Árvores de Decisão: A Evolução do CART ao BART

Cleber Batista de Souza

Dissertação
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Estatística

Orientador: Profª. Drª. Chang Chiann

São Paulo, outubro de 2021

Árvores de Decisão: A Evolução do CART ao BART

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 13/12/2021. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof[a]. Dr[a]. Chang Chiann (orientadora) - IME-USP
- Prof. Dr. Flavio Soares Correa da Silva - MAC-IME
- Prof. Dr. Kim Samejima Mascarenhas Lopes - UFBA

Agradecimentos

Não podemos negar nosso passado, a linha do tempo é única e existiu uma única realização de uma *vida aleatória* a cada instante. As pessoas que conheci, particularmente as que estudam e ensinam, sempre me inspiraram respeito, acabei me casando com uma professora. Sou grato aos meus professores do IME que me iniciaram na jornada acadêmica no milênio passado, lá se vão três décadas desde que me formei e agora retorno às minhas origens.

Grata foi a minha surpresa ao encontrar ainda na ativa professores como o prof. Pedro Morettin e a profa. Vera Giusti, só para citar os dois com os quais tive aulas na graduação e no mestrado. O primeiro ainda em pleno vigor dando aulas e escrevendo um novo livro, a segunda, ensinando alunos a ensinar, nobres tarefas. São tantas boas lembranças de outros professores e amigos que eu conseguiria escrever um livro sobre o assunto.

Sou grato à minha esposa Luciana, que me incentivou a voltar aos estudos a alguns anos atrás. Aos meus filhos que primeiro ficaram incrédulos quando disse que havia passado no exame de ingresso no mestrado, depois ficaram pasmos. À minha mãe que preparava as marmitas quando estava na graduação e ao meu pai, que já partiu.

Agradeço à secretaria Regiane, pessoa fundamental para fornecer informações importantes, sempre disposta a ajudar, nunca falhou. Ao prof. Adilson que me forneceu instruções valiosas para retomar meus estudos e ingressar no programa de mestrado. Por último, mas não menos importante, agradeço a profa Chang, que por incrível que pareça, foi minha colega de turma. Pessoa inteligente, generosa e bem humorada, seguiu a carreira acadêmica e tem uma família linda.

Thanks to mr. Knuth who brought TEX to life.

Ao Criador que me deu vida, devo tudo a Ele.

Resumo

SOUZA, C. B. **Árvores de Decisão: A Evolução do CART ao BART**. Dissertação de Mestrado - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

O objetivo deste trabalho é apresentar a evolução do uso dos modelos de Árvore de Decisão cuja linhagem remonta ao modelo CART (*Classification And RegressionTrees*) apresentado na publicação seminal Breiman *et al.* (1984). O modelo CART gerou uma sequência frutífera de modelos a partir da ideia de replicação da amostra disponível (via bootstrap) e/ou multiplicação no número de árvores (*ensembles*) para compor um resultado final. Passando pelo Bagging com replicação de amostras seguido das Florestas Aleatórias com a soma de múltiplas de árvores, apresentamos os modelos baseados em *boosting*: AdaBoost, Gradiente Boost e XGBoost. Surgidos a partir da utilização dos modelos aditivos, árvores são ajustadas em sequência onde cada árvore subsequente procura diminuir o erro cometido pela precedente e ao mesmo tempo maximizar uma função de perda que engloba o conjunto de árvores como um todo, o resultado final é a soma de todas árvores geradas. Os modelos de árvores Bayesianas também são apresentados: árvores Bayesianas CART e árvores Bayesianas BART. Para cada modelo desenvolvemos, segundo aplicável, como a árvore é construída, estimativas de erro, funções de perda adequadas, medidas de importância de variáveis, algoritmo de cálculo e uma ilustração para entendimento. No final mostramos resultados de simulação e aplicações em dados reais.

Palavras-chave: Árvores de Decisão, Árvores de Regressão, Árvores de Classificação, CART, Bagging, Florestas Aleatórias, AdaBoost, Boosting, Gradiente Boost, XGBoost, Árvores Bayesianas, BART.

Abstract

SOUZA, C. B. **Decision Tress: The Evolution from CART to BART**.

The objective of this work is to present the evolution of the use of Decision Tree models whose lineage goes back to the CART model (*Classification And RegressionTrees*) presented in the seminal publication Breiman *et al.* (1984). The CART model generated a fruitful sequence of models from the idea of replicating the available sample (via bootstrap) and/or multiplying in the number of trees (*ensembles*) to compose a final result. Going through Bagging with replication of samples followed by Random Forests with the sum of multiples of trees, we present the models based on *boosting*: AdaBoost, Gradient Boost and XGBoost. Arising from the use of additive models, trees are adjusted in sequence where each subsequent tree seeks to reduce the error made by the preceding one and at the same time maximize a loss function that encompasses the set of trees as a whole, the final result is the sum of all generated trees. Bayesian tree models are also presented: Bayesian CART trees and Bayesian BART trees. For each model we develop, as applicable, how the tree is constructed, error estimates, suitable loss functions, variable importance measures, calculation algorithm and an illustration for understanding. At the end we provide simulation results and applications on real data.

Dissertação de Mestrado - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

Keywords: Decision Trees, Regression Trees, Classification Trees, CART, Bagging, Random Forest, AdaBoost, Boosting, Gradient Boost, XGBoost, Bayesian CART Trees, BART.

All models are wrong, but some are useful.
George Cox

Conteúdo

Abreviaturas viii

Notação ix

Lista de Figuras x

Lista de Tabelas xii

1 Introdução 1

2 CART 4

2.1 Construindo Árvores . . . 4
2.2 Erro Esperado na Construção de T . . . 8
2.3 Estimativas para o Erro Esperado de T . . . 9
2.3.1 Erro Interno das Folhas . . . 9
2.3.2 Árvores de Regressão . . . 10
2.3.3 Árvores de Classificação . . . 11
2.4 Selecionando Partições . . . 12
2.4.1 Árvores de Regressão . . . 13
2.4.2 Árvores de Classificação . . . 13
2.5 Decidindo Quando Declarar um Nó Terminal . . . 15
2.6 Regularização . . . 16
2.7 Importância das Variáveis . . . 16
2.8 Vantagens do Modelo CART . . . 17
2.9 Ilustração . . . 18

3 *Random Forests* 25

3.1 *Bagging* . . . 25
3.2 Decomposição Viés-Variância . . . 26
3.2.1 Modelo de Bayes . . . 26
3.2.2 Árvores de Regressão . . . 28
3.2.3 Árvores de Classificação . . . 29
3.3 *Ensembles* . . . 30
3.3.1 Árvores de Regressão . . . 31
3.3.2 Árvores de Classificação . . . 33
3.4 *Random Forests* . . . 33

3.5 Estimativa do Erro Generalizado . . . 34
3.6 Importância das Variáveis . . . 34
3.7 Ilustração . . . 36

4 AdaBoost **42**
4.1 CART como Classificador Fraco . . . 42
4.2 Modelos Aditivos . . . 44
4.3 AdaBoost Multiclasse . . . 47
4.4 Ilustração . . . 48

5 *Gradient Boosting* **51**
5.1 O Método do Gradiente . . . 51
5.2 *Boosted Trees* . . . 52
5.3 Otimização via *Gradient Boosting* . . . 53
5.3.1 Árvores de Regressão . . . 54
5.3.2 Árvores de Classificação . . . 56
5.4 Regularização . . . 59
5.5 Importância das Variáveis . . . 60
5.6 Ilustração . . . 60

6 XGBoost **65**
6.1 Otimizando a Função Objetivo . . . 65
6.1.1 Árvores de Regressão . . . 67
6.1.2 Árvores de Classificação . . . 69
6.2 Regularização . . . 70
6.3 Ilustração . . . 70

7 Modelo Bayesiano CART **77**
7.1 Um Modelo Preditivo para cada Folha . . . 77
7.2 Uma Priori para T . . . 78
7.3 Uma Priori para $\Theta|T$ . . . 79
7.3.1 Árvores de Regressão . . . 79
7.3.2 Árvores de Classificação . . . 80
7.4 Especificando uma Regra de Particionamento . . . 80
7.5 Cálculo da Posteriori $P(T|X,Y)$ . . . 81
7.6 Identificação da Árvore Final . . . 82
7.7 Ilustração . . . 82

8 Modelo Bayesiano BART **84**
8.1 Especificando Prioris . . . 84
8.1.1 Uma Priori para $\mu_{km}|T_m$ . . . 84
8.1.2 Uma Priori para T_m . . . 85
8.1.3 Uma Priori para σ . . . 85
8.1.4 Um Valor para M . . . 85
8.2 Cálculo das Posteriores . . . 85

8.3 Inferência sobre Y . . . 87
8.4 BART para Classificação . . . 87
8.5 Ilustração . . . 88
8.5.1 GROW . . . 88
8.5.2 PRUNE . . . 90

9 Simulação **92**
9.1 Objetivo . . . 92
9.2 Árvores de Regressão . . . 92
9.2.1 Especificação das Expressões de Friedman . . . 92
9.2.2 Resultados . . . 93
9.3 Árvores de Classificação . . . 95
9.3.1 Especificação das equações das *waveforms* . . . 95
9.3.2 Resultados . . . 96

10 Aplicação **99**
10.1 Árvores de Regressão . . . 99
10.1.1 Descrição dos *datasets* . . . 99
10.1.2 Resultados . . . 101
10.2 Árvores de Classificação . . . 102
10.2.1 Descrição dos *datasets* . . . 102
10.2.2 Resultados . . . 106

11 Conclusões **109**
11.1 Considerações Finais . . . 109
11.2 Sugestões para Pesquisas Futuras . . . 109

Appendices **111**

A Tabelas de Resultados da Simulação **112**
A.1 Árvores de Regressão - *Friedman#1* . . . 112
A.2 Árvores de Regressão - *Friedman#2* . . . 117
A.3 Árvores de Regressão - *Friedman#3* . . . 121
A.4 Árvores de Classificação - *waveform* . . . 125

B Código Python da Simulação **129**
B.1 Árvores de Regressão . . . 129
B.2 Árvores de Classificação . . . 131

C Código Python Datasets **134**
C.1 Árvores de Regressão . . . 134
C.2 Árvores de Classificação . . . 135

Bibliografia **138**

Abreviaturas

AdaBoost	*Adaptative Boosting*
Bagging	Bootstrap Aggregating
BART	*Bayesian Regression Trees* - Árvores de Regressão Bayesianas
CART	*Classfication And Regression Trees* - Árvores de Classificação e Regressão
RandomForest	Random Forest - Florestas Aleatórias
XGBoost	*Extreme Gradient Boost*

Notação

$\mathbb{1}(\cdot)$	função indicadora: $\mathbb{1}(\cdot) = 1$, se a expressão $\cdot$ verdadeira, 0 caso contrário
$\mathcal{C}$	conjunto de opções de uma variável categórica
c_k	a *k-ésima* classe
d_η	profundidade do nó η (BART)
$\mathbb{E}$	esperança
$\mathbb{V}ar$	variância
$\mathbb{C}ov$	covariância
T	denota uma árvore
$\lvert\cdot\rvert$	cardinalidade de um conjunto
$\tilde{T}$	conjunto de nós terminais de uma árvore
Φ	função de impureza
$i(t)$	medida de impureza do nó t
$i_E(t)$	medida de impureza baseada na entropia cruzada
$i_G(t)$	medida de impureza baseada no índice de Gini
$i_R(t)$	medida de impureza na taxa de re-substituição
$\Delta i(s,t)$	decréscimo de impureza da partição s em um nó t
K	número de opções de uma variável categórica
	número de ordem na validação cruzada
L	função de perda
$\mathcal{D}$	conjunto de dados de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, \lvert\mathcal{D}\rvert = n\}$
n	número de observações do conjunto de aprendizado
p	número de variáveis preditoras
$\mathcal{Q}$	conjunto de perguntas possíveis em uma árvore
s	uma partição de um nó
t	um nó genérico da árvore
$\mathbf{X}$	vetor das variáveis aleatórias preditoras
$\mathbf{x}$	vetor dos valores observados de $\mathbf{X}$
$\mathcal{X}$	espaço das variáveis preditoras
X_j	variável aleatória correspondente à *j-ésima* variável preditora
$\mathbf{x}_j$	vetor de observações da variável X_j
Ω	universo das variáveis resposta e preditoras
Y	variável aleatória da variável resposta
$\mathcal{Y}$	espaço da variável resposta Y
$\mathbf{y}$	vetor de observações da variável resposta
y_i	*i-ésima* observação da variável resposta

Lista de Figuras

2.1 Diagrama de uma árvore CART. 4
2.2 A construção da árvore começa quando os dados $\mathcal{D}$ chegam ao primeiro *nó* $x_1 \leq v_1$ - *raiz*. Cada ◯ é *nó interno* e cada □ é um *nó terminal* ou *folha*. A pergunta em cada nó interno, *e.g.*, $x_1 \leq v_1$, particiona o conjunto de dados em duas partes mutuamente exclusivas - S=Sim e N=Não. Em cada folha temos uma predição $\hat{y} = T(\mathbf{x})$. 6
2.3 Exemplo de uma árvore usando 2 variáveis preditoras. 7
2.4 Diagrama com as regiões delimitadas pelos nós terminais da árvore na Figura 2.3 . . 7
2.5 Métricas de impureza para árvores de classificação com resposta binária 15
2.6 Linhagens do modelo CART. 19
2.7 Ilustração: CART Regressão - Particionamento do nó raiz. 21
2.8 Ilustração: CART Classificação - Particionamento do nó raiz. 24

3.1 Ilustração: Random Forest - Amostras Bootstrap e as respectivas Amostras *out-of-bag*. 36
3.2 Ilustração: Random Forest com 3 Árvores CART. 37
3.3 Ilustração: Random Forest - Amostras Bootstrap e as respectivas Amostras out-of-bag. 39
3.4 Ilustração: Random Forest com 3 Árvores CART. 40

4.1 Classificador fraco para o AdaBoost. Árvores CART com apenas dois nós terminais (stumps). 43
4.2 Esquema do funcionamento do AdaBoost.M1. A cada iteração uma versão do conjunto de dados é utilizada. O classificador final é a soma ponderada dos classificadores. 43
4.3 Curva do peso α em função do erro 44
4.4 Ilustração: AdaBoost Primeira Árvore CART. 50

5.1 Método do Gradiente para $f(x) = x^2 - x - 2, \rho = 0{,}25$. 52
5.2 Ilustração: Grandiente Boosting Regressão - Primeira Iteração. 61
5.3 Ilustração: Grandiente Boosting Regressão - Segunda Iteração. 62
5.4 Ilustração: Grandiente Boosting Classificação - Primeira Iteração. 64

6.1 Ilustração: XGBoost Regressão - Primeira Iteração. 73
6.2 Ilustração: XGBoost Classificação - Primeira Iteração. 76

7.1 $P(T)$ em função de α e β - no eixo x o número de nós terminais. 79
7.2 Representação das Operações: GROW e PRUNE. 82
7.3 Representação da Operação: CHANGE. 82
7.4 Representação da Operação: SWAP. 83

9.1 *Friedman #1*: MSE - Média de 100 simulações para cada amostra. 94
9.2 *Friedman #2*: MSE - Média de 100 simulações para cada amostra. 96
9.3 *Friedman #3*: MSE - Média de 100 simulações para cada amostra. 97
9.4 *Waveform*: Erro de Classificação - Média de 100 simulações para cada amostra. . . . 98

10.1 *dataset* car-price-audi: mse - Média de 10 de amostras por validação cruzada. 102
10.2 *dataset* car-price-bmw: mse - Média de 10 de amostras por validação cruzada. 102
10.3 *dataset* insurance: mse - Média de 10 de amostras por validação cruzada. 103
10.4 *dataset* wine-quality: mse - Média de 10 de amostras por validação cruzada. 103
10.5 *dataset* student-performance: mse - Média de 10 de amostras por validação cruzada. 104
10.6 *dataset* bank-note: acurácia - Média de 10 de amostras por validação cruzada. 107
10.7 *dataset* bank-chunners: acurácia - Média de 10 de amostras por validação cruzada. . 107
10.8 *dataset* spam: acurácia - Média de 10 de amostras por validação cruzada. 107
10.9 *dataset* isolet: acurácia - Média de 10 de amostras por validação cruzada. 108
10.10 *dataset* letter-recognition: acurácia - Média de 10 de amostras por validação cruzada. 108

Lista de Tabelas

2.1 Ilustração: CART Regressão - Conjunto de dados. 19
2.2 Ilustração: CART Regressão - Partições x_1. 20
2.3 Ilustração: CART Regressão - Partições x_2. 20
2.4 Ilustração: CART Regressão - Partições x_3. 21
2.5 Ilustração: CART Classificação - Conjunto de dados. 22
2.6 Ilustração: CART Classificação - Partições x_1. 23
2.7 Ilustração: CART Classificação - Partições x_2. 23
2.8 Ilustração: CART Classificação - Partições x_3. 23

3.1 Ilustração: Random Forest Regressão - Conjunto de dados. 36
3.2 Ilustração: Random Forest - Predições out-of-bag. 38
3.3 Ilustração: Random Forest Classificação - Conjunto de dados. 39
3.4 Ilustração: Random Forest - Predições out-of-bag. 40

4.1 Ilustração: AdaBoost Classificação - Conjunto de dados. 49
4.2 Ilustração: AdaBoost Classificação - iteração 1. 50

5.1 Método do Gradiente para $f(x) = x^2 - x - 2, \rho = 0{,}25$ 52
5.2 Funções de perda e Gradientes para árvores de regressão. 56
5.3 Ilustração: *gradient boosting* Regressão - Conjunto de dados. 61
5.4 Ilustração: *gradient boosting* Regressão - Pseudo-resíduos iteração 1. 61
5.5 Ilustração: *gradient boosting* Regressão - Pseudo-resíduos iteração 2. 62
5.6 Ilustração *gradient boosting* Classificação - Conjunto de dados. 63
5.7 Ilustração: *gradient boosting* Classificação - Pseudo-resíduos iteração 1. 63
5.8 Ilustração: *gradient boosting* Classificação - Pseudo-resíduos iteração 2. 64

6.1 Ilustração: XGBoost Regressão - Conjunto de dados. 71
6.2 Ilustração: XGBoost Regressão - Pseudo-resíduos iteração 1. 71
6.3 Ilustração: XGBoost Regressão - Partições x_1. 72
6.4 Ilustração: XGBoost Regressão - Partições x_2. 72
6.5 Ilustração: XGBoost Regressão - Partições x_3. 73
6.6 Ilustração: XGBoost Classificação - Conjunto de dados. 73
6.7 Ilustração: XGBoost Classificação - Pseudo-resíduos iteração 1. 74
6.8 Ilustração: XGBoost Classificação - Partições x_1. 75
6.9 Ilustração: XGBoost Classificação - Partições x_2. 75
6.10 Ilustração: XGBoost Classificação - Partições x_3. 75

9.1 Simulação Friedman#1. 93
9.2 Simulação Friedman#2. 94
9.3 Simulação Friedman#3. 95
9.4 Simulação waveform. 97

10.1 Aplicação Árvores de Regressão - mse. 101
10.2 Aplicação Árvores de Classificação - acurácia. 106

A.1 Simulação Friedman#1. 112
A.2 Simulação Friedman#2. 117
A.3 Simulação Friedman#3. 121
A.4 Simulação waveform. 125

1 Introdução

Os modelos de regressão linear são conhecidos e estudados com muito esmero há bastante tempo, por exemplo, Johnson e Wichern (1988) e mais recentemente Montgomery *et al.* (2012). Regra geral, o objetivo destes modelos é estabelecer uma relação linear entre uma variável resposta e um conjunto de variáveis preditoras, *i.e.*, variáveis que se acredita de alguma forma capazes de prever a variável resposta a partir da sua observação. Além de prever a variável resposta, o objetivo do modelo também é adquirir algum conhecimento sobre o quanto cada variável preditora permite explicar a variável resposta. Os modelos de regressão não são os únicos a buscar esse objetivo.

Os modelos baseados em *árvores* competem não só com os modelos de regressão linear tradicionais, mas com uma pletora de modelos que se propõe a modelar respostas em função de outras variáveis observadas. Competem com as árvores, por exemplo, os modelos de svm *(support vector machines)* ou máquinas de vetor de suporte, mars *(multivariate adaptative regression splines)*, k-nn *(k-nearest neighbors)* ou k-vizinhos mais próximos.

Intuitivamente as árvores de decisão são modelos simples de compreender. Por exemplo, as tradicionais perguntas que um médico faz ao paciente sobre os sintomas da possível doença em questão, traçam um caminho para o diagnóstico. A cada Sim ou Não o médico toma a decisão de fazer mais uma pergunta ou já inferir a doença e um possível tratamento. Resultados de exames clínicos e/ou laboratoriais são parte das perguntas para se chegar à resposta final.

Começamos a construção de uma árvore colocando todas as observações juntas num único nó, chamado raiz. A ideia central é procurar entre as variáveis preditoras a que, segundo alguma métrica, melhor particiona as observações que estão na raiz em dois grupos. Cada variável preditora fornece um conjunto finito de perguntas cuja resposta deve ser Sim ou Não, se a variável for numérica a pergunta será do tipo $x \leq c$, se for categórica a pergunta será $x \in S$, em que S é um subconjunto formado pelas opções disponíveis na respectiva variável. Cada pergunta define uma partição das observações: a resposta Sim a uma pergunta leva a observação a um novo nó à esquerda da raiz, a resposta Não, a um nó à direita. Definida uma métrica, doravante medida de impureza, escolhemos a partição cujo decréscimo de impureza é o maior. Uma medida de impureza fornece uma métrica de quão homogêneos os valores da variável resposta estão dentro de um nó: no limite, se todas os valores forem iguais a impureza é zero e quanto mais dispares os valores, maior a impureza.

As árvores de decisão podem ser de regressão ou classificação. No primeiro caso a variável resposta é numérica; no segundo, categórica. Considerando uma variável resposta numérica, uma medida de impureza pode ser a variância dentro do nó, considerando uma variável categórica, pode ser a proporção de observações classificadas erradas dentro do nó.

Após particionar o nó raiz, uma nova busca com todas as variáveis disponíveis, e as respectivas perguntas, em cada nó, à esquerda e à direita, deverá ser efetuada. Assim deve ocorrer sucessivamente, o que torna a construção da árvore um procedimento recursivo e guloso, um vez que a cada nó todas as variáveis disponíveis e todas as perguntas possíveis devem ser testadas. Quando um nó não é mais dividido ele é declarado terminal.

Determinar o tamanho da árvore, *i.e.*, quando declarar um nó terminal faz parte do processo de construção da árvore. Estratégias incluem crescer a árvore até um limite máximo e realizar uma poda, declarar um nó terminal se o número de observações for menor ou igual a um certo limite, ou particionar um nó somente se o decréscimo de impureza superar algum limiar pré-definido.

Uma vez que a árvore foi construída, o passo final é atribuir um valor, *i.e.*, uma predição para as observações que alocadas em cada nó terminal: para árvores de regressão pode ser a média da

variável resposta; para árvores de classificação pode ser a classe modal.

As origens dos modelos de árvore de decisão remontam ao trabalho de Morgan e Sonquist (1963) no qual os autores modelam uma pesquisa sobre renda usando como variáveis explicativas raça, idade, nível educacional e se é fazendeiro ou não.

Diversas modalidades de construção de árvores foram estudadas: as propostas variam basicamente em a) aceitação de variáveis numéricas e categóricas, seja entre as variáveis preditoras seja na variável resposta, b) medida de impureza, c) critério de partição, d) critério para declarar um nó terminal, e) se o método realiza poda ou não, f) se o método lida com variáveis missing ou não e g) se as árvores são binárias ou não.

Malehi e Jahangiri (Malehi e Jahangiri, 2019) contém um resumo das principais abordagens: THAID (Messenger e Mandell, 1972), CHAID (Biggs *et al.*, 1991; Kass, 1980), CART (Breiman *et al.*, 1984), ID3 (Quinlan, 1986), FACT (Loh e Vanichsetakul, 1988), C4.5 (Quinlan, 1993), QUEST (Loh e Shih, 1997), CRUISE (Kim e Loh, 2001) e GUIDE (Loh, 2009).

Todas estas construções modelam uma única árvore. A árvore CART, Classification And Regression Trees, é a que gerou um maior número de frutos. Estes frutos são modelos que utilizam composições (*ensembles*) de várias árvores para se chegar a um resultado final. Regra geral estas técnicas se valem da aleatoriedade através de amostras bootstrap das observações e/ou aleatoriedade na escolha das variáveis preditoras para realizar as partições na construção de cada árvore do *ensemble*.

São os modelos clássicos que evoluíram a partir das árvores CART que vamos explorar neste trabalho: Bagging (Breiman, 1996b), Florestas Aleatórias (Breiman, 2001), AdaBoost (Freund e Schapire, 1997), Gradiente Boosting (Friedman, 2000), XGBoost (Chen e Guestrin, 2016), árvores bayesianas CART (Chipman *et al.*, 2010, 1998).

As árvores CART permitem uma modelagem bastante flexível e apresentam diversas vantagens em relação aos modelos estatísticos tradicionais de predição. Entre as vantagens podemos citar:

- as variáveis preditoras podem ser numéricas ou categóricas
- a variável resposta pode ser númerica ou categórica
- resultado fácil de interpretar graficamente
- suporta base de dados de alta dimensão
- suporta grandes bases de dados
- não necessita assumir uma distribuição para os dados
- captura relacionamentos não-lineares e interações de vários níveis
- invariante a transformações das variáveis preditoras
- robustez em relação a valores missing
- robustez em relação a outliers
- robustez em relação a multicolinearidade
- permite extrair subgrupos homogêneos de observações.

Estas características tornam os algoritmos baseados em árvores bastante atrativos para diversos tipos de problemas, detalhes ver Breiman (2001) e Hastie *et al.* (2009). Especialmente os *ensembles* são populares em sites de competição de *machine learning* (Chen e Guestrin, 2016). Ainda recentetemente Yalov (2019) apresenta um novo desdobramento dos modelos bayesianos BART.

Este trabalho está divido na seguinte sequência: o Capítulo 2 apresenta em detalhes o modelo CART, na sequência no Capítulo 3 mostramos como as árvores CART são utilizadas nas *Random*

Forests com significativa diminuição nos erros de classificação e/ou regressão. As árvores AdaBoost (*Adaptative Boosting*) no Capítulo 4 introduzem a ideia de classificador fraco, *i.e.*, uma sequência de árvores com poucos nós acertando pouco mais que 50% (nos problemas de classificação) ao serem *somadas* apresentam um resultado melhor do que o CART. No Capítulo 5 detalhamos como a técnica de otimização via método do gradiente, utilizada amplamente em problemas de Cálculo, foi adaptada para construir as árvores *Gradient Boosting*. No Capítulo 6 as árvores XGBoost são apresentadas como uma variante do *Gradient Boosting*. Em cada caso além do desenvolvimento teórico de cada técnica, apresentamos um algoritmo de cálculo e um exemplo básico de como construir cada tipo de árvore.

Nos Capítulos 7 e 8 as árvores Bayesianas são desenvolvidas a partir de um novo paradigma: distribuição de probabilidade a priori para os dados e para as árvores e consequente cálculo de posteriores. A utilização do algoritmo de Metrópolis-Hastings é utilizada para geração destas árvores.

Para testar todas estas técnicas realizamos simulações cujo resultado é mostrado no Capítulo 9. Aplicações em dez conjuntos de dados colhidos no site de competição de *machine learning* https://www.kaggle.com/ e no repositório https://archive-beta.ics.uci.edu/ são avaliados, Capítulo 10. Utilizamos a linguagem Python com as respectivas bibliotecas como base para ajustar todos os modelos.

2 CART

Nosso ponto de partida para o estudo das árvores de decisão é o modelo o CART acrônimo de *Classification And Regression Trees*, título do livro de Breiman *et al.* (1984). Este é o primeiro trabalho sobre o tema que apresenta de forma integrada toda a metologia de construção das árvores. O modelo CART, como todos os outros modelos de árvores que estudaremos neste trabalho, são baseados em *árvores binárias*. As definições a seguir permitem esclarecer os elementos fundamentais que compõe uma árvore binária:

Definição 2.1 (Árvore)**.** Uma *árvore* é um grafo $G = (V, E)$ no qual quaisquer dois vértices (nós) estão conectados exatamente por um único caminho.

Definição 2.2 (Árvore com raiz)**.** Uma *árvore com raiz* é uma árvore na qual um dos nós é designado como *raiz*. Adicionalmente assumimos que uma árvore com raiz é um grafo direcionado a partir do nó raiz.

Definição 2.3 (Nós pais e nós filhos)**.** Se existe uma aresta que conecta dois nós t_1 e t_2, em que $(t_1, t_2) \in E$, então t_1 é chamado de nó *pai* de t_2, enquanto que t_2 é nó *filho* de t_1.

Definição 2.4 (Nós internos e nós terminais)**.** Numa árvore com raiz, um nó é chamado *interno* se ele tem um ou mais filhos e nó *terminal* se ele não tem filhos. Um nó terminal também é conhecido como *folha*.

Definição 2.5 (Árvore Binária)**.** Uma *árvore binária* é uma árvore com raiz na qual todos os nós internos tem exatamente dois nós filhos.

Esquematicamente a Figura 2.1 exemplifica estas definições:

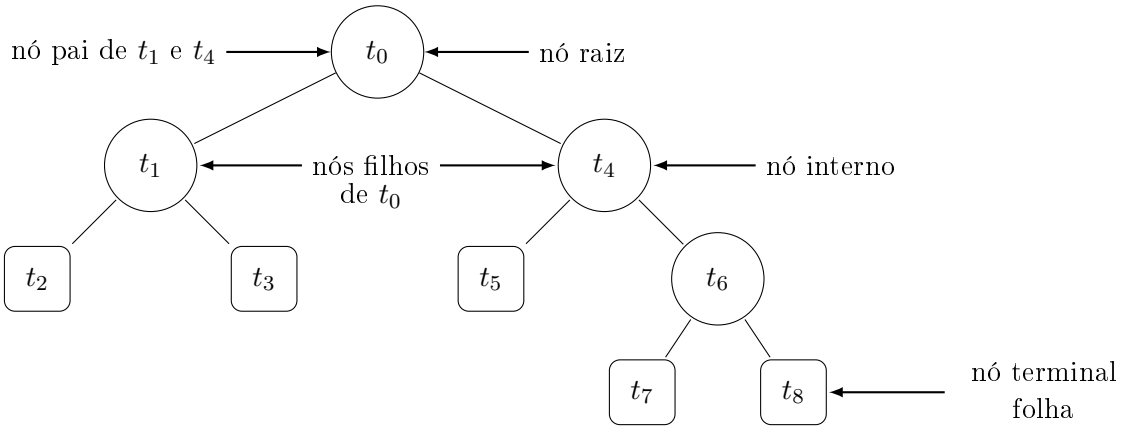


Figura 2.1: *Diagrama de uma árvore CART.*

2.1 Construindo Árvores

São dois tipos de árvores abordados na metodologia CART: *árvores de classificação* e *árvores de regressão*. As árvores de regressão têm a variável resposta numérica, enquanto que as árvores de

classificação têm uma resposta categórica. Neste caso, cada opção de resposta é chamada de *classe*. As variáveis preditoras podem ser numéricas ou categóricas, conforme explicado a seguir.

A construção de uma árvore pressupõe a disponibilidade de um conjunto de dados composto de um vetor de variáveis de entrada, ou *variáveis preditoras*, associadas a uma variável resposta. O objetivo é construir um modelo preditivo para a variável resposta baseado nas variáveis preditoras.

Formalizando, Ω é o universo de origem dos dados utilizados na construção da árvore. Vamos assumir um conjunto de valores de entrada $\mathbf{x} = (x_1, \ldots, x_p)'$ de maneira que $\mathbf{x}$ é uma realização de $\mathbf{X} = (X_1, \ldots, X_p)'$ vetor de variáveis aleatórias definidas no espaço p-dimensional $\mathcal{X}$. Assumimos também que y, a variável resposta é definida como um realização da variável $Y \in \mathcal{Y}$, de modo que $\Omega = \mathcal{X} \times \mathcal{Y}$.

As variáveis X_j, $j = 1, \ldots, p$, serão consideradas como numéricas ou categóricas:

- X_j é numérica se $X_j \in \mathcal{X}_j \subset \mathbb{R}$.
- X_j é categórica se assume um conjunto finito de valores sem uma ordem estabelecida: $X_j \in \mathcal{C} = \{c_1, \ldots, c_K\}$ em que $c_k, k = 1, \ldots, K$, são chamadas de *classes*.

O mesmo vale para Y.

Tipicamente um conjunto de dados $\mathcal{D}$ utilizado para a construção de uma árvore é constituído de n amostras de pares $(\mathbf{x}_i, y_i)$, em que $\mathbf{x}_i = (x_{1i}, \ldots, x_{ip}), i = 1, \ldots, n$, de valores observados de entrada e a respectiva variável resposta. Formalmente,

Definição 2.6. $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$.

À árvore gerada a partir do conjunto de dados $\mathcal{D}$ denominaremos T. O conjunto de nós terminais de T será anotado como $\tilde{T}$ e o número de nós terminais será $|\tilde{T}|$. Utilizaremos a notação $\hat{y} = T(\mathbf{x})$ para indicar que um vetor de entrada $\mathbf{x}$ tem como resposta $\hat{y}$, T também pode ser a própria árvore que inclui os procedimentos e as regras de construção. Eventualmente para deixar claro alguma expressão podemos utilizar $T_{\mathcal{D}}(\mathbf{x})$ para indicar o conjunto de dados utilizados na construção de T.

Exemplificando, suponha que temos disponível um conjunto de dados $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$ $p = 3$, em que X_1,$X_2 \in \mathbb{R}$, $X_3 \in \mathcal{C}$, uma variável categórica, em que $\mathcal{C} = \{c_1, \ldots, c_4\}$, $\mathcal{Y} = \mathbb{R}$. Os pares $(\mathbf{x}_i, y_i)$ são realizações das variáveis X_1, X_2, X_3, Y definidas no espaço $\Omega = \mathcal{X} \times \mathcal{Y}$, $|\mathcal{D}| = n$ é a cardinalidade de $\mathcal{D}$.

O interesse em construir uma árvore CART é obter um modelo $\hat{y} = T(\mathbf{x})$ em que T representa a árvore e $\mathbf{x} = (x_1, x_2, x_3)$. Uma árvore hipotética para esse conjunto de dados é representada pela Figura 2.2.

A construção da árvore (Figura 2.2) começa com todos os dados disponíveis em $\mathcal{D}$. A pergunta $x_1 \leq v_1$ define uma **partição** (*split*) do conjunto $\mathcal{D}$ em dois subconjuntos: $\mathcal{D}_e$ (Sim para a pergunta $x_1 \leq v_1$) e $\mathcal{D}_d$ (Não para a pergunta $x_1 \leq v_1$, *i.e.*, $x_1 > v_1$) à esquerda e à direita, respectivamente, e além disso, $\mathcal{D}_e \cap \mathcal{D}_d = \varnothing$.

Os dados $\mathcal{D}_e$ chegam ao nó à esquerda e uma nova pergunta $x_2 \leq v_2$ particiona o nó novamente. Os dados $\mathcal{D}_d$ chegam ao nó à direita e são divididos pela pergunta $x_3 \in S$, em que S é um subconjunto de $\mathcal{C}$, por exemplo, $S = \{c_2, c_3\}$. Os nós são particionados sucessivamente até se chegar aos nós terminais ou folhas em que encontramos uma predição $\hat{y}_j$ para as observações que são alocadas na respectiva folha, simbolizada por $\tilde{T}_j$. Cada folha define uma região R_j correspondente, $j = 1, \ldots, 5$.

Considere a função indicadora conforme a definição a seguir

Definição 2.7 (Função Indicadora)**.**

$$\mathbb{1}(\text{expressão}) = \begin{cases} 1, & \text{se expressão é verdadeira,} \\ 0, & \text{caso contrário.} \end{cases}$$

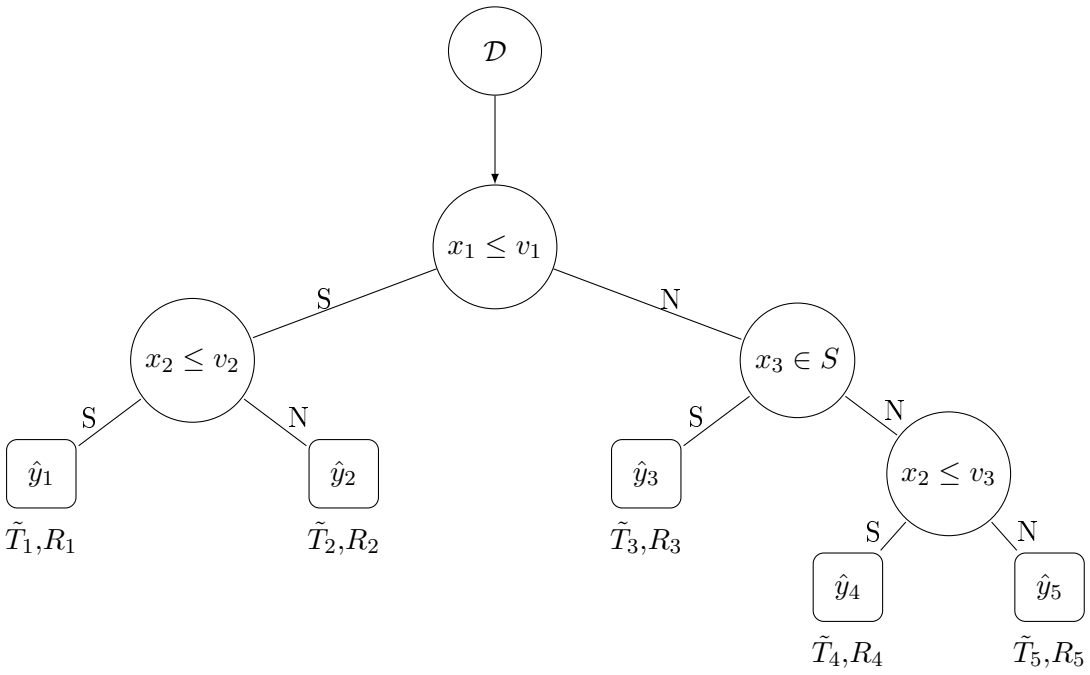


Figura 2.2: *A construção da árvore começa quando os dados $\mathcal{D}$ chegam ao primeiro* nó $x_1 \leq v_1$ - raiz. *Cada* ◯ *é* nó interno *e cada* □ *é um* nó terminal *ou* folha. *A pergunta em cada nó interno,* e.g., $x_1 \leq v_1$, *particiona o conjunto de dados em duas partes mutuamente exclusivas - S=Sim e N=Não. Em cada folha temos uma predição* $\hat{y} = T(\mathbf{x})$.

As regiões R_j que contém os dados referentes à respectiva folha $\tilde{T}_j, j = 1, \ldots, 5$, são delimitadas a partir da árvore de acordo com as seguintes expressões

$$\begin{aligned} R_1 &: \mathbb{1}(x_1 \leq v_1) \times \mathbb{1}(x_2 \leq v_2), \\ R_2 &: \mathbb{1}(x_1 \leq v_1) \times \mathbb{1}(x_2 > v_2), \\ R_3 &: \mathbb{1}(x_1 > v_1) \times \mathbb{1}(x_3 \in S)), \\ R_4 &: \mathbb{1}(x_1 > v_1) \times \mathbb{1}(x_3 \notin S) \times \mathbb{1}(x_2 \leq v_3), \\ R_5 &: \mathbb{1}(x_1 > v_1) \times \mathbb{1}(x_3 \notin S) \times \mathbb{1}(x_2 > v_3). \end{aligned} \tag{2.1}$$

Em cada região R_j é alocada a parte dos dados $\mathcal{D}_j$ que satisfaz a expressão da respectiva região. Considere $I_j = \{i | (\mathbf{x}_i, y_i) \in \mathcal{D}_j, i = 1, \ldots, n_j\}$ as amostras alocadas em $\mathcal{D}_j$, $\sum_{j=1}^{5} n_j = n$. Uma predição dos y's alocados em R_j é a média das observações em R_j, por exemplo,

$$\begin{aligned} I_1 &= \{i | (\mathbf{x}_i, y_i) \in \mathcal{D}_1, i = 1, \ldots, n_1\}, \\ n_1 &= |I_1|, \\ \hat{y}_1 &= \bar{y}_1 = \frac{1}{n_1} \sum_{i \in I_1} y_i. \end{aligned} \tag{2.2}$$

Considerando a mesma estrutura de árvore da Figura 2.2, se a variável resposta y é categórica assumindo valores em K classes, no que foi exposto até o momento, muda apenas a forma como $\hat{y}$ é calculado, o qual pode ser simplesmente a classe majoritária de y na respectiva folha.

As expressões (2.1) definem partições no espaço do conjunto de dados. Esse espaço pode ser visualizado de maneira mais simples num exemplo com somente duas dimensões $\mathbf{x} = (x_1, x_2) \in [0, 1] \times [0, 1]$. A árvore na Figura 2.3 define as expressões (2.3).

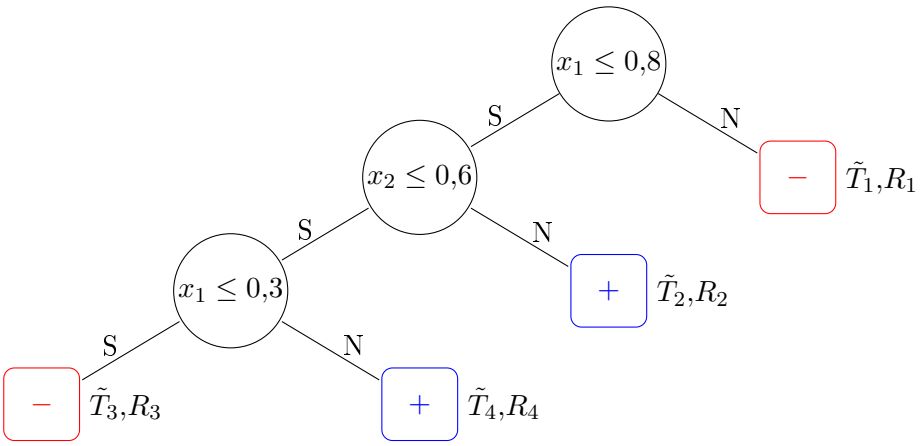


Figura 2.3: *Exemplo de uma árvore usando 2 variáveis preditoras.*

Expressões definidas pela árvore da Figura 2.3:

$$\begin{aligned} R_1 &: \mathbb{1}(x_1 > 0{,}8), \\ R_2 &: \mathbb{1}(x_1 \leq 0{,}8) \times \mathbb{1}(x_2 > 0{,}6), \\ R_3 &: \mathbb{1}(x_1 \leq 0{,}8) \times \mathbb{1}(x_2 \leq 0{,}6) \times \mathbb{1}(x_1 \leq 0{,}3), \\ R_4 &: \mathbb{1}(x_1 \leq 0{,}8) \times \mathbb{1}(x_2 \leq 0{,}6) \times \mathbb{1}(x_1 > 0{,}3). \end{aligned} \tag{2.3}$$

O espaço $[0,1] \times [0,1]$ particionado pela árvore (Figura 2.3) é visualizado na Figura 2.4.

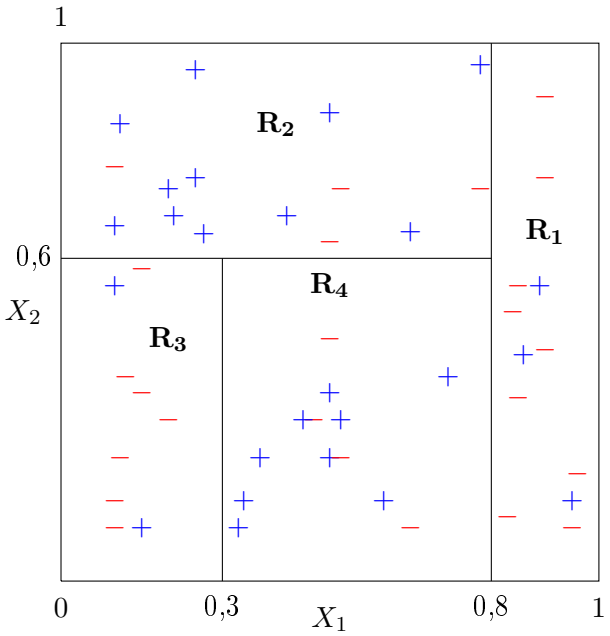


Figura 2.4: *Diagrama com as regiões delimitadas pelos nós terminais da árvore na Figura 2.3*

.

Quanto à predição de y, se $y \in \{+, -\}$ é categórica, usando o critério de classe majoritária, ou

classe modal, o resultado seria o seguinte

$$\begin{aligned} \hat{y}_1 = \hat{y}_3 = -, \\ \hat{y}_2 = \hat{y}_4 = +. \end{aligned} \tag{2.4}$$

Se $y \in \mathbb{R}$, a predição pode ser, por exemplo, o valor médio dos y's alocados em cada região.

Segundo Breiman *et al.* (1984) a construção de uma árvore, cuja resposta é categórica, o que não invalida a observação para uma resposta numérica, é composta de três elementos:

1. A seleção das partições.
2. A decisão de quando declarar um nó terminal ou continuar particionando.
3. A atribuição de uma classe a cada nó terminal.

Mas, como diz Breiman *et al.* (1984), para um conjunto de dados $\mathcal{D}$:

The crux of the problem is how to use the data $\mathcal{D}$ to determine the splits, the terminal nodes, and their assignments. It turns out that the class assignment problem is simple. The whole story is in finding good splits and in knowing when to stop splitting.

2.2 Erro Esperado na Construção de T

Uma vez que o objetivo da árvore é realizar uma predição, precisamos encontrar árvores que minimizem o erro esperado de predição (Gilles, 2014; Hastie *et al.*, 2009). Para estimar erros e definir critérios de construção da árvore, vamos considerar $\mathbf{X} = (X_1, \ldots, X_p)'$ um vetor aleatório $\mathbf{X} \in \mathcal{X}$ e Y uma variável aleatória, $Y \in \mathcal{Y}$. Definimos $\Omega = \mathcal{X} \times \mathcal{Y}$ o espaço das probabilidades e uma distribuição de probabilidade conjunta $P(X, Y) = P(X = \mathbf{x}, Y = y)$ em que $\mathbf{x}$ e y são realizações aleatórias do universo Ω.

Com estas considerações vamos definir o erro esperado de predição, ou erro generalizado, de árvore T, considerando uma função de perda L,

$$Erro(T) = \mathbb{E}_{X,Y}\{L(Y, T(X))\}, \tag{2.5}$$

em que L é uma função de perda que mede a discrepância entre Y e $T(X)$.

A intenção é obter uma árvore T que represente o universo Ω. O objetivo não é somente minimizar o $Erro(T)$ para o conjunto de aprendizado $\mathcal{D}$, mas encontrar uma árvore T que sirva também para representar todos os conjuntos de aprendizado não observados em $\Omega \setminus \mathcal{D}$. A equação (2.5) define teoricamente o erro generalizado do procedimento de construção da árvore.

Se Y é numérica, utilizamos a perda quadrática

$$L(Y, T(X)) = (Y - T(X))^2, \tag{2.6}$$

o erro esperado é

$$Erro(T) = \mathbb{E}_{X,Y}\{(Y - T(X))^2\}. \tag{2.7}$$

Se Y é categórica, tipicamente a função de perda escolhida é do tipo 0-1:

$$L(Y,T(X)) = \mathbb{1}(Y \neq T(X)), \tag{2.8}$$

desta forma L é a probabilidade da árvore realizar uma classificação errada

$$Erro(T) = \mathbb{E}_{X,Y}\{\mathbb{1}(Y \neq T(X))\} = P(Y \neq T(X)). \tag{2.9}$$

2.3 Estimativas para o Erro Esperado de T

Dado que a distribuição $P(X,Y)$ é geralmente desconhecida, o cálculo do $Erro(T)$ não é viável analiticamente. Por outro lado, uma estimativa empírica do $Erro(T)$ requer, teoricamente, um conjunto infinito de outros conjuntos $\mathcal{D}' \neq \mathcal{D}$ retirados independentemente de Ω, o que, na prática, também é inviável.

Em geral temos disponível um único conjunto de aprendizado $\mathcal{D}$ a partir do qual precisamos construir uma árvore T e estimar o erro. São três as abordagens mais comuns para calcular essas estimativas conforme Breiman *et al.* (1984).

Vamos definir $\bar{E}(T_{\mathcal{D}}, \mathcal{D}')$ como o erro de predição médio obtido a partir de uma árvore construída com o conjunto $\mathcal{D}$ distinto de $\mathcal{D}'$:

$$\bar{E}(T_{\mathcal{D}}, \mathcal{D}') = \frac{1}{n'} \sum_{(\mathbf{x},y)\in\mathcal{D}'} L(y, T_{\mathcal{D}}(\mathbf{x})). \tag{2.10}$$

1. A primeira abordagem é construir a árvore com todo o conjunto $\mathcal{D}$ e estimar o erro a partir dos mesmos dados de $\mathcal{D}$, *i.e.*, fazendo $\mathcal{D}' = \mathcal{D}$. Neste caso todo o conjunto de aprendizado se torna o conjunto de *treino* utilizado para construir a árvore T. Este é a chamada *estimativa de re-substituição* (Breiman *et al.*, 1984) ou *erro da amostra de treino*:

$$\widehat{Erro}_{treino} = \bar{E}(T_{\mathcal{D}}, \mathcal{D}). \tag{2.11}$$

   Essa estimativa geralmente é muito pobre, uma vez que o mesmo conjunto de dados é utilizado para construção e estimação do erro.

2. A segunda abordagem particiona $\mathcal{D}$ em dois subconjuntos selecionados aleatoriamente. O primeiro subconjunto denominado $\mathcal{D}_{treino}$ será utilizado para a construção da árvore T_{treino}, o segundo será denominado subconjunto de *teste*, $\mathcal{D}_{teste}$. A estimativa do $Erro(T)$ será dada por

$$\widehat{Erro}_{teste} = \bar{E}(T_{\mathcal{D}_{treino}}, \mathcal{D}_{teste}). \tag{2.12}$$

   Breiman *et al.* (1984) sugere selecionar para treino 2/3 de $\mathcal{D}$ e 1/3 para teste.

3. A terceira abordagem utiliza a **validação cruzada de ordem** K (*K-fold cross validation*). A validação cruzada consiste em dividir aleatoriamente $\mathcal{D}$ em K amostras distintas $\mathcal{D}_1, \ldots, \mathcal{D}_K$ de tamanho aproximado n/K, utilizar $K-1$ como $\mathcal{D}_{treino}$ para a construção da árvore e a amostra que ficou de fora como $\mathcal{D}_{teste}$. O procedimento é repetido K vezes até que todas as amostras sejam utilizadas como teste. A estimativa do $Erro(T)$ é dada pela média dos erros obtidos em cada amostra teste

$$\widehat{Erro}_{vc} = \frac{1}{K} \sum_{k=1}^{K} \bar{E}(T_{\mathcal{D}\setminus\mathcal{D}_k}, \mathcal{D}_k) \tag{2.13}$$

O estudo de Kohavi (1995) sugere que $K = 10$ produz estimativas estáveis e confiáveis.

2.3.1 Erro Interno das Folhas

Uma vez que a árvore T é construída, cada observação do conjunto de dados $\mathcal{D}$ é alocada em um dos nós terminais gerados em T. Considere J o número de nós terminais e $\tilde{T} = \{\tilde{T}_1, \ldots, \tilde{T}_J\}$ o conjunto de nós terminais. Cada nó terminal define uma partição $\mathcal{X}_j$ no espaço das variáveis preditoras $\mathcal{X}$. Teoricamente o erro de predição (2.5) é obtido a partir da minimização do erro em cada nó terminal, *i.e.*, minimizando o erro dentro de cada folha, minimizamos o erro de T, seja

$\tilde{T}_j(X) = \mathbb{1}(X \in \mathcal{X}_j) \cdot T(X)$:

$$\begin{aligned} Erro(T) &= \mathbb{E}_{X,Y}\{L(Y, T(X))\} \\ &= \sum_{j=1}^{J} P(X \in \mathcal{X}_j) \times \mathbb{E}_{X,Y|\tilde{T}_j}\{L(Y, \tilde{T}_j(X))\} \\ &= \sum_{j=1}^{J} P(X \in \mathcal{X}_j) \times Erro(\tilde{T}_j), \end{aligned} \tag{2.14}$$

note que $\mathbb{E}_{X,Y|\tilde{T}_j}\{L(Y, \tilde{T}_j(X))\} = Erro(\tilde{T}_j)$ é o erro esperado na folha $\tilde{T}_j$.

2.3.2 Árvores de Regressão

Utilizando a perda quadrática, para $j = 1, \ldots, J$, de acordo a equação 2.7 com temos

$$\begin{aligned} Erro(\tilde{T}_j) &= \mathbb{E}_{X,Y|\tilde{T}_j}\{(Y - \tilde{T}_j(X))^2\} \\ &= \mathbb{E}_{X,Y|\tilde{T}_j}\{(Y - \hat{y}_j)^2\}. \end{aligned} \tag{2.15}$$

A equação (2.15) é minimizada encontrando-se $\hat{y}_j^*$ de forma que

$$\hat{y}_j^* = \arg\min_{\hat{y} \in \mathcal{Y}} \mathbb{E}_{X,Y|\tilde{T}_j}\{(Y - \hat{y}_j)^2\}. \tag{2.16}$$

A equação (2.16) não pode ser resolvida analiticamente já que não conhecemos a distribuição conjunta $P(X, Y)$. No entanto, podemos utilizar uma aproximação para $\hat{y}_j^*$. Considere $n_j = |\tilde{T}_j|$ e $I_j = \{i | y_i \in \tilde{T}_j\}, j = 1, \ldots, J$, uma estimativa γ_j para $\hat{y}_j^*$ pode ser obtida minimizando

$$\begin{aligned} \gamma_j &= \arg\min_{\gamma} \sum_{i \in I_j} (y_i - \gamma)^2 \implies \\ &= \frac{\partial}{\partial \gamma} \sum_{i \in I_j} (y_i - \gamma)^2 = 0 \implies \\ \gamma_j &= \frac{1}{n_j} \sum_{i \in I_i} y_i \\ \gamma_j &= \bar{y}_j. \end{aligned} \tag{2.17}$$

Vamos mostrar que minimizando o erro localmente, de fato, minimizamos o erro de treinamento. Considere $p(j)$ a proporção de observações alocadas em $\tilde{T}_j$ como estimativa para $P(X \in \mathcal{X}_j)$:

$$\hat{P}(X \in \mathcal{X}_j) = p(j) = \frac{|\tilde{T}_j|}{|\mathcal{D}|} = \frac{n_j}{n}. \tag{2.18}$$

Conforme a equação (2.14)

$$\begin{aligned} Erro(T) &= \sum_{j=1}^{J} p(j) \times \frac{1}{n_j} \sum_{i \in I_j} (y_i - \gamma_j)^2 \\ &= \sum_{j=1}^{J} \frac{n_j}{n} \times \frac{1}{n_j} \sum_{i \in I_j} (y_i - \gamma_j)^2 \\ &= \frac{1}{n} \sum_{j=1}^{J} \sum_{i \in I_j} (y_i - \gamma_j)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} (y_i - T(\mathbf{x}_i))^2 = \frac{1}{n} \sum_{(\mathbf{x},y) \in \mathcal{D}} (y - T_{\mathcal{D}}(\mathbf{x}))^2 \\ &= \frac{1}{n} \sum_{(\mathbf{x},y) \in \mathcal{D}} L(y, T_{\mathcal{D}}(\mathbf{x})) \\ &= \bar{E}(T_{\mathcal{D}}, \mathcal{D}) = \widehat{Erro}_{treino}. \end{aligned} \tag{2.19}$$

2.3.3 Árvores de Classificação

Para a árvore de classificação, conforme a equação (2.9), temos

$$Erro(\tilde{T}_j) = P(Y \neq \tilde{T}_j(X) | X \in \mathcal{X}_j), \tag{2.20}$$

seja $\hat{y}_j^*$ o previsor que minimiza (2.20), temos

$$\begin{aligned} \hat{y}_j^* &= \arg \min_{c \in \mathcal{Y}} P(Y \neq c | X \in \mathcal{X}_j) \\ &= \arg \max_{c \in \mathcal{Y}} P(Y = c | X \in \mathcal{X}_j). \end{aligned} \tag{2.21}$$

Vemos que a classe modal c é a previsão que minimiza o erro em $\tilde{T}_j$: $\hat{y}_j^* = c$. Uma vez que p é desconhecida e $\hat{y}_j^*$ não pode ser calculada diretamente, podemos estimá-la usando a proporção de observações da classe c em $\tilde{T}_j$. Seja γ_j a estimativa de $\hat{y}_j^*$, considere n_{jc} o número de observações da classe c em $\tilde{T}_j$ e $p(c|j)$ a respectiva estimativa da probabilidade das observações serem alocadas na classe c em $\tilde{T}_j$:

$$\begin{aligned} n_{jc} &= \sum_{i \in I_j} \mathbb{1}(y_i = c), \\ p(c|j) &= \frac{n_{jc}}{n_j}. \end{aligned} \tag{2.22}$$

Temos,

$$\hat{P}(Y = c | X \in \mathcal{X}_j) = p(c|j) \quad \text{e} \quad \gamma_j = c. \tag{2.23}$$

A estimativa do erro de previsão em $\tilde{T}_j$ será então $1 - p(c|j)$. Vamos verificar que esta estimativa

Algoritmo 1: CART

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Inicializar: $t = T_0$ nó raiz com todas as observações disponíveis em $\mathcal{D}$.
2. Verificar se $(t, \mathcal{D}_t)$ é nó terminal.
3. Se t não é nó terminal particionar t em $(t_e, \mathcal{D}_{te})$ e $(t_d, \mathcal{D}_{td})$.
4. Repetir passo 2 para $(t_e, \mathcal{D}_{te})$ e $(t_d, \mathcal{D}_{td})$ até que cada nó seja declarado terminal.

Resultado: $T_{\mathcal{D}}(\mathbf{x})$

minimiza o erro de treinamento, pela equação (2.11):

$$\begin{aligned} Erro(T) &= \sum_{j=1}^{J} p(j)(1 - p(\gamma_j|j)) \\ &= \sum_{j=1}^{J} \frac{n_j}{n}\left(1 - \frac{n_{\gamma_j}}{n_j}\right) = \frac{1}{n}\sum_{j=1}^{J}\left(n_j - n_{\gamma_j}\right) \\ &= \frac{1}{n}\sum_{j=1}^{J}\sum_{i \in I_j} \mathbb{1}(y_i \neq \gamma_j) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i \neq T(\mathbf{x}_i)) \\ &= \frac{1}{n}\sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbb{1}(y \neq T_{\mathcal{D}}(\mathbf{x})) = \frac{1}{n}\sum_{(\mathbf{x},y)\in\mathcal{D}} L(y, T_{\mathcal{D}}(\mathbf{x})) \\ &= \bar{E}(T_{\mathcal{D}},\mathcal{D}) = \widehat{Erro}_{treino}. \end{aligned} \tag{2.24}$$

2.4 Selecionando Partições

O critério para seleção de uma partição tem como base os valores observados $\mathbf{x} = (x_1, \ldots, x_p)'$ em que cada $x_j, j = 1, \ldots, p$, pode ser numérica ou categórica. Considere $\mathcal{Q}$ o conjunto de perguntas que podem ser utilizadas num partição. A seleção da partição compreende os seguintes itens:

1. Cada partição depende de uma *única* variável.
2. Se $x_j \in \mathbb{R}$, $\mathcal{Q}$ inclui as perguntas do tipo "$x_j \leq v$?", para $v \in (-\infty, +\infty)$.
3. Se x_j é categórica assumindo valores em $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, então $\mathcal{Q}$ inclui as perguntas do tipos "$x_j \in S$?", em que S é um subconjunto de $\mathcal{C}$.

Particionar um nó com estas regras gera uma quantidade finita de perguntas. Se x_j assume n_j valores numéricos distintos, teremos no máximo $n_j - 1$ perguntas do tipo "$x_j \leq v$"se tomarmos v igual ao ponto médio entre dois valores consecutivos de x_j. Se x_j for categórica, com K classes, as perguntas do tipo "$x_j \in S$"serão no máximo $2^{K-1} - 1$, pois $\{x_j \in S\}$ e $\{x_j \notin S\}$ geram as mesmas partições, apenas trocando a ordem dos nós.

Esta estrutura de perguntas faz com que os nós filhos t_e e t_d de um nó t interno tenham conteúdo mutuamente exclusivo, *i.e.*, $\{(\mathbf{x}, y) \in \mathcal{D}_{td}\} \cap \{(\mathbf{x},y) \in \mathcal{D}_{te}\} = \varnothing$.

Para construir uma árvore, começamos com um nó raiz com todas as observações e, para cada variável preditora, percorremos todas as perguntas possíveis e escolhemos, dentre todas as perguntas, qual gera a *melhor partição*, dividimos o nó e recomeçamos o processo em cada novo nó particionado

ou declaramos um nó terminal - cf. o Algoritmo 1. A *melhor partição* segue um critério que minimiza o erro de previsão conforme veremos a seguir.

Suponha γ_j previsor para as observações alocadas em um nó terminal $\tilde{T}_j$, $j = 1, \ldots, J$. Se tivermos uma árvore de regressão com a função de perda (2.7), teoricamente, precisamos minimizar

$$T(X) = \sum_{j=1}^{J} \sum_{i \in I_j} (y_i - \gamma_j)^2, \tag{2.25}$$

se a árvore for de classificação, conforme a função de perda (2.9), precisamos minimizar

$$T(X) = \sum_{j=1}^{J} \sum_{i \in I_j} P(y_i \neq \gamma_j). \tag{2.26}$$

Encontrar todas as regiões possíveis associadas aos nós terminais e escolher T^* que minimiza T é uma tarefa, geralmente, computacionalmente intratável (Hyafil e Rivest, 1976).

Por isso a metodologia CART utiliza uma abordagem de **divisão binária recursiva** (*recursive binary splitting*) que é de cima para baixo e gananciosa (*top-down and greedy*) (James *et al.*, 2017). É de cima para baixo, porque começa com um nó raiz, com todas as observações alocadas nele, e particiona um nó de forma gananciosa procurando entre todas as partições binárias possíveis, qual minimiza o erro de previsão, segundo algum critério, a fim de decidir como particionar. É recursiva porque o processo se repete a cada novo nó.

Ao invés de minimizar a árvore inteira de uma vez, definimos uma *medida de impureza* $i(t)$ para um dado nó t que avalia quão homogêneas são as observações da variável resposta alocadas nele, por exemplo, se y é categórica um nó puro é um nó que contém somente uma classe, um nó totalmente impuro é um nó que contém a mesma proporção de cada classe, no caso de y ser numérico um nó com todas as observações iguais seria um nó totalmente puro. A partir dessa medida realizamos a divisão de acordo com a partição cujo decréscimo de impureza é máximo.

Considere um dado nó t com n_t observações. Seja s uma partição de t e $i(t)$ a medida de impureza, o decréscimo de impureza que obtemos ao particionar t em t_e e t_d é dado por

$$\Delta I(s, t) = i(t) - p_e \times i(t_e) - p_d \times i(t_d), \tag{2.27}$$

em que $p_e = \frac{n_e}{n_t}$ e $p_d = \frac{n_d}{n_t}$, são as respectivas proporções de observações alocadas em t_e e t_d. Escolhemos a partição s^* que maximiza (2.27). A medida $i(t)$ deve ser definida de acordo com o tipo de valor que y assume - se categórico ou numérico.

2.4.1 Árvores de Regressão

Assumindo a perda quadrática, a função de impureza em t é dada por

$$i_R(t) = \frac{1}{n_t} \sum_{i \in I_t} (y_i - \gamma_t)^2, \tag{2.28}$$

em que $I_t = \{i | i \in \mathcal{D}_t\}$ e $\gamma_t = \bar{y}_t = \frac{1}{n_t} \sum_{i \in I_t} y_i$.

Na equação (2.17) vemos que $\gamma_t = \bar{y}_t$ minimiza $i_R(t)$. Desta forma notamos que a expressão de $i_R(t)$ em (2.28) é a variância das observações em t, *i.e.*, o valor de $s^\star$ é o que maximiza a redução da variância dentro da partição.

2.4.2 Árvores de Classificação

Para árvores de classificação, segundo Breiman *et al.* (1984), uma medida de impureza deve possuir três propriedades adequadas. Considere $y \in \{c_1, \ldots, c_K\}$. Uma função Φ é uma *função de*

impureza calculada em todas as tuplas $(p_1, \ldots, p_K)$, $p_k \geq 0, k = 1, \ldots, K$ e $\sum_k p_k = 1$, que possui as seguintes propriedades:

(i) Φ é máxima somente no ponto $(\frac{1}{K}, \ldots, \frac{1}{K})$,

(ii) Φ é mínima somente nos pontos $(1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 0, 1)$,

(iii) Φ é uma função simétrica de $(p_1, \ldots, p_K)$.

Sejam n_{tk} o número de observações da classe k em t e $p(k|t)$ a estimativa da probabilidade de observações da classe k em t conforme (2.22). Uma medida de impureza $i(t)$ para qualquer nó t é definida como

$$i(t) = \Phi(p(c_1|t), \ldots, p(c_K|t)). \tag{2.29}$$

As seguintes medidas de impureza são geralmente utilizadas:

Definição 2.8 (Taxa de Erro de Classificação)**.** A função de impureza com base na taxa de erro de classificação $i_R(t)$ é dada por

$$i_R(t) = 1 - p(c|t),$$

em que $c = \hat{y}_t$ é a classe majoritária em t (2.23) e minimiza o erro de treinamento (2.24).

Definição 2.9 (Índice de Gini)**.** O Índice de Gini é uma medida de impureza cuja origem está na teoria da informação (Sammut e Webb, 2017). O cálculo é dado por

$$i_G(t) = \sum_{k=1}^{K} p(c_k|t)(1 - p(c_k|t)).$$

Definição 2.10 (Entropia Cruzada)**.** O cálculo da Entropia Cruzada também tem sua origem na teoria da informação (Sammut e Webb, 2017). O cálculo é dado por

$$i_H(t) = -\sum_{k=1}^{K} p(c_k|t) \log_2 p(c_k|t).$$

Quando o problema é de classificação binária podemos facilmente visualizar as fórmulas dessas medidas de impureza. Considere p a proporção de observações da segunda classe, estas três medidas se tornam

$$\begin{aligned} i_R(t) &= 1 - max(p, 1 - p), \\ i_G(t) &= 2p(1 - p), \\ i_H(t) &= -p \log_2 p - (1 - p) \log_2 (1 - p). \end{aligned} \tag{2.30}$$

As três métricas são similares, no entanto, o índice de Gini e a entropia cruzada são diferenciáveis e passíveis de um processo de otimização. O índice de Gini e a entropia cruzada são mais sensíveis a mudanças nas proporções encontradas nos nós do que o erro de classificação conforme Figura 2.5. Num problema de classificação binária considere, por exemplo Hastie *et al.* (2009), um nó com 400 observações em cada classe, digamos $(400, 400)$, suponha que uma partição leve a dois nós $(300, 100)$ e $(100, 300)$, enquanto outra partição leve a $(200, 400)$ e $(200, 0)$. A segunda opção parece ser preferível mas, nos dois casos o erro de classificação é de 0,25, enquanto que o índice de Gini e a entropia cruzada são menores para a segunda partição. Por esse motivo ou o índice de Gini ou a entropia cruzada deve ser utilizada para a construção da árvore. Para realizar a poda qualquer uma das três métricas pode ser utilizada mas, tipicamente o erro de classificação é o escolhido (Hastie *et al.*, 2009). A Figura 2.5 apresenta um gráfico das métricas para classificação binária. A

entropia cruzada é a mais sensível à mudança nas proporções das classes, todas atingem o máximo quando a proporção é 0,5.

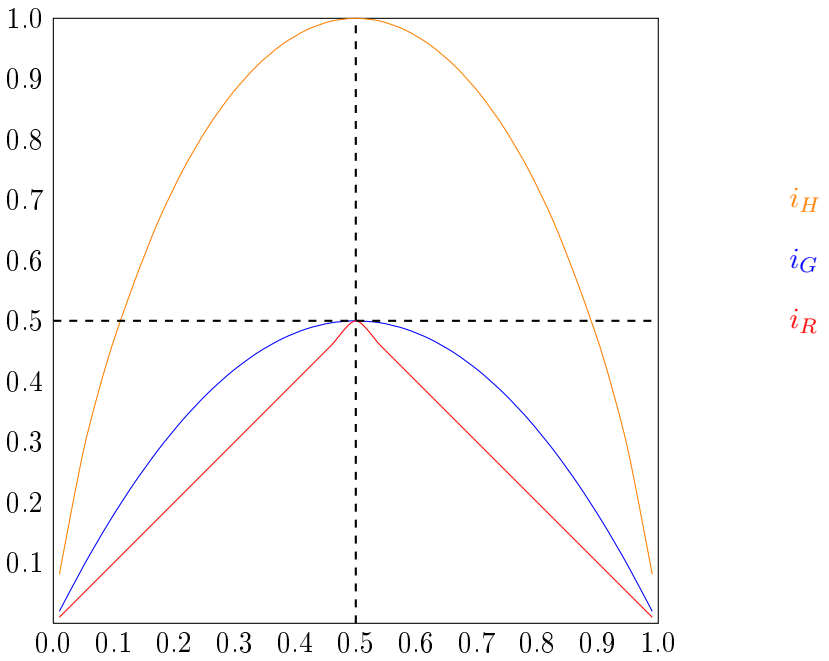


Figura 2.5: *Métricas de impureza para árvores de classificação com resposta binária*

Critério de Gini e a Árvore de Regressão

Quando a variável resposta y é binária, *i.e.*, $\mathcal{Y} = \{0, 1\}$, a construção de uma árvore de regressão usando perda quadrática ou uma árvore de classificação usando o critério de Gini são equivalentes. Considere $c_1 = 1$, $c_2 = 0$ e $\hat{y}_t = (c_1|t) = p$, vejamos

$$\begin{aligned} i_R(t) &= \frac{1}{n_t}\sum_{i\in I_t}(y_i - \hat{y}_t)^2 \\ &= \frac{1}{n_t}\sum_{i\in I_t}(y_i^2 - 2y_i\hat{y}_t + \hat{y}_t^2) \\ &= \hat{y}_t - 2\hat{y}_t^2 + \hat{y}_t^2 \\ &= \hat{y}_t(1 - y_t) \\ &= p(1-p) \\ &= \frac{1}{2}i_G(t). \end{aligned} \tag{2.31}$$

Codificando a classificação binária numericamente de maneira que $y \in \{0, 1\}$ é necessário somente usar o critério da perda quadrática, cf. (2.28), para resolver os problemas de regressão ou classificação binária.

2.5 Decidindo Quando Declarar um Nó Terminal

De posse de um critério para particionar um nó precisamos de um critério para determinar quando fixar um nó como terminal e parar de particionar. O primeiro critério sugerido é parar de

particionar se o nó tiver mais do que n_{min} observações, em que n_{min} é um número arbitrário. O segundo critério é fixar um limiar $\beta > 0$ e só particionar se $\Delta i(s^*, t) > \beta$.

O problema com a primeira abordagem é que não há uma fórmula para a escolha n_{min}, se n_{min} for *muito pequeno* podemos chegar a nós com poucas observações e alta previsibilidade para y, o que pode configurar um sobre-ajuste (*overfitting*) - a árvore se ajusta bem aos dados de $\mathcal{D}$ mas, tem pouco poder de generalização. Um valor de n_{min} *muito grande* pode levar a uma árvore pobre, com poucos nós, que representa pouco o fenômeno além dos dados da base de treino.

Já a segunda opção pode levar a uma parada prematura, já que o critério não garante que se outra variável com um decréscimo na impureza um pouco menor fosse selecionada, os nós seguintes seriam mais puros do que os obtidos pela variável selecionada.

Breiman *et al.* (1984) nota que a abordagem para se declarar um nó terminal deve ser redirecionada. A proposta é construir uma árvore até que cada nó tenha no máximo 5 observações - ou que as observações dentro do nó sejam todas iguais, e a partir desta árvore *máxima* realizar uma **poda** (*pruning*) até se chegar a uma árvore final.

Seja T_{max} a árvore máxima e $T \subseteq T_{max}$ uma subárvore que pode ser obtida podando-se T_{max}. A poda não se refere a simplesmente cortar a árvore, mas colapsar os nós internos abaixo de T_{max}. A questão é como realizar a poda, pois testar todas as subárvores para verificar qual produz o menor erro pode ser um processo inviável dado o número excessivamente grande de subárvores que podem existir. O **custo de complexidade da poda** (*cost complexity pruning*) é solução apresentada por Breiman *et al.* (1984) para resolver esse problema. Considere $i_j(T), j = 1, \ldots, J$, a medida de impureza do *j-ésimo* nó terminal da árvore T cujo número de observações alocadas é n_j, o custo de complexidade da poda é definido como

$$C_\alpha(T) = \sum_{j=1}^{J} n_j \times i_j(T) + \alpha J. \tag{2.32}$$

$C_\alpha(T)$ leva em conta a medida de impureza e o tamanho da árvore. O parâmetro $\alpha \geq 0$ governa a troca entre o erro de predição e o tamanho da árvore. Grandes valores de α resultam em árvores pequenas, baixos valores em árvores maiores, com $\alpha = 0$ obtemos a árvore sem poda T_{max}.

Breiman *et al.* (1984) mostrou que para cada valor de α corresponde uma única subárvore $T_\alpha \subseteq T_{max}$ cujo valor de $C_\alpha(T)$ é mínimo. Para encontrar α, o processo de poda é realizado a partir do colapso dos elos mais fracos (*weakest link pruning*): os nós com menor decréscimo de impureza são sucessivamente colapsados até chegar num único nó. Isto produz um sequência de subárvores que contém T_α. A estimativa de α pode ser encontrada usando validação cruzada (*cross-validation*) ou um conjunto de dados de teste.

2.6 Regularização

A discussão sobre quando declarar um nó terminal, que equivale a encontrar um tamanho de árvore que não leve a um sobre-ajuste, faz parte da discussão sobre *viés e variância*. Uma árvore com muitos nós tende a um baixo viés, *i.e.*, tem um erro de predição muito baixo na base de treino, mas uma estimativa do erro generalizado, percebido na base teste, relativamente alto. Uma árvore com sobre-ajuste tem pouco poder de generalização, e portanto alta variância.

O parâmetro α que tenta controlar o tamanho da árvore para evitar o sobre-ajuste é chamado de parâmetro de *regularização*. Regra geral os modelos de árvore tentam evitar o sobre-ajuste ao mesmo tempo que procuram obter uma baixa variabilidade.

2.7 Importância das Variáveis

Num estudo com muitas variáveis preditivas, raramente todas as variáveis influenciam a variável resposta com o mesmo peso. Geralmente, algumas poucas variáveis são importantes. Por esse motivo

é interessante entender o quão importante cada variável é para o modelo, Breiman *et al.* (1984) define uma medida de importância para as variáveis usadas na construção de uma árvore.

A medida de importância é baseada no conceito de partição substituta (*surragate split*). Considere a partição $s^\star$ a partição encontrada pelo algoritmo de construção da árvore em um dado nó t. Esta partição divide t em $t_e^\star$ à esquerda e $t_d^\star$ à direita. Sejam $I_e^\star = \{i|(\mathbf{x}_i, y_i) \in t_e^\star\}$ e $I_d^\star = \{i|(\mathbf{x}_i, y_i) \in t_d^\star\}$, as observações alocadas em t_e e t_d, respectivamente. Considere uma variável X_ℓ que gera uma partição s^ℓ no mesmo nó t e, os respectivos nós à esquerda e à direita, t_e^ℓ e t_d^ℓ . As observações alocadas em t_e^ℓ são dadas pelo conjunto $I_e^\ell = \{i|(\mathbf{x}_i, y_i) \in t_e^\ell\}$ e em t_d^ℓ pelo conjunto $I_d^\ell = \{i|(\mathbf{x}_i, y_i) \in t_d^\ell\}$.

Definição 2.11. conteúdo...

Se $I_e^\star = I_e^\ell$, e por consequência, $I_d^\star = I_d^\ell$, as duas partições $s^\star$ e s^ℓ são exatamente iguais e s^ℓ pode substituir $s^\star$ em t gerando exatamente os mesmos descendentes. Vamos definir

$$\begin{aligned} p_e^\ell(s^\star, s^\ell) &= \frac{|I_e^\star \cap I_e^\ell|}{|t|}, \\ p_d^\ell(s^\star, s^\ell) &= \frac{|I_d^\star \cap I_d^\ell|}{|t|}, \end{aligned} \tag{2.33}$$

as proporções coincidentes de observações enviadas aos nós filhos pelas duas partições, e a proporção total de observações coincidentes

$$p(s^\star, s^\ell) = p_e^\ell(s^\star, s^\ell) + p_d^\ell(s^\star, s^\ell). \tag{2.34}$$

Uma partição $\tilde{s}^\ell$ gerada pela variável X_ℓ é chamada partição substituta em t para $s^\star$ se

$$p(s^\star, \tilde{s}^\ell) = \max_{s^\ell} p(s^\star, s^\ell). \tag{2.35}$$

A partição s^ℓ de X_ℓ é a que mais se aproxima do comportamento de $s^\star$ em t. A importância de uma variável X_ℓ é definida como a soma dos decréscimos de impureza gerados pela substituição da partição $s^\star$ por $\tilde{s}^\ell$ em cada nó interno de uma árvore:

$$\mathcal{I}mp(X_\ell) = \sum_{j=1}^{J-1} \Delta I(\tilde{s}^\ell, t_j), \quad t_j \in T \setminus \tilde{T}. \tag{2.36}$$

Esta definição capta a seguinte sutileza: uma variável X_{ℓ_1} presente na árvore, pode gerar decréscimos de impureza muito próximos a uma outra variável X_{ℓ_2} que foi excluída, *i.e.*, X_{ℓ_2} tem seu efeito *mascarado* por X_{ℓ_1}. Uma árvore sem a presença de X_{ℓ_1} poderia gerar a partir de X_{ℓ_2} uma árvore quase tão boa quanto a primeira.

Geralmente, a medida de importância é apresentada de maneira relativa, *i.e.*,

$$\mathcal{I}(X_\ell) = 100 \times \frac{\mathcal{I}mp(X_\ell)}{\max_k \mathcal{I}mp(X_k)}. \tag{2.37}$$

Assim, a(s) variável(eis) mais importante(s) assume o valor 100 e as demais permanecem no intervalo de 0 a 100.

2.8 Vantagens do Modelo CART

Segundo Breiman *et al.* (1984) a estrutura de árvore é um procedimento iterativo e recursivo que requer um conjunto simples de regras para ser construída:

1. Um conjunto $\mathcal{Q}$ de perguntas;

2. Uma regra para selecionar a melhor partição a cada nó;

3. Um critério para escolher um tamanho de árvore adequado;

Estas características tornam essa abordagem poderosa e flexível para ser aplicada em problemas de classificação. Em particular:

1. Esta abordagem pode ser aplicada a qualquer estrutura de dados através de uma adequada formulação de um conjunto de perguntas $\mathcal{Q}$. Variáveis ordenadas ou categóricas são tratadas de forma simples e natural.

2. A classificação final tem uma forma simples que pode ser compactamente armazenada e que pode classificar novos dados de forma eficiente.

3. As árvores fazem um uso poderoso da informação condicional ao tratar relacionamentos não homogêneos. Por exemplo, a mesma variável pode ser selecionada em mais de um nó com valores de corte diferentes ou classes diferentes, de acordo com a natureza da variável ordenada ou categórica, respectivamente.

4. O método automaticamente realiza uma seleção *stepwise* das variáveis e reduz a complexidade. O processo de construção da árvore seleciona a cada passo a partição mais relevante e, neste sentido lembra o processo de *stepwise*, no qual a cada estágio procura extrair a informação mais importante no espaço em que está trabalhando.

5. Sem esforço adicional, fornece além da classificação, uma estimativa do erro de classificação para cada caso.

6. Em qualquer estrutura de dados padrão a construção da árvore é invariante sobre todas as transformações monótonas individualmente de qualquer variável ordenada. Por exemplo, $x'_j = x_j^3$ (x_j ao cubo) levará às mesmas partições e portanto, à mesma árvore final.

7. A abordagem é extremamente robusta com respeito a *outliers* ou pontos erradamente classificados em $\mathcal{D}$.

8. A saída do procedimento de construção da árvore é facilmente entendida e interpretável em relação à estrutura dos dados utilizada.

Além destas características, as árvores são a base sobre a qual modelos mais sofisticados são construídos (Figura 2.6). Estes modelos são explorados nos próximos capítulos: *Florestas Aleatórias* no Capítulo 3, *Gradiente Boosting* nos Capítulos 4, 5 e 6, e *Árvores Bayesianas* nos Capítulos 7 e 8.

2.9 Ilustração

A fim de ilustrar a construção de uma árvore CART vamos utilizar um exemplo fictício. A utilidade do exemplo é mostrar como alguns cálculos são realizados, não necessariamente vamos construir a árvore *completa*.

Árvores de Regressão

A Tabela 2.1 apresenta o conjunto de dados, $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^3, y_i \in \mathbb{R}, n = 8\}$, usados para ilustração de uma árvore de regressão. A variável x_3 é uma variável categórica com apenas duas classes, $x_3 \in \{0, 1\}$ de forma que $x_3 \leq 0{,}5 \rightarrow x_3 = 0$ e $x_3 > 0{,}5 \rightarrow x_3 = 1$.

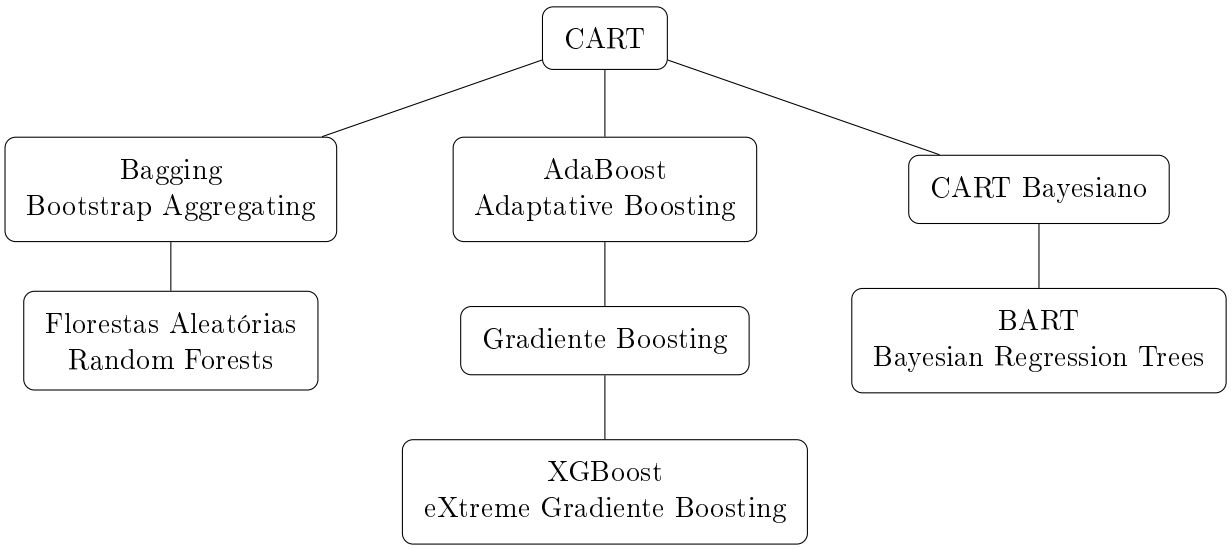


Figura 2.6: *Linhagens do modelo CART.*

Tabela 2.1: *Ilustração: CART Regressão - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
2	1,8	2,0	0	4,0
3	0,4	6,0	1	2,5
4	1,9	8,0	0	1,5
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0
7	4,0	5,0	1	0,8
8	3,5	4,5	0	0,4

Cálculo das partições

Começamos com o nó raiz t_0 no qual todas as observações em $\mathcal{D}$ estão alocadas. Calculamos o valor médio da variável resposta $\bar{y}_0$ e como medida de impureza utilizamos a variância, cf. (2.28). A predição para os valores na raiz é a média de todas observações, $\hat{y}_0 = \bar{y}_0$.

$$\begin{aligned} \bar{y}_0 &= (5{,}0 + 4{,}0 + \ldots + 0{,}4)/8 = 3{,}96, \\ i(t_0) &= [(5{,}0 - 3{,}96)^2 + \ldots]/8 = 9{,}87, \\ \hat{y}_0 &= \bar{y}_0 = 3{,}96, \end{aligned} \tag{2.38}$$

Para escolher a partição que tem o maior decréscimo de impureza $i(t_0)$ é preciso ordenar os valores únicos de cada variável e especificar os pontos de corte c igual à média entre dois pontos consecutivos. Por exemplo, na Tabela 2.2 na primeira coluna ordenamos a variável x_1 : $(0{,}4, 0{,}8, 1{,}8, 1{,}9, 2{,}5, 3{,}0, 3{,}5, 4{,}0)$, note que esta variável não tem valores repetidos, e calculamos primeiro ponte de corte $v = (0{,}4+0{,}8)/2 = 0{,}6$. O segundo ponto de corte é $v = (0{,}8+1{,}8)/2 = 1{,}30$. Procedemos assim sucessivamente até o último valor $v = (3{,}5 + 4{,}0)/2 = 3{,}75$.

Após o cálculo dos pontos de corte precisamos calcular qual deles proporciona o maior decréscimo de impureza. Para cada v particionamos o nó raiz em dois novos nós t_1 e t_2 e calculamos a impureza. Por exemplo, para $v = (1{,}8 + 1{,}9)/2 = 1{,}85$, definimos duas partições $x_1 \leq 1{,}85$ e $x_1 > 1{,}85$, e portanto, os dados a serem alocados em t_1 e t_2. Em t_1 alocamos os pares (x_1, y_1)

$$\mathcal{D}_1 = \{(0{,}4, 2{,}5), (0{,}8, 5{,}0), (1{,}8, 4{,}0)\} \tag{2.39}$$

e em t_2 alocamos

$$\mathcal{D}_2 = \{(1{,}9, 1{,}5), (2{,}5, 9{,}5), (3{,}0, 8{,}0), (3{,}5, 0{,}4), (4{,}0, 0{,}8)\}. \tag{2.40}$$

Em cada conjunto calculamos a impureza $i(t_1)$ e $i(t_2)$. Primeiro calculamos as médias:

$$\begin{aligned} \bar{y}_1 &= \frac{2{,}5 + 5{,}0 + 4{,}0}{3} = 3{,}83, \\ \bar{y}_2 &= \frac{1{,}5 + 9{,}5 + 8{,}0 + 0{,}4 + 0{,}8}{5} = 4{,}04. \end{aligned} \tag{2.41}$$

Agora calculamos as impurezas:

$$\begin{aligned} i(t_1) &= \left[(2{,}5 - 3{,}83)^2 + \ldots + (4{,}0 - 3{,}83)^2\right]/3 = 1{,}06, \\ i(t_2) &= \left[(1{,}5 - 4{,}04)^2 + \ldots + (0{,}8 - 4{,}04)^2\right]/5 = 15{,}14. \end{aligned} \tag{2.42}$$

O decréscimo de impureza para a partição $x_1 \leq 1.85$ é

$$\begin{aligned} \Delta I(x_1 \leq 1{,}85, t_0) &= i(t_0) - \frac{3}{8} \times i(t_1) - \frac{5}{8} \times i(t_2) \\ &= 9{,}87 - \frac{3}{8} \times 1{,}06 - \frac{5}{8} \times 15{,}14 \\ &= 0.01. \end{aligned} \tag{2.43}$$

As Tabelas 2.2, 2.3 e 2.4, detalham os cálculos das possíveis partições para as variáveis x_1, x_2 e x_3, respectivamente. A árvore é particionada à esquerda na folha t_1 e à direita na folha t_2. Em destaque em cada tabela a partição cujo decréscimo de impureza é maior.

Tabela 2.2: *Ilustração: CART Regressão - Partições x_1.*

x_1	0,4	0,8	1,8	1,9	2,5	3,0	3,5	4,0
y_i	2,5	5,0	4,0	1,5	9,5	8,0	0,4	0,8
v		0,60	1,30	1,85	2,20	2,75	3,25	3,75
$i(t_1)$		0,00	1,56	1,06	1,81	7,70	8,12	9,64
$i(t_2)$		10,93	12,62	15,14	16,91	12,20	0,04	0,00
$\Delta I(x_1 \leq v, t_0)$		0,31	0,02	0,01	0,51	0,48	3,77	1,43

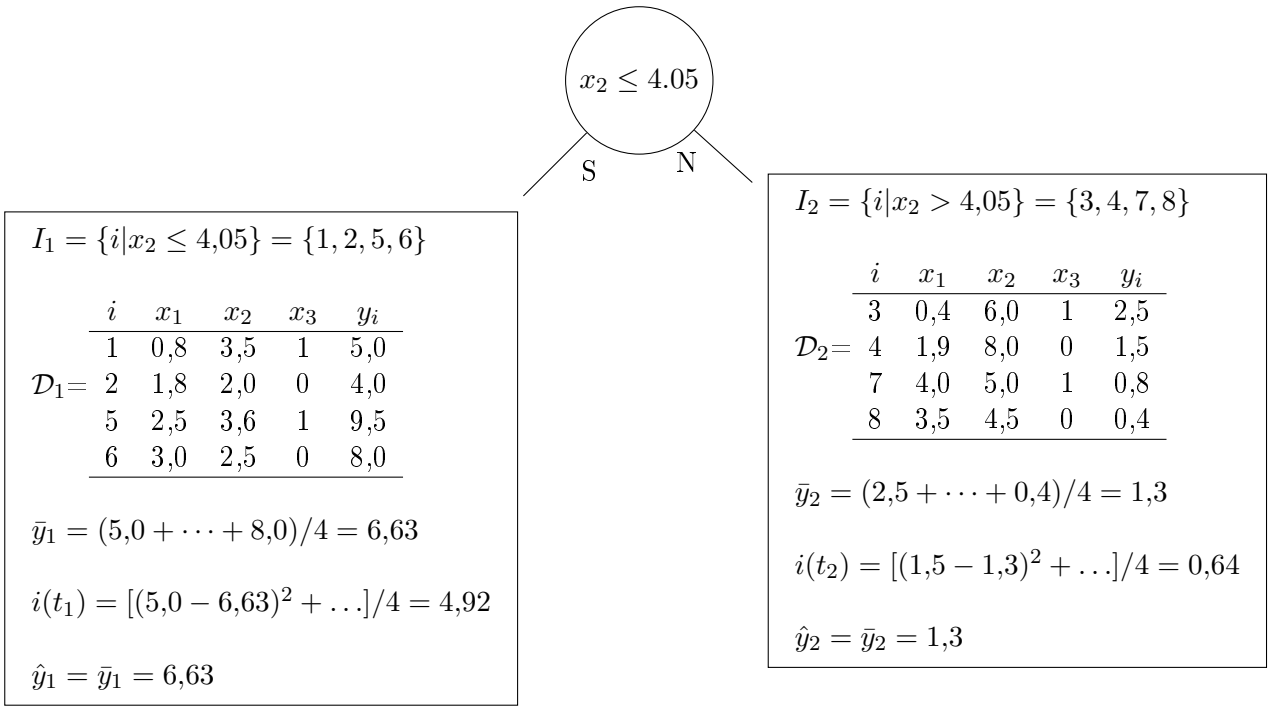
Tabela 2.3: *Ilustração: CART Regressão - Partições x_2.*

x_2	2,0	2,5	3,5	3,6	4,5	5,0	6,0	8,0
y_i	4,0	8,0	5,0	9,5	0,4	0,8	2,5	1,5
v		2,25	3,00	3,55	4,05	4,75	5,50	7,00
$i(t_1)$		0,00	4,00	2,89	4,92	10,14	11,36	10,29
$i(t_2)$		11,28	9,98	11,27	0,64	0,49	0,25	0,00
$\Delta I(x_2 \leq v, t_0)$		0,00	1,38	1,74	7,09	3,35	1,28	0,87

Tabela 2.4: *Ilustração: CART Regressão - Partições x_3.*

x_3	0	0	0	0	1	1	1	1
y_i	4	1,5	8,0	0,4	5,0	2,5	9,5	0,8
v					0,50			
$i(t_1)$					8,53			
$i(t_2)$					10,73			
$\Delta I(x_3 \leq v, t_0)$					0,24			

A partição selecionada é que tem o maior decréscimo de impureza: $x_2 \leq 4{,}05$, cujo valor é 7,09, Tabela 2.3. O resultado do particionamento do nó raiz em t_1 e t_2 gera uma árvore conforme a Figura 2.7. Note como as observações y_i's são mais homogêneas dentro de cada folha.



i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
2	1,8	2,0	0	4,0
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0

i	x_1	x_2	x_3	y_i
3	0,4	6,0	1	2,5
4	1,9	8,0	0	1,5
7	4,0	5,0	1	0,8
8	3,5	4,5	0	0,4

Figura 2.7: *Ilustração: CART Regressão - Particionamento do nó raiz.*

O erro no conjunto de dados de treino é

$$Erro_{treino} = \frac{4 \times 4{,}92 + 4 \times 0{,}64}{8} = 2{,}78. \tag{2.44}$$

Note que o valor da impureza inicial $i(t_0) = 9{,}87$, expressão (2.38), subtraído do erro 2,78, expressão (2.44), é o valor obtido no decréscimo de impureza 7,09, em destaque na Tabela 2.3:

$$\begin{aligned} i(t_0) - Erro_{treino} &= \Delta(x_2 \leq 4{,}05, t_0), \text{ isto é,} \\ 9{,}78 - 2{,}78 &= 7{,}09. \end{aligned} \tag{2.45}$$

O processo de construção é recursivo e deve ser repetido para t_1 e t_2 e assim sucessivamente.

Árvores de Classificação

Vamos modificar a Tabela 2.1 para exemplificar o cálculo de árvore de classificação, a Tabela 2.5 contém os dados, $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^3, y_i \in \{0,1\}, n = 8\}$. O Índice de Gini, cf. (2.9), será utilizado como medida de impureza. A predição será a classe majoritária.

Tabela 2.5: *Ilustração: CART Classificação - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
2	1,8	2,0	0	1
3	0,4	6,0	1	0
4	1,9	8,0	0	0
5	2,5	3,6	1	0
6	3,0	2,5	0	1
7	4,0	5,0	1	1
8	3,5	4,5	0	1

Cálculo das partições

Começando com o nó raiz t_0 com todas as observações em $\mathcal{D}$, temos

$$\begin{aligned} p(1|t_0) &= p_0 = \frac{5}{8}, \\ i(t_0) &= 2p_0(1-p_0) = 2 \cdot \frac{5}{8} \cdot \left(1 - \frac{5}{8}\right) = \frac{15}{32}, \\ \hat{y}_0 &= 1. \end{aligned} \tag{2.46}$$

Seguindo o mesmo roteiro apresentado na Ilustração - Árvores de Regressão. Após ordenar os valores da variável x_1: $(0{,}4, 0{,}8, 1{,}8, 1{,}9, 2{,}5, 3{,}0, 3{,}5, 4{,}0)$, Tabela 2.5, os valores de ponto de corte v são construídos tomando-se o ponto médio entre os valores sucessivos: por exemplo, calculamos primeiro ponte de corte $v = (0{,}4 + 0{,}8)/2 = 0{,}6$, o segundo ponto de corte $v = (0{,}8 + 1{,}8)/2 = 1{,}30$, e assim sucessivamente até o último valor $v = (3{,}5 + 4{,}0)/2 = 3{,}75$.

Cada ponto de corte v define uma partição do nó raiz em dois novos nós t_1 e t_2. Para as amostras alocadas em cada nó calculamos a impureza com base no critério de Gini (2.9). Por exemplo, para $v = (1{,}8 + 1{,}9)/2 = 1{,}85$, definimos duas partições $x_1 \leq 1{,}85$ e $x_1 > 1{,}85$, e portanto, os dados a serem alocados em t_1 e t_2. Em t_1 alocamos os pares (x_1, y_1)

$$\mathcal{D}_1 = \{(0{,}4, 0), (0{,}8, 1), (1{,}8, 1)\} \tag{2.47}$$

e em t_2 alocamos

$$\mathcal{D}_2 = \{(1{,}9, 0), (2{,}5, 0), (3{,}0, 1), (3{,}5, 1), (4{,}0, 1)\}. \tag{2.48}$$

Em cada conjunto calculamos a impureza $i(t_1)$ e $i(t_2)$. Primeiro calculamos a proporção da classe 1 em cada nó:

$$\begin{aligned} p(1|t_1) &= p_1 = \frac{2}{3}, \\ p(1|t_2) &= p_2 = \frac{3}{5}. \end{aligned} \tag{2.49}$$

Agora calculamos as impurezas:

$$\begin{aligned} i(t_1) &= 2p_1(1-p_1) = 2 \cdot \frac{2}{3} \cdot \left(1 - \frac{2}{3}\right) = \frac{4}{9}, \\ i(t_2) &= 2p_2(1-p_2) = 2 \cdot \frac{3}{5} \cdot \left(1 - \frac{3}{5}\right) = \frac{12}{25}. \end{aligned} \tag{2.50}$$

O decréscimo de impureza para a partição $x_1 \leq 1{,}85$ é

$$\begin{aligned}\Delta I(x_1 \leq 1.85, t_0) &= i(t_0) - \frac{3}{8} \times i(t_1) - \frac{5}{8} \times i(t_2) \\ &= \frac{15}{32} - \frac{3}{8} \times \frac{4}{9} - \frac{5}{8} \times \frac{12}{25} \\ &= \frac{1}{480} \approx 0{,}00.\end{aligned} \tag{2.51}$$

As Tabelas 2.6, 2.7 e 2.8, detalham os cálculos das possíveis partições para as variáveis x_1, x_2 e x_3, respectivamente. Em destaque em cada tabela a partição cujo decréscimo de impureza é maior.

Tabela 2.6: *Ilustração: CART Classificação - Partições x_1.*

x_1	0,4	0,8	1,8	1,9	2,5	3,0	3,5	4,0
y_i	0	1	1	0	0	1	1	1
c		0,60	1,30	1,85	2,20	2,75	3,25	3,75
$p(1\|t_1) = p_1$		0,00	0,50	0,67	0,50	0,40	0,50	0,57
$p(1\|t_2) = p_2$		0,71	0,67	0,60	0,75	1,00	1,00	1,00
$i(t_1)$		0,00	0,50	0,44	0,50	0,48	0,50	0,49
$i(t_2)$		0,41	0,44	0,48	0,38	0,00	0,00	0,00
$\Delta I(x_1 \leq c, t_0)$		0,11	0,01	0,00	0,03	0,17	0,09	0,04

Tabela 2.7: *Ilustração: CART Classificação - Partições x_2.*

x_2	2,0	2,5	3,5	3,6	4,5	5,0	6,0	8,0
y_i	1	1	1	0	1	1	0	0
c		2,25	3,00	3,55	4,05	4,75	5,50	7,00
$p(1\|t_1) = p_1$		1,00	1,00	1,00	0,75	0,80	0,83	0,71
$p(1\|t_2) = p_2$		0,57	0,50	0,40	0,50	0,33	0,00	0,00
$i(t_1)$		0,00	0,00	0,00	0,38	0,32	0,28	0,41
$i(t_2)$		0,49	0,50	0,48	0,50	0,44	0,00	0,00
$\Delta I(x_2 \leq c, t_0)$		0,04	0,09	0,17	0,03	0,10	0,26	0,11

Tabela 2.8: *Ilustração: CART Classificação - Partições x_3.*

x_3	0	0	0	0	1	1	1	1
y_i	1	0	1	1	1	0	0	1
c					0,50			
$p(1\|t_1) = p_1$					0,75			
$p(1\|t_2) = p_1$					0,50			
$i(t_1)$					0,38			
$i(t_2)$					0,50			
$\Delta I(x_3 \leq c, t_0)$					0,03			

A melhor partição é $x_2 \leq v = (5{,}0 + 6{,}0)/2 = 5{,}50$, Tabela 2.7. O resultado do particionamento do nó raiz em t_1 e t_2 gera uma árvore conforme a Figura 2.8. Note como as observações y_i's são mais homogêneas dentro de cada folha.

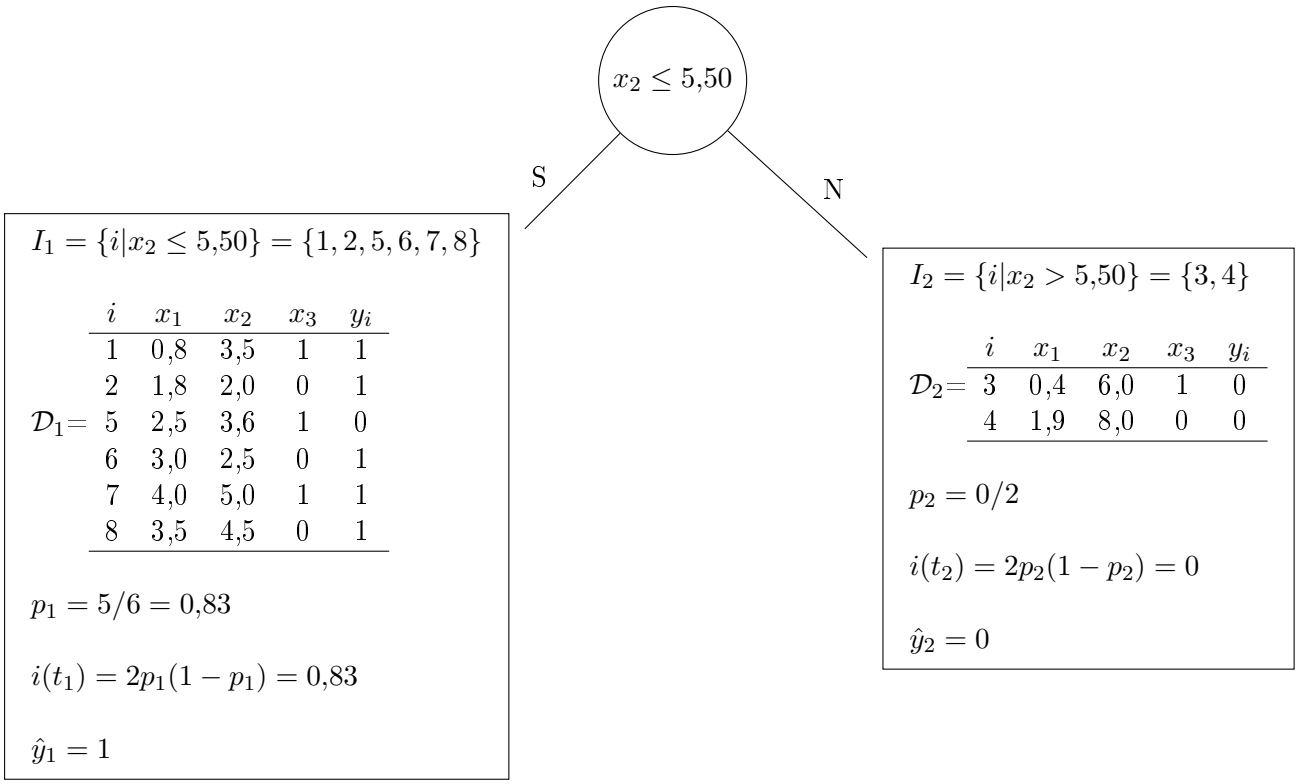


Figura 2.8: *Ilustração: CART Classificação - Particionamento do nó raiz.*

Apenas duas observações foram classificadas erradas $i = 5$ em $n = 8$. O erro no conjunto de dados de treino é

$$Erro_{treino} = \frac{1}{8} = 0{,}13, \tag{2.52}$$

que representa uma diminuição de $0{,}37 - 0{,}13 = 0{,}24$ pontos.

3 *Random Forests*

Breiman *et al.* (1984) notam que o procedimento CART gera árvores instáveis. Relações entre as variáveis preditoras e a variável reposta tornam-se mais complexas quando adicionamos possíveis relações entre as próprias variáveis preditoras. Dois grupos de variáveis distintas podem fornecer a mesma, ou quase a mesma informação, em relação à variável resposta. Como consequência, diferentes grupos de variáveis podem produzir num dado nó quase o mesmo decréscimo na impureza. Uma vez que os dados sempre contém algum ruído, a escolha das partições se dá quase de maneira aleatória. Duas partições com decréscimo de impureza parecidos podem levar a um desenvolvimento da árvore em estruturas completamente diferentes.

Breiman (1996a) define como instável um procedimento no qual pequenas alterações no conjunto de dados utilizados geram grandes alterações no erro de predição - este artigo apresenta resultados para árvores de regressão e portanto, o erro de predição está associado à minimização da soma de quadrados. No entanto, em Breiman (1996b) os resultados são aplicáveis também às árvores de classificação.

Para solucionar o problema de instabilidade, Breiman (1996b) propõe aplicar técnica de *bagging*, acrônimo de **b**oostrap **agg**regat**ing**, para aperfeiçoar o CART. A teoria de bootstrap segue a proposta de Efron e Tibshirani (1993). O procedimento bootstrap consiste em replicar amostras com reposição dentro de um mesmo conjunto de dados a fim de realizar alguma inferência sobre uma estatística de interesse, as amostras têm o mesmo tamanho que o conjunto do qual foram tiradas.

O *bagging* utiliza um conjunto de árvores CART, cada uma gerada a partir de uma amostra bootstrap, para produzir a predição final, *e.g.*, a predição final, para árvores de regressão, é simplesmente a média da predição de cada árvore para o mesmo dado de entrada.

O *bagging* é um dos métodos mais simples de uma técnica mais geral conhecida como *ensemble*. Estudos subsequentes do próprio Breiman (Breiman, 2001) e outros listados em Gilles (2014) evoluíram a ideia levando em consideração não somente amostras aleatórias bootstrap, mas também subamostras aleatórias das varáveis preditoras na construção dos nós em cada árvore. Esse tipo de ensemble, quando consideramos as árvores de decisão CART, é conhecido pelo nome de *random forest*.

As *rando forests* apresentam um desempenho consideravelmente superior ao modelo de uma única árvore CART ou ao *bagging*. Para mostrar essa superioridade, vamos primeiro apresentar a técnica *bagging*, depois mostraremos que para árvores de regressão, utilizando perda quadrática, como o erro generalizado esperado pode ser decomposto aditivamente em três termos: viés, variância e erro inerente aos dados. Essa decomposição é conhecida como *decomposição do erro em víes e variância* e permite entender por que os ensembles são superiores. Uma decomposição, não tão precisa, será feita também para as árvores de classificação.

3.1 *Bagging*

Considere um conjunto de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$. Vamos gerar M amostras *com reposição* de $\mathcal{D}$ formando um conjunto de réplicas $\{\mathcal{D}_1, \ldots, \mathcal{D}_M\}$. Para cada instância $\mathcal{D}_m, m = 1, \ldots, M$, uma árvore CART T_m correspondente é construída formando uma sequência de árvores $\{T_1, \ldots, T_M\}$.

Para árvores de regressão considere $\mathcal{Y}$ numérico. A predição final para um vetor de entrada $\mathbf{x}$

genérico é dada pela média dos previsores

$$T(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} T_m(\mathbf{x}). \tag{3.1}$$

Para árvores de classificação considere a variável resposta y assumindo valores em K classes, $y \in \{c_1, \ldots, c_K\}$. Seja $N_k(\mathbf{x})$ o número de vezes que $\mathbf{x}$ é predito na classe c_k

$$N_k(\mathbf{x}) = \sum_{m=1}^{M} \mathbb{1}(T_m(\mathbf{x}) = c_k), \tag{3.2}$$

então,

$$T(\mathbf{x}) = \arg\max_{c_k} N_k(\mathbf{x}). \tag{3.3}$$

.

$T(\mathbf{x})$ em (3.3) é chamado de *voto majoritário*, Breiman (1996b) demonstra que os resultados preditivos tanto para regressão quanto para classificação são em geral superiores, fornecendo em média menores erros de previsão, quando bagging em contrapartida a uma única árvore CART. Justificativas teóricas e práticas são apresentadas no referido artigo.

É interessante notar ao usar o procedimento de amostragem bootstrap uma observação tem aproximadamente 37% de não ser selecionada na construção de uma árvore. Podemos identificar esse fato notando que a chance de uma observação não ser selecionada em n amostras selecionadas de maneira independente e com a mesma probabilidade é

$$\left(1 - \frac{1}{n}\right)^n \xrightarrow[n\to\infty]{} e^{-1} \approx 0.37. \tag{3.4}$$

Isto significa que para conjuntos de dados com relativamente poucas observações o resultado ao aplicar *bagging* pode ficar comprometido, uma vez que informações importantes para caracterização do modelo podem estar contidas nas observações não selecionadas.

3.2 Decomposição Viés-Variância

Esta Secção foi desenvolvida em detalhes em Gilles (2014). O erro esperado de predição, ou erro generalizado (2.2) pode ser decomposto numa soma que inclui um ruído, o viés e a variância. A análise destes termos leva a um entendimento do motivo pelo qual *bagging* e as *random forests* apresentam desempenho melhor do que o de uma única árvore.

Na Secção 2.2 definimos o erro generalizado, ou erro esperado de predição, de $T_{\mathcal{D}}$ com relação à função de perda L,

$$Erro(T_{\mathcal{D}}) = \mathbb{E}_{X,Y}\{L(Y, T_{\mathcal{D}}(\mathbf{x}))\}. \tag{3.5}$$

De maneira análoga, o erro esperado de $T_{\mathcal{D}}$ em $X = \mathbf{x}$ por ser escrito como

$$Erro(T_{\mathcal{D}}(\mathbf{x})) = \mathbb{E}_{Y|X=\mathbf{x}}\{L(Y, T_{\mathcal{D}}(\mathbf{x}))\}. \tag{3.6}$$

Os cálculos de decomposição serão realizados para a função de perda quadrática para o modelo de regressão. Para o modelo de classificação usaremos a perda binária $0-1$.

3.2.1 Modelo de Bayes

O modelo de Bayes fornece um limite mínimo que o erro generalizado esperado pode atingir. É o ruído natural presente nos dados que provém da distribuição de probabilidade - em geral desconhecida - a partir da qual os dados foram gerados. Teoricamente o erro esperado de um modelo qualquer pode ser comparado ao erro do modelo de Bayes, o que nos permite avaliar o

quão bem um modelo performa: quanto mais próximo do modelo de Bayes melhor o desempenho do modelo.

Embora esteja longe de ser simples a derivação do modelo de Bayes, a não ser em dados simulados, é possível utilizá-lo na decomposição do erro em viés-variância, cujo valor mesmo teórico permite um entendimento mais aprofundado de como o erro pode ser minimizado ao gerarmos um modelo.

Se a distribuição $P(X,Y)$ fosse conhecida, teoricamente seria possível derivar analiticamente uma árvore $T_{\mathcal{B}}$ independente do conjunto de aprendizado $\mathcal{D}$ que minimizaria o erro (3.5). Usando esta árvore, $T_{\mathcal{B}}$, a expressão do erro seria escrita na forma

$$\mathbb{E}_{X,Y}\{L(Y, T_{\mathcal{B}}(\mathbf{x}))\}, \tag{3.7}$$

condicionando em X, podemos escrever

$$\mathbb{E}_X\{\mathbb{E}_{Y|X}\{L(Y, T_{\mathcal{B}}(X))\}\}. \tag{3.8}$$

A árvore que minimiza a expressão interna de (3.8) condicionando em $\mathbf{x}$ é o *modelo de Bayes*

$$T_{\mathcal{B}}(\mathbf{x}) = \arg\min_{y \in calY} \mathbb{E}_{Y|X=\mathbf{x}}\{L(Y, y)\}. \tag{3.9}$$

O erro generalizado $Erro(T_{\mathcal{B}})$ é conhecido como *erro residual*. Esse erro representa o erro mínimo que qualquer procedimento de aprendizado de árvore pode realizar: $Erro(T_{\mathcal{B}}) \leq Erro(T_{\mathcal{D}}), \forall \mathcal{D}$. Esse erro é irredutível e se deve puramente a variabilidade presente nos dados. Serve de referência segundo a qual as árvores podem ser comparadas.

Para regressão usando a perda quadrática, temos

$$\begin{aligned} T_{\mathcal{B}}(\mathbf{x}) &= \arg\min_{y \in \mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\{(Y-y)^2\} \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\{Y\}, \end{aligned} \tag{3.10}$$

i.e., o melhor preditor ponto a ponto é o valor médio de Y no ponto $X = \mathbf{x}$. No caso de classificação, perda $0-1$, com $y \in \mathcal{C} = \{c_1, \ldots, c_K\}$ o modelo de Bayes é

$$\begin{aligned} T_{\mathcal{B}}(\mathbf{x}) &= \arg\min_{y \in \mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\{\mathbb{1}(Y \neq y)\} \\ &= \arg\min_{y \in \mathcal{Y}} P(Y \neq y | X = \mathbf{x}) \\ &= \arg\max_{y \in \mathcal{Y}} P(Y = y | X = \mathbf{x}), \end{aligned} \tag{3.11}$$

i.e., o melhor preditor é sistematicamente é escolher a classe mais provável em $\mathcal{C}$ no ponto $X = \mathbf{x}$.

Na prática $P(X,Y)$ não é conhecida e portanto o modelo de Bayes não pode ser derivado. Entretanto, em modelos simulados podemos estudar o comportamento de diversas estratégias de otimização a fim de conhecer melhor os algoritmos.

3.2.2 Árvores de Regressão

Assumindo perda quadrática vamos re-escrever a expressão (3.6) em relação ao modelo de Bayes:

$$\begin{aligned} Erro(T_{\mathcal{D}}(\mathbf{x})) &= \mathbb{E}_{Y|X=\mathbf{x}}\{(Y - T_{\mathcal{D}}(\mathbf{x}))^2\} \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\{(Y - T_{\mathcal{B}}(\mathbf{x}) + T_{\mathcal{B}}(\mathbf{x}) - T_{\mathcal{D}}(\mathbf{x}))^2\} \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\{(Y - T_{\mathcal{B}}(\mathbf{x}))^2\} + \mathbb{E}_{Y|X=\mathbf{x}}\{(T_{\mathcal{B}}(\mathbf{x}) - T_{\mathcal{D}}(\mathbf{x}))^2\} \\ &\quad + \underbrace{2\mathbb{E}_{Y|X=\mathbf{x}}\{(Y - T_{\mathcal{B}}(\mathbf{x}))(T_{\mathcal{B}}(\mathbf{x}) - T_{\mathcal{D}}(\mathbf{x}))\}}_{=0} \\ &= Erro(T_{\mathcal{B}}(\mathbf{x})) + (T_{\mathcal{B}}(\mathbf{x}) - T_{\mathcal{D}}(\mathbf{x}))^2. \end{aligned} \tag{3.12}$$

O primeiro termo é o erro irredutível e o segundo mede a discrepância entre o modelo $T_{\mathcal{D}}$ e o modelo de Bayes $T_{\mathcal{B}}$. Quanto melhor o modelo, mais próximo do erro residual, mais próximo $T_{\mathcal{D}}$ está de uma solução ótima.

Agora vamos assumir que o próprio conjunto de aprendizado $\mathcal{D}$ é uma variável aleatória no espaço amostral Ω, o procedimento $T_{\mathcal{D}}$ continua determinístico. Vamos re-escrever a discrepância entre o modelo de Bayes e o modelo em termos da média esperada de predição $\mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\}$ considerando todos os modelos em que $|\mathcal{D}| = n$:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}\{(T_{\mathcal{B}}(\mathbf{x}) - T_{\mathcal{D}}(\mathbf{x}))^2\} &= \mathbb{E}_{\mathcal{D}}\{(T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\} + \mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\} - T_{\mathcal{D}}(\mathbf{x}))^2\} \\ &= \mathbb{E}_{\mathcal{D}}\{(T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\})^2 +\} + \mathbb{E}_{\mathcal{D}}\{(\mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\} - T_{\mathcal{D}}(\mathbf{x}))^2\} \\ &\quad + \underbrace{2\mathbb{E}_{\mathcal{D}}\{(T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\})(\mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\} - T_{\mathcal{D}}(\mathbf{x}))\}}_{=0} \\ &= \underbrace{(T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\})^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\{(\mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\} - T_{\mathcal{D}}(\mathbf{x}))^2\}}_{\text{variância}}. \end{aligned} \tag{3.13}$$

De (3.12) e (3.13) o erro generalizado esperado $\mathbb{E}_{\mathcal{D}}\{Erro(T_{\mathcal{D}}(\mathbf{x}))\}$ em $X = \mathbf{x}$ pode ser decomposto aditivamente conforme a seguinte expressão

$$\mathbb{E}_{\mathcal{D}}\{Erro(T_{\mathcal{D}}(\mathbf{x}))\} = ruído(\mathbf{x}) + viés^2(\mathbf{x}) + var(\mathbf{x}), \tag{3.14}$$

em que

$$\begin{aligned} ruído(\mathbf{x}) &= Erro(T_{\mathcal{B}}(\mathbf{x})), \\ viés^2(\mathbf{x}) &= (T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\})^2, \\ var(\mathbf{x}) &= \mathbb{E}_{\mathcal{D}}\{(\mathbb{E}_{\mathcal{D}}\{T_{\mathcal{D}}(\mathbf{x})\} - T_{\mathcal{D}}(\mathbf{x}))^2\}. \end{aligned} \tag{3.15}$$

Essa decomposição sugere que podemos diminuir o erro generalizado esperado principalmente controlando a variância, especialmente de procedimentos como o CART que apresentam instabilidade e, portanto, alta variabilidade (Breiman, 1996a,b). Essa diminuição do erro pode ser alcançada através da aleatorização e composição de várias árvores para produzir um resultado final. O bagging, Secção 3.1, é uma técnica cuja proposta é diminuir o erro selecionando uma amostra bootstrap para cada árvore gerada e ao final compor um resultado, no caso da regressão o resultado final é a média da predição de cada árvore.

As *random forests*, Secção 3.4, introduzem novas abordagens para aleatorização e composição de modelos. As propostas sugerem que aleatorizar tanto a escolha do conjunto de observações quanto as variáveis e os pontos de corte para cada árvore induzem a um resultado preditivo com menor erro.

3.2.3 Árvores de Classificação

A decomposição do erro generalizado esperado de classificação não é tão direta quanto a decomposição do erro esperado da regressão cf. Secção 3.2.2. Existem algumas opções para decomposição (Geurts, 2010). Vamos utilizar a que faz o câmbio entre a classe majoritária e as probabilidades preditas para cada classe, *i.e.*, calcular

$$\hat{P}_{\mathcal{D}}(Y = c|X = \mathbf{x}) \tag{3.16}$$

e, então derivar a regra de classificação

$$T_{\mathcal{D}}(\mathbf{x}) = \arg\max_{c \in \mathcal{Y}} \hat{P}_{\mathcal{D}}(Y = c|X = \mathbf{x}). \tag{3.17}$$

Desta forma podemos estudar a decomposição viés-variância para o erro esperado de classificação. O seguinte resultado de Gilles (2014) reproduz a solução de Friedman (1997) que faz uma conexão explícita entre o erro de Bayes e o erro de classificação:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\{\mathbb{E}_{Y|X=\mathbf{x}}\{\mathbb{1}(T_{\mathcal{D}}(\mathbf{x}) \neq Y)\}\} \\
&= P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq Y) \\
&= 1 - P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) = Y) \\
&= 1 - P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) = T_{\mathcal{B}}(\mathbf{x}))P(T_{\mathcal{B}}(\mathbf{x}) = Y) \\
&\quad - P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq T_{\mathcal{B}}(\mathbf{x}))P(T_{\mathcal{B}}(\mathbf{x}) \neq Y) \\
&= P(T_{\mathcal{B}}(\mathbf{x}) \neq Y) + P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq T_{\mathcal{B}}(\mathbf{x})) - 2P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq T_{\mathcal{B}}(\mathbf{x}))P(T_{\mathcal{B}}(\mathbf{x}) \neq Y) \\
&= P(T_{\mathcal{B}}(\mathbf{x}) \neq Y) + P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq T_{\mathcal{B}}(\mathbf{x}))(2P(T_{\mathcal{B}}(\mathbf{x}) = Y) - 1)
\end{aligned} \tag{3.18}$$

O primeiro termo de (3.18) é o erro irredutível do modelo de Bayes. O termo $P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq T_{\mathcal{B}}(\mathbf{x}))$ é a probabilidade do modelo produzir um resultado diferente do modelo de Bayes. Isto ocorre quando $P_{\mathcal{D}}(T_{\mathcal{D}}(\mathbf{x}) \neq T_{\mathcal{B}}(\mathbf{x})) = P_{\mathcal{D}}(\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x})) < 0{,}5)$. Assumindo a aproximação normal para a estimativa $\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))$, podemos calcular

$$P_{\mathcal{D}}(\hat{P}(Y = T_{\mathcal{B}}(\mathbf{x})) < 0{,}5) = \Phi\left(\frac{0{,}5 - \mathbb{E}_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\}}{\sqrt{\mathbb{V}ar_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\}}}\right), \tag{3.19}$$

em que $\Phi(\cdot)$ é função cumulativa de distribuição da normal padrão. O erro generalizado esperado $\mathbb{E}_{\mathcal{D}}\{Erro(T_{\mathcal{D}}(\mathbf{x}))\}$ em $X = \mathbf{x}$, considerando a perda $0-1$ para classificação binária, pode ser decomposto conforme

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\{Erro(T_{\mathcal{D}}(\mathbf{x}))\} &= P(T_{\mathcal{B}}(\mathbf{x}) \neq Y) \\
&\quad + \Phi\left(\frac{0{,}5 - \mathbb{E}_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\}}{\sqrt{\mathbb{V}ar_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\}}}\right)(2P(T_{\mathcal{B}}(\mathbf{x}) = Y) - 1).
\end{aligned} \tag{3.20}$$

A decomposição do resultado (3.20) mostra que se a probabilidade esperada $\mathbb{E}_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\}$ para a classe majoritária correta é maior do que 0,5 acarreta um decréscimo no erro esperado de classificação, conforme $\mathbb{E}_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\} \to 1 \implies \mathbb{V}ar_{\mathcal{D}}\{\hat{P}(Y = T_{\mathcal{B}}(\mathbf{x}))\} \to 0 \implies \Phi(\cdot) \to 0$ e o erro esperado do modelo tende ao modelo de Bayes. Por outro lado se $\mathbb{E}_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\} < 0{,}5$ o erro esperado de classificação tende a aumentar, $\mathbb{E}_{\mathcal{D}}\{\hat{P}_{\mathcal{D}}(Y = T_{\mathcal{B}}(\mathbf{x}))\} \to 1 \implies \mathbb{V}ar_{\mathcal{D}}\{\hat{P}(Y = T_{\mathcal{B}}(\mathbf{x}))\} \to 0 \implies \Phi(\cdot) \to 1$.

3.3 *Ensembles*

Os *ensembles* são uma maneira mais eficiente de implementação da tentativa de dimuinuir o erro generalizado esperado de predição através da composição de vários modelos utilizando a aleatoriedade.

Dado um conjunto de aprendizado $\mathcal{D}$ o algoritmo de construção da árvore $T_{\mathcal{D}}$ é determinístico e controlado pelo conjunto de parâmetros θ: $T_{\mathcal{D},\theta}$, em que θ inclui as variáveis preditoras selecionadas em cada nó da árvore e os pontos de corte. Vamos assumir que θ pode ser induzido a variações aleatórias produzindo árvores $T_{\mathcal{D},\theta}$ que são umas diferentes das outras.

Vamos supor que temos disponível um conjunto de M árvores aleatórias geradas a partir de um conjunto de aprendizado $\mathcal{D}$ em que cada árvore é controlada pelo seu próprio parâmetro θ: $\{T_{\mathcal{D},\theta_1}, \ldots, T_{\mathcal{D},\theta_M}\}$. Denotaremos o ensemble de árvores por $\mathcal{T}_{\mathcal{D},\Theta}$, em que $\Theta = \{\theta_1, \ldots, \theta_M\}$ e $\theta_m, m = 1, \ldots, M$ seguem a mesma distribuição. Quando a perda quadrática é utilizada nos problemas de regressão, a forma mais comum de combinar os resultados de um ensemble é tomar a média das predições de cada modelo como predição final

$$\mathcal{T}_{\mathcal{D},\Theta}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} T_{\mathcal{D},\theta_m}(\mathbf{x}). \tag{3.21}$$

Considerando a média das predições (Krogh e Vedelsby, 1995) demonstrou que para um *ensemble* o erro generalizado é sempre menor que seus constituintes:

$$Erro(\mathcal{T}_{\mathcal{D},\Theta}) = \overline{E} - \overline{A}, \tag{3.22}$$

em que

$$\begin{aligned} \overline{E} &= \frac{1}{M}\sum_{m=1}^{M} Erro(T_{\mathcal{D},\theta_m}), \\ \overline{A} &= \mathbb{E}_X \left\{ \frac{1}{M}\sum_{m=1}^{M} (T_{\mathcal{D},\theta_m}(X) - \mathcal{T}_{\mathcal{D},\Theta}(X))^2 \right\}. \end{aligned} \tag{3.23}$$

O primeiro termo $\overline{E}$ corresponde à média das predições das árvores individuais, o segundo $\overline{A}$ corresponde à variância em torno da média das predições individuais. Uma vez que $\overline{A} \geq 0$, o erro generalizado do ensemble é menor (ou igual) ao dos seus constituintes.

Para classificação, perda $0-1$, geralmente utilizamos o voto majoritário como predição final

$$\mathcal{T}_{\mathcal{D},\Theta}(\mathbf{x}) = \arg\max_{c\in\mathcal{Y}} \sum_{m=1}^{M} \mathbb{1}(T_{\mathcal{D},\theta_m}(\mathbf{x}) = c). \tag{3.24}$$

Uma maneira equivalente de expressar (3.24) é utilizar as probabilidades individuais das classes $\hat{P}_{\mathcal{D},\theta_m}(Y = c|X = \mathbf{x})$ quando estão disponíveis. O ensemble é escrito como a média das probabilidades individuais

$$\mathcal{T}_{\mathcal{D},\Theta}(\mathbf{x}) = \arg\max_{c\in\mathcal{Y}} \frac{1}{M}\sum_{m=1}^{M} \hat{P}_{\mathcal{D},\theta_m(Y=c|X=\mathbf{x})}. \tag{3.25}$$

Esse resultado mostra que um ensemble, de fato, pode ser utilizado para diminuir o erro esperado generalizado de predição. A decomposição do erro esperado, Secção 3.3.1 e 3.3.2, para o ensemble nos mostrará como isso pode ocorrer.

3.3.1 Árvores de Regressão

Na discussão sobre a decomposição viés-variância, Secção 3.2.2, consideramos $\mathcal{D}$ como variável aleatória. Com a proposta de criar um ensemble para diminuir o erro esperado do modelo, vamos introduzir a variabilidade sobre o modelo através de θ.

A decomposição viés-variância pode ser acomodada para incluir ambas aleatoriedades, a do conjunto de aprendizado $\mathcal{D}$ e do procedimento de construção $T_{\mathcal{D},\theta}$. Assim, a expressão para a decomposição do Erro (3.14) pode ser expressa

$$\mathbb{E}_{\mathcal{D},\theta}\{Erro(T_{\mathcal{D},\theta}(\mathbf{x}))\} = ruído(\mathbf{x}) + viés^2(\mathbf{x}) + var(\mathbf{x}), \tag{3.26}$$

em que

$$\begin{aligned} ruído(\mathbf{x}) &= Erro(T_{\mathcal{B}}(\mathbf{x})), \\ viés^2(\mathbf{x}) &= (T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D},\theta}\{T_{\mathcal{D},\theta}(\mathbf{x})\})^2, \\ var(\mathbf{x}) &= \mathbb{E}_{\mathcal{D},\theta}\{(\mathbb{E}_{\mathcal{D},\theta}\{T_{\mathcal{D},\theta}(\mathbf{x}) - T_{\mathcal{D},\theta}(\mathbf{x})\})^2\}. \end{aligned} \tag{3.27}$$

Esta decomposição contém tanto o erro devido à aleatoriedade do conjunto de dados $\mathcal{D}$ quanto do algoritmo de construção da árvore T. Veremos que a combinação dos modelos num *ensemble* (Secção 3.3.1), proporcionará um resultado melhor do que um único modelo. O intuito é mostrar que os *emsembles* de árvores apresentam melhor resultado que as árvores individuais. Uma vez que $\theta_m, m = 1, \ldots, M$ têm a mesma distribuição, podemos representar a média e a variância, respectivamente, por

$$\begin{aligned} \mathbb{E}_{\mathcal{D},\theta_m}\{T_{\mathcal{D},\theta_m}(\mathbf{x})\} &= \mu_{\mathcal{D},\theta_m}(\mathbf{x}) = \mu_{\mathcal{D},\theta}(\mathbf{x}) \quad \text{e} \\ \mathbb{V}ar_{\mathcal{D},\theta_m}\{T_{\mathcal{D},\theta_m}(\mathbf{x})\} &= \sigma^2_{\mathcal{D},\theta_m}(\mathbf{x}) = \sigma^2_{\mathcal{D},\theta}(\mathbf{x}). \end{aligned} \tag{3.28}$$

De acordo com (3.26) o erro generalizado de um *ensemble* pode ser decomposto em $ruído(\mathbf{x}) + viés(\mathbf{x}) + var(\mathbf{x})$. O *ruído* não importa o modelo, permanece o mesmo, pois depende somente da aleatoriedade intrínseca de Y

$$ruído(\mathbf{x}) = \mathbb{E}_{Y|X=\mathbf{x}}\{(Y - T_{\mathcal{B}}(\mathbf{x}))^2\}. \tag{3.29}$$

Para calcular o *viés* vamos calcular a média do ensemble

$$\begin{aligned} \mathbb{E}_{\mathcal{D},\Theta}\{\mathcal{T}_{\mathcal{D},\Theta}(\mathbf{x})\} &= \mathbb{E}_{\mathcal{D},\Theta}\left\{\frac{1}{M}\sum_{m=1}^{M} T_{\mathcal{D},\theta_m}(\mathbf{x})\right\} \\ &= \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\mathcal{D},\theta_m}\{T_{\mathcal{D},\theta_m}(\mathbf{x})\} \\ &= \mu_{\mathcal{D},\theta}(\mathbf{x}), \end{aligned} \tag{3.30}$$

já que todos as variáveis θ_m são independentes e seguem a mesma distribuição. Logo,

$$viés^2(\mathbf{x}) = (T_{\mathcal{B}}(\mathbf{x}) - \mu_{\mathcal{D},\theta}(\mathbf{x}))^2, \tag{3.31}$$

de modo que o *viés* do ensemble tem o mesmo *viés* de qualquer modelo constituinte, *i.e.*, o ensemble não tem efeito sobre o *viés* (ao quadrado).

Para encontrar a variância, precisamos calcular

$$var(\mathbf{x}) = \mathbb{V}ar_{\mathcal{D},\Theta}\left\{\frac{1}{M}\sum_{m=1}^{M} T_{\mathcal{D},\theta_m}(\mathbf{x})\right\}. \tag{3.32}$$

Primeiro vamos considerar dois modelos gerados a partir do mesmo conjunto de aprendizado $\mathcal{D}$ cujos parâmetros são θ_i e θ_j. A correlação (de Pearson) entre esses dois parâmetros é dada por

$$\begin{aligned}\rho(\mathbf{x}) &= \frac{\mathbb{C}ov_{\mathcal{D},\theta_i,\theta_j}(T_{\mathcal{D},\theta_i}(\mathbf{x}), T_{\mathcal{D},\theta_j}(\mathbf{x}))}{\sqrt{\mathbb{V}ar(T_{\mathcal{D},\theta_i}(\mathbf{x})) \cdot \mathbb{V}ar(T_{\mathcal{D},\theta_j}(\mathbf{x}))}} \\ &= \frac{\mathbb{C}ov_{\mathcal{D},\theta_i,\theta_j}(T_{\mathcal{D},\theta_i}(\mathbf{x}), T_{\mathcal{D},\theta_j}(\mathbf{x}))}{\sigma^2_{\mathcal{D},\theta}(\mathbf{x})}.\end{aligned} \tag{3.33}$$

em que $\mathbb{C}ov$ é a covariância.

Aplicando a fórmula padrão para o cálculo da variância, temos

$$\begin{aligned}var(\mathbf{x}) &= \frac{1}{M^2}\left\{\sum_{m=1}^{M}\mathbb{V}ar_{\mathcal{D},\theta_m}(T_{\mathcal{D},\theta_m}(\mathbf{x})) + 2\sum_{i=2}^{M}\sum_{j<i}\mathbb{C}ov_{\mathcal{D},\theta_i,\theta_j}(T_{\mathcal{D},\theta_i}(\mathbf{x}), T_{\mathcal{D},\theta_j}(\mathbf{x}))\right\} \\ &= \frac{1}{M^2}\left\{M\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) + M(M-1)\rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x})\right\}.\end{aligned} \tag{3.34}$$

Esse último passo pode ser visto a partir de (3.33)

$$\begin{aligned}2\sum_{i=2}^{M}\sum_{j<i}\mathbb{C}ov_{\mathcal{D},\theta_i,\theta_j}(T_{\mathcal{D},\theta_i}(\mathbf{x}), T_{\mathcal{D},\theta_j}(\mathbf{x})) &= 2\sum_{i=2}^{M}\sum_{j<i}\rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) \\ &= 2\frac{M(M-1)}{2}\rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) \\ &= M(M-1)\rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}).\end{aligned} \tag{3.35}$$

Desenvolvendo (3.34)

$$\begin{aligned}var(\mathbf{x}) &= \frac{\sigma^2_{\mathcal{D},\theta}(\mathbf{x})}{M} + \frac{M-1}{M}\rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) \\ &= \frac{\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) + M\rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) - \rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x})}{M} \\ &= \rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{M}\sigma^2_{\mathcal{D},\theta}(\mathbf{x}).\end{aligned} \tag{3.36}$$

Podemos ver que $\rho(\mathbf{x})$ é uma medida de quanto uma perturbação aleatória afeta o resultado de um modelo - através do componente da variância. Quando $\rho(\mathbf{x}) \to 1 \implies var(\mathbf{x}) \to \sigma^2_{\mathcal{D},\theta}(\mathbf{x})$ os modelos estão altamente correlacionados, sugerindo que a aleatorização tem pouco efeito no resultado, por outro lado quando $\rho(\mathbf{x}) \to 0 \implies var(\mathbf{x}) \to \frac{\sigma^2_{\mathcal{D},\theta}(\mathbf{x})}{M}$ os modelos são não correlacionados (decorrelacionados) indicando que a aleatorização tem um grande efeito nas predições, eventualmente $M \to \infty \implies var(\mathbf{x}) \to 0$.

Consolidando o resultado para os ensembles, a decomposição viés-variância do valor esperado do erro generalizado $\mathbb{E}_{\mathcal{D}}\{\mathcal{T}_{\mathcal{D},\Theta}\}(\mathbf{x}), \Theta = \{\theta_1, \ldots, \theta_M\}$ em $X = \mathbf{x}$ para a perda quadrática de um ensemble com M modelos aleatorizados $T_{\mathcal{D},\theta_m}(\mathbf{x})$ é

$$\mathbb{E}_{\mathcal{D}}\{\mathcal{T}_{\mathcal{D},\Theta}\}(\mathbf{x}) = ruído(\mathbf{x}) + viés^2(\mathbf{x}) + var(\mathbf{x}), \tag{3.37}$$

em que

$$\begin{aligned} ruído(\mathbf{x}) &= Erro(T_{\mathcal{B}}(\mathbf{x})), \\ viés^2 &= (T_{\mathcal{B}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D},\theta}\{T_{\mathcal{D},\theta}(\mathbf{x})\})^2, \\ var(\mathbf{x}) &= \rho(\mathbf{x})\sigma^2_{\mathcal{D},\theta}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{M}\sigma^2_{\mathcal{D},\theta}(\mathbf{x}). \end{aligned} \tag{3.38}$$

Ainda resta mostrar que $\rho(\mathbf{x}) \geq 0$. Retomando a expressão (3.33), precisamos apenas notar que a covariância é maior ou igual a zero, *i.e.*,

$$\begin{aligned} \mathbb{C}ov_{\mathcal{D},\theta_i,\theta_j}(T_{\mathcal{D},\theta_i}(\mathbf{x}), T_{\mathcal{D},\theta_j}(\mathbf{x})) &= \mathbb{E}_{\theta_i,\theta_j,\mathcal{D}}\{T_{\mathcal{D},\theta_i}(\mathbf{x}) \cdot T_{\mathcal{D},\theta_j}(\mathbf{x})\} - \mathbb{E}_{\mathcal{D},\theta_i}\{T_{\mathcal{D},\theta_i}(\mathbf{x})\} \cdot \mathbb{E}_{\mathcal{D},\theta_j}\{T_{\mathcal{D},\theta_j}(\mathbf{x})\} \\ &= \mathbb{E}_{\mathcal{D}}\{\mathbb{E}_{\theta_i|\mathcal{D}}\{T_{\mathcal{D},\theta_i}(\mathbf{x})\} \cdot \mathbb{E}_{\theta_j|\mathcal{D}}\{T_{\mathcal{D},\theta_j}(\mathbf{x})\}\} - \mathbb{E}^2_{\mathcal{D}}\{\mathbb{E}_{\theta|\mathcal{D}}\{T_{\mathcal{D},\theta}(\mathbf{x})\}\} \\ &= \mathbb{E}_{\mathcal{D}}\{\mathbb{E}^2_{\theta|\mathcal{D}}\{T_{\mathcal{D},\theta}(\mathbf{x})\}\} - \mathbb{E}^2_{\mathcal{D}}\{\mathbb{E}_{\theta|\mathcal{D}}\{T_{\mathcal{D},\theta}(\mathbf{x})\}\} \\ &= \mathbb{V}ar_{\mathcal{D}}\{\mathbb{E}_{\theta|\mathcal{D}}\{T_{\mathcal{D},\theta}(\mathbf{x})\}\} \geq 0. \end{aligned} \tag{3.39}$$

3.3.2 Árvores de Classificação

Na Secção 3.2.3 mostramos como o erro generalizado pode ser decomposto para classificação. Uma vez que o erro generalizado é menor do que erro dos seus constituintes (cf. 3.22), construindo um ensemble com as probabilidades das classes reduz a variância da estimativa $\mathbb{E}_{\mathcal{D},\theta}\{\hat{P}(Y = T_{\mathcal{B}}(\mathbf{x}))\}$, cf. (3.36), o que leva a um decréscimo no erro esperado de classificação - se a estimativa do erro esperado permanecer acima de 0.5 no *ensemble*.

3.4 *Random Forests*

As *random forests* podem ser induzidas através de três caminhos:

1. Aleatorização na escolha do conjunto de observações para a construção de cada árvore.
2. Aleatorização na escolha da variável da partição.
3. Após a escolha da variável da partição, a escolha do ponto de corte.

Nem todos os caminhos, nem todas as escolhas podem se revelar em bons caminhos ou boas escolhas. O sucesso de uma proposta parece depender de uma boa implementação de software bem como de uma análise mais detalhada do algoritmo tanto do ponto de vista teórico quanto empírico. Este é o caso de Breiman que a partir do artigo (Breiman, 2001) em que analisa teoricamente sua proposta para as *random forests*, além de mostrar resultados empíricos, tem o nome fortemente associado às *random forests* embora não seja o único, nem o primeiro a propor um *ensemble* para as árvores - ele também disponibilizou um software (Breiman, 2002) de uso gratuito.

Pontos chave de algumas propostas para construção das *random forests*:

- (Dietterich e Kong, 1995), A cada árvore, a cada nó
  - considere todas as variáveis disponíveis.
  - selecione uniformemente uma entre as 20 melhores partições.
- (Amit *et al.*, 1997), A cada árvore, a cada nó
  - selecione aleatoriamente $k \leq p$ entre as p variáveis disponíveis.
- (Ho, 1998), *Decision Forest*. A cada árvore

  - selecione uniformemente $k \leq p$ entre as p variáveis disponíveis, isto cria o que é chamado de *Random Subspace*.

- (Breiman, 2001), *Random Forest*, vide Algoritmo 2. A cada árvore
  - uma amostra boostrap de tamanho n com reposição igual ao tamanho do conjunto de aprendizado.
  - a cada nó
    * selecione aleatoriamente $k \leq p$ entre as p variáveis disponíveis para construir a partição.

- (Cutler e Zhao, 2001), PERT (*Perfect Random Tree Ensembles*). A cada árvore, a cada nó t
  - selecione aleatoriamente uma variável X_j entre as p disponíveis.
  - o ponto de corte para realizar a partição deve ser escolhido da seguinte forma:
    * selecione duas amostras do conjunto de observações alocadas em t: $(\mathbf{x}_1, y_1)$ e $(\mathbf{x}_2, y_2)$,
    * escolher $\alpha \in [0, 1]$ uniformemente,
    * calcular o ponto de corte: $v = \alpha x_{1j} + (1 - \alpha) x_{2j}$.

- (Geurts *et al.*, 2006), Extra-Trees (*Extremely Randomized Trees*). A cada árvore, a cada nó t
  - selecione aleatoriamente uma variável X_j entre as p disponíveis.
  - o ponto de corte para realizar a partição deve ser escolhido da seguinte forma:
    * encontre o valor mínimo de $x_{ji} | (\mathbf{x}_i, y_i) \in t = min_t$.
    * encontre o valor máximo de $x_{ji} | (\mathbf{x}_i, y_i) \in t = max_t$.
    * o ponto de corte é escolhido uniformemente em $[min_t, max_t[$.

3.5 Estimativa do Erro Generalizado

As *random forests* proporcionam uma alternativa à estimativa do erro generalizado à validação cruzada ou ao uso de uma base teste. Uma vez que a cada árvore a amostra $\mathcal{D}_{-m} = \mathcal{D} \setminus \mathcal{D}_m$ não é utilizada para a construção da árvore, ela pode ser usada para estimar o erro. Essa estimativa é chamada de *out-of-bag* do erro. Considere $m_{\ell_1}, \ldots, m_{\ell_{M_{-i}}}$ os índices das M_{-i} árvores em que o par $(\mathbf{x}_i, y_i)$ não está presente em uma floresta com M árvores $\{T_1, \ldots, T_M\}$. A estimativa do erro generalizado para regressão é

$$\widehat{Erro}_{oob} = \frac{1}{n} \sum_{i=1}^{n} L \left(\frac{1}{M_{-i}} \sum_{j=1}^{M_{-i}} T_{m_{\ell_j}}(\mathbf{x}_i), y_i \right). \tag{3.40}$$

Para árvores de classificação precisamos substituir na expressão (3.40) a média pela classe majoritária.

3.6 Importância das Variáveis

Na Secção 2.7 apresentamos o cálculo de importância das variáveis para uma única árvore CART. As *random forests*, devido ao processo de seleção das variáveis a cada nó, não necessitam do cálculo das partições substitutas. O processo de aleatorização concede chances iguais a todas as

Algoritmo 2: *Random Forest* - Regressão e Classificação.

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Para cada $m = 1, \ldots, M$:
   (a) Selecione uma amostra boostrap $\mathcal{D}_m$ de tamanho n com reposição igual ao tamanho do conjunto de aprendizado.
   (b) Construa uma árvore CART T_m utilizando a amostra $\mathcal{D}_m$ usando o seguinte procedimento:
      i. Selecione aleatoriamente em cada nó $k \leq p$.
      ii. Utilize somente as k variáveis para encontrar a melhor partição.
      iii. Declare um nó terminal quando $|t| \leq n_{min}$ (fixado).

Resultado:

Para regressão:

$$T(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} T_m(\mathbf{x}).$$

Para classificação :

$$\text{Seja:} \quad N_k(\mathbf{x}) = \sum_{m=1}^{M} \mathbb{1}(T_m(\mathbf{x}) = c_k),$$

$$\text{Então,} T(\mathbf{x}) = \arg\max_{c_k} N_k(\mathbf{x}).$$

variáveis disponíveis em um dado nó evitando o mascaramento de uma variável por outra. Além disso, a utilização do *bagging* ajuda a diminuir o efeito do mascaramento (Gilles, 2014).

Para uma *random forest* com M árvores $\{T_1, \ldots, T_M\}$, a medida de importância de uma variável X_ℓ é a média dos decréscimos de impureza, considerados os nós t_{mj}, $j = 1, \ldots, J_m - 1$ em que a variável é selecionada em T_m, ponderados pela proporção de observações em relação do total alocadas em cada respectivo nó. Seja $I_{mj_\ell} = \{j | \Delta I(s^\star, t_{mj}) = \Delta I(s^\ell, t_{mj}), j = 1, \ldots, J_m - 1\}$ o conjunto dos índices dos nós em que X_ℓ é a variável selecionada para particionar t_{mj}. A importância de X_ℓ é dada por

$$\mathcal{I}mp(X_\ell) = \frac{1}{M} \sum_{m=1}^{M} \sum_{j \in I_{mj_\ell}} p(t_{mj}) \cdot \Delta I(s^\ell, t_{mj}), \tag{3.41}$$

em que $p(t_{mj}) = \frac{|t_{mj}|}{n}$ é a proporção de amostras alocadas em t_{mj}. Uma medida relativa, de modo que a importância máxima seja majorada em 100, como nas árvores únicas, cf. Secção 2.7, pode ser derivada:

$$\mathcal{I}(X_\ell) = 100 \times \frac{\mathcal{I}mp(X_\ell)}{\max_k \mathcal{I}mp(X_k)}. \tag{3.42}$$

Assim, a(s) variável(eis) mais importante(s) assume o valor 100 e as demais permanecem no intervalo de 0 a 100.

3.7 Ilustração

Uma *random forest* é construída a partir de uma série de árvores CART geradas a partir das respectivas amostras bootstrap do conjunto de dados original. Vamos considerar uma floresta bastante simples com somente $M = 3$ árvores e cada árvore somente com dois nós terminais $|\tilde{T}| = 2$. Usaremos $p = 3$ variáveis e cada árvore será construída com somente uma variável $k = 1$ imitando uma seleção aleatória na qual uma entre as três foi usada em cada nó raiz.

Árvores de Regressão

As árvores serão construídas utilizando a perda quadrática. O conjunto de dados, $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^3, y_i \in \mathbb{R}, n = 8\}$, é o exposto na Tabela 3.1.

Tabela 3.1: *Ilustração: Random Forest Regressão - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
2	1,8	2,0	0	4,0
3	0,4	6,0	1	2,5
4	1,9	8,0	0	1,5
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0
7	4,0	5,0	1	0,8
8	3,5	4,5	0	0,4

As amostras bootstrap $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ e as respectivas *out-of-bags* $\{\mathcal{D}_1^{oob}, \mathcal{D}_2^{oob}, \mathcal{D}_3^{oob}\}$ são geradas a partir de $\mathcal{D}$, as tabelas na Figura 3.1 conservam os índices originais para frisar a amostragem com reposição.

$\mathcal{D}_1$

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
1	0,8	3,5	1	5,0
3	0,4	6,0	1	2,5
5	2,5	3,6	1	9,5
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0
7	4,0	5,0	1	0,8
7	4,0	5,0	1	0,8

$\mathcal{D}_2$

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
2	1,8	2,0	0	4,0
3	0,4	6,0	1	2,5
3	0,4	6,0	1	2,5
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0
6	3,0	2,5	0	8,0
8	3,5	4,5	0	0,4

$\mathcal{D}_3$

i	x_1	x_2	x_3	y_i
2	1,8	2,0	0	4,0
2	1,8	2,0	0	4,0
3	0,4	6,0	1	2,5
3	0,4	6,0	1	2,5
4	1,9	8,0	0	1,5
5	2,5	3,6	1	9,5
8	3,5	4,5	0	0,4
8	3,5	4,5	0	0,4

$\mathcal{D}_1^{oob}$

i	x_1	x_2	x_3	y_i
2	1,8	2,0	0	4,0
4	1,9	8,0	0	1,5
8	3,5	4,5	0	0,4

$\mathcal{D}_2^{oob}$

i	x_1	x_2	x_3	y_i
4	1,9	8,0	0	1,5
7	4,0	5,0	1	0,8

$\mathcal{D}_3^{oob}$

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
6	3,0	2,5	0	8,0
7	4,0	5,0	1	0,8

Figura 3.1: *Ilustração: Random Forest - Amostras Bootstrap e as respectivas Amostras* out-of-bag.

Utilizando os resultados da árvore CART na Secção 2.9, vamos gerar as árvores $\{T_1, T_2, T_3\}$ a partir das respectivas partições $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ conforme Figura 3.1. O resultado está na Figura 3.2 em que mostramos somente a melhor partição e as predições correspondentes.

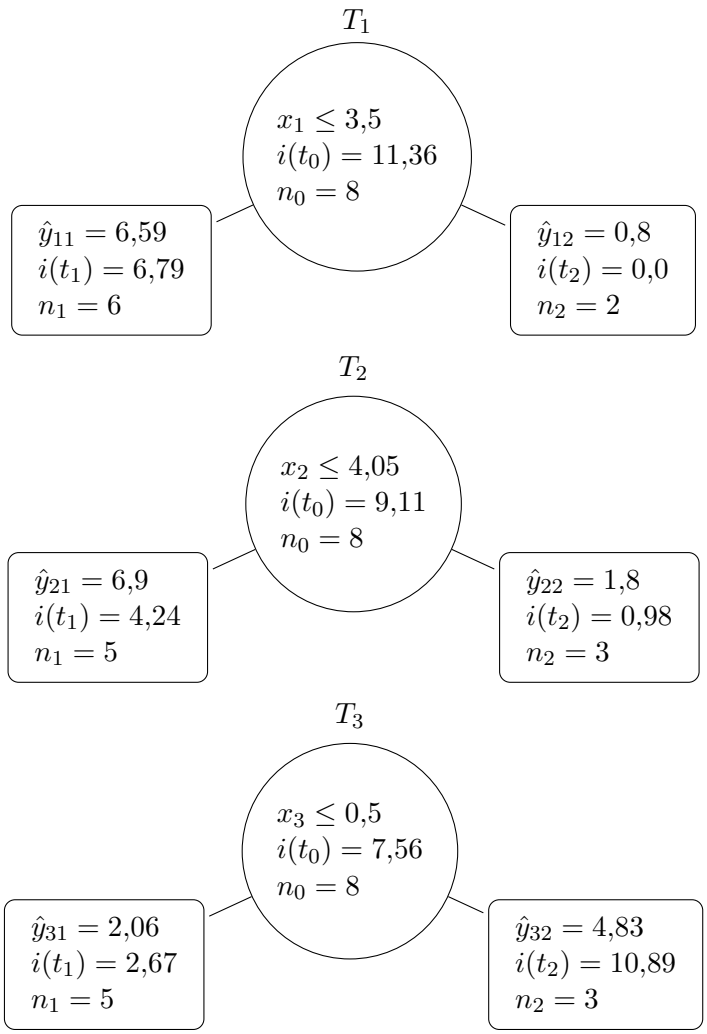


Figura 3.2: *Ilustração: Random Forest com 3 Árvores CART.*

Uma vez que a floresta foi construída podemos utilizá-la fazer predições, por exemplo, o vetor de entrada $\mathbf{x}_k = (2{,}8, 8{,}7, 0)$ tem a seguinte predição

$$\begin{aligned} T(\mathbf{x}_k) &= \frac{1}{3}\left(T_1(\mathbf{x}_k) + T_2(\mathbf{x}_k) + T_3(\mathbf{x}_k)\right) \\ &= \frac{1}{3}(\hat{y}_{11} + \hat{y}_{22} + \hat{y}_{31}) = \frac{1}{3}(6{,}59 + 1{,}8 + 2{,}06) \\ &= 3{,}48. \end{aligned} \tag{3.43}$$

Para a estimativa do erro, a predição de cada amostra *out-of-bag* deve ser feita em todas as árvores em que a respectiva amostra não está presente, a Tabela 3.2 contém o cálculo.

Tabela 3.2: *Ilustração: Random Forest - Predições out-of-bag.*

i	$T_1(\mathbf{x}_i)$	$T_2(\mathbf{x}_i)$	$T_3(\mathbf{x}_i)$	*média*	y_i
1	-	-	4,83	4,83	5,0
2	-	6,9	-	6,9	4,0
4	6,59	1,8	-	4,2	1,5
6	-	-	2,06	2,06	8,0
7	0,8	-	4,83	2,82	0,8
8	-	1,8	-	1,8	0,4

O cálculo da estimativa do erro out-of-bag é

$$\begin{aligned}\widehat{Erro}_{oob} &= \frac{1}{8}\left[(5{,}0-4{,}83)^2+(4{,}0-6{,}9)^2+\ldots+(0{,}4-1{,}8)^2\right]\\ &= 7{,}61.\end{aligned} \tag{3.44}$$

O erro do conjunto de dados original é de 9,87, cf. expressão (2.38). A floresta apresenta uma estimativa um pouco mais otimista de 7,61.

A importância das variáveis, expressões (3.41) e (3.42), é simplificada neste caso, pois cada variável é usada somente uma vez na raiz das árvores. Na Figura 3.2 podemos verificar em cada nó a impureza calculada. A importância das variáveis é

$$\begin{aligned}\mathcal{I}mp(x_1) &= \frac{1}{3}\Delta(x_1\leq 3{,}5,t_0) = \frac{1}{3}(11{,}36-\frac{6}{8}\times 6{,}79-\frac{2}{8}\times 0{,}0) = 2{,}09,\\ \mathcal{I}mp(x_2) &= \frac{1}{3}\Delta(x_2\leq 4{,}05,t_0) = \frac{1}{3}(9{,}11-\frac{5}{8}\times 4{,}24-\frac{3}{8}\times 0{,}98) = 2{,}03,\\ \mathcal{I}mp(x_3) &= \frac{1}{3}\Delta(x_3\leq 0{,}5,t_0) = \frac{1}{3}(7{,}56-\frac{5}{8}\times 2{,}67-\frac{3}{8}\times 10{,}89) = 0{,}60.\end{aligned} \tag{3.45}$$

A importância relativa pode ser calculada notando que a variável x_1 tem a maior medida de importância:

$$\begin{aligned}\mathcal{I}(x_1) &= 100,\\ \mathcal{I}(x_2) &= 100\times\frac{2{,}03}{2{,}09} = 97{,}1,\\ \mathcal{I}(x_3) &= 100\times\frac{0{,}60}{2{,}09} = 28{,}7.\end{aligned} \tag{3.46}$$

Árvores de Classificação

As árvores serão construídas utilizando o Índice de Gini como medida de impureza. O conjunto de dados, $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^3, y_i \in \{0,1\}, n = 8\}$, é o exposto na Tabela 3.3.

Tabela 3.3: *Ilustração: Random Forest Classificação - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
2	1,8	2,0	0	1
3	0,4	6,0	1	0
4	1,9	8,0	0	0
5	2,5	3,6	1	0
6	3,0	2,5	0	1
7	4,0	5,0	1	1
8	3,5	4,5	0	1

As amostras bootstrap $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ e as respectivas amostras out-of-bag $\{\mathcal{D}_1^{oob}, \mathcal{D}_2^{oob}, \mathcal{D}_3^{oob}\}$ são geradas a partir de $\mathcal{D}$, as tabelas na Figura 3.3 conservam os índices originais para frisar a amostragem com reposição.

$\mathcal{D}_1$

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
1	0,8	3,5	1	1
3	0,4	6,0	1	0
5	2,5	3,6	1	0
5	2,5	3,6	1	0
6	3,0	2,5	0	1
7	4,0	5,0	1	1
7	4,0	5,0	1	1

$\mathcal{D}_2$

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
2	1,8	2,0	0	1
3	0,4	6,0	1	0
3	0,4	6,0	1	0
5	2,5	3,6	1	0
6	3,0	2,5	0	1
6	3,0	2,5	0	1
8	3,5	4,5	0	1

$\mathcal{D}_3$

i	x_1	x_2	x_3	y_i
2	1,8	2,0	0	1
2	1,8	2,0	0	1
3	0,4	6,0	1	0
3	0,4	6,0	1	0
4	1,9	8,0	0	0
5	2,5	3,6	1	0
8	3,5	4,5	0	1
8	3,5	4,5	0	1

$\mathcal{D}_1^{oob}$

i	x_1	x_2	x_3	y_i
2	1,8	2,0	0	1
4	1,9	8,0	0	0
8	3,5	4,5	0	1

$\mathcal{D}_2^{oob}$

i	x_1	x_2	x_3	y_i
4	1,9	8,0	0	0
7	4,0	5,0	1	1

$\mathcal{D}_3^{oob}$

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
6	3,0	2,5	0	1
7	4,0	5,0	1	1

Figura 3.3: *Ilustração: Random Forest - Amostras Bootstrap e as respectivas Amostras out-of-bag.*

Utilizando os resultados da árvore CART na Secção 2.9, vamos gerar as árvores $\{T_1, T_2, T_3\}$ a partir das respectivas partições $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ conforme Figura 3.3. O resultado está na Figura 3.4 em que mostramos somente a melhor partição e as predições correspondentes.

Uma vez que a floresta foi construída podemos utilizá-la fazer predições, por exemplo, o vetor de entrada $\mathbf{x}_k = (2.8, 8.7, 0)$ tem a seguinte predição

$$\begin{cases} T_1(\mathbf{x}_k) = 0 \\ T_2(\mathbf{x}_k) = 0 \\ T_3(\mathbf{x}_k) = 1 \end{cases} \implies T(\mathbf{x}_k) = 0. \tag{3.47}$$

Para a estimativa do erro, a predição de cada amostra out-of-bag deve ser feita em todas as árvores em que a respectiva amostra não está presente, a Tabela 3.4 contém o cálculo.

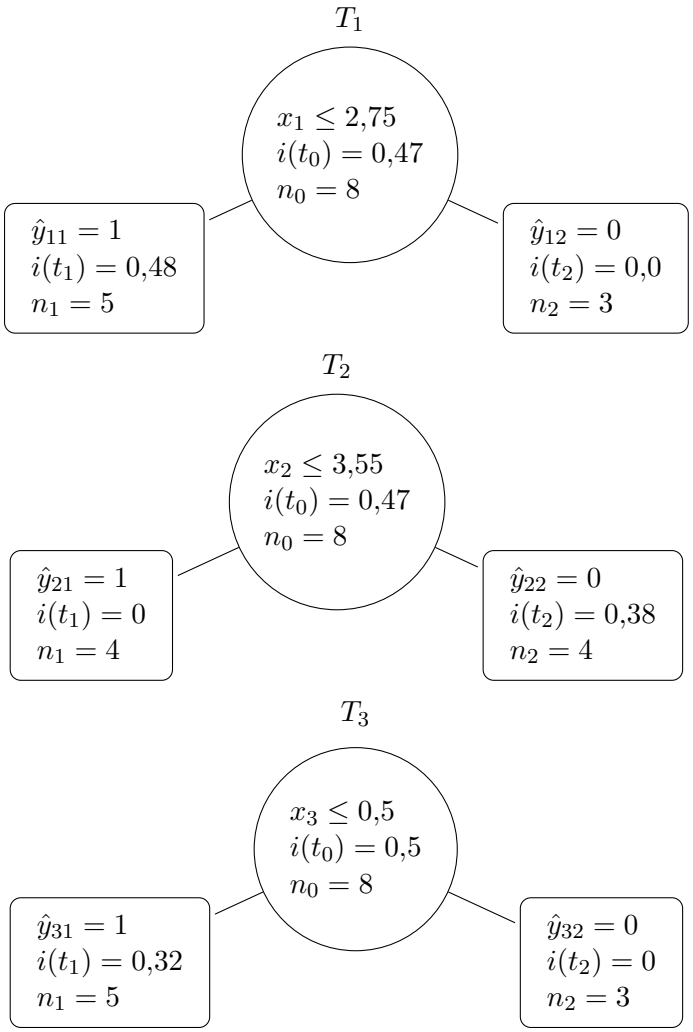


Figura 3.4: *Ilustração: Random Forest com 3 Árvores CART.*

Tabela 3.4: *Ilustração: Random Forest - Predições out-of-bag.*

i	$T_1(\mathbf{x}_i)$	$T_2(\mathbf{x}_i)$	$T_3(\mathbf{x}_i)$	$T(\mathbf{x}_i)$	y_i
1	-	-	0	0	1
2	1	-	-	1	1
4	1	0	-	0	0
6	-	-	1	1	1
7	-	0	0	0	1
8	0	-	-	0	1

O cálculo da estimativa do erro out-of-bag é

$$\begin{aligned} \widehat{Erro}_{oob} &= \frac{1}{8}\left[\mathbb{1}(T(\mathbf{x}_1) \neq y_1) + \cdots \mathbb{1}(T(\mathbf{x}_8) \neq y_8)\right] \\ &= 0{,}38. \end{aligned} \tag{3.48}$$

A importância das variáveis, expressões (3.41) e (3.42), é simplificada neste caso, pois cada variável é usada somente uma vez na raiz das árvores. Na Figura 3.2 podemos verificar em cada nó

a impureza calculada. A importância das variáveis é

$$\begin{aligned}
\mathcal{I}mp(x_1) &= \frac{1}{3}\Delta(x_1 \leq 2{,}75, t_0) = \frac{1}{3}(0{,}47 - \frac{5}{8} \times 0{,}48 - \frac{3}{8} \times 0{,}0) = 0{,}06, \\
\mathcal{I}mp(x_2) &= \frac{1}{3}\Delta(x_2 \leq 3{,}55, t_0) = \frac{1}{3}(0{,}47 - \frac{4}{8} \times 0{,}0 - \frac{4}{8} \times 0{,}38) = 0{,}09, \\
\mathcal{I}mp(x_3) &= \frac{1}{3}\Delta(x_3 \leq 0{,}5, t_0) = \frac{1}{3}(0{,}5 - \frac{5}{8} \times 0{,}32 - \frac{3}{8} \times 0{,}0) = 0{,}1.
\end{aligned} \tag{3.49}$$

A importância relativa pode ser calculada notando que a variável x_1 tem a maior medida de importância:

$$\begin{aligned}
\mathcal{I}(x_3) &= 100, \\
\mathcal{I}(x_2) &= 100 \times \frac{0{,}09}{0{,}1} = 90, \\
\mathcal{I}(x_1) &= 100 \times \frac{0{,}0{,}6}{0{,}1} = 60.
\end{aligned} \tag{3.50}$$

4 AdaBoost

As técnicas para construção de árvores apresentadas até o momento são todas baseadas num mesmo modelo de construção de árvores. O modelo básico CART serviu de base para os aperfeiçoamentos *bagging* e *random forest* (Capítulo 3). Ambos trabalham com réplicas aleatórias do modelo CART combinando os resultados das diversas árvores geradas a fim de produzir uma predição. O procedimento AdaBoost, acrônimo de **Ada**ptative **Boost**ing, introduz uma mudança de perspectiva.

AdaBoost é a combinação de duas ideias muito úteis na construção de árvores. Embora esta técnica se valha do procedimento CART, difere fundamentalmente em como as árvores construídas são utilizadas para realizar a predição.

A primeira ideia é conhecida como PAC - *Probably Approximately Correct* - Provavelmente Aproximadamente Correto. Apresentada primeiro por L. G. Valiant no artigo Valiant (1984), pode-se ver uma discussão não técnica no livro Valiant (2013). A ideia é construir uma sequência de **classificadores fracos** (*weak classifiers*) na qual cada classificador subsequente leva em consideração o erro cometido anteriormente. Cada classificador recebe um peso e ao final todos os classificadores, com seus respectivos pesos, são utilizados para realizar a predição. Um classificador fraco é um classificador que acerta um pouquinho melhor do que um palpite aleatório.

A segunda é a ideia de Boosting. Boosting é o processo pelo qual um classificador fraco é transformado em um classificador com uma grande acurácia (Freund e Schapire, 1997), ou de outra maneira, Boosting é algo que combina classificadores fracos em um *comitê* cuja predição se torna bastante acurada (Schapire e Freund, 2012).

Segundo Hastie *et al.* (2009), Breiman se referiu ao AdaBoost como *o melhor classificador de prateleira em todo o mundo* (no evento NIPS-Neural Information Processing Systems em 1996).

4.1 CART como Classificador Fraco

Friedman *et al.* (2000) menciona que muitos autores têm escrito sobre o sucesso do uso de AdaBoost na produção de classificadores acurados, menciona também que muitos têm explorado como classificador fraco algoritmos baseados em árvore e demonstraram que seu uso produz consistentemente taxas de erros menores do que uma simples árvore.

O artigo de Freund e Schapire (1997) é o marco fundador do AdaBoost, Friedman *et al.* (2000) chamam este modelo AdaBoost Discreto. Considere um conjunto de dados cuja variável resposta $\mathcal{Y} \in \{-1, +1\}$ e o vetor de variáveis preditoras $\mathbf{X} = (X_1, \ldots, X_p)$. Um conjunto de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i),\ \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ |\mathcal{D}| = n\}$ está disponível para gerar as árvores. O algoritmo de construção para cálculo da predição é iterativo. Seja $T_m(\mathbf{x})$ o preditor da árvore na *m-ésima* iteração, $m = 1, \ldots, M$. Cada árvore é construída segundo o modelo CART, mas somente com dois nós terminais (*stump*) - esse é o classificador fraco, ver detalhes Figura 4.1.

Cada árvore recebe um peso α_m e o classificador final é dado pela soma ponderada dos classificadores

$$T(\mathbf{x}) = \text{sinal}\left[\sum_{m=1}^{M} \alpha_m \cdot T_m(\mathbf{x})\right]. \tag{4.1}$$

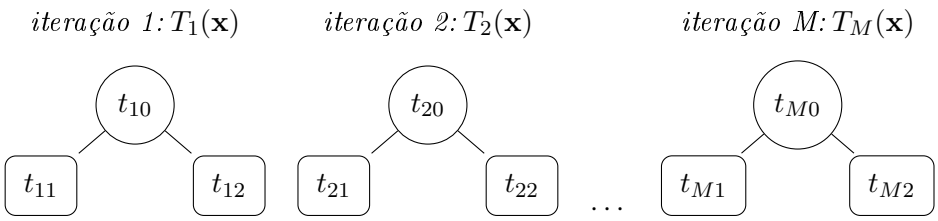


Figura 4.1: *Classificador fraco para o AdaBoost. Árvores CART com apenas dois nós terminais (stumps).*

Esquematicamente a Figura 4.2 apresenta o funcionamento do AdaBoost. O processo inicia com os pesos iguais para todas as observações - $\mathbf{w}_1 = [1/n, \ldots, 1/n]$. A cada iteração os pesos são atualizados e uma nova versão dos dados é produzida, proporcional aos pesos, que serve de entrada para a próxima iteração. Os pesos são ajustados baseados nos erros cometidos pelo classificador a cada iteração.

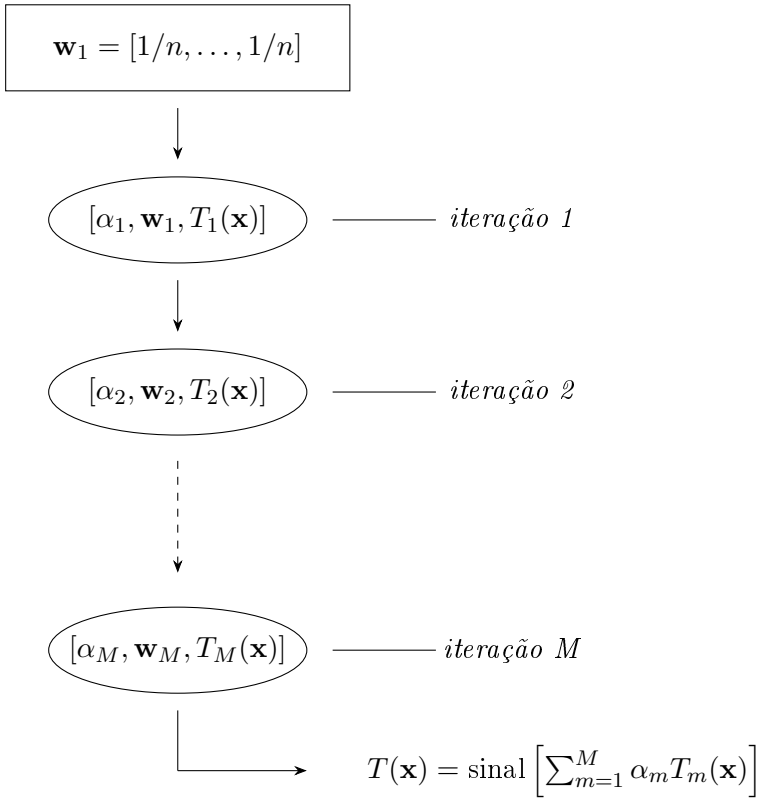


Figura 4.2: *Esquema do funcionamento do AdaBoost.M1. A cada iteração uma versão do conjunto de dados é utilizada. O classificador final é a soma ponderada dos classificadores.*

Para complementar o esquema apresentado na Figura 4.2, apresentamos o algoritmo de cálculo - Algoritmo 3. Para começar, todas as observações tem o mesmo peso proporcional à quantidade de dados disponíveis $\mathbf{w}_1 = [1/n, \ldots, 1/n]$, linha 1 do algoritmo. Um total de M iterações são realizadas para a construção do previsor AdaBoost final $T(\mathbf{x})$. A cada iteração m construímos um classificador fraco $T_m(\mathbf{x})$ a partir de uma árvore CART com apenas dois nós utilizando os pesos $\mathbf{w}_m = [w_{m1}, \ldots, w_{mn}]$, linha $2(a)$ - podemos construir um *stump* para cada variável e escolher o que tem o menor erro, em cada iteração. Na sequência calculamos o erro do classificador ϵ_m, linha $2(b)$, em que vemos que as observações cuja previsão está errada contribuem com seu respectivo peso para o cálculo do erro. O classificador T_m ganha então seu peso $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$, a Figura

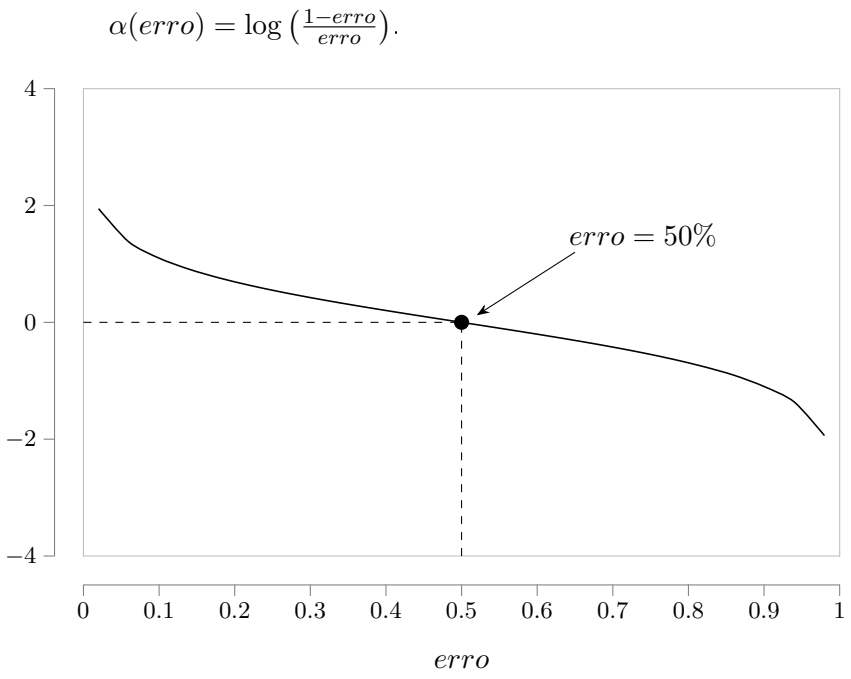


Figura 4.3: *Curva do peso α em função do erro*

4.3 ilustra a curva do peso α em função do erro.

Quanto menor o erro:

1. o classificador ganha um peso relativamente maior.
2. maior será o peso, na próxima iteração, da observação cuja predição foi errada. Isto faz com que estas observações aumentem sua influência na próxima iteração fazendo com que o algoritmo nelas se concentre a fim de tentar classificá-las corretamente.

A curva do peso α, Figura 4.3, mostra também que um classificador que erra em 50% dos casos tem peso zero, *i.e.*, se ele acerta tanto quanto uma moeda honesta, não contribui para o classificador final. Note que para um problema de classificação binária, o erro nunca é maior do que 50%.

4.2 Modelos Aditivos

O artigo Friedman *et al.* (2000) lança uma nova compreensão sobre o modelo AdaBoost. Dentro de um escopo maior AdaBoost pode ser entendido como um *modelo aditivo*. De acordo com Hastie *et al.* (2009) os modelos aditivos são representados como soma de funções *base* elementares. Genericamente

$$f(\mathbf{x}) = \sum_{m=1}^{M} \beta_m b(\mathbf{x}, \theta_m), \tag{4.2}$$

em que $\mathbf{x} \in \mathbb{R}^n$, $\beta_m, m = 1, \ldots, M$, são os coeficientes de *expansão* escalares, $b(\mathbf{x}, \theta) \in \mathbb{R}$ e θ_m é um conjunto de parâmetros que caracterizam b.

Usualmente estes modelos são ajustados a partir da minimização de uma função de perda L aplicada aos dados de treinamento,

$$(\beta_m, \theta_m) = \min_{\beta_m, \theta_m} \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta_m b(\mathbf{x}_i, \theta_m)\right), m = 1, \ldots, M. \tag{4.3}$$

Algoritmo 3: AdaBoost - Classificação Binária

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Inicializar: pesos iguais para todas as observações

$$\mathbf{w}_1 = [1/n, \ldots, 1/n].$$

2. Para cada $m = 1, \ldots, M$:

   (a) Construir o classificador fraco $T_m(\mathbf{x})$ baseado numa árvore CART com dois nós utilizando os pesos $\mathbf{w}_m$.

   (b) Calcular o erro de T_M

$$\epsilon_m = \frac{\sum_{i=1}^{n} w_{mi} \cdot \mathbb{1}(T_m(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{n} w_{mi}}.$$

   (c) Calcular o peso de T_M

$$\alpha_m = \log\left(\frac{1 - \epsilon_m}{\epsilon_m}\right).$$

   (d) Para cada $i = 1, \ldots, n$:

      i. Atualizar os pesos

$$w_{m+1,i} = w_{mi} \cdot \exp\left[\alpha_m \cdot \mathbb{1}(y_i \neq T_m(\mathbf{x}_i))\right].$$

      ii. Normalizar os pesos

$$w_{m+1,i} = \frac{w_{m+1,i}}{\sum_{i=1}^{n} w_{m+1,i}}.$$

Resultado: $T(\mathbf{x}) = \text{sinal}\left[\sum_{m=1}^{M} \alpha_m T_m(\mathbf{x})\right]$

A solução para essa minimização passa por um processo computacionalmente intensivo, no entanto, dependendo da função base b e/ou de L, o problema assume uma dimensão menor ajustando-se a minimizar uma única função base,

$$(\beta,\theta) = \min_{\beta,\theta} \sum_{i=1}^{n} L(y_i, \beta b(\mathbf{x}_i, \theta)). \tag{4.4}$$

Uma aproximação para solucionar (4.3) é o procedimento de incrementos adicionados progressivamente (*forward stagewise additive modeling*). Os cálculos são realizados sequencialmente e a cada adição de uma função base os parâmetros e os coeficientes já ajustados não são atualizados. A cada iteração $m = 1, \ldots, M$, o procedimento otimiza a função base $b(\mathbf{x}, \theta_m)$ e o coeficiente correspondente β_m. O resultado é adicionado à expansão atual $f_{m-1}(\mathbf{x})$ e o processo é repetido. O Algoritmo 4 apresenta o procedimento.

O modelo AdaBoost pode ser deduzido como um modelo aditivo (Friedman *et al.*, 2000) ajustado com o procedimento de incrementos adicionados progressivamente usando a função de perda exponencial

$$L(y,f(x)) = e^{-yf(x)}. \tag{4.5}$$

Utilizamos como função base $b(\mathbf{x},\theta)$ o classificador CART $T(\mathbf{x},\theta)$, em que $\theta = \{R_j,\gamma_j\}$, R_j são as regiões e γ_j as constantes que definem a predição em cada respectiva região da árvore T.

Considerando um conjunto de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$, adequando

Algoritmo 4: Procedimento de incrementos adicionados progressivamente

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Inicializar: $f_0(\mathbf{x}) = 0$
2. Para cada $m = 1, \ldots, M$:
   (a) Calcular
   $$(\beta_m, \theta_m) = \arg\min_{\beta,\theta} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i, \theta)).$$
   (b) Atualizar $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m b(\mathbf{x}, \theta_m)$.

Resultado: $\hat{f}(\mathbf{x}) = f_M(\mathbf{x})$

a expressão (4.3) para o AdaBoost

$$\begin{aligned}(\beta_m, T_m) &= \arg\min_{\beta,T} \sum_{i=1}^{n} e^{-y_i(f_{m-1}(\mathbf{x}_i) + \beta T(\mathbf{x}_i))} \\ &= \arg\min_{\beta,T} \sum_{i=1}^{n} e^{-y_i f_{m-1}(\mathbf{x}_i)} \cdot e^{-\beta y_i T(\mathbf{x}_i)},\end{aligned} \tag{4.6}$$

na qual o classificador $T_m(\mathbf{x}_i) \in \{-1, +1\}$ corresponde ao coeficiente β_m adicionado a cada iteração. Uma vez que $e^{-y_i f_{m-1}(\mathbf{x}_i)}$ não depende de β nem de $T(\mathbf{x})$, podemos considerar esse fator como um peso $w_{mi} = e^{-y_i f_{m-1}(\mathbf{x}_i)}$ que também é atualizado a cada iteração. Expandindo a soma em (4.6), para $\beta > 0$, temos

$$\begin{aligned}\sum_{i=1}^{n} w_{mi} \cdot e^{-\beta y_i T(\mathbf{x}_i)} &= e^{-\beta} \cdot \sum_{y_i = T(\mathbf{x}_i)} w_{mi} + e^{\beta} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_{mi} \\ &= e^{-\beta} \cdot \sum_{y_i = T(\mathbf{x}_i)} w_{mi} + e^{-\beta} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_{mi} \\ &\quad - e^{-\beta} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_{mi} + e^{\beta} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_{mi} \\ &= e^{-\beta} \cdot \sum_{i=1}^{n} w_{mi} + (e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{n} w_{mi} \mathbb{1}(y_i \neq T(\mathbf{x}_i)).\end{aligned} \tag{4.7}$$

De (4.7) podemos ver que

$$T_m(\mathbf{x}) = \arg\min_{T} \sum_{i=1}^{n} w_{mi} \mathbb{1}(y_i \neq T(\mathbf{x}_i)), \tag{4.8}$$

que corresponde à linha $2(a)$ do Algoritmo 3, *i.e.*, encontrar uma árvore usando os pesos w_{mi} que

minimizam o erro de predição. Usando essa expressão de T_m podemos calcular β_m:

$$\begin{aligned}
&\frac{\partial}{\partial\beta}\left\{e^{-\beta}\cdot\sum_{i=1}^{n}w_{mi}+(e^{\beta}-e^{-\beta})\cdot\sum_{i=1}^{n}w_{mi}\mathbb{1}(y_i\neq T_m(\mathbf{x}_i))\right\}=0\\
&\implies (e^{2\beta}+1)\cdot\sum_{i=1}^{n}w_{mi}\mathbb{1}(y_i\neq T_m(\mathbf{x}_i))-\sum_{i=1}^{n}w_{mi}=0\\
&\implies \beta_m=\frac{1}{2}\log\left(\frac{1-\epsilon_m}{\epsilon_m}\right),
\end{aligned}\tag{4.9}$$

em que

$$\epsilon_m=\frac{\sum_{i=1}^{n}w_{mi}\mathbb{1}(y_i\neq T_m(\mathbf{x}_i))}{\sum_{i=1}^{n}w_{mi}}.\tag{4.10}$$

A expressão do erro (4.10) corresponde à linha 2(b) do Algoritmo 3 e $\alpha_m=2\beta_m$ na linha 2(c). A expressão de atualização dos pesos é a seguinte

$$\begin{aligned}
w_{m+1,i}&=e^{-y_i f_m(\mathbf{x}_i)}\\
&=e^{-y_i(f_{m-1}(\mathbf{x}_i)+\beta_m T_m(\mathbf{x}_i))}\\
&=e^{-y_i f_{m-1}(\mathbf{x}_i)}\cdot e^{-\beta_m y_i T(\mathbf{x}_i)}\\
&=w_{mi}\cdot e^{-\beta_m y_i T(\mathbf{x}_i)}.
\end{aligned}\tag{4.11}$$

Notemos o seguinte

$$\begin{aligned}
-y_iT(\mathbf{x}_i)&=\begin{cases}+1, & \text{se } y_i\neq T(\mathbf{x}_i)\\ -1, & \text{se } y_i=T(\mathbf{x}_i)\end{cases}\\
&=+1\cdot\mathbb{1}(y_i\neq T(\mathbf{x}_i))-1\cdot(1-\mathbb{1}(y\neq T(\mathbf{x}_i)))\\
&=2\cdot\mathbb{1}(y_i\neq T(\mathbf{x}_i))-1.
\end{aligned}\tag{4.12}$$

Substituindo o resultado (4.12) na expressão (4.11), o peso atualizado é dado por

$$w_{m+1,i}=w_{mi}\cdot e^{2\beta_m\mathbb{1}(y\neq T(\mathbf{x}_i))}\cdot e^{-\beta_m}.\tag{4.13}$$

Uma vez que $\alpha_m=2\beta_m$ e o fator $e^{-\beta_m}$ não tem efeito, pois multiplica todos os pesos, fica estabelecida a conexão com a linha 2(d)i do Algoritmo 3.

4.3 AdaBoost Multiclasse

O Algoritmo 3 pode ser estendido para um problema de classificação de multi-classes: AdaBoost.M1 - Algoritmo (5). Suponha que y assuma valores em K classes, $y\in c_1,\ldots,c_K$. O algoritmo proposto por Freund e Schapire (1997) exige que o erro do classificador fraco ϵ_m seja $<1/2$ (considerando a distribuição da amostra de treino). Essa exigência pode ser difícil de ser alcançada já que para $K>2$ classes $1/K$ é o valor para um palpite aleatório. Valores de $\epsilon_m>1/2$ produzem valores de $\alpha_m<0$, Algoritmo 5 linha 2b, o que leva os pesos, linha 2c, a diminuírem para observações classificadas de forma errada enquanto deveriam aumentar.

Freund e Schapire (1997) calcula que o erro de $T(\mathbf{x}_i)$ é limitado por (4.14), em que os $\epsilon_m, m=1,\ldots,M$, são gerados conforme o Algoritmo 3 para classificação binária e o Algoritmo 5 para classificação multi-classes. Considere $erro=p(T(\mathbf{x}_i)\neq y_i)$:

$$erro\leq 2^M\prod_{m=1}^{M}\sqrt{\epsilon_m(1-\epsilon_m)}.\tag{4.14}$$

Algoritmo 5: AdaBoost.M1 - Classificação Multi-classes

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Inicializar: pesos iguais para todas as observações

$$\mathbf{w}_1 = [1/n, \ldots, 1/n].$$

2. Para cada $m = 1, \ldots, M$:

   (a) Construir o classificador fraco $T_m(\mathbf{x})$ baseado numa árvore CART com dois nós utilizando os pesos $\mathbf{w}_m$.

      i. Calcular o erro de T_M

$$\epsilon_m = \sum_{i=1}^{n} w_{mi} \cdot \mathbb{1}(T_m(\mathbf{x}_i) \neq y_i).$$

      ii. Se $\epsilon_m > 1/2$ tomar $M = m - 1$ e parar as iterações.

   (b) Calcular o peso de T_M

$$\alpha_m = \log\left(\frac{1 - \epsilon_m}{\epsilon_m}\right).$$

   (c) Para cada $i = 1, \ldots, n$:

      i. Atualizar os pesos

$$w_{m+1,i} = w_{mi} \cdot \exp\left[\alpha_m \cdot \mathbb{1}(y_i \neq T_m(\mathbf{x}_i))\right].$$

      ii. Normalizar os pesos

$$w_{m+1,i} = \frac{w_{m+1,i}}{\sum_{i=1}^{n} w_{m+1,i}}.$$

Resultado: $T(\mathbf{x}) = \arg\max_{c_k \in Y} \left[\sum_{m=1}^{M} \alpha_m \cdot \mathbb{1}(T_m(\mathbf{x}) = c_k)\right]$

Para corrigir o problema do AdaBoost.M1, Zhu *et al.* (2006) propõe uma modificação no algoritmo que preserva sua essência de forma simples e elegante: adiciona um fator $\log(K-1)$ no cálculo de α_m e retira a restrição $\epsilon_m < 1/2$. O Algoritmo é nomeado *SAMME - Stagewise Additive Modeling using a Multi-class Exponential loss function* - Algoritmo 6. No entanto, essa modificação não é arbitrária, e faz parte de uma compreensão mais profunda do funcionamento do AdaBoost, pois até o momento parece um método mais ou menos *ad hoc*, sem uma justificativa estatística ou matemática, de uma aplicação bem sucedida do princípio do PAC. Esta modelagem faz parte do encontro entre os modelos aditivos, uma função de perda e um ajuste, correção, que é efetuada em cada modelo intermediário que procura corrigir o erro cometido no modelo anterior. Note a sequência de classificadores T_m's e a correção que é aplicada nos pesos através dos α_m's procurando *corrigir* o erro cometido no passo anterior. Quando $K = 2$ o algoritmo se reduz ao AdaBoost.

4.4 Ilustração

Exemplificaremos o uso do algoritmo original (Algoritmo 3) do método AdaBoost. Os dados, $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^3, y_i \in \{-1, +1\}, n = 8$, acrescentados da coluna de pesos w_i iniciais estão especificados na Tabela 4.1.

Algoritmo 6: SAMME

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Inicializar: pesos iguais para todas as observações

$$\mathbf{w}_1 = [1/n, \ldots, 1/n].$$

2. Para cada $m = 1, \ldots, M$:

   (a) Construir o classificador fraco $T_m(\mathbf{x})$ baseado numa árvore CART com dois nós utilizando os pesos $\mathbf{w}_m$.

   (b) Calcular o erro de T_M

$$\epsilon_m = \sum_{i=1}^{n} w_{mi} \cdot \mathbb{1}(T_m(\mathbf{x}_i) \neq y_i).$$

   (c) Calcular o peso de T_M

$$\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) + \log(K-1).$$

   (d) Para cada $i = 1, \ldots, n$:

      i. Atualizar os pesos

$$w_{m+1,i} = w_{mi} \cdot \exp\left[\alpha_m \mathbb{1}(y_i \neq T_m(\mathbf{x}_i))\right].$$

      ii. Normalizar os pesos

$$w_{m+1,i} = \frac{w_{m+1,i}}{\sum_{i=1}^{n} w_{m+1,i}}.$$

Resultado: $T(\mathbf{x}) = \arg\max_{c_k \in Y} \left[\sum_{m=1}^{M} \alpha_m \cdot \mathbb{1}(T_m(\mathbf{x}) = c_k)\right]$

Tabela 4.1: *Ilustração: AdaBoost Classificação - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	+1
2	1,8	2,0	0	+1
3	0,4	6,0	1	-1
4	1,9	8,0	0	-1
5	2,5	3,6	1	-1
6	3,0	2,5	0	+1
7	4,0	5,0	1	+1
8	3,5	4,5	0	+1

O primeiro passo é construir uma árvore CART com somente dois nós terminais (*stump*). Como vimos na Secção 2.9 a melhor partição é $x_2 \leq 5{,}50$ - uma vez que todas as observações tem o mesmo peso, na Figura 4.4 temos a predição usada para preencher a coluna $T(\mathbf{x}_i)$ da Tabela 4.2. O resultado completo pode ser visto na referida Tabela 4.2.

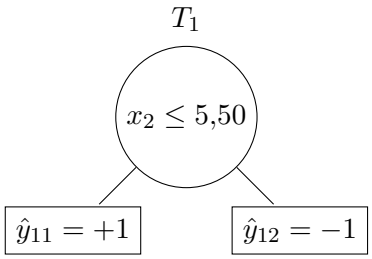


Figura 4.4: *Ilustração: AdaBoost Primeira Árvore CART.*

Tabela 4.2: *Ilustração: AdaBoost Classificação - iteração 1.*

i	x_1	x_2	x_3	y_i	w_{1i}	$T(\mathbf{x}_i)$	w_{2i}
1	0,8	3,5	1	+1	1/8	+1	1/14
2	1,8	2,0	0	+1	1/8	+1	1/14
3	0,4	6,0	1	-1	1/8	-1	1/14
4	1,9	8,0	0	-1	1/8	-1	1/14
5	2,5	3,6	1	-1	1/8	+1	7/14
6	3,0	2,5	0	+1	1/8	+1	1/14
7	4,0	5,0	1	+1	1/8	+1	1/14
8	3,5	4,5	0	+1	1/8	+1	1/14

A observação $i = 5$ foi classificada errada. Vamos calcular o ϵ_1 e o fator de peso α_1.

$$\begin{aligned} \epsilon_1 &= \sum_{i=1}^{8} \frac{1}{8} \cdot \mathbb{1}(T_1(\mathbf{x}_i) \neq y_i) = \frac{1}{8}. \\ \alpha_1 &= \log\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = \log\left(\frac{1-1/8}{1/8}\right) = \log(7). \end{aligned} \tag{4.15}$$

Ao recalcular os pesos somente os pesos das classificações erradas é que se modificam $w_{2i}^{'} = \frac{1}{8} \cdot \exp(\alpha_1 \cdot \mathbb{1}(T(\mathbf{x}_i) \neq y_i))$, o que ocorre para a observação $i = 5$:

$$w_{2i}^{'} = \frac{1}{8} \cdot \exp(\log(7)) = \frac{7}{8}, i = 5. \tag{4.16}$$

Normalizando os pesos $w_{2i} = \frac{1/8}{14/8} = \frac{1}{14}, i = 1, 2, 3, 4, 6, 7, 8$ e $w_{2i} = \frac{7/8}{14/8} = \frac{7}{14}, i = 5$, coluna w_{2i} da Tabela 4.2.

Na próxima iteração calculamos para cada partição o erro com os novos pesos. Escolhemos a partição que cujo erro é menor e calculamos o novo fator α_2. O processo segue iterativamente conforme o Algoritmo 3.

5 *Gradient Boosting*

A técnica de *gradient boosting*, desenvolvida em Friedman (2000) é inspirada nos métodos, já bastante desenvolvidos, de otimização numérica (Nocedal e Wright, 2006). O próprio nome *gradiente* sugere uma analogia com algum método que utiliza o gradiente como busca por uma solução ótima, ponto de máximo ou mínimo de uma função. A palavra boosting está associada a incrementos sucessivos na forma em que as árvores são geradas.

Não existe uma *boosted tree* final, gerada pelo método de *gradient boosting*, mas a predição é a soma de várias *boosted trees* geradas pelo método - de forma análoga a que não existe uma árvore AdaBoost, mas uma sequência de árvores que são utilizadas para compor a predição final.

5.1 O Método do Gradiente

Este método se vale de um cálculo iterativo em que a partir de um ponto inicial cada ponto subsequente na iteração está mais próximo da solução. Genericamente, o método é definido a partir da seguinte iteração

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \rho_k \mathbf{g}_k, \quad \mathbf{x}_k \in \mathbb{R}^n, k \geq 1. \tag{5.1}$$

em que ρ_k é o escalar que representa o tamanho do passo (*step length*) na direção $-\mathbf{g}_k$. Quando $\mathbf{g}_k$ é o gradiente de f:

$$\nabla f = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2,}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right), \tag{5.2}$$

o método é chamado de método do gradiente, método de Cauchy ou método do passo mais descendente. Note que $\mathbf{g}_k$ é calculado no ponto $\mathbf{x}_{k-1}$:

$$\mathbf{g}_k = - \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2,}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right]_{\mathbf{x}=\mathbf{x}_{k-1}}. \tag{5.3}$$

Sob certas condições de regularidade, *e.g.*, a função ser diferenciável, $\mathbf{x}_k$ deve convergir para um mínimo (máximo) local - que pode ser global ou não. Os valores de ρ_k são calculados a cada passo utilizando a estratégia da busca linear (*linear search*) que consiste resolver a seguinte expressão

$$\min_{\rho > 0} f(\mathbf{x}_{k-1} - \rho \mathbf{g}_k). \tag{5.4}$$

Para ilustrar o método, vamos tomar um exemplo unidimensional, *i.e.*, f definida com apenas uma variável. Sejam $f(x) = x^2 - x - 2$, $\rho_k = \rho = 0{,}25$ constante, e $g_k = -\frac{\partial f(x)}{\partial x}\Big|_{x=x_{k-1}}$, queremos encontrar o ponto em que f atinge o mínimo. Note que $g_k = -(2x_{k-1} - 1)$, partindo de um valor inicial $x_0 = 1$, a Tabela 5.1 apresenta a sequência de cálculos.

Tabela 5.1: *Método do Gradiente para* $f(x) = x^2 - x - 2, \rho = 0{,}25$

k	x_k	g_k	ρg_k
0	1,00000		
1	0,75000	−1,00000	−0,25000
2	0,62500	−0,50000	−0,12500
3	0,56250	−0,25000	−0,06250
4	0,53125	−0,12500	−0,03125
5	0,51563	−0,06250	−0,01563
6	0,50781	−0,03125	−0,00781
7	0,50391	−0,01563	−0,00391
8	0,50195	−0,00781	−0,00195
9	0,50098	−0,00391	−0,00098
10	0,50049	−0,00195	−0,00049

Neste caso, é claro que, é possível calcular algebricamente o ponto de mínimo $x^\star$ de f:

$$\frac{\partial f}{\partial x} = 0 \implies 2x^\star - 1 = 0 \implies x^\star = 0{,}5. \tag{5.5}$$

Usando o método do gradiente, a Tabela 5.1 mostra que em 10 iterações $x_{10} \approx x^\star = 0{,}5$. Graficamente, a Figura 5.1 mostra como x_k se aproxima de $x^\star$.

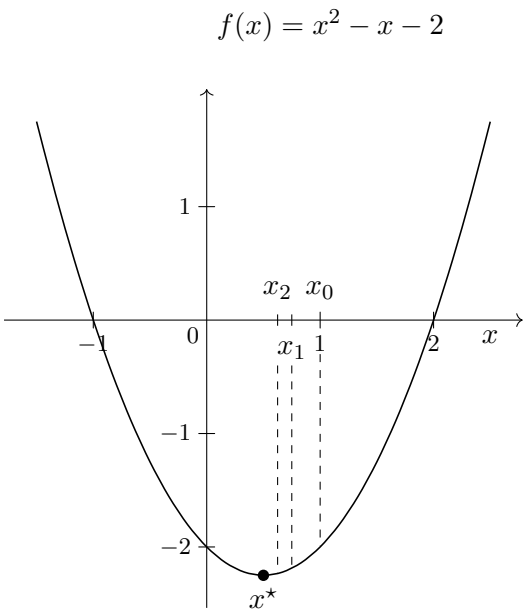


Figura 5.1: *Método do Gradiente para* $f(x) = x^2 - x - 2, \rho = 0{,}25$.

Para a construção das árvores usando *gradient boosting* algumas modificações são necessárias, no entanto, permanece a ideia de utilizar o gradiente como incrementos sucessivos (*passos* ou *boosts*).

5.2 *Boosted Trees*

Uma *boosted tree* é construída a partir de um processo de otimização, uma função de perda deve ser especificada e algoritmo de cálculo como o descrito na Secção 4.2 deve ser utilizado.

Considere T uma árvore CART que separa a variável resposta de acordo com as regiões R_j, $j =$

$1, \ldots, J$, seja γ_j a constante de valor preditivo em cada região, então

$$T(\mathbf{x}) = \gamma_j, \text{para } \mathbf{x} \in R_j, j = 1, \ldots, J, \tag{5.6}$$

ou, de outra forma

$$T(\mathbf{x},\theta) = \sum_{j=1}^{J} \gamma_j \mathbb{1}(\mathbf{x} \in R_j), \tag{5.7}$$

em que $\theta = \{R_j, \gamma_j\}$ são os parâmetros que definem a árvore, J é tratado como um meta-parâmetro. Seja $I_j = \{i|\mathbf{x}_i \in R_j\}$, para encontrar θ precisamos minimizar

$$\hat{\theta} = \arg\min_{\theta} \sum_{j=1}^{J} \sum_{i \in I_j} L(y_i, \gamma_j), \tag{5.8}$$

em que L é a função de perda. Estimar θ é estimar a estrutura da própria árvore e o modelo CART pode ser utilizado para encontrar $\hat{\theta}$. Se necessário, dependendo da complicação do cálculo de L, uma modificação em (5.8) pode ser necessária e L pode ser aproximado por $\tilde{L}$:

$$\tilde{\theta} = \arg\min_{\theta} \sum_{j=1}^{J} \sum_{i \in I_j} \tilde{L}(y_i, \gamma_j), \tag{5.9}$$

e tomando $\hat{R}_j = \tilde{R}_j$, γ_j pode ser estimado mais precisamente utilizando L.

Um modelo de *gradient boosting* é definido como a soma de várias boosted trees. Usando a notação dos modelos aditivos, Secção 4.2, $\beta_m = 1$ e $b = T$:

$$f_M(\mathbf{x}) = \sum_{m=1}^{M} T(\mathbf{x}, \theta_m). \tag{5.10}$$

Utilizando o procedimento descrito no Algoritmo 4, Secção 4.2, em cada passo é necessário resolver

$$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i, \theta_m)). \tag{5.11}$$

para $\theta_m = \{R_{jm}, \gamma_{jm}\}$,$j = 1, \ldots, J_m$, da próxima árvore, dado o resultado do modelo atual $f_{m-1}(\mathbf{x})$.

Uma vez que as regiões são encontradas, é relativamente fácil encontrar as constantes γ_{jm} em cada região R_{jm}. Seja $I_{jm} = \{i|\mathbf{x}_i \in R_{jm}\}$, então

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{i \in I_{jm}} L(y_i, f_{m-1}(\mathbf{x}_i) + \gamma). \tag{5.12}$$

5.3 Otimização via *Gradient Boosting*

Dado um conjunto de treinamento $\mathcal{D} = \{(\mathbf{x}_i, y_i),\ \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ |\mathcal{D}| = n\}$, genericamente, usamos uma função de perda L para medir a discrepância entre a resposta y e a previsão $f(\mathbf{x})$ para construir uma única árvore:

$$\mathcal{L}(f) = \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)). \tag{5.13}$$

O objetivo é minimizar $L(f)$ em relação à f. Vamos pensar em $\mathbf{f}$ como

$$\mathbf{f} = \mathbf{f}(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)). \tag{5.14}$$

Se calcularmos o gradiente em cada *ponto* $f(\mathbf{x}_i)$ de $\mathbf{f}$, podemos obter um resultado análogo à (5.1):

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_{k-1}\mathbf{g}_m, \tag{5.15}$$

considerando um valor inicial $\mathbf{f}_0$ e o vetor $\mathbf{g}_m \in \mathbb{R}^n$ cujos componentes são dados por

$$g_{im} = \left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i)} = \left[\frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)}\right] \tag{5.16}$$

e o tamanho do passo é a solução de

$$\rho_m = \arg\min_{\rho} L(\mathbf{f}_{m-1} - \rho\mathbf{g}_m). \tag{5.17}$$

A cada iteração $-\mathbf{g}_m$ é a direção em $\mathbb{R}^n$ para a qual $\mathcal{L}(\mathbf{f})$ decresce mais rapidamente em $\mathbf{f} = \mathbf{f}_{m-1}$.

A estratégia definida em (5.11) busca a minimização de θ, as predições da árvore $T(\mathbf{x}_i, \theta_m)$ são análogas ao negativo dos gradientes calculados em (5.16). No entanto, as predições das árvores são restringidas pelas regiões - *minimizadas* em direção às regiões - enquanto que o gradiente não tem restrições. A solução de (5.12) é análoga ao procedimento de busca linear (5.17) com a diferença de que a busca de (5.12) é dentro de cada região R_{jm}.

Adicionalmente se desejamos minimizar θ_m a melhor estratégia desenvolvida em Hastie *et al.* (2009) é a escolha do gradiente - devido, em parte, pela facilidade de trabalhar com vários tipos de funções de perda. Entretanto, (5.16) está definida sobre os dados $\mathbf{x}_i$ da base de treino, enquanto que o objetivo é generalizar o procedimento todo para dados que não estão na amostra. A solução é ajustar o modelo da árvore T usando os negativos dos gradientes ao invés dos dados originais. Assim, a estativa de θ_m com o negativo do gradiente vai ao encontro de do objetivo de encontrar as *melhores* regiões. Isto nos leva a

$$\tilde{\theta}_m = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_1) + T(\mathbf{x}_i, \theta_m)). \tag{5.18}$$

As regiões $\tilde{R}_{jm}$ não são idênticas as regiões R_{jm}, mas servem ao propósito, além do que tanto a construção da árvore quanto o procedimento de incrementos adicionados progressivamente são também aproximações. Após a construção da árvore (5.18), as constantes γ_{jm} em cada região são calculadas usando (5.12).

5.3.1 Árvores de Regressão

O Algoritmo 7 delineia o procedimento para a construção das árvores e consequente predição. Vamos utilizar a perda quadrática $L(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2$ para exemplificar o funcionamento do algoritmo.

O valor inicial na linha 1 pode ser encontrado minimizando-se

$$\sum_{i=1}^{n} L(y_i,\gamma) = \sum_{i=1}^{n} \frac{1}{2}(y_i - \gamma)^2, \tag{5.19}$$

cujo resultado é $\gamma = \bar{y}$.

Algoritmo 7: *gradient boosting* - Árvores de Regressão.

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathbb{R}, |\mathcal{D}| = n\}$

1. Inicializar: $f_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$.
2. Para cada $m = 1, \ldots, M$:
   (a) Para cada $i = 1, \ldots, n$ calcular
   $$\tilde{r}_{im} = -\left[\frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)}\right].$$
   (b) Ajustar uma árvore de regressão cuja variável resposta é $\tilde{r}_{im}$ encontrando as regiões terminais $R_{jm}, j = 1, \ldots, J_m$.
   (c) Para cada $j = 1, \ldots, J_m$ calcular
   $$\gamma_{jm} = \arg\min_\gamma \sum_{i \in I_{jm}} L(y_i, f_{m-1}(\mathbf{x}_i) + \gamma).$$
   (d) Atualizar $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}(\mathbf{x} \in R_{jm})$.

Resultado: $T(\mathbf{x}) = f_M(\mathbf{x})$

Na linha $2(a)$ temos o cálculo do negativo do gradiente, $\tilde{r}_{im}$ é chamado de pseudo-resíduo:

$$\tilde{r}_{im} = -g_{im} = -\left[\frac{\partial \frac{1}{2}(y_i - f_{m-1}(\mathbf{x}_i))^2}{\partial f_{m-1}(\mathbf{x}_i)}\right] = -[-2 \cdot \frac{1}{2}(y_i - f_{m-1}(\mathbf{x}_i))] = y_i - f_{m-1}(\mathbf{x}_i), \tag{5.20}$$

que coincide com o resíduo para essa função de perda - o termo multiplicador $\frac{1}{2}$ não altera o resultado final. O parâmetro γ_{jm} na linha $2(c)$ pode ser estimado tomando-se a derivada e solucionando a equação

$$\sum_{i \in I_{jm}} \frac{\partial L(y_i, f_{m-1(\mathbf{x}_i)} + \gamma)}{\partial \gamma} = 0. \tag{5.21}$$

Para a perda quadrática

$$\begin{aligned} \frac{\partial L(y_i, f_{m-1(\mathbf{x}_i)} + \gamma)}{\partial \gamma} &= \frac{\partial}{\partial \gamma}(y_i - (f_{m-1}(\mathbf{x}_i) + \gamma))^2 \\ &= -2(y_i - f_{m-1}(\mathbf{x}_i) - \gamma). \end{aligned} \tag{5.22}$$

Considere $n_{jm} = |I_{jm}|$, o número de observações na região R_{jm}. Igualando (5.22) a zero e somando as observações em R_{jm}, temos

$$\sum_{i \in I_{jm}} -2(y_i - f_{m-1}(\mathbf{x}_i) - \gamma) = 0, \tag{5.23}$$

logo,

$$\gamma_{jm} = \gamma = \frac{1}{n_{jm}} \sum_{i \in I_{jm}} (y_i - f_{m-1}(\mathbf{x}_i)), \tag{5.24}$$

que é a média dos resíduos na região R_{jm}. Outras funções de perda podem ser utilizadas para construção de árvores de regressão, vide Tabela 5.2.

Tabela 5.2: *Funções de perda e Gradientes para árvores de regressão.*

Tipo	Função de perda	$-\partial L(y_i,f(\mathbf{x}_i))/\partial f(\mathbf{x}_i)$
Quadrática	$\frac{1}{2}[y_i - f(\mathbf{x}_i)]^2$	$y_i - f(\mathbf{x}_i)$
Desvio absoluto	$\lvert y_i - f(\mathbf{x}_i)\rvert$	$\text{sinal}[y_i - f(\mathbf{x}_i)]$
Huber	Quadrática ou Desvio absoluto	$\begin{cases} y_i - f(\mathbf{x}_i) & \text{se } \lvert y_i - f(\mathbf{x}_i)\rvert \le \delta_m \\ \text{sinal}[y_i - f(\mathbf{x}_i)] & \text{se } \lvert y_i - f(\mathbf{x}_i)\rvert > \delta_m \end{cases}$ em que $\delta_m = \alpha$-ésimo-quantil$\{\lvert y_i - f(\mathbf{x}_i)\rvert\}$

5.3.2 Árvores de Classificação

As funções de perda utilizadas para as boosted trees para regressão, Tabela 5.2, não são adequadas para árvores de classificação. Para as árvores para classificação utilizamos a função de perda multinomial *deviance* ou *log loss* que é dada pelo negativo do log da função de verossimilhança.

Suponha que a variável resposta y assume valores em K classes: $y \in \{c_1, c_2, \ldots, c_K\}$. Seja $y_k = \mathbb{1}(y = c_k), k = 1, \ldots, K$. Seja

$$\begin{aligned} p_k(\mathbf{x}) &= P(y = c_k), \text{ para } k = 1, \ldots, K-1 \quad \text{e} \\ p_K(\mathbf{x}) &= 1 - \sum_{k=1}^{K-1} p_k(\mathbf{x}). \end{aligned} \tag{5.25}$$

A função de verossimilhança é dada por

$$l(y|\theta = \{p_1(\mathbf{x}), \ldots, p_K(\mathbf{x})\} \equiv p(\mathbf{x})) = \prod_{k=1}^{K} p_k(\mathbf{x})^{y_k}, \tag{5.26}$$

e função de perda é por

$$L(y, p(\mathbf{x})) = -\log(l(y|p(\mathbf{x}))) = -\sum_{k=1}^{K} y_k \log p_k(\mathbf{x}). \tag{5.27}$$

Usando a transformação *logit*, podemos linearizar o log das chances e construir K modelos

$$\begin{aligned} f_1(\mathbf{x}) &= \log \frac{p_1(\mathbf{x})}{p_K(\mathbf{x})} \\ f_2(\mathbf{x}) &= \log \frac{p_2(\mathbf{x})}{p_K(\mathbf{x})} \\ &\vdots \\ f_K(\mathbf{x}) &= \log \frac{p_K(\mathbf{x})}{p_K(\mathbf{x})}, \end{aligned} \tag{5.28}$$

e calcular $p_k(\mathbf{x})$ a partir destas razões

$$\begin{aligned} p_1(\mathbf{x}) &= e^{f_1(\mathbf{x})} p_K(\mathbf{x}) \\ p_2(\mathbf{x}) &= e^{f_2(\mathbf{x})} p_K(\mathbf{x}) \\ &\vdots \\ p_K(\mathbf{x}) &= e^{f_K(\mathbf{x})} p_K(\mathbf{x}). \end{aligned} \tag{5.29}$$

Somando

$$\begin{aligned} p_1(\mathbf{x}) + p_2(\mathbf{x}) + \cdots + p_K(\mathbf{x}) = 1 \implies \\ e^{f_1(\mathbf{x})} p_K(\mathbf{x}) + e^{f_2(\mathbf{x})} p_K(\mathbf{x}) + \cdots + e^{f_K(\mathbf{x})} p_K(\mathbf{x}) = 1 \implies \\ p_K(\mathbf{x}) = \frac{1}{\sum_{l=1}^{K} e^{f_l(\mathbf{x})}}, \end{aligned} \tag{5.30}$$

podemos obter

$$p_k(\mathbf{x}) = \frac{e^{f_k(\mathbf{x})}}{\sum_{l=1}^{K} e^{f_l(\mathbf{x})}}, k = 1, \ldots, K. \tag{5.31}$$

A função de perda (5.27) pode ser de maneira equivalente escrita da seguinte maneira:

$$\begin{aligned} L(y, p(\mathbf{x})) &= -\sum_{k=1}^{K} y_k \log \left(\frac{e^{f_k(\mathbf{x})}}{\sum_{l=1}^{K} e^{f_l(\mathbf{x})}} \right) \\ &= -\sum_{k=1}^{K} y_k f_k(\mathbf{x}) + \log \left(\sum_{l=1}^{K} e^{f_l(\mathbf{x})} \right). \end{aligned} \tag{5.32}$$

O modelo que construiremos será aditivo na razão das chances e podemos cambiar facilmente para as probabilidades conforme (5.31). Mais do que encontrar a classe modal, a qual é simplesmente

$$T(x) = c_k = \arg \max_{c_k} p_k(\mathbf{x}), \tag{5.33}$$

vamos modelar $p_k(\mathbf{x}), k = 1, \ldots, K$ induzidas pelo log da razão das chances.

O Algoritmo 8, note que para não sobrecarregar a notação omitimos a iteração m dos índices de $p_k(\mathbf{x})$, delineia o procedimento para a construção de K classificadores, supondo que assume valores em K classes: $y \in \{c_1, \ldots, c_K\}$ e $y_k = \mathbb{1}(y = c_k), k = 1, \ldots, K$. Note que o último classificador é redundante uma vez que $\sum_{k=1}^{K} p_k(\mathbf{x}) = 1$.

O algoritmo inicializa cada classificador $f_{k0} = 0, k = 1, \ldots, K$, o que equivale a assumir probabilidades iguais para cada classe c_k:

$$p_k(\mathbf{x}) = \frac{e^{f_k(\mathbf{x})}}{\sum_{\ell=1}^{K} e^{f_\ell(\mathbf{x})}} = \frac{e^0}{\sum_{\ell=1}^{K} e^0} = \frac{1}{K} \tag{5.34}$$

Algoritmo 8: *gradient boosting* - Árvores de Classificação.

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \{1, 2, \ldots, K\}, |\mathcal{D}| = n\}$

1. Inicializar: $f_{k0}(\mathbf{x}) = 0, k = 1, \ldots, K$.
2. Para cada $m = 1, \ldots, M$:
   (a) Calcular
   $$p_k(\mathbf{x}) = \frac{e^{f_k(\mathbf{x})}}{\sum_{l=1}^{K} e^{f_l(\mathbf{x})}}, k = 1, \ldots, K.$$
   (b) Para $k = 1, \ldots, K-1$:
      i. Calcular $\tilde{r}_{ikm} = y_{ik} - p_k(\mathbf{x}_i), i = 1, \ldots, n$.
      ii. Ajustar uma árvore de regressão para $r_{ikm}, i = 1, \ldots, n$, cujo resultado são as regiões terminais $R_{jkm}, j = 1, \ldots, J_m$.
      iii. Calcular
      $$\gamma_{jkm} = \frac{\sum_{i \in I_{jkm}} r_{ikm}}{\sum_{i \in I_{jkm}} p_{jkm}(\mathbf{x}_i)(1 - p_{jkm}(\mathbf{x}_i))}.$$
      iv. Atualizar $f_{km}(\mathbf{x}) = f_{k,m-1(\mathbf{x})} + \sum_{j=1}^{J_m} \gamma_{jkm} \mathbb{1}(\mathbf{x} \in R_{jkm})$.

Resultado: $T_k(\mathbf{x}) = f_{kM}(\mathbf{x}), k = 1, \ldots, K$

O pseudo-resíduo $\tilde{r}_{ikm}$ é obtido a partir do gradiente da função de perda (5.32):

$$\begin{aligned} \tilde{r}_{ikm} &= -\frac{\partial}{\partial f_k(\mathbf{x}_i)} \left[-\sum_{k=1}^{K} y_{ik} f_k(\mathbf{x}_i) + \log \left(\sum_{\ell=1}^{K} e^{f_\ell(\mathbf{x}_i)} \right) \right]_{f = f_{k,m-1}} \\ &= -\left[-y_{ik} + \frac{e^{f_k(\mathbf{x}_i)}}{\sum_{\ell=1}^{K} e^{f_\ell(\mathbf{x}_i)}} \right]_{f = f_{k,m-1}} \\ &= y_{ik} - p_k(\mathbf{x}_i), \end{aligned} \tag{5.35}$$

γ_{jkm} pode ser calculado minimizando-se

$$\sum_{i \in I_{jkm}} L(y_{ik}, f_k(\mathbf{x}_i) + \gamma) = \sum_{i \in I_{jkm}} \left[-\sum_{k=1}^{K} y_{ik}(f_k(\mathbf{x}_i) + \gamma) + \log \left(\sum_{l=1}^{K} e^{f_l(\mathbf{x}_i) + \gamma} \right) \right]. \tag{5.36}$$

No entanto, a expressão (5.36) não pode ser minimizada diretamente em γ, por esse motivo vamos utilizar o polinômio de Taylor de segunda ordem como aproximação para $L(y_{ik}, f_k(\mathbf{x}) + \gamma)$ para encontrar o valor de γ:

$$\begin{aligned} \sum_{i \in I_{jkm}} L(y_{ik}, f_k(\mathbf{x}_i) + \gamma) \approx \sum_{i \in I_{jkm}} [L(y_{ik}, f_k(\mathbf{x}_i)) &+ \frac{\partial}{\partial f_k(\mathbf{x}_i)} L(y_{ik}, f_k(\mathbf{x}_i)) \cdot \gamma \\ &+ \frac{1}{2} \frac{\partial^2}{\partial^2 f_k(\mathbf{x}_i)} L(y_{ik}, f_k(\mathbf{x}_i)) \cdot \gamma^2]. \end{aligned} \tag{5.37}$$

Derivando (5.37) em relação à γ e igualando a 0, temos

$$\gamma_{jkm} = \hat{\gamma} = -\frac{\sum_{i \in I_{jkm}} \frac{\partial}{\partial f_k(\mathbf{x}_i)} L(y_{ik}, f_k(\mathbf{x}_i))}{\sum_{i \in I_{jkm}} \frac{\partial^2}{\partial^2 f_k(\mathbf{x}_i)} L(y_{ik}, f_k(\mathbf{x}))}. \tag{5.38}$$

O numerador de (5.38) é conhecido de (5.35): $y_{ik} - p_k(\mathbf{x}_i)$. O denominador é dado por

$$\begin{aligned}\sum_{i\in I_{jkm}} \frac{\partial^2}{\partial^2 f_k(\mathbf{x}_i)} L(y_{ik}, f_k(\mathbf{x}_i)) &= \sum_{i\in I_{jkm}} \frac{\partial}{\partial f_k(\mathbf{x}_i)} \left[y_{ik} - p_k(\mathbf{x}_i)\right] \\ &= \sum_{i\in I_{jkm}} \frac{\partial}{\partial f_k(\mathbf{x}_i)} \left[y_{ik} - \frac{e^{f_k(\mathbf{x}_i)}}{\sum_{l=1}^K e^{f_l(\mathbf{x}_i)}}\right] \\ &= \sum_{i\in I_{jkm}} -\frac{\partial}{\partial f_k(\mathbf{x}_i)} \left[e^{f_k(\mathbf{x}_i)} \cdot \left(\sum_{l=1}^K e^{f_l(\mathbf{x}_i)}\right)^{-1}\right] \\ &= \sum_{i\in I_{jkm}} p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i)).\end{aligned} \tag{5.39}$$

Combinando (5.35) e (5.39), chegamos à seguinte solução

$$\gamma_{jkm} = \hat{\gamma} = \frac{y_{ik} - p_k(\mathbf{x}_i)}{\sum_{i\in I_{ikm}} p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i))}. \tag{5.40}$$

Usando (5.31) podemos reconstruir as probabilidades

$$p_k(\mathbf{x}) = \frac{e^{T_k(\mathbf{x})}}{\sum_{\ell=1}^K e^{T_\ell(\mathbf{x})}}. \tag{5.41}$$

5.4 Regularização

Ambos Algoritmos 7 e 8 para árvores de regressão e classificação, respectivamente, na forma em que estão especificados podem levar a um sobreajuste. No compromisso entre viés e variância, alguns parâmetros podem ser ajustados para evitar o sobreajuste.

Além de controlar o parâmetro M que define a quantidade de árvores é possível também limitar a contribuição de cada árvore para o modelo final. As linhas $2(d)$ e $2(b)iv$, destes algoritmos, respectivamente, podem ser modificadas de maneira que o *aprendizado* não seja tão rápido. Um fator de escala $0 < \nu < 1$ que limita a contribuição de cada árvore para o modelo final pode ser inserido. Estas linhas se tornam, respectivamente

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu \cdot \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}(\mathbf{x} \in R_{jm}) \tag{5.42}$$

e

$$f_{km}(\mathbf{x}) = f_{k,m-1(\mathbf{x})} + \nu \cdot \sum_{j=1}^{J_m} \gamma_{jkm} \mathbb{1}(\mathbf{x} \in R_{jkm}). \tag{5.43}$$

Existe um compromisso entre M e ν. Quanto menor o valor de ν maior o valor de M e vice-versa. Hastie *et al.* (2009) sugere um valor para $\nu < 0.1$ e controlar M parando o ajuste quando a diminuição no erro de classificação é desprezível. Friedman (2000) mostra através diversos estudos empíricos que $M \gtrsim 200$ torna a diminuição do erro desprezível - utilizando $\nu < 0{,}125$.

O número de nós terminais, meta-parâmetro J, também pode ser controlado. Se J variar livremente na construção de cada árvore, pode levar a árvores muito grandes nas primeiras iterações e árvores muito pequenas a medida que o número de iterações aumenta. $J = 2$ leva às árvores com somente dois nós terminais - stumps. Friedman (2000) sugere $4 \leq J \leq 8$. Notando que é possível testar diversos ajustes nesse intervalo ao procurar por um melhor ajuste, no entanto, geralmente $J \simeq 6$ é suficiente para obter um bom ajuste.

Outra proposta é a modificação que leva o nome de *Stochastic Gradient Boosting* (Friedman,

2002). Esta abordagem utiliza a cada iteração uma fração η das observações ao invés do conjunto de treino completo. Um valor típico para η é $\frac{1}{2}$, embora para grandes valores de n, η pode ser substancialmente menor. Uma combinação de ν e η pode levar a uma melhora na performance do modelo (Friedman, 2002).

5.5 Importância das Variáveis

Hastie *et al.* (2009) sugere o mesmo cálculo da importância das variáveis que utilizamos nas florestas aleatórias, Secção 3.6 com uma modificação: utiliza-se o quadrado do decréscimo da impureza no lugar do decréscimo da impureza.

Para um modelo usando *gradient boosting* com M árvores $\{T_1, \ldots, T_M\}$, a medida de importância de uma variável X_ℓ é a média dos quadrados dos decréscimos de impureza, considerados os nós $t_{mj}, j = 1, \ldots, J_m - 1$, em que a variável é selecionada em T_m, ponderados pela proporção de observações em relação do total alocadas em cada respectivo nó. Seja $I_{mj_\ell} = \{j | \Delta I(s^\star, t_{mj}) = \Delta I(s^\ell, t_{mj}), j = 1, \ldots, J_m - 1\}$ o conjunto dos índices dos nós em que X_ℓ é a variável selecionada para particionar t_{mj}. A importância de X_ℓ é dada por

$$\mathcal{I}mp(X_\ell) = \frac{1}{M} \sum_{m=1}^{M} \sum_{j \in I_{mj_\ell}} p(t_{mj}) \cdot \Delta I^2(s^\ell, t_{mj}), \tag{5.44}$$

em que $p(t_{mj}) = \frac{|t_{mj}|}{n}$ é a proporção de amostras alocadas em t_{mj}.

Quando o problema é de classificação em K classes, temos um modelo para cada classe $T_k(\mathbf{x}), k = 1, \ldots, K$

$$T_k(\mathbf{x}) = \sum_{m=1}^{M} T_{mk}(\mathbf{x}). \tag{5.45}$$

Em cada modelo k, para a variável X_ℓ, temos

$$\mathcal{I}mp(X_{\ell k}) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}mp_k(X_\ell) \tag{5.46}$$

que representa a importância de X_ℓ em separar as amostras da classe k das outras classes usando o modelo T_k. A importância geral de X_ℓ pode ser obtida tomando-se a média entre todas as classes

$$Imp(X_\ell) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{I}mp(X_{lk}). \tag{5.47}$$

Como nas florestas aleatórias, cf. Secção 3.6, uma medida relativa, de modo que a importância máxima seja majorada em 100, pode ser derivada:

$$\mathcal{I}(X_\ell) = 100 \times \frac{\mathcal{I}mp(X_\ell)}{\max_k \mathcal{I}mp(X_k)}. \tag{5.48}$$

Assim, a variável mais importante assume o valor 100 e as demais permanecem no intervalo de 0 a 100.

5.6 Ilustração

Árvores de Regressão

Para ilustrar o Algoritmo *gradient boosting* Regressão. As árvores serão geradas somente com duas folhas. Seguindo o Algoritmo 7, a função de perda quadrática $L(y_i, \gamma) = (y_i - \gamma)^2$ será utilizada.

A predição inicial para todas as observações $f_0(\mathbf{x}) = \gamma_0$ pode ser facilmente calculada. A Tabela 5.3 contém os dados para uso.

Tabela 5.3: *Ilustração:* gradient boosting *Regressão - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
2	1,8	2,0	0	4,0
3	0,4	6,0	1	2,5
4	1,9	8,0	0	1,5
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0
7	4,0	5,0	1	0,8
8	3,5	4,5	0	0,4

A predição inicial γ_0 pode ser calculada encontrando

$$\arg\min_{\gamma_0} \sum_{i=1}^{8} (y_i - \gamma_0)^2 \implies \gamma_0 = \bar{y} = \frac{5{,}0 + \cdots + 0{,}4}{8} = 3{,}96. \tag{5.49}$$

Os pseudo-resíduos $\tilde{r}_{i1} = y_i - \gamma_0$ para a construção da árvore estão na Tabela 5.4.

Tabela 5.4: *Ilustração:* gradient boosting *Regressão - Pseudo-resíduos iteração 1.*

i	x_1	x_2	x_3	$\tilde{r}_{i1}$
1	0,8	3,5	1	1,04
2	1,8	2,0	0	0,04
3	0,4	6,0	1	-1,46
4	1,9	8,0	0	-2,46
5	2,5	3,6	1	5,54
6	3,0	2,5	0	4,04
7	4,0	5,0	1	-3,16
8	3,5	4,5	0	-3,56

A partição $x_2 \leq 4{,}05$ é a melhor partição, cujo resultado é apresentado na Figura 5.2.

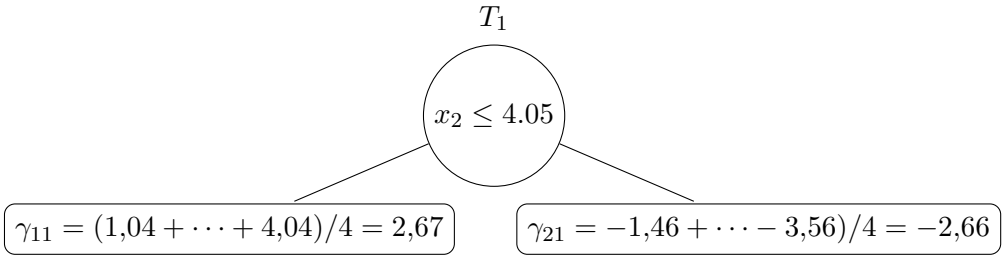


Figura 5.2: *Ilustração: Grandiente Boosting Regressão - Primeira Iteração.*

Atualizando a predição $f_1(\mathbf{x})$ com um fator de regularização $\nu = 0{,}1$:

$$f_1(\mathbf{x}_i) = \begin{cases} \gamma_0 + \nu \cdot \gamma_{11}, & i \in \{1, 2, 5, 6\} \\ \gamma_0 + \nu \cdot \gamma_{21}, & i \in \{3, 4, 7, 8\} \end{cases} \tag{5.50}$$

i.e.,

$$f_1(\mathbf{x}_i) = \begin{cases} 3{,}96 + 0{,}1 \cdot 2{,}67 = 4{,}23, & i \in \{1,2,5,6\} \\ 3{,}96 + 0{,}1 \cdot (-2{,}66) = 3{,}69, & i \in \{3,4,7,8\} \end{cases} \tag{5.51}$$

Agora com as novas predições podemos calcular os pseudo-resíduos $\tilde{r}_{i2} = y_i - f_1(\mathbf{x}_i)$ para a próxima iteração, Tabela 5.5. A árvore resultante está na Figura 5.3.

Tabela 5.5: *Ilustração:* gradient boosting *Regressão - Pseudo-resíduos iteração 2.*

i	x_1	x_2	x_3	$\tilde{r}_{i2}$
1	0,8	3,5	1	0,77
2	1,8	2,0	0	-0,23
3	0,4	6,0	1	-1,19
4	1,9	8,0	0	-2,19
5	2,5	3,6	1	5,27
6	3,0	2,5	0	3,77
7	4,0	5,0	1	-2,89
8	3,5	4,5	0	-3,29

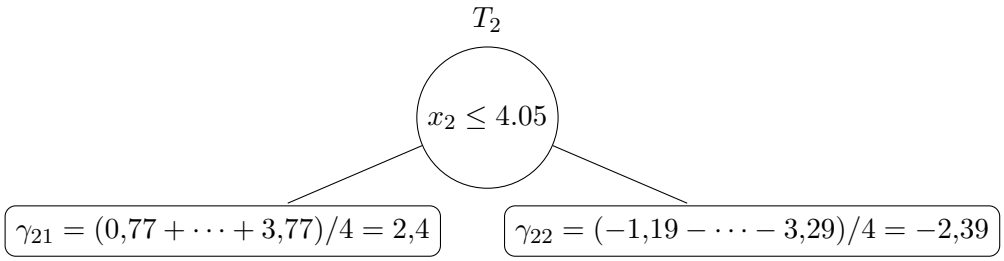


Figura 5.3: *Ilustração: Grandiente Boosting Regressão - Segunda Iteração.*

As novas predições $f_2(\mathbf{x})$ são

$$f_2(\mathbf{x}_i) = \begin{cases} f_1(\mathbf{x}_i) + \nu \cdot \gamma_{21} & i \in \{1,2,5,6\} \\ f_1(\mathbf{x}_i) + \nu \cdot \gamma_{22} & i \in \{3,4,7,8\} \end{cases} \tag{5.52}$$

i.e.,

$$f_2(\mathbf{x}_i) = \begin{cases} 4{,}23 + 0{,}1 \cdot 2{,}4 = 4{,}47 & i \in \{1,2,5,6\} \\ 3{,}69 + 0{,}1 \cdot (-2{,}39) = 3{,}45 & i \in \{3,4,7,8\} \end{cases} \tag{5.53}$$

O processo é repetido pelo número de iterações M acordado.

Árvores de Classificação

A ilustração do *gradient boosting* Classificação será feita com base no conjunto de dados disponível na Tabela 5.6. Construíremos árvores com somente duas folhas e usaremos a função de perda o negativo do log da função de verossimilhança. A variável resposta é binária $y \in \{0,1\}$.

Tabela 5.6: *Ilustração* gradient boosting *Classificação - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
2	1,8	2,0	0	1
3	0,4	6,0	1	0
4	1,9	8,0	0	0
5	2,5	3,6	1	0
6	3,0	2,5	0	1
7	4,0	5,0	1	1
8	3,5	4,5	0	1

Seguindo o Algoritmo 8 a árvore deve ser construída a partir da razão das chances, necessitamos somente de uma árvore, pois temos somente duas classes. Como previsão inicial $f_{01}(\mathbf{x}) = f_{11}(\mathbf{x}) = 0$, dessa forma as probabilidades iniciais para as classes são $p_0(\mathbf{x}) = p_1(\mathbf{x}) = 0{,}5$.

Os pseudo-resíduos $\tilde{r}_i = y_i - p_i$ para a primeira iteração estão na última coluna da Tabela 5.7.

Tabela 5.7: *Ilustração:* gradient boosting *Classificação - Pseudo-resíduos iteração 1.*

i	x_1	x_2	x_3	y_i	p_i	$\tilde{r}_{1i}$
1	0,8	3,5	1	1	0,5	0,5
2	1,8	2,0	0	1	0,5	0,5
3	0,4	6,0	1	0	0,5	-0,5
4	1,9	8,0	0	0	0,5	-0,5
5	2,5	3,6	1	0	0,5	-0,5
6	3,0	2,5	0	1	0,5	0,5
7	4,0	5,0	1	1	0,5	0,5
8	3,5	4,5	0	1	0,5	0,5

Usando a melhor partição $x_2 \leq 5{,}5$, obtemos o resultado apresentado na Figura 5.4 em que

$$\begin{aligned} \gamma_{11} &= \frac{0{,}5 + 0{,}5 - 0{,}5 + 0{,}5 + 0{,}5 + 0{,}5}{6 \times 0{,}5(1 - 0{,}5)} = \frac{2}{1{,}5} = 1{,}33, \\ \gamma_{21} &= \frac{-0{,}5 - 0{,}5}{2 \times 0{,}5(1 - 0{,}5)} = \frac{-1}{0{,}5} = -2{,}0, \end{aligned} \tag{5.54}$$

e as estimativas das probabilidades (lembrando que $f_{01}(\mathbf{x}) = f_{11}(\mathbf{x}) = 0$)

$$\begin{aligned} p_{11} &= \frac{e^{1{,}33}}{e^{1{,}33} + e^{0}} = 0{,}79, \\ p_{21} &= \frac{e^{-2{,}0}}{e^{-2{,}0} + e^{0}} = 0{,}12. \end{aligned} \tag{5.55}$$

Na próxima iteração calculamos as novas estimativas conforme a Tabela 5.8. Note como as probabilidades se aproximam dos valores observados - a exceção é $i = 5$ que ainda não está bem classificado.

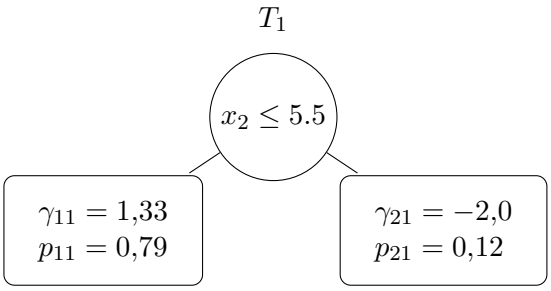


Figura 5.4: *Ilustração: Grandiente Boosting Classificação - Primeira Iteração.*

Tabela 5.8: *Ilustração:* gradient boosting *Classificação - Pseudo-resíduos iteração 2.*

i	x_1	x_2	x_3	y_i	p_i	$\tilde{r}_{2i}$
1	0,8	3,5	1	1	0,79	0,21
2	1,8	2,0	0	1	0,79	0,21
3	0,4	6,0	1	0	0,12	-0,12
4	1,9	8,0	0	0	0,12	-0,12
5	2,5	3,6	1	0	0,79	-0,79
6	3,0	2,5	0	1	0,79	0,21
7	4,0	5,0	1	1	0,79	0,21
8	3,5	4,5	0	1	0,79	0,21

Não utilizamos o parâmetro de regularização ν apenas para mostrar como as probabilidades se aproximam dos valores observados. A partir daqui o processo se repete recursivamente até o valor do número de iterações acordado.

6 XGBoost

O modelo XGBoost, acrônimo para e**X**treme **G**radient **Boost**ing tem a origem no método de *gradient boosting*, Capítulo 5, é baseado no artigo Chen e Guestrin (2016).

Este modelo difere essencialmente do gradiente boosting na medida em que acrescenta à função de perda (5.13) um parâmetro para controlar o tamanho da árvore, uma regularização para controlar os escores (valores nas folhas da árvore) e introduz um novo escore que compõe um critério para definir as partições dos nós ao usarmos o modelo CART. Vamos apresentar um desenvolvimento diferente, porém equivalente ao modelo aditivo usado para o gradiente boosting. O modelo aditivo fornece uma previsão para y_i na seguinte formulação

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{i=1}^{M} f_m(\mathbf{x}_i), \text{ em que } f_m \text{ são árvores CART.} \tag{6.1}$$

Um conjunto de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i),\ \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ |\mathcal{D}| = n\}$ está disponível, em que $\mathcal{X} \in \mathbb{R}^n$. A função objetivo para minimização, cf. (5.13), é dada pela função de perda L, convexa e diferenciável, mais o termo de regularização

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} L(\hat{y}_i, y_i) + \sum_{m=1}^{M} \Omega(f_m), \tag{6.2}$$

em que

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2. \tag{6.3}$$

O parâmetro γ controla o tamanho da árvore, $T = |T|$, e o parâmetro λ controla o tamanho dos escores w - em cada nó terminal a predição é dada pela soma dos scores alocados. Note que se $\gamma = 0$ e $\lambda = 0$ temos o modelo gradiente boosting.

6.1 Otimizando a Função Objetivo

Usando o procedimento de incrementos adicionados progressivamente, a cada iteração precisamos minimizar a função objetivo

$$\mathcal{L}^{(m)} = \sum_{i=1}^{n} L\left(y_i, \hat{y}_i^{(m-1)} + f_m(\mathbf{x}_i)\right) + \Omega(f_m), \tag{6.4}$$

em que $f_m(\mathbf{x}_i)$ é o incremento fornecido pela árvore atual.

O método XGBoost aproxima a função de perda pelo polinômio de Taylor de segunda ordem:

$$L\left(y_i, \hat{y}_i^{(m-1)} + f_m(\mathbf{x}_i)\right) \approx \tilde{L}\left(y_i, \hat{y}_i^{(m-1)} + f_m(\mathbf{x}_i)\right), \tag{6.5}$$

em que

$$\begin{aligned}\tilde{L}\left(y_i, \hat{y}_i^{(m-1)} + f_m(\mathbf{x}_i)\right) = L\left(y_i, \hat{y}_i^{(m-1)}\right) + \frac{\partial}{\hat{y}_i^{(m-1)}} L\left(y_i, \hat{y}_i^{(m-1)}\right) \cdot f_m(\mathbf{x}_i) \\ + \frac{1}{2}\frac{\partial^2}{\partial^2 \hat{y}_i^{(m-1)}} L\left(y_i, \hat{y}_i^{(m-1)}\right) \cdot f_m^2(\mathbf{x}_i).\end{aligned} \tag{6.6}$$

Na expressão (6.6) introduzimos a seguinte notação

$$g_i = \frac{\partial}{\partial \hat{y}_i^{(m-1)}} L\left(y_i, \hat{y}_i^{(m-1)}\right) \quad \text{e} \quad h_i = \frac{\partial^2}{\partial^2 \hat{y}_i^{(m-1)}} L\left(y_i, \hat{y}_i^{(m-1)}\right), \tag{6.7}$$

em que g_i é por gradiente e h_i por hessiano. Desta forma a expressão (6.6) torna-se

$$\tilde{L}\left(y_i, f_m(\mathbf{x}_i) + \hat{y}_i^{(m-1)}\right) = L\left(y_i, \hat{y}_i^{(m-1)}\right) + g_i f_m(\mathbf{x}_i) + \frac{1}{2} h_i f_m^2(\mathbf{x}_i). \tag{6.8}$$

Removendo o termo constante de (6.8) a função objetivo na *t-ésima* iteração torna-se

$$\tilde{\mathcal{L}}^{(m)} = \sum_{i=1}^{n} \left[g_i f_m(\mathbf{x}_i) + \frac{1}{2} h_i f_m^2(\mathbf{x}_i) \right] + \Omega(f_m). \tag{6.9}$$

Em cada folha delimitada pela região R_j, considere $I_j = \{i | \mathbf{x}_i \in R_j\}$, *i.e.*, $f_m(\mathbf{x}_i) = w_j \mathbb{1}(i \in I_j)$, precisamos minimizar $w_j^\star$ o escore ótimo para cada folha j, expandindo a expressão (6.9), temos:

$$\begin{aligned}\tilde{\mathcal{L}}^{(m)} &= \sum_{i=1}^{n} \left[g_i f(\mathbf{x}_i) + \frac{1}{2} h_i f^2(\mathbf{x}_i) \right] + \gamma T + \frac{1}{2} \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T,\end{aligned} \tag{6.10}$$

derivando em relação à w_j e igualando a zero:

$$\begin{aligned}\frac{\partial}{\partial w_j} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] = 0 \implies \\ \left(\sum_{i \in I_j} g_i \right) + \left(\sum_{i \in I_j} h_i + \lambda \right) w_j = 0 \implies \\ w_j^\star = -\frac{G_j}{H_j + \lambda}\end{aligned} \tag{6.11}$$

em que

$$G_j = \sum_{i \in I_j} g_i \quad \text{e} \quad H_j = \sum_{i \in I_j} h_i. \tag{6.12}$$

Substituindo (6.12) em (6.10), a função objetivo otimizada torna-se

$$\tilde{\mathcal{L}}^{(m)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T. \tag{6.13}$$

Ao final destes cálculos obtivemos o incremento $w_j^\star$ a ser adicionado a cada iteração de todo o processo. O parâmetro γ não fez parte dos cálculos por que na derivação dos valores w_j ele é constante e desaparece nas derivadas. No entanto, γ é utilizado para controlar o tamanho da árvore e serve para tomar a decisão de particionar ou não um dado nó.

Uma árvore que possui T folhas tem a complexidade γT, se esta folha for particionada, dois novos nós são acrescentados à estrutura da árvore, mas o nó deixa de ser terminal. A nova árvore terá complexidade $\gamma(T+1)$. Considere o nó j particionado em e e d nós filhos à esquerda e á direita, respectivamente. O ganho obtido pela partição do nó é dado pela expressão

$$\begin{aligned} Ganho &= \left(-\frac{1}{2}\frac{G_T^2}{H_T+\lambda}+\gamma T\right) - \left(-\frac{1}{2}\frac{G_e^2}{H_e+\lambda} - \frac{1}{2}\frac{G_d^2}{H_d+\lambda}+\gamma(T+1)\right) \\ &= \frac{1}{2}\left[\frac{G_e^2}{H_e+\lambda}+\frac{G_d^2}{H_d+\lambda}-\frac{G_j}{H_j+\lambda}\right]-\gamma. \end{aligned} \tag{6.14}$$

O ganho define uma regra para particionar um nó: a partição escolhida é a que proporciona maior ganho.

6.1.1 Árvores de Regressão

O Algoritmo 9 apresenta em linhas gerais o processo de cálculo das árvores XGBoost. Vamos utilizar a perda quadrática $L(y,f(\mathbf{x})) = \frac{1}{2}(y-f(\mathbf{x}))^2$ para exemplificar o funcionamento do algoritmo. Considere disponível o conjunto de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i),\ \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ |\mathcal{D}| = n\}$, $\mathcal{X} \in \mathbb{R}^n$. Seja

$$L(y_i,\hat{y}_i) = \frac{1}{2}(y_i-\hat{y}_i)^2 = \frac{1}{2}(y_i-\hat{y}_i)^2. \tag{6.15}$$

A árvores são construídas a partir dos pseudo-resíduos linha $2(a)i$

$$\tilde{r}_{im} = -\frac{\partial}{\partial\hat{y}_i}\frac{1}{2}(y_i-\hat{y}_i)^2 = y_i-\hat{y}_i. \tag{6.16}$$

O gradiente linha $2(a)ii$ é dado por

$$\begin{aligned} g_{im} &= \frac{\partial}{\partial\hat{y}_i}\frac{1}{2}(y_i-\hat{y}_i)^2 \\ &= -(y_i-\hat{y}_i), \end{aligned} \tag{6.17}$$

e o hessiano linha $2(a)ii$ é

$$\begin{aligned} h_{im} &= \frac{\partial^2}{\partial^2\hat{y}_i}\frac{1}{2}(y_i-\hat{y}_i)^2 \\ &= \frac{\partial}{\partial\hat{y}_i} - (y_i-\hat{y}_i) \\ &= 1. \end{aligned} \tag{6.18}$$

Para o cálculo dos escores G_j e H_j linha $2(b)i$, vamos considerar $n_j = |I_j|$,

$$G_{jm} = \sum_{i\in I_j} g_i = -\sum_{i\in I_j} y_i - \hat{y}_i \tag{6.19}$$

e

$$H_{jm} = \sum_{i\in I_j} h_i = \sum_{i\in I_{jm}} 1 = n_{jm}. \tag{6.20}$$

O escores para calcular o Ganho, linhas $2(b)i$ e $2(b)ii$, são dadas por

$$\begin{aligned} S_{jm} &= \frac{G_{jm}^2}{H_{jm}+\lambda} = \frac{\left(\sum_{i\in I_{jm}} y_i - \hat{y}_i\right)^2}{n_{jm}+\lambda}, \\ S_{jme} &= \frac{G_{jme}^2}{H_{jme}+\lambda} = \frac{\left(\sum_{i\in I_{jme}} y_i - \hat{y}_i\right)^2}{n_{jme}+\lambda}, \\ S_{jmd} &= \frac{G_{jmd}^2}{H_{jmd}+\lambda} = \frac{\left(\sum_{i\in I_{jmd}} y_i - \hat{y}_i\right)^2}{n_{jmd}+\lambda}. \end{aligned} \tag{6.21}$$

Note que esse escore, se $\lambda = 0$, é a média da soma ao quadrado dos resíduos. O Ganho é calculado como

$$Ganho = S_{jme} + S_{jmd} - S_{jm}. \tag{6.22}$$

A nova previsão γ_{jm} é

$$w_{jm} = -\frac{G_{jm}}{H_{jm}+\lambda} = \frac{\sum_{i\in I_{jm}} y_i - \hat{y}_i}{n_{jm}+\lambda}. \tag{6.23}$$

Algoritmo 9: XGBBoost - Árvores de Regressão.

Dados: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$

1. Inicializar: $f_0(\mathbf{x}) = \arg\min_\mu = \sum_{i=1}^n L(y_i, \mu)$.
2. Para cada $m = 1, \ldots, M$:
   (a) Para cada $i = 1, \ldots, n$ calcular
      i. a variável resposta
      $$\tilde{r}_{im} = -\left[\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}\right].$$
      ii. o gradiente e o hessiano
      $$g_{im} = \frac{\partial}{\partial \hat{y}_i} L(y_i, \hat{y}_i) \quad \text{e} \quad h_{im} = \frac{\partial^2}{\partial^2 \hat{y}_i} L(y_i, \hat{y}_i).$$
   (b) Ajustar uma árvore de regressão cuja variável resposta é $\tilde{r}_{im}$:
      i. Para cada folha jm calcular o escore
      $$S_{jm} = \frac{G_{jm}^2}{H_{jm} + \lambda}, \quad G_{jm} = \sum_{i \in I_{jm}} g_{im} \quad \text{e} \quad H_{jm} = \sum_{i \in I_{jm}} h_{im}.$$
      ii. Para cada possível partição de j calcular o escore para os nós filhos S_{jme} e S_{jmd}
      $$S_{jme} = \frac{G_{jme}^2}{H_{jme} + \lambda} \quad \text{e} \quad \frac{G_{jmd}^2}{H_{jmd} + \lambda}$$
      iii. Calcular o Ganho de cada partição e escolher a partição cujo Ganho é o maior
      $$Ganho = S_{jme} + S_{jmd} - S_{jm}.$$
      iv. Encontrar as regiões terminais $R_{jm}, j = 1, \ldots, T_m$.
   (c) Para cada $j = 1, \ldots, T_m$ calcular
   $$\gamma_{jm} = -\frac{G_{jm}}{H_{jm} + \lambda}.$$
   (d) Atualizar $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}(\mathbf{x} \in R_{jm})$.

Resultado: $T(\mathbf{x}) = f_M(x)$

6.1.2 Árvores de Classificação

Dado disponível o conjunto de aprendizado $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, |\mathcal{D}| = n\}$, $\mathcal{X} \in \mathbb{R}^n$ e $y \in \{c_1, \ldots, c_K\}$. Seguindo o modelo de gradiente boosting, Capítulo 5, precisamos gerar uma sequência de árvores para cada razão de log's (Secção 5.3.2). Considerando a função de perda multinomial *deviance*

$$L(y, p(\mathbf{x})) = -\log(l(y|p(\mathbf{x}))) = -\sum_{k=1}^{K} y_k \log p_k(\mathbf{x}), \tag{6.24}$$

em que $y_k = \mathbb{1}(y = c_k), k = 1, \ldots, K$, os escores necessários para a construção das árvores são apenas modificados pela inclusão do parâmetro λ nas respectivas expressões.

Os escores g_i e h_i das expressões (6.7) são os mesmos já derivados em (5.35) a (5.39),

$$g_i = -(y_i - p_k(\mathbf{x}_i)) \quad \text{e} \quad h_i = p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i)), \tag{6.25}$$

desta forma as expressões G_j e H_j (6.12), para as observações alocadas na folha j, são dados por

$$G_j = -\sum_{i \in I_j}(y_i - p_k(\mathbf{x}_i)) \quad \text{e} \quad H_j = \sum_{i \in I_j} p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i)). \tag{6.26}$$

E os escores S_j, dentro dos nós, para calcular o *Ganho* são

$$S_j = \frac{G_j^2}{H_j + \lambda} = \frac{\left(\sum_{i \in I_j}(y_i - p_k(\mathbf{x}_i))\right)^2}{\sum_{i \in I_j} p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i)) + \lambda}. \tag{6.27}$$

A nova previsão γ_j (6.11) torna-se

$$w_j = \frac{\sum_{i \in I_j}(y_i - p_k(\mathbf{x}_i))}{\sum_{i \in I_j} p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i)) + \lambda}. \tag{6.28}$$

6.2 Regularização

Na prática os meta-parâmetros λ e γ podem ser encontrados a partir de validação cruzada. Um valor relativamente grande de γ tende a produzir árvores menores. O parâmetro λ diminui o peso de cada folha na previsão final, particularmente é mais influente em nós terminais com poucas observações.

Seguindo o modelo de gradiente boosting, para prevenir sobreajuste, um parâmetro $0 < \eta < 1$ de regularização na influência de cada árvore pode ser introduzido. Na atualização das previsões em cada iteração, incluímos η

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \eta \cdot \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}(\mathbf{x} \in R_{jm}). \tag{6.29}$$

Como utilizado nas florestas aleatórias, Capítulo 3, subamostras das variáveis preditoras também podem ser utilizadas a cada iteração e, da mesma maneira que no gradiente boosting estocástico (Secção 5.4), subamostras *sem reposição* do conjunto de aprendizado podem ser utilizadas também a cada iteração.

6.3 Ilustração

Árvores de Regressão

Seguiremos o que foi exposto na Secção 6.1.1 e o Algoritmo 9. O conjunto de dados para construção das árvores é dado na Tabela 6.1.

Tabela 6.1: *Ilustração: XGBoost Regressão - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	5,0
2	1,8	2,0	0	4,0
3	0,4	6,0	1	2,5
4	1,9	8,0	0	1,5
5	2,5	3,6	1	9,5
6	3,0	2,5	0	8,0
7	4,0	5,0	1	0,8
8	3,5	4,5	0	0,4

A estimativa inicial para y, considerando a perda quadrática, é a média

$$\arg\min_{\mu} \sum_{i=1}^{8} (y_i - \mu)^2 \implies f_0(\mathbf{x}) = \mu = \bar{y} = \frac{5{,}0 + \cdots + 0{,}4}{8} = 3{,}96, \tag{6.30}$$

que fornece os pseudo-resíduos e respectivos gradientes e hessianos, cf. Tabela 6.2, para a primeira iteração.

Tabela 6.2: *Ilustração: XGBoost Regressão - Pseudo-resíduos iteração 1.*

i	x_1	x_2	x_3	$\tilde{r}_i$	g_i	h_i
1	0,8	3,5	1	1,04	-1,04	1
2	1,8	2,0	0	0,04	-0,04	1
3	0,4	6,0	1	-1,46	1,46	1
4	1,9	8,0	0	-2,46	2,46	1
5	2,5	3,6	1	5,54	-5,54	1
6	3,0	2,5	0	4,04	-4,04	1
7	4,0	5,0	1	-3,16	3,16	1
8	3,5	4,5	0	-3,56	3,56	1

A árvore de regressão ajustada aos pseudo-resíduos, para exemplificar a escolha da melhor partição as Tabelas 6.3, 6.4 e 6.5 resumem os cálculos. Lembrando que para a perda quadrática, usando o parâmetro $\lambda = 0$, para a primeira iteração as expressões de cálculo - omitindo o índice da iteração - são as seguintes

$$\begin{aligned} g_i &= -(y_i - f_0(\mathbf{x})), \\ h_i &= 1, \\ G_j &= \sum_{i \in I_j} g_i, \quad j = 1, 2 \text{ folhas}, \\ H_j &= \sum_{i \in I_j} h_i, \\ S_j &= \frac{G_j^2}{H_j + (\lambda = 0)}. \end{aligned} \tag{6.31}$$

O cálculo do escore na raiz é

$$\begin{aligned} g_i &= -(y_i - 3{,}96) \\ G_0 &= (-1{,}04 + \cdots - 3{,}56) = 5{,}70, \\ H_0 &= 1 + \cdots + 1 = 8, \\ S_0 &= \frac{G_0^2}{H_0} = \frac{5{,}70^2}{8} = 4{,}06 \end{aligned} \tag{6.32}$$

e o Ganho ao particioná-la

$$Ganho = S_1 + S_2 - S_0. \tag{6.33}$$

Tabela 6.3: *Ilustração: XGBoost Regressão - Partições x_1.*

x_1	0,4	0,8	1,8	1,9	2,5	3,0	3,5	4,0
$\tilde{r}_i$	-1,46	1,04	0,04	-2,46	5,54	4,04	-3,56	-3,16
c		0,60	1,30	1,85	2,20	2,75	3,25	3,75
g_i	1,46	-1,04	-0,04	2,46	-5,54	-4,04	3,56	3,16
h_i	1	1	1	1	1	1	1	1
G_1		1,46	0,42	0,38	2,84	-2,70	-6,74	-3,18
G_2		-1,48	-0,44	-0,40	-2,86	2,68	6,72	3,16
H_1		1	2	3	4	5	6	7
H_2		7	6	5	4	3	2	1
S_1		2,13	0,09	0,05	2,02	1,46	7,57	1,44
S_2		0,31	0,03	0,03	2,04	2,39	22,58	9,99
Ganho		-1,62	-3,94	-3,98	0,00	-0,21	26,09	7,37

Tabela 6.4: *Ilustração: XGBoost Regressão - Partições x_2.*

x_2	2,0	2,5	3,5	3,6	4,5	5,0	6,0	8,0
$\tilde{r}_i$	0,04	4,04	1,04	5,54	-3,56	-3,16	-1,46	-2,46
c		2,25	3,00	3,55	4,05	4,75	5,50	7,00
g_i	-0,04	-4,04	-1,04	-5,54	3,56	3,16	1,46	2,46
h_i	1	1	1	1	1	1	1	1
G_1		-0,04	-4,08	-5,12	-10,66	-7,10	-3,94	-2,48
G_2		0,02	4,06	5,10	10,64	7,08	3,92	2,46
H_1		1	2	3	4	5	6	7
H_2		7	6	5	4	3	2	1
S_1		0,00	8,32	8,74	28,41	10,08	2,59	0,88
S_2		0,00	2,75	5,20	28,30	16,71	7,68	6,05
Ganho		-4,06	7,01	9,88	52,65	22,73	6,21	2,87

Tabela 6.5: *Ilustração: XGBoost Regressão - Partições x_3.*

x_3	0	0	0	0	1	1	1	1
$\tilde{r}_i$	0,04	-2,46	4,04	-3,56	1,04	-1,46	5,54	-3,16
c					0,50			
g_i	-0,04	2,46	-4,04	3,56	-1,04	1,46	-5,54	3,16
h_i	1	1	1	1	1	1	1	1
G_1					1,94			
G_2					-1,96			
H_1					4			
H_2					4			
S_1					0,94			
S_2					0,96			
Ganho					-2,16			

A partição $x_2 \leq 4.05$ é a melhor partição - em destaque na Tabela 6.4, cujo resultado é apresentado na Figura 6.1.

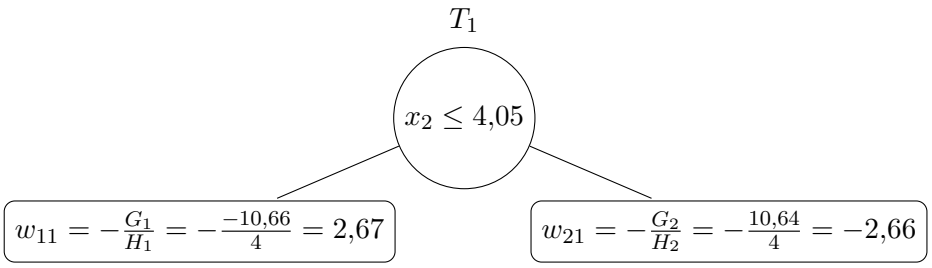


Figura 6.1: *Ilustração: XGBoost Regressão - Primeira Iteração.*

Uma vez que utilizamos $\lambda = 0$ e estamos no primeiro nó, o resultado é o mesmo apresentado pelo Gradiente Boosting, cf. Secção 5.6. Novos pseudo-resíduos são calculados e o ciclo se repete.

Árvores de Classificação

Seguiremos o que foi exposto na Secção 6.1.2 e o Algoritmo 9. O conjunto de dados para construção das árvores é dado na Tabela 6.6.

Tabela 6.6: *Ilustração: XGBoost Classificação - Conjunto de dados.*

i	x_1	x_2	x_3	y_i
1	0,8	3,5	1	1
2	1,8	2,0	0	1
3	0,4	6,0	1	0
4	1,9	8,0	0	0
5	2,5	3,6	1	0
6	3,0	2,5	0	1
7	4,0	5,0	1	1
8	3,5	4,5	0	1

Como previsão inicial $f_{01}(\mathbf{x}) = f_{11}(\mathbf{x}) = 0$, dessa forma as probabilidades iniciais para as classes são $p_0(\mathbf{x}) = p_1(\mathbf{x}) = 0{,}5$. Construíremos árvores com somente duas folhas e usaremos a função de

perda o negativo do log da verossimilhança. A variável resposta é binária $y \in \{0,1\}$. Os pseudo-resíduos e os respectivos gradientes e hessianos, para a primeira iteração estão na Tabela 6.7.

Tabela 6.7: *Ilustração: XGBoost Classificação - Pseudo-resíduos iteração 1.*

i	x_1	x_2	x_3	$\tilde{r}_i$	g_i	h_i
1	0,8	3,5	1	0,5	-0,5	0,25
2	1,8	2,0	0	0,5	-0,5	0,25
3	0,4	6,0	1	-0,5	0,5	0,25
4	1,9	8,0	0	-0,5	0,5	0,25
5	2,5	3,6	1	-0,5	0,5	0,25
6	3,0	2,5	0	0,5	-0,5	0,25
7	4,0	5,0	1	0,5	-0,5	0,25
8	3,5	4,5	0	0,5	-0,5	0,25

A árvore de regressão ajustada aos pseudo-resíduos, para exemplificar a escolha da melhor partição as Tabelas 6.8, 6.9 e 6.10 resumem os cálculos. Lembrando que para a perda quadrática, usando o parâmetro $\lambda = 0$, para a primeira iteração as expressões de cálculo - omitindo o índice da iteração - usadas são as seguintes

$$\begin{aligned} g_i &= -(y_i - p_1(\mathbf{x})), \\ h_i &= p_1(\mathbf{x})(1 - p_1(\mathbf{x})), \\ G_j &= \sum_{i \in I_j} g_i, \quad j = 1, 2, \\ H_j &= \sum_{i \in I_j} h_i, \\ S_j &= \frac{G_j^2}{H_j + (\lambda = 0)}. \end{aligned} \tag{6.34}$$

O cálculo do escore na raiz é

$$\begin{aligned} g_i &= 0{,}5(1 - 0{,}5) = 0{,}25 \\ G_0 &= -0{,}5 + \cdots - 0{,}5 = -1{,}0, \\ H_0 &= 0{,}25 + \cdots + 0{,}25 = 2{,}0, \\ S_0 &= \frac{G_0^2}{H_0} = \frac{(-1{,}0)^2}{2} = 0{,}5, \end{aligned} \tag{6.35}$$

e o Ganho ao particioná-la

$$Ganho = S_1 + S_2 - S_0. \tag{6.36}$$

Tabela 6.8: *Ilustração: XGBoost Classificação - Partições x_1.*

x_1	0,4	0,8	1,8	1,9	2,5	3,0	3,5	4,0
$\tilde{r}_i$	-0,5	0,5	0,5	-0,5	-0,5	0,5	0,5	0,5
c		0,60	1,30	1,85	2,20	2,75	3,25	3,75
g_i	0,5	-0,5	-0,5	0,5	0,5	-0,5	-0,5	-0,5
h_i	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25
G_1		0,5	0,0	-0,5	0,0	0,5	0,0	-0,5
G_2		-1,5	-1,0	-0,5	-1,0	-1,5	-1,0	-0,5
H_1		0,25	0,5	0,75	1,0	1,25	1,5	1,75
H_2		1,75	1,5	1,25	1,0	0,75	0,5	0,25
S_1		1,00	0,00	0,33	0,00	0,20	0,00	0,14
S_2		1,29	0,67	0,20	1,00	3,00	2,00	1,00
Ganho		1,79	0,17	0,03	0,50	2,70	1,50	0,64

Tabela 6.9: *Ilustração: XGBoost Classificação - Partições x_2.*

x_2	2,0	2,5	3,5	3,6	4,5	5,0	6,0	8,0
$\tilde{r}_i$	0,5	0,5	0,5	-0,5	0,5	0,5	-0,5	-0,5
c		2,25	3,00	3,55	4,05	4,75	5,50	7,00
g_i	-0,5	-0,5	-0,5	0,5	-0,5	-0,5	0,5	-0,5
h_i	1	1	1	1	1	1	1	1
G_1		-0,5	-1,0	-1,5	-1,0	-1,5	-2,0	-1,5
G_2		-1,5	-1,0	-0,5	-1,0	-0,5	0,0	-0,5
H_1		0,25	0,5	0,75	1,0	1,25	1,5	1,75
H_2		1,75	1,5	1,25	1,0	0,75	0,5	0,25
S_1		1,00	2,00	3,00	1,00	1,80	2,67	1,29
S_2		1,29	0,67	0,20	1,00	0,33	0,00	1,00
Ganho		1,79	2,17	2,70	1,50	1,63	2,17	1,79

Tabela 6.10: *Ilustração: XGBoost Classificação - Partições x_3.*

x_3	0	0	0	0	1	1	1	1
$\tilde{r}_i$	0,5	-0,5	0,5	0,5	0,5	-0,5	-0,5	0,5
c					0,50			
g_i	-0,5	0,5	-0,5	-0,5	-0,5	0,5	-0,5	-0,5
h_i	1	1	1	1	1	1	1	1
G_1					-1,0			
G_2					-1,0			
H_1					4			
H_2					4			
S_1					1,0			
S_2					1,0			
Ganho					1.5			

As partição $x_1 \leq 2.75$ e $x_2 \leq 3.55$ apresentam o mesmo Ganho - em destaque na Tabela 6.9.

Escolhemos a partição, a título de exemplo, a partição $x_2 \leq 3.55$. O resultado do modelo é

$$\begin{aligned} w_{11} &= -\frac{G_1}{H_1} = -\frac{-1{,}5}{0{,}75} = 2{,}0, \\ w_{21} &= -\frac{G_2}{H_2} = -\frac{-0{,}5}{1{,}25} = 0{,}4, \end{aligned} \tag{6.37}$$

e, lembrando que $f_{01}(\mathbf{x}) = f_{11}(\mathbf{x}) = 0$, as novas probabilidades são

$$\begin{aligned} p_{11} &= \frac{e^{2{,}0}}{e^{2{,}0} + e^{0}} = 0{,}88 \quad \text{e} \\ p_{21} &= \frac{e^{0{,}4}}{e^{0{,}4} + e^{0}} = 0{,}60. \end{aligned} \tag{6.38}$$

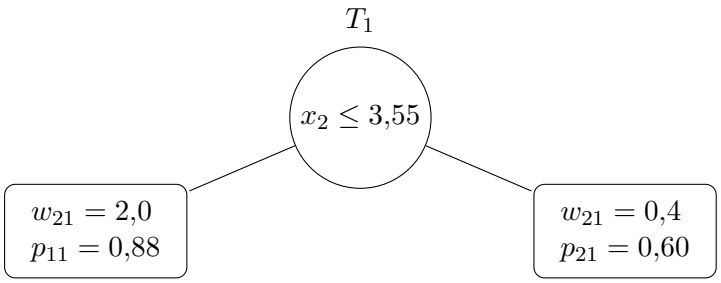


Figura 6.2: *Ilustração: XGBoost Classificação - Primeira Iteração.*

Com as novas probabilidades recalculamos $\tilde{r}_i$ e o processo se repete.

7 Modelo Bayesiano CART

Os modelos Bayesianos não nascem sem a especificação de uma distribuição de probabilidade *a priori* qualquer que seja o problema a ser tratado. Não é diferente com as árvores de decisão. Dada uma distribuição a priori, o próximo passo é calcular a distribuição a *posteriori* e realizar inferência sobre os parâmetros.

A distribuição a priori é uma hipótese e uma vez estabelecida um cálculo direto pode gerar analiticamente a distribuição a posteriori, ou em modelos mais complexos, como é o caso das árvores, uma técnica de Monte Carlo vem em auxílio para tornar a computação das probabilidades factível.

Vamos nos fixar no modelo padrão Bayesiano CART desenvolvido por Chipman *et al.* (1998). Seguiremos a notação apresentada no referido artigo para facilitar futuras pesquisas sobre o tema.

Seja (Θ, T) o conjunto de parâmetros $\Theta = (\theta_1, \ldots, \theta_b)$ a serem estimados e T uma árvore como no modelo padrão CART com b nós terminais. Considere y_{ij} a *j-ésima* observação da variável resposta dentro do *i-ésimo* nó terminal associada às observações $(x_{i1}, x_{i2}, \ldots, x_{i_{n_i}})$ das variáveis preditoras, $j = 1, 2, \ldots, n_i$ e $i = 1, 2, \ldots, b$. Considere

$$\begin{aligned} Y &= (Y_1, \ldots, Y_b)', \text{ em que } Y_i = (y_{i1}, y_{i2}, \ldots, y_{in_i})' \\ X &= (X_1, \ldots, X_b)', \text{ em que } X_i = (x_{i1}, x_{i2}, \ldots, x_{in_i})'. \end{aligned} \tag{7.1}$$

O modelo Bayesiano CART segue o roteiro:

1. A definição de um modelo preditivo em cada folha $P(Y_i|\theta_i)$.
2. A definição de uma priori para (Θ, T): $P(\Theta, T) = P(\Theta|T)\, P(T)$.
3. A definição de uma priori para T: $P(T)$.
4. A definição de uma priori $P(\Theta|T)$.
5. O cálculo de $P(Y|X, T) = \int P(Y|X, \Theta, T)\, P(\Theta|T)\, d\Theta$.
6. O Cálculo de $P(T|X, Y) \propto P(Y|X, T)\, P(T)$.

Particionar um nó envolve:

1. Uma regra para particionar um nó ou declará-lo terminal.
2. Uma regra para escolher uma variável preditora para um nó.
3. Uma regra para escolher o ponto de corte na variável preditora ao particionar um nó.

7.1 Um Modelo Preditivo para cada Folha

Assumindo que os valores $y|(\Theta, T)$ são i.i.d. dentro dos nós terminais e entre as folhas, a distribuição para os dados é dada por

$$P(Y|X, \Theta, T) = \prod_{i=1}^{b} f(Y_i|\theta_j) = \prod_{i=1}^{b} \prod_{j=1}^{n_i} f(y_{ij}|\theta_i) \tag{7.2}$$

Para árvores de regressão dois modelos são considerados. O primeiro com variância constante (*mean shift model*)

$$y_{i1}, \ldots, y_{in_i} | \theta_i \sim N(\mu_i, \sigma^2),\ i = 1, \ldots, b, \tag{7.3}$$

em que $\theta_i = (\mu_i, \sigma^2)$, e o segundo em que a variância assume um valor de acordo com o nó terminal (*mean-variance shift model*)

$$y_{i1}, \ldots, y_{in_i} | \theta_i \sim N(\mu_i, \sigma_i^2),\ i = 1, \ldots, b, \tag{7.4}$$

em que $\theta_i = (\mu_i, \sigma_i^2)$.

Para árvores de classificação consideramos o modelo de Bernoulli generalizado

$$f(y_{i1}, \ldots, y_{in_j} | \theta_i) = \prod_{j=1}^{n_i} \prod_{k=1}^{K} p_{ik}^{\mathbb{1}(y_{ij} \in \mathcal{C}_k)},\ i = 1, \ldots, b, \tag{7.5}$$

em que $\theta_i = p_i = (p_{i1}, \ldots, p_{iK})$, $p_{ik} \geq 0$, $\sum p_{ik} = 1$ e K é o número de classes, $p_{ik} = P(y_{ij} \in \mathcal{C}_k | p_i)$.

Uma vez que o modelo é especificado por (Θ, T), uma análise bayesiana necessita uma priori para (Θ, T):

$$P(\Theta, T) = P(\Theta | T)\, P(T). \tag{7.6}$$

Condicionando T ficamos livres para especificar uma distribuição para T que não depende de Θ. Desta forma, a mesma abordagem pode ser usada independente da distribuição escolhida para T, seja para árvores de regressão ou árvores de classificação. Adicionalmente uma escolha adequada para $P(\Theta|T)$ torna mais fácil computacionalmente encontrar formas analíticas para a posteriori.

7.2 Uma Priori para T

Ao invés de especificar $P(T)$ diretamente, definimos $P(T)$ implicitamente: $P(T)$ é definido como um processo aleatório no qual cada árvore gerada é uma amostra de $P(T)$. Por exemplo, fixado um valor $0 < \alpha < 1$, definimos $P(\eta, T) = \alpha$, *i.e.*, no processo de crescimento da árvore um dado nó η na árvore T é particionado com probabilidade α e com probabilidade $1 - \alpha$ ele se torna um nó terminal. Observe que α controla o tamanho da árvore: uma valor pequeno de α tende a gerar árvores com poucos nós.

Entretanto, essa proposta faz com que todas as árvores com b nós terminais tenham a mesma probabilidade sem levar em consideração a sua forma. Note que uma árvore com b nós terminais tem $b-1$ nós internos, desta maneira qualquer árvore com b nós terminais tem probabilidade $\alpha^{b-1}(1-\alpha)^b$ de ocorrer.

Uma proposta mais interessante de definir $P(\eta, T)$ é a seguinte:

$$P(\eta, T) = \frac{\alpha}{(1 + d_\eta)^\beta},\ \beta \geq 0 \text{ e } d_\eta \text{ é a profundidade do nó } \eta. \tag{7.7}$$

Esta formulação favorece árvores mais densas com os nós terminais tendendo a pertencer ao mesmo nível, no entanto, não exclui árvores de outros tamanhos. Escolhendo para o nó raiz a profundidade 0, *i.e.*, $d_\eta = 0$, à medida que a árvore cresce o parâmetro β controla a probabilidade do nó particionar. Os parâmetros α e β funcionam como parâmetros de regularização. A figura 7.1 mostra a probabilidade da árvore atingir uma certa quantidade de folhas alguns valores de α e β.

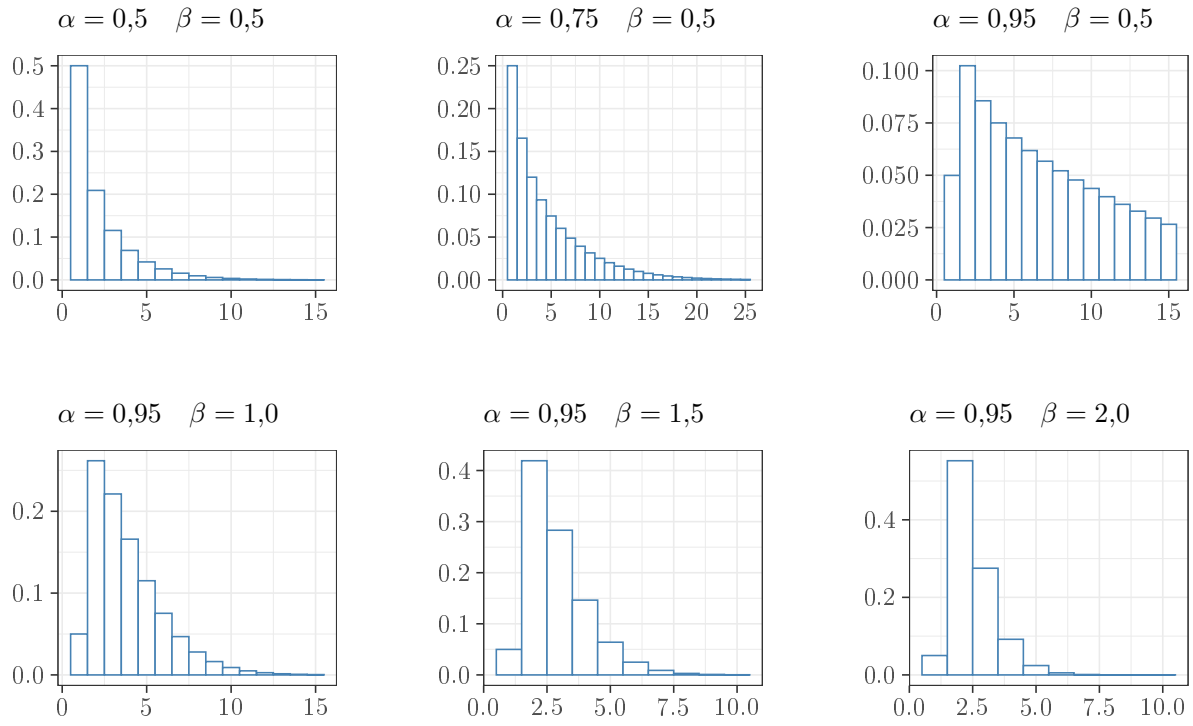


Figura 7.1: *$P(T)$ em função de α e β - no eixo x o número de nós terminais.*

7.3 Uma Priori para $\Theta|T$

A escolha de uma priori adequada para $\Theta|T$ permite calcular com maior facilidade a distribuição a posteriori, o que por sua vez torna possível obter mais facilmente $P(Y|X,T)$:

$$P(Y|X,T) = \int P(Y|X,\Theta,T)\, P(\Theta|T) d\Theta. \tag{7.8}$$

7.3.1 Árvores de Regressão

Para árvores de regressão utilizamos como priori a distribuição normal para μ_i e gama inversa para σ^2/σ_i^2:

$$\mu_1, \ldots, \mu_b | \sigma, T \; iid \sim N(\bar{\mu}, \sigma^2/a) \tag{7.9}$$

e

$$\sigma^2|T \; \sim GamaInversa(\nu/2, \nu\lambda/2) \quad (\Leftrightarrow \nu\lambda/\sigma^2 \sim \chi^2_\nu). \tag{7.10}$$

Esta escolha leva à seguinte forma analítica

$$P(Y|X,T) = \frac{c\, a^{b/2}}{\prod_{i=1}^{b}(n_i + a)^{1/2}} \left(\sum_{i=1}^{b} (s_i + t_i) + \nu\lambda \right)^{-(n+\nu)/2}, \tag{7.11}$$

em que c é constante e não depende de T, $s_i = \sum_{j=1}^{n_j} (y_{ij} - \bar{y}_i)^2$, $\bar{y}_i = (1/n_i) \sum_{j=1}^{n_i} y_{ij}$ é a média amostral de Y_i na folha i, $t_i = \frac{n_i a}{n_i + a}(\bar{y}_i - \bar{\mu})$.

No caso do modelo em que a variância não é constante, temos

$$\mu_i|\sigma_i \sim N(\bar{\mu},\sigma_i^2/a) \tag{7.12}$$

e

$$\sigma_i^2 \sim GamaInversa(\nu/2,\nu\lambda/2) \tag{7.13}$$

com os pares $(\mu_1,\sigma_1^2),\ldots,(\mu_b,\sigma_b^2)$ i.i.d. A equação (7.11) pode ser escrita torna-se

$$P(Y|X,T) = \prod_{i=1}^{b}\left[\pi^{-n_i/2}(\lambda\nu)^{\nu/2}\frac{\sqrt{a}}{\sqrt{n_i+a}} \times \frac{\Gamma((n_i+\nu)/2)}{\Gamma(\nu/2)}(s_i+t_i+\nu\lambda)^{-(n_i+\nu)/2}\right]. \tag{7.14}$$

em que s_i e t_i permanecem como no caso de variância constante. Na prática os parâmetros $(\nu,\lambda,\bar{\mu},a)$ são estimados a partir dos valores observados de Y.

7.3.2 Árvores de Classificação

Para árvores de classificação, cuja variável resposta tem K classes, a escolha para $\theta = (p_1,\ldots,p_b)$ recai sobre a priori conjugada da distribuição de Dirichlet de dimensão $K-1$ com parâmetro $\alpha = (\alpha_1,\ldots,\alpha_K)$, $\alpha_K > 0$:

$$p_1,\ldots,p_b|T\ iid \sim Dirichlet(p_i|\alpha) \propto p_{i1}^{\alpha_1-1}\cdots p_{iK}^{\alpha_K-1}. \tag{7.15}$$

Quando $K=2$ temos a distribuição Beta. Usando esta priori obtemos

$$P(Y|X,T) = \left(\frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)}\right)^b \times \prod_{i=1}^{b}\frac{\prod_k \Gamma(n_{ik}+\alpha_k)}{\Gamma(n_i+\sum_k \alpha_k)}. \tag{7.16}$$

em que $n_{ik} = \sum_j \mathbb{1}(y_{ij} \in \mathcal{C}_k)$, $n_i = \sum_k n_{ik}$ e $k = 1,\ldots,K$. Uma prior uniforme pode ser especificada tomando $\alpha = (1,\ldots,1)$.

7.4 Especificando uma Regra de Particionamento

A regra proposta para o particionamento $p_{regra} \equiv p_{rule}$ é composta de duas partes

1. Seleção aleatória da variável preditora, *e.g.*, x_ℓ.
2. Selecionar aleatoriamente, entre os valores únicos de x_ℓ um valor para efetuar a quebra caso x_ℓ seja numérica, ou um subconjunto de $\mathcal{C}$ caso x_ℓ seja categórica.

A seleção aleatória deve ser uniforme de maneira que cada variável preditora disponível tenha a mesma probabilidade ser selecionada. Da mesma forma, uma vez escolhido x_i selecionamos uniformemente entre os valores (únicos) disponíveis o valor para efetuar a quebra. Em cada nó devemos observar quais são as variáveis disponíveis, por exemplo, se uma variável binária for selecionada em algum momento, os nós abaixo não poderão levar em consideração a seleção desta variável uma vez que ela não permite mais particionar o nó.

Uma alternativa à seleção uniforme é especificar probabilidades diferentes para a seleção da variável preditora levando em consideração a importância que a variável tem como preditora em relação às outras, da mesma forma pode-se especificar uma distribuição sobre os valores de x_i que não seja uniforme.

7.5 Cálculo da Posteriori $P(T|X,Y)$

De acordo com a proposta para $P(\Theta|T)$ desenvolvida em (7.3) obtivemos expressões fechadas para $P(Y|X,T)$ em (7.11), (7.14) e (7.16). Isto nos permite calcular, a menos de uma constante:

$$P(T|X,Y) \propto P(Y|X,T)\,P(T). \tag{7.17}$$

A procura exaustiva por todas as árvores é proibitiva, mesmo sabendo que o número de árvores é finito - a forma como particionamos os dados nos leva a um número finito de possibilidades uma vez que as partições dependem dos x's observados, cujo número certamente é finito. Assim, a busca por $P(T|X,Y)$ será estocástica usando o algoritmo Metropolis-Hastings (Gelman *et al.*, 2014). Uma sequência de árvores serão geradas através de uma Cadeia de Markov:

$$T_0, T_1, T_2, T_3, \ldots \tag{7.18}$$

a qual converge em distribuição para $P(T|X,Y)$.

O algoritmo Metropolis-Hastings para simular a Cadeia de Markov $T_0, T_1, \ldots$ é definido com os seguintes passos:

1. Gere uma árvore candidata T^* com probabilidade $q(T_i, T^*)$.
2. Atribua $T_{i+1} = T^*$ com probabilidade
$$\alpha(T_i, T^*) = \min\left\{1, \frac{q(T^*, T_i)}{q(T_i, T^*)}\frac{P(Y|X,T^*)}{P(Y|X,T_i)}\frac{P(T^*)}{P(T_i)}\right\}. \tag{7.19}$$
3. Caso contrário, atribua $T_{i+1} = T_i$.

A probabilidade de transição $q(T, T^*)$ é definida como a probabilidade de gerar uma nova árvore T^* a partir de T através da escolha aleatória de uma das quatro operações:

- **GROW** Aleatoriamente selecione um nó terminal e aplique a regra de particionamento.
- **PRUNE** Aleatoriamente selecione um nó interno com dois nós e colapse os filhos, tornando-o um nó terminal.
- **CHANGE** Aleatoriamente selecione um nó interno e aleatoriamente troque o particionamento de acordo com p_{regra}.
- **SWAP** Aleatoriamente selecione um par de nós pai-filho de forma que ambos sejam internos e troque as suas regras de particionamento a menos que o outro nó filho tenha a mesma regra de particionamento, neste caso troque ambas as regras dos nós filhos com a regra do nó pai.

A seleção aleatória destas quatro operações gera uma Cadeia de Markov reversível. Cada operação que leva T a T^* tem uma contrapartida em outra que leva de T^* a T. A operação GROW tem sua contrapartida na operação CHANGE, enquanto que a operação MUDE tem a sua em SWAP. Fica implícito que estas operações utilizam as variáveis e as regras de particionamento disponíveis em cada nó em que forem aplicadas. Adicionalmente, estas operações se limitam a nós cujo número de elementos seja maior do cinco.

Na forma como está expresso este algoritmo o cálculo de $\alpha(T, T^*)$ existe uma substancial economia de cálculo de CHANGE nos cancelamentos entre $q(T, T^*)$ e $P(T^*)$ em (7.19); o mesmo é válido para $q(T^*, T)$ e $P(T)$, no caso de PRUNE. O valor da razão de q's para as operações CHANGE e SWAP é sempre 1, o que evita o cálculo.

7.6 Identificação da Árvore Final

Uma vez que o modelo Bayesino CART não gera somente uma árvore, é necessário um critério para selecionar qual árvore utilizar para o modelo final. A sugestão de (Chipman *et al.*, 1998) é utilizar a verossimilhança marginal $P(Y|X,T)$ bem como observar a soma de quadrados dos erros no caso das árvores de regressão e a taxa de erro de classificação para as árvores de classificação.

As predições podem ser realizadas calculando a média das predições das árvores selecionadas.

7.7 Ilustração

Na ilustração desta técnica vamos abordar graficamente como ocorrem as operações GROW, PRUNE, CHANGE e SWAP.

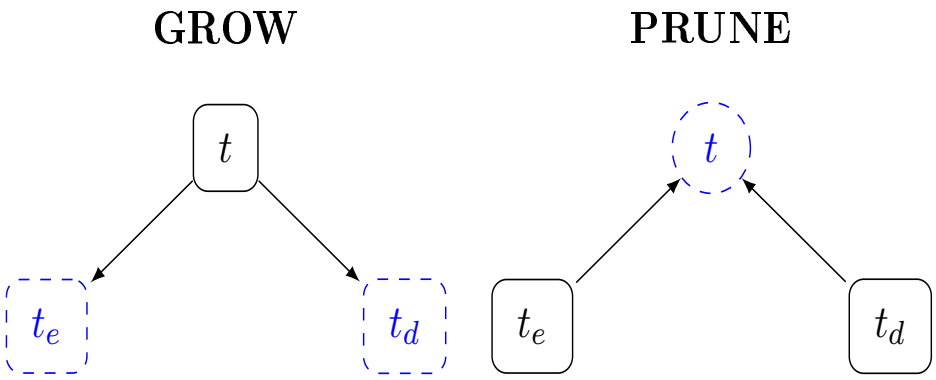


Figura 7.2: *Representação das Operações: GROW e PRUNE.*

A Figura 7.2 ilustra como as operações GROW e PRUNE ocorrem na construção das árvores BART-CART. Na operação GROW a árvore cresce a partir de um nó terminal. Na operação PRUNE, a árvore é podada: dois nós terminais voltam se tornar somente um nó terminal. Uma operação é a inversa da outra.

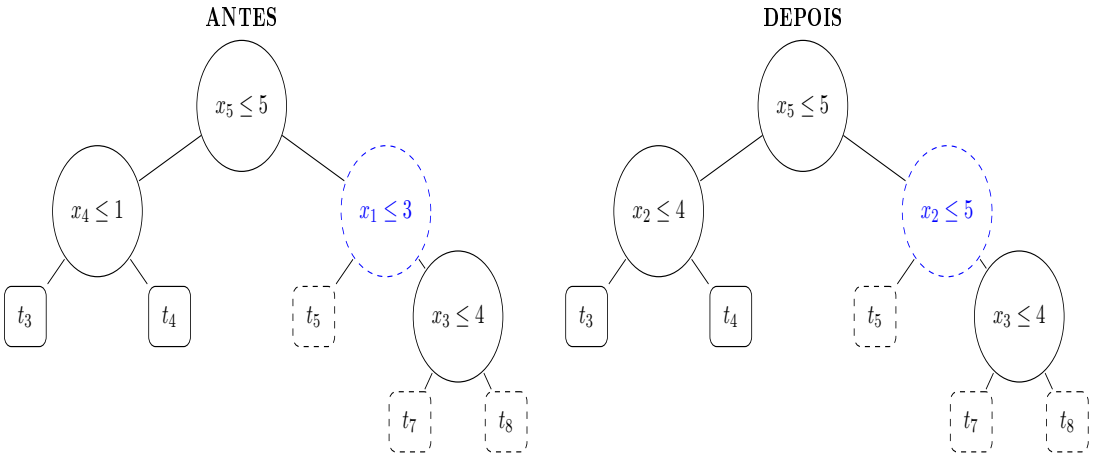


Figura 7.3: *Representação da Operação: CHANGE.*

Na Figura 7.3 representa a operação CHANGE. O regra de partição do nó $x \leq 3$ foi trocada pela regra $x_2 \leq 5$ de acordo com p_{regra}. Depois da troca as observações dos nós filhos devem ser realocadas de acordo com a nova regra.

A Figura 7.4 representa a operação SWAP contrapartida da operação CHANGE. As observações necessitam ser realocadas de acordo com a troca de regras.

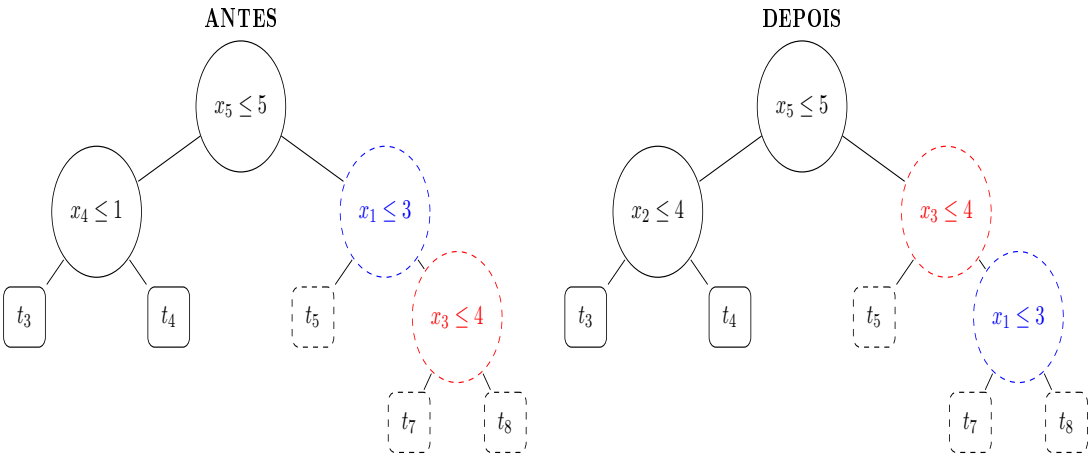


Figura 7.4: *Representação da Operação: SWAP.*

8 Modelo Bayesiano BART

O modelo Bayesiano BART acrônimo de **B**ayesian **A**dditive **R**egression **T**rees é uma evolução do modelo Bayesiano CART (Capítulo 7). Foi proposto em Chipman *et al.* (2010) pelos mesmos autores do modelo Bayesiano CART.

Ao invés de produzir uma única árvore como o modelo CART, o modelo BART se vale de uma *soma* de árvores para construir um modelo preditivo, no entanto, diferentemente dos modelos baseados em gradiente boosting, as árvores são construídas de forma independente uma das outras, e não necessariamente em sequência. Sendo Y a variável resposta, o problema geral de prever Y a partir do vetor $\mathbf{x} = (x_1, \ldots, x_p)$ de variáveis preditoras é encontrar uma função f de forma que

$$Y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \tag{8.1}$$

Com este modelo esperamos aproximar $f(\mathbf{x}) = E(Y|X = \mathbf{x})$, por uma soma de m árvores

$$f(\mathbf{x}) = \sum_{m=1}^{M} g(\mathbf{x}; T_m, \theta_m) + \epsilon, \quad \epsilon \sim N(0, \sigma^2), \tag{8.2}$$

onde T_m é a *m-ésima* árvore e $\theta_m = (\mu_{1m}, \ldots, \mu_{J_m m})$ é o vetor de parâmetros dos nós terminais associados à árvore T_m e J_m é o número de nós terminais da árvore T_m. Cada árvore T_m é gerada a partir do modelo Bayesiano CART e contém toda informação sobre as variáveis e valores utilizados nas respectivas partições.

8.1 Especificando Prioris

Note que a equação (8.2) contém explicitamente uma sequência de pares $(T_1, \theta_1), \ldots, (T_m, \theta_m)$ e σ para os quais necessitamos de uma distribuição a priori: $P((T_1, \theta_1), \ldots, (T_m, \theta_m), \sigma)$. Supondo independência entre os pares (T_j, θ_j) e entre σ esta priori pode ser escrita na forma

$$\begin{aligned} P((T_1, \theta_1), ldots, (T_m, \theta_m), \sigma) &= P((T_1, \theta_1), \ldots, (T_m, \theta_m))\, P(\sigma) \\ &= \left(\prod_{m=1}^{M} P(T_m, \theta_m) \right) P(\sigma) \\ &= \left(\prod_{m=1}^{M} P(\theta_m | T_m) P(T_m) \right) P(\sigma) \\ &= \left[\prod_{m=1}^{M} \left(\prod_{k=1}^{J_m} P(\mu_{km} | T_m) \right) P(T_m) \right] P(\sigma). \end{aligned} \tag{8.3}$$

Agora precisamos encontrar uma para cada componente de (8.3).

8.1.1 Uma Priori para $\mu_{km}|T_m$

Para $\mu_{km}|T_m$ a escolha recai sobre a distribuição normal: $\mu_{km}|T_m \sim N(\mu_\mu, \sigma_\mu^2)$, assim podemos utilizar toda facilidade oferecida pela conjugada como no modelo Bayesiano CART. Uma vez que

$E(Y|X)$ é uma soma de M árvores e os μ_{km} são considerados independentes pelo modelo, chegamos a uma priori induzida para $E(Y|X)$: $N(M\mu_\mu, M\sigma_\mu^2)$. Uma vez que é altamente provável que $y_{min} < M\mu_\mu < y_{max}$ podemos usar esse fato para escolher μ_μ e σ_μ de forma que $N(M\mu_\mu, M\sigma_\mu^2)$ esteja concentrada no intervalo (y_{min}, y_{max}). Como o desvio padrão é $\sqrt{M\sigma_\mu^2} = \sqrt{M}\sigma_\mu$, podemos escolher k convenientemente de acordo com as igualdades $y_{min} = m\mu_\mu - k\sqrt{m}\sigma_\mu$ e $y_{max} = M\mu_\mu + k\sqrt{M}\sigma_\mu$. Chipman *et al.* (2010) sugere $k = 2$, o que nos dá uma probabilidade a priori de $E(Y|\mathbf{x})$ estar no intervalo (y_{min}, y_{max}) de 95%. Adicionalmente Chipman *et al.* (2010) sugere mudar a escala de Y tornando $y_{min} = -0{,}5$ e $y_{max} = 0{,}5$ além de centralizar $\mu_\mu = 0$ e escolher σ_μ tal que $k\sqrt{M}\sigma_\mu = 0{,}5$, o que nos leva à seguinte distribuição a priori para μ_{km}:

$$\mu_{km} \sim N(0, \sigma_\mu^2), \quad \sigma_\mu = 0{,}5/k\sqrt{M}. \tag{8.4}$$

8.1.2 Uma Priori para T_m

Para $T_m, m = 1, \ldots, M$, seguimos a mesma especificação do modelo Bayesiano CART conforme desenvolvido em (7.2). Em cada nó η de T_m a probabilidade de particionar é dada por

$$P(\eta, T_m) = \frac{\alpha}{(1 + d_\eta)^\beta}, 0 < \alpha < 1, \beta \geq 0 \text{ e } d_\eta \text{ é a profundidade do nó } \eta. \tag{8.5}$$

Em cada nó a regra de particionamento segue o mesmo especificado para o modelo Bayesiano CART desenvolvido na secção 7.4: seleção uniforme entre as variáveis disponíveis para o particionamento e seleção uniforme entre os valores da variável selecionada para quebra do nó.

No modelo Bayesiano CART geramos uma única árvore ($m = 1$), no modelo BART estamos realizando uma soma de árvores, por isso é conveniente conservar cada árvore pequena. Chipman *et al.* (2010) recomenda $\alpha = 0{,}95$ e $\beta = 2$, esses valores alocam probabilidades de $0{,}05, 0{,}55, 0.28, 0.08$ e 0.03 para árvores com $1, 2, 3, 4$ e ≥ 5 nós terminais, respectivamente.

8.1.3 Uma Priori para σ

A escolha para $P(\sigma)$ recai sobre uma distribuição Gama Inversa:

$$\sigma^2|T \sim GamaInversa(\nu/2, \nu\lambda/2) \quad (\Leftrightarrow \nu\lambda/\sigma^2 \sim \chi^2_\nu). \tag{8.6}$$

Os valores para ν e λ são guiados pela escolha de σ que pode ser estimado por $\hat{\sigma}$. A primeira opção é tomar $\hat{\sigma}$ igual ao desvio padrão amostral de Y, a segunda é ajustar uma regressão linear $Y \times X$ usando mínimos quadrados e tomar $\hat{\sigma}$ sendo o desvio padrão dos resíduos. Tomamos $3 \leq \nu \leq 10$ e λ tal que o *q-ésimo* quantil da distribuição (8.6) seja $\hat{\sigma}$, *i.e.*, $P(\sigma < \hat{\sigma}) = q$.

8.1.4 Um Valor para M

Uma solução Bayesiana completa exigiria tratar M como uma variável aleatória e estabelecer uma priori. Outra solução é selecionar M através de validação cruzada. Pesa sobre ambas soluções um custo computacional que pode ser proibitivo. Chipman *et al.* (2010) sugerem que tomando $M = 200$ como valor fixo que gera bons resultados. A tendência é que a performance preditiva aumente com o incremento de M e depois passe a degradar a partir de um certo ponto.

8.2 Cálculo das Posteriores

O modelo bayesiano adotado induz à necessidade de calcular a distribuição à posteriori a partir dos dados observados na forma

$$P((T_1, \theta_1), \ldots, (T_m, \theta_m), \sigma|\mathbf{y}). \tag{8.7}$$

Como de praxe não é possível calcular essa posteriori analiticamente. Para tomar amostras dessa densidade usamos o algoritmo *backfitting Bayesiano* proposto por Hastie e Tibshirani (2000) baseado em MCMC usando um amostrador de Gibbs.

Considere $(T_{(m)}, \theta_{(m)})$ o conjunto das $M-1$ árvores extraído das M árvores excluindo o par (T_m, θ_m). O amostrador de Gibbs requer M amostragens sucessivas de (T_m, θ_m) condicionadas em $(T_{(m)}, \theta_{(m)}, \sigma)$:

$$(T_m, \theta_m)|(T_{(m)}, \theta_{(m)}, \sigma, \mathbf{y}), m = 1, \ldots, M, \tag{8.8}$$

seguida de uma amostra de σ considerando todas as m árvores:

$$\sigma|(T_1, \theta_1), \ldots, (T_m, \theta_m), \mathbf{y}. \tag{8.9}$$

Para amostrar (8.8) notamos que $P(T_m, \theta_m|T_{(m)}, \theta_{(m)}, \sigma, Y)$ depende de $(T_{(m)}, \theta_{(m)}, Y)$ somente através de

$$R_m = \mathbf{y} - \sum_{k \neq m} g(\mathbf{x}; T_k, \theta_k), \tag{8.10}$$

vetor de resíduos parciais que exclui a *m-ésima* árvore. Reformulando (8.8), obtemos

$$(T_m, \theta_m)|R_m,\sigma, m = 1, \ldots M, \tag{8.11}$$

que é equivalente à posteriori do modelo de uma única árvore $R_m = g(\mathbf{x}; T_m, \theta_m, \sigma) + \epsilon$ trocando $\mathbf{y}$ por R_m. Do modelo Bayesiano CART, combinando (7.8) e (7.17), obtemos

$$P(T_m|R_m, \sigma) \propto P(T_m) \int P(R_m|\theta_m, T_m, \sigma)\, P(\theta_m|T_m, \sigma)\, d\theta_m. \tag{8.12}$$

Como utilizamos uma priori conjugada para θ_m, (8.12) pode ser obtida em forma fechada, a menos de uma constante. A amostragem de Gibbs pode ser realizada sucessivamente em dois passos

$$T_m|R_m,\sigma \tag{8.13}$$

$$\theta_m|T_m,R_m,\sigma \tag{8.14}$$

As amostras de T_m em (8.13) podem ser geradas a partir dos procedimentos descritos na Secção 7.5 para o modelo Bayesiano CART. A diferença é que no modelo Bayesiano CART as probabilidades de seleção das operações GROW, PRUNE, CHANGE e SWAP eram iguais (0,25). Agora as probabilidades propostas são as seguintes: para crescer um nó terminal 0,25, para podar 0,35, para trocar um regra em um nó interno 0,40 e para trocar uma regra entre nós internos 0,10.

A probabilidade de aceitação $(T_i \to T^*)$ do algoritmo Metropolis-Hasting (7.19) torna-se:

$$\alpha(T_i, T^\star) = \min\left\{1, \frac{q(T^\star, T_i)}{q(T_i, T^\star)} \frac{P\left(R_i|X, T^*, \theta_i\right)}{P\left(R_i|X, T, \theta_i\right)} \frac{P(T^*)}{P(T_i)}\right\}. \tag{8.15}$$

As amostras θ_m de (8.14) são obtidas a partir dos μ_{im}'s das distribuições normais dos nós terminais. Estes valores servem para o cálculo subsequente de $R_{(m)+1}$ que servirá para a próxima amostragem de T_{m+1}. O cálculo de σ^2 pode ser feito com a distribuição a posteriori (Gama Inversa) usando os resíduos do modelo completo $\epsilon = \mathbf{y} - \sum_{m=1}^{M} g(\mathbf{x}; T_m, \theta_m)$.

Esquematicamente, uma única iteração do amostrador de Gibbs, adaptado de Kapelner e Bleich (2016), para M árvores pode ser mostrado na seguinte sequência

$$
\begin{array}{rcl}
\text{iteração} & & \text{amostra} \\
1 & : & T_1|R_1,\sigma \\
2 & : & \theta_1|T_1,R_1,\sigma \\
3 & : & T_2|R_2,\sigma \\
4 & : & \theta_2|T_2,R_1,\sigma \\
\vdots & & \\
2m-1 & : & T_m|R_m,\sigma \\
2m & : & \theta_m|T_m,R_m,\sigma \\
2m+1 & : & \sigma|T_1,\theta_1,\ldots,T_m,\theta_m
\end{array}
$$

Tipicamente 200 iterações servem para iniciar (*burn-in*) o processo e mais 1000 iterações para finalizar.

8.3 Inferência sobre Y

O procedimento MCMC descrito na seção anterior gera uma sequência (T_m,θ_m),$\sigma, j=1,\ldots,m$ que converge para a posteriori $P((T_1,\theta_1),\ldots,(T_m,\theta_m),\sigma|\mathbf{y})$. Considere $(T_1^\star,\theta_1^\star),\ldots,(T_m^\star,\theta_m^\star)$ uma sequência gerada pelo processo, após o período de *burn-in*, sendo $f^\star(\cdot)$ a soma

$$f^\star(\cdot)=\sum_{m=1}^{M} g(\cdot;T_m^\star,\theta_m^\star). \tag{8.16}$$

Uma sequência de amostras de $f^\star(\cdot)$ converge para $P(f|\mathbf{y})$, a distribuição a posteriori de $f(\cdot)$. Assim, uma sequência $f_1^\star,\ldots,f_K^\star$, pode ser considerada uma amostra de tamanho K de $P(f|\mathbf{y})$. Logo, para estimar $f(\mathbf{x})$, ou prever Y a partir de um valor $\mathbf{x}$, podemos usar

$$\hat{f}(\mathbf{x})=\frac{1}{K}\sum_{k=1}^{K} f_k^\star(\mathbf{x}) \tag{8.17}$$

que se aproxima da média da posteriori $E(f(\mathbf{x})|\mathbf{y})$. Pode-se alternativamente escolher como previsor a mediana da sequência $f_1^\star,\ldots,f_K^\star$ e/ou calcular intervalos de credibilidade para $f(\mathbf{x})$.

8.4 BART para Classificação

O modelo BART desenvolvido até aqui é para variável resposta contínua. Para variáveis categóricas, necessitamos de algumas modificações. Vamos considerar respostas binárias $Y=1$ ou $Y=0$. Sejam

$$P(\mathbf{x})=P(Y=1|\mathbf{x})=\Phi[G(\mathbf{x})], \tag{8.18}$$

em que

$$G(\mathbf{x})=\sum_{m=1}^{M} g(\mathbf{x};T_m,\theta_m) \tag{8.19}$$

e $\Phi(.)$ é função de distribuição acumulada da normal. O modelo (8.18) implicitamente assume $\sigma=1$, necessitamos apenas uma priori para $(T_1,\theta_1),\ldots,(T_m,\theta_m)$, a expressão (8.3) toma a forma

$$P((T_1,\theta_1),\ldots,(T_m,\theta_m))=\prod_{m=1}^{M}\left(\prod_{k=1}^{J_m} P(\mu_{km}|T_m)\right)P(T_m). \tag{8.20}$$

Para $P(T_m)$ adotamos o mesmo procedimento descrito na Secção 8.1.2 e para $P(\mu_{km}|T_m)$ usando a mesma abordagem da Secção 8.1.1, tomamos o intervalo $[(\Phi(-3{,}0),\Phi(3{,}0)])$ para $P(\mathbf{x})$ e esta

escolha leva a

$$\mu_{km} \sim N(0, \sigma_\mu^2) \quad \text{onde } \sigma_\mu = 3{,}0/k\sqrt{M}. \tag{8.21}$$

Tipicamente $k = 2$ é a escolha automática, como no modelo de regressão. Para utilizar o amostrador de Gibbs proposto na Secção (8.2), seguindo Albert e Chib (1993) criamos as variáveis latentes $Z_1, \ldots, Z_n$, n é o número de observações, da seguinte forma

$$\begin{aligned} Z_i|y_i = 1 &\sim max\left\{N\big(G(\mathbf{x}), 1\big), 0\right\}, \\ Z_i|y_i = 0 &\sim min\left\{N\big(G\big(\mathbf{x}\big), 1\big), 0\right\}. \end{aligned} \tag{8.22}$$

e substituímos $\mathbf{z}$ por $\mathbf{y}$ em (8.2).

Analogamente à sequência de árvores geradas em (8.3):

$$\begin{aligned} P^\star(\cdot) &= \Phi\left[G(\cdot)\right] \\ &= \Phi\left[\sum_{m=1}^{M} g(\cdot; T_m^\star, \theta_m^\star)\right]. \end{aligned} \tag{8.23}$$

pode ser usada para gerar uma sequência $p_1, \ldots, p_K$ que converge para $P(\cdot)$ e podemos utilizá-la da mesma forma que utilizamos $f_1^\star, \ldots, f_K^\star$.

8.5 Ilustração

Vamos ilustrar o cálculo da probabilidade de aceitação do algoritmo Metropolis-Hastings para as modalidades de transição GROW e PRUNE. Os cálculos são adaptados de Kapelner e Bleich (2016). Como vimos em (8.15):

$$\alpha(T_i, T^\star) = \min\left\{1, \underbrace{\frac{q(T^\star, T_i)}{q(T_i, T^\star)}}_{\text{razão da transição}} \times \underbrace{\frac{P\left(R_i|X, T^*, \theta_i\right)}{P\left(R_i|X, T, \theta_i\right)}}_{\text{razão da verossimilhança}} \times \underbrace{\frac{P(T^*)}{P(T_i)}}_{\text{razão da estrutura das árvores}}\right\}. \tag{8.24}$$

8.5.1 GROW

Razão da probabilidade de transição

A transição GROW propõe o crescimento da árvore criando dois novos nós a partir de um nó terminal:

$$\begin{aligned} q(T^*, T_i) &= P(\text{GROW}) \\ &\quad \times P(\text{selecionar o nó } \eta \text{ a partir qual a árvore cresce}) \\ &\quad \times P(\text{selecionar uma variável entre as disponíveis em } \eta) \\ &\quad \times P(\text{selecionar um valor na variável selecionada}) \\ &= P(\text{GROW}) \times \frac{1}{b} \times \frac{1}{p} \times \frac{1}{n}. \end{aligned} \tag{8.25}$$

em que

$$\begin{aligned} b &= \text{número de nós terminais,} \\ \eta &= \text{nó terminal selecionado entre os } b \text{ disponívies,} \\ p &= \text{número de variáveis preditoras disponíveis em } \eta, \\ n &= \text{número de valores únicos disponíveis na variável selecionada.} \end{aligned} \tag{8.26}$$

O cálculo de $q(T_i, T^*)$ é cálculo do retorno à situação antes da operação GROW, *i.e.*, a execução da operação PRUNE:

$$\begin{aligned} q(T_i, T^*) &= P(PRUNE) \times P(\text{selecionar um nó interno para realizar a poda}) \\ &= P(PRUNE) \times \frac{1}{w_2^*}. \end{aligned} \tag{8.27}$$

em que w_2^* denota o número de nós internos que possuem apenas 2 nós filhos. Assim, a razão da probabilidade de transição é dada por

$$\frac{q(T^\star, T_i)}{q(T_i, T^\star)} = \frac{\dfrac{P(GROW)}{bpn}}{\dfrac{P(PRUNE)}{w_2^*}} = \frac{P(GROW)}{P(PRUNE)} \frac{w_2^*}{bpn}. \tag{8.28}$$

Se não houver variáveis disponíveis com dois ou mais valores disponíveis esta razão deve ser considerada zero.

Razão da verossimilhança

Ao realizar a operação de GROW a estrutura da árvore permanece a mesma exceto pelos novos dois nós terminais acrescentados. Para o cálculo da verossimilhança precisamos somente nos concentrar na folha, denotada por t, que pode ser particionada. Considere t_e e t_d os respectivos nós candidatos à esquerda e à direita como resultado da partição proposta. Então,

$$\begin{aligned} \frac{P\left(R_i|X, T^*, \theta_i\right)}{P\left(R_i|X, T, \theta_i\right)} &= \frac{P\left(R_{t_{(e,1)},i}, \ldots, R_{t_{(e,n_e)},i}|\sigma\right) P\left(R_{t_{(d,1)},i}, \ldots, R_{t_{(d,n_d)},i}|\sigma\right)}{P\left(R_{1,i}, \ldots, R_{n_t,i}|\sigma\right)} \\ &= \sqrt{\frac{\sigma^2\left(\sigma^2 + n_t\sigma_\mu^2\right)}{\left(\sigma^2 + n_e\sigma_\mu^2\right)\left(\sigma^2 + n_d\sigma_\mu^2\right)}} \times \\ &\exp\left[\frac{\sigma_\mu^2}{2\sigma^2}\left(\frac{\left(\sum_{k=1}^{n_e} R_{t_{(e,k),i}}\right)^2}{\sigma^2 + n_e\sigma_\mu^2} + \frac{\left(\sum_{k=1}^{n_d} R_{t_{(d,k),i}}\right)^2}{\sigma^2 + n_d\sigma_\mu^2} - \frac{\left(\sum_{k=1}^{n_t} R_{t_{(t,k),i}}\right)^2}{\sigma^2 + n_t\sigma_\mu^2}\right)\right], \end{aligned} \tag{8.29}$$

em que n_e, n_d representam o número de observações alocadas em t_e e t_d, respectivamente, e, $n_t = n_e + n_d$.

Razão da estrutura das árvores

Conforme visto em (7.7) e (8.5), dada uma árvore T, a probabilidade de um nó η particionar depende da profundidade do nó d_η e dos parâmetros α e β ajustados conforme a expressão:

$$P(\eta, T) = \frac{\alpha}{(1 + d_\eta)^\beta}. \tag{8.30}$$

Considere H o conjunto dos nós terminais e I o conjunto de nós internos. Para a estrutura total

de uma árvore T,

$$P(T) = \prod_{\eta \in H} (1 - P(\eta, T)) \times \prod_{\eta \in I} P(\eta, T) \times \prod_{\eta \in I} P_{regra}(\eta), \tag{8.31}$$

em p_{reagra} é a probabilidade de que uma variável preditora seja selecionada e um correspondente ponto de corte entre os valores únicos da variável, cf. secção 7.4. Usando a notação da Razão da probabilidade de transição, $P_{regra}(\eta) = 1/p \times 1/n$.

Novamente, quando a transição GROW é selecionada para um nó η a árvore proposta difere da atual somente em relação aos dois novos nós filhos: η_e e η_d. A razão das estrutura pode ser escrita na forma

$$\begin{aligned}
\frac{P(T^*)}{P(T)} &= \frac{\prod_{\eta \in H^*} (1 - P(\eta, T^*)) \times \prod_{\eta \notin H^*} P(\eta, T^*) \times \prod_{\eta \in H^*} P_{regra}(\eta)}{\prod_{\eta \in H} (1 - P(\eta, T)) \times \prod_{\eta \notin H} P(\eta, T) \times \prod_{\eta \in H} P_{regra}(\eta)} \\
&= \frac{[1 - P(\eta_e, T^*)]\,[1 - P(\eta_d, T^*)]\; P(\eta, T)\, P_{regra}(\eta)}{1 - P(\eta, T)} \\
&= \frac{\left(1 - \dfrac{\alpha}{(1 + d_{\eta_e})^{\beta}}\right)\left(1 - \dfrac{\alpha}{(1 + d_{\eta_d})^{\beta}}\right) \dfrac{\alpha}{(1 + d_{\eta})^{\beta}} \dfrac{1}{p} \dfrac{1}{n}}{1 - \dfrac{\alpha}{(1 + d_{\eta})^{\beta}}} \\
&= \alpha \frac{\left(1 - \dfrac{\alpha}{(2 + d_{\eta})^{\beta}}\right)^2}{[(1 + d_{\eta})^{\beta} - \alpha]\, pn},
\end{aligned} \tag{8.32}$$

uma vez que $d_{\eta_e} = d_{\eta} + 1$.

8.5.2 PRUNE

A operação de PRUNE é a contrapartida da operação GROW. Os cálculos são aproximadamente o inverso dos cálculos já realizados para GROW. Note que árvores com apenas o nó raiz não são elegíveis para poda.

Razão da probabilidade de transição

$$\begin{aligned}
q(T^*, T_i) &= P(PRUNE) \times P(\text{selecionar um nó interno para realizar a poda}) \\
&= P(PRUNE) \times \frac{1}{w_2}.
\end{aligned} \tag{8.33}$$

em que w_2 é um nó interno com apenas dois filhos. A transição na direção oposta é idêntico ao já realizado na operação GROW (8.25), exceto que existem $b - 1$ nós internos disponíveis:

$$\begin{aligned}
q(T_i, T^*) &= P(\text{GROW}) \\
&\quad \times P(\text{selecionar o nó } \eta^* \text{ a partir qual a árvore cresce}) \\
&\quad \times P(\text{selecionar uma variável entre as disponíveis em } \eta^*) \\
&\quad \times P(\text{selecionar um valor na variável selecionada}) \\
&= P(\text{GROW}) \times \frac{1}{b - 1} \times \frac{1}{p^*} \times \frac{1}{n^*}.
\end{aligned} \tag{8.34}$$

Assim, a razão da probabilidade de transição torna-se

$$\frac{q(T^*, T_i)}{q(T_i, T^*)} = \frac{P(PRUNE) \times \dfrac{1}{w_2}}{P(\text{GROW}) \times \dfrac{1}{b-1} \times \dfrac{1}{p^*} \times \dfrac{1}{n^*}} = \frac{P(PRUNE)}{P(GROW)} \frac{(b-1)p^* n^*}{w_2} \tag{8.35}$$

Razão da verossimilhança

A razão da verossimilhança é o inverso da razão da operação GROW:

$$\frac{P\left(R_i|X, T^*, \theta_i\right)}{P\left(R_i|X, T, \theta_i\right)} = \sqrt{\frac{\left(\sigma^2 + n_e\sigma_\mu^2\right)\left(\sigma^2 + n_d\sigma_\mu^2\right)}{\sigma^2\left(\sigma^2 + n_t\sigma_\mu^2\right)}} \times \exp\left[\frac{2\sigma^2}{\sigma_\mu^2}\left(\frac{\left(\sum_{k=1}^{n_t} R_{t_{(t,k),i}}\right)^2}{\sigma^2 + n_t\sigma_\mu^2} - \frac{\left(\sum_{k=1}^{n_e} R_{t_{(e,k),i}}\right)^2}{\sigma^2 + n_e\sigma_\mu^2} - \frac{\left(\sum_{k=1}^{n_d} R_{t_{(d,k),i}}\right)^2}{\sigma^2 + n_d\sigma_\mu^2}\right)\right]. \tag{8.36}$$

Razão da estrutura das árvores

É simplesmente o inverso da razão da estrutura da operação GROW:

$$\frac{P(T^*)}{P(T)} = \frac{\left[(1+d_\eta)^\beta - \alpha\right] p^* n^*}{\alpha\left(1 - \dfrac{\alpha}{(2+d_\eta)^\beta}\right)^2}. \tag{8.37}$$

9 Simulação

9.1 Objetivo

O objetivo é conduzir uma simulação para os modelos de árvores de regressão a partir de dados simulados. A avaliação dos modelos é baseada no erro quadrático médio *mse* (*mean square error*) obtido a partir das predições de cada modelo ajustado numa base teste. Quanto menor o *mse*, melhor o desempenho do modelo. No caso das árvores de classificação, utilizamos como medida desempenho o erro de classificação, *i.e.*, o percentual de observações classificadas erradamente, quanto menor o erro melhor o modelo.

Utilizamos a linguagem de programação Python[1] e as respectivas bibliotecas para gerar os ajustes. Usamos a versão padrão dos parâmetros de configuração como eles se apresentam em cada função de ajuste. Nosso objetivo é comparar os ajustes de acordo com os valores *default* e não em *tunar* cada modelo procurando o menor *mse* em cada ajuste.

9.2 Árvores de Regressão

Os modelos comparados são os seguintes: CART, *Random Forest* (Florestas Aleatórias), *Gradiente Boosting*, XGBoost e BART. A ideia central é verificar o quanto as inovações surgidas a partir do CART melhoram o desempenho da metodologia diminuindo os erros de predição e, ao mesmo tempo vamos verificar, sob as condições simuladas, se existe algum modelo de árvore que apresenta um desempenho consistentemente melhor do que os outros.

No artigo sobre *Bagging*, Breiman (1996b) simulou três modelos gerados a partir da especificação de Friedman (1991), Breiman (1996b) utilizou uma amostra de tamanho 200, aplicou o *Bagging* e mediu o erro em uma amostra de teste de tamanho 1000. Simulamos os mesmos modelos fixando a base teste em 2000 amostras e a base treino variando de 100 a 2000 amostras, de 100 em 100. Isto nos permite avaliar o quanto o tamanho da amostra de treino influencia na estimativa do erro populacional. Cada modelo de árvore foi ajustado na mesma amostra para que haja comparabilidade. O *script* para executar a simulação pode ser visto em B.1.

9.2.1 Especificação das Expressões de Friedman

O primeiro modelo, *Friedman#1*, é baseado em 10 variáveis preditoras $\mathbf{x} = \{x_1, x_2, \ldots, x_{10}\}$ distribuídas uniformemente no intervalo $[0, 1]$. A variável resposta segue a seguinte fórmula:

$$Friedman\,\#1: \qquad y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0{,}5)^2 + 10x_4 + 5x_5 + \epsilon, \tag{9.1}$$

em que $\epsilon \sim N(0, 1)$. As variáveis $x_6, \ldots, x_{10}$ são inseridas como ruído puro.

Os modelos *Friedman#2* e *Friedman#3* simulam a impedância e a mudança de fase em um circuito de corrente alternada. Cada modelo tem 4 variáveis com as seguintes formulações:

$$Friedman\,\#2: \qquad y = \underbrace{(x_1^2 + (x_2 x_3 - 1/x_2 x_4))^2)^{1/2}}_{\text{sinal}} + \underbrace{\epsilon_2}_{\text{ruído}} \tag{9.2}$$

[1] https://www.python.org/

e

$$Friedman\,\#3: \qquad y = \underbrace{\tan^{-1}\left(\frac{x_2x_3 - (1/x_2x_4)}{x1}\right)}_{\text{sinal}} + \underbrace{\epsilon_3}_{\text{ruído}}, \tag{9.3}$$

em que x_1, x_2, x_3, x_4 são distribuídos uniformemente nos seguintes intervalos:

$$\begin{aligned} 0 \leq x_1 &\leq 100, \\ 40\pi \leq x_2 &\leq 560\pi, \\ 0 \leq x_3 &\leq 1, \\ 1 \leq x_4 &\leq 11. \end{aligned} \tag{9.4}$$

Os ruídos tem distribuição normal em que $\epsilon_2 \sim N(0, \sigma_2^2)$ e $\epsilon_3 \sim N(0, \sigma_3^2)$. Os valores de σ_2 e σ_3 são escolhidos de maneira que a razão sinal/ruído seja $3:1$, de forma que a variância explicada pelo sinal é de 90%:

$$\text{razão sinal/ruído} = \frac{\sigma(\text{sinal})}{\sigma_j} \implies \frac{var(\text{sinal})}{var(\text{sinal}) + var(\epsilon_j)} = 0{,}90,\ j = 1, 2. \tag{9.5}$$

Os valores são os seguintes: $\sigma_2 = 125$ e $\sigma_3 = 0{,}105$.

9.2.2 Resultados

Resultados: *Friedman#1*

A Figura 9.1 contém o resultado da média dos *mse*'s das simulações do *Friedman#1*, as barras representam a adição e subtração de 1 desvio padrão. A partir de 1500 observações na amostra de treino a curva da média tende a se estabiliza. A árvore CART apresenta sistematicamente os piores resultados, o que nos mostra o valor das modificações incluídas nos demais modelos. Gradiente Boosting e BART competem pelo melhor desempenho sendo que o BART torna-se ligeiramente melhor na medida em a amostra de treino cresce. XGBoost se sai melhor *Random Forest*. O resultado completo, com todos os tamanhos de amostra, podem ser vistos no Apêndice A.1.

A Tabela 9.1 apresenta o resultado das simulações para o tamanho de amostra de treino 2000. A redução percentual em relação ao CART varia de 57,8% (*Random Forest*) a 76,1% (BART).

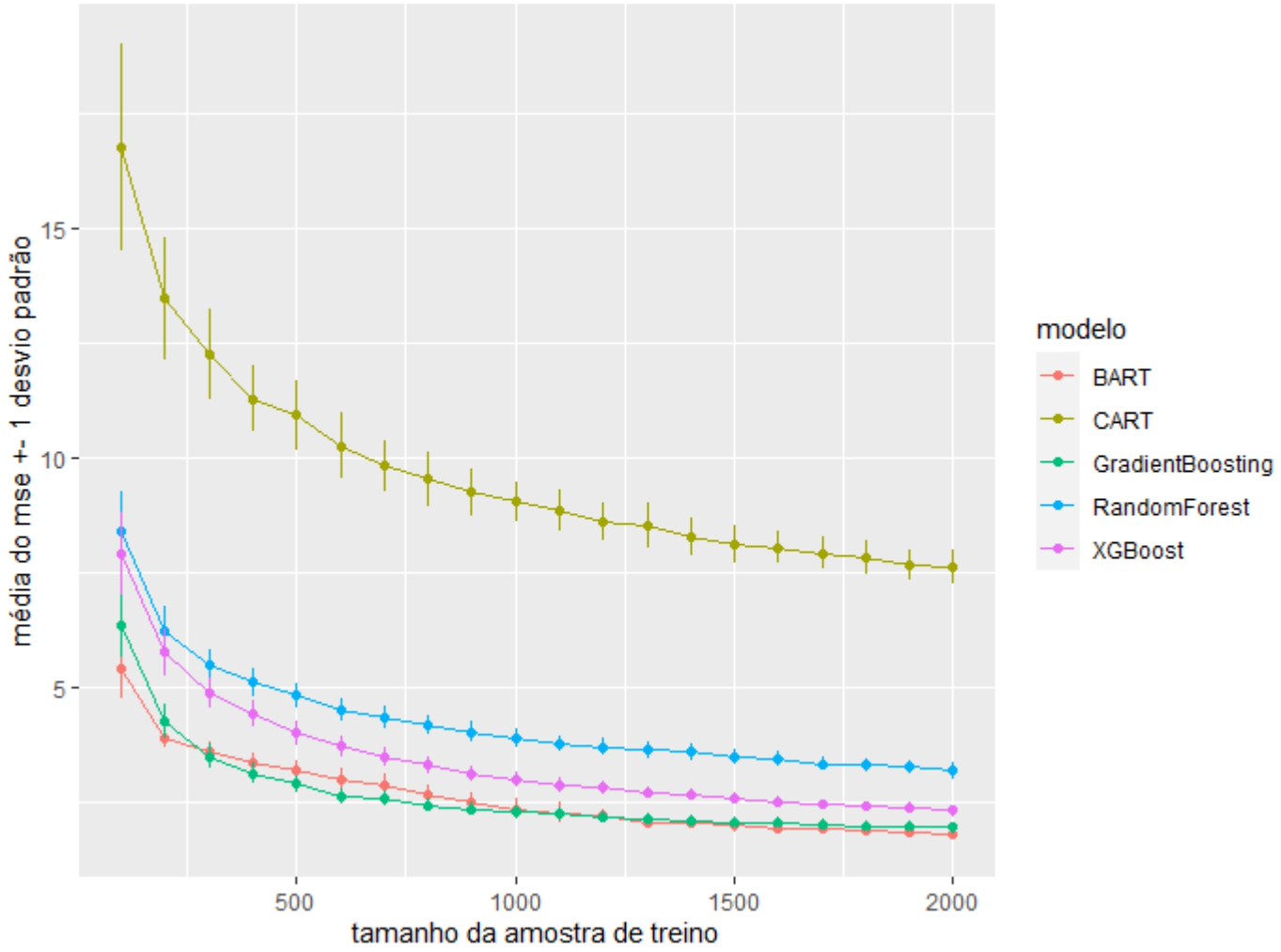
Tabela 9.1: *Simulação Friedman#1.*

Amostra de Treino: N = 2000					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	7,618	1,821	1,952	3,213	2,317
desv.pad.	0,361	0,095	0,11	0,166	0,097
mínimo	6,649	1,532	1,708	2,813	2,114
máximo	8,39	2,084	2,268	3,672	2,568
Δ% CART	0,0%	76,1%	74,4%	57,8%	69,6%

Resultados: *Friedman#2*

A Figura 9.2 contém o resultado da média dos *mse*'s das simulações do *Friedman#2*, as barras representam a adição e subtração de 1 desvio padrão. A partir do tamanho de amostra de treino 500 as árvores ajustadas por Gradiente Boosting apresentam em média o melhor resultado. As árvores BART parecem se beneficiar, mais do que os demais ajustes, do aumento no tamanho da amostra de treino, chegando a ultrapassar o XGBoost a partir do tamanho 1200. A partir do tamanho de

Figura 9.1: Friedman #1*: MSE - Média de 100 simulações para cada amostra.*



amostra 1500 os erros tendem a se estabilizar, no entanto, as árvores BART ainda apresentam um decréscimo no intervalo até 2000. As árvores CART, novamente, apresentam o pior desempenho, mesmo com o aumento no tamanho da amostra muito pouco se modifica a média do *mse* a partir do tamanho 1000. O modelo *Random Forest* sistematicamente permanecem com o segundo melhor desempenho a partir da amostra de tamanho 500.

A Tabela 9.2 mostra o resultado da simulação para o tamanho da amostra de treino 2000 em que notamos o ganho apresentado pelo Gradiente Boosting, o qual reduz o erro em quase 50% em relação ao CART, a menor redução é alcançada pelo XGBoost, 40,0%. No Apêndice A.2 pode-ser verificar o resultado para todos os tamanhos de amostra.

Tabela 9.2: *Simulação Friedman#2.*

Amostra de Treino: N = 2000					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33177,6	18994,9	16916,1	18086,1	19898,9
desv.pad.	1278,9	689,2	537,7	558,5	627,1
mínimo	30008,3	17575,3	15879,9	16640,1	18425,7
máximo	36193,3	20843,3	18452,0	19462,0	21737,0

Δ% CART	0,0%	42,7%	49,0%	45,5%	40,0%

Resultados: *Friedman#3*

A Figura 9.3 apresenta as médias dos *mse*'s dos ajustes dos modelos de árvores estudados. As árvores CART notadamente apresentam o pior desempenho para todos os tamanhos de amostra de treino. Gradiente Boosting e BART competem como melhores ajustes para praticamente todos os tamanhos de amostra. XGBoost apresenta consistentemente o terceiro melhor desempenho, enquanto que as Florestas Aleatórias apresentam o segundo pior desempenho, mesmo assim desempenho melhor que o CART.

A Tabela 9.3 mostra que a variação de ganho em relação ao CART vai de 29,5% (BART) a 49,8% (Gradiente Boosting). O resultado completo com todos os tamanhos de amostra de treino pode ser consultado no Apêndice A.2.

Tabela 9.3: *Simulação Friedman#3.*

Amostra de Treino: N = 2000					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,028	0,020	0,014	0,014	0,016
desv.pad.	0,001	0,001	0,001	0,001	0,001
minimo	0,025	0,017	0,013	0,013	0,014
maximo	0,035	0,022	0,016	0,017	0,018
Δ% CART	0,0%	29,5%	49,8%	49,0%	43,7%

9.3 Árvores de Classificação

Para avaliar o desempenho dos modelos de árvore de classificação selecionamos os seguintes modelos: CART, AdaBoost.SAMME, Gradiente Boosting e XGBoost. Simulamos os mesmos modelos fixando a base teste em 2000 amostras e a base treino variando de 100 a 2000 amostras, de 100 em 100. Isto nos permite avaliar o quanto o tamanho da amostra de treino influencia na estimativa do erro populacional. Cada modelo de árvore foi ajustado na mesma amostra para que haja comparabilidade. O *script* para executar a simulação pode ser visto em B.2.

Escolhemos realizar a simulação utilizando as equações de *ondas* (*waveform*) proposto por Breiman *et al.* (1984) e reutilizado por Zhu *et al.* (2006). Este é um problema cuja variável resposta tem 3 classes e 21 variáveis preditoras.

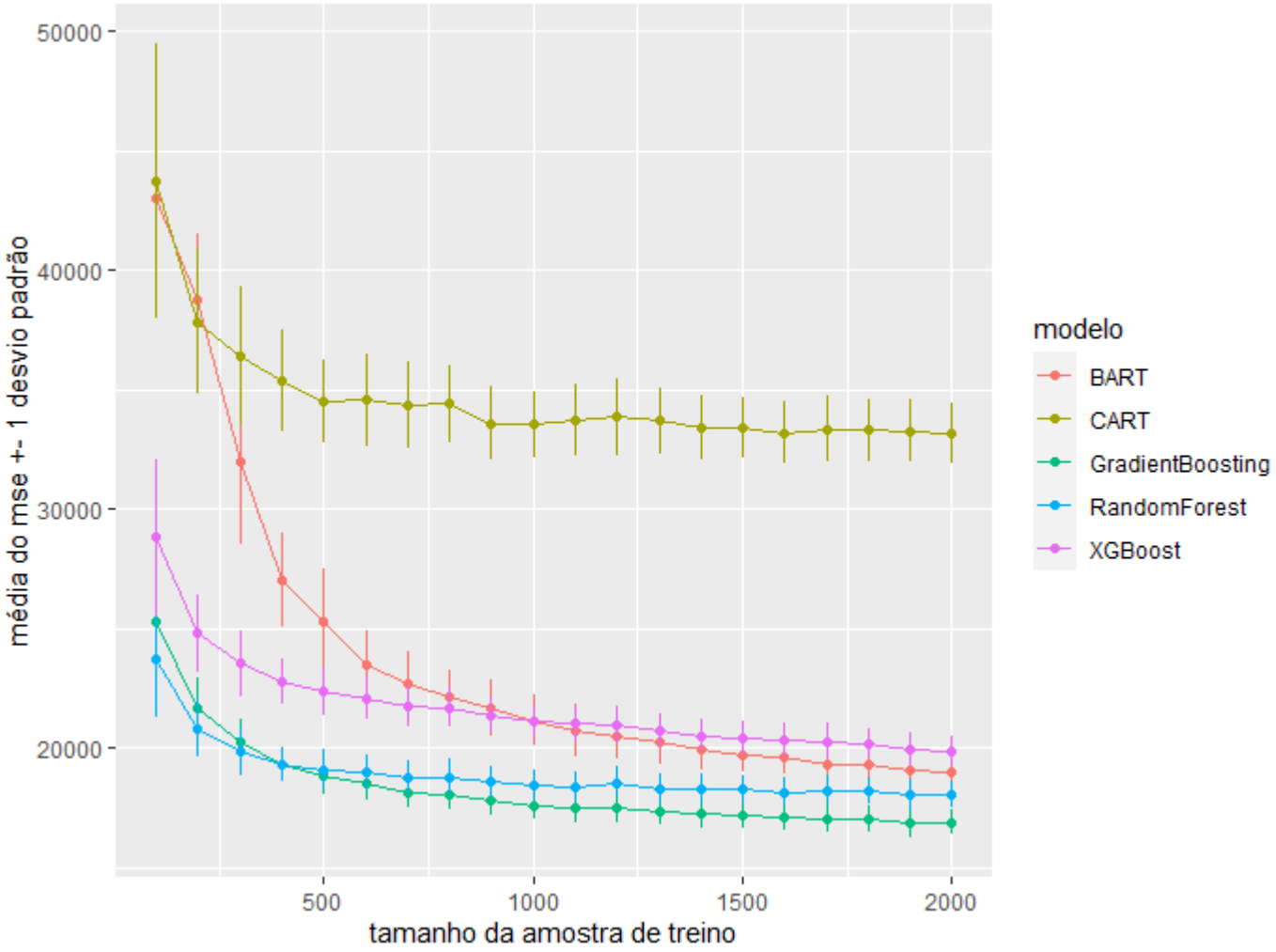
9.3.1 Especificação das equações das *waveforms*

As expressões que geram os dados simulados são as seguintes:

$$\mathbf{x}_j = \begin{cases} u \cdot v_1(j) + (1-u) \cdot v_2(j) + \epsilon_j, & \text{classe 1,} \\ u \cdot v_1(j) + (1-u) \cdot v_3(j) + \epsilon_j, & \text{classe 2,} \\ u \cdot v_2(j) + (1-u) \cdot v_3(j) + \epsilon_j, & \text{classe 3,} \end{cases} \tag{9.6}$$

em que $j = 1, 2, \ldots, 21$, $u \sim \mathcal{U}(0,1)$, $\epsilon_j \sim N(0,1)$, e as variáveis v_k são as ondas triangulares deslocadas: $v_1(j) = \max(6 - |j - 11|, 0)$, $v_2 = (j) = v_1(j-4)$ e $v_3(j) = v_1(j+4)$.

Figura 9.2: Friedman #2*: MSE - Média de 100 simulações para cada amostra.*



9.3.2 Resultados

A Figura 9.4 apresenta as médias dos erros de classificação obtidos para cada tamanho da amostra de treino. Notadamente, as árvores CART apresentam o pior desempenho. O AdaBoost.SAMME, em geral, permanece como o segundo pior desempenho para tamanho de amostra de treino a partir de 500. As predições geradas pelas árvores *Random Forest*, *Gradiente Boosting* e XGBoost apresentam desempenho muito semelhante, embora a *Random Forest*, na média seja ligeiramente melhor. A Tabela 9.4 mostra os resultados para a amostra de treino de tamanho 2000.

O desempenho de *Random Forest*, *Gradiente Boosting* e XGBoost são semelhantes quando consideramos o tamanho de amostra 2000, Tabela 9.4. A redução do erro gira em torno de 40%. O AdaBoost.SAMME é o que apresenta o pior desempenho, mas ainda assim reduz o erro em 30,5% em relação ao CART.

Figura 9.3: Friedman #3*: MSE - Média de 100 simulações para cada amostra.*

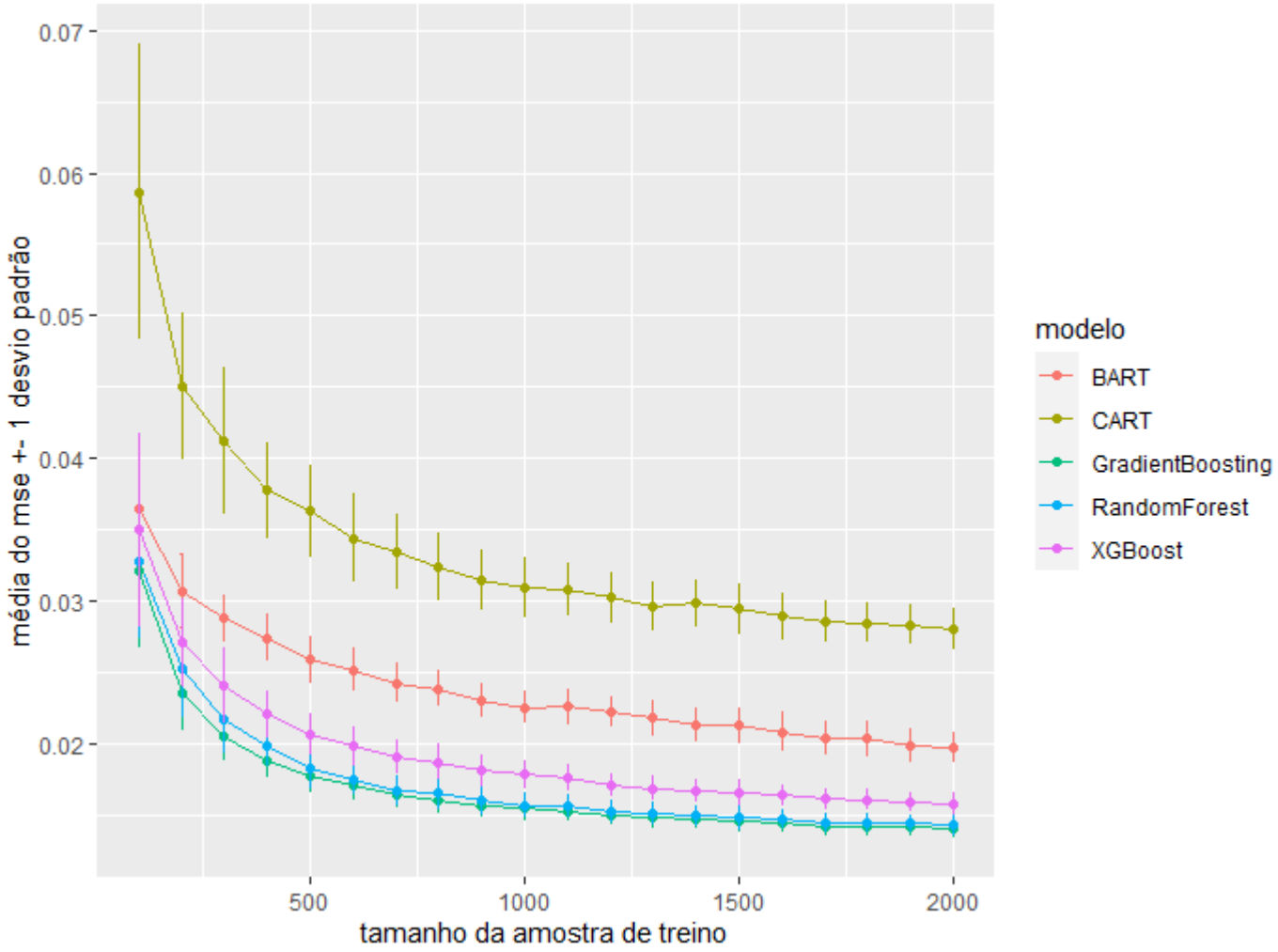
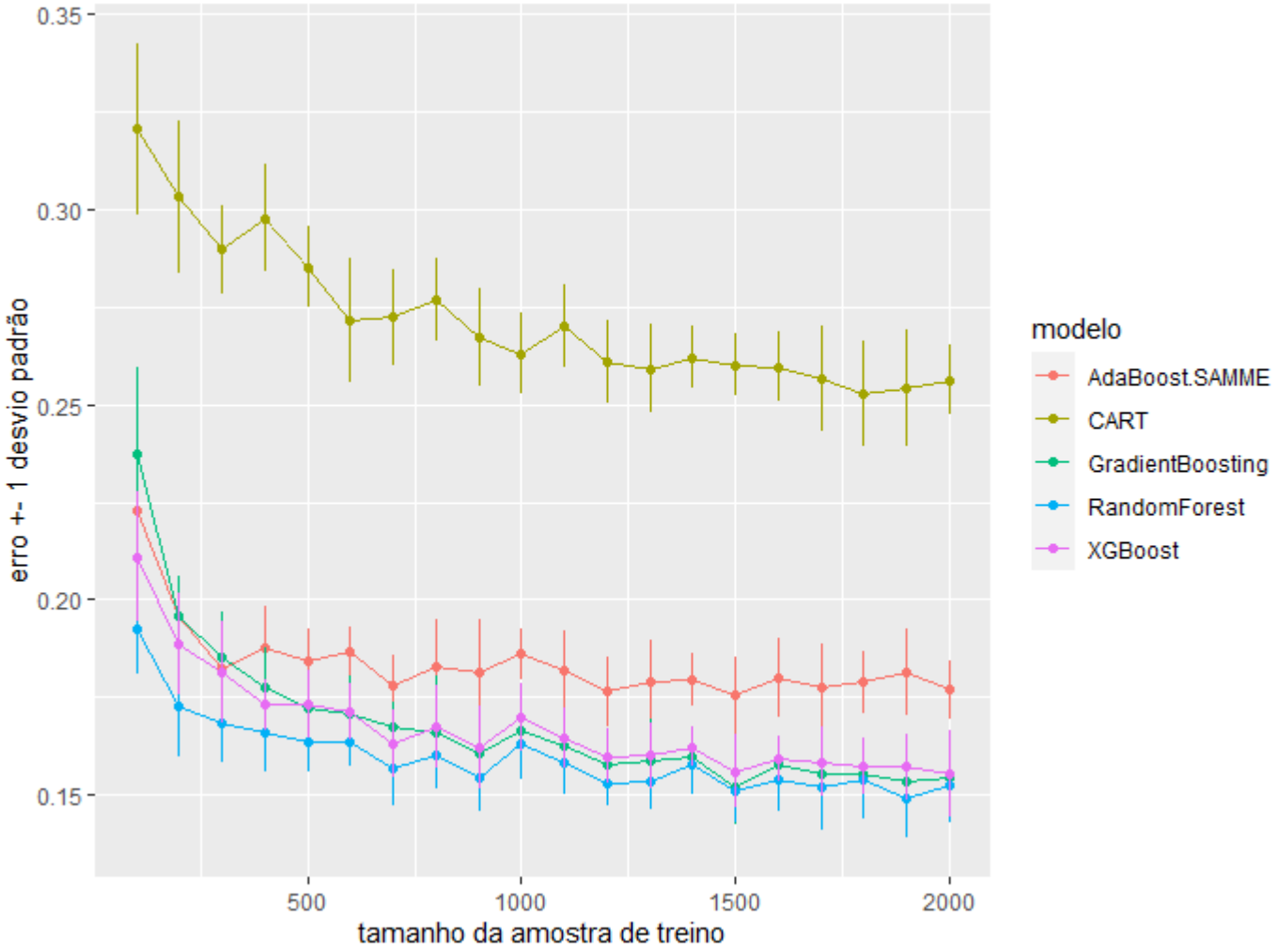


Tabela 9.4: *Simulação waveform.*

Amostra de Treino: N = 2000					
	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2563	0,1770	0,1526	0,1542	0,1553
desv.pad.	0,0089	0,0075	0,0096	0,0115	0,0112
mínimo	0,2435	0,1655	0,1405	0,1350	0,1395
máximo	0,2609	0,1808	0,1705	0,1720	0,1755
Δ% CART	0,0%	30,5%	40,5%	39,8%	39,4%

Figura 9.4: Waveform*: Erro de Classificação - Média de 100 simulações para cada amostra.*



10 Aplicação

Além da Simulação, Capítulo 9, aplicamos os modelos de árvores destacados neste trabalho em algumas bases de dados extraídas, algumas do site de competições de *machine learning*, https://www.kaggle.com/ e outras do site https://archive-beta.ics.uci.edu/.

Nosso objetivo é aplicar os modelos de árvores aos conjuntos sem nos preocuparmos com a *feature engineering*, *i.e.*, transformar e/ou criar novas variáveis a partir das existentes em cada base. Vamos nos ater a um mínimo de transformação nos dados para que as bibliotecas possam ser utilizadas, *e.g*, elas aceitam somente variáveis numéricas. A estratégia geral para as variáveis categóricas foi binarizar: cada categoria é transformada numa variável 0/1.

Para árvores de regressão avaliaremos cinco conjunto de dados (*datasets*) utilizando o *mse* ($mse = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$) como métrica de avaliação. Para as árvores de classificação, também selecionamos, cinco *datasets*, cujo critério de avaliação é a acurácia, *i.e.*, percentual de classificações corretas. Os mesmos modelos foram avaliados, em cada caso, por validação cruzada, nos mesmos subconjuntos de treino e teste. A validação cruzada foi realizada com dez rodadas, em cada rodada a base de treino representou 75% da base total e a base de teste 25%.

Os scripts em Python podem ser verificados no Apêndice C.1.

10.1 Árvores de Regressão

Abaixo apresentamos uma descrição de cada *dataset* utilizado, bem como a origem.

10.1.1 Descrição dos *datasets*

1. *dataset*: car-price audi[1].
   (a) variável resposta: preço de automóveis usados marca audi na Grã Bretanha (£) - média: 22897, desvio padrão: 11714.
   (b) $n = 10668$ casos.
   (c) $p = 8$ variáveis explicativas:
      i. model: 26 different models.
      ii. year: year of manufacture, 1997 a 2020.
      iii. transmission: Automatic, Manual, Semi-Auto.
      iv. mileage: miles driven.
      v. fuelType: Diesel, Petrol, Hybrid, Electric.
      vi. tax: tax in £.
      vii. mpg: consumption - miles per gallon.
      viii. engineSize: 0.0 a 6.3.
2. *dataset*: car-price bmw[2].
   (a) variável resposta: preço de automóveis usados marca bmw na Grã Bretanha (£) - média: 22733, desvio padrão: 11415.

[1] https://www.kaggle.com/aishwaryamuthukumar/cars-dataset-audi-bmw-ford-hyundai-skoda-vw, 20-08-2021.
[2] https://www.kaggle.com/aishwaryamuthukumar/cars-dataset-audi-bmw-ford-hyundai-skoda-vw, 20-08-2021.

(b) $n = 10781$ casos.

(c) $p = 8$ variáveis explicativas:

i. model: 24 different models.
ii. year: year of manufacture, 1996 a 2020.
iii. transmission: Automatic, Manual, Semi-Auto.
iv. mileage: miles driven.
v. fuelType: Diesel, Petrol, Hybrid, Electric, Other.
vi. tax: tax in £.
vii. mpg: consumption - miles per gallon.
viii. engineSize: 0.0 a 6.6.

3. *dataset*: insurance[3].

(a) variável resposta: custos médicos cobrados pelo seguro ($) - média: 13270, desvio padrão: 12105.

(b) $n = 1338$ casos.

(c) $p = 7$ variáveis explicativas:

i. age: age in years.
ii. sex: female, male.
iii. bmi: body mass index.
iv. children: number of dependents.
v. smoker: yes/no.
vi. region: northeast, southeast, southwest, northwest.

4. *dataset*: wine-quality[4].

(a) variável resposta: qualidade do vinho, escala de 0 a 10 - média: 5,64, desvio padrão: 0,81.

(b) $n = 1599$ casos.

(c) $p = 11$ variáveis explicativas - baseadas em testes físico-químicos:

i. fixed acidity.
ii. volatile acidity.
iii. citric acid.
iv. residual sugar.
v. chlorides.
vi. free sulfur dioxide.
vii. total sulfur dioxide.
viii. density.
ix. pH.
x. sulphates.
xi. alcohol.

5. *dataset*: students-performance[5].

(a) variável resposta: performance de estudantes, score de 0 a 100 - (math score+reading score+writing score)/3.

(b) $n = 1000$ casos.

[3] https://www.kaggle.com/mirichoi0218/insurance, 20-08-2021.
[4] https://www.kaggle.com/spscientist/students-performance-in-exams, 20-08-2021.
[5] https://www.kaggle.com/spscientist/students-performance-in-exams, 20-08-2021.

(c) $p = 5$ variáveis explicativas:

i. gender: female, male.

ii. race/ethnicity: group A, group B, ..., group E.

iii. parental level of education: some college, some high school, high school, bachelor's degree, master's degree, associate's degree.

iv. lunch: free/reduced, standard.

v. test preparation course: none, completed.

10.1.2 Resultados

A Tabela 10.1 apresenta o resultado com às médias dos *mse*'s obtidos.

Seguindo os mesmos resultados da Simulação, Capítulo 9, as árvores CART apresentam o pior resultado. Nos *datasets* analisados nenhum modelo de árvore performou em todos os casos melhor do que todos os outros. Nos dois *datasets* de preços de automóveis (car-price audi e car-price bmw) o destaque ficou para a *Random Forest* e XGBoost, com redução no *mse* em relação ao CART de pouco mais de 40%. Para o *dataset* insurance o destaque é para o BART e o Gradiente Boosting com uma redução de mais de 50% no *mse* em relação ao CART. No *dataset* wine-quality a *Random Forest* performou melhor reduzindo em 47,1% o *mse*. No *dataset* students-performance o destaque ficou para o Gradiente Boosting com uma redução de 21,2% em relação ao CART.

Nas Figuras 10.1 a 10.5 notamos o desvio padrão do *mse* obtido na validação cruzada mostra que eventualmente, em cada caso, a aplicação de um modelo ou outro levaria ao mesmo resultado, por exemplo, na Figura 10.1 as árvores CART e Gradiente Boosting algumas vezes apresentam os mesmos resultados.

Tabela 10.1: *Aplicação Árvores de Regressão - mse.*

dataset	CART	Random Forecast	Gradiente Boosting	XGBoost
car-price audi				
mse médio	8,70	5,27	7,79	5,20
desv.pad.	0,88	0,53	0,39	0,92
Δ% CART	0,0%	39,3%	10,5%	40,2%
car-price bmw				
mse médio	13,13	7,12	9,73	6,84
desv.pad.	2,58	1,97	2,09	1,94
Δ% CART	0,0%	45,8%	25,9%	47,9%
insurance				
mse médio	42,27	35,51	20,89	27,44
desv.pad.	2,89	3,31	3,52	3,29
Δ% CART	0,0%	44,4%	50,6%	35,1%
wine-quality				
média	0,66	0,35	0,39	0,39
desv.pad.	0,06	0,03	0,03	0,03
Δ% CART	0,0%	47,6%	41,2%	40,8%
students-performance				
mse médio	222,60	206,47	175,38	221,05
desv.pad.	17,27	17,34	15,63	15,43
Δ% CART	0,0%	7,3%	21,2%	0,7%

Figura 10.1: dataset *car-price-audi: mse - Média de 10 de amostras por validação cruzada.*

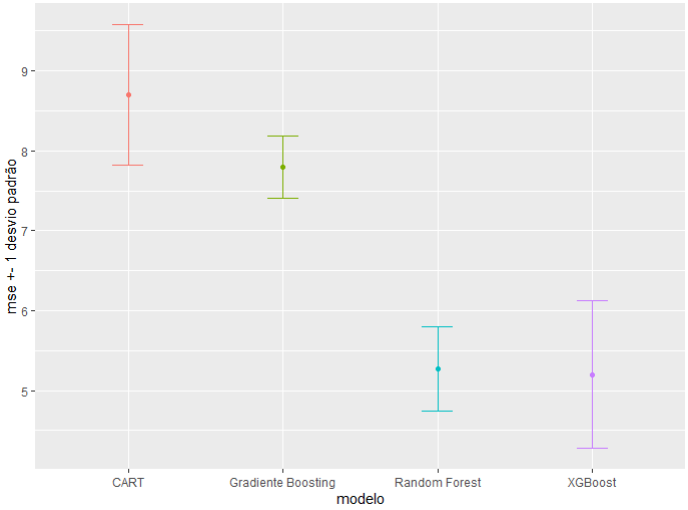
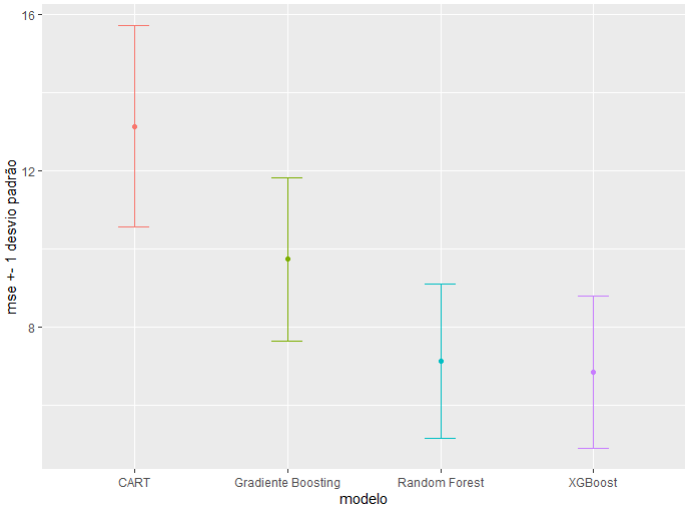


Figura 10.2: dataset *car-price-bmw: mse - Média de 10 de amostras por validação cruzada.*



10.2 Árvores de Classificação

Abaixo apresentamos uma descrição de cada *dataset* utilizado, bem como a origem.

10.2.1 Descrição dos *datasets*

1. *dataset*: bank-notes[6].

   (a) variável resposta: nota autêntica (yes/no).

   (b) $n = 1372$ casos.

   (c) $p = 4$ variáveis explicativas (*Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.*):

[6] https://www.kaggle.com/shantanuss/banknote-authentication-uci, 20-08-2021.

Figura 10.3: dataset *insurance: mse - Média de 10 de amostras por validação cruzada.*

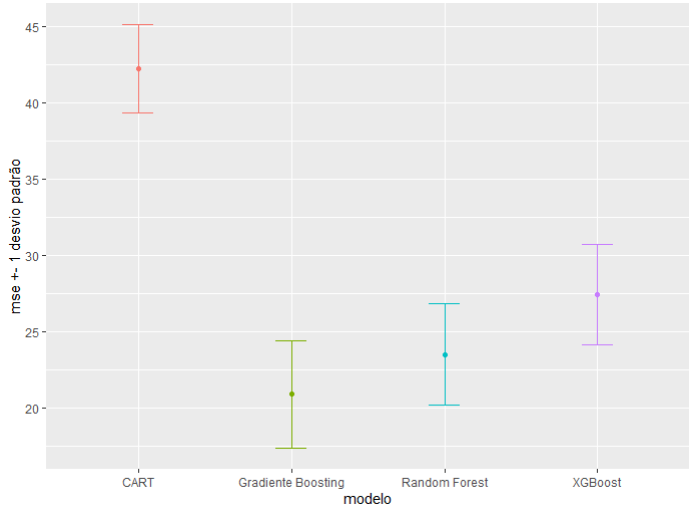
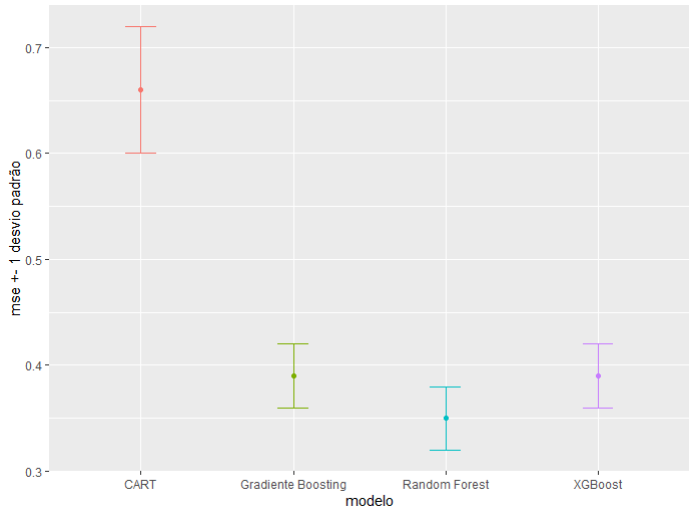


Figura 10.4: dataset *wine-quality: mse - Média de 10 de amostras por validação cruzada.*



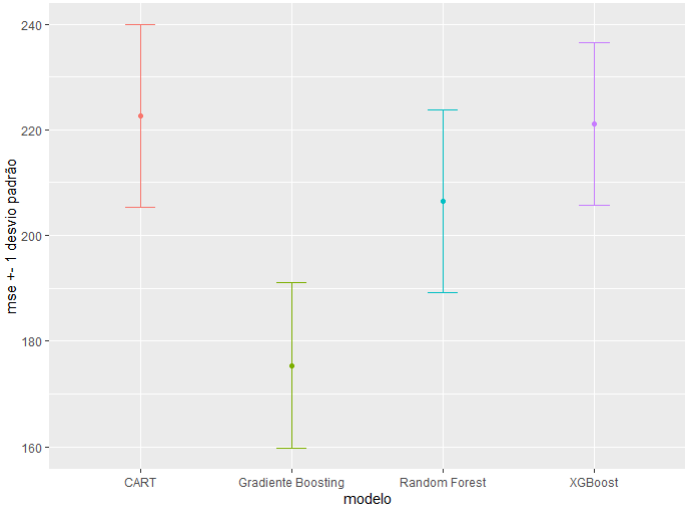
i. variance.
ii. skewness.
iii. curtosis.
iv. entropy.

2. *dataset*: bank-chunners [7].

(a) variável resposta: churn (yes/no).
(b) $n = 10127$ casos.
(c) $p = 19$ variáveis explicativas:
i. Customer-Age: customer age in years.
ii. Gender: male, female.
iii. Dependent-count: number of dependents.
iv. Education-Level: uneducated, college, high school, graduate, pos-graduate, doctorate unknown.

[7] https://www.kaggle.com/sakshigoyal7/credit-card-customers, 20-08-2021.

Figura 10.5: dataset *student-performance: mse - Média de 10 de amostras por validação cruzada.*



v. Marital-Status: divorced, single, married, unknown.
vi. Income-Category: less than $40K, $40k-$60K, $60K-$80K, $80K-$120K, $120K+, unknown.
vii. Card-Category: blue, gold, silver, platinum.
viii. Months-on-book: period of relationship with bank.
ix. Total-Relationship-Count: total number of products held by the customer.
x. Months-Inactive-12-mon: number of months inactive in the last 12 months.
xi. Contacts-Count-12-mon: number of contacts in the last 12 months.
xii. Credit-Limit: credit limit on the credit card ($).
xiii. Total-Revolving-Bal: total revolving balance on the credit card.
xiv. Avg-Open-To-Buy: optn to buy credit line (avg of last 12 months).
xv. Total-Amt-Chng-Q4-Q1: change in transaction amount (Q4 over Q1).
xvi. Total-Trans-Amt: total transaction amount (last 12 months).
xvii. Total-Trans-Ct: total transaction count (last 12 months).
xviii. Total-Ct-Chng-Q4-Q1: change in transaction count (Q4 over Q1).
xix. Avg-Utilization-Ratio: average card utilization ratio.

3. *dataset*: spam[8].

   (a) variável resposta: e-mail spam yes/no.

   (b) $n = 4601$ casos.

   (c) $p = 57$ variáveis explicativas: *word-freq-make,word-freq-address,word-freq-all, word-freq-3d, word-freq-our, word-freq-over, word-freq-remove, word-freq-internet, word-freq-order, word-freq-mail, word-freq-receive, word-freq-will, word-freq-people, word-freq-report, word-freq-addresses, word-freq-free, word-freq-business, word-freq-email, word-freq-you, word-freq-credit, word-freq-your, word-freq-font, word-freq-000, word-freq-money, word-freq-hp, word-freq-hpl, word-freq-george, word-freq-650, word-freq-lab, word-freq-labs, word-freq-telnet, word-freq-857, word-freq-data, word-freq-415, word-freq-85, word-freq-technology, word-freq-1999, word-freq-parts, word-freq-pm, word-freq-direct, word-freq-cs, word-freq-meeting, word-freq-original, word-freq-project, word-freq-re, word-freq-edu, word-freq-table, word-freq-conference, char-freq-semicolon, char-freq-left-parentesis, char-freq-left-bracket,*

[8] https://archive.ics.uci.edu/ml/datasets/spambase, 20-08-2021.

char-freq-exclamation, char-freq-dollar, char-freq-sharp, capital-run-length-average, capital-run-length-longest, capital-run-length-total .

4. *dataset*: isolet[9].
   (a) variável resposta: 26 letters of the alphabet.
   (b) $n = 7797$ casos.
   (c) $p = 618$ variáveis explicativas: *150 subjects spoke the name of each letter of the alphabet twice, the features include spectral coefficients; contour features, sonorant features, pre-sonorant features, and post-sonorant features.* All variables are continuos, some cases are missing.

5. *dataset*: letter-recognition[10].
   (a) variável resposta: 26 letters of the alphabet.
   (b) $n = 20000$ casos.
   (c) $p = 16$ variáveis explicativas.
      i. x-box horizontal position of box.
      ii. y-box vertical position of box.
      iii. width width of box.
      iv. high height of box.
      v. onpix total # on pixels.
      vi. x-bar mean x of on pixels in box.
      vii. y-bar mean y of on pixels in box.
      viii. x2bar mean x variance.
      ix. y2bar mean y variance.
      x. xybar mean x y correlation.
      xi. x2ybr mean of x * x * y.
      xii. xy2br mean of x * y * y.
      xiii. x-ege mean edge count left to right.
      xiv. xegvy correlation of x-ege with y.
      xv. . y-ege mean edge count bottom to top.
      xvi. yegvx correlation of y-ege with x.

[9] https://www.kaggle.com/gorangsolanki/isolet-dataset, 20-08-2021. Créditos: Cole e Fanty (1994).
[10] https://archive.ics.uci.edu/ml/datasets/letter+recognition, 20-08-2021. Créditos: Dua e Graff (2017).

10.2.2 Resultados

A Tabela 10.2 apresenta os resultados. De maneira geral quando a acurácia é alta (0,90 ou acima), caso dos três primeiros (*datasets* bank-note, bank-chunners e spam), o ganho ao usarmos as árvores AdaBoost, *Random Forest*, Gradiente Boosting e XGBoost embora seja menor, ainda assim é um ganho. Considerando os *datasets* de reconhecimento de letras isolet e letter-recognition temos 26 letras como resposta e portanto, múltiplas classes para prever. *Random Forest* e XGBoost apresentaram os maiores ganhos em relação ao CART. Surpreendentemente o AdaBoost performou pior do que o CART (-36,1%), comportamento único entre todos os *datasets*.

As Figuras de 10.6 a 10.10 apresentam as respectivas acurácias junto com o desvio padrão dos resultados das 10 validações cruzadas.

Tabela 10.2: *Aplicação Árvores de Classificação - acurácia.*

dataset	CART	AdaBoost	Random Forecast	Gradiente Boosting	XGBoost
bank-note					
acurácia	0,9872	0,9985	0,9936	0,9930	0,9939
desv.pad.	0,0056	0,0015	0,0025	0,0030	0,0040
Δ% CART	0,0%	1,1%	0,6%	0,6%	0,7%
bank-chunners					
acurácia	0,9353	0,9703	0,9571	0,9646	0,9711
desv.pad.	0,0057	0,0038	0,0045	0,0044	0,0035
Δ% CART	0,0%	3,7%	2,3%	3,1%	3,8%
spam					
acurácia	0,9095	0,9486	0,9507	0,9428	0,9527
desv.pad.	0,0049	0,0050	0,0019	0,0049	0,0032
Δ% CART	0,0%	4,3%	4,5%	3,7%	4,7%
adult-income					
acurácia	0,8088	0,8634	0,8520	0,8614	0,8678
desv.pad.	0,0030	0,0031	0,0024	0,0024	0,0021
Δ% CART	0,0%	6,8%	5,3%	6,5%	7,3%
isolet					
acurácia	0,8039	0,8177	0,9413	0,9252	0,9457
desv.pad.	0,0088	0,0087	0,0042	0,0035	0,0039
Δ% CART	0,0%	1,7%	17,1%	15,1%	17,6%
letter-recognition					
acurácia	0,8721	0,5570	0,9616	0,9173	0,9602
desv.pad.	0,0050	0,0187	0,0022	0,0040	0,0022
Δ% CART	0,0%	-36,1%	10,3%	5,2%	10,1%

Figura 10.6: dataset *bank-note: acurácia - Média de 10 de amostras por validação cruzada.*

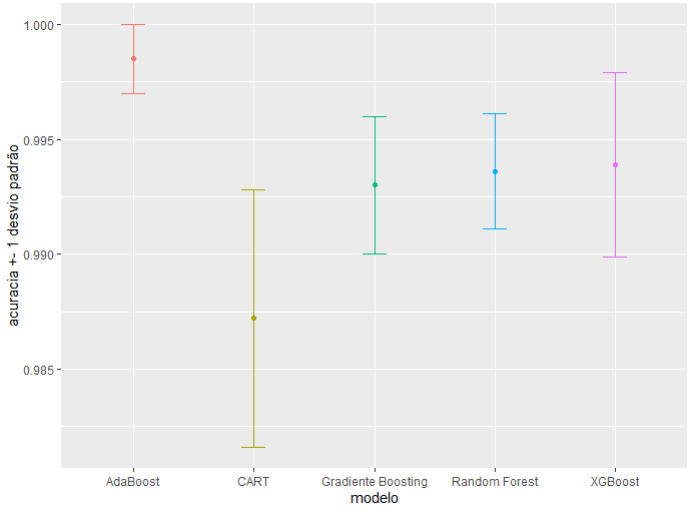


Figura 10.7: dataset *bank-chunners: acurácia - Média de 10 de amostras por validação cruzada.*

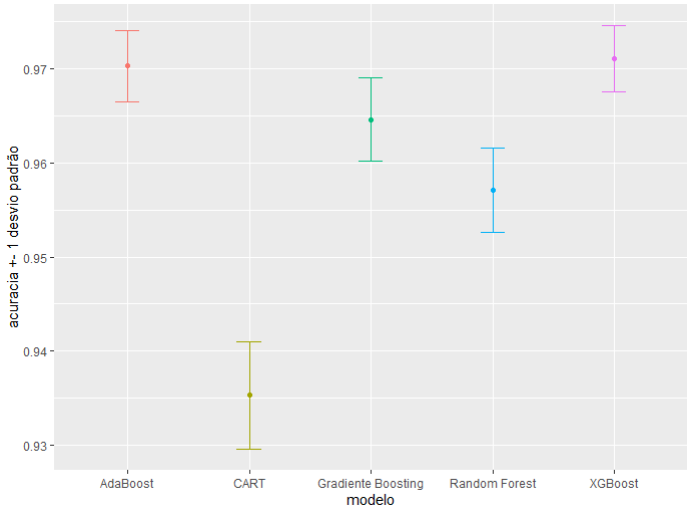


Figura 10.8: dataset *spam: acurácia - Média de 10 de amostras por validação cruzada.*

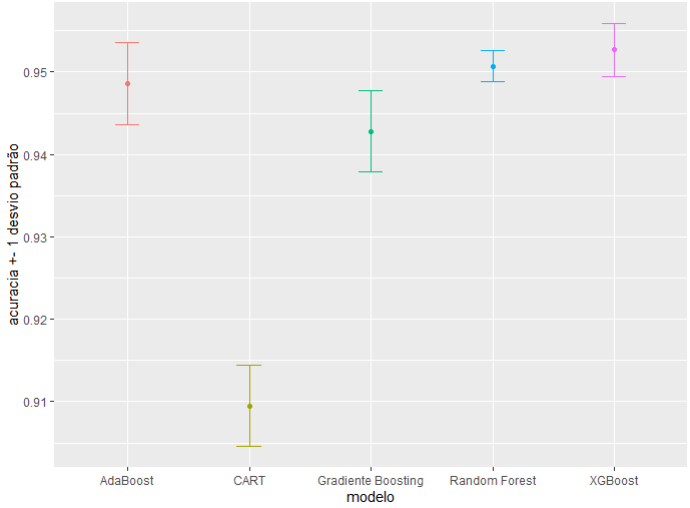


Figura 10.9: dataset *isolet: acurácia - Média de 10 de amostras por validação cruzada.*

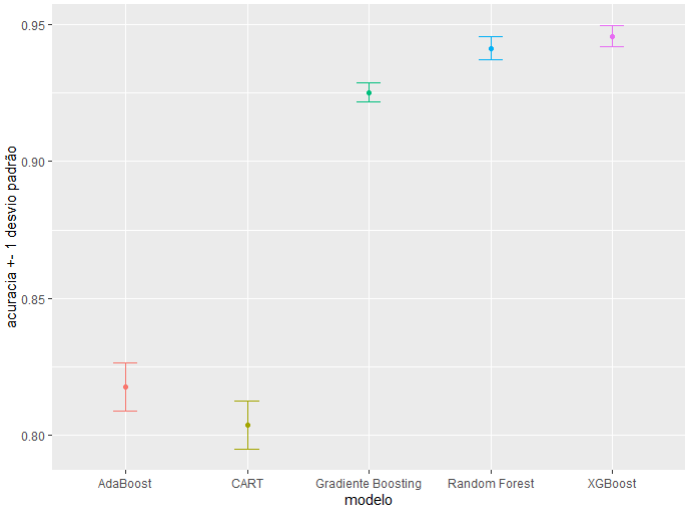
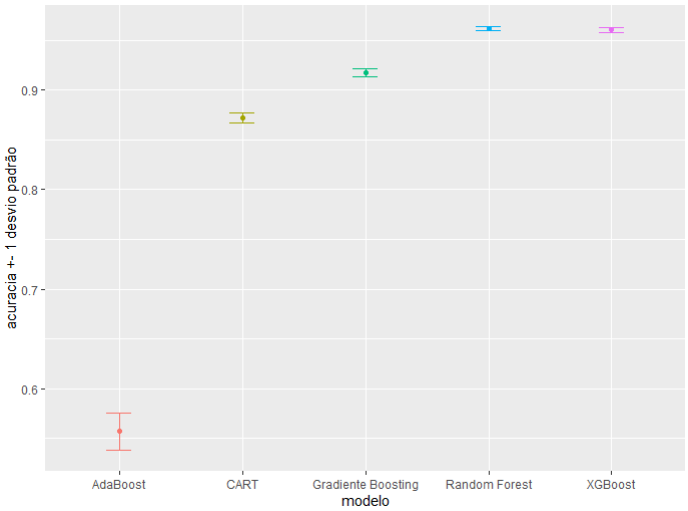


Figura 10.10: dataset *letter-recognition: acurácia - Média de 10 de amostras por validação cruzada.*



11 Conclusões

A partir de uma construção aparentemente simples e intuitiva, as árvores CART geraram uma pletora de novas abordagens bastante frutíferas. O modelo *random forest* apresentou um grande passo em obter expressivas reduções no erro em relação ao CART. Além de combinar a aleatorização da amostra de construção das árvores introduziu também a aleatorização na escolha da variável para ponto de corte em cada nó gerado em cada árvore. Esta abordagem mostrou que a aleatorização pode ser benéfica tanto na manipulação de amostras diferentes quanto nas escolhas de variáveis.

As técnicas de *gradient boosting* e XGBoost trouxeram para o universo das árvores o conceito de otimização já muito estudado em diversas aplicações no Cálculo. As árvores BART usando técnicas bayesianas surpreendem pelo desempenho, quase sempre competindo o Gradiente Boosting pelos melhores resultados entre as árvores de regressão. Esta técnica introduz aleatoriedade no tamanho das árvores, na escolha da variável e do ponto de corte em cada nó gerado, além de usar Cadeias de Markov para chegar ao resultado final. No entanto, isso não se passa sem um ganho significativo de complexidade na modelagem e tempo computacional.

Nas simulações das árvores de regressão o *grandient boosting* apresentou sistematicamente resultado melhor do que as Florestas Aleatórias. A variante XGBoost também apresentou um bom desempenho embora para a simulação *Friedman#2* as *random forests* apresentaram melhores resultados. As árvores BART, à medida em que o tamanho da amostra de treino aumenta, apresentaram o erro um pouco menor do que o *grandient boosting* na simulação *Friedman#1*. No geral o *gradiente boosting* seria a melhor escolha.

A simulações das árvores de classificação mostra que na medida que o tamanho da amostra de treino aumenta as árvores *gradient boosting*, *random forest* e XGBoost tendem a apresentar resultados semelhantes. No entanto, as *random forest* apresentam desempenho ligeiramente melhor, esta seria a melhor escolha entre esses modelos de árvores.

Nas aplicações com os *datasets* a conclusão geral é não utilizar o CART. As variantes *random forest* ou XGBoost, em geral apresentam os melhores resultados. A conclusão vale tanto para as árvores de regressão quanto para as árvores de classificação.

11.1 Considerações Finais

A facilidade de uso das árvores de decisão por meio de bibliotecas em linguagens de programação amplamente distribuídas[1] são um convite a sua ampla utilização. As árvores de decisão são capazes de enfrentar, com sucesso, dois problemas básicos da modelagem estatística em geral: 1) combinar variáveis categóricas e numéricas para 2) predizer um valor numérico (problemas de regressão) ou categórico (problemas de classificação) *sem a imposição de algumas suposições não realísticas sobre os dados.*

11.2 Sugestões para Pesquisas Futuras

O modelo de árvore bayesiano BART merece uma atenção mais apurada. Este modelo ainda não foi amplamente estudado, parcialmente devido a complexidade introduzida com o uso de Cadeias

[1] Além do Python, existem bibliotecas em R, https://www.r-project.org/, as quais não foram utilizadas nestes trabalho.

de Markov, além da pouca disponibilidade de bibliotecas/pacotes de software de alto desempenho e fácil parametrização.

O desafio é encontrar atalhos que maximizem o desempenho das cadeias de Markov sem comprometer o resultado final, cf. He *et al.* (2019). Outro ponto de interesse é pesquisar outras formas de construção da árvore além das quatro operações definidas originalmente (GROW, PRUNE, CHANGE e SWAP) que pudessem contribuir para gerar árvore mais rapidamente e/ou árvores precisas.

Atenção também pode ser dada a outras distribuições de probabilidade, no caso de árvores de regressão, por exemplo, além da distribuição normal para os valores observados.

Fica em aberto a possibilidade de comparar os modelos otimizando os parâmetros para obtenção de menores erros na amostra de teste, processo também conhecido como *tuning*. O uso de conjunto de dados reais, *i.e.*, não simulados pode gerar comparativos interessantes.

Appendices

A Tabelas de Resultados da Simulação

Nas tabelas dos resultados das árvores de regressão medimos o quanto cada modelo de árvore apresenta um desempenho melhor do que o modelo CART. Para as árvores de regressão utilizamos

$$\Delta\%\text{CART} = 100 \times \frac{mse(CART) - mse(A)}{mse(CART)}\%,$$

enquanto que para as árvores de classificação utilizamos

$$\Delta\%\text{CART} = 100 \times \frac{erro(CART) - erro(A)}{mse(CART)}\%,$$

em que A representa o modelo de árvore com o qual o CART é confrontado.

São 100 simulações para cada tamanho de amostra de treino. Em cada simulação 2000 amostras de teste são utilizadas para medir o desempenho: *mse* para as árvores de regressão e *erro de classificação* para as árvores de classficação. Todos os modelos de árvore foram ajustados na mesma amostra de treino e o *mse/erro de classificação* foram medidos na mesma amostra de teste, respectivamente.

A.1 Árvores de Regressão - *Friedman#1*

Tabela A.1: *Simulação Friedman#1.*

Amostra de Treino: N = 100					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	16,784	5,406	6,375	8,402	7,922
desv.pad.	2,258	0,660	0,700	0,880	0,895
mínimo	12,701	4,106	5,076	6,637	6,261
máximo	23,778	7,359	8,392	11,281	10,363
Δ% CART	0,0%	67,8%	62,0%	49,9%	52,8%

Amostra de Treino: N = 200					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	13,481	3,902	4,272	6,252	5,771
desv.pad.	1,333	0,231	0,379	0,518	0,514
mínimo	11,149	3,419	3,394	5,218	4,651
máximo	17,335	4,563	5,218	8,308	7,573
Δ% CART	0,0%	71,1%	68,3%	53,6%	57,2%

Amostra de Treino: N = 300

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	12,264	3,597	3,501	5,490	4,878
desv.pad.	0,968	0,213	0,268	0,349	0,312
mínimo	10,324	3,214	2,856	4,471	4,231
máximo	14,494	4,299	4,182	6,595	5,612
Δ% CART	0,0%	70,7%	71,5%	55,2%	60,2%

Amostra de Treino: N = 400

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	11,302	3,372	3,134	5,113	4,414
desv.pad.	0,736	0,216	0,227	0,314	0,283
mínimo	9,955	2,726	2,680	4,548	3,796
máximo	13,441	3,859	3,744	6,249	5,469
Δ% CART	0,0%	70,2%	72,3%	54,8%	60,9%

Amostra de Treino: N = 500

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	10,947	3,181	2,899	4,819	4,008
desv.pad.	0,753	0,236	0,195	0,284	0,261
mínimo	9,224	2,622	2,526	4,194	3,508
máximo	13,101	3,785	3,512	5,507	4,898
Δ% CART	0,0%	70,9%	73,5%	56,0%	63,4%

Amostra de Treino: N = 600

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	10,273	3,01	2,629	4,507	3,712
desv.pad.	0,718	0,237	0,176	0,245	0,21
mínimo	8,438	2,426	2,127	3,798	3,13
máximo	12,385	3,615	3,145	5,052	4,26
Δ% CART	0,0%	70,7%	74,4%	56,1%	63,9%

Amostra de Treino: N = 700

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	9,829	2,857	2,588	4,348	3,503
desv.pad.	0,565	0,242	0,155	0,228	0,206
mínimo	8,433	2,288	2,173	3,838	3,014
máximo	11,069	3,527	2,954	4,883	4,009
Δ% CART	0,0%	70,9%	73,7%	55,8%	64,4%

Amostra de Treino: N = 800

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	9,542	2,653	2,431	4,18	3,303
desv.pad.	0,585	0,23	0,15	0,209	0,188
mínimo	8,167	2,029	2,093	3,627	2,878
máximo	11,13	3,155	2,848	4,63	3,671
Δ% CART	0,0%	72,2%	74,5%	56,2%	65,4%

Amostra de Treino: N = 900

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	9,255	2,499	2,340	4,035	3,126
desv.pad.	0,503	0,227	0,147	0,215	0,161
mínimo	8,256	2,036	1,918	3,409	2,810
máximo	10,844	3,127	2,805	4,600	3,519
Δ% CART	0,0%	73,0%	74,7%	56,4%	66,2%

Amostra de Treino: N = 1000

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	9,048	2,347	2,288	3,895	2,995
desv.pad.	0,437	0,217	0,137	0,187	0,147
mínimo	7,771	1,960	1,942	3,353	2,673
máximo	10,583	3,076	2,673	4,407	3,335
Δ% CART	0,0%	74,1%	74,7%	57,0%	66,9%

Amostra de Treino: N = 1100

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	8,862	2,271	2,239	3,783	2,873
desv.pad.	0,447	0,216	0,141	0,166	0,165
mínimo	7,826	1,876	1,930	3,363	2,531
máximo	10,099	2,946	2,621	4,255	3,375
Δ% CART	0,0%	74,4%	74,7%	57,3%	67,6%

Amostra de Treino: N = 1200

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	8,604	2,194	2,19	3,696	2,824
desv.pad.	0,410	0,157	0,121	0,185	0,141
mínimo	7,602	1,832	1,887	3,253	2,451
máximo	9,692	2,51	2,542	4,214	3,181
Δ% CART	0,0%	74,5%	74,5%	57,0%	67,2%

Amostra de Treino: N = 1300

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	8,515	2,069	2,142	3,632	2,712
desv.pad.	0,492	0,138	0,124	0,170	0,135
mínimo	7,170	1,771	1,866	3,214	2,436
máximo	9,808	2,414	2,430	3,991	3,054
Δ% CART	0,0%	75,7%	74,8%	57,3%	68,2%

Amostra de Treino: N = 1400

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	8,293	2,041	2,104	3,597	2,668
desv.pad.	0,406	0,135	0,12	0,19	0,122
mínimo	7,399	1,765	1,816	3,143	2,42
máximo	9,666	2,429	2,4	4,136	2,913
Δ% CART	0,0%	75,4%	74,6%	56,6%	67,8%

Amostra de Treino: N = 1500

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	8,113	2,002	2,063	3,496	2,579
desv.pad.	0,412	0,141	0,114	0,166	0,115
mínimo	7,045	1,712	1,747	3,031	2,332
máximo	9,542	2,339	2,344	3,849	2,853
Δ% CART	0,0%	75,3%	74,6%	56,9%	68,2%

Amostra de Treino: N = 1600

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	8,057	1,936	2,063	3,445	2,520
desv.pad.	0,359	0,12	0,115	0,157	0,119
mínimo	7,266	1,603	1,829	3,012	2,241
máximo	9,012	2,243	2,477	3,908	2,809
Δ% CART	0,0%	76,0%	74,4%	57,2%	68,7%

Amostra de Treino: N = 1700

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	7,929	1,911	2,002	3,341	2,451
desv.pad.	0,344	0,102	0,112	0,148	0,11
mínimo	7,247	1,656	1,752	2,901	2,117
máximo	8,887	2,153	2,246	3,626	2,69
Δ% CART	0,0%	75,9%	74,8%	57,9%	69,1%

Amostra de Treino: N = 1800

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	7,844	1,875	1,985	3,312	2,404
desv.pad.	0,377	0,099	0,099	0,145	0,105
mínimo	6,88	1,678	1,744	3,003	2,123
máximo	8,743	2,165	2,237	3,683	2,655
Δ% CART	0,0%	76,1%	74,7%	57,8%	69,4%

Amostra de Treino: N = 1900

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	7,687	1,853	1,985	3,273	2,372
desv.pad.	0,329	0,1	0,112	0,138	0,111
mínimo	6,871	1,665	1,707	2,971	2,074
máximo	8,55	2,078	2,252	3,684	2,674
Δ% CART	0,0%	75,9%	74,2%	57,4%	69,1%

Amostra de Treino: N = 2000

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	7,618	1,821	1,952	3,213	2,317
desv.pad.	0,361	0,095	0,11	0,166	0,097
mínimo	6,649	1,532	1,708	2,813	2,114
máximo	8,39	2,084	2,268	3,672	2,568
Δ% CART	0,0%	76,1%	74,4%	57,8%	69,6%

A.2 Árvores de Regressão - *Friedman#2*

Tabela A.2: *Simulação Friedman#2.*

Amostra de Treino: N = 100					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	43731,4	42977,6	25339,5	23702,1	28824,7
desv.pad.	5739,3	3353,5	2457,4	2432,4	3251,8
mínimo	32300,6	36569,8	21194,2	19709,7	23380
máximo	57462,5	56510,8	34093,9	33369,6	38560,7
Δ% CART	0,0%	1,7%	42,1%	45,8%	34,1%

Amostra de Treino: N = 200					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	37850,5	38772,2	21685,7	20824,9	24790,4
desv.pad.	3027	2711,8	1215,9	1215	1598
mínimo	31321,5	30793,8	19008,7	17790,7	21561,4
máximo	44615,5	43535,3	24880,5	24488,6	29882,9
Δ% CART	0,0%	-2,4%	42,7%	45,0%	34,5%

Amostra de Treino: N = 300					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	36376,9	31956,9	20274,3	19898,4	23549,1
desv.pad.	2929,3	3423,3	960,4	1018,3	1381,3
mínimo	30303,2	25994,4	18310,3	17377,2	20657,2
máximo	42788	41942,5	22391,9	22799	27320,7
Δ% CART	0,0%	12,2%	44,3%	45,3%	35,3%

Amostra de Treino: N = 400					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	35374,3	27023,3	19315,3	19329,2	22755,9
desv.pad.	2146,9	1983,7	669,3	716,4	950,1
mínimo	31369	22757	18098,3	17755,9	20929,2
máximo	41663,1	31875,3	20881,3	20772,2	25558,3
Δ% CART	0,0%	23,6%	45,4%	45,4%	35,7%

Amostra de Treino: N = 500					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	34537	25302,6	18884,4	19122,7	22395,5
desv.pad.	1736	2235,2	791,7	806,4	1019,5
mínimo	29931,8	20410,5	16784,9	16844,8	20092,7
máximo	38672	31128,6	21495,2	21395,2	24802,4
Δ% CART	0,0%	26,7%	45,3%	44,6%	35,2%

Amostra de Treino: N = 600

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	34552,7	23526	18526,1	18966,6	22076,2
desv.pad.	1897,3	1382,5	686,6	734,4	874,3
mínimo	30181,5	20848,8	16453,7	16763,6	19787,3
máximo	39856,4	27288,1	19981	20602,5	23845,8
Δ% CART	0,0%	31,9%	46,4%	45,1%	36,1%

Amostra de Treino: N = 700

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	34346,6	22695	18150,5	18786,8	21780
desv.pad.	1811,2	1384	644,9	715,5	863,1
mínimo	30284,6	19214,4	16865,6	17072,5	19884
máximo	38781,2	27739,5	20089,5	20420,4	23876,8
Δ% CART	0,0%	33,9%	47,2%	45,3%	36,6%

Amostra de Treino: N = 800

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	34421,1	22137,6	18051,4	18778,9	21710,5
desv.pad.	1606,1	1153,1	617,3	738,8	836,3
mínimo	31077	19903,2	16715,1	17052,1	19548,1
máximo	39113,3	24688,8	19764,2	21478,5	24933
Δ% CART	0,0%	35,7%	47,6%	45,4%	36,9%

Amostra de Treino: N = 900

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33600,1	21666,5	17807,6	18595,8	21403,5
desv.pad.	1521,4	1171,3	621	677,6	775,5
mínimo	30355,7	19095,1	15928,4	16418	19309,8
máximo	37316,3	25590,7	19104,3	20348,2	23281,8
Δ% CART	0,0%	35,5%	47,0%	44,7%	36,3%

Amostra de Treino: N = 1000

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33533,5	21149,3	17621,9	18457,4	21107,7
desv.pad.	1400,8	1065,7	602,1	652,4	684,2
mínimo	30034,4	18712,8	16306	16966,9	19390,2
máximo	36474,6	24611,6	19143,7	20078,2	23044,3
Δ% CART	0,0%	36,9%	47,4%	45,0%	37,1%

Amostra de Treino: N = 1100

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33709,8	20751,5	17477,1	18380,5	21017,3
desv.pad.	1501,9	1106,1	596,1	663,3	747,2
mínimo	30258	18285,9	16100,5	16937,6	19329,2
máximo	37150,6	22855,1	19000,1	19882,4	22913
Δ% CART	0,0%	38,4%	48,2%	45,5%	37,7%

Amostra de Treino: N = 1200

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33873,4	20515,5	17512,7	18560,8	20963,3
desv.pad.	1607,9	962,3	601,8	661,1	812,2
mínimo	29490,8	18060,4	16181,6	16709,5	19089,6
máximo	38021,7	22560,1	19215,3	20378,8	24068
Δ% CART	0,0%	39,4%	48,3%	45,2%	38,1%

Amostra de Treino: N = 1300

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33705,7	20245,3	17355,1	18318,9	20733,9
desv.pad.	1380	915,5	570,1	624,7	696,2
mínimo	30477,5	18148,5	15868,4	16401,7	18429,2
máximo	37263,7	23023,5	18688,2	19672,1	22155,2
Δ% CART	0,0%	39,9%	48,5%	45,7%	38,5%

Amostra de Treino: N = 1400

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33376,1	19948,6	17243,5	18274,9	20539,7
desv.pad.	1343,7	831,2	560,6	622,1	689,6
mínimo	30813,6	18053,4	15906,1	16415,2	18496,3
máximo	36182,3	21874,8	18510,4	19543,2	22337,2
Δ% CART	0,0%	40,2%	48,3%	45,2%	38,5%

Amostra de Treino: N = 1500

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33412,6	19712,7	17164,7	18264,6	20428,4
desv.pad.	1255,5	710,3	537,5	596,6	713,5
mínimo	30529,1	17891,6	16050	16939,6	18104,1
máximo	36159,4	21584,8	18979,5	19907,8	22393
Δ% CART	0,0%	41,0%	48,6%	45,3%	38,9%

Amostra de Treino: N = 1600

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33207,4	19653,4	17137,4	18175	20354,4
desv.pad.	1317,9	719,2	580	586,8	700,3
mínimo	30237,4	17723,6	15790,3	16805,9	18535,8
máximo	36793,8	21540,9	18487	19609,4	22136,3
Δ% CART	0,0%	40,8%	48,4%	45,3%	38,7%

Amostra de Treino: N = 1700

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33352,8	19342,2	17029,9	18222,4	20299,4
desv.pad.	1379,5	752,8	557,6	643,3	752,5
mínimo	29963,2	17327,5	15292,8	16304,5	17591,1
máximo	36659,8	20817	18128,7	19536,8	21997,4
Δ% CART	0,0%	42,0%	48,9%	45,4%	39,1%

Amostra de Treino: N = 1800

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33309,1	19301,1	17033,6	18202,1	20156,5
desv.pad.	1310,9	686,9	564	550,6	629,5
mínimo	30168,7	17775,9	15819,6	16998,4	18760,4
máximo	37306,8	21128,6	18539,1	19935,2	21795,4
Δ% CART	0,0%	42,1%	48,9%	45,4%	39,5%

Amostra de Treino: N = 1900

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33285,8	19087,9	16872,4	18039,9	19936,9
desv.pad.	1293,3	854,3	610,9	652,8	720,4
mínimo	30022,9	16667	15293,9	16461,2	18392,5
máximo	36332,9	21807,2	18205,1	19569,7	22009,4
Δ% CART	0,0%	42,7%	49,3%	45,8%	40,1%

Amostra de Treino: N = 2000

	CART	BART	Gradient Boosting	Random Forest	XGBoost
média	33177,6	18994,9	16916,1	18086,1	19898,9
desv.pad.	1278,9	689,2	537,7	558,5	627,1
mínimo	30008,3	17575,3	15879,9	16640,1	18425,7
máximo	36193,3	20843,3	18452	19462	21737
Δ% CART	0,0%	42,7%	49,0%	45,5%	40,0%

A.3 Árvores de Regressão - *Friedman#3*

Tabela A.3: *Simulação Friedman#3.*

Amostra de Treino: N = 100					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,059	0,036	0,032	0,033	0,035
desvio	0,010	0,004	0,005	0,005	0,007
minimo	0,042	0,028	0,023	0,023	0,025
maximo	0,091	0,048	0,05	0,052	0,054
Δ% CART	0,0%	37,8%	45,3%	44,0%	40,3%

Amostra de Treino: N = 200					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,045	0,031	0,024	0,025	0,027
desvio	0,005	0,003	0,003	0,003	0,003
minimo	0,033	0,026	0,018	0,019	0,021
maximo	0,062	0,040	0,033	0,038	0,037
Δ% CART	0,0%	31,8%	47,6%	43,9%	39,9%

Amostra de Treino: N = 300					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,041	0,029	0,021	0,022	0,024
desvio	0,005	0,002	0,002	0,002	0,003
minimo	0,033	0,025	0,018	0,017	0,020
maximo	0,062	0,035	0,027	0,029	0,032
Δ% CART	0,0%	30,0%	50,0%	47,1%	41,4%

Amostra de Treino: N = 400					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,038	0,027	0,019	0,02	0,022
desvio	0,003	0,002	0,001	0,001	0,002
minimo	0,031	0,023	0,016	0,017	0,019
maximo	0,049	0,032	0,022	0,023	0,028
Δ% CART	0,0%	27,3%	50,0%	47,2%	41,5%

Amostra de Treino: N = 500					
	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,036	0,026	0,018	0,018	0,021
desvio	0,003	0,002	0,001	0,001	0,001
minimo	0,029	0,022	0,016	0,015	0,018
maximo	0,048	0,03	0,021	0,023	0,025
Δ% CART	0,0%	28,6%	51,1%	49,6%	43,0%

Amostra de Treino: N = 600

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,034	0,025	0,017	0,018	0,020
desvio	0,003	0,002	0,001	0,001	0,001
minimo	0,028	0,022	0,015	0,015	0,017
maximo	0,045	0,028	0,02	0,021	0,025
Δ% CART	0,0%	26,9%	50,4%	48,9%	42,4%

Amostra de Treino: N = 700

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,033	0,024	0,016	0,017	0,019
desvio	0,003	0,001	0,001	0,001	0,001
minimo	0,029	0,021	0,015	0,015	0,017
maximo	0,041	0,028	0,019	0,02	0,022
Δ% CART	0,0%	27,4%	50,8%	49,8%	42,9%

Amostra de Treino: N = 800

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,032	0,024	0,016	0,017	0,019
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,028	0,021	0,014	0,014	0,016
maximo	0,043	0,027	0,020	0,020	0,023
Δ% CART	0,0%	26,4%	50,5%	48,8%	42,2%

Amostra de Treino: N = 900

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,032	0,023	0,016	0,016	0,018
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,026	0,020	0,014	0,014	0,016
maximo	0,037	0,025	0,018	0,019	0,022
Δ% CART	0,0%	26,8%	50,3%	49,1%	42,5%

Amostra de Treino: N = 1000

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,031	0,023	0,015	0,016	0,018
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,027	0,020	0,014	0,014	0,016
maximo	0,038	0,026	0,018	0,018	0,020
Δ% CART	0,0%	27,0%	49,9%	49,1%	42,2%

Amostra de Treino: N = 1100

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,031	0,023	0,015	0,016	0,018
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,027	0,02	0,014	0,014	0,016
maximo	0,037	0,027	0,018	0,018	0,021
Δ% CART	0,0%	26,7%	50,3%	49,2%	42,7%

Amostra de Treino: N = 1200

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,03	0,022	0,015	0,015	0,017
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,027	0,020	0,014	0,013	0,015
maximo	0,035	0,026	0,017	0,018	0,019
Δ% CART	0,0%	26,3%	50,5%	49,5%	43,4%

Amostra de Treino: N = 1300

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,030	0,022	0,015	0,015	0,017
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,025	0,019	0,013	0,013	0,015
maximo	0,034	0,025	0,018	0,018	0,019
Δ% CART	0,0%	26,4%	50,0%	49,1%	42,9%

Amostra de Treino: N = 1400

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,030	0,021	0,015	0,015	0,017
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,027	0,019	0,013	0,013	0,015
maximo	0,035	0,025	0,017	0,017	0,019
Δ% CART	0,0%	28,6%	50,5%	49,8%	43,8%

Amostra de Treino: N = 1500

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,029	0,021	0,015	0,015	0,017
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,025	0,017	0,013	0,013	0,015
maximo	0,036	0,025	0,017	0,018	0,019
Δ% CART	0,0%	27,7%	50,3%	49,3%	43,4%

Amostra de Treino: N = 1600

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,029	0,021	0,015	0,015	0,016
desvio	0,002	0,001	0,001	0,001	0,001
minimo	0,025	0,018	0,013	0,013	0,015
maximo	0,034	0,025	0,017	0,017	0,018
Δ% CART	0,0%	27,9%	49,8%	48,9%	43,1%

Amostra de Treino: N = 1700

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,029	0,020	0,014	0,015	0,016
desvio	0,001	0,001	0,001	0,001	0,001
minimo	0,024	0,018	0,013	0,013	0,015
maximo	0,032	0,024	0,016	0,016	0,018
Δ% CART	0,0%	28,5%	49,9%	49,1%	43,3%

Amostra de Treino: N = 1800

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,028	0,020	0,014	0,014	0,016
desvio	0,001	0,001	0,001	0,001	0,001
minimo	0,025	0,017	0,013	0,013	0,015
maximo	0,032	0,024	0,016	0,016	0,019
Δ% CART	0,0%	28,5%	50,0%	49,2%	43,4%

Amostra de Treino: N = 1900

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,028	0,020	0,014	0,014	0,016
desvio	0,001	0,001	0,001	0,001	0,001
minimo	0,025	0,018	0,013	0,013	0,014
maximo	0,034	0,022	0,016	0,016	0,017
Δ% CART	0,0%	29,8%	49,9%	49,0%	43,7%

Amostra de Treino: N = 2000

	CART	BART	Gradient Boosting	Random Forest	XGBoost
media	0,028	0,020	0,014	0,014	0,016
desvio	0,001	0,001	0,001	0,001	0,001
minimo	0,025	0,017	0,013	0,013	0,014
maximo	0,035	0,022	0,016	0,017	0,018
Δ% CART	0,0%	29,5%	49,8%	49,0%	43,7%

A.4 Árvores de Classificação - *waveform*

Tabela A.4: *Simulação waveform.*

Amostra de Treino: N = 100					
	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,3204	0,2228	0,1925	0,2373	0,2110
desv.pad.	0,0219	0,0223	0,0117	0,0220	0,0168
mínimo	0,2875	0,1804	0,1705	0,1905	0,1785
máximo	0,3505	0,2560	0,2090	0,2675	0,2400
Δ% CART	0,0%	30,5%	39,9%	25,9%	34,1%

Amostra de Treino: N = 200					
	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,3031	0,1959	0,1725	0,1960	0,1886
desv.pad.	0,0195	0,0083	0,0127	0,0102	0,0129
mínimo	0,2725	0,1770	0,1555	0,1795	0,1720
máximo	0,3320	0,2050	0,1920	0,2155	0,2085
Δ% CART	0,0%	35,4%	43,1%	35,3%	37,8%

Amostra de Treino: N = 300					
	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2895	0,1823	0,1685	0,1854	0,1812
desv.pad.	0,0112	0,0131	0,0100	0,0112	0,013
mínimo	0,2725	0,1535	0,1475	0,1695	0,159
máximo	0,3045	0,1985	0,1845	0,202	0,202
Δ% CART	0,0%	30,5%	41,8%	36,0%	37,4%

Amostra de Treino: N = 400					
	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2977	0,1878	0,1661	0,1776	0,1731
desv.pad.	0,0139	0,0104	0,0103	0,0100	0,0065
mínimo	0,2730	0,1765	0,1520	0,1602	0,1615
máximo	0,3170	0,2065	0,1815	0,1945	0,1805
Δ% CART	0,0%	30,5%	44,2%	40,3%	41,9%

Amostra de Treino: N = 500					
	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2852	0,1841	0,1636	0,1725	0,1733
desv.pad.	0,0103	0,0086	0,0074	0,0090	0,0088
mínimo	0,2720	0,1735	0,1475	0,1635	0,1620
máximo	0,2995	0,1906	0,1725	0,1915	0,1870
Δ% CART	0,0%	30,5%	42,6%	39,5%	39,2%

Amostra de Treino: N = 600

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2715	0,1867	0,1636	0,1701	0,1714
desv.pad.	0,0160	0,0063	0,0063	0,0093	0,0072
mínimo	0,2505	0,1765	0,1565	0,1575	0,1590
máximo	0,2990	0,1945	0,1765	0,1808	0,1802
Δ% CART	0,0%	30,5%	39,7%	37,0%	36,9%

Amostra de Treino: N = 700

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2722	0,1781	0,1570	0,1673	0,1632
desv.pad.	0,0124	0,0077	0,0100	0,0066	0,0087
mínimo	0,2525	0,1630	0,1470	0,1595	0,1520
máximo	0,2905	0,1890	0,1780	0,1830	0,1800
Δ% CART	0,0%	30,5%	42,3%	38,5%	40,0%

Amostra de Treino: N = 800

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2768	0,1830	0,1600	0,1663	0,1673
desv.pad.	0,0105	0,0121	0,0084	0,01420	0,0106
mínimo	0,2508	0,1620	0,1445	0,1385	0,1515
máximo	0,2975	0,2010	0,1750	0,1830	0,1830
Δ% CART	0,0%	30,5%	42,2%	39,9%	39,6%

Amostra de Treino: N = 900

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2672	0,1815	0,1545	0,1609	0,1624
desv.pad.	0,0126	0,0136	0,0086	0,0092	0,0107
mínimo	0,2490	0,1610	0,1402	0,1480	0,1470
máximo	0,2905	0,2085	0,1690	0,1755	0,1795
Δ% CART	0,0%	30,5%	42,2%	39,8%	39,2%

Amostra de Treino: N = 1000

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,263	0,1861	0,1630	0,1663	0,1698
desv.pad.	0,0105	0,0066	0,0089	0,0074	0,0086
mínimo	0,2415	0,1750	0,1520	0,1570	0,1600
máximo	0,2790	0,1940	0,1815	0,1785	0,1825
Δ% CART	0,0%	30,5%	38,0%	36,8%	35,4%

Amostra de Treino: N = 1100

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2702	0,1818	0,1586	0,1628	0,1647
desv.pad.	0,0106	0,0105	0,0084	0,0073	0,0076
mínimo	0,2560	0,1660	0,1435	0,1515	0,1540
máximo	0,2870	0,1995	0,1725	0,1760	0,1775
Δ% CART	0,0%	30,5%	41,3%	39,7%	39,0%

Amostra de Treino: N = 1200

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2606	0,1765	0,153	0,1578	0,1600
desv.pad.	0,0106	0,0090	0,0059	0,0091	0,0070
mínimo	0,2455	0,1610	0,1440	0,1450	0,1420
máximo	0,2775	0,1960	0,1610	0,1685	0,1685
Δ% CART	0,0%	30,5%	41,3%	39,4%	38,6%

Amostra de Treino: N = 1300

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2591	0,179	0,1537	0,1589	0,1601
desv.pad.	0,0114	0,0106	0,0076	0,0107	0,0086
mínimo	0,2475	0,1690	0,1385	0,1490	0,1470
máximo	0,2830	0,2020	0,1685	0,1850	0,1750
Δ% CART	0,0%	30,5%	40,7%	38,7%	38,2%

Amostra de Treino: N = 1400

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2620	0,1795	0,1580	0,1596	0,1624
desv.pad.	0,0079	0,0068	0,0078	0,0058	0,0050
mínimo	0,2440	0,1660	0,1415	0,1465	0,1515
máximo	0,2730	0,1875	0,1680	0,1655	0,1695
Δ% CART	0,0%	30,5%	39,7%	39,1%	38,0%

Amostra de Treino: N = 1500

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2600	0,1754	0,1509	0,1523	0,1561
desv.pad.	0,0080	0,0097	0,0078	0,0100	0,0093
mínimo	0,2445	0,1595	0,1390	0,1365	0,1420
máximo	0,2745	0,1903	0,1675	0,1700	0,1715
Δ% CART	0,0%	30,5%	42,0%	41,4%	40,0%

Amostra de Treino: N = 1600

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2596	0,1801	0,1540	0,158	0,1593
desv.pad.	0,0089	0,0102	0,0084	0,0061	0,0059
mínimo	0,2520	0,1675	0,1355	0,1475	0,1500
máximo	0,2820	0,1970	0,1625	0,1700	0,1700
Δ% CART	0,0%	30,5%	40,7%	39,1%	38,6%

Amostra de Treino: N = 1700

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2566	0,1775	0,1520	0,1552	0,1585
desv.pad.	0,0135	0,0112	0,0111	0,0110	0,0089
mínimo	0,2335	0,1545	0,135	0,1395	0,1485
máximo	0,2785	0,1930	0,1695	0,178	0,1705
Δ% CART	0,0%	30,5%	40,8%	39,5%	38,2%

Amostra de Treino: N = 1800

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2528	0,1788	0,1539	0,1554	0,1574
desv.pad.	0,0135	0,0079	0,0098	0,0086	0,0074
mínimo	0,2335	0,1665	0,1315	0,1375	0,1440
máximo	0,2725	0,1915	0,1700	0,1670	0,1715
Δ% CART	0,0%	30,5%	39,1%	38,5%	37,7%

Amostra de Treino: N = 1900

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2543	0,1813	0,1491	0,1536	0,1572
desv.pad.	0,0149	0,0110	0,0099	0,0098	0,0084
mínimo	0,2310	0,1660	0,1335	0,1395	0,1455
máximo	0,2800	0,1965	0,1660	0,1700	0,1740
Δ% CART	0,0%	30,5%	41,4%	39,6%	38,2%

Amostra de Treino: N = 2000

	CART	AdaBoost SAMME	Random Forest	Gradient Boosting	XGBoost
média	0,2563	0,1770	0,1526	0,1542	0,1553
desv.pad.	0,0089	0,0075	0,0096	0,0115	0,0112
mínimo	0,2435	0,1655	0,1405	0,1350	0,1395
máximo	0,2609	0,1808	0,1705	0,1720	0,1755
Δ% CART	0,0%	30,5%	40,5%	39,8%	39,4%

B Código Python da Simulação

B.1 Árvores de Regressão

```
# Python 3.8.3
#
# bibliotecas e versões
#
# numpy         version 1.18.5
# scikit-learn  version 0.23.1
# xgboost       version 1.3.3
# bartpy        version 0.0.2

import numpy as np
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from bartpy.sklearnmodel import SklearnModel
from sklearn.metrics import mean_squared_error

#
# modelos de árvores simulados e amostras de treino
#
names  = ['CART', 'Random Forest', 'Gradient Boosting', 'XGBoosting', 'BART']
models = [tree.DecisionTreeRegressor(), RandomForestRegressor(),
          GradientBoostingRegressor(), XGBRegressor(), SklearnModel()]
train_sizes = np.arange(start = 100, stop = 2001, step = 100)   # [100, 200,
    300, ..., 2000]

#
# simula dados para simulação Friedman #1
#
def friedman1(n_samples, n_features=10):
  X = np.zeros((n_samples, n_features))
  X[:,0] = np.random.uniform(low = 0, high = 1, size = n_samples)
  X[:,1] = np.random.uniform(low = 0, high = 1, size = n_samples)
  X[:,2] = np.random.uniform(low = 0, high = 1, size = n_samples)
  X[:,3] = np.random.uniform(low = 0, high = 1, size = n_samples)
  for j in range(4, n_features):
  X[:,j] = np.random.uniform(low = 0, high = 1, size = n_samples)
  eps = np.random.normal(loc = 0, scale = 1, size = n_samples)
  y = 10 * np.sin(np.pi * X[:,0] * X[:,1]) + 20 * (X[:,2] - 0.5)**2 +
      10 * X[:,3] + 5 * X[:,4] + eps
  return X, y

#
# simula dados para simulação Friedman #2
#
def friedman2(n_samples):
  X = np.zeros((n_samples, 4))
  X[:,0] = np.random.uniform(low = 0, high = 100, size = n_samples)
  X[:,1] = np.random.uniform(low = 40*np.pi, high = 560*np.pi, size = n_samples)
  X[:,2] = np.random.uniform(low = 0, high = 1, size = n_samples)
```

```
  X[:,3] = np.random.uniform(low = 1, high = 11, size = n_samples)
  eps = np.random.normal(scale = 125, size = n_samples)
  y = (X[:,0]**2 + (X[:,1]*X[:,2] - (1/(X[:,1]*X[:,3])))**2)**(0.5) + eps
  return X, y

#
# simula dados para simulação Friedman #3
#
def friedman3(n_samples):
  X = np.zeros((n_samples, 4))
  X[:,0] = np.random.uniform(low = 0, high = 100, size = n_samples)
  X[:,1] = np.random.uniform(low = 40*np.pi, high = 560*np.pi, size = n_samples)
  X[:,2] = np.random.uniform(low = 0, high = 1, size = n_samples)
  X[:,3] = np.random.uniform(low = 1, high = 11, size = n_samples)
  eps = np.random.normal(scale = 0.105, size = n_samples)
  y = np.arctan((X[:,1]*X[:,2] - 1/(X[:,1]*X[:,3]))/X[:,0]) + eps
  return X, y

#
# simula as bases de treino e teste para cada Friedman #n
#
def gen_data(smp_type, train_size, test_size):
  if smp_type == 'friedman1':
    X_train, y_train = friedman1(n_samples = train_size)
    X_test,  y_test  = friedman1(n_samples = test_size)
  elif smp_type == 'friedman2':
    X_train, y_train = friedman2(n_samples = train_size)
    X_test,  y_test  = friedman2(n_samples = test_size)
  elif smp_type == 'friedman3':
    X_train, y_train = friedman3(n_samples = train_size)
    X_test,  y_test  = friedman3(n_samples = test_size)
  else:
    print('invalid type')
  return X_train, X_test, y_train, y_test

#
# treina um modelo e calcula o mse na base teste
#
def run_model(md, X_train, X_test, y_train, y_test):
  md.fit(X_train, y_train)
  y_pred = md.predict(X_test)
  mse = mean_squared_error(y_pred, y_test)
  return(mse)

#
# executa as simulações para todos os modelos de árvores
#  para um determinado tamanho de amostra de treino
# smp_type   = 'friedman1' ou 'friedman2' ou  'friedman3'
# nruns      = número de simulações
# train_size = tamanho da amostra de treino
# test_size  = tamanho da amostra de teste
#
# retorna uma matriz de mse's: [modelo, mse](5 x nruns)
#
def run_sample_type(smp_type, train_size, test_size = 2000, nruns = 100):
  mse = np.zeros((len(models), nruns))
  for n in range(nruns):
    X_train, X_test, y_train, y_test = gen_data(smp_type, train_size, test_size)
    for i, md in enumerate(models):
      mse[i, n] = run_model(md, X_train, X_test, y_train, y_test)
  return(mse)
```

```
#
# guarda os resultados da simulação de cada Friedman #n por tamanho de amostra
#
def run_friedman(n):
  smp_type = 'friedman' + str(n)
  for train_size in train_sizes:
    mse = run_sample_type(smp_type, train_size)
    file_name = './' + smp_type + '_' + str(train_size) + '.txt'
    np.savetxt(file_name, mse, delimiter = ';')

#
# execução
#
if __name__ == '__main__':
  run_friedman(1)
  run_friedman(2)
  run_friedman(3)
```

B.2 Árvores de Classificação

```
# Python 3.8.3
#
# bibliotecas e versões
#
# numpy         version 1.18.5
# scikit-learn  version 0.23.1
# xgboost       version 1.3.3

import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

#
# modelos de árvores simulados e amostras de treino
#
names  = ['CART', 'AdaBoost', 'Random Forest', 'Gradient Boosting', 'XGBoost']
models = [DecisionTreeClassifier(), AdaBoostClassifier(algorithm="SAMME"),
    RandomForestClassifier(), GradientBoostingClassifier(), XGBClassifier(eval_
    metric= 'mlogloss', use_label_encoder=False)]
train_sizes = np.arange(start = 100, stop = 2001, step = 100)   # [100, 200,
    300, ..., 2000]
#
#
def v1(j):
  return max(6-abs(j-11),0)

def v2(j):
  return v1(j-4)

def v3(j):
  return v1(j+4)

#
# simula dados das waves: 3 classes
# aproximadamente 1/3 para cada classe
#
```

```
def wave_form(n_samples, n_vars = 21):

  X = np.zeros((n_samples, n_vars))
  y = np.random.randint(low = 0, high = 3, size = n_samples)  # classe
  u = np.random.uniform(low = 0, high = 1, size = n_samples)

  for i in range(n_samples):
    eps = np.random.normal(loc = 0, scale = 1, size = n_vars)
    if y[i] == 1:
      X[i,:] = [u[i]*v1(j) + (1-u[i])*v2(j) for j in range(n_vars)] + eps
    elif y[i] == 2:
      X[i,:] = [u[i]*v1(j) + (1-u[i])*v3(j) for j in range(n_vars)] + eps
    else:
      X[i,:] = [u[i]*v2(j) + (1-u[i])*v3(j) for j in range(n_vars)] + eps

  return X, y

#
# simula as bases de treino e teste
#
def gen_data(train_size, test_size):

  X_train, y_train = wave_form(n_samples = train_size)
  X_test,  y_test  = wave_form(n_samples = test_size)

  return X_train, X_test, y_train, y_test

#
# treina um modelo e calcula a acurácia na base teste
#
def run_model(md, X_train, X_test, y_train, y_test):

  md.fit(X_train, y_train)
  y_pred = md.predict(X_test)
  accuracy = accuracy_score(y_pred, y_test)

  return(accuracy)

#
# executa 100 runs para cada tamanho de amostra
# calcula a acurácia para cada run
#
def run_class(train_size, test_size = 2000, nruns = 100):

  accuracy = np.zeros((len(models), nruns))
  for n in range(nruns):
    X_train, X_test, y_train, y_test = gen_data(train_size, test_size)
    for i, md in enumerate(models):
    accuracy[i, n] = run_model(md, X_train, X_test, y_train, y_test)

  return(accuracy)

#
# guarda a acurácia para cada modelo-tamanho de amostra
#
def run_waveform():

  for train_size in train_sizes:
    accuracy = run_class(train_size)
    file_name = './classification_' + str(train_size) + '.txt'
    np.savetxt(file_name, accuracy, delimiter = ';')


#
# execução
```

```
#
if __name__ == '__main__':
    run_waveform()
```

C Código Python Datasets

C.1 Árvores de Regressão

```
# Python 3.8.3
#
# bibliotecas e versões
#
# numpy         version 1.18.5
# scikit-learn  version 0.23.1
# xgboost       version 1.3.3
# bartpy        version 0.0.2

import numpy as np
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

#
# modelos de árvores de regressão utilizados para testar os datasets
#
names  = ['CART', 'Random Forest', 'Gradient Boosting', 'XGBoosting']
models = [tree.DecisionTreeRegressor(), RandomForestRegressor(),
  GradientBoostingRegressor(), XGBRegressor()]

#
# avalia os modelos por validação cruzada
#
def cross_validation(X, y):
  cv = ShuffleSplit(n_splits=10, test_size=0.25, random_state=314)
  for i, md in enumerate(models):
    scores = -1 * cross_val_score(md, X, y, cv=cv, scoring='neg_mean_squared_
        error')
    print('{:17}'.format(names[i]),
      ' - mean:', '{:7.2f}'.format(score_model),
      ' - std:', '{:7.2f}'.format(np.std(scores)),
      ' - delta:', '{:7.1f}'.format(100*(score_cart-score_model)/score_cart))

#
# separa variáveis explicativas da variável reposta
#
def split_data(df, target):
  X = df.loc[:, df.columns != target].to_numpy()
  y = df.loc[:, df.columns == target].to_numpy().ravel()
  return X, y

#
# binariza as variáveis categóricas
#
def fit_binary(df, lst_cols):
  for col in lst_cols:
```

```
        df[col] = df[col].apply(lambda x: str(x))
        categories = list(set(df[col]))
        if len(categories)==2:
          categories.pop()
        new_columns = [col + '_' + str(cat) for cat in categories]
        for i, new_col in enumerate(new_columns):
          df[new_col] = df[col].apply(lambda x: 1 if x==categories[i] else 0)
        df.drop(columns=[col], inplace=True)
      return df

    #
    # car-price audi
    #
    df = pd.read_csv('./audi.csv')
    df['price'] = df['price'] / 1000
    df = fit_binary(df, ['model', 'transmission', 'fuelType'])
    X, y = split_data(df,'price')
    cross_validation(X, y)

    #
    # car-price bmw
    #
    df = pd.read_csv('./bmw.csv')
    print(df.shape)
    df['price'] = df['price'] / 1000
    df = fit_binary(df, ['model', 'transmission', 'fuelType'])
    X, y = split_data(df,'price')
    cross_validation(X, y)

    #
    # insurance
    #
    df = pd.read_csv('./insurance.csv')
    df = fit_binary(df, ['sex', 'smoker', 'region'])
    df['charges'] = df['charges'] / 1000
    X, y = split_data(df,'charges')
    cross_validation(X, y)

    #
    # wine-quality
    #
    df = pd.read_csv('./winequality-red.csv')
    X, y = split_data(df,'charges')
    cross_validation(X, y)

    #
    # students performance
    #
    df = pd.read_csv('./students_performance.csv', sep=',')
    df['score'] = (df['math score'] + df['reading score'] + df['writing score'])/3
    df = fit_binary(df, ['gender', 'race/ethnicity', 'parental level of education'
        , 'lunch', 'test preparation course'])
    df.drop(columns=['math score', 'reading score', 'writing score'], inplace=True
        )
    X, y = split_data(df,'score')
    cross_validation(X, y)
```

C.2 Árvores de Classificação

```
    # Python 3.8.3
    #
    # bibliotecas e versões
```

```
#
# numpy         version 1.18.5
# scikit-learn  version 0.23.1
# xgboost       version 1.3.3

import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

#
# modelos de árvores de classificação utilizados para testar os datasets
#
names  = ['CART', 'AdaBoost', 'Random Forest', 'Gradient Boosting', '
    XGBoosting']
models = [DecisionTreeClassifier(), AdaBoostClassifier(
DecisionTreeClassifier(max_depth=2),
n_estimators=600,
learning_rate=1.5,
algorithm="SAMME"), RandomForestClassifier(), GradientBoostingClassifier(),
    XGBClassifier(eval_metric= 'mlogloss', use_label_encoder=False)]

#
# avalia os modelos por validação cruzada
#
def cross_validation(X, y):
  cv = ShuffleSplit(n_splits=10, test_size=0.25, random_state=314)
  for i, md in enumerate(models):
    scores = -1 * cross_val_score(md, X, y, cv=cv, scoring='accuracy')
  print('{:17}'.format(names[i]),
  ' - mean:', '{:7.2f}'.format(score_model),
  ' - std:', '{:7.2f}'.format(np.std(scores)),
  ' - delta:', '{:7.1f}'.format(100*(score_cart-score_model)/score_cart))

#
# separa variáveis explicativas da variável reposta
#
def split_data(df, target):
  X = df.loc[:, df.columns != target].to_numpy()
  y = df.loc[:, df.columns == target].to_numpy().ravel()
  return X, y

#
# bin ariza as variáveis categóricas
#
def fit_binary(df, lst_cols):
  for col in lst_cols:
    df[col] = df[col].apply(lambda x: str(x))
    categories = list(set(df[col]))
    if len(categories)==2:
      categories.pop()
    new_columns = [col + '_' + cat for cat in categories]
    for i, new_col in enumerate(new_columns):
      df[new_col] = df[col].apply(lambda x: 1 if x==categories[i] else 0)
    df.drop(columns=[col], inplace=True)
return df

#
# bank-note
#
df = pd.read_csv('./BankNoteAuthentication.csv', sep=',')
```

```
X, y = split_data(df,'class')
cross_validation(X, y)

#
# bank-chunners
#
df = pd.read_csv('./BankChurners.csv', sep=',')
df.drop(columns=['CLIENTNUM', 'Naive_Bayes_Classifier_Attrition_Flag_Card_
    Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_
    Inactive_12_mon_1',
'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_
    Dependent_count_Education_Level_Months_Inactive_12_mon_2'], inplace=True)
df['Attrition_Flag'] = df['Attrition_Flag'].apply(lambda x: 0 if x=='Existing
    Customer' else 1)
df = fit_binary(df, ['Gender', 'Marital_Status', 'Card_Category', 'Education_
    Level', 'Income_Category'])
X, y = split_data(df,'Attrition_Flag')
cross_validation(X, y)

#
# spam
#
df = pd.read_csv('./spambase.data.csv')
X, y = split_data(df,'spam')
cross_validation(X, y)

#
# isolet
#
df = pd.read_csv('./isolet.csv')
df.letter = df.letter.apply(lambda x:int(x)-1)
X, y = split_data(df,'letter')
cross_validation(X, y)

#
# letter recoginition
#
df = pd.read_csv('./letter-recognition.data.csv')
import string
alphabet = list(string.ascii_uppercase)
d = dict(zip(alphabet, np.arange(26)))
df['letter'] = df.letter.apply(lambda x: d[x])
X, y = split_data(df,'letter')
cross_validation(X, y)
```

Bibliografia

Albert e Chib(1993) Jim Albert e Siddhartha Chib. Bayesian analysis of binary and polychotomous response data. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 88:669–679. doi: 10.1080/01621459.1993.10476321. Citado na pág. 88

Amit *et al.***(1997)** Y. Amit, D. Geman e K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305. doi: 10.1109/34.632990. Citado na pág. 33

Biggs *et al.***(1991)** David Biggs, Barry De Ville e Ed Suen. A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18(1):49–62. doi: 10.1080/02664769100000005. URL https://doi.org/10.1080/02664769100000005. Citado na pág. 2

Breiman(1996a) L. Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24:2350–2383. Citado na pág. 25, 28

Breiman(1996b) L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140. Citado na pág. 2, 25, 26, 28, 92

Breiman(2002) L. Breiman. Manual on setting up, using, and understanding random forests v3.1. *Statistics Department University of California Berkeley, CA, USA*. Citado na pág. 33

Breiman *et al.***(1984)** L. Breiman, Jerome H. Friedman, Richard A. Olshen e Charles J. Stone. *Classification And Regreesion Trees*. Chapman & Hall/CRC. Citado na pág. ii, iii, 2, 4, 8, 9, 13, 16, 17, 25, 95

Breiman(2001) Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32. doi: 10.1023/A:1010933404324. Citado na pág. 2, 25, 33, 34

Chen e Guestrin(2016) Tianqi Chen e Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785. Citado na pág. 2, 65

Chipman *et al.***(2010)** Hugh Chipman, Edward George e Robert McCulloch. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4. doi: 10.1214/09-AOAS285. Citado na pág. 2, 84, 85

Chipman *et al.***(1998)** Hugh A. Chipman, Edward I. George e Robert E. McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948. doi: 10.1080/01621459.1998.10473750. URL https://doi.org/10.1080/01621459.1998.10473750. Citado na pág. 2, 77, 82

Cole e Fanty(1994) Ron Cole e Mark Fanty. ISOLET. UCI Machine Learning Repository, 1994. Citado na pág. 105

Cortez *et al.***(2009)** Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos e J. Reis. Wine quality. UCI Machine Learning Repository, 2009. Citado na pág.

Cutler e Zhao(2001) A. Cutler e G. Zhao. Pert - perfect random tree ensembles. *Computing Science and Statistics*, página 497. Citado na pág. 34

Dietterich e Kong(1995) T. Dietterich e E. B. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Relatório técnico, Oregon State University. Citado na pág. 33

Dua e Graff(2017) Dheeru Dua e Casey Graff. Uci machine learning repository. UCI Machine Learning Repository, 2017. Citado na pág. 105

Efron e Tibshirani(1993) B. Efron e R. Tibshirani. *An Introduction to then Bootstrap*. Monographs on Statistics and Applied Probability. Springer Science+Bussines Media. Citado na pág. 25

Freund e Schapire(1997) Y. Freund e R. Schapire. A decision-theoretic generalization of on-line learning algorithms and an application to boosting. *Journal of Computer and System Sciences*. Citado na pág. 2, 42, 47

Friedman(1997) J. Friedman. On bias, variance, 0/1?loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77. Citado na pág. 29

Friedman(2002) Jerome Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378. doi: 10.1016/S0167-9473(01)00065-2. Citado na pág. 59, 60

Friedman *et al.*(2000) Jerome Friedman, Robert Tibshirani e Trevor Hastie. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of Statistics*, páginas 337–407. Citado na pág. 42, 44, 45

Friedman(1991) Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1 – 67. doi: 10.1214/aos/1176347963. URL https://doi.org/10.1214/aos/1176347963. Citado na pág. 92

Friedman(2000) Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232. Citado na pág. 2, 51, 59

Gelman *et al.*(2014) Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari e Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 3rd ed. edição. Citado na pág. 81

Geurts(2010) P. Geurts. *Bias vs Variance Decomposition For Regression and Classification*. Springer, 2nd ed. edição. Citado na pág. 29

Geurts *et al.*(2006) P. Geurts, D. Ernst e L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42. Citado na pág. 34

Gilles(2014) Louppe Gilles. *Understanding Random Forests: from theory to practice*. Tese de Doutorado, Unversité de Liège, Belgium. Citado na pág. 8, 25, 26, 29, 35

Hastie *et al.*(2009) T. Hastie, R. Tibshirani e J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition edição. Citado na pág. 2, 8, 14, 42, 44, 54, 59, 60

Hastie e Tibshirani(2000) Trevor Hastie e Robert Tibshirani. Bayesian backfitting (with comments and a rejoinder by the authors). *Statist. Sci.*, 15(3):196–223. doi: 10.1214/ss/1009212815. URL https://doi.org/10.1214/ss/1009212815. Citado na pág. 86

He *et al.***(2019)** Jingyu He, Saar Yalov e P. Richard Hahn. Xbart: Accelerated bayesian additive regression trees. Em Kamalika Chaudhuri e Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, páginas 1130–1138. PMLR. URL http://proceedings.mlr.press/v89/he19a.html. Citado na pág. 110

Ho(1998) T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844. doi: 10.1109/34.709601. Citado na pág. 33

Hyafil e Rivest(1976) L. Hyafil e R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17. Citado na pág. 13

James *et al.***(2017)** G. James, D. Witten, T. Hastie e R. Tibshirani. *Introduction to Statistical Learning*. Springer. Citado na pág. 13

Johnson e Wichern(1988) R. A. Johnson e D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Inc., USA. ISBN 0130411469. Citado na pág. 1

Kapelner e Bleich(2016) Adam Kapelner e Justin Bleich. bartMachine: Machine learning with Bayesian additive regression trees. *Journal of Statistical Software*, 70(4):1–40. doi: 10.18637/jss.v070.i04. Citado na pág. 86, 88

Kass(1980) G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(2):119–127. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2986296. Citado na pág. 2

Kim e Loh(2001) H. Kim e W. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96:589 – 604. Citado na pág. 2

Kohavi(1995) Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence*, 14. Citado na pág. 9

Krogh e Vedelsby(1995) Anders Krogh e Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. Em *Advances in Neural Information Processing Systems*, páginas 231–238. MIT Press. Citado na pág. 30

Loh e Shih(1997) W. Loh e Yu-Shan Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840. Citado na pág. 2

Loh e Vanichsetakul(1988) W. Loh e N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83:715–725. Citado na pág. 2

Loh(2009) WieYin Loh. Improving the precision of classification trees. *The Annals of Applied Statistics*, 3(4):1710–1737. Citado na pág. 2

Malehi e Jahangiri(2019) Amal Saki Malehi e Mina Jahangiri. Classic and bayesian tree-based methods. doi: http://dx.doi.org/10.5772/intechopen.83380. Citado na pág. 2

Messenger e Mandell(1972) Robert Messenger e Lewis Mandell. A modal search technique for predictibe nominal scale multivariate analys. *Journal of the American Statistical Association*, 67 (340):768–772. ISSN 01621459. URL http://www.jstor.org/stable/2284634. Citado na pág. 2

Montgomery *et al.***(2012)** Douglas C. Montgomery, Elizabeth A. Peck e Geoffrey G. Vining. *Introduction to Linear Regression Analysis (5th ed.)*. Wiley & Sons. Citado na pág. 1

Morgan e Sonquist(1963) James N. Morgan e John A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434. ISSN 01621459. URL http://www.jstor.org/stable/2283276. Citado na pág. 2

Nocedal e Wright(2006) J. Nocedal e Stephen J. Wright. *Numerical Optimization*. Springer. Citado na pág. 51

Quinlan(1986) J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106. Citado na pág. 2

Quinlan(1993) J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. Citado na pág. 2

Sammut e Webb(2017) Claude Sammut e Geoffrey I. Webb. *Encyclopedia of Machine Learning and Data Mining*, página 333. Springer US. Citado na pág. 14

Schapire e Freund(2012) Robert E. Schapire e Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, Cambridge, Massachusetts. Citado na pág. 42

Valiant(1984) G. Leslie Valiant. A theory of the learnable. *Communications of The ACM*, páginas 1134–1142. Citado na pág. 42

Valiant(2013) Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books. Citado na pág. 42

Yalov(2019) Saar Yalov. A study of accelarated bayesian additive regression trees. Dissertação de Mestrado, Arizona State University, The address of the publisher. A Thesis Presented in Partial Fulfillment of the Requirements for the Degree Master of Science. Citado na pág. 2

Zhu *et al.***(2006)** Ji Zhu, Saharon Rosset, Hui Zou e Trevor Hastie. Multi-class adaboost. *Statistics and its interface*, 2. doi: 10.4310/SII.2009.v2.n3.a8. Citado na pág. 48, 95