram que as variáveis $\boldsymbol{a}_v = \mathbf{0}$ e $b_v = 0$, onde $\boldsymbol{a}_v = \mathbf{0}$ determina que nenhum atributo será utilizado para qualquer divisão em v e ainda $b_v = 0$ define o limiar em 0. Considerando que todo o valor de qualquer atributo de qualquer elemento $x_{ip} \in X^P$ está normalizado em $[0, 1]^P$, então qualquer possível comparação para separação dos elementos em v, seria $x_{ip} < 0$ e, neste caso, jamais será verdadeira, o que consequentemente enviará qualquer possível elemento para o nó filho direito de v. Contudo, ao invés de a estrutura enviar todos os nós para um suposto filho direito de v, os elementos são conceitualmente mantidos em v. Bertsimas e Dunn (2017) modelaram a interrupção da divisão de cada nó desta maneira, pois assim não serão necessárias variáveis adicionais para tratar a interrupção do crescimento da árvore.

Eles também descreveram o **limiar** sendo um valor compreendido entre 0 e 1 como

$$0 \leq b_v \leq d_v, \quad \forall v \in V, \tag{3.23}$$

e $\boldsymbol{a}_v$ como um vetor de valores binários, 0 ou 1, de tamanho igual à quantidade P de atributos, os quais já foram descritos neste capítulo, tal que

$$a_{pv} \in \{0,1\}, \quad p = 1, \ldots, P, \quad \forall v \in V_b, \tag{3.24}$$

As definições acima colaboram para definir que sempre será verdade a expressão $0 \leq \boldsymbol{a}_v^\intercal \boldsymbol{x}_i \leq d_v$, que diz que o valor de qualquer atributo p, considerado

para uma divisão (a_{pv}) do elemento $\boldsymbol{x}_i$, está sempre compreendido entre 0 e 1, para todo nó galho $v \in V_b$. Portanto, o limiar expresso por b_v para dividir o nó v também é um valor compreendido entre 0 e 1.

Para definir a **estrutura hierárquica da árvore** e relacionar os nós através de ligações que formam **caminhos consistentes até todos os nós da árvore**, é definido que todo nó só possa fazer particionamento dos dados se seu pai também o fez. Desta forma, é garantido que, ao percorrer um caminho através da árvore, cada elemento que chega a um determinado nó v, passe por outros nós $v \in V$ que obrigatoriamente fizeram particionamento, pelos quais ao menos um dos caminhos certamente conduz os dados até v. Essa definição não se aplica ao nó raiz da árvore, já que este é o único nó da árvore que não possui nó ancestral. Matematicamente, isto é demonstrado por

$$d_v \leq d_{p(v)}, \quad \forall v \in V_b V \setminus \{1\}, \tag{3.25}$$

onde, conforme já descrito, $p(v)$ é o nó pai de v.

Para **mapear em qual nó v está um determinado elemento $\boldsymbol{x}_i$**, é introduzida a variável $z_{iv} = \mathbb{1}$, sendo z_{iv} cada posição de uma matriz $\mathbf{Z}$ onde a dimensão $\mathbf{z}_i$ se refere ao elemento $x \in X$ de índice i e $\mathbf{z}_v$ se refere ao nó $v \in V$ onde $\boldsymbol{x}_i$ se encontra. Junto com a variável $l_v = \mathbb{1}$, que define **se um dado nó v contém algum elemento nele**, é possível modelar as próximas definições da seguinte maneira:

Primeiro, se um nó v contém elementos, sejam quantos forem, então $l_v = 1$. E se quaisquer elementos x_i puderem estar em v, tornando verdade $z_{iv} = 1$, então a Expressão (3.26) é satisfeita. Mas se não há quaisquer elementos em v, sendo $l_v = 0$, então z_{iv} só pode ser 0 para que a Expressão (3.26) seja verdadeira.

$$z_{iv} \leq l_v, \quad v \in V_b, \tag{3.26}$$

Segundo, para satisfazer a condição de que **todo nó folha não só possua elementos, mas também tenha a quantidade mínima necessária deles para a solução considerá-lo um nó folha**, então a soma de todos os elementos $\boldsymbol{x}_i$ contidos em v deve ser maior ou igual n_{min}.

$$\sum_{i=1}^{n} z_{iv} \geq n_{min} l_v, \quad v \in V_f. \tag{3.27}$$

Observe que, se não há elementos em v, ou seja, $l_v = 0$, então não importa qual valor mínimo de elementos, n_{min} para que v seja um nó folha, pois o produto $n_{min}l_v$ será 0, tornando a restrição (3.27) inativa e, portanto, indiferente neste caso. Por outro lado, se há elementos em v, ou seja, $l_v = 1$, então a restrição se aplica, impedindo que haja um nó folha $v \in V_f$ com menos de n_{min} elementos.

É preciso também **restringir que cada elemento seja atribuído a apenas um nó folha** que o classificará, evitando, por exemplo, que um mesmo elemento seja classificado por duas ou mais classes distintas. Para isso, foi definido que, fixado um determinado elemento $\boldsymbol{x}_i$ na dimensão z_i, o somatório dos nós v para os quais $\boldsymbol{x}_i$ está atribuído deve ser apenas 1, tal como expresso por

$$\sum_{v \in V_f} z_{iv} = 1, \quad i = 1, \ldots, n. \tag{3.28}$$

Por fim, as definições de **particionamento** que atribuem um determinado elemento $\boldsymbol{x}_i$ à um nó v são dadas pelas expressões

$$\boldsymbol{a}_m^\intercal \boldsymbol{x}_i < b_m + M_1(1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_L(v), \tag{3.29}$$

$$\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \geq b_m - M_2(1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_R(v), \tag{3.30}$$

onde a expressão (3.29) diz que um elemento x_i em um nó folha $v \in V_f$ atende a todas condições impostas nos nós galhos v_m, para todo m pertencente a $A_L(v)$, onde $A_L(v)$ é o subconjunto de nós galho ancestrais de v onde o valor $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i$ foi menor que o limiar b_m de cada nó v_m, conforme destaca a Figura 4.

Entendendo cada parte, tem-se que o vetor $\boldsymbol{a}$ possui valor 1 apenas em um índice a_p qualquer, tal que $p \in [1, \ldots, P]$, isto significa que todos os demais índices p tem valor 0. Portanto, o produto $\boldsymbol{a}_v^\intercal \boldsymbol{x}_i$ conserva um valor x_{ip} enquanto anula todos os demais. Sendo assim, o produto no lado esquerdo da inequação será, na verdade, o valor x_{ip} de um dado atributo p e este valor de atributo precisa ser menor que a soma do limiar b_v mais o produto $M_1(1 - z_{iv})$ que será detalhado adiante.

Da mesma forma, a expressão (3.30) diz que um elemento x_i em um nó folha $v \in V_f$ atende a todas condições impostas nos nós galhos v_m, para todo m

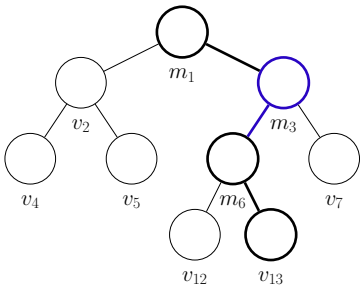

Figura 4 – Nós cujo valor de um atributo do elemento é menor que o limiar naquele nó

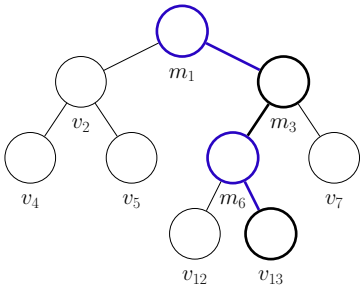

Figura 5 – Nós cujo valor de um atributo do elemento é maior ou igual que o limiar naquele nó

pertencente a $A_R(v)$, onde $A_R(v)$ é o subconjunto de nós galho ancestrais de v onde o valor $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i$ foi maior ou igual que o limiar b_m de cada nó v_m, conforme destaca a Figura 4.

Portanto, o lado esquerdo de ambas expressões nada mais é que a escolha de um atributo de $\boldsymbol{x}_i$ para ser comparado à b_v.

Para entender melhor o segundo componente, $\pm M(1 - z_{iv})$, considere primeiro o termo $(1 - z_{iv})$. z_{iv} vale 1 se um dado elemento $\boldsymbol{x}_i$ está no nó v, e vale 0 caso contrário. Sendo assim, sempre que um dado elemento $\boldsymbol{x}_i$ estiver no nó v, a diferença entre parênteses será $1 - 1 = 0$, e então não importa mais o valor de M

pois o produto $M \times 0$ também é 0. Mas se um elemento $\boldsymbol{x}_i$ não estiver em v, sendo então $z_{iv} = 0$, a diferença entre parênteses é 1 e, agora, o produto é o próprio M. Portanto é possível perceber que termo $(1 - z_{iv})$ age como uma chave que remove a constante M da expressão apenas quando um dado elemento $\boldsymbol{x}_i$ está em um determinado nó v em questão. Esse raciocínio é valido tanto para a expressão (3.29) quanto para expressão (3.30).

Conforme apresentado adiante, o objetivo de M é empurrar o limiar b_v para a extremidade ou além da amplitude de $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i$, relaxando as restrições (3.29) e (3.30). Para que isso ocorra, é necessário que M seja, no mínimo, maior que o módulo da maior diferença $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i - b_v$ possível, aqui definido como 1, já que $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \in [0{,}1] \subseteq \mathbb{R}$ e $b_v \in [0, 1] \subseteq \mathbb{R}$.

Em ambos os casos, quando M participa das expressões, ele modifica o valor do limiar para cima na Expressão (3.29) e para baixo na expressão (3.30). Na expressão (3.29), a soma $b_v + M_1$ tal que $M = 1$ e $b_v = [0{,}1] \subseteq \mathbb{R}$ aumenta o valor de b_v enquanto na expressão (3.30) a subtração $b_v - M_1$, tal que $M = 1$ e $b_v = [0{,}1] \subseteq \mathbb{R}$ diminui o valor de b_v. O efeito disso é que M empurra o valor do limiar b_v para o limite ou para fora da amplitude de $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i = [0{,}1] \subseteq \mathbb{R}$, relaxando as restrições sempre que um dado elemento $\boldsymbol{x}_i$ não está em um nó v, ou seja, sempre que $z_{iv} = 0$.

Nos exemplos a seguir, considere $(b_v)_{minimum} = 0.001$, onde 0.001 pudesse ser o menor valor ainda maior que 0, até que o uso da inequação seja justificada mais adiante com a introdução do termo ϵ na equação (3.34).

Note na expressão (3.29) que, por mais que $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i$ seja muito grande, como 1, e b_v seja um valor muito pequeno, como 0.001, o que obviamente tornaria falso $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i < b_v = 1 < 0.001$, adicionando $b_v + M_1$ sendo $M_1 = 1$, a expressão $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i < b_v + M_1 = 1 < 0.001 + 1 = 1 < 1.001$ passa a ser verdadeira.

A mesma coisa ocorre de forma oposta na expressão (3.30). Ainda que $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i$ seja o menor valor possível, como $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i = 0$ e $b_v = 1$ sendo um valor grande, tornando assim a expressão $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \geq b_v = 0 \geq 1$ falsa, mas se M_2 for subtraído de b_v na expressão $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \geq b_v - M_2 = 0 \geq 1 - 1 = 0 \geq 0$, sendo $M_2 = 1$, ela também passa a ser verdadeira. Em outras palavras, quando M participa da expressão, a

restrição é relaxada e, por isso, não tem efeito prático sobre a solução, enquanto que quando M é anulado por $(1 - z_{iv})$ quando $z_{iv} = 1$, a expressão resultante, seja ela $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i < b_v$ ou $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \geq b_v$, promove efetivamente a separação dos elementos no nó v.

Neste caso, onde o valor de $z_{iv} = 1$, as restrições podem ser redefinidas, tornando-as mais simples:

$$\boldsymbol{a}_m^\intercal \boldsymbol{x}_i < b_m + (1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_L(v), \tag{3.31}$$

$$\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \geq b_m - (1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_R(v) \tag{3.32}$$

Nos exemplos anteriores acerca da inequação (3.29) o valor mínimo assumido para b_v foi propositalmente maior que 0 porque assumir $b_v = 0$ em $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i < b_v + M$, onde $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i = 1$ e $M = 1$, ainda não torna verdadeira a expressão (3.29), tal como $1 < 0 + 1 = 1 < 1$ é falsa.

Cientes desta questão e da impossibilidade do uso de inequações em resolvedores de problemas de otimização inteira mista, Bertsimas e Dunn (2017) adéquam a expressão (3.32) adicionando um valor mínimo ϵ aos dois termos envolvidos na comparação $(<)$ e alterando-a de $(<)$ para $(\leq)$.

Para que ϵ não seja tão pequeno a ponto de causar instabilidades ao classificador, eles sugerem que ϵ seja diferente para cada atributo p e que seja máximo, porém menor ou igual a menor diferença entre dois valores x_{ip} para um mesmo atributo p, portanto

$$\epsilon = min\{x_{(i+1)p} - x_{ip} \quad | \quad x_{(i+1)p} \neq x_{ip}, \quad i = \{1,2,\ldots,n-1\}\}. \tag{3.33}$$

Entretanto, neste trabalho assumiu-se o menor valor de ϵ dentre todos menores valores ϵ_p de todos os atributos, onde $p = 1,2,\ldots,P$. Então, se $\epsilon_1 = 0.001$ para um atributo p_1 e $\epsilon_2 = 0.002$ para um atributo p_2, assume-se $\epsilon_p = 0.001$ para todo $p \in \{1,2,\ldots,P\}$.

Definido um único valor para ϵ, a expressão (3.31) é reescrita, tal como a expressão (3.34)

$$\boldsymbol{a}_m^\intercal(\boldsymbol{x}_i + \epsilon) \leq b_m + (1+\epsilon)(1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_L(v); \tag{3.34}$$

$$\boldsymbol{a}_m^\intercal \boldsymbol{x}_i \geq b_m - (1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_R(v). \tag{3.35}$$

Assim, seja $\epsilon = 0.001$ um valor suficientemente pequeno e $x_{ip} = 1$, a expressão (3.34) é adequadamente atendida, seja para um valor mínimo de $b_v = 0$, como

$$\begin{aligned} &\boldsymbol{a}_m^\intercal(\boldsymbol{x}_i + \epsilon) \leq b_m + (1 + \epsilon)(1 - z_{iv}) \\ &= 1(1 + 0.001) \leq 0 + (1 + 0.001)(1 - 0) \\ &= 1.001 \leq 1.001, \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_L(v), \end{aligned} \tag{3.36}$$

quanto para um valor máximo de $b_v = 1$, como

$$\begin{aligned} &\boldsymbol{a}_m^\intercal(\boldsymbol{x}_i + \epsilon) \leq b_m + (1 + \epsilon)(1 - z_{iv}) \\ &= 1(1 + 0.001) \leq 1 + (1 + 0.001)(1 - 0) \\ &= 1.001 \leq 2.001, \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_L(v), \end{aligned} \tag{3.37}$$

pois a tanto a diferença $[\boldsymbol{a}_m^\intercal(\boldsymbol{x}_i + \epsilon)] - [b_m + (1 + \epsilon)(1 - z_{iv})]$, quanto a diferença $[\boldsymbol{a}_m^\intercal \boldsymbol{x}_i] - [b_m + 1(1 - z_{iv})]$, continuam sendo iguais a 1, para $b = 0$, $\boldsymbol{a}_m^\intercal \boldsymbol{x}_i = 1$ e $M = 1$.

O estudo, implementação e aplicação da Árvore de Classificação Ótima usando Otimização Inteira Mista, modelada por Bertsimas e Dunn (2017), permitiu observar resultados de treinamento que geravam uma árvore que atendia todas as restrições definidas em seu artigo, mas, ainda assim, eventualmente possuía nós galho $v \in V_b$ que tinham apenas uma ramificação. Também foi possível observar que estes nós, de fato, possuíam um limiar b_v definido tal como na Restrição (3.23). Apesar disso, o nó $v \in V_b$ não criava efetiva separação dos dados que chegavam até ele, pois, apesar do valor do seu limiar b_v estar dentro da amplitude definida, b_v ainda podia ser, e eventualmente era, superior ao maior valor ou inferior ao menor valor x_{ip} de um dado atributo p de todos elementos x_i que passaram pelo nó v. Além disso, como o limiar b_v de um nó galho $v \in V_b$ ora era inferior, ora superior, aos valores x_{ip} em v, então ora o nó v se ramificava somente para a esquerda, ora somente para a direita.

Para impedir que a árvore resultante tivesse nós galhos $v \in V_b$ com apenas uma ramificação e ainda garantir que a última ramificação antes de um nó folha

$v \in V_f$ ocorresse sempre à direita do nó $v \in V_b$, foi necessário definir que, se um nó $v \in V_b$ faz divisão, deve haver ao menos dois nós folhas $g(v_L) \in V_f$ e $g(v_R) \in V_f$ tanto na subárvore esquerda, quanto direita de $v \in V_b$ respectivamente. Além disso, os nós folha $g(v_L)$ e $g(v_R)$ devem ambos necessariamente ser os nós folha mais à direita de v_L e v_R respectivamente.

Para modelar estas definições, foram adicionadas as restrições:

$$d_v \leq l_{g(v_L)}, \quad \forall v \in V_b, \quad g(v_L) \in V_f, \tag{3.38}$$

$$d_v \leq l_{g(v_R)}, \quad \forall v \in V_b, \quad g(v_R) \in V_f, \tag{3.39}$$

onde $d_v = 1$ se o nó v aplica divisão aos dados que passam por ele ou $d_v = 0$, caso contrário, conforme já descrito anteriormente, e $l_{g(v)} = 1$ se o descendente mais à direita de $v \in V_b$ possui algum elemento $\boldsymbol{x}_i$, caso contrário, $l_{g(v)} = 0$. Sendo assim, a Restrição (3.38) se aplica à subárvore esquerda, v_L, enquanto a restrição (3.39) se aplica à subárvore direita, v_R, ambas de $v \in V_b$.

De acordo com Bertsimas e Dunn (2017), o objetivo do problema é minimizar o número total de classificações incorretas da árvore. Para isso, foi definido que a classificação incorreta de cada elemento $\boldsymbol{x}_i$ é penalizada com um custo 1 e cada classificação correta com custo 0.

O **número de classificações incorretas**, $R(v)$, nada mais é que o número de elementos $\boldsymbol{x}_i$ em v cuja classe y_i é diferente da classe que rotula v. Para modelar a restrição do número de classificações incorretas primeiro são utilizadas as variáveis n_{kv}, que contém o número de elementos $\boldsymbol{x}_i$ de cada classe k no nó v, e n_v que contém o número de elementos em um nó v, conforme as Expressões

$$n_{kv} = \sum_{i \in X^k} z_{iv}, \quad k = 1, \ldots, K, \quad v \in V_f, \quad X^k \subseteq X = \{i | y_i = k\}, \tag{3.40}$$

$$n_v = \sum_{i=1}^{n} z_{iv}, \quad \forall v \in V_f \tag{3.41}$$

onde, $z_{iv} = 1$ se um elemento $\boldsymbol{x}_i$ está em v e $z_{iv} = 0$ caso contrário. Observe que, na Equação (3.40), a variável n_{kv} obtém separadamente o número de elementos $\boldsymbol{x}_i$ de cada classe k em v, pois, para cada valor de $k = 1, \ldots, K$, soma-se apenas os elementos $\boldsymbol{x}_i$ cuja classe y_i é igual à k, isto é,$x_i \in X^k \subseteq X = \{i | y_i = k\}$.

A classe que rotula cada nó folha $v \in V_f$ e, por sua vez, classifica todos os elementos contidos em v só é definida em cada solução do problema, mas pode ser descrita matematicamente desde já como

$$c_v = \arg \max_{k=1,\dots,K} \{n_{kv}\} \tag{3.42}$$

onde c_v é a classe de maior número de elementos $\boldsymbol{x}_i$ contidos em v. Assim, o número de classificações incorretas pode ser obtido pela diferença entre o número total de elementos $\boldsymbol{x}_i$ em v e o número de elementos de classe majoritária em v.

$$R(v) = n_v - \max_{k=1,\dots,K} \{n_{kv}\} = \min_{k=1,\dots,K} \{n_v - n_{kv}\}. \tag{3.43}$$

Além disso, para restringir que cada nó folha possua apenas um rótulo de classe, foi definida a variável binária $c_{kv} = \mathbb{1}\{c_v = k\}$, onde $c_{kv} = 1$ se uma classe k rotula um dado nó v e $c_{kv} = 0$ se uma classe k não rotula um dado nó v, mapeando assim, qual classe k rotula cada nó v. Desta forma, a Equação (3.44) restringe que se v é um nó folha ($l_v = 1$), então a soma de todos valores binários nos índices das classes k que rotulam v, deve ser estritamente 1. Por outro lado, se v não é um nó folha ($l_v = 0$), então a soma de todo os valores binários nos índices das classes k que rotulam v deve ser estritamente 0.

$$\sum_{k=1}^{K} c_{kv} = l_v, \quad \forall v \in V_f. \tag{3.44}$$

A partir da Equação (3.43) que determina o número de classificações incorretas em um dado nó v, é possível perceber que $R(v) \in \{0, \dots, n\}$. Portanto, para elementos $\boldsymbol{x}_i$ contidos em um dado nó v, o número de classificações incorretas está restrito entre 0 e n, onde $R(v) = 0$ significa que não há elementos $\boldsymbol{x}_i$ classificados incorretamente no nó v enquanto que $R(v) = n$ significa que todos os n elementos estão em um dado nó v, mas não deveriam estar, ou seja, estão classificados incorretamente.

Para linearizar isto, Bertsimas e Dunn (2017) definiram as seguintes restrições

$$R(v) \geq n_v - n_{kv} - M(1 - c_{kv}), \quad k = 1, \dots, K, \quad \forall v \in V_f, \tag{3.45}$$

$$R(v) \leq n_v - n_{kv} + M c_{kv}, \quad k = 1, \dots, K, \quad \forall v \in V_f, \tag{3.46}$$

$$R(v) \geq 0, \quad \forall v \in V_f, \tag{3.47}$$

onde novamente M é uma constante grande o suficiente para afrouxar as restrições (3.45) e (3.46), dependendo do valor de c_{kv}. De forma análoga às expressões (3.29) e (3.30), o objetivo de M nas expressões (3.45) e (3.46) é empurrar o resultado do lado direito da inequação para fora da amplitude de $R(v)$ afrouxando a restrição, contudo tanto a Restrição (3.45) quanto a Restrição (3.46) se aplicam sobre todas as K possíveis classes, portanto o afrouxamento de tais restrições está condicionado à variável c_{kv}.

Assim, na Restrição (3.45) aplica-se o afrouxamento somente às classes diferentes da classe que rotula o nó v, pois quando $c_{kv} = 0$, $M(1 - c_{kv}) = M$. Já na Restrição (3.46), por outro lado, aplica-se o afrouxamento somente à classe que rotula o nó v, pois quando $c_{kv} = 1$, $Mc_{kv} = M$. Por fim, a Restrição (3.47) limita que o número de classificações incorretas seja maior ou igual a zero. Assim, as Expressões 3.45, 3.46 e (3.47) delimitam a amplitude de $R(v)$ dentro de $\{0, \ldots, n\}$.

Finalmente, o **número de classificações incorretas da árvore** é a soma dos números de classificações incorretas de todos os nós folha da árvore, dada por $\sum_{v \in V_f} R(v)$, enquanto que a complexidade é a quantidade de nós galho da árvore, descrita por $\sum_{v \in V_b} d_v$, conforme definido na abordagem CART (BREIMAN et al., 2017).

Como cada conjunto de dados possível para a criação de uma árvore de classificação tem características individuais, com isso o efeito da constante α poderia ser diferente, dependendo do conjunto de dados de treinamento em questão. Para resolver isso, Bertsimas e Dunn (2017), baseados em Breiman et al. (2017), optaram por normalizar o número de classificações incorretas sobre a linha de base da acurácia, $\hat{R}(v)$, obtida através da classe associada ao maior número de elementos dentro do conjunto de dados. Desta forma, a **função objetivo do problema** pode ser escrita como

$$min \quad \frac{1}{\hat{R}} \sum_{v \in V_f} R(v) + \alpha \sum_{v \in V_b} d_v. \tag{3.48}$$

Observe que a função objetivo representa o equilíbrio entre a acurácia da árvore, dada pelos termos $\frac{1}{\hat{R}} \sum_{v \in V_f} R(v)$ e a sua complexidade, representada por $\alpha \sum_{v \in V_b} d_v$.

O modelo completo, então, é dado por:

$$
\begin{aligned}
min \quad & \frac{1}{\hat{R}} \sum_{v \in V_f} R(v) + \alpha \sum_{v \in V_b} d_v, \\
\\
s.\ a. \quad & \sum_{p=1}^{P} a_{pv} = d_v, \quad a_{pv} \in \{0,1\}, \quad p = 1, \ldots, P, \quad \forall v \in V_b, \\
& 0 \leq b_v \leq d_v, \quad \forall v \in V_b, \\
& d_v \leq d_{p(v)}, \quad \forall v \in V_b V \setminus \{1\}, \\
& z_{iv} \leq l_v, \quad v \in V_b, \\
& \sum_{i=1}^{n} z_{iv} \geq n_{min} l_v, \quad v \in V_f, \\
& \sum_{v \in V_f} z_{iv} = 1, \quad i = 1, \ldots, n, \\
& \boldsymbol{a}_m^{\intercal} \boldsymbol{x}_i < b_m + M_1(1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_L(v), \\
& \boldsymbol{a}_m^{\intercal} \boldsymbol{x}_i \geq b_m - M_2(1 - z_{iv}), \quad \forall v \in V_f, \quad i = 1, \ldots, n, \quad \forall m \in A_R(v), \\
& d_v \leq l_{g(v_L)}, \quad \forall v \in V_b, \quad g(v_L) \in V_f, \\
& d_v \leq l_{g(v_R)}, \quad \forall v \in V_b, \quad g(v_R) \in V_f, \\
& n_{kv} = \sum_{i \in X^k} z_{iv}, \quad k = 1, \ldots, K, \quad v \in V_f, \quad X^k \subseteq X = \{i | y_i = k\}, \\
& n_v = \sum_{i=1}^{n} z_{iv}, \quad \forall v \in V_f, \\
& c_v = \arg \max_{k=1,\ldots,K} \{n_{kv}\}, \\
& \sum_{k=1}^{K} c_{kv} = l_v, \quad \forall v \in V_f, \\
& R(v) \geq n_v - n_{kv} - M(1 - c_{kv}), \quad k = 1, \ldots, K, \quad \forall v \in V_f, \\
& R(v) \leq n_v - n_{kv} + M c_{kv}, \quad k = 1, \ldots, K, \quad \forall v \in V_f, \\
& R(v) \geq 0, \quad \forall v \in V_f.
\end{aligned}
\tag{3.49}
$$

De acordo com Bertsimas e Dunn (2017), este modelo é perfeitamente solucionável usando otimização inteira mista (MIO) em alguns minutos e sua complexidade está relacionada principalmente à variável z_{it} que é $n \times 2^D$. Para

conjuntos de dados mais complexos ou árvores de altura superiores a 4, o tempo de execução pode ser maior. Os parâmetros deste modelo são a altura máxima da árvore dada por D, a quantidade mínima de elementos para definir um nó folha, dada por n_{min}, e o parâmetro de complexidade α. Bertsimas e Dunn (2017) ainda apresentam métodos mais efetivos para melhorar estes parâmetros em seu artigo *Optimal Classification Trees*.

O modelo OCT-MIO implementado neste trabalho está disponível em https://github.com/jaderfigueredo/tree_mio.

3.3 Métricas de Desempenho

Existem algumas métricas que mostram os principais aspectos de desempenho de modelos de classificação. Essas métricas baseiam-se na proporção de acertos e erros do modelo para evidenciar diferentes aspectos de seu desempenho na tarefa de predição de classes.

3.3.1 Matriz de confusão

Matriz de confusão é uma representação tabular $M_{ij} \in \mathbb{N}$, $i,j = \{1,2,\ldots,K\}$ que apresenta todas as relações de acertos e erros de predição do modelo para cada classe $k = \{1,2,\ldots,K\}$ do problema. Tanto as linhas (i) quanto as colunas (j) da matriz representam as K classes do problema, dispostas nas mesma ordem (GRANDINI; BAGLI; VISANI, 2020), conforme a Figura 6.

As linhas (i) correspondem às classes reais do problema e as colunas (j) correspondem às classes preditas. Isto posto, então cada cruzamento M_{ij} de linhas (i) e colunas (j) contém um valor que significa a quantidade de elementos da classe real i que foram preditos como sendo da classe j (GRANDINI; BAGLI; VISANI, 2020).

Desta forma, a diagonal principal $M_{ii}, i = \{1,2,\ldots,K\}$ contém os números de elementos preditos corretamente para cada classe, uma vez que a diagonal principal é formada exatamente pelos cruzamentos de cada linha com sua coluna correspondente. De forma inversa, quaisquer cruzamentos fora da diagonal principal,

isto é M_{ij}, para $i,j = \{1,2,\ldots,K\}$ e $i \neq j$, correspondem a uma predição incorreta do modelo, pois, se a linha é diferente da coluna relacionada em um cruzamento, significa que a classe predita (coluna) é diferente da classe real (linha).

Os conceitos: **verdadeiro positivo**, **verdadeiro negativo**, **falso positivo**, **falso negativo**, são conceitos relacionados apenas à problemas de 2 classes. Considere inicialmente um problema de predição, cujo objetivo é identificar quais elementos de um conjunto de dados pertencem a uma dada classe objetiva (1) e quais não (2). Os elementos pertencentes à classe 1 são denominados "positivos", enquanto quaisquer outros elementos não pertencentes à classe 1, isto é, pertencentes a 2, são denominados "negativos". Já o termo "verdadeiro" refere-se a cada elemento classificado corretamente pelo modelo, enquanto "falso" refere-se a cada elemento classificado incorretamente, independentemente se eles forem positivos (1) ou negativos (2) (GRANDINI; BAGLI; VISANI, 2020).

Desta forma, em uma matriz de confusão ($M_{ij} \in \mathbb{N}, i,j = \{1,2\}$), como na Figura 6, verdadeiros positivos são os elementos cujo modelo classificou como sendo positivos (classe 1) e, de fato, eles eram da classe 1 no conjunto de dados. Já verdadeiros negativos são os elementos cujo modelo classificou como sendo negativos (classe 2) e, de fato, eles eram da classe 2 (GRANDINI; BAGLI; VISANI, 2020). Por outro lado, falsos positivos são os elementos cujo modelo classificou de forma incorreta como sendo positivos (1), mas, na verdade, eles eram negativos (2) no conjunto de dados. Já falsos negativos são os elementos cujo modelo classificou de forma incorreta como sendo negativos (2), mas, na verdade, eles eram positivos (1) no conjunto de dados (GRANDINI; BAGLI; VISANI, 2020).

Já em um problema de três ou mais classes, como na Figura 7, os conceitos verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo só podem ser analisados olhando uma classe de cada vez.

Isto posto, ao fixar uma classe l, onde $l = \{1,2,\ldots,K\}$, como positiva, todas as outras classes k diferentes de l serão consideradas como uma única classe negativa. Na matriz de confusão, a linha M_{l*}, correspondente à classe fixada l, conterá todos os elementos positivos, enquanto as demais linhas, conterão todos os elementos negativos.

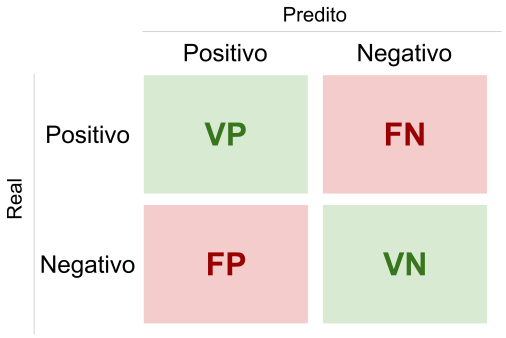

Figura 6 – Matriz de Confusão com 2 classes

Dito isso, o número de verdadeiros positivos sempre estará posicionado na célula M_{ll}, enquanto os verdadeiros negativos são todos aqueles que não pertencem a classe l e de fato não foram classificados como l, ou seja, todas as células M_{ij}, em que $i,j = \{1,2,\ldots,K\}$, $i \neq l$ e $j \neq l$). Já os falsos positivos são correspondidos pelos números contidos nas células M_{il}, para $i = \{1,2,\ldots,K\}$, e $i \neq l$, enquanto os falsos negativos são correspondidos pelos números contidos nas células M_{lj}, para $j = \{1,2,\ldots,K\}$, e $j \neq l$ (GRANDINI; BAGLI; VISANI, 2020; PAGANO et al., 2023).

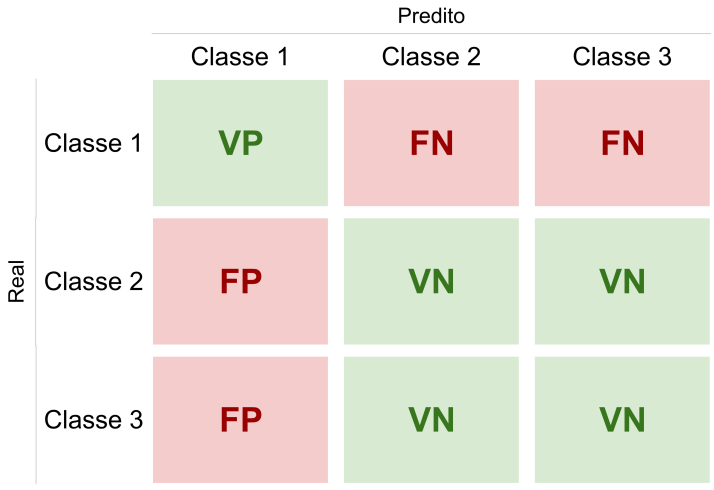

Figura 7 – Exemplo de Matriz de Confusão Multiclasse

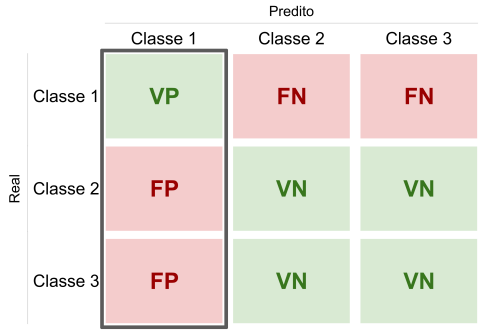

Figura 8 – Componentes da Precisão na Matriz de Confusão

A partir do entendimento das relações da matriz de confusão é possível obter diversas métricas que representam diferentes aspectos de desempenho de modelos na tarefa de classificação. Para isto, os conceitos de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos são essenciais.

3.3.2 Precisão

Precisão (*precision*, em inglês) ou ainda valor preditivo positivo é uma métrica calculada considerando uma única classe como positiva de cada vez. Dito isso, precisão é a razão entre o número de elementos classificados como verdadeiros positivos e todos os elementos classificados como positivos, sejam eles verdadeiros ou falsos, conforme:

$$Precisão = \frac{VP}{VP + FP}. \tag{3.50}$$

Portanto, quanto mais próximo de 1, melhor a precisão de um modelo de classificação, e quanto mais próximo de 0, pior. Note que na expressão não existe valores de FN e VN, isto significa que a precisão concentra-se na capacidade do modelo em não classificar classes negativas como sendo da classe positiva (GRANDINI; BAGLI; VISANI, 2020).

Na matriz de confusão da Figura 8, a precisão é a razão entre o cruzamento referente aos verdadeiros positivos (VP) e a soma de todos os cruzamentos da mesma coluna (VP+FP).

3.3.2.1 Macro Precisão

Macro Precisão nada mais é que a média aritmética de todas as precisões (Seção 3.3.2) de um modelo para cada classe do problema, utilizada para aferir a precisão de um modelo em um problema multiclasse através de um único valor (GRANDINI; BAGLI; VISANI, 2020), conforme:

$$Macro\ Precisão = \frac{\sum_{j=1}^{K} Precisão_j}{K}. \tag{3.51}$$

3.3.3 Sensibilidade / Revocação

Sensibilidade, Revocação (*recall*, em inglês) ou ainda taxa de verdadeiros positivos, também é uma métrica calculada considerando uma única classe como positiva de cada vez. Dito isso, Sensibilidade é a razão entre o número de elementos classificados como verdadeiros positivos e a soma tanto dos verdadeiros positivos quanto dos falsos negativos:

$$Sensibilidade = \frac{VP}{VP + FN}. \tag{3.52}$$

Portanto, quanto mais próximo de 1, melhor a Sensibilidade de um modelo de classificação, e quanto mais próximo de 0, pior. Isto significa que a Sensibilidade concentra-se unicamente na classe objetivo, pois leva em consideração tanto os acertos quantos os erros de predição apenas da classe positiva, em outras palavras, dado que o elemento é da classe l, a Sensibilidade estima a probabilidade deste elemento ser predito corretamente (GRANDINI; BAGLI; VISANI, 2020).

Observe, que a Sensibilidade é uma métrica que não está interessada em quantos elementos da classe negativa foram preditos incorretamente, isto significa que uma boa Sensibilidade preocupa-se apenas em garantir que a classe positiva seja classificada corretamente, ao passo que tolera que quaisquer elementos da classe negativa sejam previstos incorretamente (GRANDINI; BAGLI; VISANI, 2020).

Na matriz de confusão da Figura 9, a Sensibilidade é a razão entre o cruzamento referente aos verdadeiros positivos (VP) e a soma de todos os elementos da mesma linha (VP+FN).

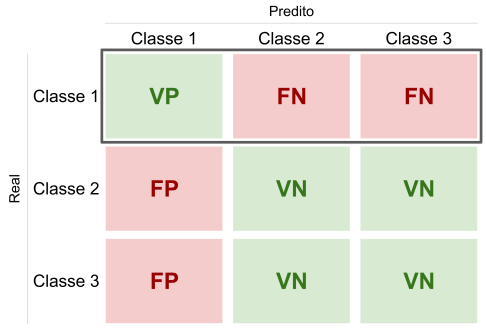

Figura 9 – Componentes da Sensibilidade na Matriz de Confusão

3.3.3.1 Macro Revocação

Macro Revocação, como é conhecido em português ou ainda *Macro Recall*, em inglês, nada mais é que a média aritmética de todas as Sensibilidades (Seção 3.3.3) de um modelo para cada classe do problema, utilizada para aferir a Sensibilidade de um modelo em um problema multiclasse através de um único valor (GRANDINI; BAGLI; VISANI, 2020), conforme:

$$Macro\ Revocação = \frac{\sum_{i=1}^{K} Sensibilidade_i}{K}. \tag{3.53}$$

3.3.4 Acurácia

A definição de Acurácia pode variar dependendo do contexto. No contexto de classificação por modelos de aprendizado de máquina, o termo Acurácia refere-se à exatidão, que indica a taxa de predições corretas feitas pelo modelo em relação ao total de predições realizadas, ou seja, o quão similar as predições do modelo são em relação ao conjunto original de dados (GRANDINI; BAGLI; VISANI, 2020).

Acurácia é uma métrica simples, dada pela razão entre o número de predições corretas e o total de todas as predições, isto é,

$$Acurácia = \frac{Total\ de\ Acertos}{Total\ de\ Predições} = \frac{VP + VN}{VP + VN + FP + FN}, \tag{3.54}$$

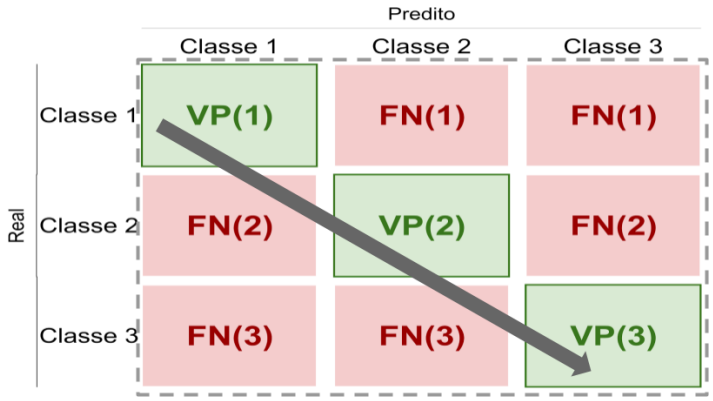

Figura 10 – Componentes da Acurácia (Simples) na Matriz de Confusão Multiclasse

para um problema de 2 classes conforme Figura 6, ou ainda

$$\begin{aligned} Acurácia &= \frac{Total\ de\ Acertos}{Total\ de\ Predições} = \frac{\sum_{i=1}^{K} M_{ii}}{n} \\ &= \frac{VP_1 + VP_2 + \ldots + VP_K}{(VP_1 + FN_1) + (VP_2 + FN_2) + \cdots + (VP_K + FN_K)}, \end{aligned} \tag{3.55}$$

para um problema multiclasse, onde n é o total elementos preditos, conforme a Figura 10. A Acurácia tem valor 1 se o modelo acerta a predição de todos os elementos, e tem valor 0, caso erre todas.

A predição de qualquer elemento tem o mesmo peso no resultado, portanto a Acurácia é mais efetiva à medida que as proporções de cada classe no conjunto são mais equilibradas, ao passo que quanto mais desequilibradas as proporções entre as classes, menor a efetividade da medida da Acurácia, pois ela tende a esconder erros de classificação sobre as classes de menor proporção (GRANDINI; BAGLI; VISANI, 2020), todavia, existem métricas alternativas que minimizam essa desvantagem.

3.3.5 Acurácia Balanceada

Acurácia balanceada é uma métrica que avalia o desempenho preditivo do modelo considerando que o conjunto de dados possa ser composto por múltiplas

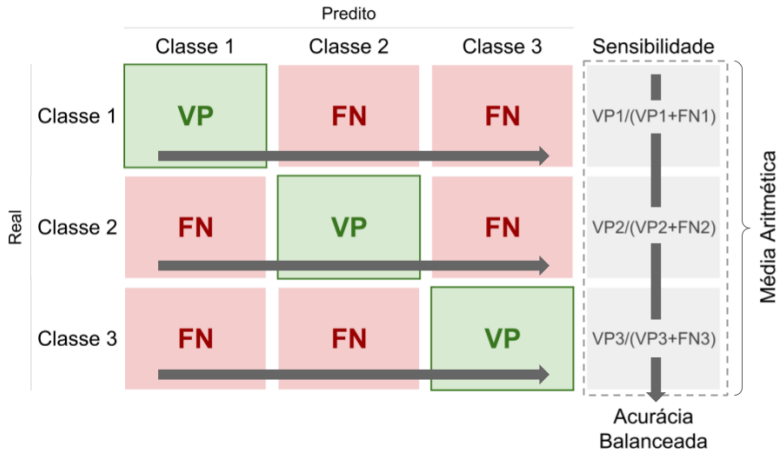

Figura 11 – Componentes da Acurácia Balanceada na Matriz de Confusão Multiclasse

classes e ainda desbalanceado, ou seja, a proporção de elementos de cada classe não seja equilibrada.

Ela fornece um único valor que, diferente da Acurácia Simples, é a média aritmética das taxas de verdadeiros positivos de cada classe, independentemente das proporções delas no conjunto de dados, isto é, a média aritmética das Sensibilidades obtidas a partir da matriz de confusão (Figura 11), conforme a Expressão (3.56) (GRANDINI; BAGLI; VISANI, 2020).

$$\begin{aligned} Acurácia\ Balanceada &= \frac{Soma\ das\ Sensibilidades}{Número\ de\ Classes} \\ &= \frac{(\frac{VP_1}{VP_1+FN_1}) + (\frac{VP_2}{VP_2+FN_2}) + \ldots + (\frac{VP_K}{VP_K+FN_K})}{K} \end{aligned} \tag{3.56}$$

O termo “balanceada” significa que todas as classes tem o mesmo peso, afetando igualmente a média das Sensibilidades, independentemente de suas proporções (GRANDINI; BAGLI; VISANI, 2020). Observe que, ainda que exista, por exemplo, apenas 1 elemento de uma classe positiva e 99 elementos de uma classe negativa em um conjunto de dados de 2 classes, se o modelo classificar todos os elementos como sendo da classe negativa, ele terá errado apenas 1 predição, porém,

enquanto a Acurácia simples

$$Acurácia\ simples = \frac{99}{100} = 0.99, \tag{3.57}$$

seria sutilmente afetada por apenas uma predição errada, a Acurácia Balanceada seria afetada de forma equivalente, tanto pela taxa de acertos da classe positiva quanto da classe negativa, diminuindo seu valor significativamente:

$$Acurácia\ balanceada = \frac{(\frac{0}{1} + \frac{99}{99})}{2} = \frac{1}{2} = 0.5. \tag{3.58}$$

Portanto, a Acurácia Balanceada mostra mais expressivamente quando há alguma falha de predição sobre quaisquer classes do problema.

A Expressão

$$\begin{aligned} Acurácia\ balanceada &= \frac{(\frac{37}{40} + \frac{35}{40} + \frac{10}{10} + \frac{2}{7} + \frac{1}{3})}{5} \\ &= \frac{(0{,}925 + 0{,}875 + 1{,}000 + 0{,}286 + 0{,}333)}{5} \\ &= \frac{3{,}419}{5} = 0{,}684 \end{aligned} \tag{3.59}$$

exemplifica um suposto problema de 5 classes contendo tanto classes com Sensibilidade acima de 0.7, quanto abaixo de 0.3, onde são 85 predições corretas contra apenas 15 predições incorretas em um total de 100 predições. É possível perceber que as Sensibilidades para as classes 4 e 5 foram muito baixas, puxando o valor final para baixo, assim ela reflete o desempenho geral do modelo, independentemente quantas classes e em quais proporções elas componham o problema.

O fato de a Acurácia Balanceada evidenciar falsos positivos em quaisquer proporções pode ser uma vantagem, principalmente na abordagem de problemas críticos, onde quaisquer falhas podem ter consequências significativas, principalmente quando se trata de pessoas (GRANDINI; BAGLI; VISANI, 2020).

3.3.6 Pontuação F1

Pontuação F1 simples (*F1-Score*, em inglês) é uma métrica, definida para $K = 2$, também extraída da matriz de confusão, para avaliar o desempenho preditivo de um modelo de classificação através da média harmônica entre precisão e Sensibilidade. A pontuação F1 fornece um único valor que varia entre 0 e 1, onde 0 é o pior valor, que pode acontecer se o modelo tem $Precisão = 0$ ou $Sensibilidade = 0$, enquanto 1 é o melhor valor que acontece quando $Precisão = 1$ e $Sensibilidade = 1$ e, portanto, $Pontuação\ F1 = 1$ (GRANDINI; BAGLI; VISANI, 2020), formalmente

$$Pontuação\ F1 = \frac{2}{(\frac{1}{Precisão} + \frac{1}{Sensibilidade})} = \frac{2 \times Precisão \times Sensibilidade}{Precisão + Sensibilidade}. \quad (3.60)$$

Contudo, pontuação F1, precisão e Sensibilidade são métricas obtidas considerando apenas 2 classes por vez, portanto, para problemas de múltiplas classes é necessário considerar então a média da precisão e a média da Sensibilidade conforme a definição de Macro Pontuação F1 a seguir.

3.3.6.1 Macro Pontuação F1

Macro Pontuação F1 é uma variação da métrica Pontuação F1 (3.3.6), cuja diferença está na consideração da média aritmética das precisões (Macro Precisão 3.3.2.1) e da média aritmética das Sensibilidades (Macro Revocação 3.3.3.1). Assim, de maneira análoga à Acurácia Balanceada (3.3.5), a Macro Pontuação F1 atribui peso igual a todas as classes, independente da sua proporção, refletindo assim os efeitos de predições incorretas, mesmo apenas em classes minoritárias (GRANDINI; BAGLI; VISANI, 2020).

$$Macro\ Pontuação\ F1 = 2 \times \left(\frac{Macro\ Precisão \times Macro\ Revocação}{Macro\ Precisão^{-1} + Macro\ Revocação^{-1}} \right)$$

$$= 2 \times \left(\frac{\left(\frac{\sum_{i=1}^{K} Precisão_i}{K} \right) \times \left(\frac{\sum_{i=1}^{K} Sensibilidade_i}{K} \right)}{\left(\frac{1}{\frac{\sum_{i=1}^{K} Precisão_i}{K}} \right) + \left(\frac{1}{\frac{\sum_{i=1}^{K} Sensibilidade_i}{K}} \right)} \right)$$

Portanto, a Macro Pontuação F1 indica que, quanto melhor o desempenho do modelo sobre todas as classes, mais próximo de 1 será seu valor, e quanto pior, mais próximo de 0 (GRANDINI; BAGLI; VISANI, 2020).

Estas são algumas das métricas mais utilizadas na avaliação de modelos de aprendizado supervisionado para classificação. Existem métricas ainda mais ajustadas à cada configuração de conjunto de dados, criticidade de problema e outras variações, mas as métricas apresentadas neste trabalho já permitem uma análise comparativa geral para diversos modelos de classificação e diversos conjuntos de dados.

3.4 Validação Cruzada K-Fold

K-Fold Cross-Validation (**Validação Cruzada K-Dobras**, em tradução direta), ou simplesmente *K-Fold*, é uma técnica utilizada para avaliação de modelos de aprendizado de máquina. Seus objetivos principais são promover uma avaliação mais confiável do modelo, reduzir o *overfitting* e permitir um melhor aproveitamento do conjunto de dados para treinamento e teste (KOHAVI et al., 1995).

Quanto mais elementos do conjunto original são utilizados para treinamento, melhor será o ajuste do modelo, pois ele pode aprender de forma mais abrangente as características subjacentes aos dados. No entanto, se o modelo for muito complexo em relação ao tamanho do conjunto de dados, ele pode se tornar suscetível ao *overfitting*, onde o modelo se ajusta tão bem aos dados de treinamento que perde a capacidade de generalizar soluções para novos dados. Por outro lado, se o conjunto de treinamento não for suficientemente grande, o desempenho do modelo será comprometido, pois ele não terá dados suficientes para aprender as características gerais do problema. Além disso, a divisão arbitraria do conjunto, tal como apenas dividi-lo em 1/2 ou 1/3, onde uma parte é utilizada para teste e as demais para treinamento, implica que o modelo não seja treinado com uma parte do conjunto de dados (KOHAVI et al., 1995).

Isto posto, o K-Fold consiste em dividir o conjunto de dados em K partições (ou "*folds*") de tamanhos similares, onde, a cada iteração, $K-1$ *folds* são utilizados

para treinamento e o *fold* restante é utilizada para teste do modelo. Portanto, as proporções dos conjuntos de treinamento e teste em relação ao conjunto total dos dados são dadas por $(K-1)/K$ para o conjunto de treinamento e $1/K$ para o conjunto de teste. Observe que, quanto maior o valor de K, maior a proporção do conjunto de treinamento e menor a proporção do conjunto de teste (KOHAVI et al., 1995).

A técnica consiste ainda em K iterações, onde, em cada iteração, o conjunto de teste é dado por uma partição X_k, onde $k = \{1, 2, \ldots, K\}$, e o conjunto de treinamento é dado pela união das partições restantes X_k, tal que $X_k = \bigcup_{j \neq k} X_j$. Assim, a cada iteração do K-Fold, o modelo é treinado e testado com diferentes conjuntos (KOHAVI et al., 1995), conforme ilustrado na Figura 12.

	Fold 1	Fold 2	Fold 3	...	Fold K
Iter. 1	Teste	Treino	Treino	Treino	Treino
Iter. 2	Treino	Teste	Treino	Treino	Treino
Iter. 3	Treino	Treino	Teste	Treino	Treino
...	Treino	Treino	Treino	Teste	Treino
Iter. K	Treino	Treino	Treino	Treino	Teste

Figura 12 – K-Fold Cross Validation, onde K=5

Os experimentos descritos a seguir foram estruturados com o objetivo de responder às questões de pesquisa, tanto gerais, quanto específicas. Neste sentido, os experimentos foram realizados utilizando-se a implementação de duas técnicas sob olhar comparativo, sendo a primeira o modelo CART e a segunda o modelo OCT-MIO, onde ambos os modelos foram utilizados para classificar os mesmos conjuntos de dados com a mesma variação de parâmetros, exceto apenas pelos parâmetros específicos de cada modelo. Em uma primeira seção, os dois modelos foram treinados com todos os dados de cada conjunto. Em uma segunda seção, eles foram treinados usando a técnica K-Fold, que particiona o conjunto de dados em uma porção de treinamento e outra de testes. Para ambos os casos, algumas das métricas mais comuns para classificação foram utilizadas para compará-los. Ao final, espera-se verificar o desempenho dos dois modelos em relação a cada uma das métricas analisadas, como também conhecer em quais configurações cada modelo

destaca-se melhor. Espera-se ainda analisar a interpretabilidade de cada modelo de forma empírica.

3.5 Teste de Wilcoxon

Teste de Wilcoxon (*Wilcoxon Test*) ou Teste dos Postos Sinalizados de Wilcoxon (*Wilcoxon Signed-Rank Test*) é um teste estatístico não paramétrico utilizado para determinar se a diferença entre duas séries pareadas advindas de um mesmo conjunto de valores ordinários (de médias, neste caso) é estatisticamente significativa (WILCOXON, 1992).

Para isso, o utiliza-se das somas das diferenças sinalizadas obtida à partir dos dados para determinar se aceita-se ou rejeita-se a hipótese nula, ou seja, a hipótese de que não há diferenças estatisticamente significativas entre as duas séries de valores (WILCOXON, 1992).

À partir de cálculos realizados sobre as séries pareadas, obtêm-se o menor valor dentre as somas absolutas dos rankings positivos e negativos como sendo o valor estatístico. Em seguida compara-se este valor estatístico a um valor de referência na Tabela de Valores Críticos to Teste dos Postos Sinalizados de Wilcoxon. Se o valor estatístico for menor que o valor referente na Tabela, rejeita-se a hipótese nula, mas se o valor estatístico for maior ou não for possível compará-lo, aceita-se a hipótese nula (WILCOXON, 1992).

3.6 Apresentação dos Dados

Os dados utilizados neste estudos estão apresentados adiante em uma subseção introdutória sobre cada conjunto de dados, contendo os motivos pelos quais eles foram selecionados para este trabalho. Em seguida, a Seção 3.6.1.5 analisa os dados através das métricas.

3.6.1 Escolha dos dados

Dado que os modelos abordados tratam-se de classificadores baseados em aprendizado de máquina supervisionado, os dados escolhidos para classificação devem possuir rótulos de classes associados à eles.

Conhecendo as características dos modelos estudados, foi considerada a escolha de conjuntos de dados tabulares, isto é, um conjunto de n linhas e P colunas, onde n é o número de elementos (observações) e P é o número de atributos (*features*) destes elementos, os quais cada um possui um rótulo de classe $y_i \in \{1, 2, \ldots, K\}, i = \{1, 2, \ldots, n\}$ onde K é o número de rótulos de classe do problema.

Por se tratar de modelos de aprendizado de máquina interpretáveis, úteis em cenários críticos reais, a escolha de alguns conjuntos de dados também foi pautada em conjuntos reais, cujas decisões tomadas sobre eles podem ser críticas, com exceção do conjunto de dados *Iris Species* 3.6.1.1, que foi pautada com intuito exploratório. Os conjuntos de dados a seguir estão identificados entre parênteses na frente de seus respectivos nomes. Para cada conjunto, também há um gráfico bidimensional gerado a partir dos 2 atributos que proporcionam uma visualização mais compreensível dos mesmos.

3.6.1.1 Iris Species (Iris)

Iris Species (FISHER, 1988) é um pequeno conjunto de dados, onde uma das classes é distintamente separada das demais e as outras duas possuem uma região de interseção de difícil separação, como é possível ver na Figura 13.

3.6.1.2 Human Stress Detection in and through Sleep (Stress)

Human Stress Detection in and through Sleep (LAAVANYA, 2022) é um conjunto de dados utilizado nos trabalho de Rachakonda et al. (2021) e Rachakonda et al. (2021) que classifica indivíduos em 5 níveis de estresse baseado em 8 atributos medidos durante e através do sono destes indivíduos. Cada nível de estresse denomina uma classe do problema. As 5 classes estão bem separadas no espaço dimensional dos atributos, porém há pequenas interseções de classes quando

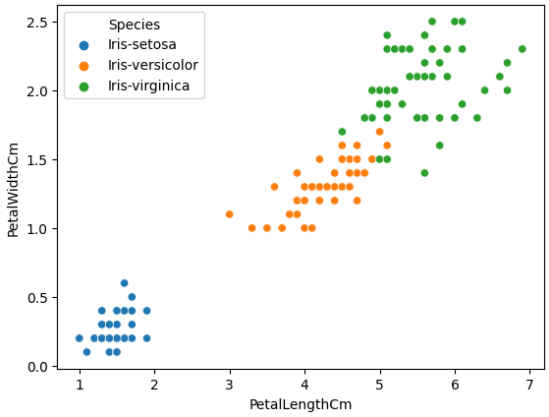

Figura 13 – Conjunto Iris Species - Gráfico de Dispersão

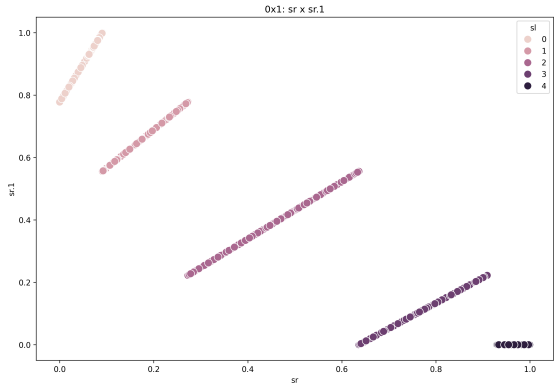

Figura 14 – Conjunto Stress - Gráfico de Dispersão

divididos por apenas uma dimensão, o que oferece desafios à modelos de árvores de classificação que aplicam apenas 1 divisão por vez, conforme a Figura 14.

3.6.1.3 Credit Card Approvals (Clean Data) (Credit)

O conjunto de dados *Credit Card Approvals (Clean Data)* (CORTINHAS, 2022) trata-se de uma variação do conjunto original *Credit Approval* (QUINLAN, 1987) cujos dados já passaram pelo processo de "limpeza". Os dados tratam-se de um conjunto de indivíduos cujo o rótulo de classe atribuído é "sim", caso tenha sido concedido crédito bancário ao indivíduo ou "não", caso contrário. Apesar do conjunto de dados utilizado não possuir dados faltantes e existir uma tendência entre os dados, ainda assim, não há uma separação clara entre as classes, conforme

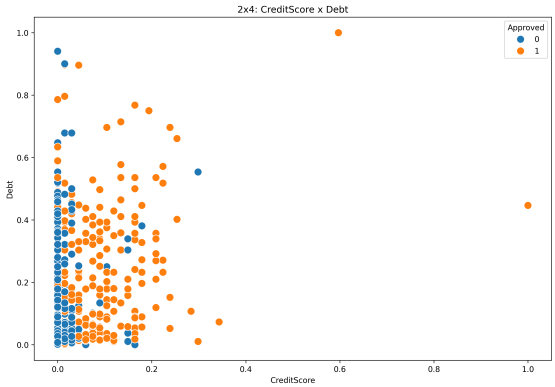

Figura 15 – Conjunto Credit - Gráfico de Dispersão

ilustrado pela Figura 15.

3.6.1.4 Students Dropout and Academic Success Dataset (Academic)

Sem dúvida este é o conjunto de dados mais desafiador dentre os demais para o propósito deste trabalho. Composto por milhares de amostras, dezenas de atributos e 3 classes, o conjunto de dados *Students Dropout and Academic Success* (KHALIL, 2022) foi selecionado para este estudo com o objetivo de predizer quais alunos estão potencialmente próximos de serem desmatriculados de seus respectivos cursos de graduação, uma vez que o conjunto de dados contém dados de desempenho de alunos durante a graduação. O conjunto está rotulado com 3 classes (*Dropout*, *Enrolled* e *Graduate*, traduzidas respectivamente como Desistiu, Matriculado e Graduado) de difícil separação no espaço de atributos, sendo ainda mais desafiador em função das classes estarem bastante misturadas (Figura 16). A classificação de alunos de graduação é interessante, uma vez que instituições de ensino superior eventualmente precisam tomar decisões importantes sobre quais alunos podem ter predisposição em desistir dos estudos, quais alunos precisam de apoio acadêmico antes que seu desempenho esteja comprometido ou, ainda, sobre quais alunos selecionar em um eventual processo de desligamento, por exemplo para realocação de vagas e recursos.

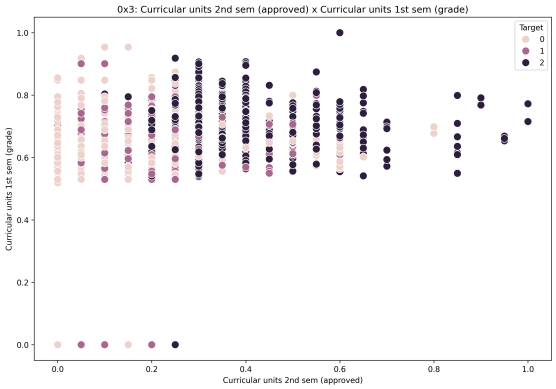

Figura 16 – Conjunto Academic - Gráfico de Dispersão

3.6.1.5 Análise dos conjuntos de dados

A Tabela 1 apresenta os números de elementos, atributos e classes de cada conjunto de dados, enquanto a Figura 17 as proporções de cada conjunto de dados bem com a distribuição de suas classes.

Tabela 1 – Conjuntos de Dados: Número de elementos, atributos e classes

Conjunto	**N. elementos**	**N. atributos**	**N. classes**
Iris Species	150	4	3
Credit Approval (Clean Data)	690	15	2
Human Stress Detection in and Through Sleep	630	8	5
Students Dropout and Academic Success	4424	36	3

Tabela 2 – Conjunto de Dados: Proporções das classes

Conjunto	**1**	**2**	**3**	**4**	**5**
Iris	50 (33,33%)	50 (33,33%)	50 (33,33%)	-	-
Credit	383 (55,5%)	307 (44,5%)	-	-	-
Stress	126 (20%)	126 (20%)	126 (20%)	126 (20%)	126 (20%)
Academic	1421 (32,12%)	794 (17,95%)	2209 (49,93%)	-	-

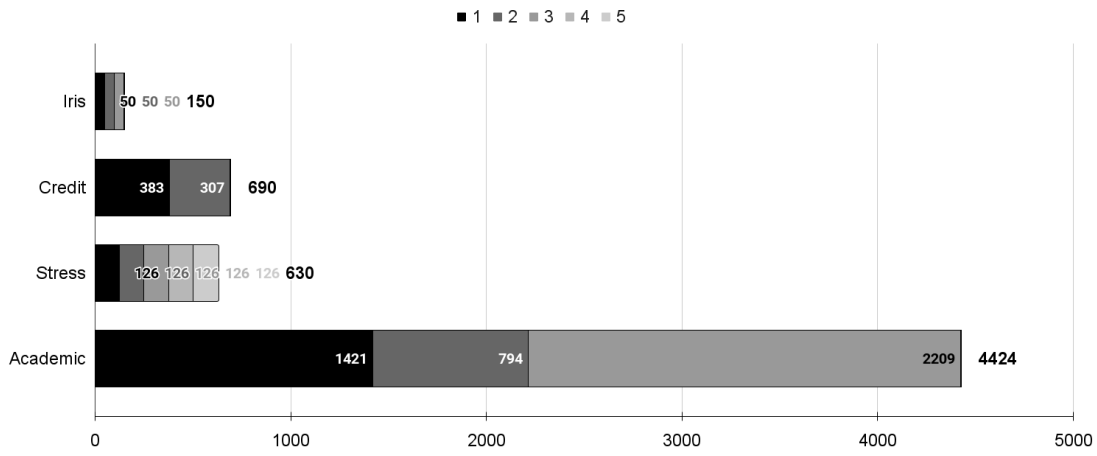

Figura 17 – Proporções dos Conjuntos de Dados

3.7 Pré-processamento dos Dados

Cada conjunto de dados tem características particulares de quantidade de elementos, quantidade e tipos de atributos, quantidade de classes e proporções de elementos de cada classe. Além disso, é comum que os conjuntos de dados apresentem desafios tais como dados faltantes, inconsistentes, ausentes, duplicados, entre outros. Todos estes fatores podem interferir negativamente no desempenho de modelos de classificação, por isso é importante que os dados sejam preparados antes de serem utilizados (CHU et al., 2016) e (RAO; SAIKRISHNA; SUPRIYA, 2023).

A etapa de preparação dos dados é bastante dispendiosa e pode levar um tempo significativo dentro do processo até a utilização destes dados, contudo a devida preparação dos dados melhora significativamente os resultados (CHU et al., 2016) e (RAO; SAIKRISHNA; SUPRIYA, 2023).

3.7.0.1 Limpeza dos dados

Limpeza é o nome dado a uma das etapa de preparação dos dados antes da sua utilização para aplicação de técnicas de análise, apresentação, investigação, aprendizado de máquina. Nesta etapa, eventuais falhas no conjunto, tais como duplicações, inconsistências, dados faltantes, entre outros, são encontradas e corri-

gidas utilizando técnicas apropriadas (CHU et al., 2016). Entretanto, os conjuntos obtidos já encontravam-se publicados de forma adequada para serem utilizados por modelos de aprendizado, apesar disso, ainda foi feita apenas uma análise visual preliminar em gráficos de dispersão e histogramas a fim de encontrar *outliers* e outras anomalias apenas para ratificação.

3.7.0.2 Transformação para dados numéricos

Alguns modelos de árvore de classificação suportam dados categóricos, assim como os modelos estudados, todavia dados numéricos são mais adequados para utilização por diversos motivos, desde a visualização de pontos no gráfico, até o atendimento de restrições de algoritmos de aprendizagem que trabalham com vetores de valores numéricos. Portanto, ainda que modelos de aprendizagem de máquina suportem dados categóricos, eles os mapeiam para valores numéricos (RAO; SAIKRISHNA; SUPRIYA, 2023). Neste trabalho, as implementações das árvores de classificação são modeladas baseadas em comparações de menor ($<$) ou maior-ou-igual ($\geq$), sendo necessário que todos os dados sejam numéricos, ainda que mapeados a partir de categorias.

Nesta etapa, portanto, todos os dados ainda não numéricos de todos os conjuntos foram transformados para dados numéricos, inclusive seus rótulos de classe. Eles foram transformados para valores inteiros, a partir do zero, seguindo a ordem alfabética dos mesmos, assim, independentemente do arranjo dos dados no conjunto, os valores numéricos serão sempre relacionados aos mesmos valores categóricos, conforme apresenta o Algoritmo 1.

3.7.0.3 Normalização dos dados

Normalização é uma das etapas de transformação dos dados durante o pré-processamento que melhora a qualidade dos dados, melhorando também o desempenho dos modelos de classificação. Ela consiste em mapear todos os valores de atributos do conjunto de dados dentro de uma mesma escala. Dentre os tipos de normalização existentes, este trabalho utilizou a Normalização Min-Max (*MMN0*), onde $Min = 0$ e $Max = 1$ (SINGH; SINGH, 2020), adequada à modelagem da Árvore de Classificação Ótima usando Otimização Inteira Mista (Seção 3.2), sem

Algoritmo 1: Transformar Atributos em Valores Numéricos

Entrada:

- X: Conjunto de dados, sendo uma matriz$[n][P+1]$

Saída: Conjunto de dados mapeados para valores numéricos

Início:

```
1 XNumérico ← matriz[n][P + 1];
2 para todo coluna ∈ X faça
3     se coluna não é numérica então
4         valoresUnicos ← obter valores únicos em coluna;
5         índices, valores ← indexar valores de coluna conforme ordem
            alfabética;
6         coluna ← índices;
7     XNumérico[coluna] ← coluna;
8 retorna XNumérico;
```

Fim:

quaisquer prejuízos à Árvore de Classificação e Regressão (Seção 3.1), conforme apresentado no Algoritmo 2.

3.7.0.4 Seleção e Eliminação de Atributos

Seleção de Atributos (*Feature Selection*, em inglês) é uma etapa do pré-processamento dos dados para classificação cujo objetivo de identificar um subconjunto de atributos com maior potencial de contribuição na classificação dos dados, segundo algum critério. A seleção de atributos permite que se possa trabalhar com um subconjunto de atributos, reduzindo assim a dimensionalidade do problema e o tempo de treinamento do modelo com o mínimo de perda de desempenho ou, ainda, aumentado o desempenho e reduzindo o *overfitting* do modelo (DASH; LIU, 1997; KHALID et al., 2023; TOUATI; SLIMANE; SAIDANI, 2024).

SelectKBests é a técnica utilizada neste trabalho para a seleção de P atributos mais relevantes para a classificação. Disponível como um módulo da biblioteca de código de programação SciKit Learning (PEDREGOSA et al., 2011) da linguagem *Python*, *SelectKBest* faz a seleção dos atributos baseada na análise

Algoritmo 2: Normalização de Atributos

Entrada:

- X: Conjunto de dados, sendo uma matriz$[n][P+1]$

Saída: Conjunto de dados normalizados

```
1  colunaDeÍndices ← X[1];
2  colunaDeClasses ← X[P + 1];
3  colunasParaNormalizar ← X[colunas de 2 até P];
4  XNormalizado ← matriz[n][P + 1];
5  para todo coluna ∈ colunasParaNormalizar faça
6      valorMínimo ← valor mínimo de coluna;
7      valorMáximo ← valor máximo de coluna;
8      se valorMínimo ≠ valorMáximo então
9          para todo valor ∈ coluna faça
10             coluna ←
                (valor − valorMìnimo)/(valorMáximo − valorMínimo);
11     XNormalizado ← adicionar coluna coluna no fim de XNormalizado;
12 Adicionar colunaDeÍndices no início de XNormalizado;
13 Adicionar colunaDeClasses no fim de XNormalizado;
14 retorna XNormalizado;
```

estatística univariada de cada um deles. Outro ponto positivo na técnica *SelectKBest* é que o subconjunto de atributos fornecidos por ela não sofre transformação, tal como acontece com a técnica PCA, ou seja, os valores dos P atributos selecionados são os mesmos do conjunto de dados de entrada, contribuindo para fins de interpretabilidade e explicabilidade (KHALID et al., 2023; TOUATI; SLIMANE; SAIDANI, 2024).

Os gráficos da Figura 18 destacam os 5 melhores atributos selecionados pela técnica *SelectKBest* para todos os conjuntos de dados.

Mesmo dentre os 5 atributos mais significativos, é possível perceber uma redução expressiva da pontuação dada pelo *SelectKBest* sobre contribuição de cada atributo, na maioria dos conjuntos de dados, sugerindo, portanto, que um subconjunto de P=5 atributos contenha os atributos mais prováveis à serem utilizados pelos modelo.

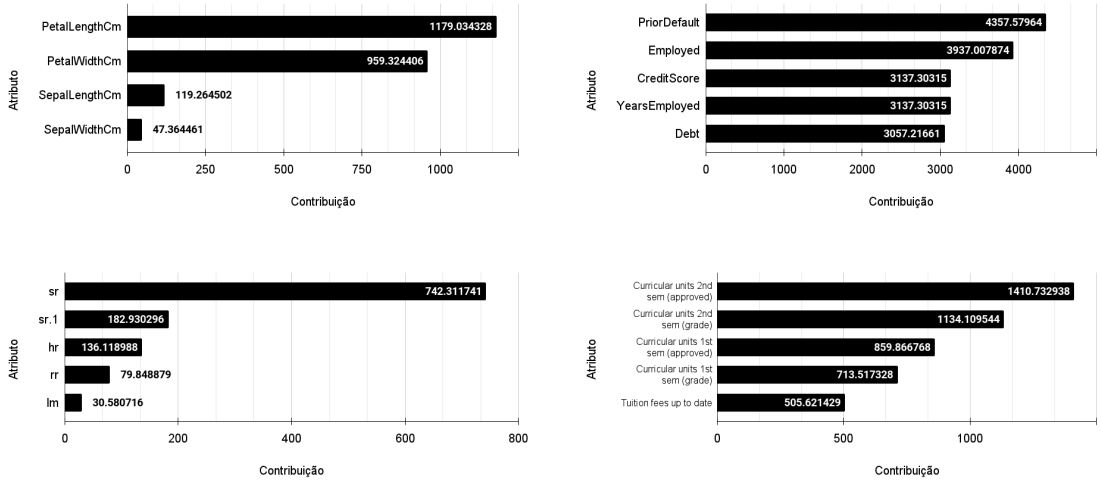

Figura 18 – Máximo 5 Atributos Mais Significativos dos Conjuntos de Dados

4 Experimentos e Resultados

4.1 Parâmetros e Ambiente de Execução

Diversos experimentos foram realizados à fim de responder às questões de pesquisa e analisar mais profundamente o desempenho dos algoritmos OCT-MIO e CART nos conjuntos de dados selecionados.

As implementações de ambos possuem parâmetros equivalentes em relação à capacidade de classificação do modelo treinado, divergindo apenas pelos seus parâmetros específicos acerca do método utilizado.

4.1.1 Ambiente de Execução dos Experimentos

Todos os experimentos executados neste trabalho foram realizados em um único ambiente descrito pela Tabela 3.

4.1.2 Parâmetros dos Dados

Os experimentos foram realizados sobre os 4 conjuntos de dados: Iris, Stress, Credit e Students. Todos passaram pelo pré-processamento, onde seus atributos foram:

- Convertidos para valores numéricos começados em 1;
- Normalizados entre 0 e 1;
- A redução da dimensionalidade foi aplicada utilizando a técnica *SelectKBest* com K = 5.

4.1.3 Parâmetros Gerais dos Modelos

Se uma árvore de classificação pudesse se dividir infinitamente, no caso extremo, a árvore se dividirá até que restasse apenas um elemento em cada folha,

Tabela 3 – Ambiente de Execução dos Experimentos

Item	**Descrição**
Processador	12th Gen Intel® Core™ i5-1235U × 12
Memória	RAM 32,0 GB, DDR4, 3200 Mhz
Processador Gráfico	Mesa Intel® Graphics (ADL GT2)
Armazenamento	1 TB HD + 0,5 TB SSD
Sistema Operacional	Ubuntu 22.04.4 LTS - 64-bit
Compilador C++	gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Resolvedor (*Solver*)	IBM ILOG CPLEX Optimization Studio_22.1
Linguagem de implementação dos modelos	C++
Linguagem de implementação de encapsuladores dos modelos	Python 3.10.12
Linguagem do tratamento de dados e execução dos experimentos	Python 3.10.12
Validação Cruzada K-Fold (*K-Fold Cross Validation*)	scikit-learn v1.5.1 (Python 3.10.12)

independentemente do método aplicado para a separação dos elementos. Sendo assim, a **altura máxima** da árvore é um parâmetro limitante comum a ambas implementações, pois, dentro da mesma altura, é possível analisar ambos os modelos de forma equivalente. Já os demais parâmetros não são passíveis de comparação entre os modelos. A partir disto, os experimentos comparam diversas métricas sobre os resultados obtidos pelos dois modelos e sobre os mesmos valores dos parâmetros em comum.

Altura mínima, diferente da altura máxima, é um parâmetro que não é passado explicitamente para o modelo, pois ele não aplica qualquer restrição à árvore, contudo é necessário que a altura mínima de uma árvore de classificação cheia seja tal, que sua quantidade de nós folha seja suficiente para classificar todas as classes do problema, portanto o parâmetro altura máxima precisa ser maior ou igual à altura mínima, conforme a relação dada por:

$$\begin{aligned} &\lceil \log_2 K \rceil \leq D, \\ &\lceil \log_2 K \rceil < K \leq 2^{\lceil \log_2 K \rceil}, \end{aligned} \tag{4.1}$$

onde K é o número de classes do problema, D é a altura máxima da árvore, $\lceil \log_2 K \rceil$ é a altura mínima necessária para que o número de folhas ($2^{\lceil \log_2 K \rceil}$) seja maior ou igual ao número de classes do problema.

Então, a Tabela 4 apresenta as alturas mínima, mínima+1 e mínima+2 já calculadas conforme a expressão (4.1) para todos os conjuntos de dados, as quais serão praticadas em todos os experimentos.

Tabela 4 – Alturas de Árvores Para Cada Conjunto de Dados Praticadas nos Experimentos

Conjunto	**Alt. Min.**	**Alt. Min. +1**	**Alt. Min. +2**
Iris	2	3	4
Credit	1	2	3
Stress	3	4	5
Academic	2	3	4

Dito isto, as alturas limites das árvores de classificação aplicadas aos experimentos foram definidas em uma variação de 3 alturas, sendo elas: altura mínima, altura mínima+1 e altura mínima+2. Desta forma, garante-se que os modelos gerem uma árvore cuja estrutura seja matematicamente viável para a classificação de todas as classes do problema. Portanto as alturas mínima+1 e altura mínima+2 foram propostas com o objetivo de explorar a variação do desempenho dos modelos em função do aumento da altura da árvore. Não foram propostas mais alturas, tanto em função da interpretabilidade, quanto do aumento da complexidade do problema.

É importante destacar que a altura máxima impõe uma restrição máxima de crescimento da árvore gerada pelo modelo de aprendizagem, mas não impõe nenhuma restrição sobre sua altura mínima. Em função disso, ainda que haja uma correlação direta entre a altura da árvore e seu desempenho, é possível que o modelo gere uma árvore menor que o tamanho máximo limitado ou, ainda, menor que a altura mínima matematicamente necessária para a classificação da quantidade de classes do problema. A árvore gerada pode ter altura menor que a altura mínima ou ainda não ter todas as classes do problema rotuladas em suas folhas, em particular,

quando mesmo assim ela tem melhor desempenho, conforme ilustrado pela Figura 19.

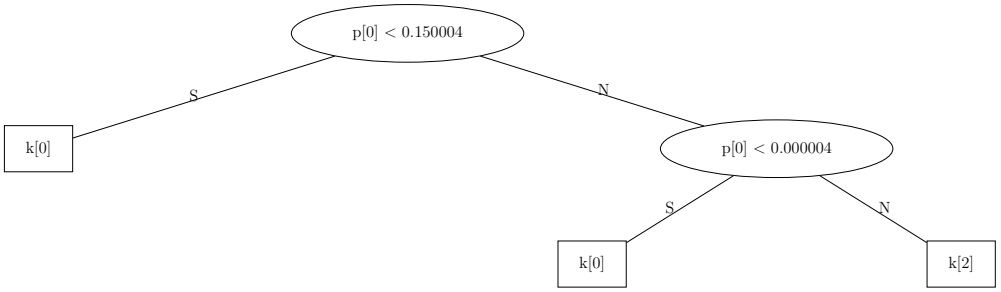

Figura 19 – Árvore de classificação ilustrativa para um problema de 3 classes, mas com folhas rotuladas com apenas 2 classes

4.1.4 Parâmetros Específicos dos Modelos

Os parâmetros específicos são parâmetros relacionados à estratégia aplicada por cada um dos modelos e não são passíveis de comparação entre os modelos, ainda que estes parâmetros influenciem no desempenho de cada modelo.

No modelo CART, o parâmetro **"*mínima variação de impureza*"** define um valor mínimo de redução da impureza absoluta que compõe a decisão se um dado nó da árvore irá ou não se dividir, evitando o *overfitting*. Quando a **"*mínima variação de impureza*" = 0.0**, o CART sempre irá optar por fazer uma divisão quando houver qualquer redução de impureza, até que a altura máxima para árvore seja alcançada.

Já o modelo OCT-MIO possui como parâmetros específicos: $\boldsymbol{\alpha}$, "**número mínimo de elementos por folha**" e "**tempo máximo de execução**", onde α determina o equilíbrio entre eficiência do modelo e a complexidade da árvore de classificação gerada por ele, o "número mínimo de elementos por folha" determina um número mínimo de elementos existentes em uma folha da árvore e, por fim, "tempo máximo de execução" determina o tempo máximo em que o modelo poderá tentar elaborar ou melhorar a solução, caso ainda não encontre a solução ótima.

Neste trabalho, assumiu-se o valor de $\alpha = 0.0$, pois foram consideradas quaisquer configurações de árvores, permitindo que ambos os modelos obtivessem suas melhores Acurácias sem quaisquer penalidades sobre o número de nós da árvore, limitados à altura máxima definida. Ainda que $\alpha > 0.0$ possa gerar árvores de menor altura ou com menos nós, considerou-se que as árvores dentro da altura máxima estabelecida ainda são interpretáveis. O **"número mínimo de elementos por folha" foi definido em 1**, assumindo que o modelo possa fazer quaisquer divisões que achar necessário, ainda que a divisão separe apenas 1 elementos em um subconjunto. O **"tempo máximo de execução" foi limitado em 10 horas**, ainda que, para alguns casos, o modelo consiga obter a otimalidade em menor tempo. Ainda assim, conjuntos de dados mais complexos podem dispender um tempo considerável até obter uma solução comprovadamente ótima ou mesmo factível.

4.2 Execução dos Experimentos

Cada modelo possui suas vantagens e desvantagens. Apesar do CART ser um modelo rápido, de menor complexidade, e capaz de encontrar ótimas soluções em grande parte dos casos, ele não garante uma solução ótima. Já o OCT-MIO, por ser um modelo de otimização, requer mais tempo para treinamento, mas, garante que a solução encontrada é a melhor possível, dentro de suas restrições de tempo e complexidade do problema.

4.2.1 Ajustes dos Modelos

Nos resultados a seguir, os valores entre parênteses "( )" significam que aquele valor advêm de uma solução provada ótima, enquanto os valores em **negrito** significam que estes foram os melhores resultados, dada a comparação OCT-MIO vs CART para uma mesma altura e um mesmo conjunto de dados. Por fim asterisco (*) significa que o resolvedor (no caso do OCT-MIO) não conseguiu obter uma solução factível dentro do tempo limite e dos parâmetros aplicados. É válido destacar que, se o CART obtém valores de Acurácia iguais ao OCT-MIO, quando este obtém uma solução ótima, isto significa que o CART também obteve uma solução ótima,

ainda que ele não a prove, mas não necessariamente significa que as árvores geradas por ambos são iguais, pois pode haver mais de uma solução ótima para o mesmo problema, mesmos parâmetros e com o mesmo valor de Acurácia.

O primeiro experimento analisa o desempenho de cada modelo em condições equivalentes somente durante o processo de aprendizagem, ou seja, analisa o quanto o modelo é capaz de identificar padrões intrínsecos aos dados e se ajustar a eles.

Para isso, ambos os modelos foram treinados com todos os elementos de cada conjunto de dados. Os treinamentos foram aplicados com 3 limites de alturas máximas, sendo altura mínima, mínima+1 e mínima+2. Por fim, os modelos foram avaliados com diversas métricas, destacando-se: Acurácia (Tabela 5 e Figura 20), Acurácia Balanceada (Tabela 6 e Figura 21) e Macro Pontuação F1 (Tabela 7 e Figura 22).

Tabela 5 – Acurácias dos Modelos Treinados em Cada Conjunto de Dados Completo

	Alt. Min.		**Alt. Min. +1**		**Alt. Min. +2**	
Conjunto	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**
Iris	**(0.960)**	**(0.960)**	**(0.993)**	0.973	**(1.000)**	0.993
Credit	**(0.855)**	**(0.855)**	**(0.859)**	0.855	**0.864**	0.856
Stress	**(1.000)**	**(1.000)**	**(1.000)**	**(1.000)**	**(1.000)**	**(1.000)**
Academic	**0.728**	0.712	0.504	**0.727**	*	**0.733**

Tabela 6 – Acurácias Balanceadas dos Modelos Treinados em Cada Conjunto de Dados Completo

	Alt. Min.		**Alt. Min. +1**		**Alt. Min. +2**	
Conjunto	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**
Iris	**(0.960)**	**(0.960)**	**(0.993)**	0.973	**(1.000)**	0.993
Credit	**(0.862)**	**(0.862)**	**(0.866)**	0.862	**0.870**	0.864
Stress	**(1.000)**	**(1.000)**	**(1.000)**	**(1.000)**	**(1.000)**	**(1.000)**
Academic	**0.583**	0.564	(0.339)	**0.580**	*	**0.626**

Também foi observado o tempo gasto por cada modelo para treinamento em cada conjunto de dados e em cada uma das alturas variadas. Os tempos, em

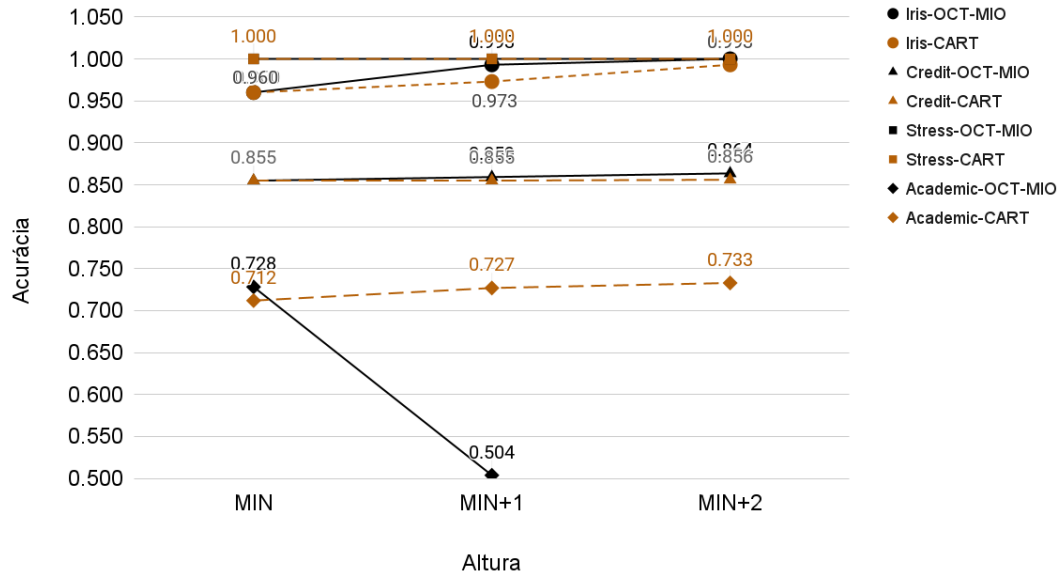

Figura 20 – Gráfico de Acurácias dos Modelos Treinados em Cada Conjunto de Dados Completo

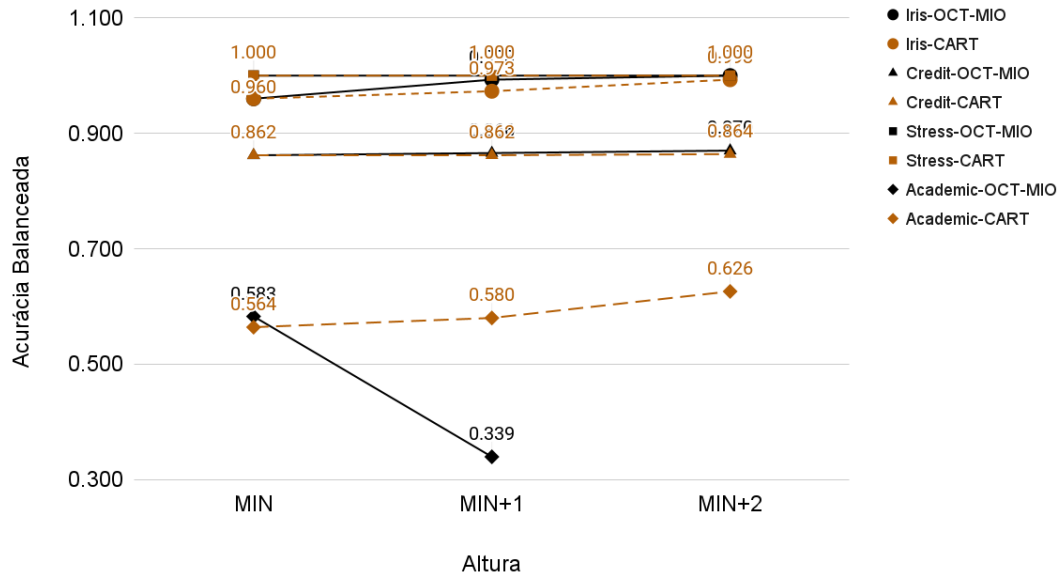

Figura 21 – Gráfico de Acurácias Balanceadas dos Modelos Treinados em Cada Conjunto de Dados Completo

Tabela 7 – Pontuações F1 dos Modelos Treinados em Cada Conjunto de Dados Completo

Conjunto	Alt. Min.		Alt. Min. +1		Alt. Min. +2	
	OCT-MIO	**CART**	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**
Iris	**(0.960)**	0.862	**(0.993)**	0.973	**(1.000)**	0.993
Credit	**(0.855)**	**0.855**	**(0.859)**	0.855	**(0.863)**	0.856
Stress	**(1.000)**	**1.000**	**(1.000)**	**1.000**	**(1.000)**	**1.000**
Academic	**0.528**	0.514	(0.235)	**0.527**	*	**0.621**

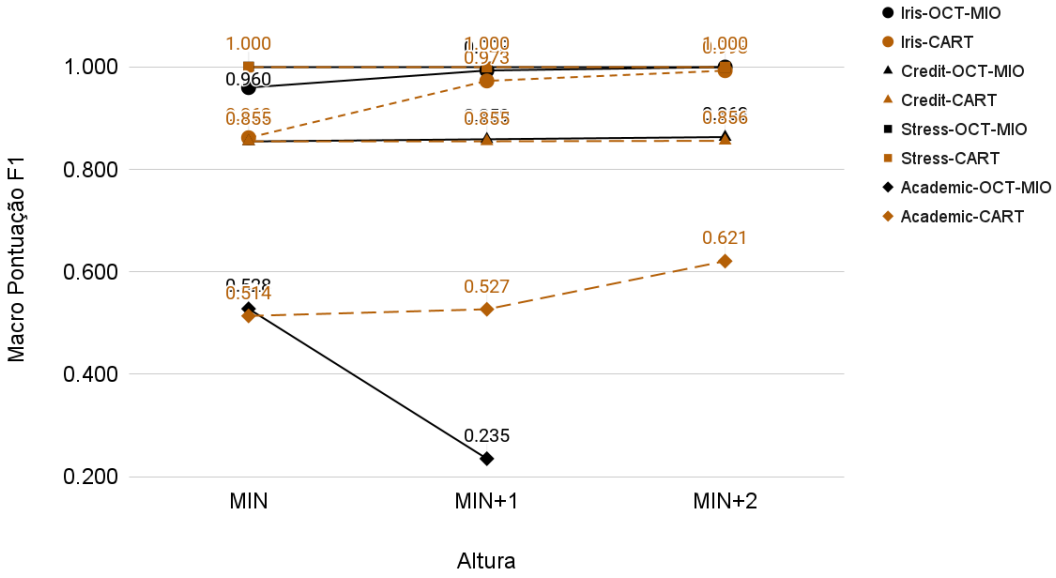

Figura 22 – Gráfico de Pontuações F1 dos Modelos Treinados em Cada Conjunto de Dados Completo

segundos, estão anotados na Tabela 8 e Figura 23.

4.2.2 OCT-MIO Aplicado Como Heurística vs CART

No caso do modelo OCT-MIO, baseado em otimização inteira mista, apesar do tempo final para obtenção da melhor solução (ótima ou apenas factível), observou-se que, em alguns casos, o modelo obtém um valor objetivo igual ou próximo ao da melhor solução, antes mesmo do tempo total.

Tabela 8 – Tempo em segundos de Treinamento dos Modelos em Cada Conjunto de Dados Completo

	Alt. Min.		Alt. Min. +1		Alt. Min. +2	
Conjunto	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**	**OCT-MIO**	**CART**
Iris	(9.759)	**0.010**	(1363.128)	**0.009**	(83.999)	**0.011**
Credit	(0.782)	**0.033**	(1275.052)	**0.042**	36004.951	**0.053**
Stress	(828.075)	**0.270**	(16132.162)	**0.304**	(7991.822)	**0.288**
Academic	36002.915	**0.868**	36001.074	**1.110**	*	**1.212**

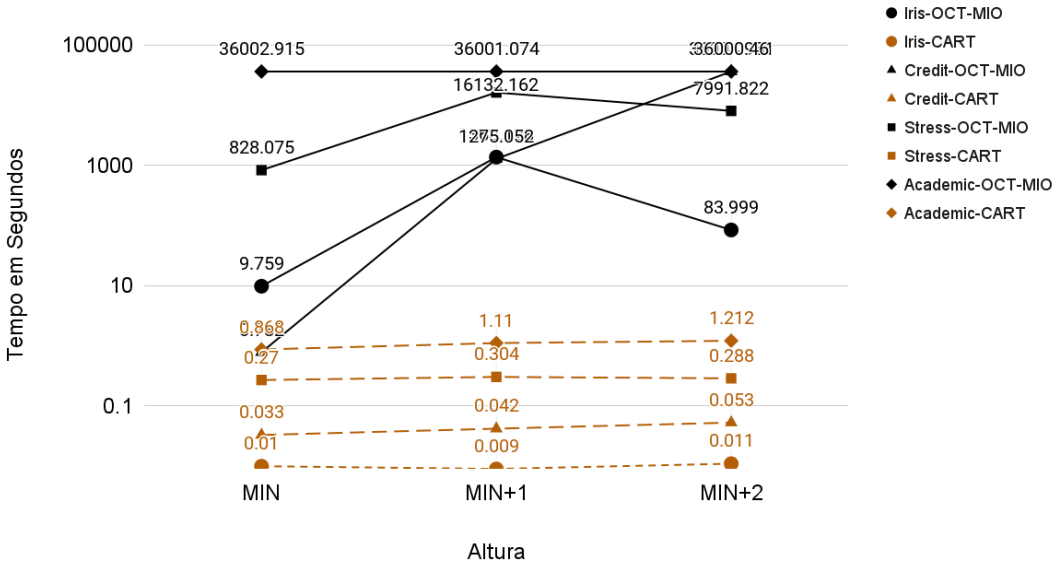

Figura 23 – Tempos de Treinamento dos Modelos nos Conjuntos de Dados Inteiros em Segundos em Escala Logaritmica

Portanto os experimentos descritos nesta seção visam analisar a capacidade do modelo OCT-MIO em obter soluções em tempo reduzido, sem necessariamente provar a otimalidade das soluções, e avaliar o desempenho destas soluções em relação ao CART.

A Tabela 9 e a Figura 24 mostram os tempos mínimos e tempos finais em que o modelo OCT-MIO demorou para obter o melhor valor objetivo em cada conjunto de dados completo até o tempo máximo de 10 horas de treinamento.

Tabela 9 – Tempo Mínimo e Tempo Total (em segundos) de Treinamento até Obtenção da Solução de Melhor Valor Objetivo do modelo OCT-MIO em no máximo 10 horas

	Alt. Min.		**Alt. Min. +1**		**Alt. Min. +2**	
Conjunto	**T. Min**	**T. Total**	**T. Min**	**T. Total**	**T. Min**	**T. Total**
Iris	9.750	(9.759)	1363.090	(1363.128)	83.950	(83.999)
Credit	0.760	(0.782)	1275.000	(1275.052)	267.470	36004.951
Stress	827.970	(828.075)	16131.920	(16132.162)	7391.430	(7991.822)
Academic	25124.200	36002.915	29197.100	36001.074	*	*

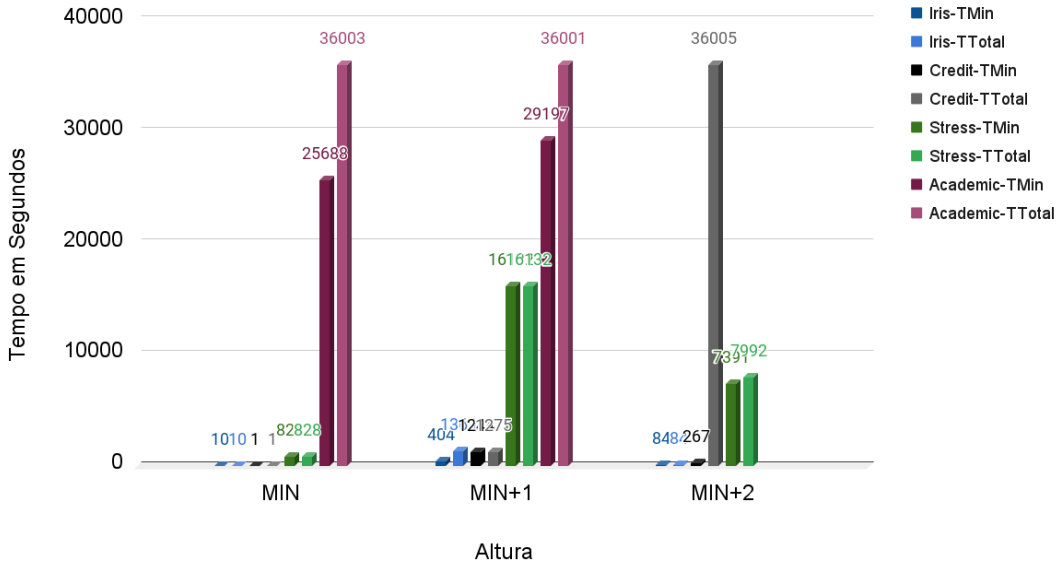

Figura 24 – Tempo Min. e Max. em segundos de Treinamento até Obtenção da Solução de Melhor Valor Objetivo do modelo OCT-MIO em Cada Conjunto de Dados Completo em Escala Logaritmica

Os gráficos da Figura 25, Figura 26 e Figura 27 apresentam valores de otimização do modelo OCT-MIO em alguns conjuntos de dados com tempo limite de treinamento de 36000 segundos (10 horas). Nestes gráficos, *best integer* é o valor da função objetivo do modelo que, em problemas de minimização, quanto menor, melhor. Já *best bound* é o valor do limite inferior da função objetivo obtido através da relaxação das restrições do problema. Ele determina um limite até o qual a

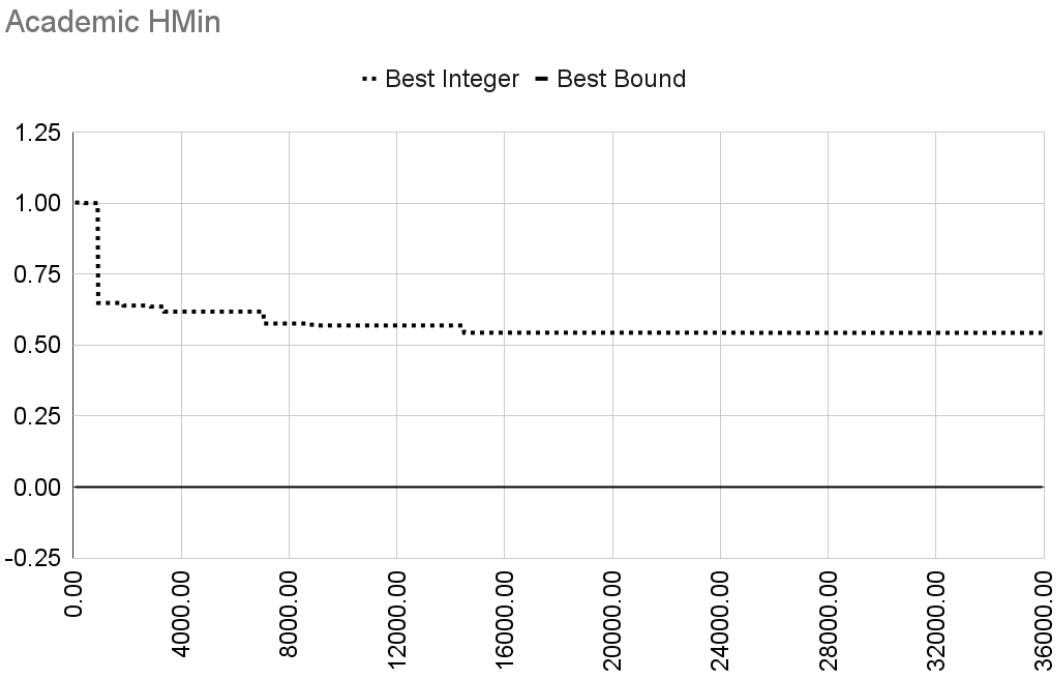

Figura 25 – Best Integer e Best Bound - Academic - Altura Mínima

função objetivo é capaz de melhorar. Quando *best integer* e *best bound* se encontram, isto significa que o resolvedor encontrou a solução ótima para o problema.

Para a altura mínima do conjunto Academic, os experimentos mostraram que o modelo OCT-MIO já havia obtido o último melhor valor objetivo em 7 horas (25124.2 s) e é possível perceber na Figura 25 que o modelo já havia obtido valor objetivo próximo a este em aproximadamente 4 horas (14469.84 s).

Já no gráfico da Figura 26, para a altura mínima + 1 do conjunto Stress, cujo tempo total de treinamento foi de 4 h, 28 min e 51 s (16132.162 s), o modelo OCT-MIO obteve uma solução ótima de valor objetivo 0.0079 em aproximadamente 4 h 14 min 52 s (15292.20 s), próximo tempo ao final, no entanto, o modelo já havia obtido uma solução de valor objetivo 0.0159 desde aproximadamente 4min 37s (277.04 s).

Em último exemplo, o gráfico da Figura 27 mostra que o modelo OCT-MIO já havia obtido a solução ótima desde aproximadamente 5 min e 47 s (347.06 s),

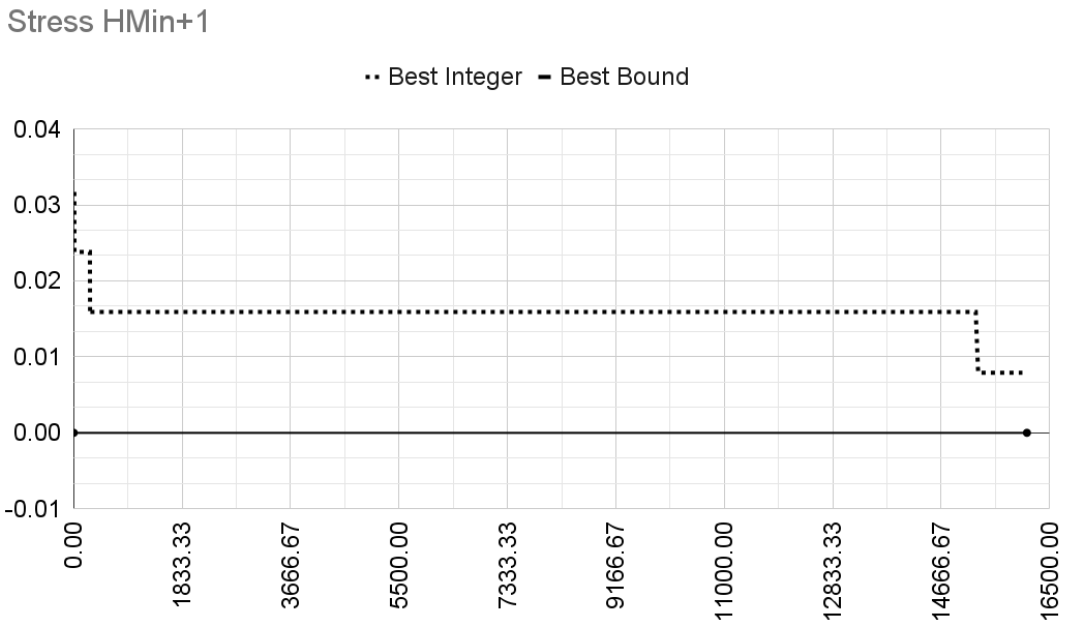

Figura 26 – Best Integer e Best Bound - Stress - Altura Mínima + 1

embora o modelo tenha provado a otimalidade da solução somente aos 22 min e 43 s (1363.128 s).

Portanto, é possível observar que, ainda que o modelo OCT-MIO possa levar um tempo muito maior que o CART para fornecer a melhor solução, o modelo é capaz de encontrar soluções factíveis em um tempo intermediário.

Para descobrir se o OCT-MIO é capaz de encontrar soluções melhores que o CART desde um tempo de treinamento menor, também foi observado o desempenho do modelo OCT-MIO com tempos máximos de treinamento de 5 horas e também 5 minutos. Ao reduzir o tempo de treinamento do OCT-MIO, consequentemente as soluções obtidas serão iguais ou inferiores à solução obtida com tempo de treinamento até 10h do mesmo modelo, mas, ainda assim, tais soluções podem ser melhores que as soluções obtidas pelo CART em alguns casos.

Os valores de Acurácia (Tabela 10), Acurácia Balanceada (Tabela 11) e Macro Pontuação F1 (Tabela 12) do modelo OCT-MIO, treinado com os **tempos máximos de 5 minutos, 5 horas e 10 horas**, foram comparados ao CART,

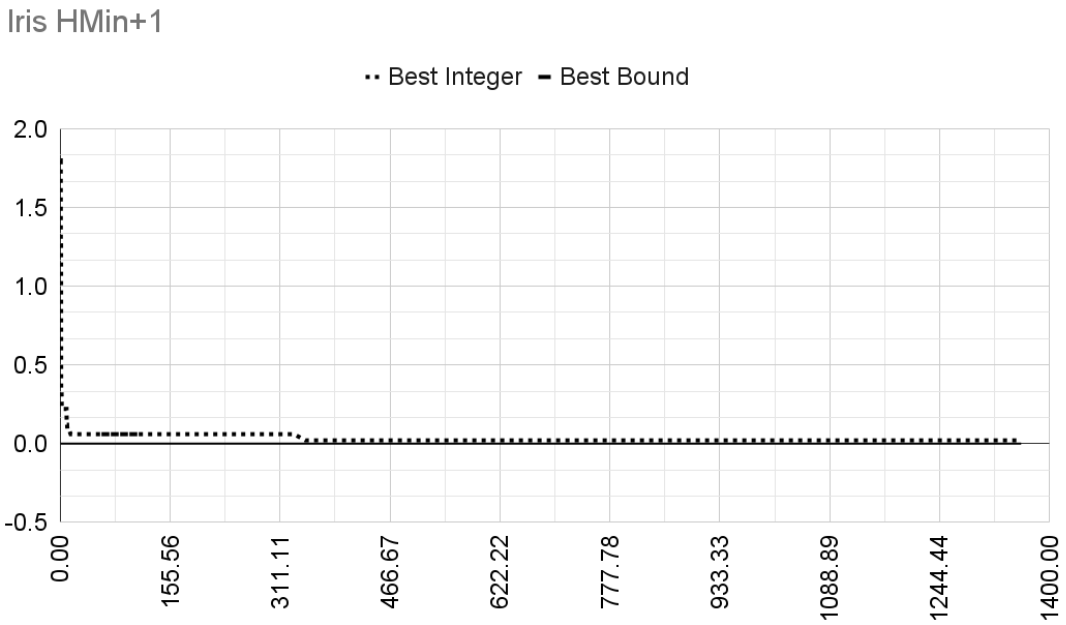

Figura 27 – Best Integer e Best Bound - Iris - Altura Mínima + 1

para diferentes conjuntos de dados. Ao analisar linha a linha de cada uma destas tabelas, é possível estabelecer que, ainda que o OCT-MIO tenha sido treinado com tempo reduzido, algumas soluções ainda são iguais ou melhores que a solução dada pelo CART.

4.2.3 Validação Cruzada K-Fold (*K-Fold Cross-Validation*)

Os experimentos deste trabalho visam avaliar não só a capacidade de aprendizado dos modelos, mas também sua capacidade de obter soluções generalizadas, capazes de classificar também dados diferentes dos quais os modelos foram treinados. Além disso, estes experimentos também permitem observar o desempenho e o tempo de treinamento dos modelos, especialmente do OCT-MIO, em conjuntos de dados reduzidos.

Os experimentos estratificados desta seção foram realizados usando K-Fold, onde $K = 5$, conforme Figura 12. Ou seja, para cada conjunto de dados, são realizadas 5 iterações onde, em cada uma delas, o conjunto de dados é dividido

Tabela 10 – Acurácia: OCT-MIO 5min, OCT-MIO 5h, OCT-MIO 10h vs CART

Iris		Acurácia		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	**(0.960)**	**(0.960)**	**(0.960)**	**0.960**
Hmin+1	**0.993**	**(0.993)**	**(0.993)**	0.973
Hmin+2	**(1.000)**	**(1.000)**	**(1.000)**	0.993
Credit		Acurácia		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	**(0.855)**	**(0.855)**	**(0.855)**	**0.855**
Hmin+1	**0.858**	**(0.859)**	**(0.859)**	0.855
Hmin+2	**0.862**	**0.864**	**0.864**	0.856
Stress		Acurácia		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	0.997	**(1.000)**	**(1.000)**	**1.000**
Hmin+1	0.997	0.998	**(1.000)**	**1.000**
Hmin+2	0.997	**(1.000)**	**(1.000)**	**1.000**
Academic		Acurácia		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	0.499	**0.728**	**0.728**	0.712
Hmin+1	*	0.502	0.504	**0.727**
Hmin+2	*	*	*	**0.733**

em 5 partes de proporções similares, das quais, 1 parte é utilizada para teste e as demais para treinamento.

Para isso, utilizou-se o objeto `KFold` (*K-Fold Cross Validator*) (KOHAVI et al., 1995) do pacote `sklearn.model_selection` para fornecer conjuntos de índices de elementos para treinamento e teste. Instanciado como `KFold(n_splits=5, shuffle=True, random_state=42)`, onde `n_splits` é o número de *folds*, `shuffle=True` define que os dados sejam misturados antes de quaisquer divisão e `random_state=42` é a "semente" para a aleatoriedade da mistura dos dados. O valor fixado 42 permite que, ainda que a mistura dos dados seja aleatória, esta aleatoriedade possa ser reproduzida em uma repetição dos experimentos. Desta forma, ambos os modelos são treinados 5 vezes para cada

Tabela 11 – Acurácia Balanceada: OCT-MIO 5min, OCT-MIO 5h, OCT-MIO 10h vs CART

Iris		Acurácia Balanceada		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	**(0.960)**	**(0.960)**	**(0.960)**	**0.960**
Hmin+1	**0.993**	**(0.993)**	**(0.993)**	0.973
Hmin+2	**(1.000)**	**(1.000)**	**(1.000)**	0.993
Credit		Acurácia Balanceada		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	**(0.862)**	**(0.862)**	**(0.862)**	**0.862**
Hmin+1	**0.865**	**(0.866)**	**(0.866)**	0.862
Hmin+2	**0.869**	**0.870**	**0.870**	0.864
Stress		Acurácia Balanceada		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	0.997	**(1.000)**	**(1.000)**	**1.000**
Hmin+1	0.997	0.998	**(1.000)**	**1.000**
Hmin+2	0.997	**(1.000)**	**(1.000)**	**1.000**
Academic		Acurácia Balanceada		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	0.333	**0.583**	**0.583**	0.564
Hmin+1	*	0.336	0.339	**0.580**
Hmin+2	*	*	*	**0.626**

conjunto de dados, onde, a cada vez, o modelo é treinado com aproximadamente 80% do conjunto de dados e testado com aproximadamente 20%, para os quais os dados de testes nunca se repetem.

Ao final de cada execução de cada iteração do *K-Fold*, são calculados os valores de Acurácia, Acurácia Balanceada e Macro Pontuação F1. E, ao final das 5 iterações, também é calculada a média aritmética e o desvio padrão das métricas já calculadas para cada iteração, conforme o Algoritmo 3. Os desvios padrão (DP) estão dispostos na coluna à direita das suas respectivas médias.

Ambos os modelos também foram analisados sobre todos os conjuntos de dados para as alturas mínima, mínima+1 e mínima+2. As métricas analisadas foram:

Tabela 12 – Macro Pontuação F1: OCT-MIO 5min, OCT-MIO 5h, OCT-MIO 10h vs CART

Iris		Macro Pontuação F1		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	**(0.960)**	**(0.960)**	**(0.960)**	**0.960**
Hmin+1	**0.993**	**(0.993)**	**(0.993)**	0.973
Hmin+2	**(1.000)**	**(1.000)**	**(1.000)**	0.993
Credit		Macro Pontuação F1		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	**(0.855)**	**(0.855)**	**(0.855)**	**0.855**
Hmin+1	**0.858**	**(0.859)**	**(0.859)**	0.855
Hmin+2	**0.862**	**0.863**	**(0.863)**	0.856
Stress		Macro Pontuação F1		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	0.997	**(1.000)**	**(1.000)**	**1.000**
Hmin+1	0.997	0.998	**(1.000)**	**1.000**
Hmin+2	0.997	**(1.000)**	**(1.000)**	**1.000**
Academic		Macro Pontuação F1		
Altura	OCT-MIO 5 min	OCT-MIO 5 h	OCT-MIO 10 h	CART
Hmin	0.222	**0.528**	**0.528**	0.514
Hmin+1	*	0.229	(0.235)	**0.527**
Hmin+2	*	*	*	**0.621**

Acurácia (Tabela 13 e Figura 28), Acurácia Balanceada (Tabela 14 e Figura 29) e Macro Pontuação F1 (Tabela 15 e Figura 30), onde os resultados referem-se à média aritmética das métricas obtidas nos 5 *folds*, sendo valores entre parênteses "( )" quando o modelo OCT-MIO obteve todas as 5 soluções provadas ótimas, enquanto resultados sem parênteses referem-se a médias onde ao menos 1 iteração não provou a otimalidade da sua solução. Os valores em **negrito** significam que estes foram os melhores resultados de média na a comparação OCT-MIO vs CART para uma mesma altura e um mesmo conjunto de dados. Por fim, asterisco (*) significa que o resolvedor (no caso do OCT-MIO) não conseguiu obter solução factível em ao menos uma das K iterações, dentro do tempo limite. Assim, um valor seguido de asterisco significa que $1 < k < K$ iterações não obtiveram solução factível e a

Algoritmo 3: K-Fold

Entrada:

- X: Conjunto de dados inicial, sendo uma matriz$[n][P+1]$
- K: Número de *folds*
- *modelo*: Modelo de aprendizado de máquina
- *métricas*: Conjunto de métricas a calcular, sendo um vetor de tamanho M

Saída: Média das métricas recebidas por parâmetro

Início:

```
1  Divida o conjunto X em K folds X_f, para f = 1,2,...,K ;
2  Inicializa matriz métricasMedidas[M][K] vazia;
3  para k ← 1 até K faça
4      dadosTreinamento ← ⋃ X_{f≠k}, f = {1,2,...,K};
5      dadosTeste ← X_k;
6      XTreinamento ← dadosTreinamento[ colunas de 1 até P ];
7      yTreinamento ← dadosTreinamento[ coluna P + 1 ];
8      XTeste ← dadosTeste[ colunas de 1 até P ];
9      yTeste ← dadosTeste[ coluna P + 1 ];
10     modeloTreinado ← treinar modelo com XTreinamento e
        yTreinamento;
11     yPreditos ← testar modeloTreinado com XTeste;
12     para m ← 1 até tamanho do vetor métricas faça
13         métricasMedidas[m][k] ← avaliar a métricas[m] do
            modeloTreinado comparando yPreditos e yTeste;
14 Calcula médias[M] com as médias do vetor métricasMedidas[M];
15 retorna médias;
   Fim:
```

média corresponde a $k < K$ iterações. E apenas um asterisco significa todas as K iterações não obtiveram solução factível.

Também foram analisados os tempos de treinamento. A Tabela 16 e Figura 31 apresentam as médias de tempo de treinamento em k=5 *folds* para cada modelo e para cada conjunto de dados separados desde a altura mínima até a altura mínima+1.

Tabela 13 – Acurácias dos Modelos Treinados em Cada Conjunto de Dados Estratificados (K-Fold5)

Conjunto	Alt. Min.				Alt. Min. +1				Alt. Min. +2			
	OCT-MIO	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP
Iris	**(0.947)**	±0.038	**0.947**	±0.018	**(0.960)**	±0.043	**0.960**	±0.043	(0.940)	±0.060	**0.953**	±0.030
Credit	**(0.855)**	±0.011	**0.855**	±0.011	(0.846)	±0.006	**0.855**	±0.011	0.824*	±0.015	**0.843**	±0.021
Stress	**(0.994)**	±0.009	0.987	±0.012	**(0.992)**	±0.010	0.987	±0.012	**(0.992)**	±0.008	0.987	±0.012
Academic	**0.717**	±0.022	0.709	±0.019	0.652	±0.082	**0.726**	±0.017	0.569*	±0.107	**0.732**	±0.013

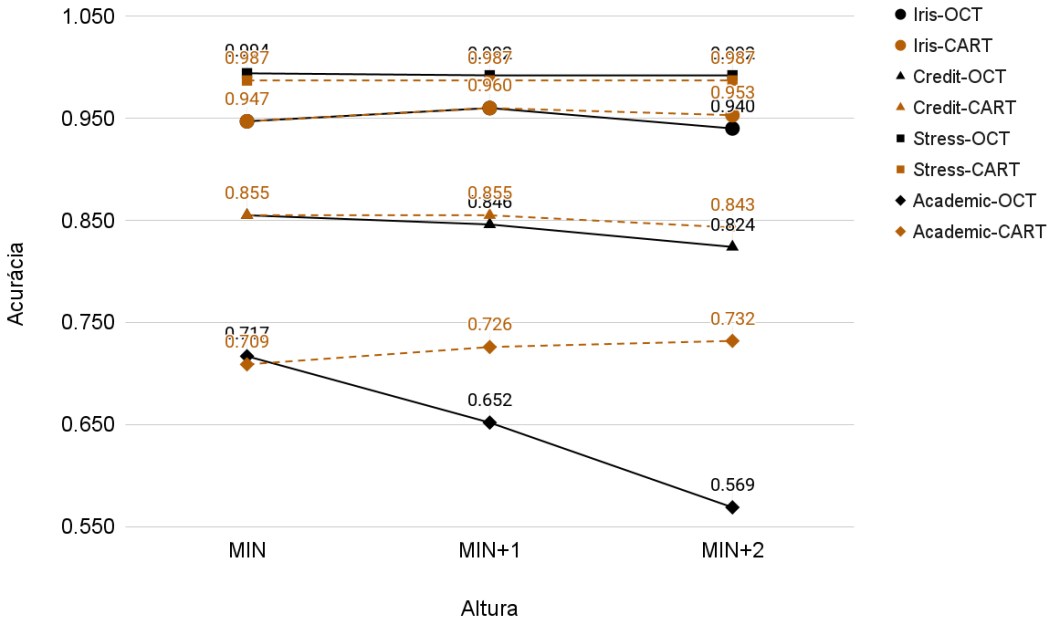

Figura 28 – Acurácias dos Modelos nos Conjuntos de Dados Estratificados (K-Fold5)

4.3 Análise dos Resultados

Esta seção descreve as análises obtidas a partir dos experimentos realizados, destacando pontos de superação e limitações dos modelos comparados. As conclusões obtidas nesta seção são conclusões técnicas sobre resultados de métricas dos modelos. Conclusões sobre a aplicabilidade dos mesmos serão feitas na Seção 4.4.

Tabela 14 – Acurácias Balanceadas dos Modelos Treinados em Cada Conjunto de Dados Estratificados (K-Fold5)

Conjunto	Alt. Min.				Alt. Min. +1				Alt. Min. +2			
	OCT-MIO	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP
Iris	(0.945)	±0.037	**0.948**	±0.011	**(0.963)**	±0.038	**0.963**	±0.040	(0.946)	±0.054	**0.954**	±0.027
Credit	**(0.862)**	±0.012	**0.862**	±0.012	(0.853)	±0.009	**0.862**	±0.012	0.826*	±0.012	**0.850**	±0.027
Stress	**(0.994)**	±0.008	0.988	±0.011	**(0.992)**	±0.010	0.988	±0.011	**(0.992)**	±0.008	0.988	±0.011
Academic	**0.587**	±0.015	0.564	±0.010	0.501	±0.098	**0.620**	±0.034	0.399*	±0.114	**0.640**	±0.031

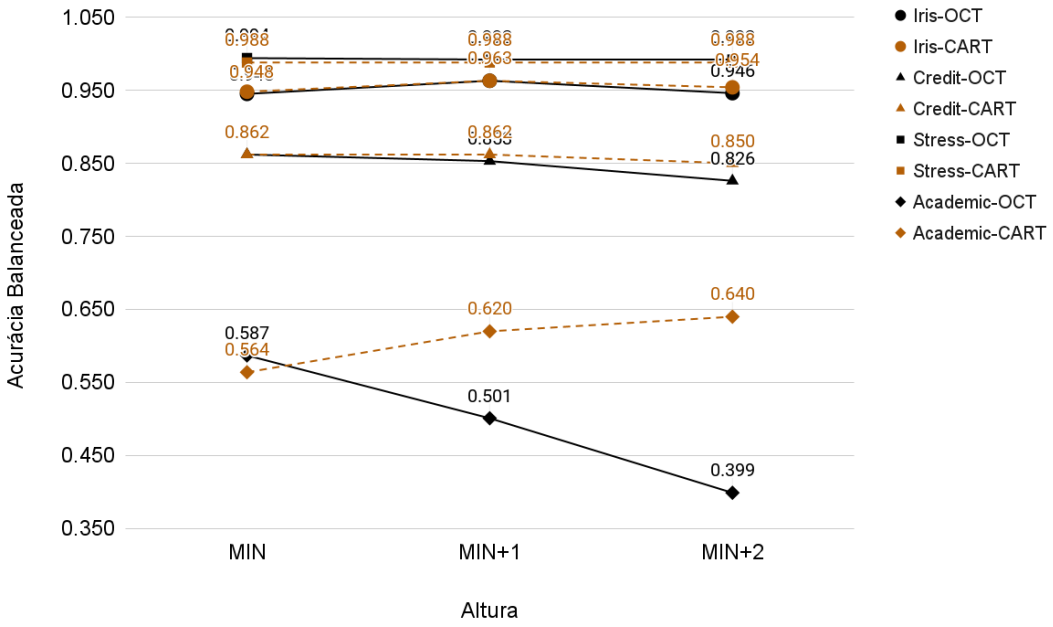

Figura 29 – Acurácias Balanceadas dos Modelos nos Conjuntos de Dados Estratificados (K-Fold5)

4.3.1 Ajustes dos Modelos

Nos resultados da Seção 4.2.1 observa-se que, não apenas as soluções ótimas, mas também as soluções factíveis do modelo OCT-MIO, obtiveram resultados iguais ou superiores ao CART em Acurácia, Acurácia Balanceada e Macro Pontuação F1, destacando a capacidade de aprendizado do modelo. Ao mesmo tempo, a proximidade dos resultados obtidos pelo CART em relação aos resultados do OCT-MIO, sobretudo quando estes resultados são ótimos, destacam também a boa eficiência da heurística utilizada pelo CART.

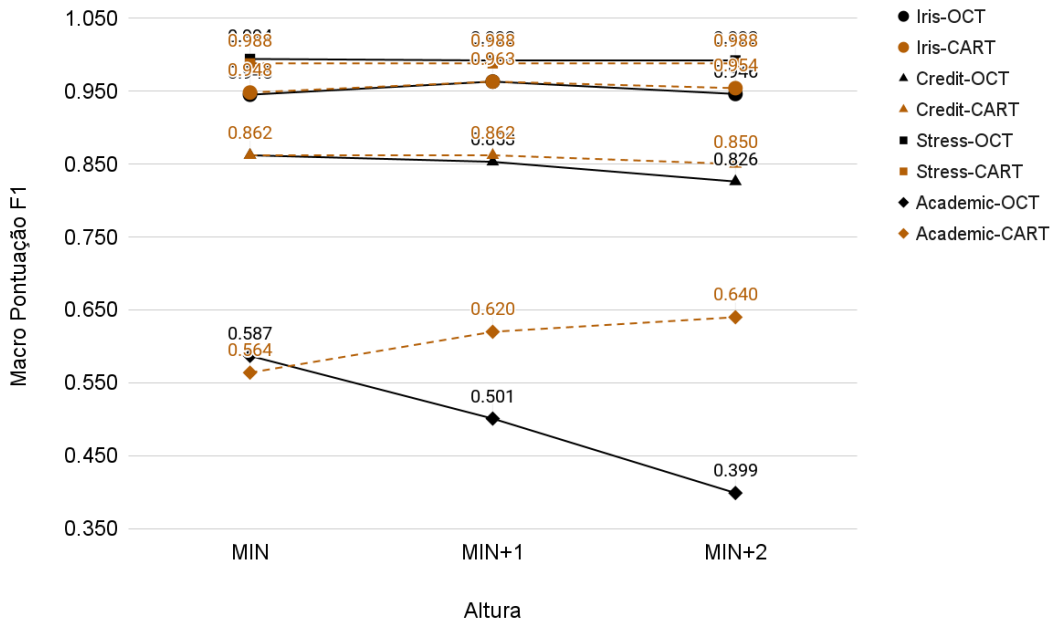

Figura 30 – Pontuações F1 dos Modelos nos Conjuntos de Dados Estratificados (K-Fold5)

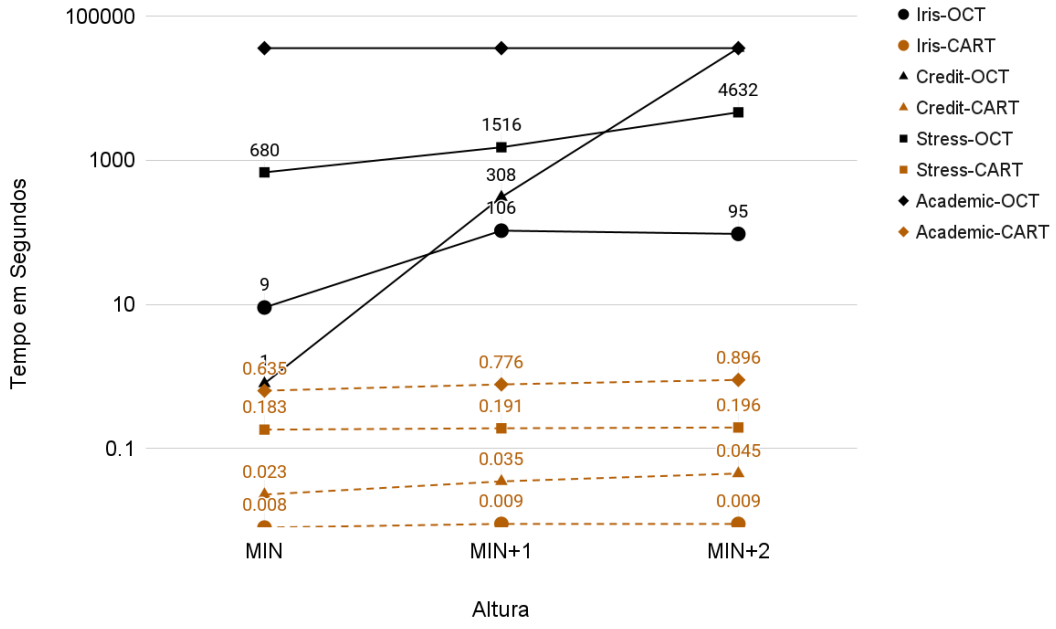

Figura 31 – Média de Tempos de Treinamento dos Modelos no K-Fold5

Tabela 15 – Pontuações F1 dos Modelos Treinados em Cada Conjunto de Dados Estratificados (K-Fold5)

	Alt. Min.				**Alt. Min. +1**				**Alt. Min. +2**			
Conjunto	**OCT-MIO**	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP
Iris	**(0.946)**	±0.036	**0.946**	±0.016	**(0.961)**	±0.041	**0.961**	±0.042	(0.942)	±0.057	**0.953**	±0.029
Credit	**(0.854)**	±0.010	**0.854**	±0.010	(0.845)	±0.004	**0.854**	±0.010	0.823*	±0.015	**0.841**	±0.021
Stress	**(0.994)**	±0.008	0.987	±0.012	**(0.992)**	±0.010	0.987	±0.012	**(0.992)**	±0.008	0.987	±0.012
Academic	**0.558**	±0.032	0.512	±0.011	0.441	±0.121	**0.615**	±0.058	0.314*	±0.155	**0.643**	±0.036

Tabela 16 – Média do Tempo de Treinamento em Segundos dos Modelos em Cada Conjunto de Dados Estratificados (K-Fold5)

	Alt. Min.				**Alt. Min. +1**				**Alt. Min. +2**			
Conjunto	**OCT-MIO**	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP	**OCT-MIO**	DP	**CART**	DP
Iris	(9.060)	±5.632	**0.008**	±0.000	(105.581)	±44.174	**0.009**	±0.002	(95.189)	±134.127	**0.009**	±0.000
Credit	(0.799)	±0.359	**0.023**	±0.001	(308.177)	±66.066	**0.035**	±0.004	36004.830*	±2.042	**0.045**	±0.003
Stress	(680.148)	±736.185	**0.183**	±0.008	(1515.665)	±2805.574	**0.191**	±0.004	(4631.813)	±4014.700	**0.196**	±0.004
Academic	36005.023	±9.892	**0.635**	±0.043	36010.559	±7.487	**0.776**	±0.020	36000.759*	±0.501	**0.896**	±0.054

Ambos os modelos obtiveram resultados de métricas variando entre 0.85 e 1.0 em todos os conjuntos, com exceção do conjunto Academic, reforçando novamente uma boa capacidade de ajuste aos dados. O conjunto de dados Academic, é muito complexo, tanto em número de elementos, quanto em número e distribuição espacial dos atributos. Por isso, neste conjunto de dados, ambos os modelos obtiveram métricas inferiores a 0.85. Mas, especialmente no caso do OCT-MIO, o modelo ainda conseguiu obter uma solução factível superior ao CART nas 3 métricas para a altura mínima. Entretanto, para a altura mínima+1, o modelo obteve valores muito baixos nas 3 métricas e, para a altura mínima+2, o modelo, se quer, obteve uma solução factível, em função do aumento da dificuldade do problema. Tais resultados reforçam que o modelo de Árvore de Classificação Ótima usando Otimização Inteira Mista é capaz de gerar árvores igualmente ou mais adequadas aos dados em relação a Árvore de Classificação e Regressão, desde que o tempo e recursos computacionais disponíveis sejam suficientes.

Quanto ao tempo de treinamento, enquanto o modelo CART levou alguns milissegundos para treinamento em cada conjunto de dados, exceto no Academic, no qual ele ainda ficou abaixo dos 2 segundos, o OCT-MIO levou de alguns segundos até algumas horas para treinamento dos conjuntos menos complexos, mas levou o tempo

máximo (10h) para a altura mínima+2 do conjunto Stress. Já para as 3 alturas do conjunto Academic, o modelo também levou tempo máximo de treinamento, mas obteve resultados insuficientes, devido à dificuldade deste problema, conforme a Tabela 8 e no gráfico da Figura 23.

Tais resultados colocam o CART a frente do OCT-MIO quando o parâmetro de comparação é o menor tempo de treinamento do modelo, principalmente quando comparado ao tempo de treinamento de resultados ótimos do OCT-MIO. Entretanto, o modelo OCT-MIO ainda pode alcançar soluções factíveis melhores que o CART, em uma ou mais métricas, em um tempo muito menor que o tempo total gasto até que o modelo consiga provar a otimalidade da solução. O gráfico da Figura 24 evidencia essa discrepância principalmente nos casos mais complexos, onde o OCT-MIO atingiu o tempo máximo de treinamento, mas já havia obtido o melhor valor objetivo em menos tempo.

É possível perceber também que, para quaisquer conjuntos de apenas 2 classes, como é o caso do conjunto Credit, os resultados, tanto do OCT-MIO quanto do CART, serão iguais para sua altura mínima (altura 1), uma vez que ambos os modelos fornecem a solução que tem a maior redução da impureza com o máximo de apenas 1 divisão, como evidenciado pelas métricas da Seção 4.2.1. Em outras palavras, para uma árvore que faz apenas 1 divisão, a solução ótima local também é a solução ótima global.

Outra inferência observada sobre o tempo de treinamento foi que nem sempre o tempo de treinamento aumenta proporcionalmente à altura da árvore. Pode ser que uma árvore mais alta leve menos tempo para ser treinada que uma árvore mais baixa, tal como ocorre da altura mínima+1 para a altura mínima+2 dos conjuntos Iris e Stress, no gráfico da Figura 23.

Por fim, os experimentos mostraram que, para conjuntos de dados muito complexos, como o Academic, o modelo OCT-MIO, o modelo surpreende ao obter solução factível para altura mínima, com as 3 métricas superiores ao CART. Já para altura mínima+1, o modelo ainda apresenta uma solução factível, mas com um desempenho ruim. E, para altura mínima+2, o modelo foi incapaz de obter, se quer, uma solução factível, dadas as condições de limite de tempo e hardware

utilizadas.

4.3.2 OCT-MIO Aplicado Como Heurística vs CART

Os experimentos da Seção 4.2.2 visam analisar o desempenho de soluções intermediárias do OCT-MIO em relação ao CART.

Gráficos como o da Figura 24 e a Tabela 9 permitem constatar, antes de tudo, que o OCT-MIO é capaz de fornecer soluções ótimas em alguns poucos minutos para alguns conjuntos de dados mais simples. Também é possível constatar que, apesar do OCT eventualmente levar horas ou até mais tempo no treinamento de conjuntos de dados mais complexos, ele ainda pode fornecer soluções factíveis até melhores que o CART também em um tempo intermediário. Por exemplo, para a altura mínima+2 do conjunto Credit, o OCT-MIO não provou a solução ótima em até 10h (36000 s), mas já havia encontrado a solução final desde 4 min e 27 s (267 s). O mesmo aconteceu para o conjunto Academic que, tanto para altura mínima quanto mínima+1, levou o tempo máximo de 10 horas para concluir o treinamento sem obtenção da solução ótima, mas já há havia obtido a solução final desde 6 h 58 min e 44s (25124 s) para a altura mínima e 8h 6min e 37 s (29197 s) para a altura mínima+1.

O tempo de treinamento do OCT-MIO até a obtenção da solução depende diretamente da dificuldade do problema, podendo levar de alguns segundos, horas, dias ou até mais para encontrar a solução ótima ou mesmo factível. Também, em função da dificuldade do problema, não é possível saber previamente se o OCT-MIO irá convergir rápido ou devagar até a solução ótima. Portanto, ainda que seja possível visualizar a convergência dos modelos, como nos gráficos que apresentam valores de *Best Integer* e *Best Bound*, não é possível estabelecer uma previsão de tempo até a otimalidade.

4.3.3 Validação Cruzada K-Fold (*K-Fold Cross-Validation*)

Os resultados apresentados na Seção 4.2.3 podem soar inicialmente controversos, pois é possível ver valores advindos de soluções ótimas do OCT-MIO inferiores à valores advindos de soluções factíveis do CART, como ocorre na tabelas:

Tabela 13, Tabela 14 e Tabela 15. Isso ocorre porque a otimalidade de uma solução é provada sob os dados de treinamento, enquanto as métricas utilizadas na validação das soluções no *K-Fold* são obtidas submetendo ao modelo um conjunto de dados diferente dos quais ele foi treinado. É exatamente essa metodologia que evidencia a capacidade de generalização dos modelos para novos dados.

Isto posto, dadas as configurações do K-Fold descritas na Seção 4.2.3, ambos modelos obtiveram soluções generalizadas, onde cada modelo teve melhor desempenho em determinados conjuntos de dados, sugerindo que, apesar do OCT-MIO se sobressair sobre o CART para métricas aferidas durante o treinamento, cada modelo pode ser mais adequado dependendo do problema.

Observou-se que, apesar do OCT-MIO obter soluções ótimas para as alturas mínima e mínima+1, exceto para o conjunto Academic, o CART ainda obteve melhor resultado de Acurácia Balanceada para altura mínima do conjunto Iris e obteve melhores resultados tanto de Acurácia, quanto de Acurácia Balanceada, para a altura mínima+1 do conjunto Credit.

Já para o conjunto Stress, o OCT-MIO não só encontrou a solução ótima em todas as 3 alturas, como também obteve valores melhores que o CART para as 3 métricas analisadas, com médias acima de 0.99, sugerindo que, o modelo teve bom desempenho em vários aspectos, pois, não apenas obteve altas taxas de acertos preditivos sobre dados para os quais ele não foi treinado, como tais acertos também tiveram um bom equilíbrio entre as classes, evitando que uma ou outra classe tivesse menos importância na classificação.

Por fim, ambos os modelos tiveram baixo desempenho sobre o conjunto Academic, devido a sua complexidade. O modelo OCT-MIO até obteve resultados melhores que o CART apenas para a altura mínima deste conjunto de dados, ainda assim, apesar do conjunto de treinamento utilizado no *K-Fold* ser menor que o conjunto completo, o OCT-MIO obteve valores cada vez piores, à medida que a altura máxima da árvore aumentava, a ponto de não conseguir obter alguma solução factível para a altura mínima+2.

Os resultados de experimentos aplicando-se K-Fold se mostraram, em geral, bastante variados, principalmente em função da dificuldade do conjunto de dados,

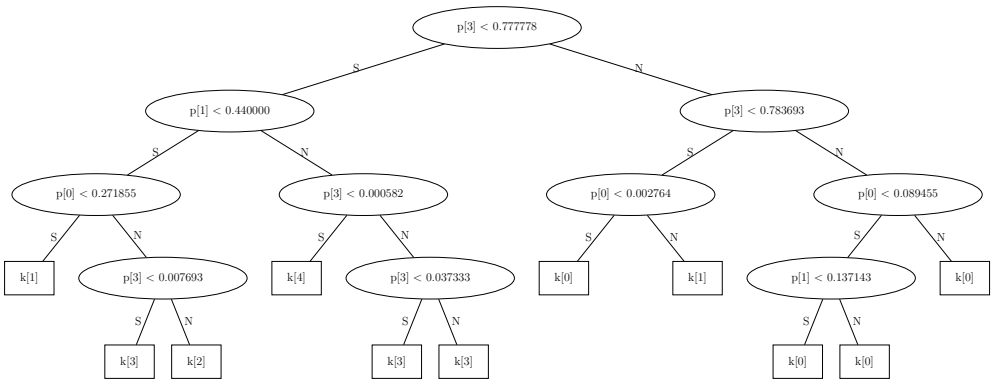

Figura 32 – Árvore de classificação para o conjunto Stress com altura mín. +1 pelo modelo OCT-MIO (Acc. 1.000)

sugerindo, portanto, uma equiparação dos dois modelos, a depender dos cenários nos quais eles forem utilizados. Ainda assim, há uma possível tendência em o OCT-MIO obter melhores desempenhos em acurácia para árvores de altura mínima, confome mostra a Tabela 13.

4.3.4 Análise Visual dos Diagramas das Árvores de Classificação

Os diagramas das árvores de classificação permitem analisar e interpretar os resultados dos modelos de forma gráfica e intuitiva, contribuindo, inclusive, para sugestão de ajustes e identificação de vieses. Isso é possível devido às características interpretáveis de modelos de árvore de classificação.

Conforme descrito na Seção 4.1, o crescimento das árvores foi determinado apenas pela altura máxima, permitindo que a árvore se divida continuamente até certa altura. Com isso, as divisões das árvores geradas podem ser revistas e ajustadas posteriormente à etapa de treinamento, como no caso da árvore Árvore de Classificação para o Conjunto Stress (100%) com Altura mín. +1 pelo modelo OCT-MIO na Figura 32 que poderia ser redefinida como na Figura 33, como um tipo de "poda" sem prejuízos em métricas, apenas unificando ramificações que levam à folhas de mesmo rótulo de classe.

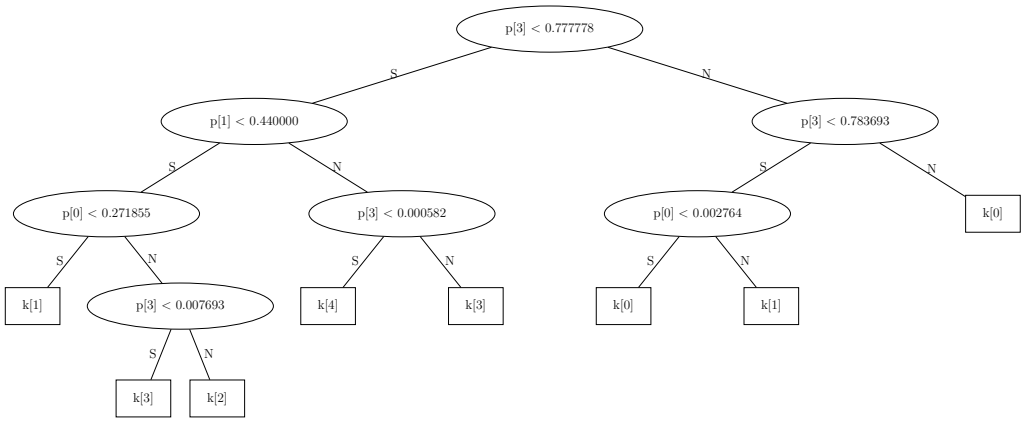

Figura 33 – Árvore de classificação (redefinida/"podada") para o conjunto Stress com altura mín. +1 pelo modelo OCT-MIO (Acc. 1.000)

As Árvores de Classificação para o Conjunto Stress (100%) com Altura Mínima até Altura Mínima +2 geradas pelo CART (Figura 34) são idênticas, uma vez que a árvore de altura mínima obteve acurácia de 1.000 e, portanto, não há necessidade de uma árvore de altura maior que a mínima para o mesmo conjunto de dados. O mesmo fenômeno já não acontece com a árvore gerada pelo OCT-MIO (Figura 35), possivelmente porque, tanto o OCT-MIO faz divisões baseado em todas as possibilidades, quanto, com $\alpha = 0.0$, nenhuma punição foi aplicada em função ao número de nós da árvore.

Já para o conjunto Iris, enquanto a Árvore de classificação com altura mín. +1 gerada pelo modelo CART (Figura 36) é mais simples, isto é, possui menos nós, a árvore equivalente gerada pelo modelo OCT-MIO (Figura 38), mesmo sendo gerada em apenas 5 minutos, possui melhor ajuste, com acurácia superior, conforme observado nos resultados da Seção 4.2.1, colocando o modelo OCT como eventual alternativa ao CART.

Se, por um lado, o fato de eventualmente o CART ficar preso a uma solução local pode contribuir para que o CART obtenha soluções mais simples e interpretáveis, por outro lado, isso impede que ele obtenha resultados melhores, principalmente a partir da altura mínima +1, como acontece para o conjunto Iris

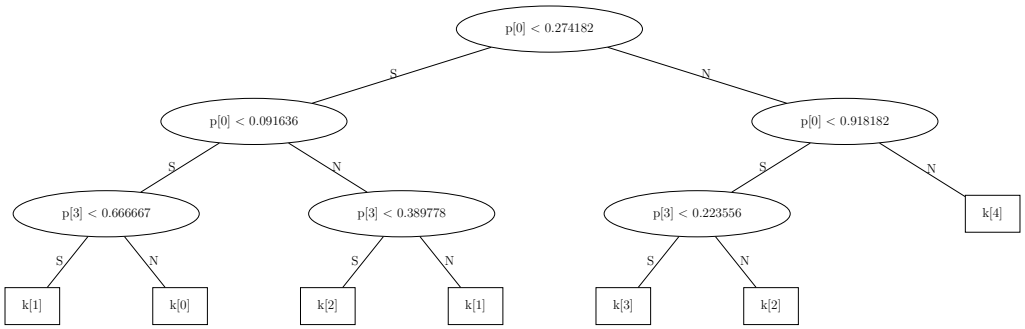

Figura 34 – Árvore de classificação para o conjunto Stress com altura mínima pelo modelo CART (Acc. 1.000)

Figura 35 – Árvore de classificação para o conjunto Stress com altura mín. +2 pelo modelo OCT-MIO (Acc. 1.000)

(100%) com altura mín. +1 (Figura 36) e mín. +2 (Figura 37), onde mesmo as árvores geradas em até 5 min pelo OCT-MIO se mostraram de 0.7% a 2.0% mais acuradas.

Em geral, as árvores geradas pelo OCT-MIO tem se mostrado mais complexas, ou seja, com maior número de nós, que as árvores geradas pelo CART, mesmo assim, as árvores do OCT-MIO ainda são interpretáveis, com alturas suficientes para serem compreendidas de forma visual.

Quanto às árvores geradas usando a técnica *K-Fold Cross Validation*, merecem destaque as árvores geradas pelo modelo OCT-MIO para o conjunto Stress que obtiveram acurácias superiores ao CART nas 3 variações de altura. Novamente, o CART fica preso à soluções ótimas locais, criando uma tendência a repetir as mesmas divisões para quaisquer alturas de árvore, impedindo-o de obter soluções

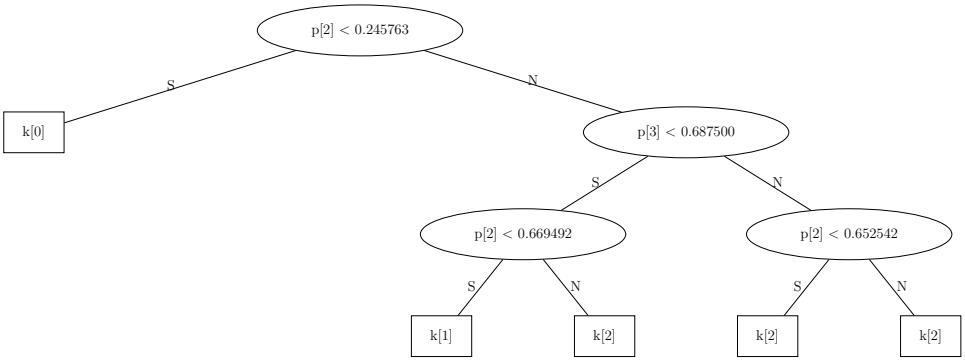

Figura 36 – Árvore de classificação para o conjunto Iris com altura mín. +1 pelo modelo CART (Acc. 0.973)

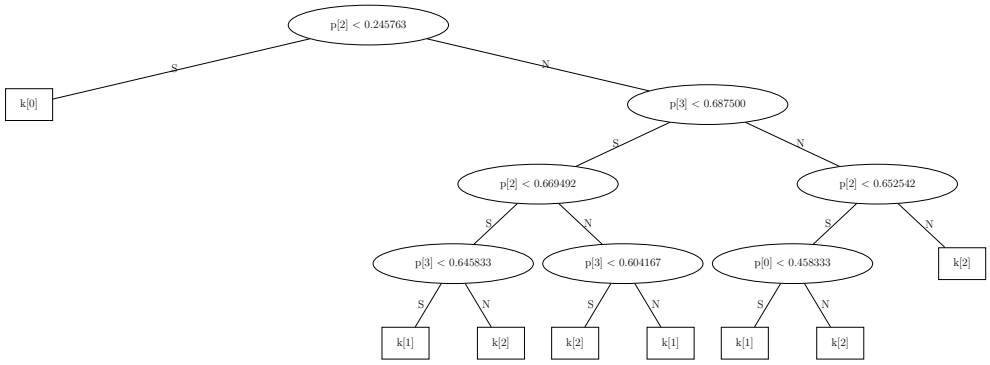

Figura 37 – Árvore de classificação para o conjunto Iris com altura mín. +2 pelo modelo CART (Acc. 0.993)

melhores, enquanto o OCT-MIO é capaz de encontrar as divisões que proporcionam igual ou melhor acurácia, conforme é possível ver comparando as árvores da Figura 41 e Figura 40; Figura 43 e Figura 42; E Figura 44 e Figura 45.

Por outro lado, para o conjunto Iris, o maior crescimento da árvore, gerado pelo modelo OCT-MIO, à medida que a altura máxima da árvore também aumenta, sugere algum grau de sobre ajuste (*overfitting*), evidenciado pela técnica de validação cruzada através da redução da acurácia da altura mínima+1 para a altura mínima+2, cujas árvores estão na sequência de imagens: Figura 46 e Figura 47; Figura 48 e

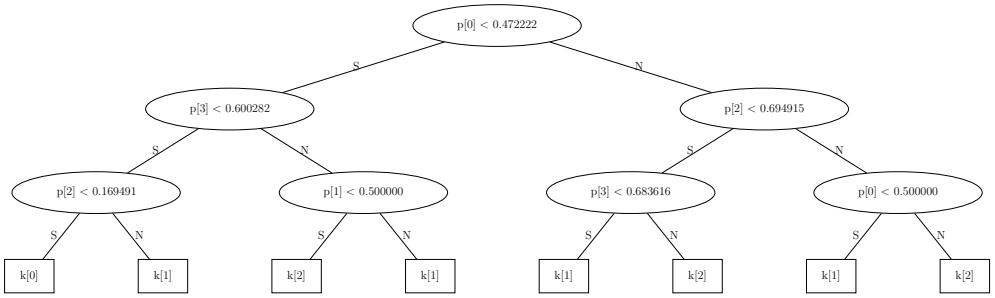

Figura 38 – Árvore de classificação para o conjunto Iris com altura mín. +1 pelo modelo OCT-MIO (Acc. 0.993)

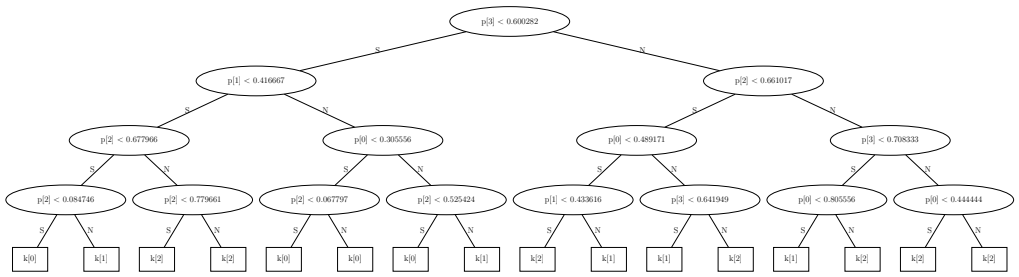

Figura 39 – Árvore de classificação para o conjunto Iris com altura mín. +2 pelo modelo OCT-MIO (Acc. 1.000)

Figura 49; E Figura 50 e Figura 51.

Por fim, a interpretabilidade inerente à uma árvore de classificação permite entender as principais diferenças entre duas árvores de mesmo tamanho e para o mesmo conjunto de dados com acurácias muito próximas, mas com tempos de treinamento muito diferentes, como é o caso da Figura 52 referente à Árvore de classificação para o conjunto Credit com altura mín. +2 pelo modelo OCT-MIO (Acc. 0.864) gerada após 10h de treinamento e da Figura 53 referente a outra árvore de mesma acurácia, porém treinada apenas por 5 minutos.

É possível perceber limiares similares, ainda que em nós diferentes entre elas, ratificando a assertiva de que a árvore gerada aos 5 minutos de treinamento já pode ser uma solução tão boa quanto à árvore gerada às 10 h e, portanto, ratifica

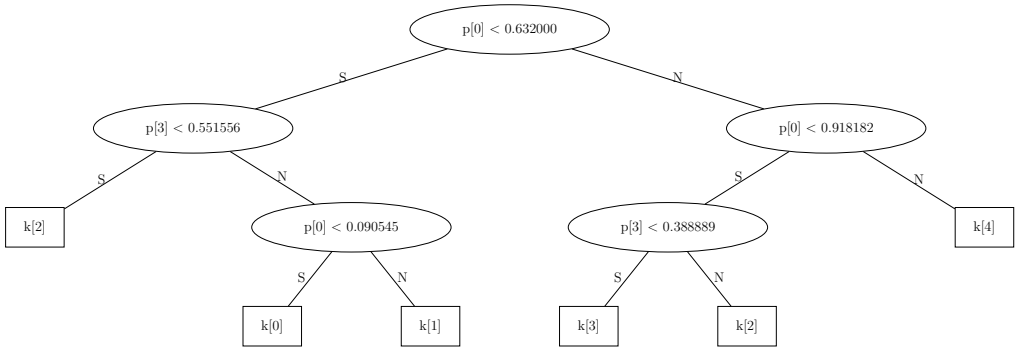

Figura 40 – Árvore de classificação para o conjunto Stress (80%) com altura mín. pelo modelo CART (Acc. média 0.987)

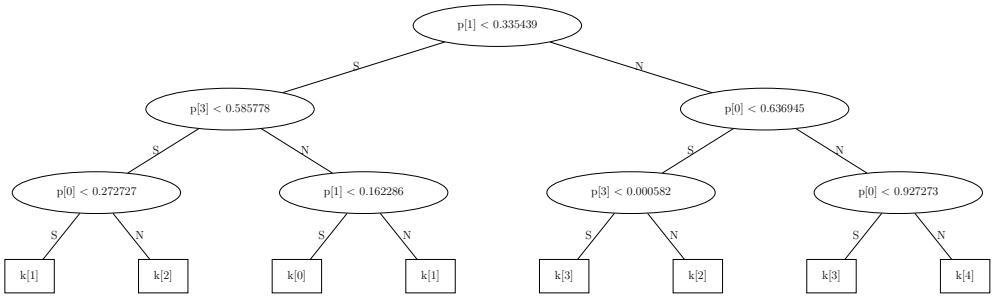

Figura 41 – Árvore de classificação para o conjunto Stress (80%) com altura mín. pelo modelo OCT-MIO (Acc. média 0.994)

que, apesar de ser necessário um tempo maior para provar a otimalidade de uma solução, é possível obter uma solução de igual desempenho em um tempo menor.

4.3.5 Teste de Wilcoxon

Não foi possível obter um valor para o teste de Wilcoxon pois o número de amostras ($n = k = 5$, no K-Fold) em cada teste (para cada conjunto de dados, e cada altura de árvore) não possui valor estatístico crítico correspondente na Tabela de Valores Críticos do Teste dos Postos Sinalizados de Wilcoxon.

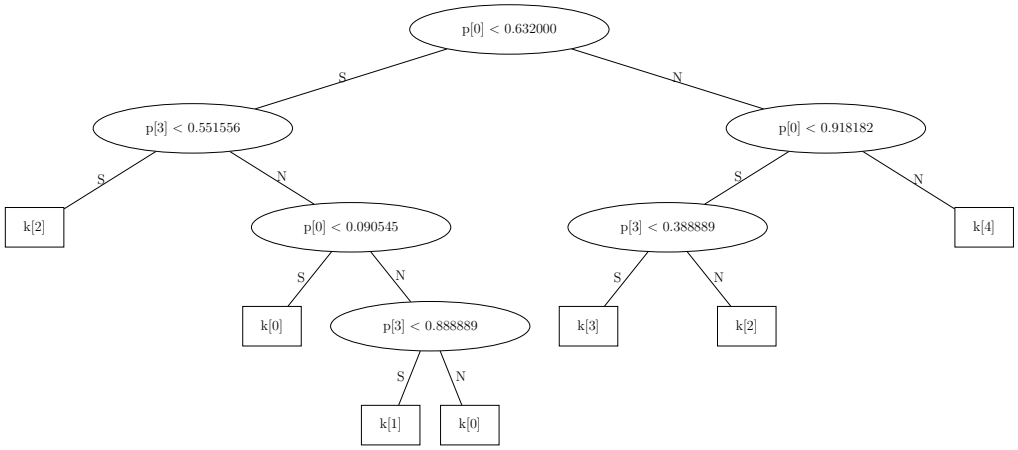

Figura 42 – Árvore de classificação para o conjunto Stress (80%) com altura mín. +1 pelo modelo CART (Acc. média 0.987)

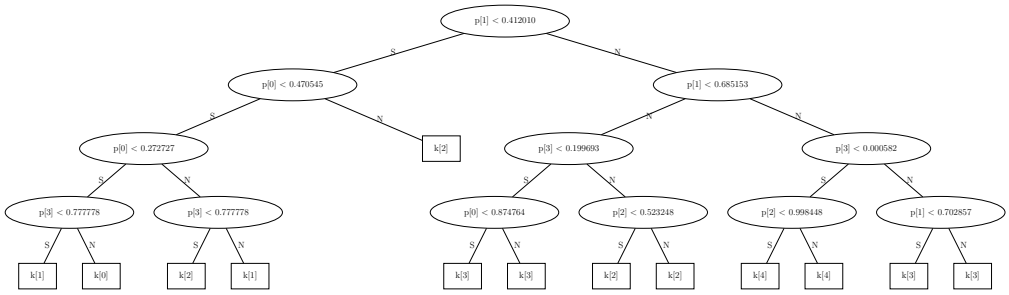

Figura 43 – Árvore de classificação para o conjunto Stress (80%) com altura mín. +1 pelo modelo OCT-MIO (Acc. média 0.992)

Experimentos futuros podem executar a validação cruzada usando $n = K = 10$, pois para assim, haverá valores correspondentes na Tabela de Valores Críticos do Teste dos Postos Sinalizados de Wilcoxon. Em outras palavras, para $n = 10$, há amostras suficientes para aceitar ou rejeitar a hipótese nula dos experimentos deste trabalho.

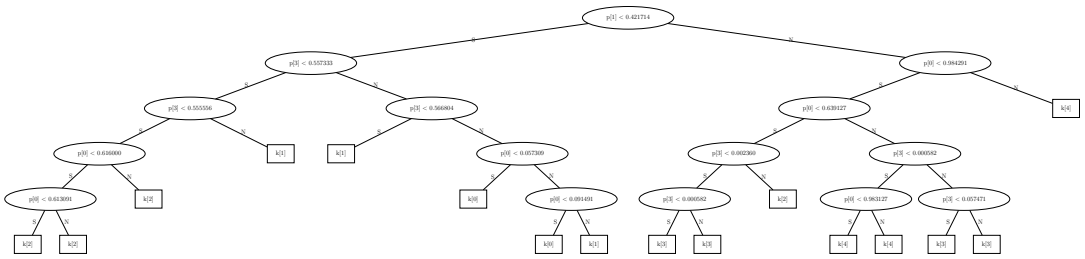

Figura 44 – Árvore de classificação para o conjunto Stress (80%) com altura mín. +2 pelo modelo OCT-MIO (Acc. média 0.992)

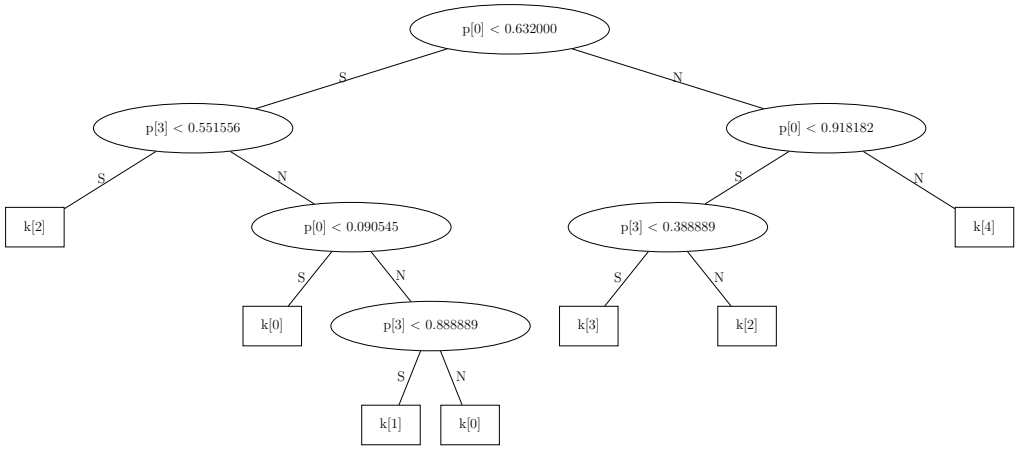

Figura 45 – Árvore de classificação para o conjunto Stress (80%) com altura mín. +2 pelo modelo CART (Acc. média 0.987)

4.4 Discussão dos Resultados

Os resultados mostraram que, de forma geral, o modelo OCT-MIO se ajusta aos dados de treinamento melhor que o CART. Isso possivelmente se dá principalmente pela capacidade do OCT-MIO de encontrar soluções ótimas globais e, assim, gerar árvores que refletem melhor os padrões inerentes aos dados. Contudo, apesar deste ajuste soar interessante, é possível que, em alguns casos, a otimalidade do OCT-MIO cause algum excesso de ajuste (*overfitting*) do modelo, prejudicando levemente seu desempenho na classificação dos dados de teste. Ainda assim, há

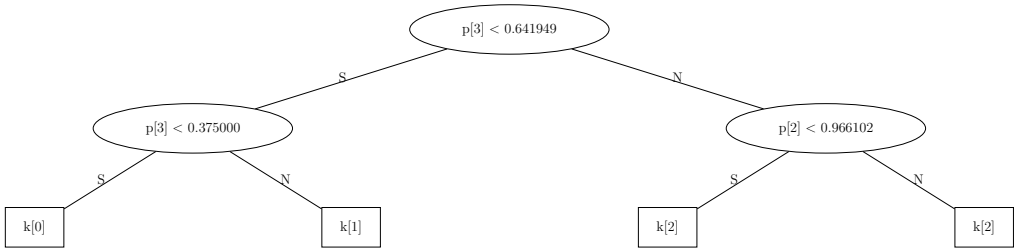

Figura 46 – Árvore de classificação para o conjunto Iris (80%) com altura mín. pelo modelo OCT-MIO (Acc. média 0.947)

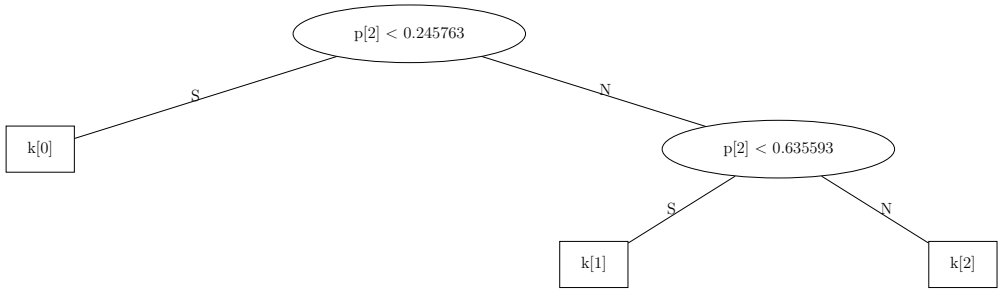

Figura 47 – Árvore de classificação para o conjunto Iris (80%) com altura mín. pelo modelo CART (Acc. média 0.947)

casos onde o OCT-MIO obtém igual ou melhor Acurácia que o CART, tanto em treinamento quanto em teste. Portanto, como principal resultado, apesar do OCT-MIO só garantir métricas superiores ao CART apenas sobre os dados de treinamento e considerando apenas soluções ótimas, ainda assim, o OCT-MIO tende a encontrar soluções equivalentes ou melhores que o CART, quanto menor a altura da árvore. Isto significa que, considerando as árvores de classificação de altura mínima para um problema e, portanto, mais interpretáveis, o OCT-MIO tende a obter resultados de Acurácia sobre os dados de teste iguais ou melhores que o CART.

Sem dúvida o CART é mais rápido que o OCT-MIO em tempo de trei-

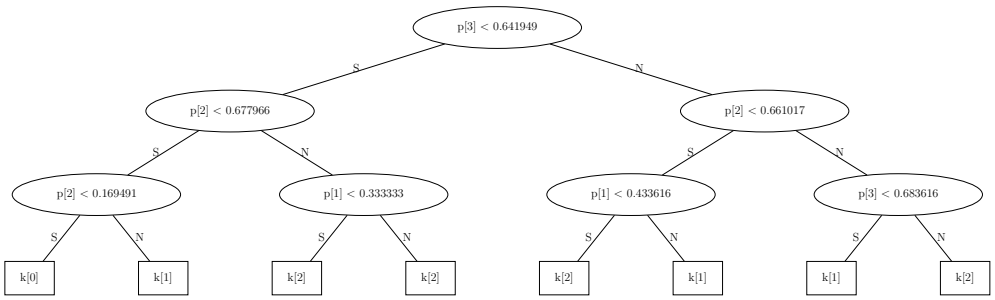

Figura 48 – Árvore de classificação para o conjunto Iris (80%) com altura mín. +1 pelo modelo OCT-MIO (Acc. média 0.960)

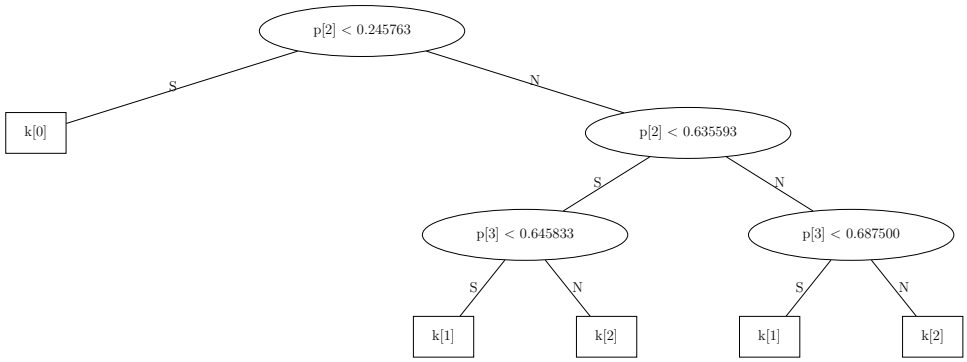

Figura 49 – Árvore de classificação para o conjunto Iris (80%) com altura mín. +1 pelo modelo CART (Acc. média 0.960)

namento do modelo, colocando o CART como melhor opção entre os dois neste quesito. A dificuldade de um problema pode fazer o tempo de treinamento do OCT-MIO aumentar exponencialmente, tornado-o inviável para eventuais cenários com restrição de tempo e capacidade computacional. Observou-se que o tempo de treinamento (otimização) do OCT-MIO aumenta em função de alguns fatores como:

- Número de elementos de um conjunto;
- Número de atributos de um conjunto;

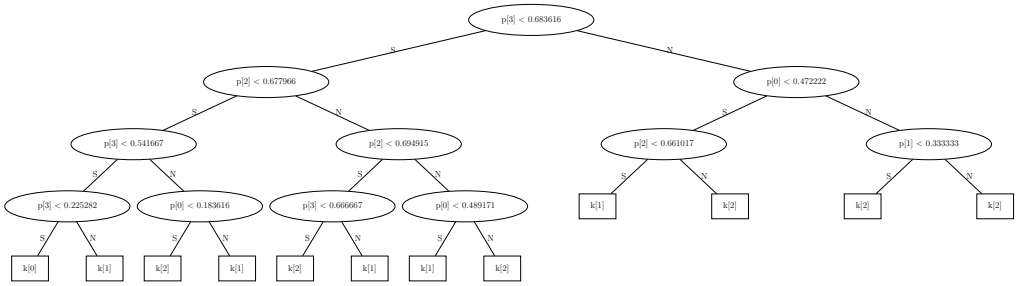

Figura 50 – Árvore de classificação para o conjunto Iris (80%) com altura mín. +2 pelo modelo OCT-MIO (Acc. média 0.940)

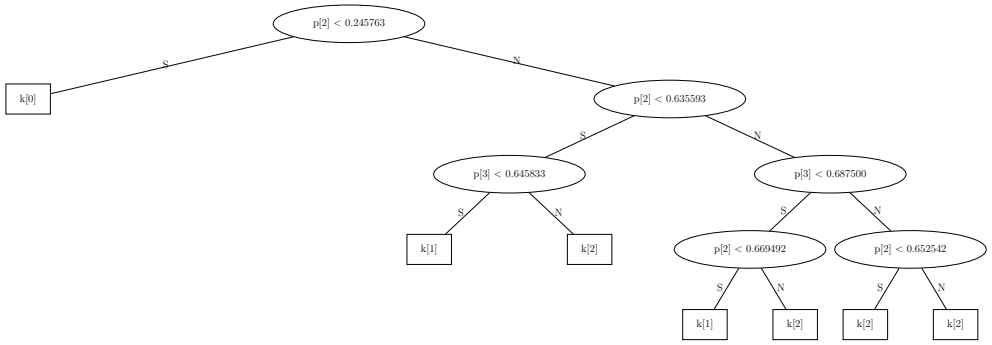

Figura 51 – Árvore de classificação para o conjunto Iris (80%) com altura mín. +2 pelo modelo CART (Acc. média 0.953)

- Número de classes do problema;
- Dispersão dos elementos do conjunto dentro do hiperespaço de atributos.

Cada um destes fatores interfere conjuntamente na dificuldade do problema e, consequentemente, no tempo de treinamento do OCT-MIO. Entretanto, em relação a altura máxima da árvore, o tempo de treinamento nem sempre obedece uma relação direta, podendo ser menor mesmo para uma altura maior.

Contudo, seja para treinamento com o conjunto de dados completo ou estratificado, o OCT-MIO defende sua competitividade, ainda que como um modelo heurístico, ao obter obter soluções factíveis, em um tempo reduzido (poucas horas,

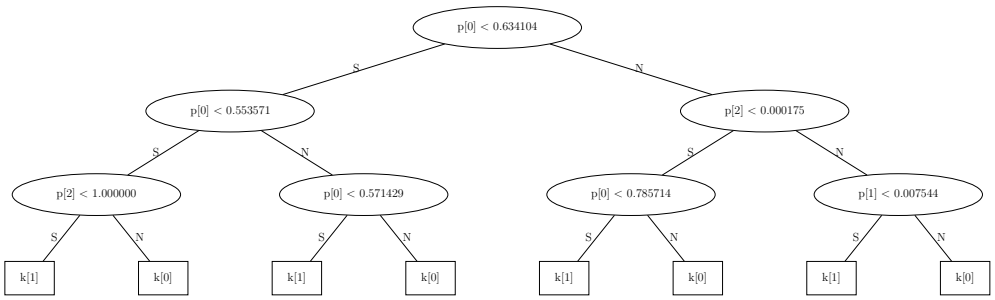

Figura 52 – Árvore de classificação para o conjunto Credit com altura mín. +2 pelo modelo OCT-MIO em 10 h de Treinamento (Acc. 0.864)

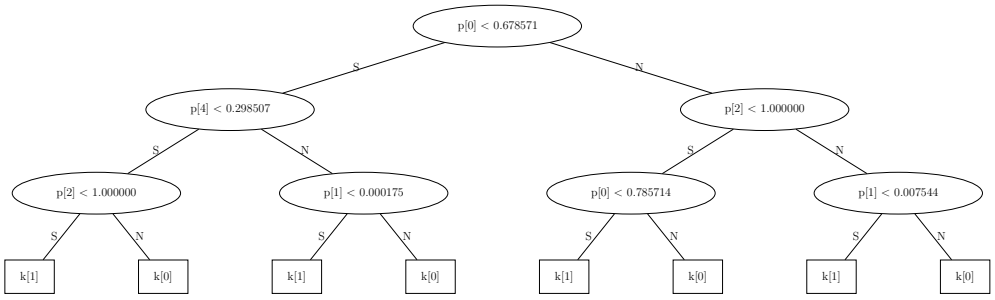

Figura 53 – Árvore de classificação para o conjunto Credit com altura mín. +2 pelo modelo OCT-MIO em 5 min. de Treinamento (Acc. 0.862)

minutos ou apenas segundos), alcançando, em alguns casos, métricas iguais ou ainda sutilmente superiores ao CART.

Ao comparar os dois modelos, foi possível perceber que, dado um problema de 2 classes, onde, consequentemente, a altura mínima é igual a 1, quaisquer resultados ótimos locais são também resultados ótimos globais. Portanto, para a solução de problemas de duas classes com altura mínima (1), o CART mostra-se o modelo mais adequado, uma vez que ele irá obter o resultado ótimo em menos tempo que o OCT-MIO.

4.5 Limitações dos Experimentos

Problemas de otimização naturalmente podem demorar de alguns segundos até algumas horas ou ainda mais até a prova da solução ótima em função das restrições e do tamanho do problema abordado. Portanto o tempo dispendido para a solução destes problemas está diretamente associado aos limites da capacidade computacional do hardware.

Inclusive, a ocorrência de falhas na execução de alguns experimentos possivelmente foram causadas por limitações de hardware e de tempo, uma vez que, dadas as capacidades computacionais descritas na seção de experimentos, o tempo máximo definido para otimização não foi suficiente para obtenção de uma solução factível.

Apesar dos conjuntos de dados utilizados serem conjuntos reais diversos e de contexto crítico (com exceção do Iris), este trabalho restringiu-se a analisar 4 conjuntos de dados. Os parâmetros dos modelos também foram fixados, variando-se apenas a altura das árvores geradas por cada um. E alguns diagramas de árvores de classificação podem ter sua escala afetada devido ao seu tamanho e complexidade.

Também, os gráficos referentes à Tabela 10, Tabela 11 e Tabela 12, assim como alguns gráficos de convergência da função objetivo da Seção 4.2.2, não foram apresentados, visando maior síntese deste documento. Além disso, os tempos intermediários para avaliação de árvores factíveis, quando o OCT-MIO é aplicado como heurística, foram restritos a 3 (5min, 5 horas e 10 horas) em função do tempo disponível para conclusão deste trabalho.

É importante destacar que os experimentos realizados neste trabalho são diferentes dos experimentos realizados por Bertsimas e Dunn (2017), no artigo *Optimal classification trees*, onde os dados foram divididos em 50% para treinamento, 25% validação e 25% teste, com ajustes de parâmetros (α e mínimo de elementos por folha) sobre 75% dos dados (treinamento e teste), além de alturas máximas diferentes.

Por fim, quaisquer eventuais artefatos deste trabalho que não estejam contidos neste documento podem ser solicitados ao autor sem nenhum constrangimento.

5 Conclusão

A implementação dos modelos CART e OCT-MIO permitiu profundo entendimento dos seus respectivos mecanismos e estratégias, com isso observou-se que, apesar de ambos os modelos implementarem função objetivo de maneiras diferentes, ambos modelos objetivam alcançar o menor número de erros na classificação de elementos do conjunto, tornando-os passíveis de comparação através da métrica Acurácia, a qual mede a taxa de classificação corretas e, por consequência também incorretas, do modelo.

Ainda que, sobre os dados de treinamento, o modelo OCT-MIO tenha obtido resultados de Acurácia, Acurácia Balanceada e Macro Pontuação F1, em geral, iguais ou melhores que o CART, evidenciando que as árvores geradas pelo treinamento do OCT-MIO são mais ajustadas aos dados e possuem nível de acerto mais equilibrado entre as classes, quando as mesmas métricas são analisadas sobre os dados de teste, para os quais o modelo não foi treinado, o OCT-MIO nem sempre se sobressai sobre o CART, sugerindo que o ajuste melhor do modelo eventualmente pode impactar negativamente na capacidade de generalização do mesmo, especialmente quanto maior a altura máxima da árvore.

Apesar dos modelos se alterarem em acurácia na maioria dos casos, o modelo OCT-MIO mostrou igual ou melhor desempenho em acurácia quanto menor a altura máxima da árvore. Obtendo acurácia igual ou superior ao CART em todos os conjuntos de dados testados onde a altura da árvore era a mínima necessária para classificação de todas as classes do problema. Com isso conclui-se que o OCT-MIO pode ser melhor alternativa para árvores de classificação em cenários onde deseja-se que:

- A altura da árvore resultante seja a menor possível, proporcionando maior interpretabilidade;
- Todas as classes do problema tenham pesos equivalentes na classificação;
- O tempo de treinamento do classificador seja flexível.

Contudo, as limitações destes experimentos ainda não foram suficientes para ratificar se, em geral, o modelo OCT-MIO supera o CART em termos de Acurácia, Acurácia Balanceada e Macro Pontuação F1.

Ainda assim, o OCT-MIO é uma alternativa que pode fazer a diferença entre, por exemplo, mais ou menos pacientes diagnosticados com nível de ansiedade incorreta, mais ou menos clientes classificados equivocadamente como potencialmente inadimplentes, mais ou menos alunos que ainda poderiam se graduar, mas estão sujeitos ao desligamento ou não foram identificados para receber apoio pedagógico ou psicológico. Esta mesma alternativa poderia fazer a diferença em diversos outros cenários críticos, onde qualquer classificação incorreta poderia trazer grandes prejuízos.

6 Trabalhos futuros

O presente trabalho analisou e comparou 2 modelos de aprendizado de máquina que baseado em árvores de decisão aplicados em alguns conjuntos de dados reais. Com a realização deste trabalho foram identificados variações e extensões do mesmo que poderiam contribuir ainda mais para o avanço do conhecimento nesta área, como por exemplo:

- ***Warm starts***: Estudar o modelo de aprendizado OCT-MIO utilizando *warm starts* (partidas quentes), pois os experimentos levam a crer que os resultados dos experimentos seria ainda melhores.
- **Mais conjuntos de dados**: Utilizar uma maior amostragem de conjuntos de dados poderia dar uma visão mais robusta sobre o desempenho e limitações dos modelos.
- **Maior variedade de parâmetros**: Explorar uma maior variedade de parâmetros nos experimentos também poderiam incrementar ainda mais os resultados. Neste sentido, sugere-se o uso da técnica *Grid Search*.
- **Diversificação de resolvedores**: A implementação dos modelos em outras linguagens de programação assim como a utilização de resolvedores diferentes também podem proporcionar avanços significativos ao modelos de otimização tal como o OCT-MIO.
- **Decisões multivariáveis:** Acredita-se ser viável também a implementação de modelos de árvore de classificação que utilizam decisões multivariáveis, isto é, tomam uma decisão sobre um valor composto ou gerado a partir de dois ou mais atributos.
- **Dados Originais:** Trabalhos futuros poderão reverter os valores normalizados para os dados originais para melhor interpretabilidade e intuitividade dos modelos de classificação.

- **Teste de Wilcoxon:** Aumentar a validação cruzada para $K = 10$ e aplicar o teste de Wilcoxon.

Referências

AGHAEI, S.; GÓMEZ, A.; VAYANOS, P. Strong optimal classification trees. *Operations Research*, INFORMS, 2024. Citado na página 38.

BERTSIMAS, D.; DUNN, J. Optimal classification trees. *Machine Learning*, Springer, v. 106, n. 7, p. 1039–1082, 2017. Citado 15 vezes nas páginas 27, 29, 33, 35, 58, 60, 62, 67, 68, 69, 70, 71, 72, 73 e 131.

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: Routledge, 2017. Citado 11 vezes nas páginas 35, 37, 45, 46, 47, 54, 55, 56, 57, 60 e 71.

CARBONELL, J. G.; MICHALSKI, R. S.; MITCHELL, T. M. An overview of machine learning. *Machine learning*, Elsevier, p. 3–23, 1983. Citado na página 33.

CHU, X. et al. Data cleaning: Overview and emerging challenges. In: *Proceedings of the 2016 international conference on management of data*. [S.l.: s.n.], 2016. p. 2201–2206. Citado 2 vezes nas páginas 90 e 91.

CORTINHAS, S. *Credit Card Approval Clean Data*. 2022. <https://www.kaggle.com/datasets/samuelcortinhas/credit-card-approval-clean-data>. Kaggle Dataset. Citado na página 87.

COSTA, V. G.; PEDREIRA, C. E. Recent advances in decision trees: An updated survey. *Artificial Intelligence Review*, Springer, v. 56, n. 5, p. 4765–4800, 2023. Citado 2 vezes nas páginas 27 e 28.

DAS, S. et al. Taxonomy and survey of interpretable machine learning method. In: IEEE. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.], 2020. p. 670–677. Citado na página 40.

DASH, M.; LIU, H. Feature selection for classification. *Intelligent data analysis*, Elsevier, v. 1, n. 1-4, p. 131–156, 1997. Citado na página 92.

FISHER, R. A. *Iris*. 1988. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56C76. Citado na página 86.

GRANDINI, M.; BAGLI, E.; VISANI, G. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020. Citado 11 vezes nas páginas 73, 74, 75, 76, 77, 78, 79, 80, 81, 82 e 83.

KHALID, N. H. M. et al. Performance comparison of feature selection methods for prediction in medical data. In: SPRINGER. *International Conference on Soft Computing in Data Science*. [S.l.], 2023. p. 92–106. Citado 2 vezes nas páginas 92 e 93.

KHALIL, M. *Students Dropout and Academic Success Dataset*. 2022. <https://www.kaggle.com/datasets/mahwiz/students-dropout-and-academic-success-dataset>. Kaggle Dataset. Citado na página 88.

KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: MONTREAL, CANADA. *Ijcai*. [S.l.], 1995. v. 14, n. 2, p. 1137–1145. Citado 3 vezes nas páginas 83, 84 e 108.

LAAVANYA, R. *Human Stress Detection in and through Sleep*. 2022. <https://www.kaggle.com/datasets/laavanya/human-stress-detection-in-and-through-sleep>. Kaggle Dataset. Citado na página 86.

LINARDATOS, P.; PAPASTEFANOPOULOS, V.; KOTSIANTIS, S. Explainable ai: A review of machine learning interpretability methods. *Entropy*, MDPI, v. 23, n. 1, p. 18, 2020. Citado 3 vezes nas páginas 27, 39 e 40.

LINDEN, J. van der; WEERDT, M. de; DEMIROVIć, E. Necessary and sufficient conditions for optimal decision trees using dynamic programming. In: OH, A. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2023. v. 36, p. 9173–9212. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2023/file/1d5fce9627e15c84db572a66e029b1fc-Paper-Conference.pdf>. Citado na página 38.

MOHSENI, S.; ZAREI, N.; RAGAN, E. D. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, ACM New York, NY, v. 11, n. 3-4, p. 1–45, 2021. Citado 2 vezes nas páginas 39 e 43.

MOLNAR, C. *Interpretable machine learning*. [S.l.]: Lulu. com, 2020. Citado 3 vezes nas páginas 41, 42 e 43.

NASTESKI, V. An overview of the supervised machine learning methods. *Horizons. b*, v. 4, n. 51-62, p. 56, 2017. Citado na página 34.

PAGANO, T. P. et al. Bias and unfairness in machine learning models: A systematic review on datasets, tools, fairness metrics, and identification and mitigation methods. *Big Data and Cognitive Computing*, v. 7, n. 1, 2023. ISSN 2504-2289. Disponível em: <https://www.mdpi.com/2504-2289/7/1/15>. Citado na página 75.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011. Citado na página 92.

QUINLAN, J. R. *Credit Approval*. 1987. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5FS30. Citado na página 87.

RACHAKONDA, L. et al. Sayopillow: Blockchain-integrated privacy-assured iomt framework for stress management considering sleeping habits. *IEEE Transactions on Consumer Electronics (TCE)*, v. 67, n. 1, p. 20–29, Feb 2021. Citado na página 86.

RAO, K. M.; SAIKRISHNA, G.; SUPRIYA, K. Data preprocessing techniques: emergence and selection towards machine learning models-a practical review using hpa dataset. *Multimedia Tools and Applications*, Springer, v. 82, n. 24, p. 37177–37196, 2023. Citado 2 vezes nas páginas 90 e 91.

SATHYA, R.; ABRAHAM, A. et al. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, Citeseer, v. 2, n. 2, p. 34–38, 2013. Citado 2 vezes nas páginas 33 e 34.

SINGH, D.; SINGH, B. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, Elsevier, v. 97, p. 105524, 2020. Citado na página 91.

TOUATI, Y. E.; SLIMANE, J. B.; SAIDANI, T. Adaptive method for feature selection in the machine learning context. *Engineering, Technology & Applied Science Research*, v. 14, n. 3, p. 14295–14300, 2024. Citado 2 vezes nas páginas 92 e 93.

VERWER, S.; ZHANG, Y. Learning optimal classification trees using a binary linear program formulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, n. 01, p. 1625–1632, Jul. 2019. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/3978>. Citado na página 38.

WILCOXON, F. Individual comparisons by ranking methods. In: _____. *Breakthroughs in Statistics: Methodology and Distribution*. New York, NY: Springer New York, 1992. p. 196–202. ISBN 978-1-4612-4380-9. Disponível em: <https://doi.org/10.1007/978-1-4612-4380-9_16>. Citado na página 85.

Apêndices