Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

ÁRVORES HÍBRIDAS: PARTICIONAMENTO RECURSIVO BASEADO EM MODELOS PARAMÉTRICOS

Vinicius Hideki Yamada Santiago

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Árvores Híbridas: Particionamento Recursivo Baseado em Modelos Paramétricos

Vinicius Hideki Yamada Santiago
Orientador: Prof. Dr. Afrânio Márcio Corrêa Vieira

Trabalho de Conclusão de Curso apresentado ao Departamento de Estatística da Universidade Federal de São Carlos - DEs-UFSCar, como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Abril de 2022

Federal University of São Carlos
Exact and Technology Sciences Center
Department of Statistics

Hybrid Trees: Recursive Partitioning Based on Parametric Models

Vinicius Hideki Yamada Santiago
Advisor: Prof. Dr. Afrânio Márcio Corrêa Vieira

Bachelor dissertation submitted to the Department of Statistics, Federal University of São Carlos - DEs-UFSCar, in partial fulfillment of the requirements for the degree of Bachelor in Statistics.

São Carlos
April 2022

Vinicius Hideki Yamada Santiago

Árvores Híbridas: Particionamento Recursivo Baseado em Modelos Paramétricos

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Vinicius Hideki Yamada Santiago e aprovado pela banca examinadora.

Aprovado em 12 de Abril de 2022

Banca Examinadora:

- Prof. Dr. Afrânio Márcio Corrêa Vieira (Orientador)
- Prof. Dr. Anderson Luiz Ara Souza
- Prof. Dr. Michel Helcias Montoril

Agradecimentos

"Primeiramente agradeço à minha família me tornar quem sou hoje. Agradeço especialmente à minha mãe e ao meu pai, Vilma e Valdecir, e aos meus tios, Vânia e Márcio. Sem vocês, eu não seria nem 10% de quem sou, e graças a vocês, eu tenho orgulho de quem estou me tornando.

Agradeço também ao Prof° Dr. Afrânio Vieira, pelas longas discussões (que até mesmo passavam do tempo) sobre metodologias, o que está certo e errado, o que seria melhor tirar ou acrescentar ao texto, sobre carreiras, sobre a vida, etc.

Aos professores da banca, Dr. Anderson Ara e Dr. Michel Montoril, pelas ótimas críticas, sugestões e dicas em relação meu texto. E pelo tempo dedicado para ler meu trabalho e assistir a minha apresentação.

Aos professores do Departamento de Estatística, por me ensinarem técnicas, métodos, teorias, por me fornecerem esse cinto de ferramentas chamado Estatística.

Por fim, ao Uniforme, pelas peladas, momento de "recapitulation", discussões sobre a matéria, pela estimação, por tornarem esses anos de graduação mais leves e divertidos."

Resumo

O presente estudo busca explorar árvores híbridas, formalmente chamadas de particionamentos recursivos baseados em modelos paramétricos (*model-based recursive partitioning*). Tal método baseia-se em árvores de decisão tradicionais (árvore de regressão e classificação) e em modelos paramétricos (modelos lineares generalizados). Com isso, espera-se explicar o funcionamento das árvores híbridas, tanto em aspectos teóricos quanto em implementações computacionais. Por fim, será aplicado a técnica de árvores híbridas em um problema real relativo a perdas pré-abate de frangos de corte, enunciando os resultados obtidos, de modo a evidenciar as vantagens e desvantagens de se utilizar referida técnica. Como conclusão, percebeu-se que *model-based recursive partitioning* é uma técnica mais bem usufruída se aplicada em bases de dados de tamanho pequeno ou médio.

Palavras-chave: *árvore híbrida, árvore de decisão, modelo linear generalizado.*

Abstract

The present study seeks to explore hybrid trees, so called called recursive partitioning based on parametric models. Such method is based on traditional decision trees (regression and classification tree) and parametric models (generalized linear models). With this, it is expected to explain the functioning of hybrid trees, both in theoretical aspects and in implementations computational. Finally, the hybrid trees technique will be applied to a real problem concerning pre-slaughter losses of broilers, stating the obtained results, in order to highlight the advantages and disadvantages of this technique. As a conclusion, it was noticed that *model-based recursive partitioning* is a technique best used if applied to small to medium-sized databases.

Keywords: *hybrid tree, decision tree, generalized linear model, CART, regression tree.*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A utilização de algoritmos de aprendizado de máquina (*machine learning*) vem crescendo bastante nos últimos anos. Dentre esses algoritmos, *Random Forest* e *Extreme gradient boosting* são ferramentas extremamente poderosas no quesito preditivo (Marchese Robinson *et al.*, 2017; González *et al.*, 2020). Estas são extensões do modelo de árvores de decisão, também conhecido como árvores de classificação e regressão, ou CART© (Breiman *et al.*, 1984).

Em relação ao CART©, James *et al.* (2013); Strobl *et al.* (2009) discorreram sobre as vantagens que o referido método possui. Dentre elas, destaca-se a fácil visualização (desde de árvore esteja podada) e compreensão da árvore, a semelhança com o processo de decisão humano e a possibilidade de interação de ordens superiores serem modeladas facilmente, bem como relações não lineares. Todavia, o CART© nem sempre apresenta vantagens, há alguns casos em que ocorre um sobreajuste da árvore em relação à base de treinamento, fazendo com que que a acurácia preditiva seja prejudicada.

Outra técnica extremamente poderosa, mas agora no quesito interpretativo de um dado problema, são os modelos paramétricos, mais especificamente os modelos lineares generalizados. Esses modelos são excelentes para verificar de que forma determinada variável preditora influencia na variável resposta (McCullagh e Nelder, 2019). Entretanto, quando há muitas variáveis preditoras e/ou a interação entre elas é muito complexa, a interpretação dos coeficientes é dificultada, tornando árduo o uso de somente modelos lineares generalizados. Logo, o poder preditivo é limitado quando há relações não lineares e interações de ordem superior que não estão presentes no preditor linear do modelo (Strobl *et al.*, 2009).

De forma a combinar tanto os benefícios do CART© e de modelos lineares genera-

lizados, Zeileis *et al.* (2008) propôs o uso de particionamentos recursivos baseados em modelos paramétricos (*model-based recursive partitioning*, ou simplesmente MOB), nomeado como árvore híbrida para o presente trabalho. Esse método mescla árvores de decisão, resgatando assim o caráter visual facilitado, com modelos paramétricos, recobrando a interpretação inteligível dos fatores de interesse.

Sendo assim, este estudo pretende entender e divulgar o funcionamento de árvores híbridas, desde o processo de estimação do modelo paramétrico de cada nó e testes de instabilidade dos parâmetros, até a etapa de particionamento e poda. Além disso, a árvore híbrida será aplicada em uma base de dados real, a qual há informações sobre perdas pré-abate de frangos de corte (Vieira *et al.*, 2015), de modo a investigar os particionamentos criados e interpretar os modelos lineares generalizados de cada nó.

Por conseguinte, a árvore híbrida poderá auxiliar os ramos de avicultura e bem-estar animal, difundindo-se pela comunidade acadêmica.

Como objetivos principais deste trabalho, pode-se destacar,

- Explicar, em linguagem acessível, o algoritmo e a teoria por trás da técnica de árvore híbrida.

- Aplicar árvores híbridas em um problema real e enunciar os resultados obtidos;

1.1 Árvore de decisão

Árvore de decisão é uma técnica que consiste em um processo (algoritmo) que auxilia na tomada decisão. O nome árvore vem do fato desse algoritmo retornar uma representação visual que se assemelha a uma árvore, mais especificamente a uma folha com nós e ramificações. Um exemplo ilustrativo de árvore de decisão pode ser encontrada na Figura 1.1.

De forma a facilitar o entendimento e linguagem de uma árvore de decisão, pode-se destacar alguns termos recorrentes:

- **Nó**: É um momento da árvore em que pode haver um particionamento, isto é, uma ramificação em que o nó se divide. Na Figura 1.1, alguns exemplos de nós seriam os textos “Eu quero fazer uma caminhada?”, “Eu dormi bem na noite anterior?” e “Deixe pra amanhã”.

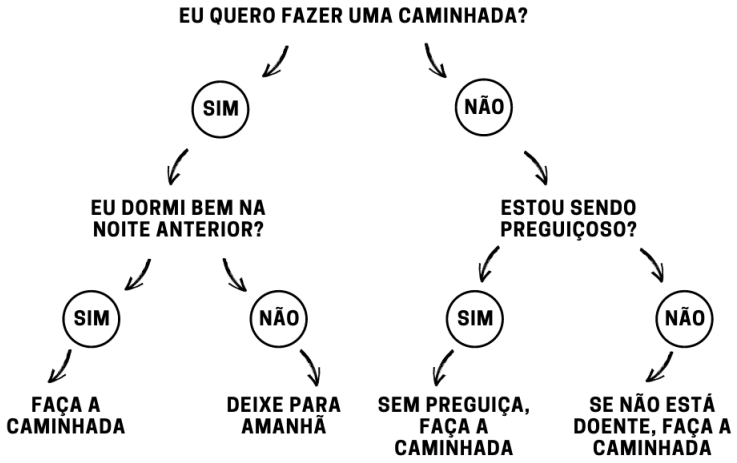

Figura 1.1: Exemplo ilustrativo de árvore de decisão sobre o ato de fazer uma caminhada.
Fonte: Tabela gerada pelo próprio autor.

- **Nó pai**: É um nó que origina outro nó, isto é, o nó vem vem antes de outro nós. Na Figura 1.1, alguns exemplos de nós pais seriam os textos "Eu quero fazer uma caminhada?" e "Eu dormi bem na noite anterior?". Lembrando que o termo nó pai é relativo a algum nó filho.

- **Nó filho**: É um nó que se originou de um nó pai. Na Figura 1.1, um nó filho seria "Faça a caminhada", uma vez que esse se origina do nó pai "Eu dormi bem na noite anterior", e este se origina do nó pai "Eu quero fazer uma caminha".

- **Nó final**: É o nó que está ao término da árvore, isto é, no fim dela, no momento em que a mesma para de crescer. Pode-se comentar que todo nó final é um nó filho (pois foi originado de algum nó anterior), mas nem todo nó filho é um nó final. Na Figura 1.1, alguns exemplos de nós finais seriam "Faça a caminha" e "Deixe para amanhã".

- **Ramificação**: É uma linha que liga um nó pai com um nó filho. Na Figura 1.1, as ramificações seriam os termos "Sim" e "Não".

Percebe-se que a árvore ajuda no processo de decisão. No início, há um questionamento inicial que deve ser respondido. Em seguida, um nó é particionado em novos nós com base nas perguntas e respostas. Em seguida, chega-se aos nós finais respondendo a pergunta inicial, daí o nome árvore de decisão.

O Algoritmo 1 generaliza o funcionamento computacional de uma árvore de decisão. Perceba que a árvore em si é um algoritmo recursivo, por isso o nome "particionamentos recursivos", ou seja, há o particionamento iterativo da árvore sendo executado diversas vezes (Carvalho *et al.*, 2011).

Algoritmo 1: Algoritmo genérico de uma árvore de decisão

Entrada: Base de dados $\boldsymbol{D}$

Saída: Árvore treinada

Função **GeraÁrvore($\boldsymbol{D}$);**

início

```
inicialização;
if critério de parada(D) = Verdadeiro then
    execute alguma operação;
    pare o crescimento da árvore;
else
    escolha uma covariável que otimiza determinado critério;
    particione de alguma forma o nó atual com base na covariável escolhida;
end
```

fim

Ainda no Algoritmo 1, nota-se a árvore possui um critério de parada, um momento no qual a árvore para de crescer. Muitas vezes isso depende bastante de algoritmo para algoritmo, mas geralmente se fixa um tamanho mínimo como requisito para os nós serem particionados. Então se o tamanho mínimo for atingindo, aquele nó não pode ser mais ramificado (particionado), e consequentemente a árvore não cresce mais ali.

Há algoritmos que se baseiam em testes de hipóteses estatísticos como critério de parada do crescimento da árvore. Dentre esses algoritmos, pode-se citar árvore de inferência condicional (CTREE) e *model-based recursive partitioning* (MOB).

Em seguida, escolhe-se a covariáveis que particionará a árvore de decisão otimizado alguma função. Essa função dependente de qual algoritmo a árvore de decisão está se baseando. Alguns métodos utilizam medidas baseadas em entropia, outros métodos usam somas de quadrados e enquanto outros lidam com a log-verossimilhança.

1.2 Cronologia das árvores de decisão

É difícil datar quando surgiu o primeiro algoritmo baseado na lógica de uma árvore de decisão. Muitas vezes tal técnica é referida como uma estratégia de "dividir e conquistar", isto é, tem-se um problema complexo (questionamento inicial) e, com base em divisões desse problema complexo (particionamentos recursivos), cria-se um caminho para a resposta.

Em torno dos anos 300 d.C. (depois de Cristo), o filósofo Porfírio desenvolveu uma representação visual para as "categorias de Aristóteles" (categorias metafísicas do ser). Referida representação ficou conhecida mais tarde como "Árvore de Porfírio", se assemelhando muito com uma árvore de decisão, de forma em que há ramificações, particionamentos, nós pais, nós filhos e nós finais para responder determinada questão (Franklin, 1986).

Loh (2014), em seu artigo *Fifty Years of Classification and Regression Trees*, faz uma breve recapitulação histórica de alguns dos algoritmos mais relevantes no contexto de árvores de decisão. Ele divide as implementações em 4 grandes gerações.

1.2.1 1ª e 2ª Gerações

Dentre os algoritmos da 1ª geração, pode-se enunciar o algoritmo AID, ou *Automatic Interaction Detection*. Tal algoritmo trabalha apenas com uma covariável e uma variável resposta, sendo ambas quantitativas. Para definir a impureza de um nó (o quão heterogêneo o nó está), o algoritmo utiliza soma de quadrados. O algoritmo busca particionar binariamente o nó de forma a minimizar essa soma de quadrados (Morgan e Sonquist, 1963).

Em seguida, tem-se o algoritmo THAID, ou *THeta Automatic Interaction Detection*, que é simplesmente uma adaptação do AID para quando a variável resposta é qualitativa (Messenger e Mandell, 1972). Mais tarde na história, houve o algoritmo CHAID, ou *CHi-squared Automatic Interaction Detector*. Esse algoritmo não realiza particionamentos recursivos, e sim junções recursivas, isto é, funde dois ou mais nós com base em testes de hipótese estatísticos (Kass, 1980).

Partindo para a 2ª geração, existiram os algoritmos ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) e CART© (Breiman *et al.*, 1984). Esses três algoritmos serão melhores explanados ao longo do presente trabalho.

1.2.2 3ª e 4ª Gerações

Na 3ª geração, surgiu o algoritmo QUEST (*Quick, Unbiased and Efficient Statistical Tree*). Tal técnica consiste em se utilizar testes de hipóteses baseados na distribuição F de Snedecor em conjunto de estatísticas de ordens das covariáveis (Loh e Shih, 1997). Posteriormente, houve-se o CRUISE *(Classification Rule with Unbiased Interaction Selection and Estimation)*, que utiliza particionamentos não-binários com teste de hipóteses baseados na distribuição Qui-Quadrados (Kim e Loh, 2001).

Ainda na 3ª geração, houve o início do *Bayesian CART*, que consiste do $CART^{©}$ no contexto Bayesiano, utilizando de distribuições a priori e a posteriori para modelar a distribuição da variável levando em conta os particionamentos (Denison *et al.*, 1998).

Agora na 4ª geração, destaca-se o GUIDE (*Generalized, Unbiased, Interaction Detection and Estimation*), o qual consiste em uma evolução dos algoritmos QUEST e CRUISE anteriormente citados (Chaudhuri e Loh, 2002). Além disso, existem também o CTREE e MOB, dois algoritmos que serão explorados nesse trabalho.

Importante enaltecer que o termo árvore híbrida vem do fato de juntar um ou mais métodos a uma árvore de decisão. De modo a facilitar a leitura desse trabalho, o termo árvore híbrida será usado exclusivamente para o MOB, algoritmo que combina o uso de modelos paramétricos nos nós finais com os particionamentos recursivos baseados nesses próprios modelos paramétricos.

Mas além do MOB como árvore híbrida, existem outros algoritmos que implementaram essa ideia de modelos paramétricos nos nós finais da árvore de decisão, dentre esses outros algoritmos, pode-se enunciar o SUPPORT, ou *Smoothed and Unsmoothed Piecewise POlynomial Regression Trees* (Chaudhuri *et al.*, 1995), o GUIDE estendido e o LOTUS, ou *Logistic Tree with Unbiased Selection* (Chan e Loh, 2004). Cada algoritmo possui suas particularidades e semelhanças, mas o MOB apresenta uma gama maior de modelos paramétricos disponíveis para se trabalhar. Pode-se resumir toda cronologia comentada pela Figura 1.2.

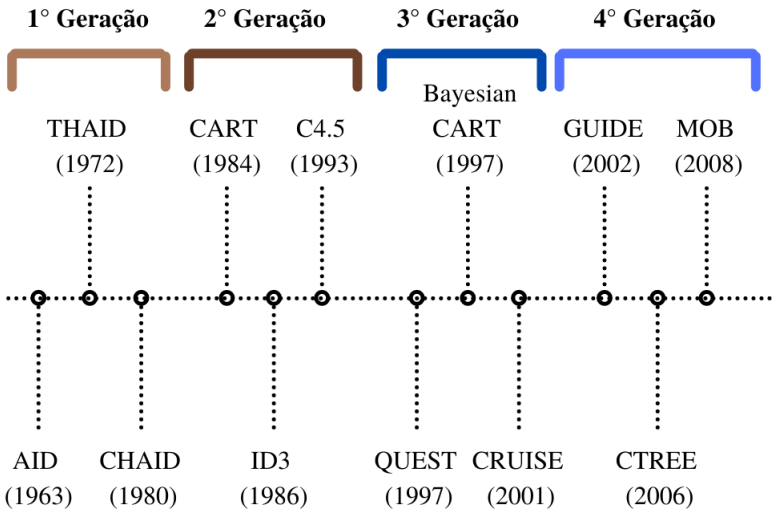

Figura 1.2: Cronologia das árvores de decisão mais relevantes.
Fonte: Figura gerada pelo próprio autor.

Capítulo 2

Revisão de métodos tradicionais

2.1 ID3 - *Iterative Dichotomiser* 3

Em meados de 1975, Ross Quinlan inicialmente apresentou o algoritmo ID3, ou chamado também de *Iterative Dichotomiser 3*. Anos depois, o conceito do algoritmo se consolida no artigo *Induction of decision trees* (Quinlan, 1986). Por sua vez, o ID3 foi baseado no CLS (*Concept Learning System*), tal algoritmo possui a ideia bem simples de "dividir e conquistar", ou seja, dado uma base de treinamento S, o CLS divide o mesmo em partições $S_1, S_2, ..., S_N$ para caracterizar a variável de interesse (Hunt *et al.*, 1966). Antes de explicar o ID3 de fato, é preciso enunciar o que é entropia e ganho de informação (medidas usadas pelo *Iterative Dichotomiser 3*).

A entropia mensura o quanto os elementos daquele subconjunto está caótico (as categorias de interesse estão todas misturadas nas partições criadas) ou não (as categorias de interesse estão bem separadas de acordo com as partições feitas). Salienta-se que há vários tipos de entropia, tais como a entropia normalizada, entropia diferencial, dentre outras. Para o uso do algoritmo ID3, será aplicada a entropia de Shannon (1948).

Definição 2.1 (Entropia para dados qualitativos)

$$H(S) = -\sum_{i=1}^{nClasses} p_i \log_2(p_i),$$

- *S é o subconjunto no qual a entropia está sendo calculada (pode ser a base de dados toda, ou apenas uma partição dela);*

- *$nClasses$ é o número de categorias que a variável de interesse possui (por exemplo,*

se a variável de interesse é binária, nClasses é 2);

- p_i *é a proporção de elementos da categoria* i*;*
- *É assumido* $0 \cdot log_2(0) = 0$*;*

Salienta-se que a entropia varia de 0 a 1, de modo que quanto maior for, mais a partição S está caótica, e quanto menor for, menos caótica está. Por exemplo, se a entropia é 0, significa que todos as observações daquela divisão são de uma mesma categoria da variável de interesse.

Além da entropia, há o ganho de informação. Essa medida mensura a redução da entropia em um subconjunto com a escolha de uma covariável.

Definição 2.2 (Ganho de informação)

$$GI(S, A) = H(S) - \sum_{c \in Categorias(A)} \frac{|S_c|}{|S|} H(S_c);$$

- $G(A, S)$ *é o ganho de informação na partição* S *com a escolha da covariável* A*;*
- $Categorias(A)$ *são as categorias de uma covariável* A *(por exemplo, se a covariável é sexo, tem-se então masculino e feminino);*
- $|S_c|$ *é a quantidade de observações que estão na partição criada pela categoria* c *da covariável* A *(por exemplo, dada a covariável sexo, quantos indivíduos são masculinos da partição* S*);*
- $|S|$ *é a quantidade de observações que estão no subconjunto* S*;*
- $H(S_c)$ *é a entropia calculada em um subconjunto determinado pela categoria* c *da covariável* A *(por exemplo, dada a covariável sexo, calcula-se a entropia somente com os indivíduos masculinos);*

Voltando para o ID3, esse inicia a iteração calculando a entropia para todo o conjunto de dados (no caso, o conjunto de treinamento S) e em seguida, calcula-se o ganho de informação de cada covariável, selecionando aquela que mais reduz a entropia. Após a seleção da covariável, é feita uma divisão com base nas suas categorias, e para cada partição feita, calcula-se novamente a entropia e o ganho de informação, selecionando mais uma vez a covariável que mais reduz a entropia, e assim iterativamente.

O processo do ID3 é feito até que haja subconjuntos "puros", isto é, até que os subconjuntos tenham apenas categorias de um tipo. Matematicamente, isso significa que a entropia de cada subconjunto deve ser 0. O Algoritmo 2 explana melhor tal processo.

Algoritmo 2: Algoritmo ID3

Entrada: Covariáveis $\boldsymbol{X}_{n\times c}$ e Variável resposta $\boldsymbol{Y}$

Saída: Árvore ID3 treinada

início

inicialização;

if *partição é pura* **then**

o rótulo é a categoria presente nessa partição;

else

calcule ganho de informação para cada covariável;

for *cada covariável* **do**

calcule o ganho de informação

end

escolha a covariável com maior ganho de informação;

particione a base em relação a essa covariável;

aplique o Algoritmo ID3 em cada partição nova;

end

fim

2.1.1 Exemplo

Para ilustrar o uso do algoritmo ID3, considera-se o seguinte conjunto de dados da Tabela 2.1. Nesse exemplo, a variável de interesse é Sucesso, isto é, se a série de TV obteve sucesso ou não, sendo assim uma variável binária. As outras colunas são covariáveis, sendo Rede televisiva A, B ou C, ou seja, a rede que fez a série. Há também a covariável Forma, a qual retrata como a série foi transmitida, isto é, se foi dublada ou legendada. Por fim, há a covariável Premiação, mostrando se série ganhou um prêmio, se foi apenas indicada ou se não foi indicada. Dado o contexto, aplica-se o algoritmo ID3.

Passo 1 - Cálculo da entropia para a base toda

Inicialmente, calcula-se a entropia para todas as Covariáveis $\boldsymbol{X}_{n\times c}$ e Variável resposta

Tabela 2.1: Informações sobre série televisivas e seu sucesso.

X_1 - Rede televisiva	X_2 - Forma	X_3 - Premiação	Y - Sucesso
A	dublado	ganhou	sim
A	legendado	indicado	nao
A	dublado	indicado	sim
B	legendado	indicado	sim
B	dublado	ganhou	nao
B	dublado	não indicado	nao
C	dublado	indicado	nao
C	legendado	ganhou	nao
B	dublado	indicado	sim
A	dublado	indicado	sim

Fonte: Tabela gerada pelo próprio autor.

$\boldsymbol{Y}$ (partição S). Desse modo,

$$\begin{aligned} H(S) &= \sum_{i=1}^{2} -p_i \log_2(p_i) \\ &= -0.5 \log_2(0.5) – 0.5 \log_2(0.5) \\ &= 0.5 + 0.5 = 1 \end{aligned}$$

Ou seja, como $H(S)$, o nível de incerteza da base de dados é máximo, uma vez que há o mesmo número de categorias sim e não da variável resposta Sucesso.

Passo 2 - Cálculo do ganho de informação de cada covariável em relação aa base toda

Em seguida, deve-se escolher uma das três covariáveis para realizar a partição da árvore. Para tal, calcula-se o ganho de informação de cada uma. Inicialmente, para a

covariável Rede televisiva,

$$
\begin{aligned}
H(X_{1,A}) &= -0.75\log_2(0.75)–0.25\log_2(0.25) \\
&= 0,311 + 0,5 = 0,811. \\
H(X_{1,B}) &= -0.5\log_2 0.5–0.5\log_2 0.5 \\
&= 0.5 + 0.5 = 1. \\
H(X_{1,C}) &= -0\log_2(0)–1\log_2(1) = 0. \\
GI(S;X_1) &= H(S) - [0.4H(X_{1,A}) + 0.4H(X_{1,B}) + 0.2H(X_{1,C})] \\
&= 1 - [0.4 \cdot 0.811 + 0.4 \cdot 1 + 0.2 \cdot 0] \\
&= 1 - 0.7244 = 0.2756
\end{aligned}
$$

Lembre-se que $(X_{1,A})$ é uma partição dos dados criada somente com aqueles que são da rede A de televisão, e consequentemente, $H(X_{1,A})$ é a entropia calculada exclusivamente com as observações de tal característica. O mesmo ocorre para $H(X_{1,B})$ e $H(X_{1,C})$.

De mesmo modo, calcula-se o ganho de informação para as covariáveis Forma e Premiação, em relação a toda a base de dados S,

$$
\begin{aligned}
GI(S;X_2) &= 1 - (0.7 \cdot 0.9852 + 0.3 \cdot 0.9199) \\
&= 1 - (0.6896 + 0.2759) \\
&= 1 - 0.9656 = 0.0344 \\
GI(S;X_3) &= 1 - (0.6 \cdot 0.9199 + 0.1 \cdot 0 + 0.3 \cdot 0.9199) \\
&= 1 - (0.5519 + 0.2759) \\
&= 1 - 0.8279 = 0.1721
\end{aligned}
$$

Por conseguinte, dentre todas as covariáveis, aquele que obteve o maior ganho de informação, em relação à partição S, foi a covariável Rede televisiva, e assim essa será a escolhida para realizar a primeira partição da árvore. Posteriormente, com base nas categorias de Rede televisiva, será novamente calculada a entropia e o ganho de informação para que se faça a próxima partição.

Passo 3 - Cálculo do ganho de informação de cada covariável em relação a categoria A de Rede televisiva

Dessa forma, o ganho de informação calculado para a covariável Forma em relação a partição criada pela categoria A de Rede Televisiva é

$$\begin{aligned} H(X_{1,A,X_{2,dublado}}) &= -1\log_2(1) - 0\log_2(0) \\ &= 0 \\ H(X_{1,A,X_{2,legendado}}) &= -0\log_2(0) - 1\log_2(1) \\ &= 0 \\ GI(X_{1,A}; X_2) &= H(X_{1,A}) - (0.75 \cdot 0 + 0.25 \cdot 0) \\ &= 0.811 - 0 = 0.811 \end{aligned}$$

Importante notar que $H(X_{1,A,X_{2,dublado}})$ é a a entropia calculada dentro da partição com categoria dublada de Forma, e este dentro da partição com categoria A de Rede televisiva.

Agora, o ganho de informação calculado para a covariável em relação a partição criada pela categoria A de Rede Televisiva é

$$\begin{aligned} H(X_{1,A,X_{3,ganhou}}) &= -1\log_2(1) - 0\log_2(0) \\ &= 0 \\ H(X_{1,A,X_{3,indicado}}) &= -0.6667\log_2(0.6667) - 0.3333\log_2(0.3333) \\ &= 0.9182 \\ H(X_{1,A,X_{3,não\ indicado}}) &= 0 \\ GI(X_{1,A}; X_3) &= H(X_{1,A}) - (1 \cdot 0 + 0.75 \cdot 0.9182 + 0 \cdot 0) \\ &= 0.811 - 0.68865 = 0.12235 \end{aligned}$$

Portanto, considerando a partição da categoria A de Rede televisiva, a covariável que mais aumenta o ganho de informação é a Forma.

Passo 4 - Cálculo do ganho de informação de cada covariável em relação a categoria B de Rede televisiva

O cálculo é feito do mesmo modo que no Passo 3 anterior, sendo assim,

$$GI(X_{1,B}; X_2) = 0.3112781$$
$$GI(X_{1,B}; X_3) = 1$$

Portanto, considerando a partição da categoria B de Rede televisiva, a covariável que mais aumenta o ganho de informação é a Premiação.

Passo 5 - Critério de parada

Como todas as partições feitas possuem uma entropia 0 (são puras), o ID3 é finalizado, criando assim a seguinte árvore na Figura 2.1. Analisando-a, nota-se que a categoria C de Rede televisiva já é o suficiente para classificar a série como um não sucesso. Já na categoria A, tem-se o nó da covariável Forma, dividida em dublada (classificando assim a resposta em sim) e legendada (classificando a resposta em não).

Finalmente na categoria B, no nó Premiação, tem-se que as séries não indicadas e até mesmo aquelas que ganharam um prêmio tendem a não fazer sucesso. Já aquelas que foram indicadas ao prêmio, fazem um sucesso.

Ademais, todo o exemplo resolvido manualmente pode ser também resolvido com no *software* `R` (R Core Team, 2021). Para isso, utiliza-se do pacote `data.tree` implementado por Glur (2020) e mais algumas funções, as quais são responsáveis por calcular a entropia e o ganho de informação ao longo dos laços.

2.1.2 Limitações

Salienta-se que o ID3, por ser um dos primeiros algoritmos de particionamentos recursivos, possui diversas limitações. Uma delas é a falta de um critério de parada, pois o algoritmo continua fazendo iterações até encontrar partições "puras". Isso pode levar a um custo computacional elevado e tempo de execução enorme em bases de dados com o número de observações e covariáveis grandes.

Além disso, a falta de um critério de parada não somente atrapalha no quesito computacional, mas também no poder preditivo, promovendo um sobreajuste. Tal problema é melhor explorado no artigo de Hssina *et al.* (2014).

O ID3 é um algoritmo de árvore de classificação, ou seja, sua variável resposta deve ser do tipo qualitativa. Já em relação às covariáveis, o mesmo ocorre, sendo necessário

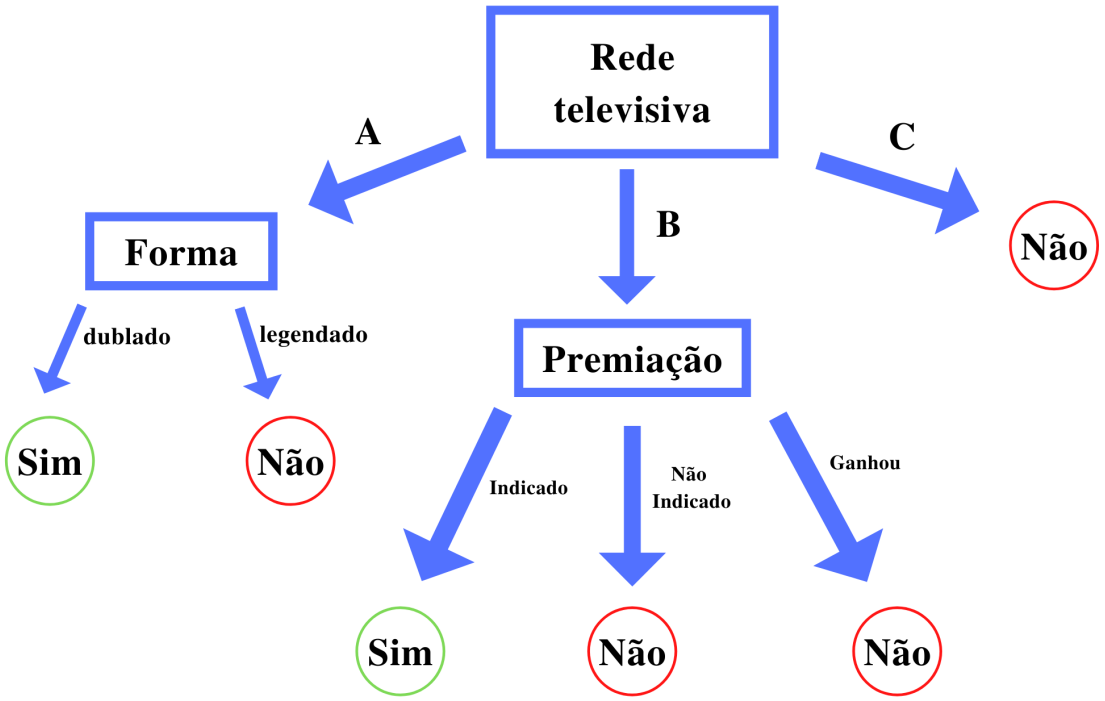

Figura 2.1: Árvore criada pelo algoritmo ID3 no exemplo de séries televisivas.
Fonte: Figura gerada pelo próprio autor.

que elas também sejam qualitativas. Caso se deseje trabalhar com variáveis contínuas, é preciso categorizá-las primeiro.

2.2 C4.5

Visando suprir as limitações do algoritmo ID3, Quinlan (1993) apresentou o algoritmo C4.5. Esse novo algoritmo consegue trabalhar com covariáveis qualitativas e quantitativas, apresenta uma nova medida relacionada ao ganho de informação, possui método de poda e também trata de eventuais valores faltantes.

A ideia do C4.5 não é muito diferente do ID3, uma vez que ambos criam a árvore de decisão de forma bem parecida, usando a entropia (2.1) e o ganho de informação (2.2). Todavia, além dessas medidas, o C4.5 usa também a razão do ganho de informação.

Definição 2.3 (Razão do ganho de informação)

$$RGI(S, A) = \frac{GI(S, A)}{FN(S, A)};$$

- *$G(A, S)$ é o ganho de informação na partição S com a escolha da covariável A (2.2);*
- *$FN(S, A)$ é o fator de normalização na partição S com a escolha da covariável A;*
- *$RGI(S, A)$ é a razão do ganho de informação na partição S com a escolha da covariável A;*

O uso de somente ganho de informação é extremamente lento para partições em que se tem diversas categorias da variável resposta. Quando isso ocorre, o ganho de informação faz criar muitas e muitas partições, para então dividir essas categorias. Ao término do algoritmo que usa o ganho de informação, cada partição final (nó filho) terá apenas uma observação de cada categoria da variável de interesse, e só assim o ganho de informação será maximizado (Quinlan, 1993).

Para evitar esse problema, divide-se o ganho de informação por um fator de normalização, fazendo com que o ganho de informação seja escalonado de acordo com o número de categorias da variável resposta que está naquela partição. Dessa forma, aquela partição que tem muitas categorias da variável resposta não será beneficiada e ficará em consoante com uma partição que tem menos categorias.

Contudo, para se calcular a razão do ganho de informação 2.3, é preciso saber o que é o fator de normalização.

Definição 2.4 (Fator de normalização)

$$FN(S, A) = -\sum_{c \in Categorias(A)} \frac{|S_c|}{|S|} \log_2 \left(\frac{|S_c|}{|S|} \right)$$

- *$FN(S, A)$ é o fator de normalização na partição S com a escolha da covariável A;*

O fator de normalização 2.4 retrata o potencial de informação que se obtém ao dividir S em partições menores de S_c. Por conseguinte, a razão do ganho de informação expressa a proporção de informação de acordo com o "tamanho/diversidade" da partição em teste.

Para exemplificar, considera-se os mesmos dados da Tabela 2.1 e calcula-se a razão do ganho de informação da covariável X_1 (Rede televisa) em relação à base de dados S.

$$
\begin{aligned}
GI(S; X_1) &= 0.2756 \ \text{(já calculado anteriormente)} \\
FN(S, X_1) &= 0.4\log_2(0.4) + 0.4\log_2(0.4) + 0.2\log_2(0.2) \\
&= 1.521 \\
RGI(S, X_1) &= \frac{GI(S; X_1)}{FN(S, X_1)} = \frac{0.2756}{1.521} = 0.181.
\end{aligned}
$$

O Algoritmo 3 ilustra como funciona o C4.5 em termos simplificados. Perceba que o processo é bem semelhante ao ID3, com a diferença que este utiliza o ganho de informação, já o C4.5 utiliza a razão de ganho de informação normalizada.

Algoritmo 3: Algoritmo C4.5

Entrada: Covariáveis $\boldsymbol{X}_{n\times c}$ e Variável resposta $\boldsymbol{Y}$

Saída: Árvore C4.5 treinada

início

inicialização;

if *partição é pura* **then**

o rótulo é a categoria presente nessa partição;

else

for *cada covariável* **do**

calcule a razão de ganho de informação normalizada

end

encontre a covariável com maior ganho de informação normalizado;

particione a base em relação a essa covariável;

aplique o Algoritmo C4.5 em cada partição nova;

end

fim

2.2.1 Covariável contínua

Dado uma covariável V contínua, seus valores são ordenados em ordem crescente. Como essa covariável V já foi observada, então há um conjunto finito desse valores, ou seja, $v_{(1)}, v_{(2)}, ..., v_{(m)}$, em que $v_{(m)}$ é a última observação da ordenação.

Sendo assim, deseja-se encontrar o melhor ponto de divisão binária em $v_{(1)}, v_{(2)}, ..., v_{(m)}$ de modo a maximizar a razão do ganho de informação, dividindo o vetor em $v_{(1)}, v_{(2)}, ..., v_{(i)}$ e em $v_{(i+1)}, v_{(i+1)}, ..., v_{(m)}$.

Dado o exposto, há $m-1$ possibilidades de divisão em relação a covariável V. Por mais que essas $m-1$ possibilidades possam parecer um grande número dependendo da base de dados, uma vez que os valores estão ordenados, o cálculo computacional fica relativamente rápido.

Logo, será feito um laço em que o ponto de divisão binário é alterado a cada iteração, sendo tal ponto definido como $\frac{v_{(i)} + v_{(i+1)}}{2}$. Ou seja, tendo o vetor de valores ordenados, para cada par de valores consecutivos, será calculado o ponto de divisão e depois a razão de ganho de informação.

O ponto que obtiver um maior ganho de informação será aquele em que a divisão binária ocorrerá, e consequentemente, o particionamento do nó da árvore em relação aquela covariável V. Salienta-se que esse procedimento de divisão binária em covariáveis contínuas também é usado no algoritmo CART© (Breiman *et al.*, 1984).

2.2.2 Poda

No ID3, a árvore é particionada até que todas as categorias da variável resposta estejam em partições de entropia 0, ou seja, até que a partição seja "pura". Já no C4.5, a árvore é "podada" no final de sua execução algorítmica, reduzindo o tamanho e complexidade da árvore, bem como o sobreajuste da mesma. Tal poda remove ramificações da árvore que fornecem pouco poder preditivo para classificar a variável de interesse.

Para isso, será estimada a taxa de erro da árvore como um todo (incluindo suas sub-árvores e folhas). Em seguida, basta substituir a sub-árvore com a folha que a faz ter a menor taxa de erro. Sendo assim, a estimação dessas taxas de erro leva a duas grandes famílias de técnicas.

A primeira família baseia-se em estimar o erro por meio de um conjunto de dados que não seja o conjunto treinamento (que fez a árvore), fazendo com que as estimativas sejam de fato não viesadas. Dentre as técnicas pertencentes a tal família, destacam-se

- *Cost-complexity pruning*: prediz o erro usando um novo subconjunto de dados diferente do conjunto treinamento. Em seguida, modela-se o erro a partir de uma soma de quadrados ponderada (Breiman *et al.*, 1984).

- *Reduzed-error pruning*: para cada sub-árvore não folha, observa-se o erro ao considerar a sub-árvore original ou sua folha recém podada. O referido processo continua até que a árvore final tenham melhor acurácia possível (Quinlan, 1987). Outros

métodos baseados nesse algoritmo foram estudados e desenvolvidos, tais como em Esposito *et al.* (1997); Oates e Jensen (1999).

Em contrapartida, a segunda família de técnicas, as quais o C4.5 usa por padrão, é uma em que a poda é feita apenas com base no conjunto treinamento, ou seja, não há um conjunto de dados separado/externo.

Dessa maneira, baseia-se em uma estimativa pessimista da taxa de erro associada a um conjunto de N casos, sendo E o número de casos que não pertence à categoria (da variável resposta) mais frequente. O C4.5 determina o limite superior da probabilidade binomial quando E eventos foram observados em N ensaios, usando um nível de confiança especificado pelo usuário.

Como consequência, o erro estimado em uma folha com N casos e E erros é N vezes a taxa de erro pessimista. Se erro estimado para sub-árvore for maior do que o erro considerando apenas a folha, a sub-árvore é podada e substituída por tal folha. Mais detalhes sobre esse processo podem ser encontrado no capítulo 4 do livro *C4.5: Programms for Machine Learning* de Quinlan (1993).

2.2.3 Exemplo

Considerou-se a base de dados sobre a ausência de estudantes de uma escola na zona rural de Nova Gales do Sul (*New South Wales*) na Austrália. Esta base de dados fora apresentado e trabalhado no livro *Modern applied statistics with S-PLUS* de Venables e Ripley (2002). As variáveis que a compõem estão na Tabela 2.2 a seguir.

Para aplicar o C4.5, considerou-se que a variável resposta seria o Status de aprendizagem *(Lrn)* e as outras variáveis seriam simplesmente covariáveis. Desse modo, o objetivo é classificar o estudante quanto a velocidade de aprendizagem dada a etnia, o sexo, a idade e o número de dias que faltou a escola.

É importante notar que a base de dados possui apenas 146 observações, então não é sensato ter um nó com poucas observações para realizar a classificação. Isto posto, escolheu-se que 10 observações como número mínimo de estudantes em cada nó. Dessa forma, o processo de poda ficará mais facilitado (uma vez que árvore não crescerá tanto) e a classificação não dependerá de um número pequeno de observações.

Observando a Figura 2.2, nota-se que a covariável *Age* foi a primeira escolhida para realizar o particionamento. Nota-se também que o C4.5 por padrão, realiza particiona-

Tabela 2.2: Descrição da base de dados sobre a ausência de estudantes.

Nome	**Sigla**	**Descrição**	**Tipo**
Etnia	*Eth*	A: estudante é do grupo de aborígine N: estudante não é do grupo de aborígine	qualitativa nominal
Sexo	*Sex*	F: estudante é do sexo masculino M: estudante é do sexo feminino	qualitativa nominal
Idade	*Age*	F0: se o estudante está na creche (“F0”), F1: se está na primeira série F2: se está na segunda série F3: se está na terceira série (“F3”)	qualitativa nominal
Status de aprendizagem	*Lrn*	AL: o estudante tem uma velocidade de aprendizado média SL: o estudante tem uma velocidade de aprendizado lenta	qualitativa nominal
Dias de ausência	*Days*	Dias que o estudante faltou na escola em um intervalo de 1 ano	quantitativa discreta

Fonte: Figura gerada pelo próprio autor.

mentos não-binários, dado que o nó foi dividido em 4 partes (número de categorias de *Age*) na primeira divisão.

Em seguida, para a categoria de segunda série (F2 de *Age*), há mais dois particionamentos, os quais ocorrem no nó 4 e no nó 6. Ademais, nota-se que há a presença de uma covariável quantitativa, algo que não poderia ocorrer com o uso do ID3 apresentado anteriormente.

Nos nós finais, que são os nós folhas 2, 3, 5, 7, 8 e 9, percebe-se que o nó não é composto de apenas uma categoria da variável resposta, e sim de um percentual das categorias. Isso ocorre porque o processo de poda é feito para deixar a árvore menos complexa e visando reduzir o sobreajuste aos dados.

Em relação a interpretação, tem-se que a série da qual o estudante está é importante para classificar o aluno quanto ao aprendizado. Já para a segunda série (F2), é preciso o uso das covariáveis sexo e dias para maiores informações.

Além disso, o menor dos nós folhas foi de $N = 10$ (no nó 8), isso ocorre devido a imposição feita no começo do exemplo. Mas também pode-se verificar o que ocorreria sem essa imposição.

Com o tamanho mínimo do nó sendo 2, criou-se a árvore retratada na Figura 2.3. Em contraste com a árvore da Figura 2.2 (que tem tamanho mínimo do nó sendo 10), percebe-se que esta possui uma ramificação maior, até chegar em nós filhos com pouquíssimas observações. Posto isso, levanta-se a questão para o leitor: será que classificar uma nova

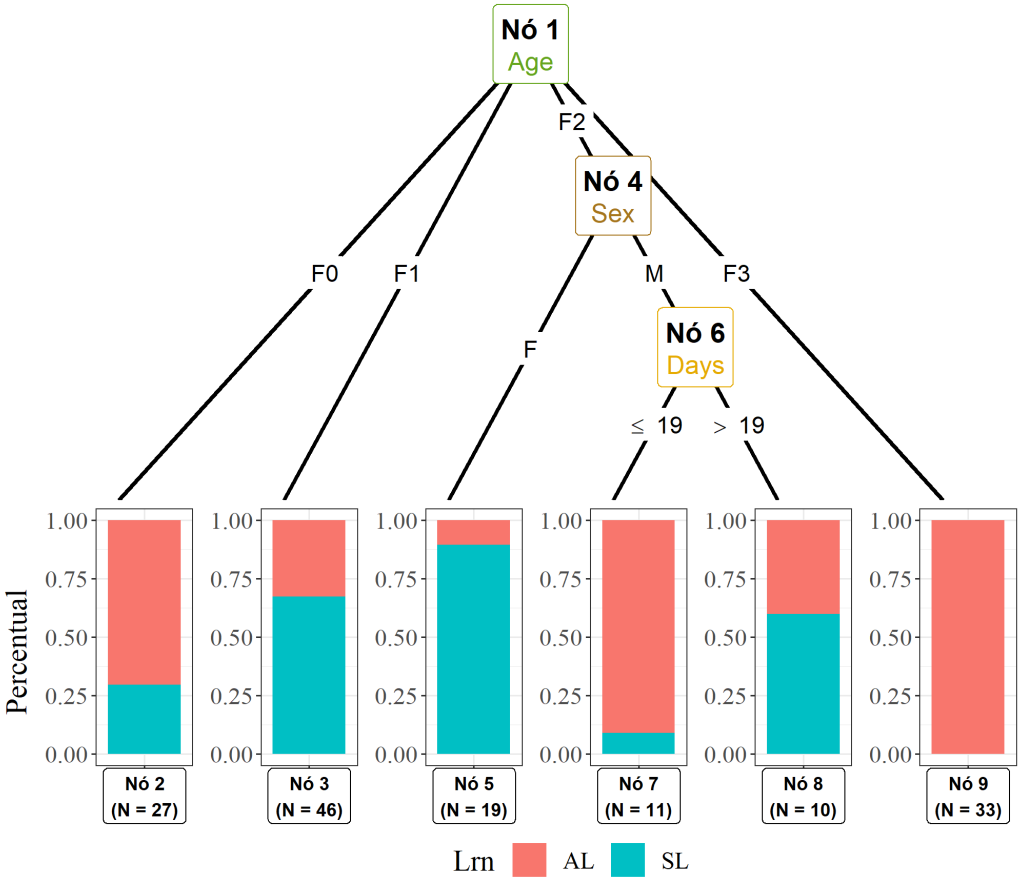

Figura 2.2: Árvore criada pelo algoritmo C4.5 para o problema de status de aprendizagem.
Fonte: Figura gerada pelo próprio autor.

observação com base em um nó filho de tamanho $N = 2$ é viável?

Destaca-se que o C4.5 está implementado no *software* `R` (R Core Team, 2021) por meio da função `J48()` do pacote *RWeka* (Hornik *et al.*, 2009). Mais informações sobre os hiperparâmetros da árvore C4.5 (como número mínimo de observações, tipo de poda, se binário ou não, etc) podem ser encontrados ao digitar `WOW(J48)` no terminal do `R`. É interessante apreciar que J48 foi o nome dado a implementação do código aberto em Java do algoritmo C4.5 para o aplicativo de mineração de dados Weka (Witten *et al.*, 2005).

2.3 CART

O CART© *(Classification and Regression Trees)* é um método de árvore de decisão extremamente versátil e poderoso, sendo tal método proposto por Breiman *et al.* (1984). Assim como o C4.5, o CART© consegue trabalhar com covariáveis contínuas. Contudo, diferente do C4.5, o CART© pode ter a variável resposta de natureza qualitativa (árvore

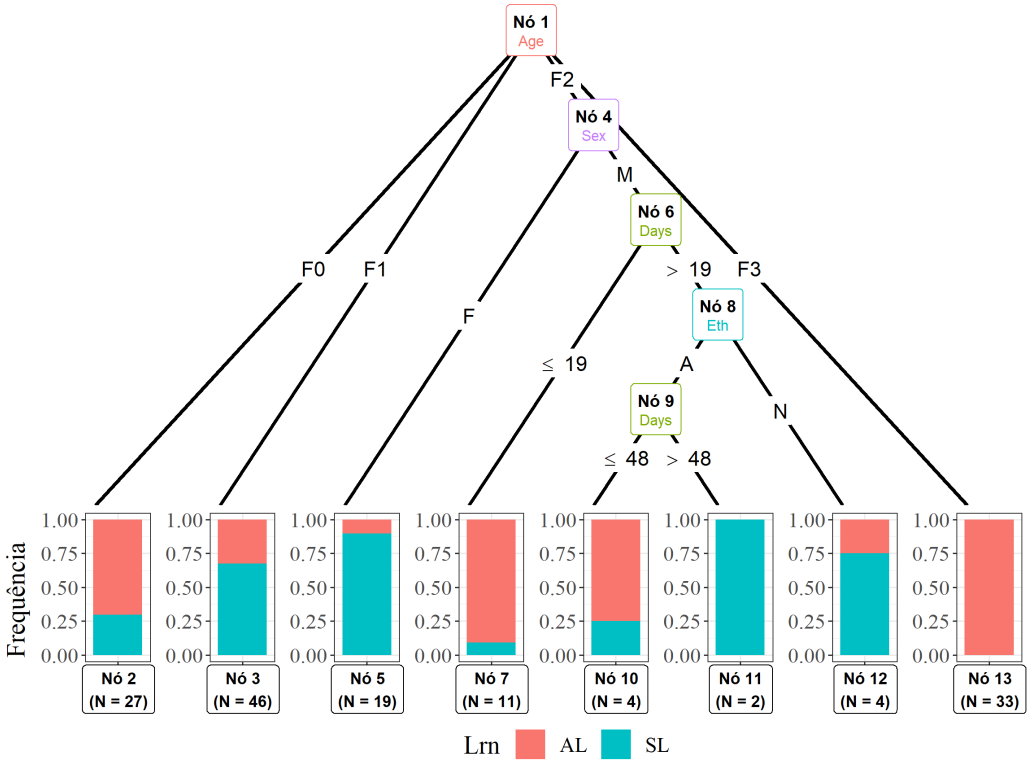

Figura 2.3: Árvore criada pelo algoritmo C4.5 para o problema de status de aprendizagem (tamanho mínimo do nó de 2).
Fonte: Figura gerada pelo próprio autor.

de classificação) ou quantitativa (árvore de regressão).

2.3.1 Árvore de classificação

Assim como o ID3 e o C4.5, o conceito tradicional de uma árvore de decisão se mantém no CART©. Inicialmente a melhor covariável será escolhida para realizar o particionamento. O ponto ótimo do particionamento será feito de modo a otimizar alguma medida de "pureza" do nó. Posteriormente, o processo será feito recursivamente, até que haja uma etapa de poda.

Considerando um caso de classificação (variável resposta qualitativa), o diferencial do CART© para os outros métodos vistos até agora é que este utiliza o índice de Gini (Gini, 1921).

Definição 2.5 (Índice de Gini)

$$IG(S) = \sum_{i=1}^{nClasses} p_i(1 - p_i),$$

- *S é o subconjunto no qual o índice de Gini está sendo calculado (pode ser a base de dados todo, ou apenas uma partição dele);*

- *$nClasses$ é o número de categorias que a variável de interesse possui (por exemplo, se a variável de interesse é binária, $nClasses$ é 2);*

- *p_i é a proporção do número de elementos da categoria i da variável resposta pelo número de elementos no subconjunto S;*

O índice de Gini (Definição 2.5) é bem semelhante à entropia (Definição 2.1), de modo que quanto maiores as medidas, mais a partição S está caótica. Salienta-se que, na prática, não há grandes diferenças entre as duas medidas. Isto é, a árvore de decisão final não será muito diferente porque a entropia foi utilizada ou porque o índice de Gini foi utilizado.

Raileanu e Stoffel (2004) compararam exaustivamente o índice de Gini e a entropia, tanto no aspecto teórico, analisando a convergência, situações possíveis e casos particulares, como também em situações práticas, por meio de exemplos computacionais de pequeno e médio porte. Os autores do artigo concluíram que em apenas 2% dos casos as medidas resultaram em árvores de decisão levemente diferentes e que a entropia é mais lenta de se calcular, devido a presença do $\log_2$.

Essas duas métricas surgiram em épocas próximas, mas em disciplinas diferentes. A entropia foi desenvolvida por Shannon (1948), e o mesmo fora matemático, engenheiro elétrico e criptógrafo nos Estados Unidos da América. Já o índice de Gini foi criado por Gini (1921), que fora um estatístico, demográfico e sociólogo na Itália.

2.3.2 Árvore de regressão

O método CART© pode também ser definido como uma árvore de regressão, isto é, a árvore que consegue lidar com a variável resposta sendo quantitativa. Para entender mais sobre esse caso, é preciso definir 3 elementos essenciais desta árvore de decisão (Breiman *et al.*, 1984).

1. Definir um modo de particionar binariamente o nó;

2. Definir uma regra para saber quando o nó é terminal;

3. Definir um valor y para os nós filhos da árvore de regressão.

Modo de particionar binariamente o nó

Nos métodos estudados anteriormente (ID3, C4.5, CART© como classificador), as medidas usadas para o particionamento eram a entropia, a razão do ganho de informação e o índice de Gini, respectivamente. Contudo, essas medidas foram idealizadas para situações com variáveis resposta qualitativas, e não quantitativas. Para contornar esse problema, utiliza-se o conceito da soma de quadrados, mas agora na árvore de decisão.

Definição 2.6 (Soma de quadrados para árvore de regressão em CART©)

$$SQ(s,t) = SQ_{Total} - (SQ_{Esquerda} + SQ_{Direita})$$

- *t é um nó da árvore de regressão;*
- *s é um ponto de divisão do nó (s pode ser um conjunto de categorias de dada covariável qualitativa, ou pode ser um valor de uma covariável quantitativa. Por exemplo, pode ser homem e mulher, se a covariável for sexo, ou pode ser 67kg se a covariável for massa);*
- *$SQ(s,t)$ é a diferença de soma de quadrados entre o nó pai e seus nós filhos;*
- *SQ_{Total} é a soma de quadrados calculada em todo o nó t;*
- *$SQ_{Esquerda}$ é a soma de quadrados calculada para os y_i "à esquerda de s", que estão em um lado da partição, $y_i < s$;*
- *$SQ_{Direita}$ é a soma de quadrados calculada para os y_i "à direita de s" s, isto é, que estão em um outro lado da partição;*

Lembrando que a soma de quadrados é a usual, ou seja,

$$SQ_{grupo} = \sum_{i=1}^{|grupo|} (y_i - \bar{y})^2,$$

em que SQ_{grupo} é a soma de quadrados calculada em "grupo", y_i é o valor do i-ésimo indivíduo na variável resposta e $\bar{y}$ é a média de todos os y_i. Dado o exposto, para decidir o melhor ponto de divisão binária do nó basta escolher o s que minimize $SQ(s,t)$.

Regra para saber quando o nó é terminal

O processo é feito de maneira recursiva até particionar completamente a base de dados. Ou o usuário pode definir um número mínimo de observações em cada nó filho.

Para evitar um sobreajuste, é realizado uma poda. No CART©, referida poda é semelhante tanto para o caso de uma árvore de classificação quanto para uma árvore de regressão, sendo assim será melhor explicada ela ao fim desta subseção.

Valor y para os nós filhos

Na árvore de classificação, após obter o nó filho, a classe predita é alguma categoria da variável resposta. Isto é, ou é a categoria majoritária de y que está nó filho, ou, se houver apenas uma categoria de y, é ela própria.

Entretanto na árvore de regressão, o nó filho é composto por observações que possuem diversos valores de y. Por mais que esses valores sejam próximos (devido ao processo de particionamento da árvore), ainda sim é preciso escolher um valor como saída do nó. Para tal, usa-se uma estatística dos y que estão naquele nó filho. Tal estatística pode ser a média (geralmente é a medida usada no CART©), a moda, a mediana, etc.

Algoritmo

De forma a entender como funciona o CART©, pode-se observar o Algoritmo 4. Perceba que é preciso um cuidado especial em relação a variável resposta $\boldsymbol{Y}$ para quando ela for quantitativa ou qualitativa. Em relação aos passos posteriores, eles são os mesmos,

mudando apenas o cálculo de índice de Gini ou do $SQ(t, s)$.

Algoritmo 4: Algoritmo CART©

Entrada: Covariáveis $\boldsymbol{X}_{n\times c}$ e Variável resposta $\boldsymbol{Y}$

Saída: Árvore CART© treinada

início

inicialização;

if $\boldsymbol{Y}$ *é qualitativa* **then**

if *partição é pura* **then**

o rótulo é a categoria presente nessa partição;

else

for *cada covariável* **do**

calcule o índice de gini

end

encontre a covariável com menor índice de Gini;

particione a base em relação a essa covariável;

aplique o Algoritmo CART© em cada partição nova;

end

else

if *tamanho mínimo do nó for atingido* **then**

o particionamento é finalizado;

else

for *cada covariável* **do**

calcule SQ(s,t)

end

encontre a covariável com menor SQ(s,t);

particione a base em relação a essa covariável;

aplique o Algoritmo CART© em cada partição nova;

end

end

fim

2.3.3 Validação cruzada para a poda

Depois da árvore estar feita com seus particionamentos recursivos, o CART© realiza a poda a fim de que a mesma não fique sobrejustada aos dados de treinamento. Dessa forma, utilizando a mesma notação do livro *Classification and Regression Trees* (Breiman

et al., 1984), define-se o que é custo e custo-complexidade de uma árvore de decisão.

Definição 2.7 (Nós e custo de uma árvore)

Sejam $T_1, T_2, ..., T_k$ sub-árvores de uma árvore T_{max}, define-se

Definição 2.8 (Custo-complexidade de uma árvore)

Para qualquer sub-árvore $T \leq T_{max}$, define-se sua complexidade como $|T|$. Seja $\alpha \leq 0$ número real chamado de parâmetro de complexidade. Desse modo, o custo-complexidade de uma árvore é medido como

$$R_\alpha(T) = R(T) + \alpha|T|.$$

- *$|T|$ é o número de nós folhas de uma árvore T;*
- *Custo de T $= R(T) = \sum_{i=k}^{k} \frac{|T_i|}{|T|} R(T_i)$;*

Por conseguinte, o custo de complexidade de uma árvore T é uma combinação linear do custo e de sua complexidade. Pensando em um método de poda, deseja-se encontrar $T_\alpha \leq T_{max}$ que minimize $R_\alpha(T)$, isso para cada valor de α. Isto é,

$$R_\alpha(T_\alpha) = \underset{T \leq T_{max}}{\operatorname{argmin}} R_\alpha(T). \tag{2.9}$$

Intuitivamente, nota-se que, se α for pequeno, a penalidade por ter um grande número de nós filhos será pequena e T_α (sub-árvore podada por α) será grande. Por outro lado, para α suficientemente grande, a sub-árvore T_α consistirá apenas no nó raiz e a árvore T_{max} terá sido completamente podada. Nesse contexto, e usando as definições acima, é interessante salientar 3 propriedades resultantes.

Propriedades 2.10 (Propriedades de uma árvore no CART)

1. *Se T_1 e T_2 são sub-árvores de T com $R_\alpha(T_1) = R_\alpha(T_2)$, então T1 é uma subárvore de T_2 ou T_2 é uma subárvore de T_1; e portanto, $|T_1| < |T_2|$ ou $|T_2| < |T_1|$;*
2. *Se $\alpha \leq \beta$, então $T_\alpha = T_\beta$ ou T_α é uma sub-árvore estrita de T;*

- *Dado algum conjunto de números* $\alpha_1, \alpha_2, ..., \alpha_m$*; tanto* $T_{\alpha_1}, ..., T_{\alpha_m}$ *quanto* $R(T_{\alpha_1}), ..., R(T_{\alpha_m})$ *podem ser calculados de forma eficiente.*

Tais propriedades (2.10) estão provadas no livro *Classification and Regression Trees* (Breiman *et al.*, 1984) e também são discutidas na vinheta de Therneau e Atkinson (1997). Com essas propriedades, é possível dizer que todos os valores possíveis de α (parâmetro de complexidade que minimiza o custo-complexidade) podem ser agrupados em m intervalos, os quais $m \leq |T|$,

$$I_1 = (0, \alpha_1], \quad I_2 = (\alpha_1, \alpha_2], \quad ..., \quad I_m = (\alpha_{m-1}, \infty);$$

onde todos os $\alpha \in I_i$ advêm de uma mesma sub-árvore podada. Desse modo, emprega-se validação cruzada para escolher o melhor valor de α que minimiza $R_\alpha(T)$.

O passo a passo da validação cruzada pode ser escrita como,

1. Ajuste a árvore em todo o conjunto (treinamento) e calcule $I_1, ..., I_m$. Seja

$$\beta_1 = 0, \quad \beta_2 = \sqrt{\alpha_1 \alpha_2}, \quad ..., \quad \beta_{m-1} = \sqrt{\alpha_{m-2} \alpha_{m-1}}, \quad \beta_m = \infty,$$

   sendo que cada β_i é um valor típico de I_i;

2. Divida os dados em s grupos $G_1, ...G_s$, cada um de tamanho s/n. E para cada grupo G_i, faça

   2.1. Ajuste a árvore para todos os G grupos, exceto para o G_i. Em seguida, determine $T_{\beta_1}, ..., T_{\beta_m}$;

   2.2 Obtenha a classe predita para as observações em G_i, para cara $T_{\beta j}$ em $1 \leq j \leq m$. Em seguida, calcule o custo-complexidade.

3. Sobre os G_i, some para obter uma estimativa de custo-complexidade para cada β_j. Para o β (parâmetro de complexidade, visto na Definição 2.8) com menor custo-complexidade estimado, modele a árvore T_β, sendo esta a escolhida como a árvore podada.

2.3.4 Exemplo

Considerou-se a base de dados sobre locais de terremotos ao torno de Fiji (país insular da Oceania, composto por 332 ilhas no Oceano Pacífico) em 1964. Essa base fora obtida

do projeto *Harvard PRIM-H*. Para acessá-la, basta utilizar o pacote `datasets` do próprio R (R Core Team, 2021). Os dados estão sob o nome de *quakes*.

Tabela 2.3: Descrição da base sobre terremotos em Fiji.

Nome	**Sigla**	**Descrição**	**Tipo**
Latitude	*lat*	Latitude do terremoto	quantitativa contínua
Longitude	*long*	Longitude do terremoto	quantitativa contínua
Profundidade	*depth*	Profundidade do terremoto em km	quantitativa contínua
Magnitude	*mag*	Magnitude do terremoto em escala *Richter*	quantitativa contínua
Estações	*stations*	Número de estações que notificaram o terremoto	quantitativa discreta

Fonte: Tabela gerada pelo próprio autor.

Para aplicar o CART©, considerou-se que a variável resposta seria a Magnitude *(mag)* e as outras variáveis seriam simplesmente covariáveis. Desse modo, o objetivo é estudar a magnitude do terremoto dado a latitude, longitude, profundidade e número de estações. Além do mais, nota-se que a variável de interesse é quantitativa, por conseguinte, a árvore de decisão é uma árvore de regressão.

Destaca-se que a base de dados possui 1000 observações. Então escolheu-se que o número mínimo de observações que deve existir em um nó para que o particionamento seja tentado foi de 150. Já o número mínimo de observações que o nó folha deve ter foi de 50. Em relação ao parâmetro de complexidade para o processo de poda, escolheu-se um valor de 0.01, o qual já é o usual do algoritmo.

Estudando a Figura 2.4, evidencia-se que a covariável número de estações é a mais presente para os particionamentos realizados. Em seguida, no nó 2, percebe-se a presença da covariável profundidade em ambos os lados do nó. Por fim, tem-se no nó 4, a covariável longitude. Pode-se assim dizer que o aspecto mais fundamental para determinar a magnitude de um terremoto em Fiji foi o número de estações, seguido da profundidade, e em último, a longitude.

Em relação à interpretação, pode-se dizer que quanto mais estações estão notificando o terremoto, maior sua escala *Ritcher*, cerca de 5.5. Fato esse verificado pela partição nos nós 1 e 11. Por outro lado, para terremotos notificados por menos de 24 estações (nó 2), profundidade maior ou igual a 68.5 (nó 3) e longitude maior de 179 (nó 4), há presença de terremotos com escala *Ritcher* menor, em torno de 4.3.

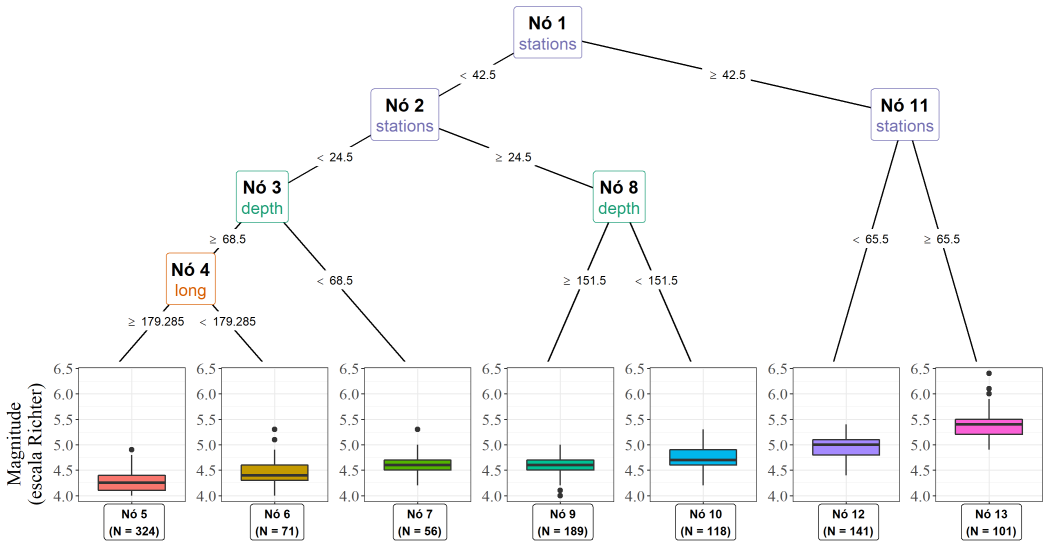

Figura 2.4: Árvore criada pelo algoritmo CART© para o problema de terremotos em Fiji (parâmetro de complexidade de 0.01).
Fonte: Figura gerada pelo próprio autor.

Considerando agora um parâmetro de complexidade para poda no valor de 0.02 (o dobro do anterior), obtém-se a árvore da Figura 2.5. Nota-se que a árvore é menor que a anterior, ou seja, a poda é mais rigorosa. E em relação as covariáveis, foram usadas apenas o número de estações e a profundidade.

Salienta-se que o CART© está implementado no *software* `R` (R Core Team, 2021) por meio da função `rpart()` do pacote *rpart* (Therneau e Atkinson, 2019). Mais informações sobre os hiperparâmetros da árvore CART© (como número mínimo de observações, parâmetro de complexidade da poda, se binário ou não, etc) podem ser encontrados ao digitar `?rpart.control` no terminal do `R`. Aprecia-se que `rpart` foi o nome dado a implementação do código aberto em `R` do algoritmo CART©.

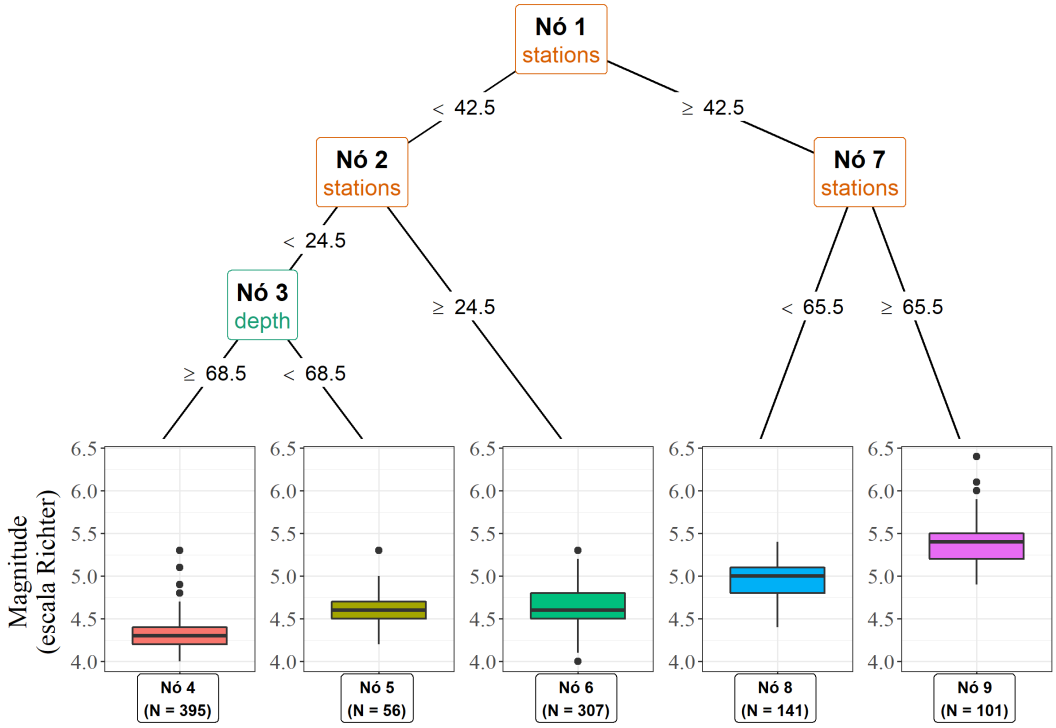

Figura 2.5: Árvore criada pelo algoritmo CART© para o problema de terremotos em Fiji (parâmetro de complexidade de 0.02).

Fonte: Figura gerada pelo próprio autor.

Capítulo 3

Revisão de métodos baseados em testes de hipótese

3.1 CTree

Os algoritmos vistos até agora (ID3, C4.5 e o CART©) realizam buscas pelos possíveis particionamentos de modo a otimizar alguma função. Por exemplo, no ID3 e no C4.5, busca-se maximizar o ganho de informação ou sua razão. No CART©, procura-se otimizar o índice Gini (classificação) ou a soma de quadrados (regressão). Em vista disso, alguns autores perceberam que havia certos problemas de se focar apenas nessa maximização, pois pode ocorrer um sobreajuste na base de treino e uma seleção viesada para covariáveis com grande número de particionamentos possíveis.

De forma a contornar esses problemas, Hothorn *et al.* (2006) propuseram o uso de árvores de inferência condicional (CTree). Essa metodologia não foca apenas em maximizar uma medida, mas se utiliza também do conceito de significância estatística para selecionar as covariáveis do particionamento, melhorando assim a eficácia preditiva da árvore de decisão.

Ademais, o CTree possui diversas vantagens, tais como a possibilidade de trabalhar com a variável resposta sendo quantitativa contínua ou discreta, qualitativa nominal ou ordinal, dados censurados (sobrevivência) e multivariados.

3.1.1 Particionamento binário com pesos

Define-se algumas notações.

Definição 3.1 (Distribuição condicional da variável resposta)

$$D(\boldsymbol{Y}|\boldsymbol{X}) = D(\boldsymbol{Y}|X_1, ..., X_m) = D(\boldsymbol{Y}|f(X_1, ..., X_m)).$$

- $\boldsymbol{Y}$ *é o vetor da variável resposta;*
- $\boldsymbol{X} = (X_1, ..., X_m)$ *é o vetor m-dimensional de covariáveis;*
- $D(\boldsymbol{Y}|\boldsymbol{X})$ *é a distribuição condicional da resposta* $\boldsymbol{Y}$ *dadas as covariáveis* $\boldsymbol{X}$*.*

Além disso, sabe-se que o espaço das covariáveis será divido em $B_1, ..., B_r$ partições devido a natureza de árvores de decisão (particionamentos recursivos), desse modo, pode-se definir o espaço das covariáveis da seguinte forma:

Definição 3.2 (Espaço de covariáveis)

$$\mathcal{X} = \cup_{k=1}^{r} B_k.$$

- B_k *é partição do espaço de covariáveis.*

Ademais, a árvore de decisão será modelada com base em uma amostra de aprendizado L_n. Tal amostra possui n observações independentes e identicamente distribuídas, e podem também possuir algumas covariáveis X_{ij} ausentes.

Definição 3.3 (Amostra de aprendizado)

$$\mathcal{L}_n = (\boldsymbol{Y}_i, X_{1i}, ..., X_{mi}); i = 1, ..., n.$$

Dado o exposto, um algoritmo de particionamento recursivo binário irá aprender de $\mathcal{L}_n$ atribuindo pesos $\boldsymbol{w} = (w_1, ..., w_n)$ para as observações dos dados de treinamento. Isto é, cada nó da árvore possuirá um vetor de pesos relativos às observações, sendo que o peso w_i será positivo, se observação está presente naquele nó, ou será nulo, se a observação não está naquele nó. Por conseguinte, pode-se definir 3 passos para o algoritmo de particionamento binário com pesos:

1. Teste a hipótese nula de independência entre qualquer uma das m covariáveis e a variável resposta (o teste de hipótese é global, pois relaciona-se com todas as variáveis);

   1.1. Pare a recursão se a hipótese não puder ser rejeitada;

   1.2. Caso contrário, seleciona-se a covariável com maior associação com a variável resposta $\boldsymbol{Y}$;

2 Divide-se o espaço das covariáveis da iteração atual ($\mathcal{X}_{j^*}$) na partição A^* e na partição $A^* - \mathcal{X}_{j^*}$;

   2.1 Haverá dois subgrupos criados, os quais serão o subgrupo da "esquerda" e o da "direita". Determina-se os vetores de pesos para cara um, isto é, $\boldsymbol{w}_{esquerda}$ e $\boldsymbol{w}_{direita}$;

   2.2 Dentro de cada vetor de pesos para as i observações, tem-se $w_{esquerda,i} = w_i \mathbb{I}(X_{j*i} \in A^*)$ e $w_{direita,i} = w_i \mathbb{I}(X_{j*i} \notin A^*)$, com $\mathbb{I}(.)$ denotando uma função indicadora;

3. Repete-se os passos 1. e 2. alterando-se os vetores de pesos $\boldsymbol{w}_{esquerda}$ e $\boldsymbol{w}_{direita}$.

Seguindo esses passos, Hothorn *et al.* (2006) mostram que a seleção de variável (passo 1) e o processo de particionamento (passo 2) não sofrem de um viés tendendo para covariáveis com grande número de divisões. Outrossim, como no passo 1 é feito um teste de hipótese estatístico, pode-se fixar um nível de significância α global como critério de parada do algoritmo.

3.1.2 Seleção de covariável

O passo 1 do algoritmo explicado na subseção 3.1.1 consiste em selecionar a melhor covariável para o particionamento ou parar a recursão, que é a não rejeição da hipótese de independência.

Definição 3.4 (Hipóteses de independência global e parcial)

$$\textit{Hipótese de independência global: } H_0 = \cap_{j=1}^{m} H_0^j$$
$$\textit{Hipótese de independência parcial: } H_0^j = D(\boldsymbol{Y}|X_j) = D(\boldsymbol{Y})$$

A recursão do algoritmo CTree será interrompida quando não for possível rejeitar H_0 (Definição 3.4) a um nível α especificado. Para realizar tal teste, define-se uma estatística que mede a associação entre $\boldsymbol{Y}$ e $X_j, j = 1, ..., m$.

Definição 3.5 (Estatística de associação entre $\mathbf{Y}$ e covariáveis)

$$\boldsymbol{T}_j(\mathcal{L}_n, \boldsymbol{w}) = vec\left(\sum_{i=1}^{n} w_i g_j(X_{ij}) h(\boldsymbol{Y}_i, (\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n))^T\right) \in \mathbb{R}^{p_j q \times 1}.$$

- *$vec(.)$ é um operador que converte uma matriz $p_j \times q$ em um vetor de coluna $p_j q \times 1$. Este operador faz essa conversão empilhando os vetores coluna da matriz original;*
- *$g_j : \mathcal{X}_j \to \mathbb{R}^{p_j}$ é transformação não aleatória nas covariáveis X_j;*
- *$h : \mathcal{Y} \times \mathcal{Y}^n \to \mathbb{R}^q$ é a função de influência. Tal função depende da natureza da variável resposta;*

A distribuição de $\boldsymbol{T}_j(\mathcal{L}_n, \boldsymbol{w})$ sob a hipótese nula H_0^j (Definição 3.4) depende da distribuição conjunta de $\boldsymbol{Y}$ e X_j, mas esta é desconhecida em situações práticas. Para contornar isso, utiliza-se testes de permutação, uma vez que, sob a hipótese nula, pode-se descartar a dependência fixando as covariáveis e condicionando a todas as possíveis permutações das respostas.

Assim, Strasser e Weber (1999) demonstraram a esperança condicional $\mu_j \in \mathbb{R}^{p_j q \times 1}$ e a covariância $\Sigma_j \in \mathbb{R}^{p_j q \times p_j q}$ de $\boldsymbol{T}_j(\mathcal{L}_n, \boldsymbol{w})$ sob a hipótese nula H_0 (Definição 3.4) dado todas as permutações das respostas. Possuindo a esperança condicional e a covariância, pode-se obter estatísticas de teste e valores-p das hipóteses global e parciais (Definição 3.4).

Usando os valores-p, por exemplo, mensura-se a associação entre a variável resposta e as covariáveis, escolhendo aquela com maior associação para fazer o particionamento. No entanto, os valores-p (e estatísticas teste) de diferentes covariáveis não podem ser comparados entre si, pois as covariáveis podem estar em escalas diferentes ou serem de naturezas distintas. Para contornar essa adversidade, existem alguns métodos, dentre eles destaca-se

- Reamostragem de valores-p mínimos (Chaubey, 2012);

- Valores-p ajustados pela correção de *Bonferroni*, usando o fator $1-(1-P_j)^m$;

Portanto, rejeita-se H_0 (3.4) quando o mínimo dos valores-p reajustados for menor que um nível de significância α predeterminado. Se H_0 não for mais rejeitada, interrompe-se a recursão do algoritmo.

3.1.3 Critério de divisão do nó

Depois de realizar o passo 1, selecionando uma covariável X_j*, faz-se o passo 2 do algoritmo explicado na subseção 3.1.1. Passo este que consiste em particionar o nó no ponto ótimo de modo a criar dois nós filhos e assim continuar a recursão da árvore de inferência condicional.

Para tal, considera-se a estatística 3.5, mas agora com um caso particular levando em conta apenas uma partição a partir do nó pai. Consequentemente, a transformação $g_j(.)$ será a indicadora $\mathbb{I}(X_{j*i} \in A)$.

Definição 3.6 (Caso particular de 3.5 considerando um subconjunto A)

$$\boldsymbol{T}_{j*}^{A}(\mathcal{L}_n, \boldsymbol{w}) = vec\left(\sum_{i=1}^{n} w_i \mathbb{I}(X_{j*i} \in A) h(\boldsymbol{Y}_i, (\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n))^T\right) \in \mathbb{R}^q.$$

Dessa maneira, seja a partição A^* um subgrupo e a partição $A^* - \mathcal{X}_{j^*}$ um outro subgrupo obtidos no passo anterior. Deseja-se obter A^* que maximize a diferença entre esses grupos. Usando a esperança condicional μ_{j*} e a covariância Σ_{j*} de $\boldsymbol{T}_{j*}^{A}(\mathcal{L}_n, \boldsymbol{w})$ (Strasser e Weber, 1999), obtém-se que

$$A^* = \arg\max_x \; c(\boldsymbol{t}_{j*}^{A}, \mu_{j*}^{a}, \Sigma_{j*}^{A}),$$

é o particionamento que maximiza a diferença entre todos os possíveis subgrupos originados do nó pai. Ademais, tem-se que $c(\boldsymbol{t}_{j*}^{A}, \mu_{j*}^{a}, \Sigma_{j*}^{A})$ é a estatística do teste calculada.

Em linhas gerais, o CTREE pode ser retratado no Algoritmo 5. Note que o crescimento da árvore é pautado pelos valores-p, tanto na escolha da covariável particionadora quando

no critério de parada do algoritmo.

Algoritmo 5: Algoritmo CTREE

Entrada: Covariáveis $\boldsymbol{X}_{n\times c}$, Variável resposta $\boldsymbol{Y}$ e α

Saída: Árvore CTREE treinada

início

inicialização;

teste a hipótese nula de independência das m covariáveis;

if *todos os valores-p das m covariáveis forem maior que* α **then**

pare o crescimento da árvore;

else

selecione a variável com menor valor-p;

encontre o particionamento A^* que maximize a estatística;

aplique o Algoritmo CTREE em cada partição nova;

end

fim

3.1.4 Exemplo

Considerou-se a base de dados sobre produção acadêmica de doutorandos em bioquímica, com as informações obtidas dos próprios departamentos de bioquímica e registros universitários, além de uma procura feita nos arquivos do Conselho Nacional de Pesquisa dos EUA (ou chamada também de NRC, *National Research Council*). A população do estudo são aqueles que receberam o título de doutor no período de 1956 à 1958 e 1961 à 1963. Ademais, a amostra obtida tem 915 observações (Long, 1990).

Para aplicar o CTREE, considerou-se que a variável resposta seria o número de artigos publicados pelo doutorando e as outras variáveis seriam covariáveis. Desse modo, o objetivo é entender se o orientador influência no número de artigos do orientando, se aspectos como sexo, se é casado, além de outros que possam influenciar no número de publicações por parte do aluno. Além do mais, nota-se que a variável de interesse é quantitativa, por conseguinte a árvore de decisão é uma árvore de regressão.

Como a base possui 915 observações, escolheu-se que o número mínimo de observações em um nó para que o particionamento seja cogitado foi de 21. Já o número mínimo de observações que o nó folha deve ter foi de 7. Inicialmente, considera-se $\alpha = 5\%$ para que um nó seja particionado.

Tabela 3.1: Descrição da base sobre número de artigos.

Nome	Sigla	Descrição	Tipo
Número de Artigos publicados pelo doutorando	*articles*	Número de artigos publicados durante os últimos 3 anos de doutorado	quantitativa discreta
Sexo	*gender*	Sexo do estudante de doutorado	qualitativa nominal
Casado	*married*	Se o doutorando é casado ou não	qualitativa nominal
Filhos	*kids*	Número de crianças com menos de 6 anos que o doutorando possui	quantitativa discreta
Prestígio	*prestige*	Prestígio do programa de doutorado (varia de 0 a 5, quanto maior, melhor)	quantitativa contínua
Número de Artigos publicados pelo orientador	*mentor*	Número de artigos publicados durante com autoria do orientador durante os últimos 3 anos de doutorado do aluno	quantitativa discreta

Fonte: Tabela gerada pelo próprio autor.

Estudando a Figura 3.1, evidencia-se que a variável *mentor* (número de artigo publicados pelo orientador) é a covariável que mais aparece nos nós internos da árvore. Além do mais, nota-se que a covariável casado também aparece. Contudo, em todos os 5 nós finais, parece que não há uma diferença muito grande no número de artigos publicados pelos alunos, uma vez que os quantis foram bem similares, apenas o nó 9 se destacando.

Considerando agora um $\alpha = 15\%$, isto é, sendo mais flexível com a escolha da covariável para o particionamento binário, obtém-se o resultado da Figura 3.2. Logo de início nota-se que a árvore cresceu mais em comparação com a Figura 3.1 anterior. Percebe-se também a presença das covariáveis sexo e casado, covariáveis essas que não estavam presentes na árvore anterior.

Portanto, o nível de significância é de extrema importância para o desenvolvimento de uma árvore de decisão no algoritmo CTREE. Além de que como não há procedimento de poda, a árvore em si cresce naturalmente com os testes de hipótese explicados anteriormente, reforçando ainda mais a influência de α.

Salienta-se que o CTREE está implementado no *software* `R` (R Core Team, 2021) por meio da função `ctree()` do pacote `partykit` (Hothorn *et al.*, 2006). Mais informações

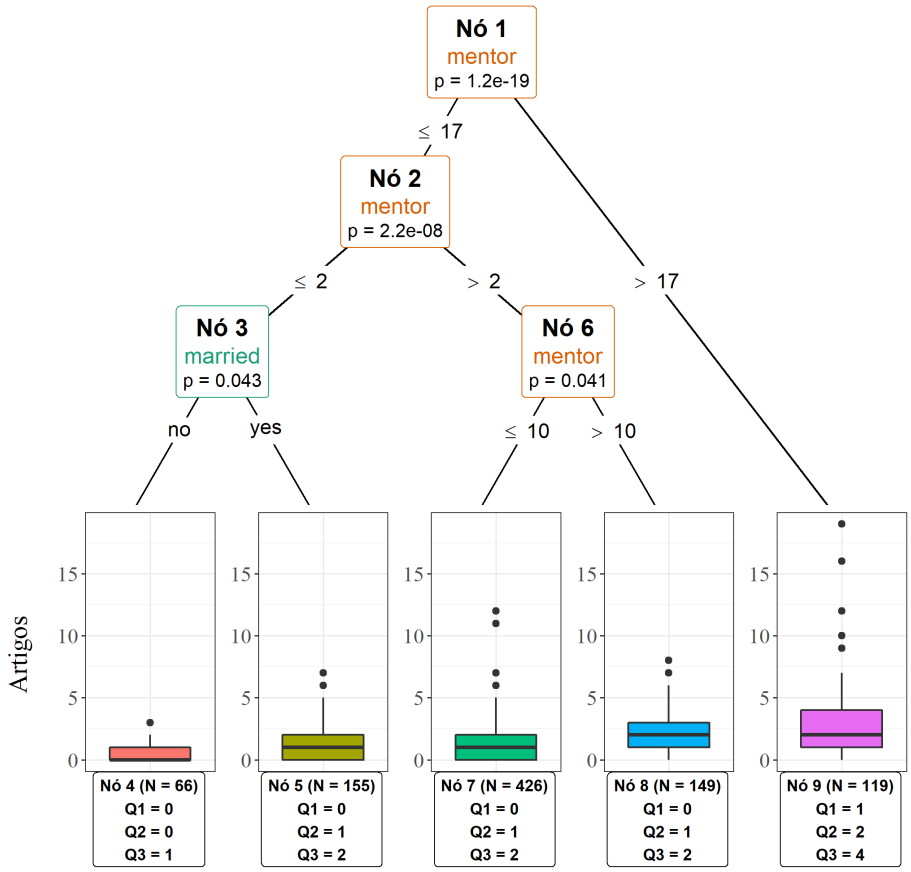

Figura 3.1: Árvore criada pelo algoritmo CTREE para o problema de artigos com $\alpha = 5\%$.
Fonte: Figura gerada pelo próprio autor.

sobre os hiper-parâmetros da árvore CTREE (como número mínimo de observações, tipo de teste de hipótese, etc) podem ser encontrados ao digitar `?ctree_control()` e `?ctree()` no terminal do `R`.

3.2 *Model-Based Recursive Partitioning*

A ideia de particionamentos recursivos baseados em modelos paramétricos consiste em utilizar uma árvore de decisão na qual há modelos paramétricos em seus nós. Diversos algoritmos foram propostos com esse conceito, tais como as *functional trees* (Gama, 2004), GUIDE (Loh, 2002), CRUISE (Kim e Loh, 2001) e LOTUS (Chan e Loh, 2004).

Além disso, ao se aplicar um modelo paramétrico nos nós, tem-se o caráter interpretativo do próprio modelo paramétrico. Ainda assim, pode-se deixar outras covariáveis responsáveis pelo particionamento da árvore, sendo essas árvores covariáveis "secundárias" na visão do pesquisador ou pesquisadora, isto é, a variável é importante para o estudo,

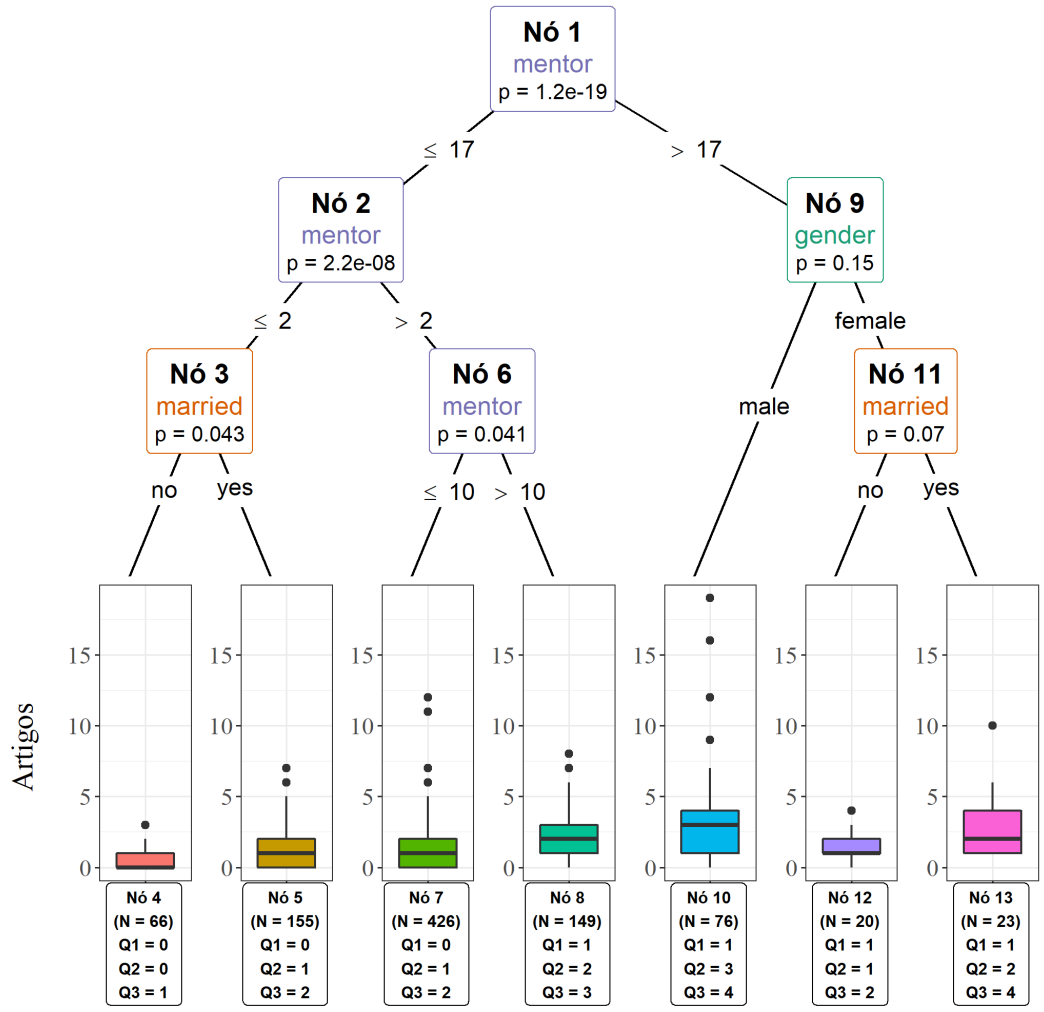

Figura 3.2: Árvore criada pelo algoritmo CTREE para o problema de artigos com $\alpha = 15\%$.

Fonte: Figura gerada pelo próprio autor.

mas não se deseja interpretá-la mais a fundo.

Para este trabalho, considerou-se o algoritmo MOB (*Model-Based Recursive Partitioning*) de Zeileis *et al.* (2008). Salienta-se que o MOB pode ser dividido em 4 passos, os quais são

1. Ajuste do modelo paramétrico à base de dados;

2. Teste para verificar a instabilidade dos parâmetros (do modelo paramétrico) em relação a um conjunto de variáveis particionadoras $\boldsymbol{Z}$;

   2.1. Se houver instabilidade, particiona-se o nó;

   2.2. Se não houver instabilidade, o processo é interrompido;

3. Com a variável de particionamento escolhida no passo anterior, encontra-se o ponto de corte ótimo para a divisão binária do nó. Este ponto é aquele que otimiza determinada função objetivo;

4. Repita o processo para cada nó filho criado.

Antes de explicar minuciosamente cada passo, é preciso estabelecer algumas notações. $\mathcal{M}(\boldsymbol{Y}, \boldsymbol{\theta})$ é o modelo paramétrico que será ajustado, com $\boldsymbol{Y}$ sendo a variável resposta e $\boldsymbol{\theta}$ o vetor de parâmetros a ser estimado. Além disso, $\boldsymbol{Z}$ é o conjunto de variáveis particionadoras da árvore, sendo diferente de $\boldsymbol{X}$. E $\boldsymbol{X}$ são covariáveis que fazem parte do modelo paramétrico como variáveis independentes.

Definição 3.7 (MOB - Estimação e função objetivo)

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} \Psi(Y_i, \theta).$$

- $\Psi(.,.)$ *é a função objetivo;*
- θ *é o parâmetro (ou vetor de parâmetros) a ser estimado;*
- $\hat{\theta}$ *é a estimação dos parâmetros;*

A função objetivo (Definição 3.7) varia dependendo do modelo paramétrico utilizado e sua estimação. Se for uma regressão linear, a estimação será feita por mínimos quadrados, logo $\Psi(Y_i, \theta)$ é a soma de quadrados dos resíduos. Já se o modelo é um modelo linear generalizado, a estimação se dá por máxima verossimilhança negativa, por consequência, $\Psi(Y_i, \theta)$ é a log-verossimilhança negativa.

3.2.1 Modelo linear generalizado (passo 1)

No passo 1 do algoritmo MOB, é preciso ajustar um modelo paramétrico à base de dados e, iterativamente, ajustar esse mesmo formato de modelo (mesma distribuição, função de ligação e covariáveis $\boldsymbol{X}$) aos nós filhos subsequentes. Para este trabalho, considerou-se a classe de modelos lineares generalizados.

Considera-se uma matriz de covariáveis $\boldsymbol{X}$ e um vetor de resposta Y. Isto posto, pode-se escrever uma relação entre Y e $\boldsymbol{X}$ da seguinte forma,

$$Y = \boldsymbol{X}\beta + \epsilon,$$

em que β é um vetor de parâmetros a ser estimado e ϵ é um vetor de erros aleatórios, e portanto, desconhecidos.

Em uma situação de regressão linear comum, deseja-se que os resíduos (valores preditos subtraídos dos valores observados) sejam normais, homocedásticos, independentes e que haja uma relação linear entre $\boldsymbol{X}$ e Y. Muitas vezes não é possível atingir a normalidade dos resíduos e/ou a relação linear entre $\boldsymbol{X}$ e Y. Quando isso ocorre, utiliza-se da classe de Modelos Lineares Generalizados (MLG).

Com o uso dessa classe, há maior flexibilidade na escolha da distribuição da variável resposta Y, uma vez que a distribuição precisa apenas pertencer à família exponencial . Além disso, com o uso da função de ligação, é possível construir uma relação que não seja necessariamente linear, podendo ser uma curva monótona.

Sendo assim, para que a variável Y seja da família exponencial (linear), é preciso que a função de densidade de probabilidade seja escrita na seguinte forma (Olsson, 2002).

Definição 3.8 (Formato da Família Exponencial)

$$f(y;\ \theta,\ \phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

- $a(.), b(.), c(.)$ *são funções;*
- *O parâmetro canônico* θ *é função do parâmetro de locação da distribuição;*
- ϕ *é o parâmetro de dispersão;*

Desse modo, a regressão (na classe de modelos lineares generalizados) pode ser escrita como

$$g(\mu_i) = \eta_i = \boldsymbol{X}_i^T\beta,$$

no qual

- $\eta_i = \boldsymbol{X}_i^T\beta$ é o preditor linear;

- $\beta = (\beta_1, ...\beta_p)^T$ é o vetor parâmetros desconhecidos que serão estimados;

- $g(.)$ é a função de ligação. Essa deve ser monótona e ser diferenciável (possuir duas derivadas);

- $\mu_i = \mathbb{E}[y_i] = b^{'}(\theta_i)$;

Ademais, dentre algumas características em relação a y (estando na classe de modelos lineares generalizados), pode-se citar sua média e variância, que são

$$\mu = \mathbb{E}[y] = b^{'}(\theta) = \frac{db(\theta)}{d\theta};$$

$$\sigma^2 = \mathbb{V}[y] = b^{''}(\theta)a(\phi) = a(\phi)\frac{d^2b(\theta)}{d\theta^2}.$$

Por fim, há também a função variância $V(y)$,

$$V(\mu) = \frac{\mathbb{V}[y]}{a(\phi)} = \frac{d^2b(\theta)}{d\theta^2} = \frac{d\mu}{d\theta}.$$

Já em relação a estimação dos β's, basta encontrar os valores que maximizam a função de máxima verossimilhança. O algoritmo para esse processo é melhor explicado por Cordeiro e Demétrio (2008).

Salienta-se que o algoritmo MOB, no momento da escrita desse trabalho, aceita uma classe bem ampla de modelos paramétricos, não precisando ser somente modelos lineares generalizados. Além disso, o próprio pacote `partykit` (pacote no `R` no qual o MOB está implementado) aceita modularizações para que o usuário implemente suas próprias distribuições.

3.2.2 Teste para instabilidade do parâmetro (passo 2)

Ajustado o modelo paramétrico no passo 1 anterior, é preciso testar se os parâmetros do modelo estão estáveis dado as variáveis particionadoras $\boldsymbol{Z}$. Para isso, serão aplicados testes generalizados de M-Flutuação. Esses testes foram propostos pelos próprios Zeileis e Hornik (2007) e têm como propósito para identificar a instabilidade para que o particionamento ser realizado.

Definição 3.9 (Hipóteses do teste de instabilidade)

Hipótese nula: $H_0 : \theta_i = \theta_0 \quad (para\ todo\ vetor\ de\ parâmetro\ i = 1, ..., p)$

Hipótese alternativa: $H_1 : \theta_i \neq \theta_0$ *(para pelo menos um* i*).*

- *A hipótese nula* H_0 *representa que todos os parâmetros do modelo paramétrico estão estáveis, ou seja, não variam de acordo com dada variável particionadora;*

- *A hipótese alternativa* H_1 *representa que, em pelo menos um dos parâmetros, há uma instabilidade, ou seja, há uma variação do parâmetro em relação a variável particionadora.*

Além disso, escrito a função objetivo $\Psi(Y_i, \theta)$ do passo 1, define-se as seguintes matrizes

$$A(\theta) = \mathbb{E}[-\Psi^{'}(Y_i, \theta)], \quad B(\theta) = \mathbb{V}[\Psi(Y_i, \theta)]$$
$$C(\theta) = \mathbb{E}[\Psi(Y_i, \theta)u(Y, \theta)^T].$$

com $\Psi^{'}(.)$ sendo a derivada parcial de $\Psi(.)$ em relação a θ e com $u(y, \theta) = \frac{\partial log(f(y,\theta))}{\partial \theta}$.

Por conseguinte, usando o teorema central do limite funcional e sob H_0 (Definição 3.9), Zeileis e Hornik (2007) enunciaram que o processo

$$W_n(t, \theta) = n^{-1/2} \sum_{i=1}^{\lfloor nt \rfloor} \Psi(Y_i, \theta). \tag{3.10}$$

converge em distribuição para um processo gaussiano de caminho contínuo $Z(.)$, de média $\mathbb{E}[Z(t)] = 0$ e covariância $Cov[Z(t), Z(s)] = min(t, s) \cdot B(\theta_0)$.

Ademais, se $B(\theta_0)$ for não singular, então o processo $B(\theta_0)^{-1/2} W_n(., \theta_0)$ converge em distribuição para um processo de movimento browniano padrão $W(.)$. Todavia, esses resultados são para os casos em que θ é conhecido.

Agora, para os casos em que θ fora estimado (com a classe de estimadores M), usando o teorema central do limite funcional e sob H_0 (Definição 3.9), Zeileis e Hornik (2007) mostraram que

$$W_n(., \hat{\theta}_n) \xrightarrow{d} Z^0(.),$$

e que

$$B(\hat{\theta}_n)^{-1/2} W_n(., \hat{\theta}_n) \xrightarrow{d} W^0(.),$$

em que $Z^0(t) = Z(t) - tZ(1)$ e $W^0(t) = Z(t) - tW(1)$ (sendo que $\xrightarrow{d}$ significa convergência em distribuição). Em seguida, para se realizar o teste de hipótese 3.9, define-se o processo de flutuação empírica.

Definição 3.11 (MOB - Processo de Flutuação Empírica)

$$efp(t) = \hat{B}_n^{-1/2} W_n(., \hat{\theta}_n).$$

Dessa forma, ao substituir os termos do problema em 3.11, obtém-se o seguinte processo de flutuação empírica,

$$W_j(t) = \hat{J}^{-1/2} n^{-1/2} \sum_{i=1}^{\lfloor nt \rfloor} \hat{\Psi}_{\sigma(Z_{ij})},$$

em que

- $\sigma(Z_{ij})$ é a permutação ordenada a qual fornece o *antirank* da observação Z_{ij} (*antirank* é a observação já ordenada) em relação a variável particionadora $Z_j = (Z_{1j}, ..., Z_{nj})^T$;
- $\hat{J}$ é a estimativa da matriz de covariância $Cov[\Psi(Y, \hat{\theta})]$;
- $W_j(t)$ a soma dos escores $\hat{\Psi}(.)$ padronizados pelo número de observações n e pela matriz de covariância $\hat{J}$.

A ideia do processo de flutuação empírica (3.11) é verificar se os escores $\hat{\Psi}$ estão aleatoriamente distribuídos em torno de uma média 0 (estão estáveis) ou se eles estão desviando de 0 em relação a Z_j (estão instáveis). Posteriormente, é preciso realizar um tratamento especial com $W_j(.)$ para quando a variável particionadora analisada for quantitativa e para quando ela for qualitativa.

Z_j quantitativa

Se Z_j for quantitativa, Zeileis *et al.* (2008) propuseram

$$\lambda_{supLM}(W_j) = \max_{i=\underline{i},...,\bar{i}} \left(\frac{i}{n} \cdot \frac{n-i}{n}\right)^{-1} \left|\left| W_j\left(\frac{i}{n}\right)\right|\right|_2^2,$$

em que é $\lambda_{supLM}(W_j)$ calcula o máximo de uma normalização quadrada L_2 do processo de flutuação empírica em questão. Assintoticamente, a estatística segue a mesma distribuição da estatística de Chow (1960) (supremo das estatísticas da razão de verossimilhança). Por fim, Hansen (1997) mostrou que é possível calcular o valor-p e a estatística de teste para tal distribuição.

Z_j qualitativa

Se Z_j for qualitativa com C categorias, Zeileis *et al.* (2008) propuseram

$$\lambda_{\chi^2}(W_j) = \sum_{c=1}^{C} \frac{|I_c|^{-1}}{n} \left|\left| \Delta_{I_c} W_j\left(\frac{i}{n}\right)\right|\right|_2^2,$$

em que $\Delta_{I_c} W_j\left(\frac{i}{n}\right)$ é um incremento do próprio processo de flutuação empírica em relação às observações em cada categoria $c = 1, ..., C$. Tal estatística tem uma distribuição assintótica χ^2 com $k \cdot (C-1)$ graus de liberdade. Tanto o valor da estatística teste quando o valor-p associado à mesma podem ser calculados como mostrado por Hjort e Koning (2002).

Destaca-se que esses escalares funcionais $\lambda(.)$ são apenas sugestões dos autores Zeileis *et al.* (2008) em relação ao algoritmo MOB. Os próprios autores, no artigo *Generalized M-fluctuation tests for parameter instability* (Zeileis e Hornik, 2007), sugerem outros escalares para diferentes contextos.

Calculado o valor-p, se este é menor que dado α nível de significância e é o menor valor-p dentre os computados para cada variável particionadora, então essa covariável é a escolhida para particionar a árvore de decisão.

3.2.3 Particionamento do nó (Passo 3)

No passo 2, foi encontrado a variável particionadora que mais causa a instabilidade nos parâmetros do modelo ajustado, desse modo, usa-se essa variável para fazer o par-

ticionamento no nó atual. Algumas estratégias são discutidas no artigo de Zeileis *et al.* (2008) a fim de se realizar tal particionamento.

No fim, Zeileis *et al.* (2008) optaram pela divisão binária, seja para variável particionadora quantitativa ou qualitativa. Essa escolha se dá pelo fato da divisão binária ser rápida e viável computacionalmente. O único porém é que esse tipo de divisão tende a criar árvores profundas verticalmente, mas isso pode ser tratado com uma poda, tópico a ser discorrido a seguir.

A estratégia da divisão binária busca encontrar o ponto ótimo que divida a base de dados em 2 partes, otimizado alguma função. No caso do MOB, a função que será otimizada no particionamento é a própria função objetivo $\Psi(.,.)$ discutida anteriormente, isto é, deseja-se encontrar o particionamento binário $P_{ótimo}$ que otimize $\sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b)$, em que

- b_1 é um lado da divisão binária e b_2 é o outro lado;
- $P_{ótimo}$ é o particionamento binário ótimo;
- θ_b são os coeficientes estimados do modelo paramétrico com os dados do conjunto b;
- $\boldsymbol{Y}_b$ são os valores da variável resposta no conjunto b;

Portanto, a função que é usada na estimação do modelo paramétrico e nos testes de instabilidade é também usada no particionamento.

Por exemplo, considere um contexto o qual alunos estudam X_1 horas por dia e Y são as notas desses alunos em simulados. Além disso, a variável X_2 é a classe de aula dos alunos (A, B ou C), sendo uma variável particionadora. Por conseguinte,

- $log()$ é a função de ligação;
- Y é a variável resposta;
- X_1 é o tempo de estudo diário;
- X_2 é a classe dos alunos (A, B ou C);

Suponha que o algoritmo MOB está no início do processo de treinamento da árvore. A variável escolhida (no exemplo, é a única covariável particionadora) para particionar é X_2, ou seja, ela conseguiu valor-p menor que α. Suponha também que é ajustado um modelo linear generalizado. Por conseguinte a função objetivo $\Psi(Y, \theta)$ é a log-verossimilhança.

Então, para particionar esse nó, considera-se todos os particionamentos binários possíveis em relação a variável X_2 classe dos alunos. Os particionamentos possíveis são

$P_1 : b_1 = (A, B)$ e $b_2 = (C)$;

$P_2 : b_1 = (A, C)$ e $b_2 = (B)$;

$P_3 : b_1 = (B, C)$ e $b_2 = (A)$;

Para cada particionamento, calcula-se $\sum_{b\in\{b_1,b_2\}} \Psi(Y, \theta_b)$, obtendo

$P_1 : \sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b) = -30$;

$P_2 : \sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b) = -25$;

$P_3 : \sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b) = -11$;

Destaca-se que $\Psi(\boldsymbol{Y}_b, \theta_b)$ é a log-verossimilhança, então o objetivo é encontrar o valor que maximize $\sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b)$. Por conseguinte, o particionamento ótimo $P_{ótimo}$ é $P_3 : b_1 = (B, C)$ e $b_2 = (A)$, criando-se dois novos nós de forma que um nó tenha como os alunos da classe (B,C) e o outro nó tenha da classe (A).

Para uma covariável particionadora contínua, o processo é semelhante. Basta fazer o particionamento binário como explicado na subseção Covariável contínua de capítulos anteriores. Em seguida, o processo é o mesmo, em que se deseje encontrar o particionamento binário $P_{ótimo}$ que otimize $\sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b)$.

Entendendo todo o processo teórico do MOB, pode-se retratá-lo por meio do Algoritmo 6. Note que o pseudocódigo do MOB é bem parecido com o do CTREE, uma vez que ambos utilizam de testes de hipótese para continuar ou não o particionamento da árvore,

além de escolher a variável particionadora com esse base nesse critério.

Algoritmo 6: Algoritmo MOB

Entrada: Covariáveis $\boldsymbol{X}_{n\times c}$, Variável resposta $\boldsymbol{Y}$ e α

Saída: Árvore MOB treinada

início

inicialização;

ajusta-se o modelo paramétrico na base de dados inteira;

teste de instabilidade dos parâmetros em relação as covariáveis $\boldsymbol{Z}$;

if *se todos os valores-p forem maiores que* α **then**

pare o crescimento da árvore;

else

selecione a variável particionadora com menor valor-p;

encontre o particionamento que otimize $\sum_{b\in\{b_1,b_2\}} \Psi(\boldsymbol{Y}_b, \theta_b)$;

aplique o Algoritmo MOB em cada partição nova;

end

fim

3.2.4 Poda

Feito todos os 4 passos do algoritmo MOB, a árvore está concluída. Contudo, pode ocorrer que ela tenha crescido demais, consequentemente ficando sobreajustada aos dados. Para contornar esse problema, Zeileis *et al.* (2008) utilizam de técnicas de poda na qual se baseiam nos critérios de informação (AIC ou BIC) dos modelos paramétricos ajustados.

Suponha que em um dado nó é feito uma partição binária e a partir dele crescem dois novos nós filhos. Se os modelos paramétricos ajustados nos nós filhos não tiverem o critério de informação menor que o nó pai, então significa que o modelo paramétrico que está no nó filho não melhora o desempenho da árvore (em relação ao pai).

Dado que isso ocorre, o nó filho é podado e o nó pai passa a ser o novo nó filho. O processo continua iterativamente, até que o modelo paramétrico, em algum nó filho, obtenha o critério de informação (AIC ou BIC) menor que o critério do nó pai.

Por outro lado, lembra-se que, assim como ocorreu com as árvores de inferência condicional (CTREE), as árvores criadas pelo MOB crescem com base em testes estatísticos. Então, por natureza, elas não tendem a crescer tanto.

3.2.5 Exemplo

Considerou-se a base de dados sobre a qualidade do ar em Nova Iorque, de maio até setembro de 1973. A base de dados consiste em 153 observações, contudo após a retirada das observações com caselas ausentes, o tamanho da base de dados se reduziu para 111 observações (Chambers *et al.*, 2018).

Tabela 3.2: Descrição da base sobre qualidade do ar.

Nome	**Sigla**	**Descrição**	**Tipo**
Ozônio	*Ozone*	Média do ozônio em parte por bilhão	quantitativa contínua
Radiação solar	*Solar.R*	Radiação solar de frequência de de banda entre 4000 a 7700	quantitativa contínua
Velocidade do Vento	*Wind*	Velocidade do vento média em milhas	qualitativa nominal
Temperatura	*Temp*	Temperatura diária máxima em °F	quantitativa contínua

Fonte: Tabela gerada pelo próprio autor.

Para aplicar o MOB, considerou-se que a variável resposta $\boldsymbol{Y}$ seria ozônio e a covariável $\boldsymbol{X}$ seria somente a temperatura. Já as outras variáveis, as quais são radiação solar, velocidade do vento e temperatura, seriam variáveis particionadoras $\boldsymbol{Z}$. Desse modo, o objetivo é saber de que forma a temperatura influencia no ozônio, dadas as outras variáveis particionadoras da árvore.

Além disso, utilizou-se a classe de modelos lineares generalizados, com ozônio seguindo distribuição gama. A função de ligação foi a logarítmica. Tal como se dá em

$$log(Y) = \beta_0 + \beta_1 X_1 \mid X_2 + X_3;$$

em que

- $log()$ é a função de ligação;
- Y é o ozônio;
- X_1 é a temperatura;
- X_2 é a velocidade do vento;
- X_3 é a radiação solar.

Por padrão do algoritmo, número mínimo de observações que um nó deve ter (tanto para o particionamento quanto para o nó final) é 20, sendo 10 vezes o número de parâmetros a ser estimado (considerando cada nó). Então como deseja-se estimar β_0 e β_1 (em cada nó), o número mínimo de observações foi $10 \cdot 2 = 20$.

Inicialmente, considerou-se um $\alpha = 5\%$ para que um nó seja particionado, e não foi considerado nenhuma técnica de pós poda. Desse modo, obteve-se a Figura 3.3.

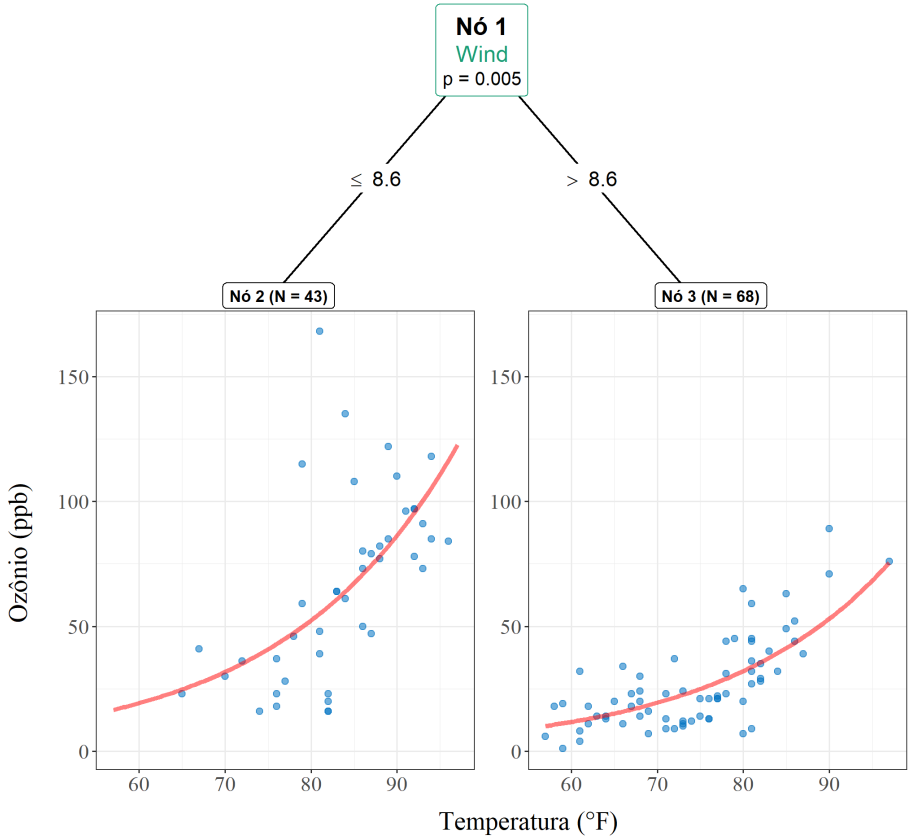

Figura 3.3: Árvore criada pelo algoritmo MOB para o problema de qualidade do ar, com $\alpha = 5\%$.

Fonte: Figura gerada pelo próprio autor.

Estudando a Figura 3.3, nota-se que a variável velocidade do vento (*Wind*) foi a responsável pelo particionamento da árvore, criando assim dois modelos paramétricos. O nó 2 possui um elevação mais rápida, já o nó 3 possui uma elevação menor. Essa diferença pode ser observada na Tabela 3.3. No modelo do nó 2, há um crescimento de 5.13% do ozônio para cada 1 °C de temperatura. Já para o nó 3, há crescimento maior, sendo de 5.15% do ozônio para cada 1 °C de temperatura.

Por mais que a análise de resíduos não sejo comentada no artigo de Zeileis *et al.* (2008),

Tabela 3.3: Coeficientes estimados nos modelos paramétricos do MOB no problema de qualidade do ar ($\alpha = 5\%$).

Nó	Intercepto (β_0)	(β_1)	exp(β_1)
2	-0.0433	0.05	1.0513
3	-0.5523	0.0503	1.0515

Fonte: Tabela gerada pelo próprio autor.

é interessante verificar o gráfico de resíduos contra predito para cada nó. Analisando a Figura 3.4, evidencia-se que os resíduos parecem estar aleatórios no nó 2. Todavia, no nó 3, parece haver um leve "funil" conforme o ozônio aumenta, indicando assim que um modelo que trata da variabilidade pode vir a ser mais adequado.

Considera-se agora um $\alpha = 100\%$, isto é, a árvore cresce indefinidamente, não dependendo do teste de hipótese para verificar a instabilidade. Desse modo, há uma maior flexibilidade na escolha da covariável particionadora. Salienta-se que o tamanho mínimo do nó ainda é 20, como no exemplo anterior. Ademais, não foi aplicada nenhuma técnica de poda.

Estudando a Figura 3.5, percebe-se que a variável velocidade do vento ($Wind$) passa a aparecer duas vez na árvore. Além disso, a radiação solar ($Solar.R$) passa a particionar também. Porém, nota-se que a regressão dos nós 3, 4 e 6 são semelhantes. Apenas o nó 7 apresenta uma regressão diferente das demais, retratando o efeito de $\alpha = 100\%$.

Também é possível observar os coeficientes do modelo de cada nó por meio da Tabela 3.4. Nota-se que os nós 3 e 4 são bem semelhantes no crescimento de ozônio dado o aumento de 1 de temperatura, dado que ambos crescem torno de 4%. Em contrapartida, para o aumento de 1 de temperatura, o ozônio no nó 6 cresce 8.18% enquanto no nó 7, cresce apenas 3.73%.

Tabela 3.4: Coeficientes estimados nos modelos paramétricos criados o MOB no problema de qualidade do ar ($\alpha = 100\%$).

Nó	Intercepto (β_0)	(β_1)	exp(β_1)
3	-0.1070	0.0476	1.0487
4	0.7538	0.0429	1.0439
6	-2.7523	0.0786	1.0818
7	0.4643	0.0366	1.0373

Fonte: Tabela gerada pelo próprio autor.

Em relação aos resíduos da árvore MOB construída com $\alpha = 100\%$, nota-se que no nó 6 e 7, os resíduos parecem estar bem espalhados. Já no nó 3, percebe-se um leve comportamento de "funil", isto é, conforme o valor predito aumenta, o resíduo tende a se

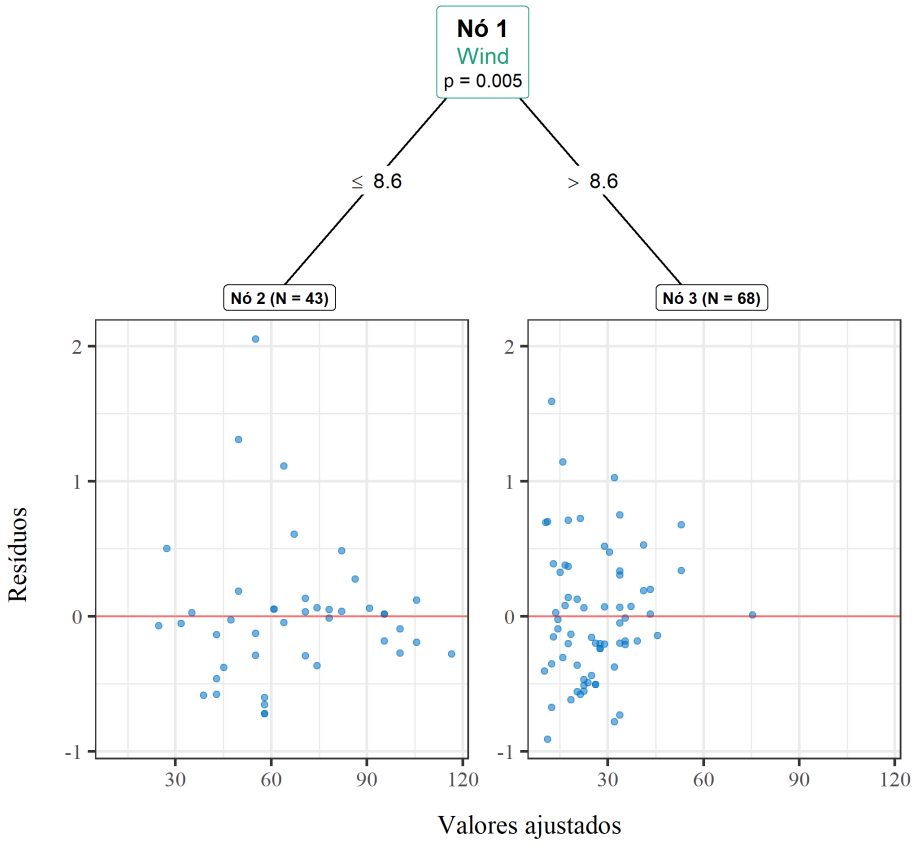

Figura 3.4: Análise dos resíduos da árvore criada pelo algoritmo MOB para o problema de qualidade do ar, com $\alpha = 5\%$.
Fonte: Figura gerada pelo próprio autor.

afunilar. Por fim, o comportamento do nó 4 está levemente com uma tendência crescente.

Portanto, o nível de significância é de extrema importância para o desenvolvimento de uma árvore de decisão no algoritmo MOB, como ocorreu no algoritmo CTREE. Além disso, nota-se que o MOB fornece aspectos poderosíssimos no quesito de interpretabilidade e aspectos visuais, fornecendo, não somente uma ideia de como o modelo se comporta, mais também qual covariável é responsável pelo particionamento do conjunto de dados.

Salienta-se que o MOB (para a classe de modelos lineares generalizados) está implementado no *software* `R` (R Core Team, 2021) por meio da função `glmtree()` do pacote *partykit* (Hothorn e Zeileis, 2015). Mais informações sobre os hiperparâmetros da árvore MOB (como a poda, tipo de teste de hipótese, etc) podem ser encontrados ao digitar `?mob_control()` no terminal do `R`.

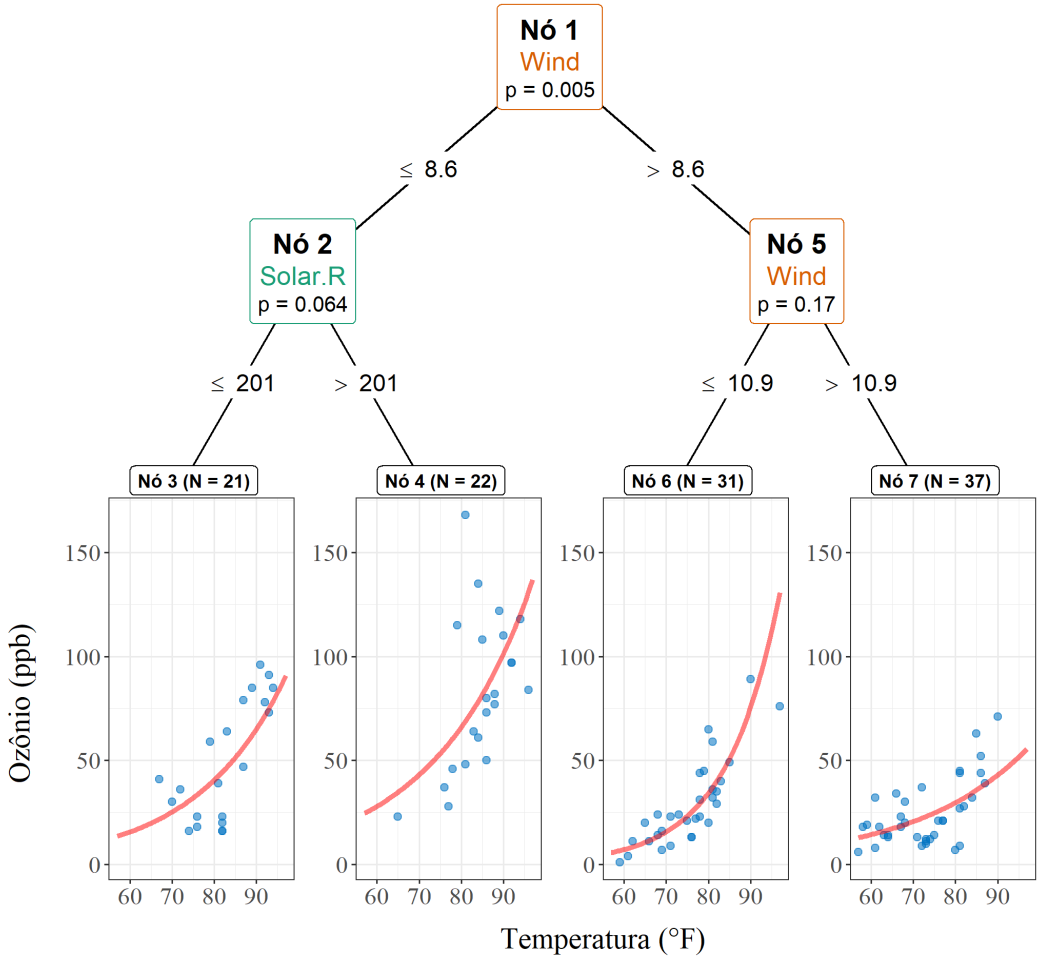

Figura 3.5: Árvore criada pelo algoritmo MOB para o problema de qualidade do ar, com $\alpha = 100\%$.

Fonte: Figura gerada pelo próprio autor.

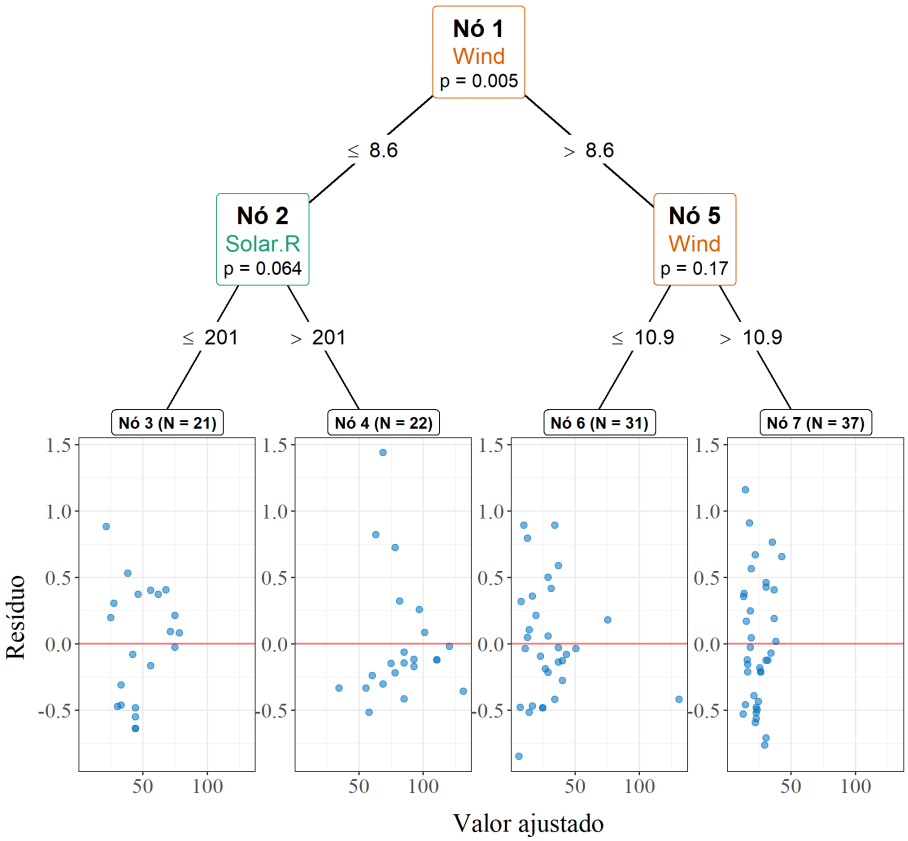

Figura 3.6: Análise dos resíduos da árvore criada pelo algoritmo MOB para o problema de qualidade do ar, com $\alpha = 100\%$.

Fonte: Figura gerada pelo próprio autor.

Capítulo 4

Aplicações e Estudo de casos

Neste capítulo será apresentado um exemplo com dados simulados. Além disso, também serão apresentados aplicações em dois conjuntos de dados proveniente de um estudo sobre perdas pré-abates de frangos de corte, com os dados coletados de um batedouro comercial no Estado de São Paulo, no ano de 2006. Tais dados estão melhores descritos em Vieira *et al.* (2015).

4.1 Exemplo simulado

Dado que o problema a ser desenvolvido na segunda parte do trabalho de graduação trata-se de um problema de regressão (árvore de decisão a qual tem a variável resposta quantitativa), serão explorados os algoritmos que tratam disso, tais como o CART©, CTREE e o MOB (para os dois últimos, fixou-se $\alpha = 0.01$). Além deles, é interessante compará-los também com algoritmos de aprendizado de máquina mais poderosos, tais com o XGBoost (Chen *et al.*, 2021) e Florestas Aleatórias (Wright e Ziegler, 2017).

Para isso, gerou-se 14000 observações da seguinte forma,

$$\begin{aligned} Y = 100 + X_1 - \exp(1.5 \cdot X_2) + 2 \cdot ln(X_{12}) + \\ + 0.7 \cdot X_3 + 0.3 \cdot X_4 + X_5 + 0.3 \cdot (X_{345}) + \epsilon, \end{aligned}$$

em que,

- X_1 e X_2 são valores gerados uma distribuição uniforme entre 0 e 1;
- $X_{12} = X_1 + X_2$; X_3 são valores gerados uma distribuição uniforme entre 3 e 4;

- X_4 são valores gerados uma distribuição uniforme entre 10 e 12;
- X_5 são valores gerados uma distribuição uniforme entre 5 e 6;
- $X_{345} = \dfrac{X_3 + X_4}{log_5(X_5)}$;
- ϵ seria um ruído aleatório com média 0 e desvio padrão 4 gerados da distribuição normal;

Nota-se que há algumas relações lineares e não lineares, como funções logarítmicas e exponencias. Ademais, aplicou-se também algumas interações $(ln(X_{12}))$ e (X_{345}).

Após a geração, considerou-se uma base de dados com Y sendo a variável resposta e X_1, X_2, X_3, X_4 e X_5 as covariáveis. Em seguida, dividiu-se 70% das observações para um conjunto de treinamento (será utilizado para o ajuste dos modelos) e 30% para um conjunto de teste (será utilizado para verificar o poder preditivo). Em relação à medida de erro, considerou-se o erro quadrático médio.

Para os intervalos de confiança, considera-se

$$\hat{R}(g) = \frac{1}{s}\sum_{i=1}^{s}(Y_i - g(\boldsymbol{X}_i))^2 = \frac{1}{s}\sum_{i=1}^{s} W_i.$$

no qual W_i será o cálculo do risco (erro quadrático médio). Dessa forma,

$$\hat{S}^2 = \frac{1}{s}\sum_{i=1}^{s}(W_i - \hat{R}(g))^2.$$

Logo, considerando uma amostra suficientemente grande e uma distribuição normal (confiança de 95%), tem-se então

$$\hat{R}(g) \pm 1.96\sqrt{\frac{1}{s}\hat{S}^2}$$

Já para os intervalos de confiança, considerou-se o quantil $1 - \frac{\alpha}{2}$ da normal padrão vezes o desvio padrão do erro quadrático médio.

4.1.1 CART©

Dado a execução do CART©, encontrou-se um parâmetro de complexidade para a poda no valor de 0.01. Desse modo, ajustou-se a árvore que está na Figura 4.1. Nota-se que apenas a covariável X_1 é importante para o particionamento da árvore. Além disso,

entre os nós finais, os *boxplots* apresentam um comportamento parecido, indicando não haver mudanças no comportamento dos dados.

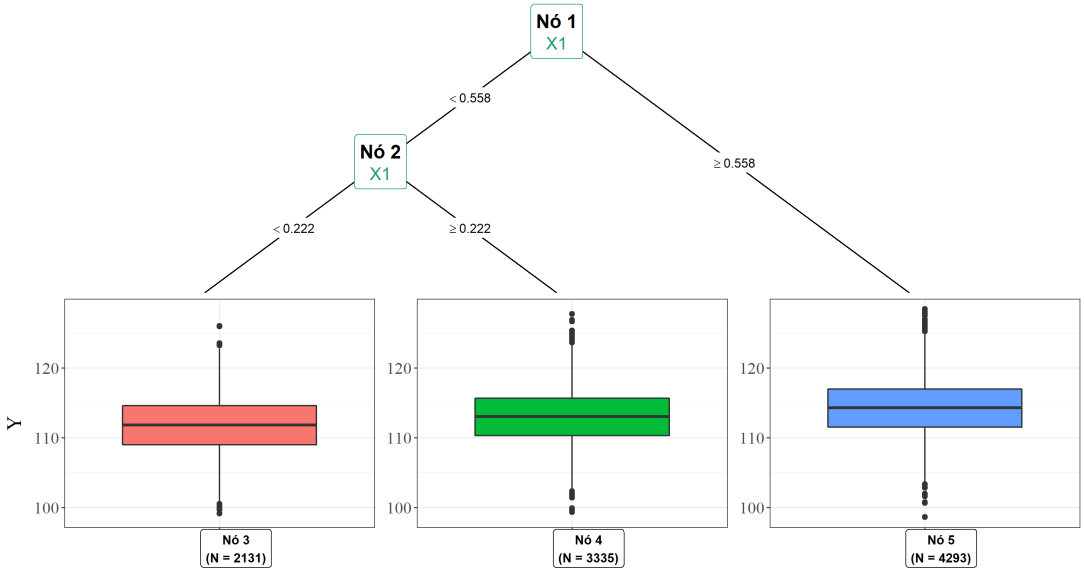

Figura 4.1: Árvore criada pelo CART© (problema de simulação).
Fonte: Figura gerada pelo próprio autor.

4.1.2 CTREE

Para o CTREE, optou-se por um $\alpha = 1\%$ para a realização do particionamento da árvore (testes de permutação). Com isso obteve-se uma árvore relativamente grande, não podendo ser visualizada no presente trabalho. Lembrando-se que o CTREE não possui um método de poda, então seu crescimento é determinando unicamente pelo $\alpha = 1\%$.

4.1.3 MOB

Para o MOB, considerou-se as covariáveis X_1 e X_2 como sendo as covariáveis do modelo paramétrico. Por outro lado, as covariáveis X_3, X_4 e X_5 serão as covariáveis particionadoras. Além disso, é preciso escolher alguma distribuição para a modelagem da parte paramétrica, para tal, observa-se a Figura 4.2.

Pela Figura 4.2, nota-se uma distribuição composta inteiramente por valores positivos. Além disso, percebe-se uma certa simetria, quase parecendo com distribuição normal. Ainda assim, escolhe-se a distribuição gama com a função de ligação logarítmica (MLG) para a parte do modelo paramétrico. Após a execução do MOB, obteve-se a árvore a

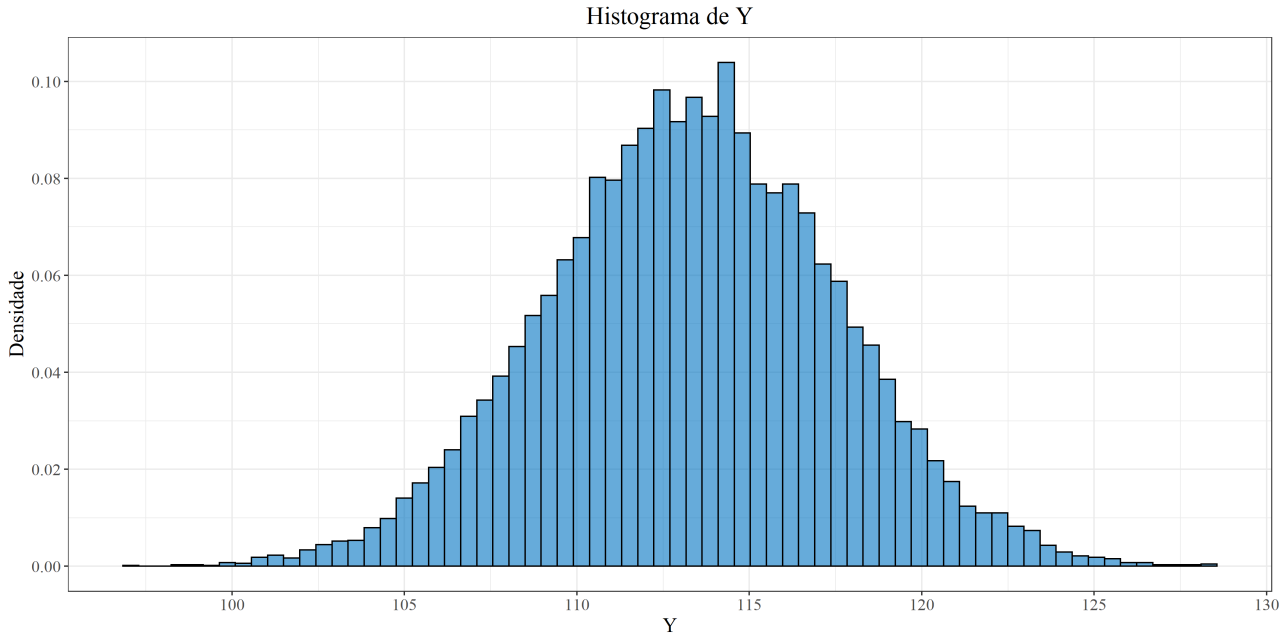

Figura 4.2: Histograma dos valores gerados de Y (problema de simulação).
Fonte: Figura gerada pelo próprio autor.

seguir na Figura 4.3.

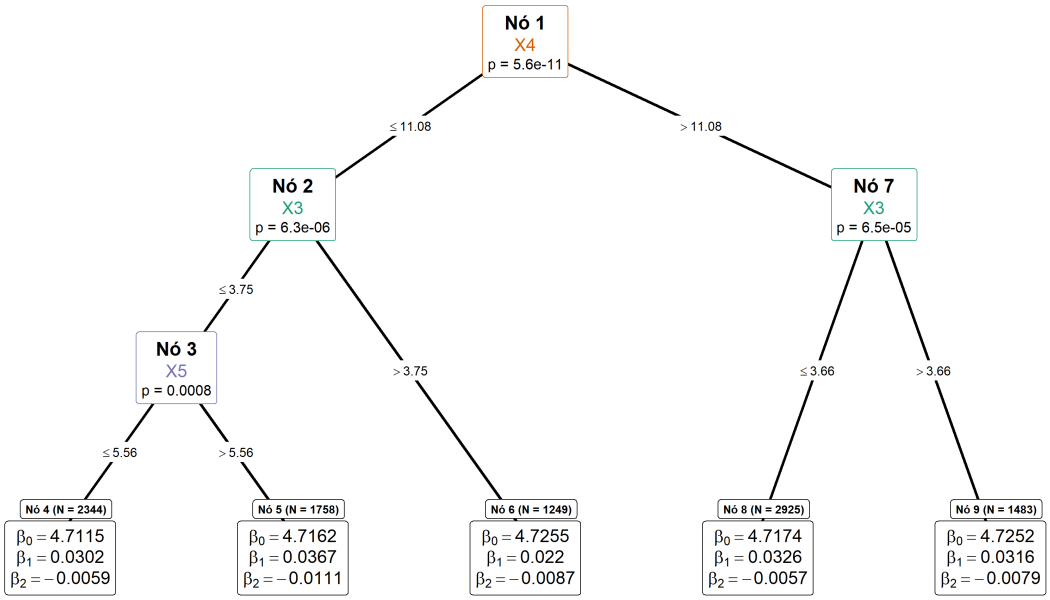

Figura 4.3: Árvore gerada pelo MOB (problema de simulação).
Fonte: Figura gerada pelo próprio autor.

Estudando a Figura 4.3, percebe-se que essa foi diferente da árvore do CTREE, que estava muito grande. Além disso, como X_1 foi usada no modelo paramétrico, tal covariável não aparece no particionamento da árvore. Além disso, é interessante notar os coeficientes β_1 foram alguns decimais maiores do que o β_2, indicando assim que X_1 influencia mais do

que X_2 nos modelos paramétricos.

Outro detalhe que é importante ressaltar é a velocidade computacional do MOB. Enquanto os outros algoritmos levaram apenas alguns poucos segundos (ou no máximo 1 minuto), o MOB levou 5 minutos para ser executado completamente.

Ademais, observe que existe uma interação entre X_1 e X_2 no processo gerador dos dados simulados. Como apenas os efeitos principais de X_1 e X_2 foram colocados na parte, não é possível que árvore modele a interação entre X_1 e X_2, daí a importância de se conhecer bem a relação da variável resposta com as covariáveis.

4.1.4 Floresta aleatória

Em relação à floresta aleatória, ela foi executada com uma otimização nos hiperparâmetros número de árvores e número mínimo de observações por nó. Tal otimização se deu por otimização Bayesiana (Yan, 2021), utilizando 30 iterações e 5 valores iniciais aleatórios e validação cruzada com 5 *folds*. O intervalo para o número de árvore e o número de observações mínimo por nó foi de 100 a 1000 e de 5 a 30, respectivamente.

Depois da otimização, obteve-se 1000 como sendo o número de árvores e 30 o número de observações por nó. Estudando a Figura 4.4, nota-se que como o CART©, a covariável X_1 foi de extrema importância na criação do seu modelo. Em seguida, vem X_2 com uma importância intermediária, e depois X_3, X_4 e X_5 respectivamente.

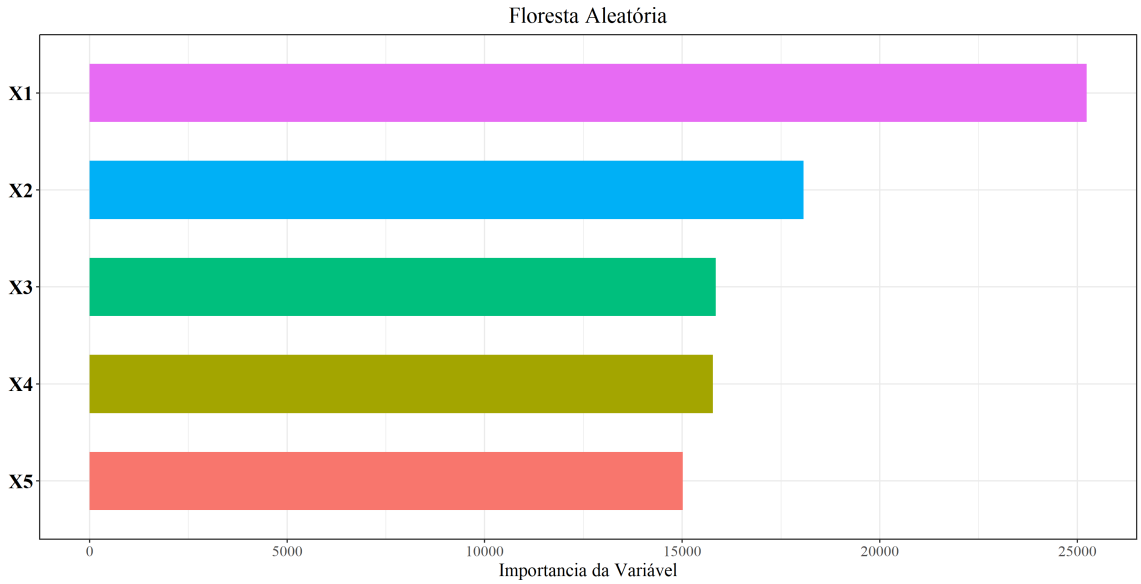

Figura 4.4: Importância das covariáveis na floresta aleatória (problema de simulação).
Fonte: Figura gerada pelo próprio autor.

4.1.5 XGBoost

Para o algoritmo XGBoost, o único hiperparâmetro otimizado foi o número de árvores (iterações). Para isso, considerou-se uma validação cruzada (no conjunto de treinamento) de *10 folds*, um total de árvores sendo 1000 e um parada breve de 30. Portanto, deseja-se encontrar o número de árvore que obtém o menor erro nos conjuntos de validação.

Após a execução da validação cruzada, obteve-se que a melhor iteração (número de árvores) foi 19. Por conseguinte, esse número será o número total de árvores do XGBoost. Analisando a Figura 4.5, percebe-se o mesmo comportamento que ocorreu na floresta aleatória, em que X_1 e X_2 são as covariáveis mais importantes e, em seguida, X_3, X_4 e X_5.

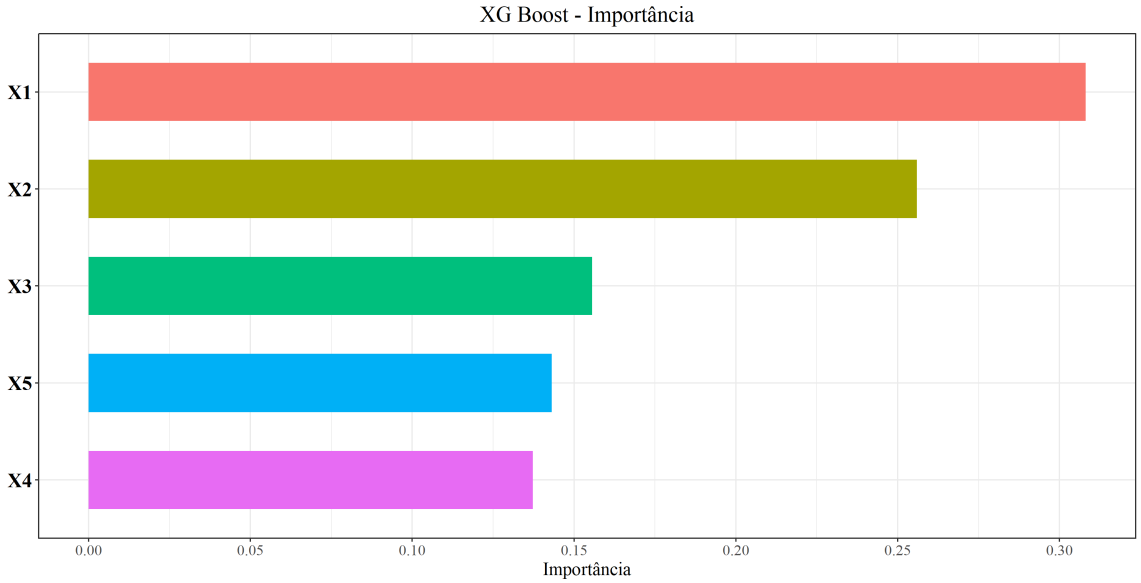

Figura 4.5: Importância das covariáveis no XGBoost (problema de simulação).
Fonte: Figura gerada pelo próprio autor.

4.1.6 GLM

Ajustou-se um modelo linear generalizado considerando como covariáveis do modelo paramétrico as variáveis X_1, X_2, X_3, X_4 e X_5 (diferente do MOB, que foi apenas X_1 e X_2). Além disso, atribui-se a distribuição gama para a variável resposta Y com o uso da função de ligação logarítmica (como no MOB). Após a execução, obteve-se a Tabela 4.1. Por ela, nota-se que todos os coeficientes possuem significância estatística para explicar Y.

Tabela 4.1: Coeficientes e significância do MLG (problema de simulação).

Covariável	Coeficiente	Valor-p
(Intercepto)	4.6090	$< 2 \cdot 10^{-16}$
X_1	0.0311	$< 2 \cdot 10^{-16}$
X_2	-0.0075	$< 2 \cdot 10^{-16}$
X_3	0.0091	$< 2 \cdot 10^{-16}$
X_4	0.0046	$< 2 \cdot 10^{-16}$
X_5	0.0048	$< 2 \cdot 10^{-16}$

Fonte: Tabela gerada pelo próprio autor.

4.1.7 Desfecho

Ajustados os modelos explicados anteriormente, calcula-se seus erros quadráticos médios (EQM) em relação a amostra de teste, obtendo a Tabela 4.2. Percebe-se que todos os erros pontuais foram bem próximos uns dos outros, sendo o CART©, CTREE e Floresta aleatória aqueles com piores desempenho, MOB e XGBoost com um desempenho intermediário e o MLG com o melhor desempenho, isto é, o menor EQM.

Tabela 4.2: EQMs e seus intervalos de confiança para cada algoritmo (problema de simulação) calculados na amostra de teste.

Algoritmo	EQM	Limite inferior	Limite superior
CART©	16.79662	16.08498	17.50826
CTREE	16.57734	15.87954	17.27515
MOB	16.35601	15.66563	17.04639
Floresta aleatória	16.46117	15.77584	17.14649
XGBoost	16.38458	15.70005	17.06911
MLG	16.28504	15.59440	16.97568

Fonte: Tabela gerada pelo próprio autor.

Isso ocorre pois, mesmo incluindo funções não lineares como o logarítmica e a exponencial, a estrutura da geração de Y ainda continua bem linear, por isso o MLG apresenta melhor desepenho. Com essa mesma lógica, o MOB mesmo usando apenas X_1 e X_2 na parte paramétrica, consegue o segundo menor erro. Por outro lado, já era esperado que o CART© não desempenhasse tão bem, uma vez que os outros algoritmos são simplesmente evoluções do próprio CART©.

Por fim, o CTREE não desempenhou bem uma vez que ele não possui um método de poda e sua árvore cresce dependendo apenas do valor de α relativo aos testes de permutação, sendo assim bem sensível ao tamanho da amostra.

4.2 Base de dados da Etapa 1

As metodologias vistas anteriormente serão aplicadas em um contexto real, dando um foco maior para a técnica de particionamento recursivo baseado em modelos paramétricos (*Model-Based Recursive Partitioning*). Tal contexto é referente a perda pré-abate de frangos de corte, com os dados coletados de um abatedouro comercial no Estado de São Paulo, no ano de 2006. Mais de 13 mil caminhões foram catalogados, bem como os possíveis variáveis influentes nas perdas, as quais foram fatores bioclimáticos, horário diurno do transporte e tempo de espera pré-abate (Vieira *et al.*, 2015).

Esta base de dados consiste nas seguintes variáveis (Tabela 4.3).

Tabela 4.3: Descrição da base de dados sobre a Etapa 1 de frangos.

Nome	**Sigla**	**Descrição**	**Tipo**
Número de Mortes	*NumMort*	Número de avez mortas por caminhão	Quantitativa discreta
Tempo de Espera	*TempEsp*	Tempo de Espera do caminhão no momento de descarga (em minutos))	Quantitativa contínua
Temperatura Externa	*TempExt*	Temperatura Externa da granja (em °C)	Quantitativa contínua
Umidade Externa	*URExt*	Umidade externa da granja (em %)	Quantitativa contínua
Estação	*Estacao*	Estação do ano do transporte	Qualitativa nominal (verão, outono, inverno e primavera)
Turno	Turno	*Turno do transporte*	Qualitativa nominal (dia, tarde e noite)
Total de aves	*TotalAves*	Total de aves em cada caminhão	quantitativa discreta

Fonte: Tabela gerada pelo próprio autor.

O experimento consistiu em 13939 caminhões saindo de granjas do estado de SP até chegar ao abatedouro para descarregar os frangos para o abate. Tais caminhões realizaram trajetos em turnos e estações do ano diferentes. No momento do descarregamento, cada caminhão teve sua temperatura e umidade externas medidas, além do tempo de espera até descarregar cronometrado (tempo que o caminhão ficou parado esperando).

No momento da descarga, aves de determinados caminhões podem ter morrido (seja ao longo da viagem ou no momento de espera), sendo essa a variável de interesse ($\boldsymbol{Y}$). Para complementar, cada caminhão também teve o total de aves contabilizado.

4.2.1 Análise

Análise Descritiva

Percebe-se pela Tabela 4.4 que o número de aves mortas é bem heterogêneo, uma vez que o desvio padrão é de 30.2 e o coeficiente de variação é 2.77. Além disso, tem-se que a média é de 10 aves mortas por caminhão. Nota-se também que há pelo menos um caminhão com nenhuma ave. Pode-se ter uma noção do número de aves mortas de acordo com o total de aves (Tabela 4.5).

Tabela 4.4: Medidas descritivas da variável número de mortes por caminhão.

Min	Q1	Med	Media	Q3	Max	Dp	CV
0	5	8	10.885	12	2280	30.253	2.779

Fonte: Tabela gerada pelo próprio autor.

Tabela 4.5: Medidas descritivas da variável número total de aves por caminhão.

Min	Q1	Med	Media	Q3	Max	Dp	CV
55	3213	3402	3357.048	3672	4860	467.461	0.139

Fonte: Tabela gerada pelo próprio autor.

Como o número de aves mortas está muito concentrado em valores pequenos e há a presença alguns valores grande (Tabela 4.4), será trabalhado com uma transformação do número de mortes nesta análise descritiva. Tal transformação é parecida com a logito (log de chance) e se dá como

$$Morte\_Transformada = \log\left(\frac{\frac{NumMort+1}{TotalAves+1}}{1-\frac{NumMort+1}{TotalAves+1}}\right).$$

Observando a Figura 4.6, percebe-se o comportamento simétrico da temperatura externa, tempo de espera e morte transformada, com destaque para os dois últimos na presença de valores extremos. Já a umidade externa possui um comportamento assimétrico a esquerda.

Pelas Tabelas 4.6 e 4.7, nota-se que a distribuição dos caminhões é semelhante para as estações do ano, uma vez que todos os caminhões que descarregaram, no ano de 2006, estão presentes na base de dados. Por outro lado, a maioria dos caminhões trafegaram durante o turno da noite.

Pela Figura 4.7, nota-se que apenas a variável temperatura e umidade estão levemente correlacionadas negativamente. Todas as outras covariáveis não apresentaram correlação relevante.

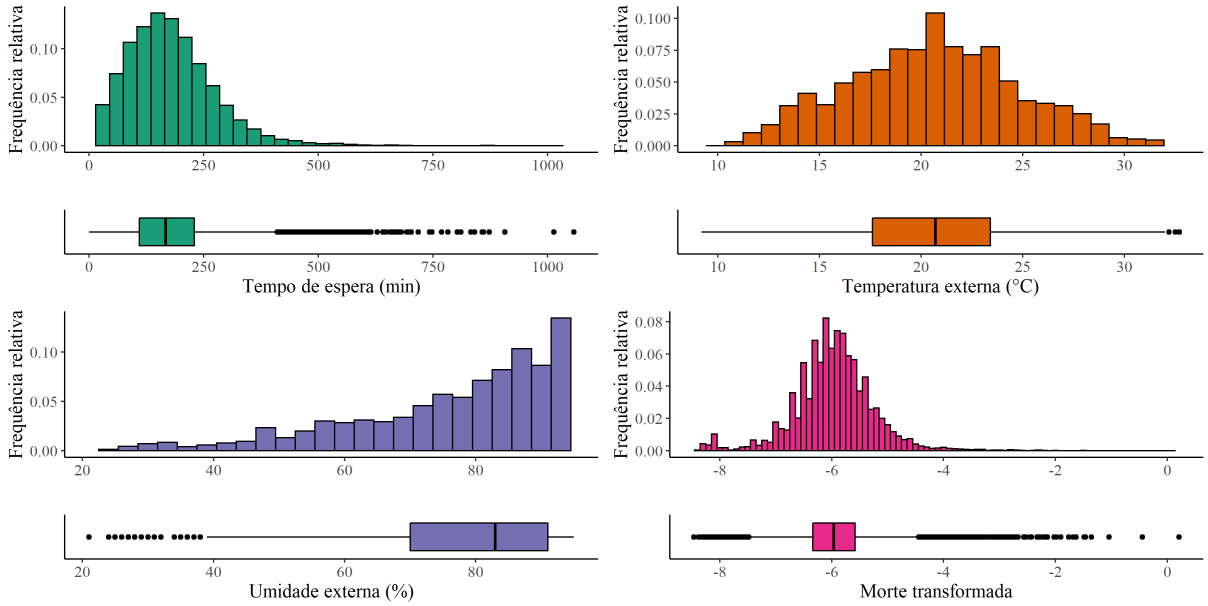

Figura 4.6: Histograma e *box-plots* das variáveis quantitativas do banco da ETAPA 1.
Fonte: Figura gerada pelo próprio autor.

Tabela 4.6: Frequência percentual das estações.

verao	outono	inverno	primavera
0.239	0.234	0.264	0.263

Fonte: Tabela gerada pelo próprio autor.

Tabela 4.7: Frequência percentual dos turnos.

manha	tarde	noite
0.272	0.271	0.457

Fonte: Tabela gerada pelo próprio autor.

Na Figura 4.8A, observa-se que a morte é levemente menor para as estações de inverno e primavera em relação às estações de verão e outono. Já na 4.8B, percebe-se que o *boxplot* para o turno da tarde está levemente acima do que os *boxplots* de outros turnos.

Na Figura 4.9, destaca-se a queda abrupta da média da morte transformada quando é considerado as estações de outono e inverno.

4.2.2 Modelagens na base ETAPA 1

Continuando com a base de dados da ETAPA1, dividiu-se aleatoriamente o mesmo em 80% amostra de treino (para os ajustes dos modelos) e 20% amostra de teste (para os cálculos de erros). Utilizou-se o erro quadrático médio e o erro absoluto médio, assim como seus respectivos intervalos de confiança com quantil 0.975 da normal padrão vezes o desvio padrão do erro calculado.

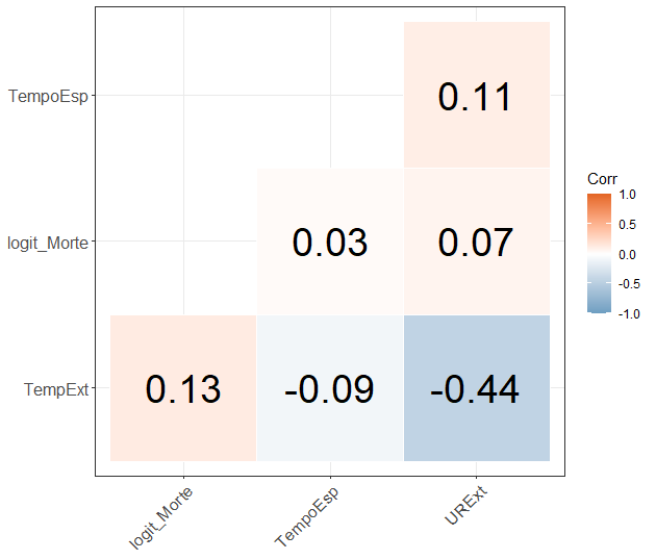

Figura 4.7: Correlação de *Spearman* para as variáveis quantitativas da ETAPA 1.
Fonte: Figura gerada pelo próprio autor.

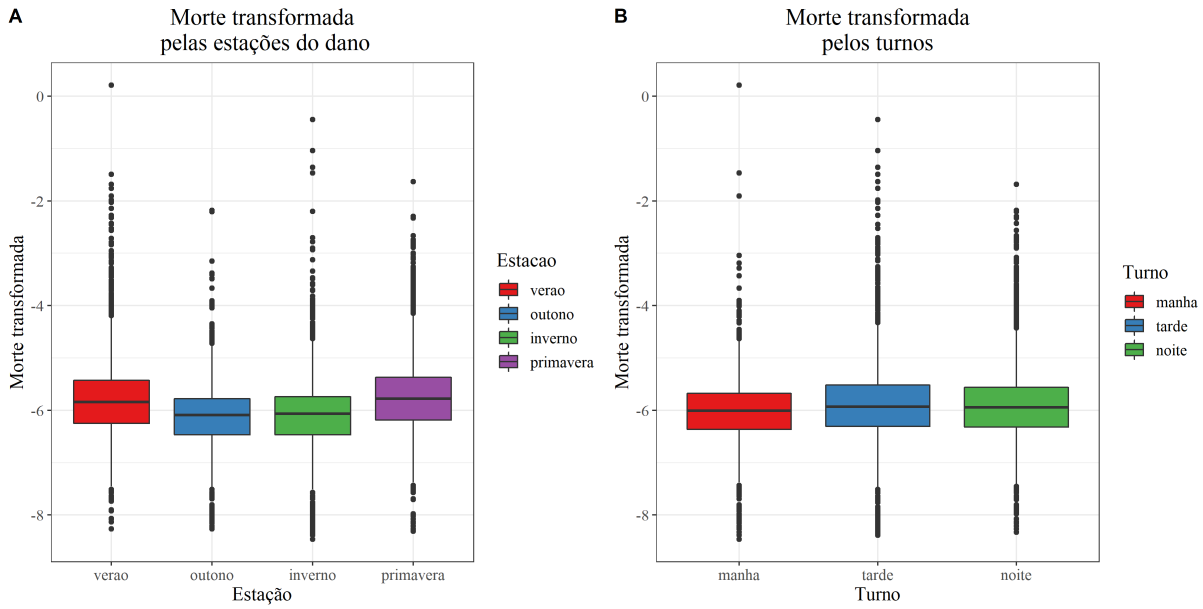

Figura 4.8: *Boxplots* da morte transformada pelas covariáveis qualitativas.
Fonte: Figura gerada pelo próprio autor.

MOB 1

Tem-se que a variável resposta será $\boldsymbol{Y}$ número de mortes e a covariável do modelo paramétrico será apenas X_1 tempo de espera. Todas as outra covariáveis (Temperatura Externa, Umidade Externa, Estação e Turno) ficarão como covariáveis particionadoras. A escolha de covariáveis da parte paramétrica e da parte particionadora fica a cargo do pesquisador. Para esse estudo, considerou-se colocar na parte paramétrica covariáveis que são controláveis por parte do abatedouro (pode-se melhorar a logística do local para

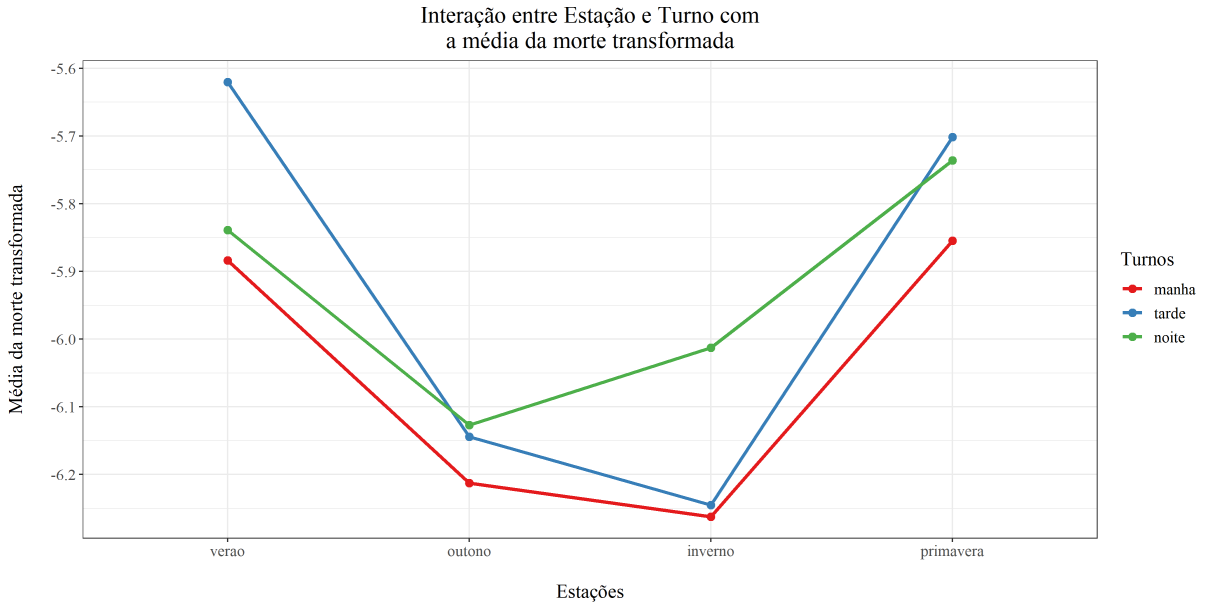

Figura 4.9: Média da morte transformada pelas covariáveis qualitativas.
Fonte: Figura gerada pelo próprio autor.

diminuir o tempo de espera) e colocar covariáveis naturais (não depende do fator humano) na parte particionadora.

Ademais, pressupôs-se que

$$Y_i \sim Poisson(\mu_i),$$

por conseguinte, pode-se escrever a média da distribuição como $\mu_i = \lambda_i t_i$, de forma que

- λ_i denota a proporção de aves mortas do i-ésimo caminhão;
- t_i detona o total de aves do i-ésimo caminhão;

Portanto, considerando a função de ligação log(), tem-se que

$$\begin{aligned} \log(\mu_i) &= \log(\lambda_i t_i) \\ &= \log(\lambda_i) + \log(t_i) \\ &= \beta_0 + \beta_1 X_{1,i} + \log(t_i), \end{aligned}$$

em que $\beta_0 + \beta_1 X_1$ é a parte desconhecida do modelo e $\log(t_i)$ é parte conhecida (chamada de *offset*).

Para este modelo (chamaremos de MOB 1), optou-se por otimizar o nível α de significância. Para isso, utilizou-se de otimização Bayesiana, com 5 iterações e 2 passos

aleatórios e validação cruzada de 5 *folds*. Após a execução da otimização, obteve-se um $\alpha = 0.00068$.

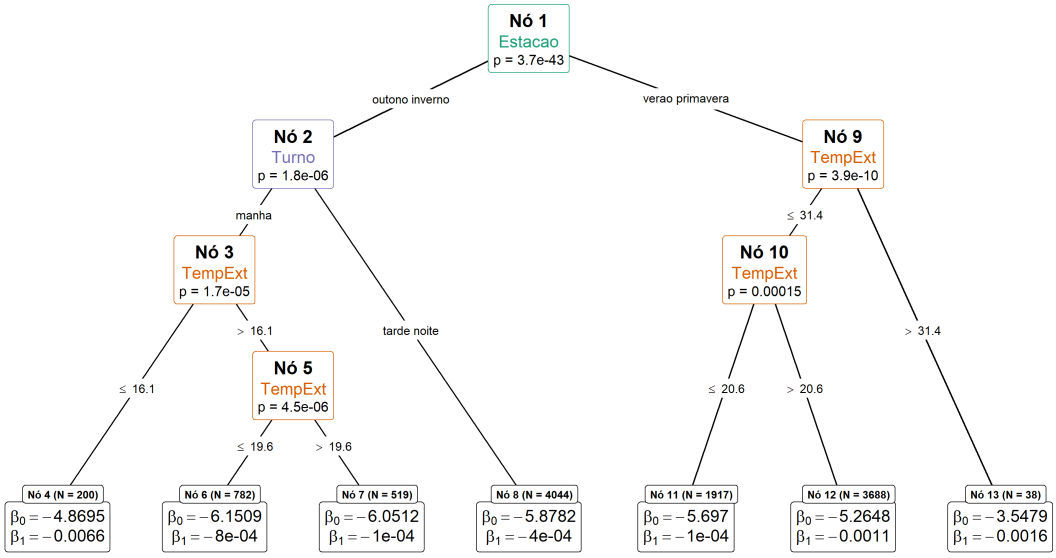

Figura 4.10: Árvore do modelo MOB 1 para a base ETAPA 1.
Fonte: Figura gerada pelo próprio autor.

Pela Figura 4.10, percebe-se que a covariável Estação foi a primeira a ser utilizada para particionar. Em seguida, o Turno e Temperatura Externa foram utilizadas, com destaca para esta aparecendo em 4 nós particionadores.

MOB 2

O modelo MOB 2 consistiu no mesmo do MOB 1, com a diferença que o nível de significância será fixado em $\alpha = 0.05$. Nota-se pela Figura 4.11 que a árvore cresceu bastante. Isso se dá devido ao nível de significância escolhido em combinação com o grande número de observações do banco de dados.

CART

Para o modelo utilizando a árvore CART©, o mesmo consegue incorporar o total de aves no caminha como um *offset*, modificando assim a Soma de Quadrados para realizar o particionamento binário. Tal processo está melhor explicado na seção sobre regressão Poisson da vinheita *An Introduction to Recursive Partitioning Using the RPART Routines*, presente na documentação da biblioteca *RPART* (Therneau e Atkinson, 1997).

Após a execução do CART©, podou-se a árvore criada com o valor de custo-complexidade

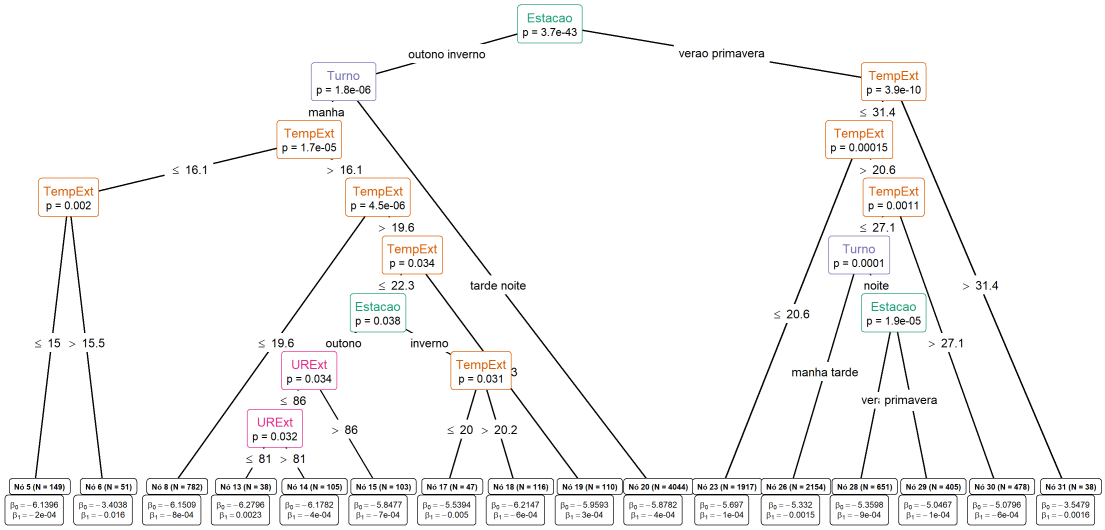

Figura 4.11: Árvore do modelo MOB 2 para a base ETAPA 1.
Fonte: Figura gerada pelo próprio autor.

sendo 0.026, obtendo a árvore representada na Figura 4.12. Por tal figura, percebe-se que o Tempo de Espera foi a covariável mais presente na árvore. Ademais, nota-se que o Nó 8 apresenta apenas 18 observações, com média e desvio padrão do número de aves mortas altíssimas, indicando que este nó incorporou algumas observações extremas.

CTREE 1

Na árvore criada pelo CTREE, não há como incorporar o Total de Aves por caminhão, então essa covariável entrará no modelo como uma das possíveis covariáveis particionadoras. O nível de significância escolhido foi de $\alpha = 0.0001$. Após a execução do algoritmo obteve a Figura 4.13. Nota-se que a árvore ficou extremamente pequena, com apenas 3 nós particionadores e, diferente da árvore do CART©, o Tempo de Espera não apareceu.

CTREE 2

Foi ajustado um outro modelo CTREE, chamado de CTREE 2. Esse modelo é o mesmo do CTREE 1 (mesmas covariáveis e nível de significância), mas com uma modificação na variável resposta. Em vez de usar o número de mortes, usou-se

$$Y_i^* = \log(1 + NumMorte),$$

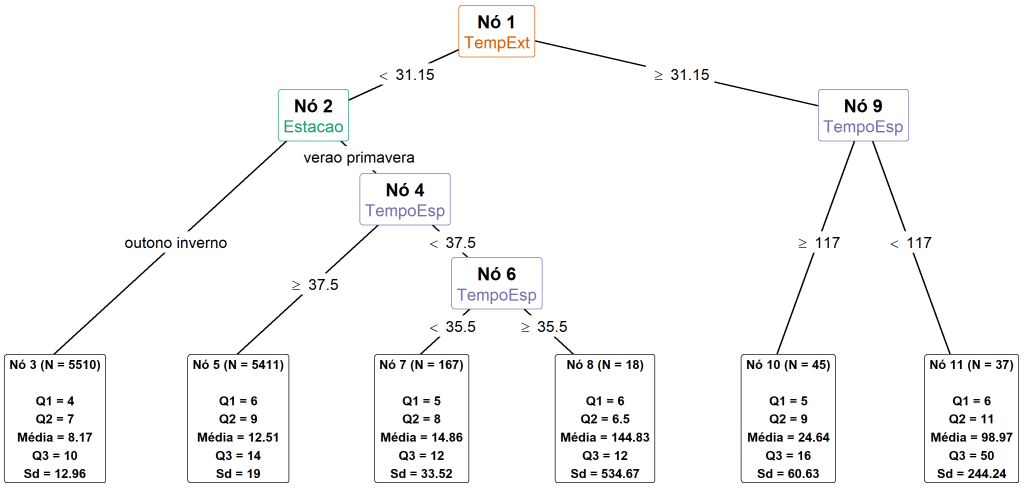

Figura 4.12: Árvore do modelo CART para a base ETAPA 1 (com medidas descritivas do número de aves mortas).
Fonte: Figura gerada pelo próprio autor.

e para predizer o número de mortes, se fez

$$\hat{Y}_i = \exp(\hat{Y}_i^*) - 1.$$

Após a execução do algoritmo, o mesmo resultado em uma árvore extremamente grande.

Floresta aleatória 1

Em relação a floresta aleatória 1, essa foi executada com uma otimização nos hiperparâmetros número de árvores e número mínimo de observações por nó. Tal otimização se deu por otimização Bayesiana (Yan, 2021), utilizando 30 iteração e 5 chutes aleatórios e validação cruzada com 5 *folds*. O intervalo para o número de árvore e o número de observações mínimo por nó foi de 100 a 1000 e de 5 a 30 respectivamente.

Depois da otimização, obteve-se 100 como sendo o número de árvores e 9 o número de observações mínimas por nó. Estudando a Figura 4.14A, percebe-se que Tempo de Espera e Temperatura Externa foram as covariáveis mais presentes nas árvores. Os outros hiperparâmetros ficaram como padrões do pacote *ranger* do R.

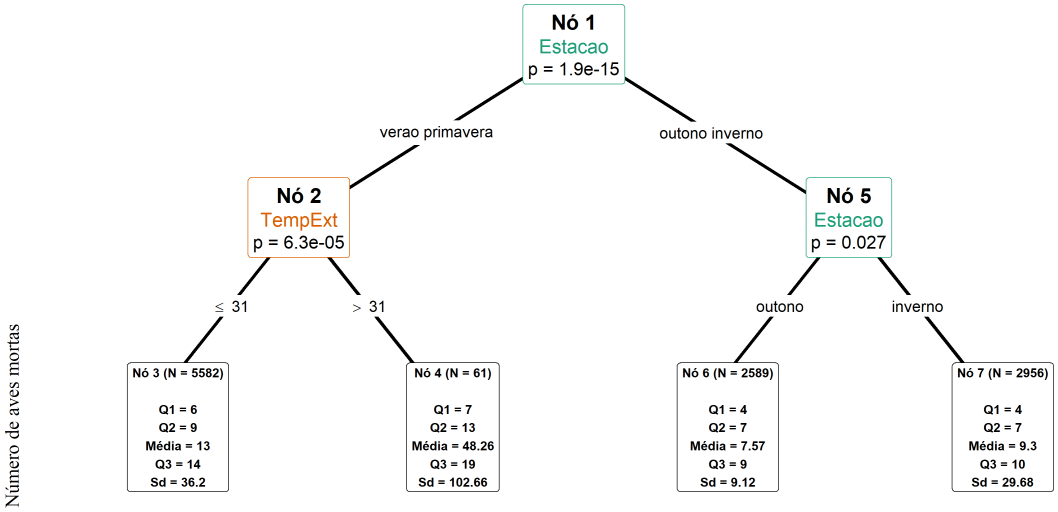

Figura 4.13: Árvore do modelo CTREE 1 para a base ETAPA 1.
Fonte: Figura gerada pelo próprio autor.

Floresta aleatória 2

Em relação a floresta aleatória 2, assim como no CTREE 2, uma modificação na variável resposta fora aplicada. Em vez de usar o número de mortes, usou-se

$$Y_i^* = \log(1 + NumMorte),$$

e para predizer o número de mortes, se fez

$$\hat{Y}_i = \exp(\hat{Y}_i^*) - 1.$$

Também fora executada com uma otimização nos hiperparâmetros número de árvores e número mínimo de observações por nó. Tal otimização se deu por otimização Bayesiana (Yan, 2021), utilizando 30 iteração e 5 chutes aleatórios e validação cruzada com 5 *fold*. O intervalo para o número de árvore e o número de observações mínimo por nó foi de 100 a 1000 e de 5 a 30 respectivamente.

Depois da otimização, obteve-se 100 como sendo o número de árvores e 5 o número de observações mínimas por nó, diferente do número 9 para a floresta 1. Estudando a Figura 4.14B, percebe-se o mesmo comportamento para floresta aleatória 1, em que Tempo de Espera e Tempera Externa foram as covariáveis mais presentes.

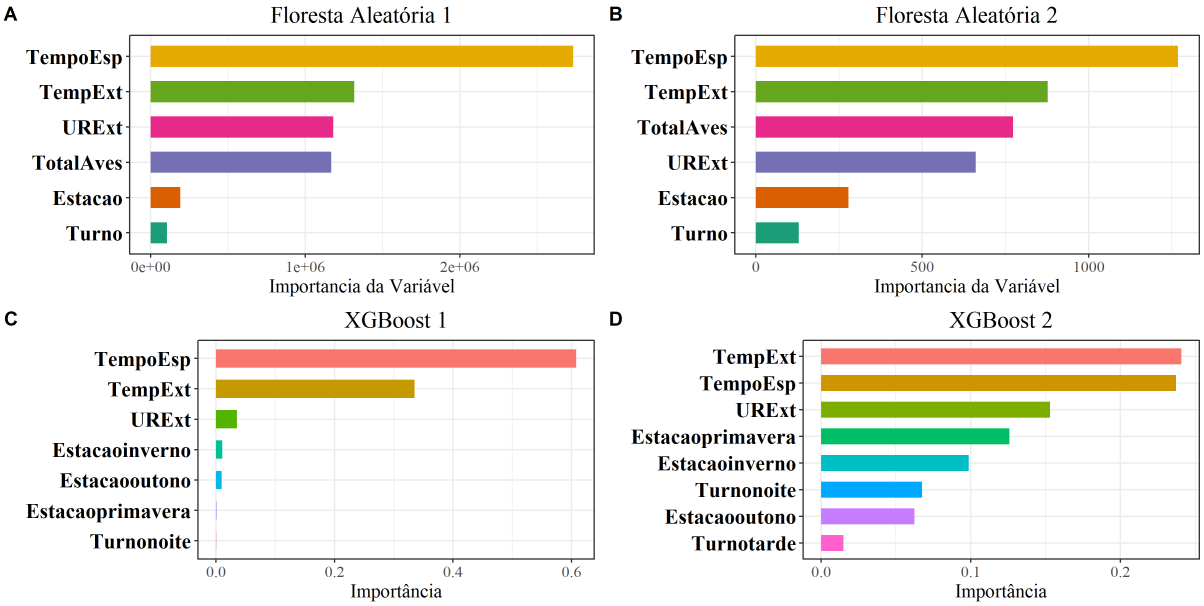

Figura 4.14: Importância das covariáveis na Floresta aleatória 1 e 2 e no XGBoost 1 e 2 para a base ETAPA 1.

Fonte: Figura gerada pelo próprio autor.

XGBoost 1

Para o XGBoost 1, o único hiperparâmetro otimizado foi o número de árvores (iterações). Para isso, considerou-se uma validação cruzada (no conjunto de treinamento) de *10 folds*, um total de árvore sendo 1000 e um parada breve de 100. Portanto, deseja-se encontrar o número de árvore que obtém o menor erro nos conjuntos de validação.

Após a execução da validação cruzada, obteve-se que a melhor iteração (número de árvores) foi 16. Por conseguinte, esse número será o número total de árvores do XGBoost. Analisando a Figura 4.14C, observa-se que, tal como para Floresta Aleatória 1 e 2, Temperatura Externa e Tempo de Espera são as covariáveis mais presentes.

XGBoost 2

Em relação ao XGBoost 2, assim como no CTREE 2 e Floresta Aleatória 2, uma modificação na variável resposta fora aplicada. Em vez de usar o número de mortes, usou-se

$$Y_i^* = \log(1 + NumMorte),$$

e para predizer o número de mortes, se fez

$$\hat{Y}_i = \exp(\hat{Y}_i^*) - 1.$$

Para o XGBoost 2, o único hiperparâmetro otimizado foi o número de árvores (iterações). Para isso, considerou-se uma validação cruzada (no conjunto de treinamento) de *10 folds*, um total de árvore sendo 1000 e um parada breve de 100. Portanto, deseja-se encontrar o número de árvore que obtém o menor erro nos conjuntos de validação.

Após a execução da validação cruzada, obteve-se que a melhor iteração (número de árvores) foi 23. Por conseguinte, esse número será o número total de árvores do XGBoost. Analisando a Figura 4.14D, observa-se que, tal como para Floresta Aleatória 1 e 2 e no XGBoost 1, Temperatura Externa e Tempo de Espera são as covariáveis mais presentes. Contudo, nota-se que essas variáveis não estão tão acentuadas em relação às demais como estavam no modelo XGBoost 1 por exemplo.

GLM 1

Para o GLM 1, ajustou-se um modelo da seguinte forma

$$NumMort = TempoEsp + TempExt + URExp + Estacao + Turno$$

considerando o log do Total de Aves como offset (parte conhecida). A distribuição utilizada fora Poisson com função de ligação log. Analisando a Tabela 4.19, percebe-se que todas as covariáveis foram significativas, com exceção do nível Turno Tarde.

Tabela 4.8: Coeficientes estimados e significância para o modelo GLM 1.

Coeficiente	Estimativa	Erro padrão	valor-z	Pr(>|z|)
(Intercept)	-7.0323	0.0487	-144.3538	< 0.0001
TempoEsp	-0.0008	0.0001	-26.4248	< 0.0001
TempExt	0.0608	0.0013	45.0745	< 0.0001
URExt	0.0025	0.0003	8.7096	< 0.0001
Estacaooutono	-0.4008	0.0102	-39.1367	< 0.0001
Estacaoinverno	-0.2260	0.0113	-20.0201	< 0.0001
Estacaoprimavera	-0.0437	0.0074	-5.8766	< 0.0001
Turnotarde	0.0012	0.0099	0.1222	0.9027
Turnonoite	0.2675	0.0076	35.1630	< 0.0001

GLM 2

Para o GLM 2, ajustou-se um modelo da mesma forma que o GLM 1, mas agora com iteração entre Estacao e Turno, isto é

$$NumMort = TempoEsp + TempExt + URExp + Estacao + Turno + Estacao \cdot Turno,$$

Analisando a Tabela 4.9, percebe-se que todas as covariáveis foram significativas, com exceção do nível Turno Tarde, tal como ocorreu no modelo GLM 1.

Tabela 4.9: Coeficientes estimados e significância para o modelo GLM 2.

Coeficiente	Estimativa	Erro padrão	valor-z	Pr(>|z|)
(Intercept)	-7.0265	0.0492	-142.9240	< 0.0001
TempoEsp	-0.0008	0.0001	-24.4245	< 0.0001
TempExt	0.0666	0.0014	48.7491	< 0.0001
UExt	0.0017	0.0003	5.8989	< 0.0001
Turnotarde	0.0170	0.0142	1.1968	0.2314
Turnonoite	0.0547	0.0129	4.2415	< 0.0001
Estacaooutono	-0.4366	0.0176	-24.7769	< 0.0001
Estacaoinverno	-0.3089	0.0179	-17.2235	< 0.0001
Estacaoprimavera	-0.2138	0.0148	-14.4655	< 0.0001
Turnotarde:Estacaooutono	-0.1872	0.0235	-7.9696	< 0.0001
Turnonoite:Estacaooutono	0.2647	0.0217	12.2083	< 0.0001
Turnotarde:Estacaoinverno	-0.1763	0.0219	-8.0455	< 0.0001
Turnonoite:Estacaoinverno	0.3292	0.0200	16.4898	< 0.0001
Turnotarde:Estacaoprimavera	0.0503	0.0199	2.5317	0.0114
Turnonoite:Estacaoprimavera	0.3638	0.0184	19.7802	< 0.0001

4.2.3 Desfecho

Percebe-se meio da Tabela 4.10, nota-se que os modelos MOB 1 e MOB 2, Floresta 1 e XGBoost 1 foram aqueles com os maiores erros. Por outro lado, ao fazer uma transformação na variável resposta, o desempenho da Floresta 2 e XGBoost 2 retrataram erros bem menores. Além disso, os modelos GLM 1 e 2, CTREE 1 e 2, e XGBoost 2 apresentaram erros intermediários. Por fim, os modelos CART e Floresta 2 apresentaram os menores erros.

Tabela 4.10: Erros quadráticos e absolutos médios com seus respectivos intervalos calculados em relação a amostra de teste para cada um dos modelos ajustados (ETAPA 1).

Modelo	EQM	LI do EQM	LS do EQM	EAM	LI do EAM	LS do EAM
MOB 1	680.01	0.00	1369.84	6.88	5.94	7.82
MOB 2	680.30	0.00	1370.21	6.93	5.99	7.87
CART	628.19	29.50	1226.88	6.77	5.86	7.67
CTREE 1	673.57	0.00	1364.27	6.77	5.84	7.71
CTREE 2	676.86	0.00	1373.64	5.87	4.92	6.82
floresta 1	843.41	102.01	1584.80	6.94	5.89	7.99
floresta 2	658.75	0.00	1349.96	5.77	4.83	6.70
XGBoost 1	741.52	42.14	1440.91	7.64	6.67	8.62
XGBoost 2	686.04	0.00	1382.29	5.93	4.98	6.89
GLM 1	662.05	0.00	1343.17	6.86	5.94	7.79
GLM 2	662.51	0.00	1345.64	6.87	5.94	7.79

4.3 Base de dados da Etapa 2

A base de dados da Etapa 2 consiste no mesmo contexto da base da Etapa 1, mas agora foram coletadas mais variáveis e o número de observações diminuiu, totalizando 218 observações (Vieira *et al.*, 2015). Consequentemente, será considerado um estudo mais inferencial e interpretativo para a base Etapa 2. Essa base de dados consiste nas seguintes variáveis (Tabela 4.11).

No momento da descarga, aves de determinados caminhões podem ter morrido (seja ao longo da viagem ou momento de espera), sendo essa a variável de interesse ($\boldsymbol{Y}$). Para complementar, cada caminhão também teve o total de aves contabilizado.

4.3.1 Análise

Análise descritiva

Percebe-se pela Tabela 4.12 que o número de aves mortas é levemente heterogêneo, um pouco menos do que o número de mortes na ETAPA 1 (Tabela 4.4), uma vez que o coeficiente de variação da ETAPA 1 foi de 2.779 e da ETAPA 2 foi de 0.827. Além disso, nota-se que, diferente da ETAPA 1, não houve nenhuma caminhão com 0 mortes. Pode-se ter uma noção do número de aves mortas de acordo com o total de aves pela Tabela 4.13.

Tal como na ETAPA 1, será trabalhado com uma transformação do número de mortes

Tabela 4.11: Descrição da base de dados sobre a Etapa 2 de frangos.

Nome	**Sigla**	**Descrição**	**Tipo**
Número de Mortes	*NumMort*	Número de avez mortas por caminhão	Quantitativa discreta
Tempo de Espera	*TempEsp*	Tempo de Espera do caminhão no momento de descarga (em minutos))	Quantitativa contínua
Temperatura Externa	*TExt*	Temperatura Externa da granja (em °C)	Quantitativa contínua
Umidade Externa	*URExt*	Umidade externa da granja (em %)	Quantitativa contínua
Temperatura Interna	*TInt*	Temperatura Interna da granja (em °C)	Quantitativa contínua
Umidade Interna	*URInt*	Umidade Interna da granja (em %)	Quantitativa contínua
Tempo Total	*Tempototal*	Quanto tempo levou a descarga dos frangos (em min)	Quantitativa contínua
Diferença de Temperatura Retal	*Dif_TR*	Diferença absoluta da Temperatura Retal antes e depois (em °C)	Quantitativa contínua
Distância	*Distancia*	Distância da fazenda até o abatedouro (em km)	Quantitativa contínua
Turno	Turno	*Turno do transporte*	Qualitativa nominal (dia, tarde e noite)
Total de aves	*TotalAves*	Total de aves em cada caminhão	quantitativa discreta

Fonte: Tabela gerada pelo próprio autor.

Tabela 4.12: Medidas descritivas da variável número de mortes por caminhão (ETAPA 2).

Min	Q1	Med	Media	Q3	Max	Dp	CV
2	9.25	14	17.491	20	102	14.473	0.827

Fonte: Tabela gerada pelo próprio autor.

Tabela 4.13: Medidas descritivas da variável número total de aves por caminhão (ETAPA 2).

Min	Q1	Med	Media	Q3	Max	Dp	CV
1626	3024	3402	3446.505	3774	6891	625.068	0.181

Fonte: Tabela gerada pelo próprio autor.

na descritiva. Tal transformação será a logito (log de chance) e se dará como

$$Logito\_Morte = \log\left(\frac{\frac{NumMort}{TotalAves}}{1-\frac{NumMort}{TotalAves}}\right).$$

Observando a Figura 4.15, percebe-se um comportamento assimétrico a direita nas variáveis Tempo Total, Temperatura Externa e Diferença de Temperatura Retal. Por outro lado, as variáveis Umidade Externa e Umidade Interna apresentam assimetrias

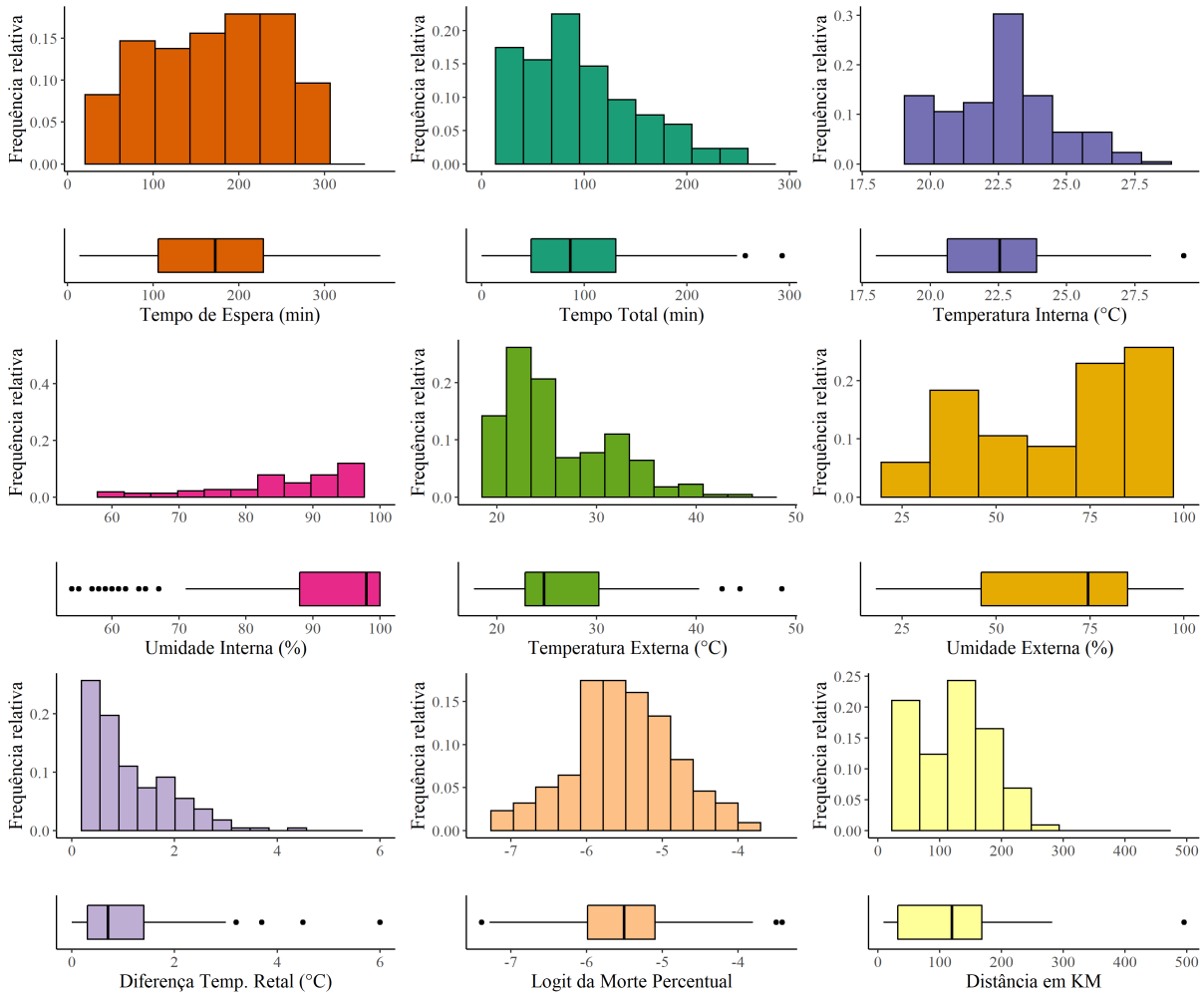

Figura 4.15: Histograma e *box-plots* das variáveis quantitativas (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

à esquerda. As outras variáveis retratam bimodalidade (Distância), em quanto outras estão concentradas em seus valores intermediários (Tempo de Espera e Logit da Morte Percentual).

Pelas Tabela 4.14, nota-se que, diferente da distribuição de Turnos da ETAPA 1, a maioria dos caminhões fazem suas viagens de tarde e de noite.

Tabela 4.14: Frequência percentual dos turnos (ETAPA 2).

manha	tarde	noite
0.280	0.349	0.372

Fonte: Tabela gerada pelo próprio autor.

Pela Figura 4.16, evidencia-se que apenas a variável temperatura e umidade externas estão correlacionadas negativamente, comportamento semelhante àquele da ETAPA 1. Além disso, as duplas Tempo de Espera, Tempo Total e Temperatura Externa, Temperatura Interna apresentam uma correlação positiva razoável. Por outro lado, todas as

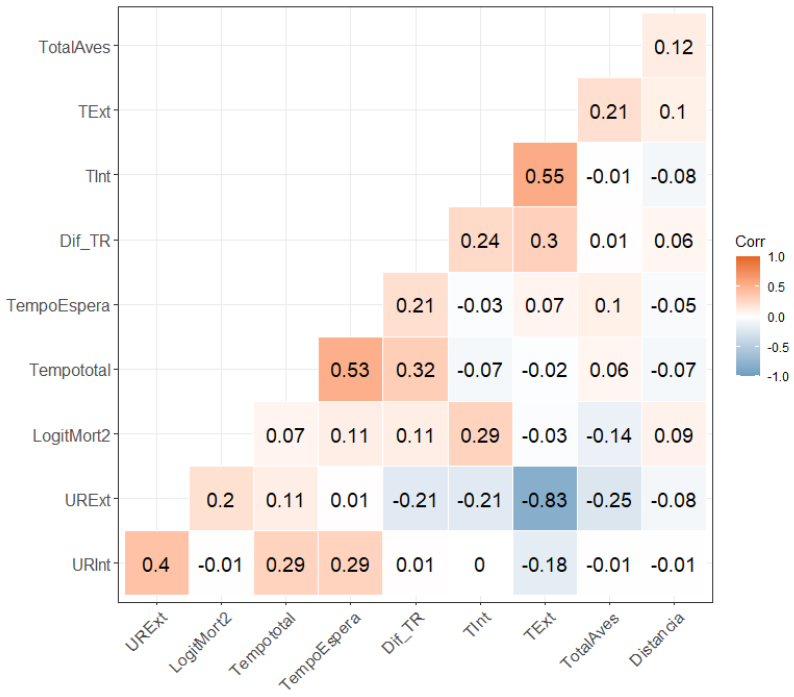

Figura 4.16: Correlação de *Spearman* para as variáveis quantitativas (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

outras covariáveis não apresentaram correlação relevante 2 a 2.

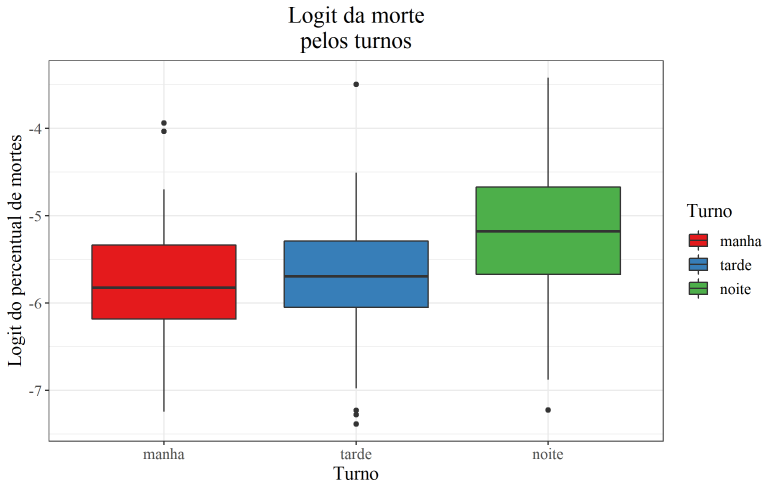

Figura 4.17: *Boxplots* do logito do percentual de mortes por Turnos (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

Na Figura 4.17, percebe-se que a variabilidade do turno da noite é maior do que a variabilidade nos outros turno. Além disso, os caminhões que viajaram à noite apresentaram o logito da morte maior do que os caminhões que viajaram nos outros turnos.

4.3.2 Modelagens na base ETAPA 2

Continuando com a base de dados da ETAPA2, considerou-se a base toda para ajustar os modelos seguintes.

MOB 1

Tem-se que a variável resposta será $\boldsymbol{Y}$ número de mortes e a covariável do modelo paramétrico será apenas X_1 tempo de espera. Todas as outra covariáveis (Turno, Tempo Total, Temperatura Externa e Interna, Umidade Externa e Interna, Distância e Diferença Retal) ficarão como covariáveis particionadoras. Assim como dito anteriormente, a escolha de covariáveis da parte paramétrica e da parte particionadora fica a cargo do pesquisador. Para esse estudo, considerou-se colocar na parte paramétrica covariáveis que são controláveis por parte do abatedouro (pode-se melhorar a logística do local para diminuir o tempo de espera) e colocar covariáveis naturais (não dependem do fator humano) na parte particionadora.

Tal como nos modelos MOBs anteriores, pressupôs-se que

$$Y_i \sim Poisson(\mu_i),$$

por conseguinte, pode-se escrever a média da distribuição como $\mu_i = \lambda_i t_i$, de forma que

- λ_i denota a proporção de aves mortas do i-ésimo caminhão;
- t_i detona o total de aves do i-ésimo caminhão;

Portanto, considerando a função de ligação log, tem-se que

$$\begin{aligned} \log(\mu_i) &= \log(\lambda_i t_i) \\ &= \log(\lambda_i) + \log(t_i) \\ &= \beta_0 + \beta_1 X_{1,i} + \log(t_i), \end{aligned}$$

em que $\beta_0 + \beta_1 X_1$ é a parte desconhecida do modelo e $\log(t_i)$ é parte conhecida (chamada de *offset*). Para este modelo (MOB 1), optou-se por fixar o nível de significância como $\alpha = 0.05$.

Observando a Figura 4.18, percebe-se que apenas um nó particionador foi utilizando, sendo o nó 1 com a covariável Turno. Dois novos nós foram criados, um com manhã e

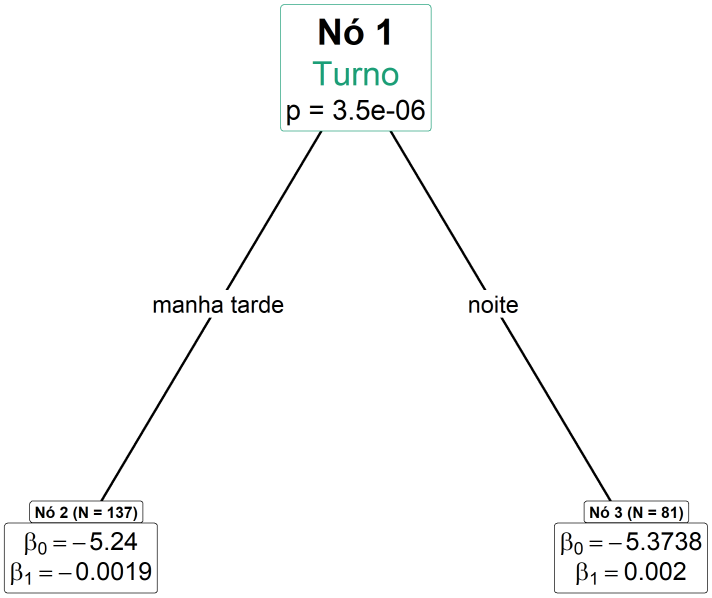

Figura 4.18: Árvore do modelo MOB 1 (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

tarde e outro com noite. Pela Tabela 4.15, percebe-se que ambos os coeficientes dos nós 2 e 3 foram significativos ao nível de 5% de significância.

Tabela 4.15: Coeficientes estimados e significância para os modelos paramétricos do MOB 1 (ETAPA 2).

	Coeficiente	Estimativa	Erro padrão	valor-z	Pr(>|z|)
Nó 2	(Intercept)	-5.2400	0.0482	-108.75	< 0.0001
	TempoEspera	-0.0019	0.0003	-6.52	< 0.0001
Nó 3	(Intercept)	-5.3738	0.0828	-64.92	< 0.0001
	TempoEspera	0.0020	0.0004	5.35	< 0.0001

Observando a Tabela 4.16, percebe-se que o nó 3 possui valores de AIC e BIC menores do que os valores do nó 2.

Tabela 4.16: AICs e BICs dos modelos paramétricos do MOB 1 (ETAPA 2)

	AIC	BIC
Nó 2	1456.082	1461.921
Nó 3	1211.408	1216.197

Na Figura 4.19, tem-se os valores da covariável paramétrica Tempo de espera e da variável resposta Número de aves mortas em relação aos dados observados. É possível perceber que o nó 2 possui os pontos mais próximos um dos outros, caracterizando uma menor variabilidade. Já o nó 3 possui uma maior dispersão dos pontos, caracterizando uma maior variabilidade do número de aves mortas.

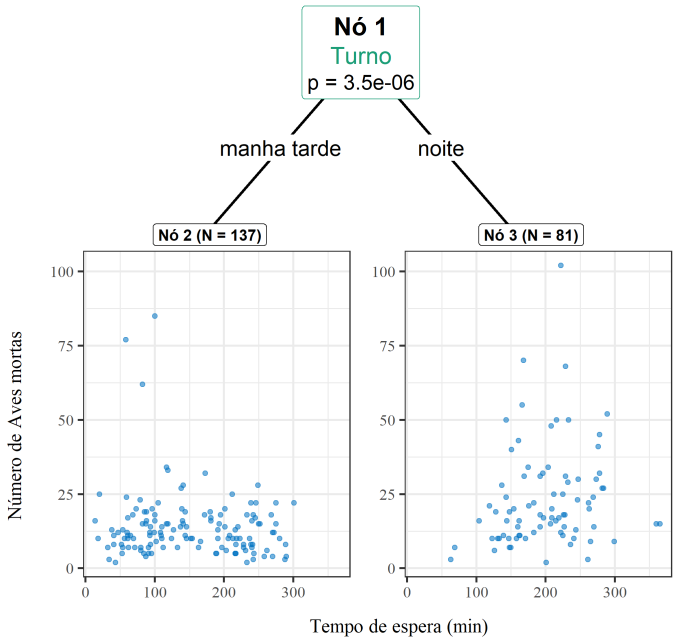

Figura 4.19: Árvore do modelo MOB 1 com dados observados (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

Por meio da Figura 4.20, nota-se que os resíduos estão dispersos aleatoriamente em torno do 0. Com destaque que para o nó 2, em que os pontos estão mais concentrados, já para o nó 3, os pontos estão mais um poucos mais dispersos. Mesma característica de variabilidade comentada na Figura 4.19.

Finalmente, para a Figura 4.21, tem-se as curvas obtidas a partir dos modelos paramétricos da árvore MOB 1 para os nós 2 e 3. Consequentemente, o modelo foi escrito como

$$\hat{Y}_i, k = \exp(\beta_{0,k} + \beta_{1,k} TempoEspera_{i,k}) \cdot TotalAves_{i,k} \; ;$$

no qual k é índice referente ao nó 1 ou 2. Para a representação da Figura 4.21, fixou-se $TotalAves_{i,k}$ como a média do total de aves de todo o banco de dados.

A Figura 4.21 mostra que, para o nó 2, conforme o tempo de espera aumenta, o número de aves mortas tendem a cair. Por outro lado, para o nó 3, o comportamento é o oposto, conforme o tempo ode espera aumenta, o número de aves mortas aumenta também.

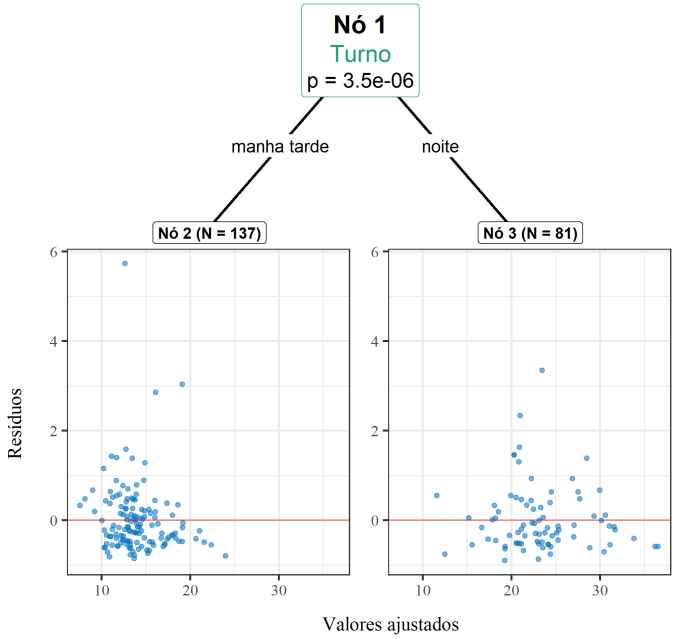

Figura 4.20: Árvore do modelo MOB 1 com resíduos (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

MOB 2

Utilizou-se o mesmo nível de significância do MOB 1 da ETAPA 1, fixando-se em $\alpha = 0.05$. Além disso, a estrutura de *offset* foi a mesma com uso do total de aves por caminhão. Mas agora optou-se por colocar a Temperatura Interna e Umidade Interna na parte paramétrica, uma vez que tais covariáveis são parcialmente controláveis pela granja.

A Figura 4.22 retrata a árvore criada no modelo MOB 2. Nota-se que o nó particionador foi Tempera Externa, diferente do MOB 1 o qual utilizou o Turno. Além disso, com o uso da Tabela 4.17, percebe-se que quase todos os coeficientes foram significativos para ambos os nós, com exceção do coeficiente Umidade Interna para a no 2.

Tabela 4.17: Coeficientes estimados e significância para os modelos paramétricos do MOB 2 (ETAPA 2).

	Coeficiente	Estimativa	Erro padrão	valor-z	Pr(>|z|)
Nó 2	(Intercept)	-7.7818	0.2883	-26.99	< 0.0001
	TempoEspera	0.0030	0.0003	11.45	< 0.0001
	TInt	0.1173	0.0089	13.18	< 0.0001
	URInt	-0.0058	0.0023	-2.48	0.0132
Nó 3	(Intercept)	-8.2990	0.6455	-12.86	< 0.0001
	TempoEspera	-0.0072	0.0006	-12.52	< 0.0001
	TInt	0.0805	0.0218	3.68	0.0002
	URInt	0.0247	0.0034	7.34	< 0.0001

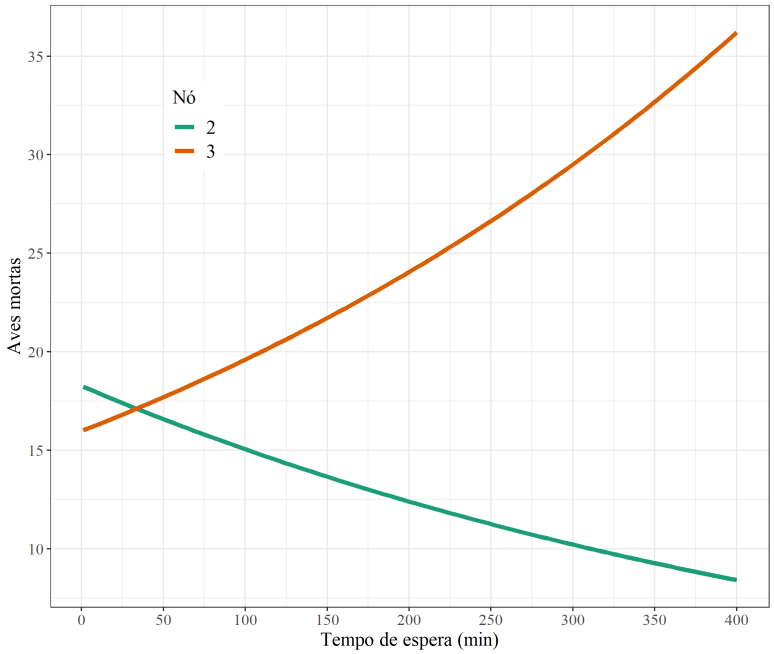

Figura 4.21: Curvas paramétricas da árvore do modelo MOB 1 (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

Observando a Tabela 4.18, tem-se que o nó 3 apresenta valores de AIC e BIC menores quando comparado ao nó 2, caracterizando um melhor ajusto do nó 3 do que o nó 2.

Tabela 4.18: AICs e BICs dos modelos paramétricos do MOB 2 (ETAPA 2)

	AIC	BIC
Nó 2	1619.732	1631.639
Nó 3	866.4519	875.6138

Na Figura 4.23, tem-se as covariáveis da parte paramétrica contra os valores da variável resposta número de aves mortas. Nota-se que, no nó 2, a Temperatura Interna fica mais concentrada nos valores de 18 °C a 23 °C. Já no nó 3, as Temperatura Interna está acima de 22.5 °C aproximadamente. Portanto, o nó 3 é um nó em os a temperatura interna do galpão de descarga é mais quente.

Ainda na Figura 4.23, percebe-se que para o nó 2, há uma concentração de Umidade Interna em 90% e 100%, sendo o nó 2 caracterizado como um nó de granjas mais úmidos internamente. Já o nó 3 apresenta a umidade interna mais espalhada ao longo do intervalo observado.

Finalmente, analisa-se os resíduos contra preditos dos modelos paramétricos do MOB 2 (Figura 4.24), observando que os gráficos de resíduos apresentam um leve comportamento de funil triangular, em que a variabilidade dos resíduos diminui ao se considerar valores

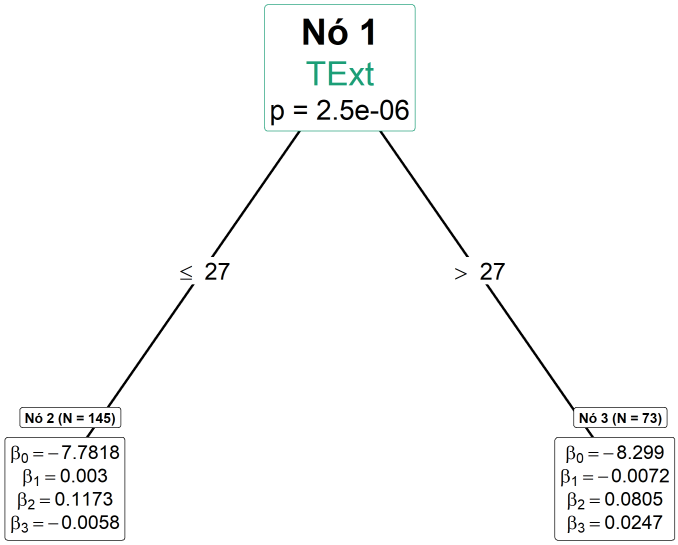

Figura 4.22: Árvore do modelo MOB 2 (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

ajustados maiores.

CART

Para o modelo utilizando a árvore CART©, o mesmo consegue incorporar o total de aves no caminhão como um *offset* (assim como fora feito nos modelos da base ETAPA 1), modificando assim a Soma de Quadrados para realizar o particionamento binário. Após a execução do CART©, podou-se a árvore criada com o valor de custo-complexidade sendo 0.02492, obtendo a árvore representada na Figura 4.25.

Estudando a Figura 4.25, percebe-se que a covariável Turno e Temperatura Externa foram utilizadas para particionar a árvore, assim como essas mesma covariáveis foram utilizadas no particionamento do MOB 1 e 2 respectivamente (na ETAPA 2). Além disso, tem-se que o Tempo de Espera fora utilizado, criando os nós 5 e 6, tal como o Tempo Total, criando os nós 8 e 9.

Ainda na Figura 4.25, percebe-se que os nós 3 e 5 são nós com variabilidade pequenas. Já o nó 8 apresenta uma variabilidade intermediária. Por fim, os nós 6 e 9 apresentam as maiores variabilidades.

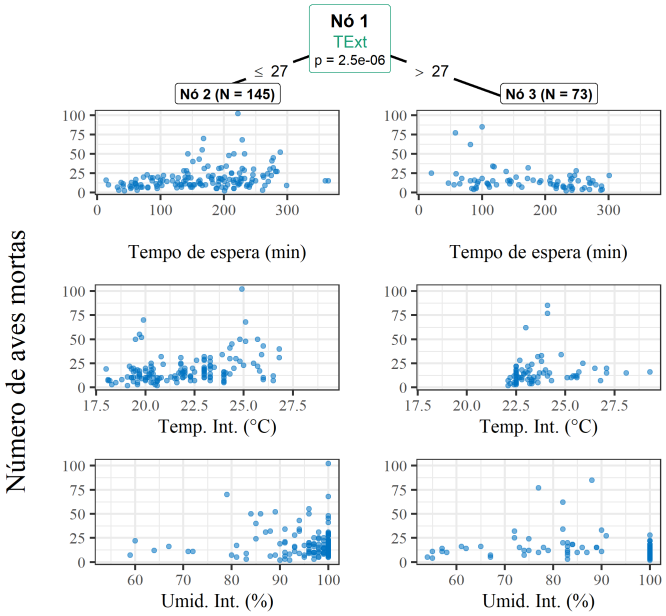

Figura 4.23: Árvore do modelo MOB 2 com dados observados (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

CTREE

Na árvore criada pelo CTREE, não há como incorporar o Total de Aves por caminhão, então essa covariável entrará no modelo como uma possível covariável particionadora (tal como nos modelos da base ETAPA 1). O nível de significância escolhido foi de $\alpha = 0.05$. Após a execução do algoritmo obteve-se a Figura 4.26.

Analisando a Figura 4.26, percebe-se que a única covariável particionadora foi Turno, dividindo em um nó com manhã e tarde e outro nó com noite. Tal particionamento binário é o mesmo usado no MOB 1 da ETAPA 2.

GLM 1

O último modelo da base de dados ETAPA 2 foi um GLM. Considerou-se a distribuição Poisson com função de ligação log e *offset* como sendo o total de aves. Após o ajuste do modelo, obteve-se a Tabela 4.19, a qual mostra que os coeficientes Tempo de Espera, Turno da Tarde, Umidade Interna e Tempera Externa não foram significativos ao nível de 5% de significância.

Observando a Figura 4.27A, tem-se que os pontos estão espalhados aleatoriamente em torno da média 0 dos resíduos deviance. Contudo, há uma leve concentração de pontos em torno do valor 10 de preditos, algo para se atentar. Partindo para a Figura 4.27B, tem-se

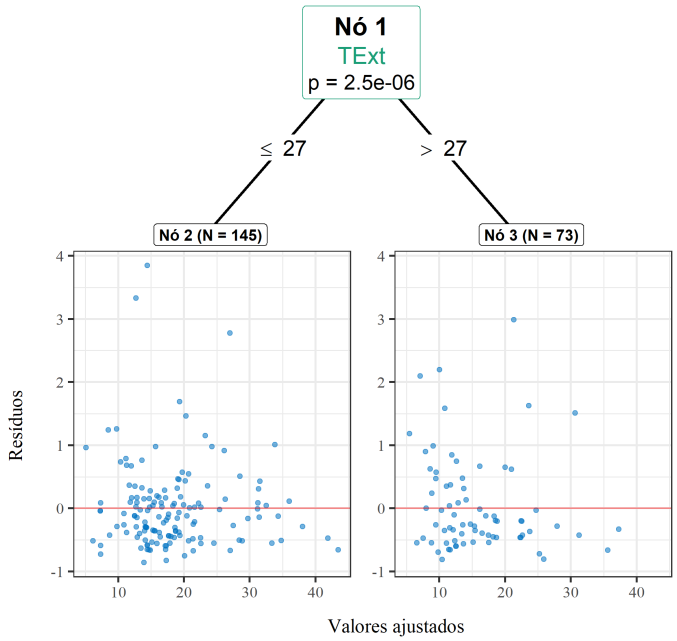

Figura 4.24: Árvore do modelo MOB 2 com resíduos (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

Tabela 4.19: Coeficientes estimados e significância para o modelo GLM 1 (ETAPA 2).

Coeficiente	Estimativa	Erro padrão	valor-z	Pr(>|z|)
(Intercept)	-7.5030	0.3234	-23.2010	< 0.0001
TempoEspera	-0.0003	0.0003	-1.1122	0.2661
Turnotarde	-0.0423	0.0660	-0.6408	0.5217
Turnonoite	0.4195	0.0611	6.8637	< 0.0001
Tempototal	0.0010	0.0004	2.7456	0.0060
TExt	0.0167	0.0087	1.9286	0.0538
URExt	0.0076	0.0023	3.3597	0.0008
TInt	0.0510	0.0129	3.9569	0.0001
URInt	-0.0040	0.0027	-1.4994	0.1338
Dif_TR	0.1091	0.0181	6.0251	< 0.0001
Distancia	0.0011	0.0002	6.8757	< 0.0001

que conforme o índice aumenta (ordem de coleta), comportamento dos pontos aleatória dos pontos continua, exceto por alguns pontos valores extremos. Finalmente, na Figura 4.27C, tem-se o QQPlot dos resíduos quantílicos. Essa figura mostra alguns pontos saindo da banda nas extremidades, um possível indicativo que o modelo possa não ser o mais adequado.

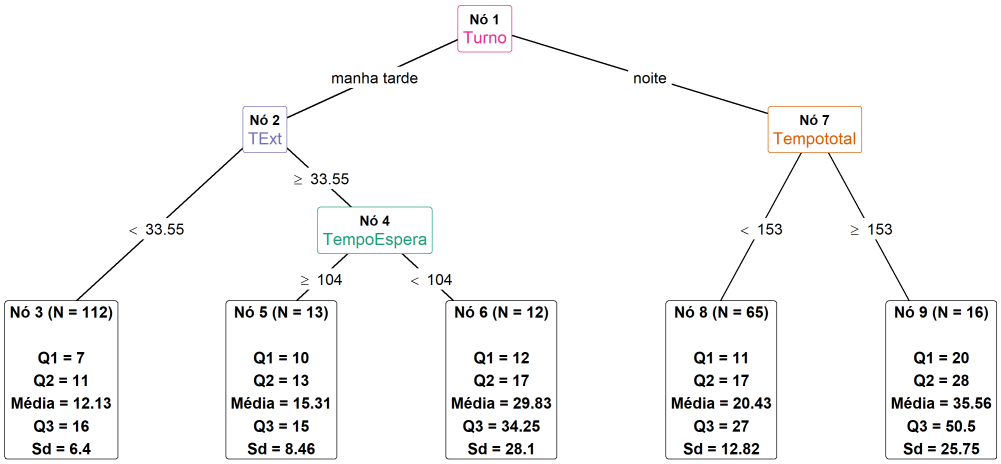

Figura 4.25: Árvore do modelo CART com medidas descritivas do número de aves mortas (ETAPA 2).

Fonte: Figura gerada pelo próprio autor.

4.3.3 Desfecho

O intuito de se utilizar a base de dados da ETAPA 2 foi verificar como o algoritmo MOB se comporta em uma situação com mais covariáveis mas com menos observações. Consequentemente, optou-se por uma abordagem mais interpretativa, fixando o nível de significância $\alpha = 0.05$ em todos os algoritmos que se utilizam testes de hipótese.

Após os ajustes dos modelos, verificou-se que a covariável Turno foi essencial para particionar as árvores (tanto no MOB 1 quanto no CTREE), uma vez que o nível noite causava uma diferença tanto na média quanto na variabilidade do número de mortes quando, fato esse identificado na análise descritiva.

Por outro lado, algo interessante de se verificar foi que ao adicionar mais covariáveis na parte paramétrica do modelo MOB, o tamanho da árvore se manteve o mesmo, mas a covariável particionadora mudou de Turno para Temperatura Externa, isto é, MOB 1 e MOB 2.

O algoritmo CART© resultou em uma árvore com mais alguns nós, mas ainda assim com os nós contendo Turno e Tempera Externa (covariáveis particionadoras para o MOB 1 e 2). No GLM, a covariável Tempo de Espera não fora significativa, já no MOB 1 e 2, o Tempo de Espera foi significativo.

Além disso, pode-se calcular os valores preditos, e consequentemente, erros (valor observado menos predito), utilizando a mesma amostra de treino. Contudo, não será

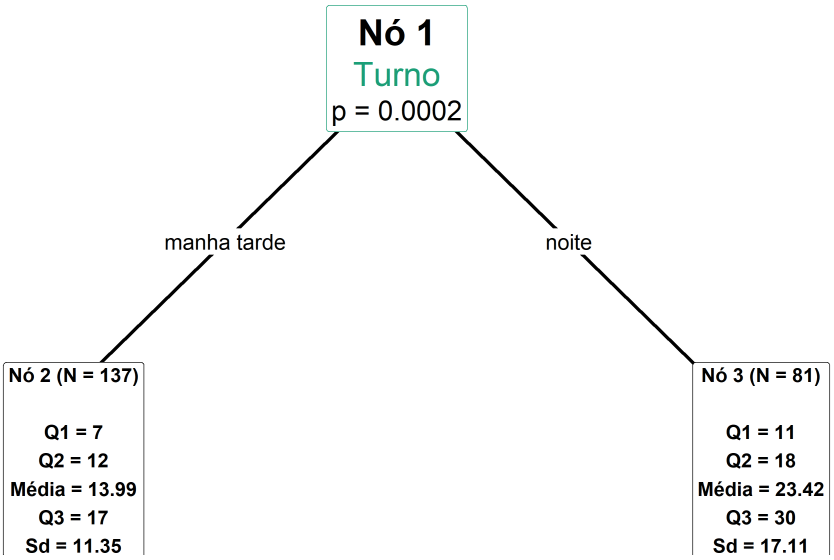

Figura 4.26: Árvore do modelo CTREE (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

possível dizer se o modelo está bem ou mal ajustado, uma vez que ao utilizar os próprios dados de treino como teste, os erros tenderão a ser pequenos por natureza. Logo, ao utilizar tais erros, será apenas possível dizer qual modelo está mais sobreajustado aos dados.

Tabela 4.20: Erros quadráticos e absolutos médios com seus respectivos intervalos calculados em relação a amostra de treino para cada um dos modelos ajustados (ETAPA 2).

Modelo	EQM	LI do EQM	LS do EQM	EAM	LI do EAM	LS do EAM
MOB1	186.97	102.21	271.74	8.97	7.60	10.34
MOB2	170.20	95.60	244.81	8.44	7.12	9.76
CART	161.71	100.57	222.85	8.83	7.61	10.04
CTREE	187.73	101.49	273.97	8.89	7.51	10.28
GLM	174.37	96.91	251.83	8.79	7.48	10.10

Ao analisar a Tabela 4.20, percebe-se que os modelos CART© e MOB foram aqueles que apresentaram menores EQMs. Além disso, o GLM ficou com um erro intermediário e os modelos MOB 1 e CTREE apresentaram os maiores erros.

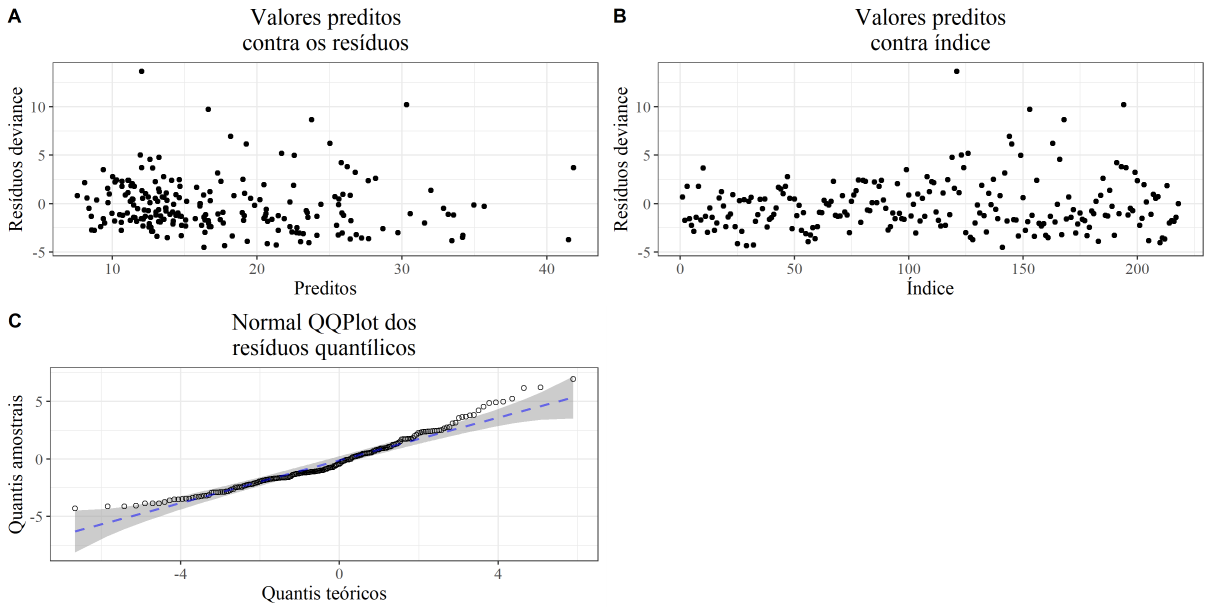

Figura 4.27: Análise dos resíduos deviance para o GLM 1 (ETAPA 2).
Fonte: Figura gerada pelo próprio autor.

Capítulo 5

Conclusão

Em situações controladas ou que se tenha alguma informação a priori da relação entre a variável resposta e covariáveis, o MOB possui um excelente desempenho preditivo e interpretativo, uma vez que é possível escolher sabiamente quais covariáveis farão parte do particionamento e qual farão parte do ajuste paramétrico. Assim como se verificou no exemplo simulado.

Por outro lado, em aspectos puramente preditivos, o MOB ficou relativamente atrás na corrida quando comparado a outros algoritmos preditivos. Isso se dá pois entram muitas questões no ajuste do modelo, tal como escolher quais serão as covariáveis particionadoras e quais farão parte do modelo paramétrico, escolher também quais distribuições testar e quais funções de ligação usar.

Ademais, tem-se que devido a natureza do próprio *Model-Based Recursive Partitioning* ajustar diversos modelos paramétricos, o uso de algoritmos de otimização e de validações cruzadas torna-se custoso computacionalmente, afetando o uso da técnica em grandes bases de dados (muitas observações e covariáveis).

Entretanto, para bases de dados pequenas ou médias, tal como a base de dados da ETAPA 2, tem-se que o MOB desempenhou muito bem no quesito interpretativo. Uma vez que as árvores foram pequenas e objetivas, além de possibilitar diversas visualizações, tais como as curvas paramétricas, observações e resíduos de cada nó.

Como sugestão para estudos porvindouros, aconselha-se explorar a questão de análise de resíduos do *Model-Based Recursive Partitioning*, algo que não é muito abordado pelos criadores e é uma área interessante de se explorar, consolidando o algoritmo MOB como um todo.

Agora se tratando especificamente do banco de dados da ETAPA 2, seria interessante

tratar da média e da variabilidade do número de mortes simultaneamente, utilizando o algoritmo MOB. Este trabalho empregou a distribuição Poisson nos modelos paramétricos, porém a Poisson exige que a média e variância da variável resposta sejam iguais. Para contornar esse problema, pode-se pensar em modelos de quase-verossimilhança e modelos duplos como parte paramétrica do *Model-Based Recursive Partitioning*.

Tanto modelos duplos quanto modelos de quase-verossimilhança são melhores elucidados na tese de Vieira (2008). Além disso, tais técnicas foram aplicadas na mesma base de dados da ETAPA 2.

Referências Bibliográficas

Breiman, L., Friedman, J., Olshen, R. e Stone, C. (1984). Classification and regression trees. wadsworth int. *Group*, **37**(15), 237–251.

Carvalho, A., Faceli, K., Lorena, A. e Gama, J. (2011). Inteligência artificial–uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*.

Chambers, J. M., Cleveland, W. S., Kleiner, B. e Tukey, P. A. (2018). *Graphical methods for data analysis*. Chapman and Hall/CRC.

Chan, K.-Y. e Loh, W.-Y. (2004). Lotus: An algorithm for building accurate and comprehensible logistic regression trees. *Journal of Computational and Graphical Statistics*, **13**(4), 826–852.

Chaubey, Y. (2012). Resampling-based multiple testing: Examples and methods for p-value adjustment. *Technometrics*, **35**, 450–451.

Chaudhuri, P. e Loh, W.-Y. (2002). Nonparametric estimation of conditional quantiles using quantile regression trees. *Bernoulli*, **8**(5), 561–576.

Chaudhuri, P., Lo, W.-D., Loh, W.-Y. e Yang, C.-C. (1995). Generalized regression trees. *Statistica Sinica*, **5**(2), 641–666.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y. e Li, Y. (2021). *xgboost: Extreme Gradient Boosting*. R package version 1.4.1.1.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, **28**(3), 591–605.

Cordeiro, G. M. e Demétrio, C. G. (2008). Modelos lineares generalizados e extensões. *Piracicaba: USP*.

Denison, D. G., Mallick, B. K. e Smith, A. F. (1998). A bayesian cart algorithm. *Biometrika*, **85**(2), 363–377.

Esposito, F., Malerba, D., Semeraro, G. e Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence*, **19**(5), 476–491.

Franklin, J. (1986). Aristotle on species variation. *Philosophy*, **61**(236), 245–252.

Gama, J. (2004). Functional trees. *Machine learning*, **55**(3), 219–250.

Gini, C. (1921). Measurement of inequality of incomes. *The economic journal*, **31**(121), 124–126.

Glur, C. (2020). *data.tree: General Purpose Hierarchical Data Structure*. R package version 1.0.0.

González, S., García, S., Del Ser, J., Rokach, L. e Herrera, F. (2020). A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. *Information Fusion*, **64**, 205–237.

Hansen, B. E. (1997). Approximate asymptotic p values for structuras-change tests. *Journal of Business & Economic Statistics*, **15**(1), 60–67.

Hjort, N. L. e Koning, A. (2002). Tests for constancy of model parameters over time. *Journal of Nonparametric Statistics*, **14**(1-2), 113–132.

Hornik, K., Buchta, C. e Zeileis, A. (2009). Open-source machine learning: R meets Weka. *Computational Statistics*, **24**(2), 225–232.

Hothorn, T. e Zeileis, A. (2015). partykit: A modular toolkit for recursive partytioning in R. *Journal of Machine Learning Research*, **16**, 3905–3909.

Hothorn, T., Hornik, K. e Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

Hssina, B., Merbouha, A., Ezzikouri, H. e Erritali, M. (2014). A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, **4**(2), 13–19.

Hunt, E. B., Marin, J. e Stone, P. J. (1966). *Experiments in Induction*. Academic Press. ISBN 9780123623508.

James, G., Witten, D., Hastie, T. e Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **29**(2), 119–127.

Kim, H. e Loh, W.-Y. (2001). Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, **96**(454), 589–604.

Loh, W.-Y. (2002). Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, **12**, 361–386.

Loh, W.-Y. (2014). Fifty years of classification and regression trees. *International Statistical Review*, **82**(3), 329–348.

Loh, W.-Y. e Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, **7**(4), 815–840.

Long, J. S. (1990). The origins of sex differences in science. *Social forces*, **68**(4), 1297–1316.

Marchese Robinson, R. L., Palczewska, A., Palczewski, J. e Kidley, N. (2017). Comparison of the predictive performance and interpretability of random forest and linear models on benchmark data sets. *Journal of chemical information and modeling*, **57**(8), 1773–1792.

McCullagh, P. e Nelder, J. A. (2019). *Generalized linear models*. Routledge.

Messenger, R. e Mandell, L. (1972). A modal search technique for predictive nominal scale multivariate analysis. *Journal of the American statistical association*, **67**(340), 768–772.

Morgan, J. N. e Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, **58**(302), 415–434.

Oates, T. e Jensen, D. (1999). Toward a theoretical understanding of why and when decision tree pruning algorithms fail. Em *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, página 372–378, USA. ISBN 0262511061.

Olsson, U. (2002). *Generalized linear models: an applied approach*. Lund, Studentlitteratur.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, **1**(1), 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, **27**(3), 221–234.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*. Morgan Kaufmann.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Raileanu, L. E. e Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, **41**(1), 77–93.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, **27**(3), 379–423.

Snoek, J., Larochelle, H. e Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *arXiv*.

Strasser, H. e Weber, C. (1999). *On the asymptotic theory of permutation statistics*. Report. Vienna University of Economics and Business Administration.

Strobl, C., Malley, J. e Tutz, G. (2009). An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, **14**(4), 323.

Therneau, T. e Atkinson, B. (2019). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15.

Therneau, T. e Atkinson, E. (1997). An introduction to recursive partitioning using the rpart routines. *Mayo Clinic*, **61**.

Venables, B. e Ripley, B. (2002). *Modern Applied Statistics With S-Plus*. ISBN 978-1-4899-2821-4.

Vieira, A. M. C. (2008). *Modelagem simultânea de média e dispersão e aplicações na pesquisa agronômica*. Tese de doutorado, Universidade de São Paulo.

Vieira, F. M. C., Deniz, M., Silva, I. J. O. d., Barbosa Filho, J. A. D., Vieira, A. M. C. e Gonçalves, F. S. (2015). Perdas pré-abate de frangos de corte: efeito dos períodos diários e do tempo de espera em clima subtropical. *Medicina Veterinária e Zootecnia*, **36**, 3887–3896.

Witten, I. H., Frank, E., Hall, M. A., Pal, C. J. e DATA, M. (2005). Practical machine learning tools and techniques. Em *DATA MINING*, volume 2, página 4.

Wright, M. N. e Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, **77**(1), 1–17.

Yan, Y. (2021). *rBayesianOptimization: Bayesian Optimization of Hyperparameters*. R package version 1.2.0.

Zeileis, A. e Hornik, K. (2007). Generalized m-fluctuation tests for parameter instability. *Statistica Neerlandica*, **61**(4), 488–508.

Zeileis, A., Hothorn, T. e Hornik, K. (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, **17**(2), 492–514.

Apêndice A

Otimização Bayesiana

O otimização Bayesiana consiste em um método para otimizar determinada função $f(\boldsymbol{x})$. Essa função aceita valores $\boldsymbol{x} \in \mathbb{R}^n$ e retorna um valor $\boldsymbol{y} \in \mathbb{R}$. Geralmente utiliza-se tal método para encontrar hiperparâmetros ótimos de um algoritmo de aprendizado de máquina.

Para isso, testa-se alguns valores iniciais de $\boldsymbol{x}$. Testado tais valores, utiliza-se a informação de todos eles para construir uma distribuição de probabilidade à posteriori. Para construir essa posteriori, utiliza-se prioris que são processos gaussianos. Posteriormente, preciso definir uma função de aquisição para avaliar qual próximo ponto é o melhor de se escolher.

A.1 Processo Gaussiano

Se uma sequência finita de pontos $a_1, ..., a_n$ é um processo Processo Gaussiano, então essa sequência induz a uma distribuição multivariada da Normal Padrão.

A.2 Função de aquisição

Assumido que valores de $f(\boldsymbol{x})$ vem de uma priori de Processos Gaussianos, é preciso escolher uma função de aquisião $a(.)$ de modo a encontrar o próximo valor de $\boldsymbol{x}$ otimizando a função $f(.)$. Na literatura, há algumas sugestões para $a(.)$, dentre elas, destaca-se

- Probabilidade de Melhora;
- Melhora esperada;

- Limite superior de confiança do Processo Gaussiano;

A.3 Funcionamento do algoritmo

Primeiramente é testado alguns valores iniciais de $\boldsymbol{x}$. Em seguida, constroí-se uma distribuição de probabilidade à posteriori utilizando processos gaussianos. O próximo valor de $\boldsymbol{x}$ é escolhido levando em conta o formato estimado de $f(\boldsymbol{x})$ e o resultado obtido pela função de aquisição. O processo é repetido algumas vezes, e em cada iteração, a função $f(.)$ é melhor modelada pelos processos gaussianos.

Finalmente, após algumas iterações, é possível escolher um valor $\boldsymbol{x}$ que otimize a função $f(.)$. Snoek *et al.* (2012) explica melhor como funciona a otimização bayesiana e a técnica em si está implementada no `R` no pacote `rBayesianOptimization` (Yan, 2021).

Apêndice B

Códigos

Todos os códigos utilizados encontram-se no repositório

- Repositório:

  https://github.com/ViniciusHideki/TG_ArvoreHibrida;