PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM ALGORITMO EVOLUTIVO PARA INDUÇÃO DE ÁRVORES DE REGRESSÃO ROBUSTO A VALORES AUSENTES

LUCIANO COSTA BLOMBERG

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz

Porto Alegre
2014

Dados Internacionais de Catalogação na Publicação (CIP)

B653a Blomberg, Luciano Costa
Um algoritmo evolutivo para indução de árvores de regressão robusto a valores ausentes / Luciano Costa Blomberg. – Porto Alegre, 2014.
97 p.

Tese (Doutorado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz.

1. Informática. 2. Aprendizagem de Máquina. 3. Algoritmos (Programação). I. Ruiz, Duncan Dubugras Alcoba. II. Título.

CDD 006.35

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE TESE DE DOUTORADO

Tese intitulada "Um Algoritmo Evolutivo para Indução de Árvores de Regressão Robusto a Valores Ausentes", apresentada por Luciano Costa Blomberg, como parte dos requisitos para obtenção do grau de Doutor em Ciência da Computação, aprovada em 22/08/2014 pela Comissão Examinadora:

Prof. Dr. Duncan Dubugras Alcoba Ruiz — PPGCC/PUCRS
Orientador

Prof. Dr. Rodrigo Coelho Barros — PPGCC/PUCRS

Prof. Dr. Márcio Porto Basgalupp — UNIFESP

Prof. Dr. Andre Carlos Ponce de Leon Ferreira de Carvalho — USP

Homologada em 11/09/14, conforme Ata No. 17 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P. 32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

DEDICATÓRIA

Dedico este trabalho à minha esposa Anelise e aos nossos filhos Isabela e Rafael.

AGRADECIMENTOS

Depois de finalizada mais uma etapa de minha formação acadêmica, gostaria de registrar meus sinceros agradecimentos a todas as pessoas que contribuíram para tanto.

Primeiramente, gostaria de agradecer aos meus pais, Arão e Elza, e à minha irmã, Patrícia, pela educação e por todas as lições que contribuíram para formação de meus princípios morais e éticos.

Agradeço aos meus tios, tias, primos, primas e demais familiares queridos que estiveram sempre na torcida para que tudo desse certo. Agradeço também à família de minha esposa pelo carinho e incentivo constante.

Agradeço aos meus amigos, pelo apoio, compreensão e tolerância nos meus momentos de ausência.

Agradeço aos meus colegas e amigos do GPIN: Christian, Renata, Juliano e Daiane, em especial, e também aos mais novos: Sílvia, Holisson e Carlos. Todos vocês foram muito importantes nesta caminhada. Obrigado “messsmo”!

Não poderia deixar de agradecer também aos meus ex-colegas e amigos(as): Ana Winck, Nelson, Patrícia Hubler, Peterson e Tiago Silva.

Um muito obrigado aos amigos da Odontologia, professores José Figueiredo e Eduardo Mota, pela parceria nos projetos e por todo o incentivo.

Agradeço também ao meu orientador, Prof. Duncan, pela oportunidade dada, pela confiança em meu trabalho, e especialmente, por ter me ensinado tudo o que sei sobre pesquisa. Deixo parte desse mérito também ao Prof. Rodrigo Barros, ex-colega de mestrado e amigo, cuja colaboração foi inestimável dentro de minha pesquisa. Não tenho palavras para agradecer a ambos professores por suas respectivas contribuições e amizades.

Finalmente, gostaria de agradecer ao CNPq por ter financiado meus estudos.

UM ALGORITMO EVOLUTIVO PARA INDUÇÃO DE ÁRVORES DE REGRESSÃO ROBUSTO A VALORES AUSENTES

RESUMO

Uma preocupação comum em muitas áreas do conhecimento envolve problemas de baixa qualidade de dados, tais como ruídos e dados ausentes. Na área de aprendizado de máquina, por exemplo, dados ausentes têm gerado sérios problemas no processo de extração de conhecimento, ocultando importantes informações sobre o *dataset*, enviesando resultados e afetando o desempenho preditivo dos modelos induzidos. Para lidar com esse problema, muito tem se discutido na literatura sobre estratégias de tratamento, seja por pré-processamento ou por meio do desenvolvimento de algoritmos robustos a dados ausentes. Neste trabalho, propõe-se um novo algoritmo evolutivo para indução de árvores de regressão, agregando em seu ciclo evolutivo múltiplas estratégias para lidar com dados ausentes. Com o objetivo de fazer uma análise comparativa, foram avaliados 6 tradicionais algoritmos de regressão, considerando para tanto, 10 *datasets* artificialmente modificados para manterem diferentes níveis de dados ausentes. Resultados da análise experimental mostram que a solução proposta apresenta uma boa relação custo-benefício entre compreensibilidade dos modelos e desempenho preditivo, especialmente para as bases de dados com mais de 40% de dados ausentes.

Palavras-Chave: aprendizado de máquina, regressão, dados ausentes, imputação-múltipla.

AN EVOLUTIONARY ALGORITHM FOR REGRESSION TREE INDUCTION ROBUST TO MISSING VALUES

ABSTRACT

A common concern in many fields of knowledge involves problems of low quality data, such as noise and missing data. In the machine learning area, for example, missing data has generated serious problems in the knowledge extraction process, hiding important information about the dataset, skewing results and affecting the accuracy of the induced models. In order to deal with these problems, much has been discussed in the literature about missing values treatment strategies, either by preprocessing tasks or by the implementation of robust algorithms to missing data. In this thesis, we introduce a new evolutionary algorithm for induction of regression trees, including multiple strategies in its evolutionary cycle for dealing with missing data. Aiming to make a comparative analysis, we evaluated six traditional regression algorithms over 10 public datasets artificially modified to present different levels of missing data. Results from the experimental analysis show that the proposed solution presents a good trade-off between model interpretability and predictive performance, especially for datasets with more than 40% of missing data.

Keywords: machine learning, regression, missing data, multiple imputation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

AE – Algoritmo Evolutivo

AG – Algoritmo Genético

AltIvo – Algoritmo de Indução de Árvores de Regressão

AM – Aprendizado de Máquina

BMI – Bayesian Multiple Imputation

CSV – Comma-Separated Values

EM – Expectation Maximization

FCS – Fully Conditional Specifications

IM – Imputação Múltipla

k-NN – k-Nearest Neighbors

MAR – Missing at Random

MAE – Mean Absolute Error

MCAR – Missing Completely at Random

MCMC – Markov Chain Monte Carlo

MLP – Multilayer Perceptron

NMAR – Missing Not at Random

PG – Programação Genética

RMSE – Root Mean Square Error

UCI – University of California, Irvine

SDR – Standard Deviation Reduction

SVM – Support Vector Machine

SVR – Support Vector Regression

SUMÁRIO

1. INTRODUÇÃO

1.1 Caracterização do problema

Dados ausentes é um problema típico de baixa qualidade de dados, sendo muito comum em aplicações do mundo real. Na área da saúde, por exemplo, é usualmente produzido pelo mau funcionamento de máquinas ou pela recusa de pacientes em responder certas questões (ex: renda, idade, peso, etc). Esse tipo de ocorrência, no entanto, tem chamado a atenção nas mais diversas áreas do conhecimento, uma vez que a incidência de altos percentuais de dados ausentes é conhecidamente prejudicial ao processo de análise dos dados.

De acordo com Acuna e Rodrigues [AR04], taxas menores do que 1% de dados ausentes são consideradas triviais, e 1 a 5% gerenciáveis. No entanto, taxas entre 5% e 15% requerem métodos sofisticados para manipulá-los e mais do que 15% podem impactar severamente em qualquer tipo de interpretação. Essa preocupação tem tomado maiores dimensões desde que a maior parte dos métodos estatísticos tem sido projetada para assumir que os dados são completos.

Em um estudo recente na área de AM (Aprendizado de Máquina) envolvendo dados ausentes, Blomberg et al. [BHR13] reportou o uso de 20 *datasets* da UCI (University of California, Irvine) com o objetivo de analisar a influência de dados ausentes sob 6 tradicionais algoritmos de regressão extraídos do pacote de *software* WEKA[1]: M5P e RepTree (implementações para os métodos de árvore modelo e árvore de regressão, respectivamente), IBK (implementação para o método de *k*-Vizinhos mais Próximos), MultiLayerPerceptron (implementação para o método de Redes Neurais), SMOReg (implementação para o método de Máquina de Vetores de Suporte) e LinearRegression (Implementação para o tradicional método de Regressão Linear). Ao final deste estudo, a análise empírica revelou que o desempenho preditivo da regressão diminuiu após significante inserção de dados ausentes em todos os *datasets* testados.

Para lidar com este tipo de problema, grande parte dos algoritmos de AM tem baseado sua implementação a partir de uma única estratégia de tratamento. Algoritmos como M5P, SMOReg e LinearRegression, por exemplo, substituem globalmente todos valores ausentes pela média/moda do atributo. Esse tipo de método, no entanto, não se reflete necessariamente em uma melhor solução, uma vez que a média tende a subestimar a variância da amostra [LR02, AR04, SRP+11].

[1] WEKA (Waikato Environment for Knowledge Analysis) é uma popular ferramenta de aprendizado de máquina escrita em Java, desenvolvida na Universidade de Waikato, Nova Zelândia.

Por outro lado, a manipulação de dados ausentes durante o pré-processamento tem se mostrado uma possível alternativa ao tratamento interno, onde a escolha do método é normalmente realizada pelo usuário. À despeito dessa "facilidade", a descoberta do método ótimo é um processo tipicamente oneroso, quase sempre realizado em uma estratégia de tentativa e erro.

Independentemente da forma de tratamento escolhida, não há um consenso na literatura sobre um método universal para o problema de dados ausentes. Esse fato deve-se principalmente à variabilidade de características do *dataset* (ex: mecanismos de distribuição dos dados, correlação dos dados, percentual de dados ausentes, etc.), fazendo com que diferentes configurações de *datasets* exijam diferentes soluções.

1.2 Objetivos da Tese

Levando em consideração a relevância do problema aqui descrito, bem como o impacto gerado no processo de extração de conhecimento por algoritmos de AM, propõe-se neste trabalho um novo AE (Algoritmo Evolutivo) para a Indução de Árvores de Regressão (AltIvo), agregando em seu ciclo evolutivo múltiplas estratégias para lidar com dados ausentes. As estratégias implementadas são: *i)* eliminação de Atributos, *ii)* imputação pelo método *Majority*, iii) imputação pelo método *k-Nearest Neighbors* e *iv)* imputação pelo método *Hot-Deck*.

Como principal objetivo, deseja-se otimizar o desempenho preditivo do algoritmo, especialmente para bases de dados com altos percentuais de dados ausentes (conforme classificação realizada por Acuna e Rodrigues [AR04]). Adicionalmente, espera-se obter uma boa relação de custo-benefício entre o poder preditivo da solução e a compreensibilidade dos modelos gerados.

Em razão de fazer uma análise comparativa para a solução proposta, avaliou-se 6 tradicionais algoritmos de regressão, considerando para tanto, 10 *datasets* públicos, artificialmente modificados para manterem diferentes níveis de dados ausentes (de 1% a 70%). Para avaliar a significância estatística entre os resultados produzidos por todos algoritmos, aplicou-se os testes de Friedman [Fri37, Fri40] e Nemenyi [Nem63], conforme abordagem proposta por Demsar [Dem06].

Ao final, deseja-se saber se o desempenho preditivo do algoritmo AltIvo, que incorpora múltiplas estratégias para lidar com dados ausentes, é igual ao desempenho preditivo dos algoritmos que implementam apenas uma estratégia.

1.3 Organização do Volume

Para melhor compreensão, este trabalho segue organizado da seguinte maneira: nos Capítulos 2, 3 e 4 são apresentados a fundamentação teórica do trabalho, abordando os temas de dados ausentes, métodos de regressão em aprendizado de máquina e algoritmos evolutivos. No Capítulo 5, uma solução para indução de árvores de regressão em cenários de baixa completude dos dados é descrita em detalhes. Ao final do capítulo, os trabalhos relacionados ao tema de pesquisa são apresentados. No Capítulo 6, a análise experimental, realizada em razão de avaliar a qualidade da solução proposta é descrita. Para tanto, tem-se avaliado 6 tradicionais algoritmos de regressão, e aplicado-os sob 10 *datasets* públicos, artificialmente modificados para manterem diferentes níveis de dados ausentes (1% a 70%). Neste mesmo capítulo, também são descritos os resultados obtidos a partir de um estudo na área de saúde bucal. Finalmente, este documento é concluído com a apresentação das considerações finais e trabalhos futuros.

2. DADOS AUSENTES

Dados ausentes é um problema típico de baixa qualidade dos dados que tem sido amplamente discutido nas últimas décadas. Isso deve-se principalmente aos seus malefícios gerados no processo de análise dos dados.

Na área de AM, por exemplo, os efeitos produzidos por dados ausentes têm compreendido sérios problemas na extração de conhecimento, ocultando importantes informações sobre o *dataset*, enviesando resultados e afetando a acurária dos modelos induzidos [JW04, LLW05, NSZ07, ZWZ10].

Com relação à ocorrência de dados ausentes, Little e Rubin [LR87] apresentam uma classificação para o modo com que os dados são distribuídos no *dataset*:

- **MCAR (Missing Completely at Random):** Esta distribuição ocorre quando a probabilidade de uma instância ter um valor ausente para um atributo não depende de qualquer outro dado observado ou dado ausente;
- **MAR (Missing at Random):** Esta distribuição ocorre quando a probabilidade de uma instância ter um dado ausente para um atributo depende de valores conhecidos, mas não dos próprios valores ausentes;
- **NMAR (Not Missing at Random):** Esta distribuição ocorre quando a probabilidade de uma instância ter um valor ausente para um atributo depende de seus próprios valores.

Para melhor compreensão dos conceitos apresentados, considere os exemplos ilustrados nas Tabelas 2.1, 2.2, 2.3 e 2.4:

Tabela 2.1 – Exemplo de *dataset* sem dados ausentes.

Id	Gênero	Peso (Kg)
1	M	87
2	F	102
3	F	65
4	M	120
5	F	76
6	M	87

A Tabela 2.1 mostra um exemplo de *dataset* para os seguintes atributos: gênero e peso de 6 pacientes. Como é possível verificar, este caso não apresenta dado ausente. Contudo, alguém poderia simular algumas situações nas quais o *dataset* inicial pudesse apresentar diferentes distribuições para os dados ausentes. Para todas as situações ilustradas, utiliza-se o caractere “?” para representar a ausência de valor nas instâncias.

A Tabela 2.2 mostra que a ocorrência do dado ausente no atributo peso não apresenta nenhuma relação com os valores presentes nos demais atributos, e nem mesmo com os seus próprios valores, o que caracteriza, neste caso, a distribuição MCAR. Por outro lado, o relacionamento de dependência entre o dado ausente do atributo peso e o atributo gênero (mais precisamente para os casos onde o seu valor é “F”) é mostrado na Tabela 2.3. Esta situação representa a distribuição MAR.

Tabela 2.2 – Exemplo de *dataset* com distribuição MCAR.

Id	Gênero	Peso (Kg)
1	M	87
2	F	?
3	F	?
4	M	120
5	F	76
6	M	?

Tabela 2.3 – Exemplo de *dataset* com distribuição MAR.

Id	Gênero	Peso (Kg)
1	M	87
2	F	?
3	F	?
4	M	120
5	F	?
6	M	87

Finalmente, na Tabela 2.4 é possível verificar que a incidência de dado ausente para o atributo peso está associada aos seus próprios valores ausentes. Neste caso, alguém poderia supor que pacientes acima de 100Kg (Id2 e Id4) têm omitido seu próprio peso, independente de gênero. Este tipo de distribuição é conhecido como NMAR.

Tabela 2.4 – Exemplo de *dataset* com distribuição NMAR.

Id	Gênero	Peso (Kg)
1	M	87
2	F	?
3	F	65
4	M	?
5	F	76
6	M	87

Nas próximas seções, as distribuições MCAR, MAR e NMAR serão novamente discutidas, destacando-se suas respectivas influências no processo de escolha de uma melhor estratégia para lidar com dados ausentes.

2.1 Estratégias para lidar com Dados Ausentes

Do ponto de vista da literatura, é possível categorizar dois maiores tipos de estratégias para lidar com dados ausentes, os quais são apresentados a seguir.

2.1.1 Estratégias de Pré-processamento

- **Eliminação de instâncias e/ou atributos:** Em uma forma mais clássica, esta estratégia consiste em determinar a extensão de dados ausentes sob cada instância, e então remover as instâncias e/ou atributos com altos níveis de dados ausentes [BM03].

  Outras variações desta estratégia têm sido amplamente difundidas na literatura, como por exemplo, *ListWise Deletion* e *PairWise Deletion*. A primeira, também conhecida como *complete case analysis* [LR02], consiste em descartar as instâncias onde haja, pelo menos um valor ausente. A segunda, também referenciada como *avaliable case analysis* [LR02], considera somente as instâncias com dados completos sob os atributos selecionados para uma determinada análise. Isto significa que a amostra sendo analisada pode mudar dependendo dos atributos que são considerados para análise [Osb12].

  À despeito da simplicidade de implementação dos métodos baseados em eliminação de atributos, autores da área têm recomendado cautela para o seu uso, uma vez que a redução da amostra quase sempre resulta em perda de poder estatístico [TCS06, LCL12]. Adicionalmente, esse método deve ser aplicado apenas se os dados forem MCAR, pois dados ausentes que não são MCAR apresentam elementos randômicos que podem enviesar os resultados [BM03, LCL12, JMGL+10].

- **Estimação de Parâmetros:** Esta estratégia compreende a aplicação do método de máxima verossimilhança, onde são estimados os parâmetros de um modelo estatístico para os dados completos, e mais tarde utilizado para imputação por meio de amostragem [LGH12]. Em geral, dado um conjunto de dados e um modelo estatístico, o método da máxima verossimilhança calcula os valores dos diferentes parâmetros do modelo estatístico em razão de maximizar a semelhança dos dados observados. De acordo com Magnani [Mag04], EM (*Expectation Maximization*) é baseado sob a

maximização da função de verossimilhança, sendo capaz de estimar diretamente estatísticas sem imputar valores - mesmo que eles possam ser utilizados para este fim.

- **Imputação Única:** Esta estratégia consiste em preencher os dados faltantes com valores plausíveis. Uma das vantagens do uso dessa estratégia durante o pré-processamento é a possibilidade de escolha de um método que melhor se ajuste às características do *dataset*. Este tipo de decisão é geralmente inflexível nas implementações internas de tradicionais algoritmos de AM. Entre os métodos comumente utilizados, destacamos:

  - **Uso de constante global:** Este método consiste na substituição de todos os valores ausentes de um atributo por uma mesma constante, tal como "desconhecido" ou ∞ [HKP11]. Contudo, em alguns casos, esse valor pode ser confundido com um padrão, de modo que todos os atributos imputados tenham o mesmo valor em comum;
  - **Atribuição de todos os valores possíveis:** Neste método, os valores ausentes são substituídos por todos os valores possíveis para um dado atributo. De acordo com Liu Peng [Liu05], uma instância com valor ausente será substituída por um conjunto de novas instâncias. Esta característica torna este método computacionalmente inviável.
  - **Uso de uma medida de tendência central:** Este método é comumente aplicado por meio da substituição do valor ausente por uma média/mediana (atributos numéricos) ou pela moda (atributos categóricos). Embora seja de fácil aplicação, o método apresenta a desvantagem de subestimar a variância da amostra, uma vez que que todos os valores imputados são idênticos. Uma alternativa para minimizar este problema consiste na aplicação do método *Majority* [KBR84]. Neste caso, utiliza-se a informação do atributo alvo para criação de estratos para os quais é calculada a medida de tendência central. No entanto, sua aplicação pode não produzir bons resultados para os casos onde os dados sejam MAR ou NMAR.
  - **Uso de modelos preditivos:** Este método consiste em utilizar a informação presente no *dataset* para prever cada atributo com valor ausente. Assim, o atributo com valor ausente é usado como atributo alvo e os demais atributos como variáveis preditivas. Redes Neurais e Árvores de Regressão são exemplos típicos da aplicação de modelos preditivos para atributos categóricos e contínuos, respectivamente.

    Um importante argumento em favor desta abordagem é que, frequentemente, atributos têm relacionamentos (correlações) entre si [BM03]. De acordo com Acuna e Rodrigues [AR04], algumas desvantagens deste método são: *i)* os valores do modelo estimado são geralmente mais bem comportados do que os reais

valores deveriam ser, *ii)* se não existe relacionamento entre os atributos do dataset e o atributo com dado ausente (distribuição MCAR), então o modelo não será útil para estimar dados ausentes e, *iii)* alto custo computacional, desde que um grande número de modelos é necessário para predizer cada atributo com dado ausente.

– **k-NN (*k-Nearest Neighbor* ou k-Vizinhos mais próximos):** Neste método, busca-se encontrar uma ou mais instâncias mais similar àquela com dado ausente. Esta similaridade é usualmente obtida através do cálculo de uma medida, tal como a Distância Euclidiana (Equação 2.1).

$$\sqrt{\sum_{i=1}^{N}(P_i - Q_i)^2} \tag{2.1}$$

onde N é o número de dimensões e P_i e Q_i são, respectivamente, os atributos (componentes) de índice *i* de P e Q. No Algoritmo IBK, por exemplo, esta medida é normalizada para a padronização de valores apresentados em diferentes escalas (Equação 2.2).

$$\sqrt{\sum_{i=1}^{N}\frac{1}{max_i - min_i}(P_i - Q_i)^2} \tag{2.2}$$

Então, uma vez conhecidas as instâncias mais similares, o valor predito é obtido, utilizando-se a média dos *k* valores para atributos numéricos, e a moda para atributos categóricos. Entre as vantagens da imputação por k-NN, estão: *i)* não requer a criação de um modelo preditivo para cada atributo com dado ausente; na verdade, k-NN não cria modelos explícitos (tal como Árvores de Regressão ou Redes Neurais) já que o *dataset* é usado como um *"lazy model"* [BM03]; *ii)* ele pode facilmente tratar instâncias com múltiplos dados ausentes enquanto o método de modelo preditivo pode manipular somente um por vez; *iii)* ele pode prever atributos contínuos e categóricos.

– **Abordagem *Hot-Deck*:** Semelhantemente ao método de k-NN, a imputação por meio da abordagem Hot-Deck envolve a substituição dos valores ausentes em uma instância (receptora) por aqueles verificados na instância mais similar (doadora) [AL10]. No entanto, diferentemente do método de k-NN, a imputação por *Hot-Deck* considera apenas a instância mais similar (com *k=1*) dentro de um determinado grupo de instâncias no *dataset*. Existe também o método de imputação *Cold-Deck* onde a fonte de dados utilizada para estimar os valores é diferente daquela onde encontra-se a instância receptora [AR04]. Por via de regra, ambos métodos são compostos pelas seguintes etapas: *i)* os dados são particionados em grupos e *ii)* os dados ausentes são substituídos dentro do grupo.

À despeito de seu uso na prática, parece não haver um consenso sobre a teoria por trás da abordagem *Hot-Deck*. Em algumas versões, o doador é selecionado randomicamente a partir de um *pool* de potenciais doadores (*Random Hot-Deck*). Por outro lado, um *Hot-Deck* determinístico pode utilizar a instância mais similar para identificar o doador. Existe ainda uma versão onde o doador é identificado através da média dos valores das *k* instâncias mais similares (neste caso, *k>1*). Contudo, alguns autores não têm considerado este como um método *Hot-Deck*.

Entre as vantagens de sua aplicação, destaca-se: *i) Hot-Deck* não requer a criação de modelos preditivos para cada atributo com dado ausente [SSCL08]; *ii)* O valor substituído é influenciado apenas pelos casos mais similares, ao contrário da média, que é influenciada por todos os casos [JW04]; *iii) Hot-Deck* tende a apresentar menor custo computacional do que a aplicação tradicional do k-NN, uma vez que o espaço de busca é reduzido pelo particionamento em grupos.

- **IM (Imputação Múltipla):** Proposta por Rubin [Rub76], IM fornece uma estratégia útil para o tratamento de dados ausentes. Em vez de preencher um único valor para cada dado ausente, IM substitui cada valor ausente por um conjunto de valores plausíveis. Esses valores representam a incerteza sobre o valor correto a ser imputado, o que é tipicamente ignorado pela imputação única [ZWZ10]. O funcionamento da IM pode ser compreendido em três maiores passos, conforme ilustrado na Figura 2.1:

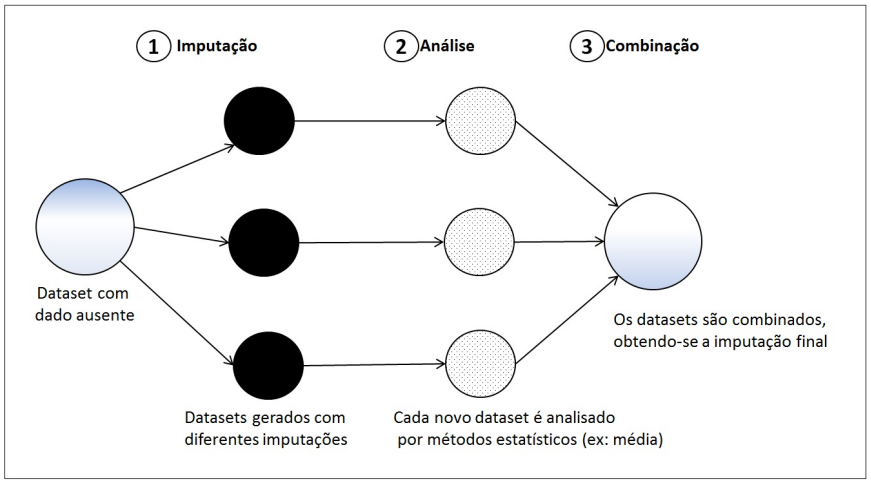


Figura 2.1 – Processo simplificado de imputação múltipla.

  - **Imputação:** O dado ausente é preenchido *N* vezes, gerando assim, N *datasets* completos;
  - **Análise:** Os *N datasets* gerados são analisados por métodos estatísticos para dados completos;
  - **Combinação:** A partir da análise dos *datasets*, os resultados são combinados para a inferência final.

IM utiliza-se de uma variedade de métodos, como por exemplo:

- **Hot-Deck Multiple Imputation:** Este método é uma variação da tradicional imputação *Hot-Deck* que é combinada com iterativas imputações e com o típico método de estimação de imputação múltipla paramétrica. Isto, por via de regra, significa que valores de vários "doadores" serão usados para um único "receptor".
- **BMI (Bayesian Multiple Imputation):** De acordo com Su [SKG08], BMI é um método de IM que usa o *framework* Bayesiano. Assim, ele especifica um modelo paramétrico para dados completos com uma prévia distribuição sob os parâmetros desconhecidos do modelo θ. Então, ele simula *n* maneiras independentes da distribuição condicional do dado ausente, considerando para tanto, os dados observados pelo teorema de Bayes. MCMC (Markov Chain Monte Carlo) e FCS (Fully Conditional Specifications) são dois tradicionais algoritmos baseados na abordagem Bayesiana.
- **Métodos de Regressão:** Neste método, um modelo de regressão é ajustado para cada variável com dado ausente. Baseado sob o resultado do modelo, um novo modelo de regressão é então elaborado e usado para imputar os dados ausentes [Rub87]. Uma vez que o *dataset* tem um padrão de dados monotônico, o processo é repetido sequencialmente para variáveis com dados ausentes. Regressão Linear e Regressão Logística são comumente aplicados para imputação de valores contínuos e categóricos, respetivamente.

2.1.2 Estratégias Implementadas Internamente por Algoritmos

Uma outra forma de lidar com o problema de dados ausentes é tratá-lo internamente dentro do algoritmo. Neste tipo de situação, utiliza-se usualmente um único método. O método implementado pelo IBK, por exemplo, consiste em calcular as distâncias entre as instâncias como se todos seus valores fossem conhecidos. Em geral, se o valor de um dado atributo *A* é ausente na tupla X_1 e/ou X_2, assumimos que cada um dos atributos tenha sido mapeado no intervalo entre [0,1]. Para atributos categóricos, atribuímos o valor 1 para esta diferença se um ou ambos os valores de *A* em X_1 ou X_2 estiverem ausentes. Se o atributo é numérico e o dado é ausente em ambas as tuplas, então esta diferença também é igual a 1. Se somente um dos valores está ausente e o outro esta presente e normalizado (no qual chamamos de *v'*), então nós podemos tomar a diferença para ser |1-v'| ou |0-v'|, qualquer que seja a maior [HKP11]. Assim, o algoritmo IBK calcula a distância entre exemplos como se todos seus valores fossem conhecidos.

O algoritmo MultiLayerPerceptron (MLP), por sua vez, à despeito de ser capaz de resolver problemas complexos e não lineares, não pode manipular dados ausentes direta-

mente [CS06]. Enquanto o algoritmo MLP (em sua versão no WEKA) apenas substitui os valores ausentes por zero, o SMOReg, M5P e LinearRegression substituem globalmente todos valores ausentes pela média ou pela moda do atributo através do método *ReplaceMissingValues* (do pacote de *software* WEKA).

No RepTree, os dados ausentes são tratados por meio do particionamento das instâncias correspondentes em pedaços [WF11]. Assim, uma instância com valor ausente no atributo testado é particionada em múltiplas pseudoinstâncias, cada uma com diferentes valores para a característica ausente e um peso correspondente para a probabilidade estimada em um particular valor ausente (baseado sob a frequência de valores neste particionamento dentro do conjunto de treino). Esta é a mesma estratégia implementada pelo C4.5 [Qui93].

2.2 Considerações do Capítulo

O tópico de qualidade dos dados tem recebido considerável atenção nas últimas décadas. Dados ausentes, por exemplo, é um típico problema de qualidade dos dados, comumente produzido em aplicações do mundo real. A discussão em torno de dados ausentes, em particular, tem sido consequência de seus malefícios no processo de análise dos dados. Para trazer maior esclarecimento sobre o assunto, o presente Capítulo apresentou dois tipos de estratégias para o tratamento de dados ausentes: *i)* Estratégias de pré-processamento e *ii)* Estratégias internamente implementadas nos algoritmos.

Independente da estratégia utilizada, não há um consenso na literatura sobre um método universal para o tratamento de dados ausentes, o que tem motivado a busca de novas soluções, como por exemplo, a combinação de múltiplas estratégias.

3. MÉTODOS DE REGRESSÃO EM APRENDIZADO DE MÁQUINA

Existem dois tipos de tarefas em modelagem preditiva: classificação, na qual a variável dependente é discreta, e regressão, para os casos onde a variável dependente é contínua. A maneira clássica de lidar com predição contínua é escrever o resultado como uma soma linear dos valores das variáveis independentes com pesos apropriados, tais como:

$$Y_i = \alpha + \beta X_I + \epsilon_i \tag{3.1}$$

onde,

Y_i é a variável dependente;

α é uma constante, que representa a interceptação da reta com o eixo vertical [1];

β é outra constante, que representa o declive (coeficiente angular) da reta;

X_i é a variável independente e;

ϵ_i inclui todos os fatores residuais mais os possíveis erros de medição.

Este tipo de representação é chamado de equação de regressão, e o processo de determinar os pesos, de regressão. Contudo, o método básico de regressão é incapaz de descobrir relacionamentos não-lineares, e assim, diferentes representações podem ser usadas para predição de quantidades numéricas [WF11].

Na área de AM, por exemplo, existe uma variedade de métodos que são projetados para a manipulação eficiente de grandes quantidades de dados, possivelmente multidimensionais e não lineares. Alguns destes métodos foram originalmente propostos para problemas de classificação e posteriormente adaptados para regressão, tais como: Redes Neurais, Máquinas de Vetores de Suporte e k-Vizinhos mais Próximos. Até mesmo as Árvores de Decisão foram modificadas de forma a tratar problemas de regressão. Árvores de Regressão e Árvores Modelo são dois exemplos típicos desta adaptação. Neste capítulo são apresentados 5 tradicionais métodos de AM utilizados para predição de valores contínuos.

3.1 Árvores de Regressão

Uma Árvore de Regressão é uma estrutura hierárquica de nodos e arestas comumente utilizada para solução de problemas onde o atributo alvo é contínuo. Seu princípio de crescimento é baseado em uma estratégia gulosa, *top-down* e com particionamento recursivo,

[1] Assumindo-se uma representação 2D.

sendo localmente ótima em suas partições sob os nodos. Assim, a partir do nodo raiz, testes lógicos são realizados sob determinados atributos (nodos não terminais), particionando a árvore em ramos com novos nodos. Este processo é então repetido, até que um nodo terminal (nodo folha ou variável dependente) com o valor da predição seja alcançado.

Enquanto estatísticos usam o termo regressão para o processo de cálculo de uma expressão que prediz uma quantidade contínua, Árvores de Regressão diferenciam-se por guardar em seus nodos folhas as médias dos valores presentes nestes nodos. No exemplo extraído de Tan et al. [PNSK05] foi adaptada uma árvore de regressão (Figura 3.1) para predição do atributo alvo idade (*age*).

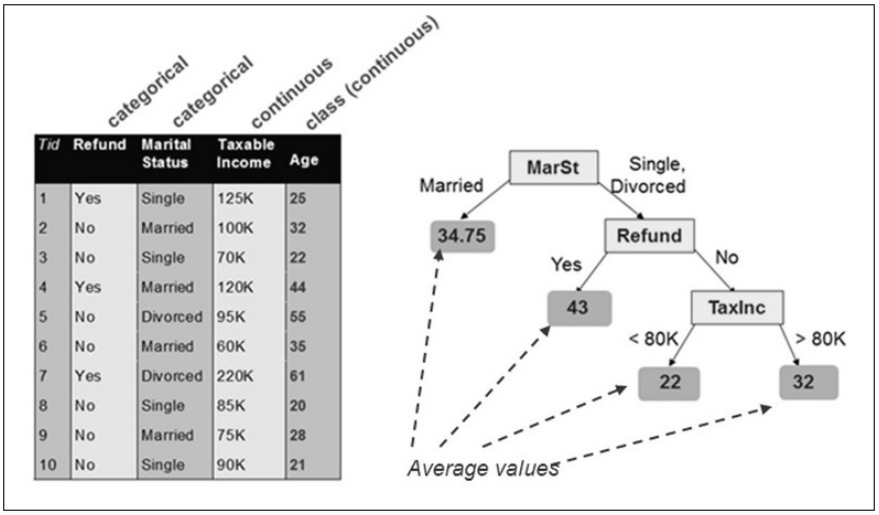


Figura 3.1 – Exemplo de Árvores de Regressão.

De acordo com Witten et al. [WF11], a acurácia do método baseado em árvores é competitivo com o tradicional método de Regressão Linear, com o potencial de ser mais preciso sob problemas não lineares.

3.2 Árvores Modelo

Assim como Árvores de Regressão, Árvores Modelo são estruturas hierárquicas para predição de variáveis dependentes contínuas. Contudo, diferentemente do primeiro método, Árvores Modelo são caracterizadas por guardar planos de regressão linear em seus nodos folhas. De acordo com Faceli et al. [FLGC11], a estrutura da árvore divide o espaço dos atributos em subespaços, e os exemplos em cada um dos subespaços são aproximados por uma função linear.

O critério de escolha destes atributos é usualmente implementado em razão de minimizar a variação intra conjunto nos valores da classe abaixo de cada ramo [WF11]. No

algoritmo M5P, por exemplo, este processo é realizado por meio do cálculo da redução do desvio padrão (Equação 3.2) sob uma porção D de instâncias do conjunto de treino.

$$SDR = sd(D) - \sum_{i=1}^{N} \frac{D_i}{D} \, x \, sd(D_i) \tag{3.2}$$

onde sd(.) é o cálculo do desvio padrão, D é a porção do conjunto de treino que atinge o nodo que está sendo testado e D_i é a porção do conjunto de treino que resulta do particionamento no nodo.

O processo de particionamento termina quando os valores das classes das instâncias que atingiram o nodo variam muito ligeiramente, tal que, quando seu desvio padrão é somente uma fração pequena (usualmente menos do que 5%) do desvio padrão do conjunto original da instância[WF11].

Em um segundo exemplo extraído de Tan et al. [PNSK05], é ilustrada a representação correspondente para o problema de predição de idade, utilizando para tanto, uma Árvore Modelo (Figura 3.2).

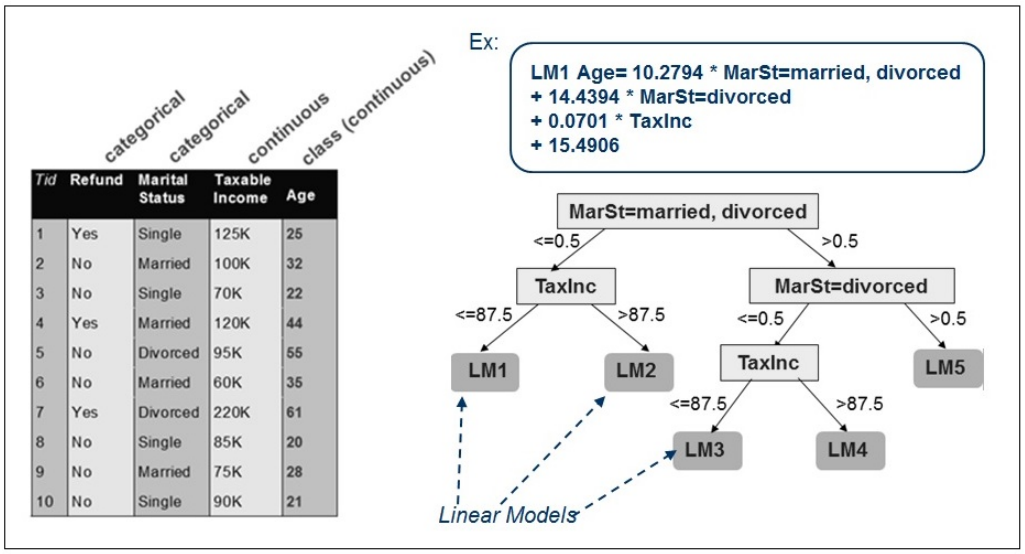


Figura 3.2 – Exemplo de Árvore Modelo.

3.3 Máquina de Vetores de Suporte

SVM (*Support Vector Machine*) é um método supervisionado de aprendizado de máquina usado para tarefas de classificação e regressão de dados lineares e não-lineares. No método SVM para regressão, também conhecido como SVR (*Support Vector Regression*), a ideia básica é encontrar uma função que aproxima bem os pontos de treinamento pela

minimização dos erros de predição. A diferença crucial é que todos os desvios acima de um parâmetro definido pelo usuário são simplesmente descartados.

De acordo com Witten et al. [WF11], um parâmetro ε definido pelo usuário define um tubo em torno da função de regressão no qual o erro é ignorado (para regressão de vetores de suportes lineares, o tubo é cilíndrico). Se todos os pontos de treino podem estar dentro de um tubo de largura 2ε, o algoritmo produz a função no meio do tubo mais plano que os abrange. Neste caso, o erro total percebido é zero. A Figura 3.3, mostra um problema de regressão com um atributo, uma classe contínua, e oito instâncias. Neste caso, ε é atribuído como 1 de modo que a largura do tubo em torno da função de regressão (indicada por linhas tracejadas) é 2.

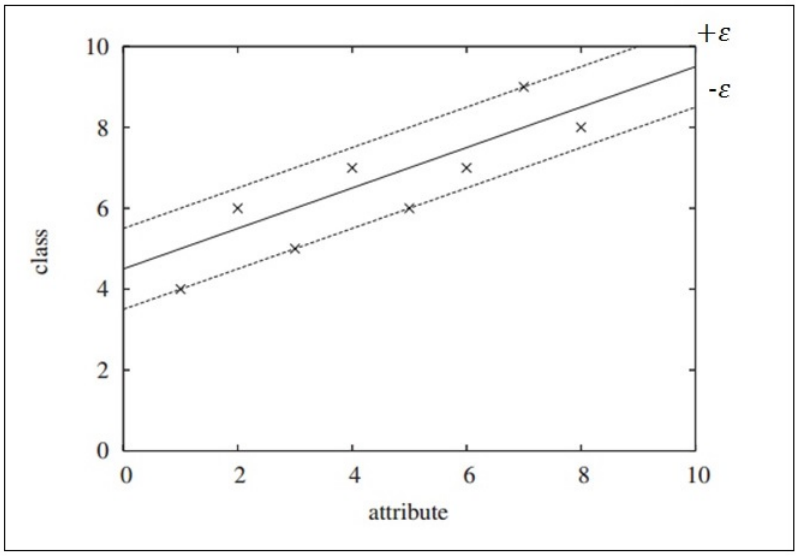


Figura 3.3 – Ilustração simplificada do procedimento realizado por SVR [WF11].

O valor de ε controla o quão próximo a função estará no conjunto de treino. Um valor muito grande produzirá um preditor sem sentido - dentro de um caso extremo, quando 2ε excede o intervalo de valores da classe dentro dos dados de treino, a linha de regressão é horizontal e o algoritmo apenas irá prever a média do valor da classe. O SMOReg implementa o algoritmo de otimização mínima sequencial para problemas de regressão [SS04].

3.4 k-Vizinhos mais Próximos

O método de k-Vizinhos mais Próximos consiste em encontrar os *k* exemplos de treino mais similares aos atributos do exemplo de teste. Neste método, cada tupla é classificada como um ponto de dado em um espaço n-dimensional, sendo *n* o número de atributos. Assim, dado um exemplo de teste, uma medida de similaridade é usada para calcular sua proximidade com outros pontos de dados do conjunto de treino. De acordo com Han et

al. [HKP11], proximidade é definida em termos de uma métrica de distância, tal como a distância Euclidiana (também definida na Seção 2.1, Equação 2.1).

Desde o momento que a lista de vizinhos mais próximos é obtida, o exemplo de teste é predito a partir do cálculo da média do atributo alvo nos *k* vizinhos. Na figura 3.4, é ilustrado um exemplo de aplicação do método para *k=1* e *k=3*.

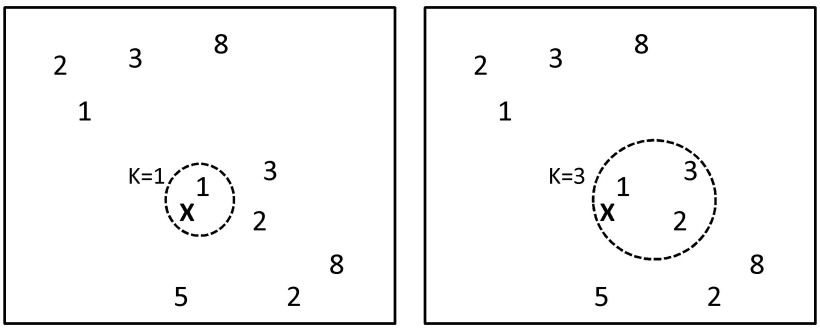


Figura 3.4 – Exemplo do método k-NN para k=1 e k=3.

3.5 Redes Neurais

Redes Neurais é um método de AM utilizado para tarefas de classificação e regressão. Seu princípio de funcionamento é baseado em uma tentativa de simular o sistema neural biológico humano. De acordo com Kuhn e Johnson (2013) [KJ13], o resultado de uma rede neural é modelado por um conjunto intermediário de variáveis não observadas (chamadas de variáveis ocultas ou camadas ocultas). Essas camadas ocultas, por sua vez, são combinações lineares de algumas ou de todas as variáveis preditoras.

Um modelo de rede neural usualmente envolve múltiplas camadas ocultas para modelar o resultado. A rede Perceptron Multicamadas, por exemplo, apresenta uma ou mais camadas intermediárias de neurônios e uma camada de saída. A arquitetura mais comum para uma rede Perceptron Multicamadas é a completamente conectada, de forma que os neurônios de uma camada l estão conectados a todos os neurônios da camada l + 1. Caso a camada l seja a primeira camada intermediária, cada um de seus neurônios estará conectado a todos os atributos de entrada x^j, j = 1..., d, para um objeto de entrada X. A Figura 3.5 ilustra uma típica rede MLP [FLGC11].

Embora seja comumente aplicada para problemas de classificação, deve-se observar que a utilização de uma rede Perceptron Multicamadas também pode ser diretamente estendida a problemas de regressão, mas neste caso não se tem a discretização imposta pela escolha do neurônio com maior saída na predição. Para melhor detalhamento sobre o método, sugere-se as seguintes literaturas: [KJ13, FLGC11].

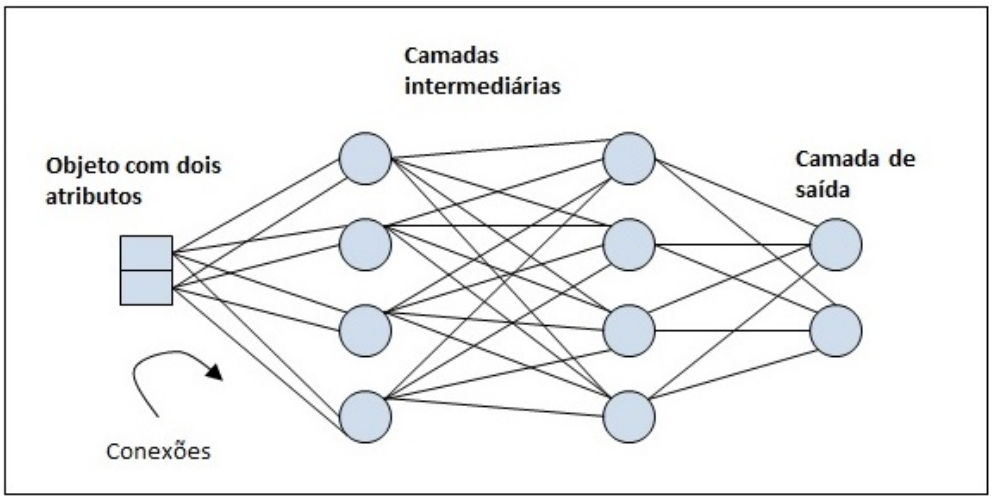


Figura 3.5 – Exemplo de Redes Neurais Multicamadas típica.

3.6 Considerações do Capítulo

Nesse capítulo foi apresentada uma visão geral sobre os métodos de SVM, k-NN, Redes Neurais, Árvores Modelo e Árvores de Regressão. Esses métodos são comumente aplicados para solução de problemas de regressão, e sua aplicação tem mostrado uma série de vantagens em relação às abordagens clássicas estatísticas [WF11]. Entre as principais vantagens, destacam-se:

- **Escalabilidade:** À despeito dos avanços na geração de grandes conjuntos de dados, algoritmos de aprendizado têm se mostrado capazes de manipular *datasets* massivos. De acordo com Witten e Frank [WF11], muitos algoritmos de AM (utilizados para aplicações de mineração de dados), empregam estratégias de busca especiais para manipulação de problemas de busca exponencial.
- **Alta dimensionalidade:** Atualmente, tem sido cada vez mais comum encontrar *datasets* com centenas ou milhares de atributos. Este tipo de característica é considerado problemático para tradicionais técnicas de análise que são desenvolvidas para trabalhar com dados de baixa dimensionalidade.
- **Dados heterogêneos e complexos:** Tradicionais métodos de análise frequentemente lidam com *datasets* contendo atributos do mesmo tipo, sejam eles contínuos ou categóricos. No entanto, percebe-se um crescimento na produção de dados complexos nos últimos anos, como por exemplo, dados semi-estruturados ou dados espaciais. Desde então, tem sido desejável o desenvolvimento de novas soluções para manipular este tipo de dado.

4. ALGORITMOS EVOLUTIVOS

A busca da melhor solução dentre uma coleção de soluções candidatas é um objetivo comum no desenvolvimento de algoritmos, tanto que na área da Ciência da Computação esse processo é conhecido por “espaço de busca” [Mel99]. Desta forma, para um espaço de busca pequeno, todas as soluções podem ser examinadas em um tempo razoável, e ao final, a ótima é encontrada. No entanto, esta busca exaustiva rapidamente se torna inviável quando o espaço de busca cresce em tamanho. Neste caso, tem-se denominado o problema como difícil ou intratável, o que em termos práticos, significa que seu tempo de execução é da ordem de uma função exponencial ou fatorial.

Em contrapartida, a evolução biológica é uma interessante fonte de inspiração para lidar com esse tipo de problema [Mel99]. AEs (Algoritmos Evolutivos), por exemplo, são uma coleção de técnicas de otimização inspiradas no processo de evolução Darwiniana. Essas técnicas têm se mostrado eficientes para varrer o espaço de busca e encontrar uma aproximação da solução ótima. Por esta mesma razão, AEs são considerados uma heurística, uma vez que não são capazes de garantir uma solução ótima, mas sim, uma aproximação desta.

O princípio de funcionamento de um AE é que a partir de uma população de indivíduos[1], cada indivíduo deva ser avaliado como um possível candidato à solução do problema em questão. Uma vez avaliados, deseja-se que aqueles mais aptos tenham maiores chances de serem selecionados para reprodução. Nesse processo, os pais selecionados recombinam suas características (*crossover*), gerando assim novos descendentes que, na maior parte dos casos, os substituirão em uma nova geração. Esses descendentes, por sua vez, estarão sujeitos à incidência de alterações em suas características (mutação), tal como acontece no processo biológico natural. Esta sequência de eventos é então repetida por novas gerações até que um determinado critério de parada seja alcançado. Para melhor compreensão, o fluxo de funcionamento de um AE é ilustrado na Figura 4.1.

Com base na literatura, pode-se verificar diversos tipos de AEs, tais como: Programação Evolutiva [FOW66], Estratégias Evolutivas [Rec73], Algoritmos Genéticos [Hol75] e Programação Genética [Koz92]. Neste trabalho é aprofundado o estudo sob AGs (Algoritmos Genéticos) e algoritmos de PG (Programação Genética), que são os tipos de AEs mais utilizados para tarefas de mineração de dados [Fre02]. O pseudocódigo comum a ambos é ilustrado no Algoritmo 4.1:

Embora AGs e algoritmos de PG compartilhem o mesmo pseudocódigo, existem importantes diferenças entre esses dois tipos de algoritmos. De acordo com Freitas [Fre02], as principais diferenças entre eles referem-se a suas respectivas representações e aplicações de operadores genéticos. Enquanto AGs frequentemente utilizam-se da representa-

[1] O termo cromossomo também é utilizado para referenciar um indivíduo.

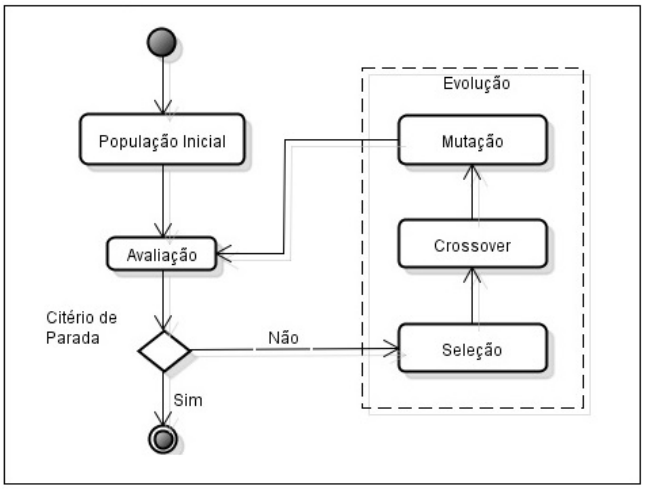


Figura 4.1 – Fluxo de um Algoritmo Evolutivo (Adaptado de [BRB11]).

Algoritmo 4.1 Pseudocódigo genérico para AGs e PGs (Adaptado de [Fre08])

```
1: Crie a população inicial de indivíduos
2: Calcule o fitness
3: Repita
4:     Selecione os indivíduos baseado no fitness
5:     Aplique operadores genéticos sob os indivíduos selecionados
6:     Crie novos indivíduos
7:     Calcule o fitness de cada novo indivíduo
8:     Atualize a população atual(novos indivíduos substituem velhos indivíduos)
9: Até(Critério de parada);
```

ção clássica de *array* de caracteres, a maior parte dos algoritmos de PG usa a representação de árvore. Neste caso, um indivíduo (solução candidata) é representado por uma árvore, onde em geral os nodos internos são funções e os nodos terminais são variáveis do problema sendo resolvido ou constantes [Fre02]. John Koza, pesquisador responsável por disseminar o conceito de PG para comunidade científica, argumenta que essas funções podem ser operações aritméticas, operadores de programação, funções matemáticas ou funções de domínio específico [Koz92].

Com base nas definições acima descritas, classifica-se a solução apresentada nesta Tese como um AE mais aproximado ao conceito de PG, uma vez que a representação a ser adotada não é baseada sob *arrays* de caracteres (binários ou não), mas sim, sob árvores dinâmicas. Entretanto, tem-se a compreensão de que tal classificação não é consensual na literatura.

4.1 Representação do Indivíduo

Encoding é o termo em inglês designado para referenciar a forma de codificação (ou representação) de cada indivíduo. Conforme observado neste capítulo, este processo é comumente realizado por meio de estruturas clássicas de *array* de caracteres ou árvores (embora não se limitem a essas). Para um maior detalhamento sobre possíveis variações, sugere-se Sivanandam e Deepa (2008) [SD08] como leitura complementar. Para fins de compreensão da solução proposta, prioriza-se aqui a representação baseada em árvores.

- **representação em *array*:** Essa representação foi adotada inicialmente na forma binária por Holland em seu livro seminal [Hol75], sendo hoje em dia ainda utilizada por pesquisadores da área devido à sua facilidade de implementação. Nesta forma de representação, cada indivíduo é um *array* de caracteres de tamanho fixo. Neste caso, alguém poderia facilmente representar os números entre 0 e 31 com um *array* binário de *5* bits, conforme ilustrado na Figura 4.2. À despeito de sua facilidade de implementação, o uso da representação baseada em *arrays* de caracteres não tem sido considerada natural para muitos problemas, e algumas vezes, correções devem ser feitas depois que a operação genética é completada. De acordo com Koza [Koz92], a seleção inicial do tamanho do *array* limita o número de estados internos do sistema e também limita o que o sistema pode aprender. Adicionalmente, a forma de representação baseada sob *array* de caracteres também não fornece uma maneira conveniente de representar procedimentos computacionais ou de incorporar iteração ou recursão quando essas capacidades poderiam ser desejáveis para um determinado problema.

- **representação em árvore:** Árvores, por sua vez, utilizam-se de representações gráficas podendo crescer em tamanho e forma de modo muito dinâmico, embora sua profundidade também deva ser definida previamente [Fre02]. De acordo com Barros et al. [BBDCF12], outro fator favorável à representação baseada em árvore diz respeito à sua facilidade para lidar com problemas de AM que envolvam a indução de árvores de decisão, regressão ou modelo. Barros argumenta que *array* de caracteres de tamanho fixo são tipicamente complicados para serem representados na forma de árvores de decisão/regressão/modelo não binário. Desta forma, a representação baseada em árvore parece ser a escolha natural para condução deste trabalho, uma vez que se propõe utilizar um AE para indução de árvores de regressão.

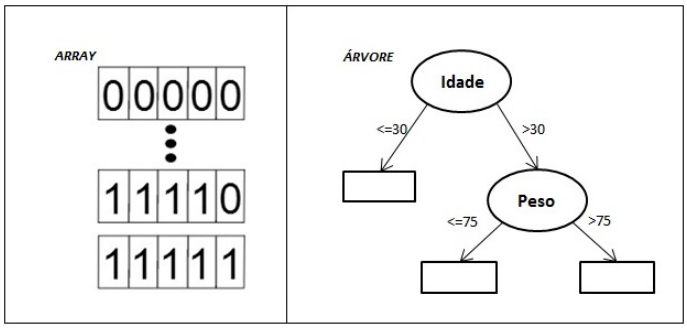


Figura 4.2 – *Enconding* em array e árvore.

4.2 Criação da População Inicial

Uma vez definida a forma de representação do indivíduo, o próximo passo de um AE consiste na definição da população inicial. Neste ponto, duas maiores questões são consideradas inicialmente: *i)* o tamanho da população, e posteriormente *ii)* a diversidade dos indivíduos. Com relação à definição do número de indivíduos da população inicial, pesquisadores da área têm argumentado que o uso de uma pequena população poderia levar os algoritmos a não explorar o espaço de busca eficientemente. Por outro lado, grandes populações poderiam fazê-los levar mais tempo para a convergência de uma solução. Desta forma, este tipo de decisão tem sido comumente parametrizado no algoritmo e definida de forma empírica.

Já para a questão relacionada à diversidade da população, é comum que a geração de indivíduos com características muito semelhantes seja evitada, uma vez que tal ocorrência poderia resultar na rápida convergência do algoritmo em uma solução ótima local. Para lidar com esse tipo de problema, a maior parte dos trabalhos sobre indução de árvores tem adotado uma inicialização parcialmente randômica dos indivíduos [BBDCF12]. De acordo com o autor, o termo “parcialmente” significa que a aleatoriedade dos indivíduos é restringida aos atributos do *dataset* e seus valores possíveis no conjunto de treino.

Entre os métodos mais conhecidos para geração randômica da população inicial, destacam-se:

- ***Full*:** No método *full*, atributos e limiares são selecionados randomicamente a partir de uma lista pré-definida até que a profundidade máxima seja alcançada. Esta profundidade também é definida aleatoriamente, variando do tamanho mínimo 2 até uma profundidade qualquer. Ao final deste processo, todas as folhas estão em um mesmo nível na árvore.
- ***Grow*:** O método *grow* funciona de maneira muito semelhante ao método *full*, mas ao contrário desse, permite que nodos terminais estejam em níveis diferentes, produzindo assim, árvores de diferentes formas;

- ***Ramped half-and-half*:** O método *Ramped half-and-half* foi proposto por Koza [Koz92], tendo como principal motivação o não fornecimento natural de tamanhos e formas por parte dos métodos *full* e *grow*. Desta forma, o método proposto consiste na combinação dos métodos *grow* e *full* de modo que, metade da população inicial é gerada por um e a outra metade por outro. Isto é feito utilizando-se um intervalo de profundidade limite para ajudar a garantir que as árvores geradas terão uma maior variedade de tamanhos e formas.

  Na Figura 4.3 ilustramos os métodos *Full*, *Grow* e *Ramped half-and-half*.

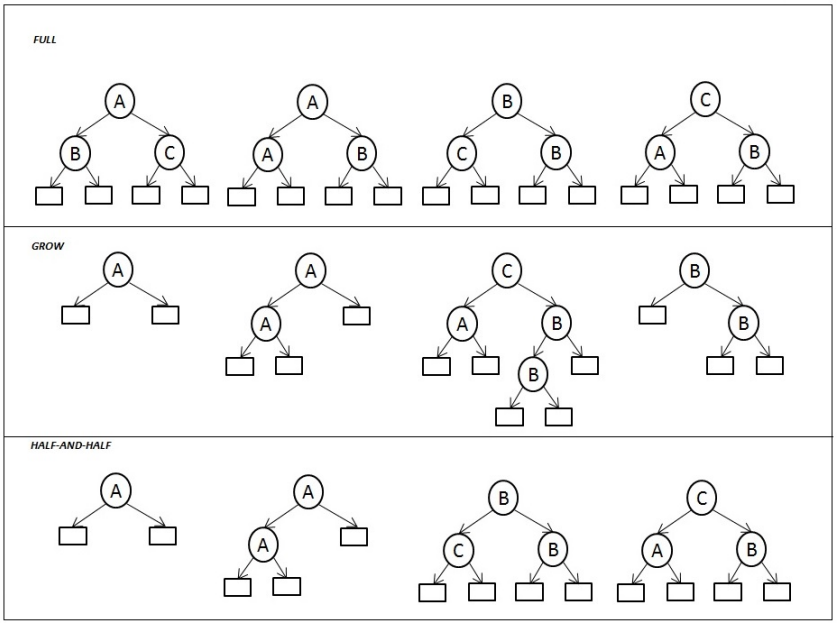


Figura 4.3 – Métodos de geração da população inicial.

Outra abordagem para inicialização da população inicial é dividir este processo em duas etapas. Em um primeiro passo são geradas diferentes árvores de dois níveis (nó raiz mais suas respectivas folhas). Posteriormente, essas árvores são combinadas de forma aleatória, podendo ter seu crescimento guiado por qualquer uma das formas anteriormente descritas (*Full*, Grow ou *Ramped half-and-half*). Mais recentemente, esta abordagem foi empregada nos trabalhos de Basgalupp et al. [BBdC+09] e Barros et al. [BBR+10]. Uma representação simplificada deste método é ilustrada na Figura 4.4.

4.3 Função de Avaliação

A função de avaliação é a maneira utilizada por um AE para determinar a qualidade do indivíduo como solução do problema em questão. Neste sentido, uma boa função de avaliação deve tanto ser capaz de diferenciar boas e más soluções, assim como, duas subótimas, deixando claro qual delas está mais próxima da solução desejada [Lin08].

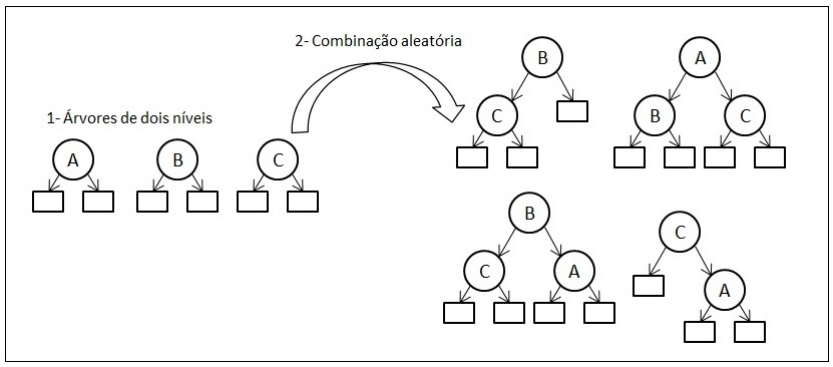


Figura 4.4 – Geração da população inicial com árvores de dois níveis.

Outra característica importante de uma função de avaliação diz respeito à maximização de um determinado objetivo, o que via de regra é feito de maneira numérica. Para o problema da construção de uma árvore de regressão, por exemplo, alguém poderia querer encontrar a melhor solução (ou uma aproximação desta) a partir da escolha da árvore com menor erro na predição. Neste caso, dizemos que a função de avaliação é mono-objetiva, pois usa um único critério para guiar a busca de soluções ótimas. Em contrapartida, poderia se estabelecer também uma relação mais complexa de avaliação, analisando o custo-benefício entre desempenho preditivo e compreensibilidade do modelo (dada pela quantidade de nodos da árvore). Para estes casos, onde se deseja otimizar mais de um critério, a função de avaliação é denominada multiobjetiva.

Entre as estratégias utilizadas para avaliação multiobjetiva, a análise lexicográfica tem seu funcionamento baseado na definição de uma lista de prioridades a partir de um conjunto de medidas (critérios) consideradas(os). Neste processo, a melhor solução é aquela significativamente melhor para a medida de maior prioridade. Caso esta diferença não seja significativa, a próxima medida na lista de prioridades é utilizada, e assim por diante até que os valores absolutos sejam comparados.

Em razão de determinar se a diferença é significativa ou não, a abordagem lexicográfica se utiliza de um limiar de tolerância para cada critério. De acordo com Ghosh et al. [GDG08], esses limiares podem ser obtidos a partir de procedimentos baseados em estatística (ex: desvio padrão), o qual permite rejeitar a hipótese nula de diferenças insignificantes entre dois valores objetivos com um certo grau de confiança.

Para uma melhor compreensão da abordagem lexicográfica, considere o exemplo a seguir. Assuma que x e y sejam dois indivíduos, e a e b dois critérios. Adicionalmente, considere que a tem prioridade sob b e que t_a e t_b são limiares de tolerância associadas com a e b, respectivamente. Com base nessas definições, a abordagem lexicográfica trabalha com a seguinte análise: se $|a_x - a_y| > t_a$, então é possível estabelecer uma escolha pelo melhor indivíduo, considerando apenas o critério a. Ao contrário, o critério b deve ser avaliado. Neste caso, se $|b_x - b_y| > t_b$, então o indivíduo com melhor avaliação entre x e

y pode ser decidido pela consideração do critério *b*. Se ainda assim a diferença não for significativa, utiliza-se apenas o valor absoluto de *a* para determinar o melhor indivíduo.

4.4 Critérios de Parada

De uma forma resumida, é possível citar os seguintes critérios de parada em um AE: *i)* Número máximo de gerações (o algoritmo para quando um número específico de gerações é atingido); *ii)* Tempo decorrido (assim como no número máximo de gerações, o algoritmo também termina ao atingir um determinado tempo de processamento); *iii)* Nenhuma mudança no *fitness* (o processo genético termina caso não haja nenhuma mudança no melhor *fitness* da população por um número específico de gerações); *iv)* Tempo de melhora (o algoritmo para se não houver nenhuma mudança no valor do *fitness* durante um determinado período de tempo); e *v)* Solução satisfatória (Neste caso, assume-se que o melhor indivíduo tenha sido encontrado).

4.5 Seleção de Pais

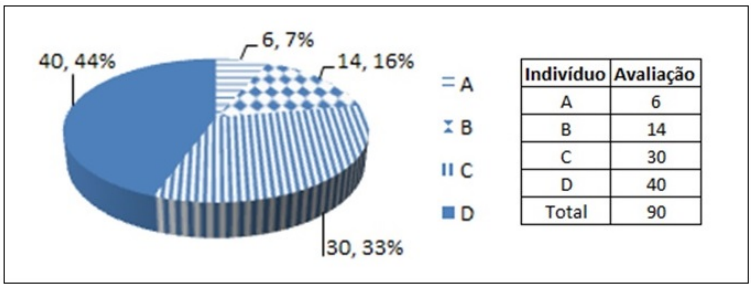
Seleção é um método que seleciona indivíduos da população de acordo com suas respectivas funções de avaliação. Neste tipo de método, é comum que indivíduos com melhor avaliação sejam privilegiados, no entanto, sem desprezar totalmente aqueles com menor avaliação. Esta decisão é razoável, pois até indivíduos com função de avaliação extremamente baixa podem ter características genéticas que sejam favoráveis à criação de um indivíduo que seja a melhor solução para o problema em questão [Lin08]. De acordo com Sivanandam [SD08], é possível distinguir dois tipos de esquemas de seleção:

- **Seleção proporcional:** A escolha dos indivíduos é baseada no valor de suas funções de *fitness* em relação a outros indivíduos da população. Neste caso, podemos dizer que as chances de um indivíduo ser selecionado são proporcionais ao seu valor de avaliação.
- **Seleção baseada em ordem:** A escolha dos indivíduos não é baseada apenas sobre o valor bruto da função de *fitness*, mas sim, a partir do ranqueamento do indivíduo dentro da população. Isto, em termos práticos, significa que a diferença entre as posições no *ranking* é mais relevante do que a margem da diferença dos valores brutos de *fitness*.

Nas próximas seções, são apresentados três tradicionais métodos de seleção envolvendo seleção proporcional e seleção baseada em ordem.

4.5.1 Roleta Viciada

A Roleta Viciada é um método de seleção conhecido que simula uma roleta virtual. Nessa roleta, cada indivíduo é representado por uma fatia proporcional ao seu *fitness*. Para selecionar *n* indivíduos, a roleta é girada *n* vezes (onde *n* é o número de indivíduos da população). A cada giro da roleta, um número *k* é obtido aleatoriamente. Esse número *k*, por sua vez, deve estar contido no intervalo de números possíveis para cada fatia da roleta. Na roleta ilustrada pela Figura 4.5, por exemplo, há 40 números possíveis na fatia que representa o melhor indivíduo. Isto, em termos probabilísticos, significa aproximadamente 44% de chances de ser selecionado (40 / 90).



Indivíduo	Avaliação
A	6
B	14
C	30
D	40
Total	90

Figura 4.5 – Ilustração do método Roleta Viciada.

4.5.2 Torneio

Neste método, *N* indivíduos são sorteados aleatoriamente, competindo um contra o outro para seleção do indivíduo com melhor *fitness*. Após uma primeira rodada, todos os indivíduos, inclusive o vencedor, retornam à população, podendo ser selecionados novamente para um novo torneio. Este processo se repete até que o número desejado de reprodutores seja alcançado. O método de Torneio é ilustrado na Figura 4.6.

4.5.3 Ranking

Ranking é um método de seleção baseado em ordem, onde os indivíduos da população são ordenados de acordo com o seu *fitness*. Assim, as chances de seleção de cada indivíduo dependem mais de sua posição no *ranking* do que do valor absoluto de seu *fitness*. Diferentemente da Roleta Viciada, o método de *Ranking* evita dar probabilidade maior de seleção para um grupo pequeno de indivíduos com alta avaliação, reduzindo, assim, a pressão da seleção quando a variação do *fitness* é muito elevada.

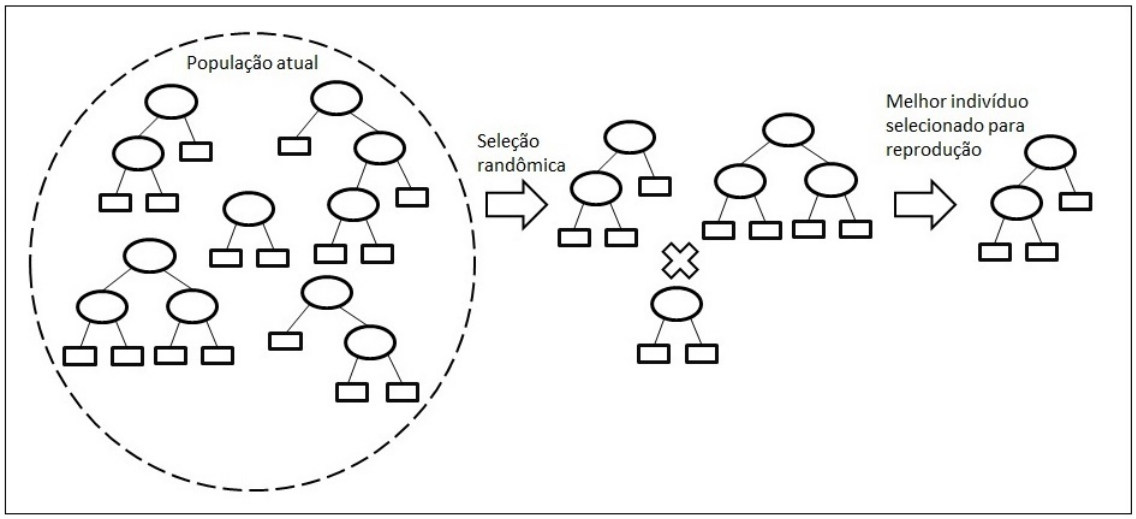


Figura 4.6 – Ilustração do método Torneio.

4.6 Operadores de Crossover e de Mutação

Os operadores baseados em árvore têm operação extremamente simples. O operador de *crossover* serve para intercambiar informações entre pais, e o operador de mutação serve para inserir variedade genética na população de forma aleatória [Lin08]. No operador de *crossover*, nodos são selecionados aleatoriamente em cada uma das árvores selecionadas para reprodução, e realiza-se o intercâmbio entre as subárvores enraizadas em cada um destes nós. Ao final deste processo, dois novos filhos são gerados, contendo "fragmento" de cada um dos pais, conforme ilustrado na Figura 4.7.

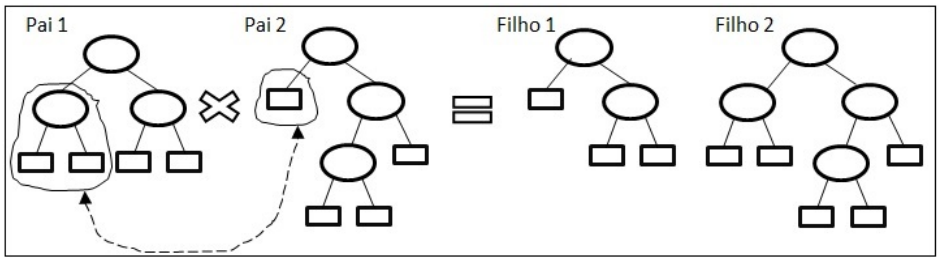


Figura 4.7 – Exemplo de crossover.

Para mutação, dois métodos são utilizados. No primeiro, escolhe-se uma subárvore aleatoriamente para depois trocá-la por um nodo folha. O segundo método, por sua vez, consiste em selecionar um nodo folha do indivíduo e posteriormente trocá-la por uma das árvores básicas produzidas na criação da população inicial. Ambos processos são ilustrados na Figura 4.8.

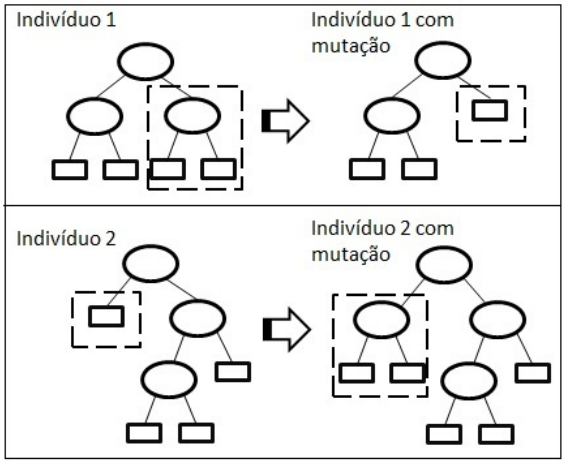


Figura 4.8 – Exemplo de mutação.

4.7 Considerações do Capítulo

A computação evolutiva é uma das sub-áreas da computação bioinspirada que tem apresentado um forte crescimento nos últimos anos. Isto deve-se principalmente a um vasto conjunto de técnicas que têm se espelhado no processo de evolução Darwiniana.

De um modo geral, o princípio de funcionamento de um AE consiste em manter uma população de indivíduos, onde cada um desses é uma potencial solução para um determinado problema. Esses indivíduos são então avaliados e posteriormente selecionados para aplicação de operadores genéticos. Os operadores genéticos, por sua vez, consistem em aproximações computacionais de fenômenos vistos na natureza, tal como a recombinação de genes e as mutações genéticas. Este processo é então repetido por novas gerações até que uma solução satisfatória seja obtida para o problema em questão.

Neste capítulo, apresentamos uma breve revisão sobre algoritmos evolutivos, abordando métodos que são essenciais para compreensão da solução proposta neste trabalho. Para uma análise mais abrangente da área, as seguintes literaturas são sugeridas: [Fre02], [SD08] e [Lin08].

5. ALTIVO (ALGORITMO EVOLUTIVO PARA INDUÇÃO DE ÁRVORES DE REGRESSÃO)

Neste capítulo é apresentado o algoritmo AltIvo, uma nova abordagem evolutiva para indução de árvores de regressão. Como diferencial, AltIvo tem sua implementação focada na otimização de desempenho preditivo sobre *datasets* com altos percentuais de dados ausentes. Para tanto, essa solução incorpora uma estratégia baseada no conceito de IM. Nas próximas seções são apresentados maiores detalhes sobre o funcionamento do algoritmo.

5.1 Motivação

Conforme pode-se observar no Capítulo 2, dados ausentes têm sido uma característica comum em aplicações do mundo real. Para lidar com esse problema, grande parte dos algoritmos de aprendizado de máquina tem baseado sua implementação a partir de uma única estratégia de tratamento (ex: média), o que não se reflete necessariamente em uma melhor solução. Esse fato deve-se principalmente à variabilidade de características do *dataset* (ex: mecanismos de distribuição dos dados, correlação dos dados, percentual de dados ausentes, etc.), fazendo com que diferentes configurações de *datasets* exijam diferentes soluções. Em outras palavras, isto significa que não há uma solução universal para o problema de dados ausentes.

Por outro lado, a manipulação de dados ausentes durante o pré-processamento tem se mostrado uma possível alternativa ao tratamento interno, onde a escolha do método é normalmente realizada pelo usuário. No entanto, a descoberta do método ótimo é um processo extremamente oneroso, quase sempre realizado por uma estratégia de tentativa e erro.

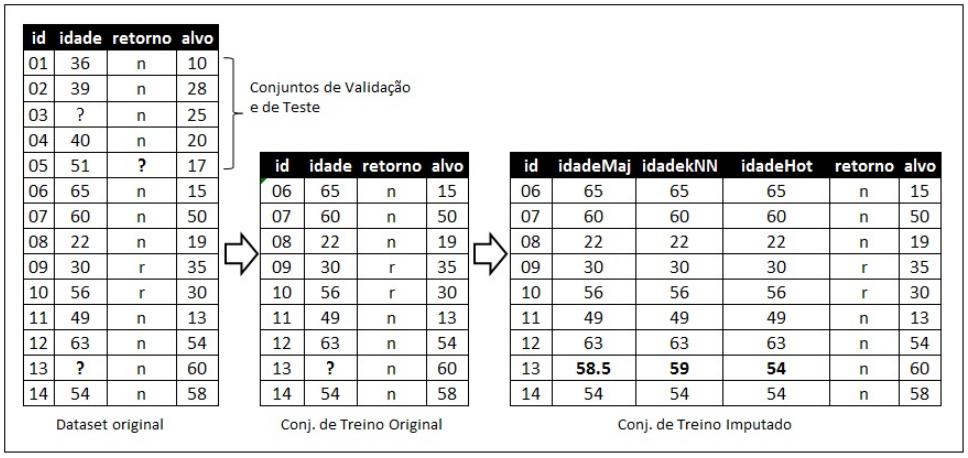
Com base no exposto até aqui, trabalha-se com a hipótese de que a solução ideal para indução de modelos robustos aos dados ausentes deva incorporar múltiplos métodos de tratamento e, ao mesmo tempo, não onerar o usuário. Partindo desse princípio, propõe-se um novo algoritmo evolutivo para indução de árvores de regressão, utilizando-se para tanto, de uma estratégia baseada no conceito de IM. Desde então, tem-se incorporado esse conceito no ciclo evolutivo do algoritmo AltIvo. Como principal objetivo a ser alcançado, espera-se estimar melhor os dados faltantes, e consequentemente, otimizar o desempenho preditivo dos modelos gerados. A seguir, são apresentados maiores detalhes sobre a solução proposta.

5.2 Módulo de Tratamento de Dados Ausentes

O módulo de tratamento de dados ausentes do algoritmo AltIvo consiste na implementação de diferentes estratégias. Embora o algoritmo permita a inclusão de novas estratégias, tem-se avaliado inicialmente estratégias conhecidas na literatura: Eliminação de atributo, *Majority*, k-NN e *Hot-Deck*. A habilitação dessas estratégias é definida por parâmetro, onde 0 significa inativa, e 1 ativa. Para a eliminação de atributos, em particular, tem-se utilizado o intervalo de 1 a 100 para definir o limiar de incompletude com que o atributo deve ser removido do *dataset*. Para fins de avaliação, é considerado um k=3 para aplicação do método de k-NN. Esse valor é igualmente parametrizável no algoritmo.

Depois de definidos os parâmetros de imputação, as estratégias são aplicadas sob os atributos do conjunto de treino onde existe pelo menos uma instância com valor ausente. Esse processo é executado para cada iteração na validação cruzada de 10 folds, excluindo-se os conjuntos de validação e de teste, onde, em teoria, desconhecemos a informação necessária (sobre o atributo alvo) para aplicação de métodos de imputação supervisionada, como é o caso dos métodos *Majority* e *Hot-Deck*.

Ao final das imputações, o atributo original é descartado, e as versões imputadas são adicionadas em um novo *dataset*, aqui denominado: Conjunto de treino imputado. Uma primeira visão sobre imputações é fornecida na Figura 5.1



id	idade	retorno	alvo
01	36	n	10
02	39	n	28
03	?	n	25
04	40	n	20
05	51	?	17
06	65	n	15
07	60	n	50
08	22	n	19
09	30	r	35
10	56	r	30
11	49	n	13
12	63	n	54
13	?	n	60
14	54	n	58

Dataset original

id	idade	retorno	alvo
06	65	n	15
07	60	n	50
08	22	n	19
09	30	r	35
10	56	r	30
11	49	n	13
12	63	n	54
13	?	n	60
14	54	n	58

Conj. de Treino Original

id	idadeMaj	idadekNN	idadeHot	retorno	alvo
06	65	65	65	n	15
07	60	60	60	n	50
08	22	22	22	n	19
09	30	30	30	r	35
10	56	56	56	r	30
11	49	49	49	n	13
12	63	63	63	n	54
13	**58.5**	**59**	**54**	n	60
14	54	54	54	n	58

Conj. de Treino Imputado

Figura 5.1 – Criação do conjunto de treino imputado.

Para uma melhor compreensão sobre o processo de imputação, considere o atributo idade com valor ausente na instância de id=13. Como primeiro passo a ser tomado, buscou-se particionar as instâncias do conjunto de treino em quatro grupos, utilizando para tanto, a mediana dos valores contidos no atributo alvo. Tendo o valor 35 como elemento central, repetimos o procedimento mais duas vezes: uma para os valores à esquerda da mediana (13, 15, 19, 30) e outra para os valores à direita (50, 54, 58, 60). Neste ponto, têm-se os elementos necessários para definição dos seguintes limiares: >=0 e <19, >=19 e <35, >=35

e <54 e >=54. Conforme visto na fundamentação teórica, este procedimento é realizado em função dos métodos *Majority* e *Hot-Deck*, os quais baseiam seu processo de imputação a partir de uma estratégia de estimativa por grupos.

Desta forma, assumindo que o processo de particionamento das instâncias tenha identificado os grupos G1(id06,id11), G2(id08,id10), G3(id07,id09) e G4(id12,id13,id14), aplica-se os métodos *Majority* e *Hot-Deck* (ver Figura 5.2) apenas para aqueles grupos onde haja alguma instância com valor ausente (neste caso, apenas para o grupo G4). Para o cálculo no método *Majority* é calculado a média interna dentro do grupo, obtendo assim, o valor 58,5 para idade. Já para o método *Hot-Deck*, busca-se a instância mais similar no grupo através da aplicação do método de k-NN (k=1). Como resultado, o valor 54 é obtido para idade uma vez que sua instância apresenta a menor distância euclidiana (0,6168) dentre as instâncias do grupo.

No entanto, em algumas situações (ex: normalmente em *datasets* pequenos), será possível ter grupos formados por apenas uma instância (aquela com dado ausente), o que necessariamente precisa ser tratado. Neste caso, adota-se a média/moda do conjunto de treino imputado como procedimento padrão para ambos métodos de imputação. Para o método *Majority*, em particular, o procedimento também tem sido admitido na ocorrência de grupos com duas instâncias, uma vez que não faz sentido calcular a média da única instância do grupo com valor para o atributo.

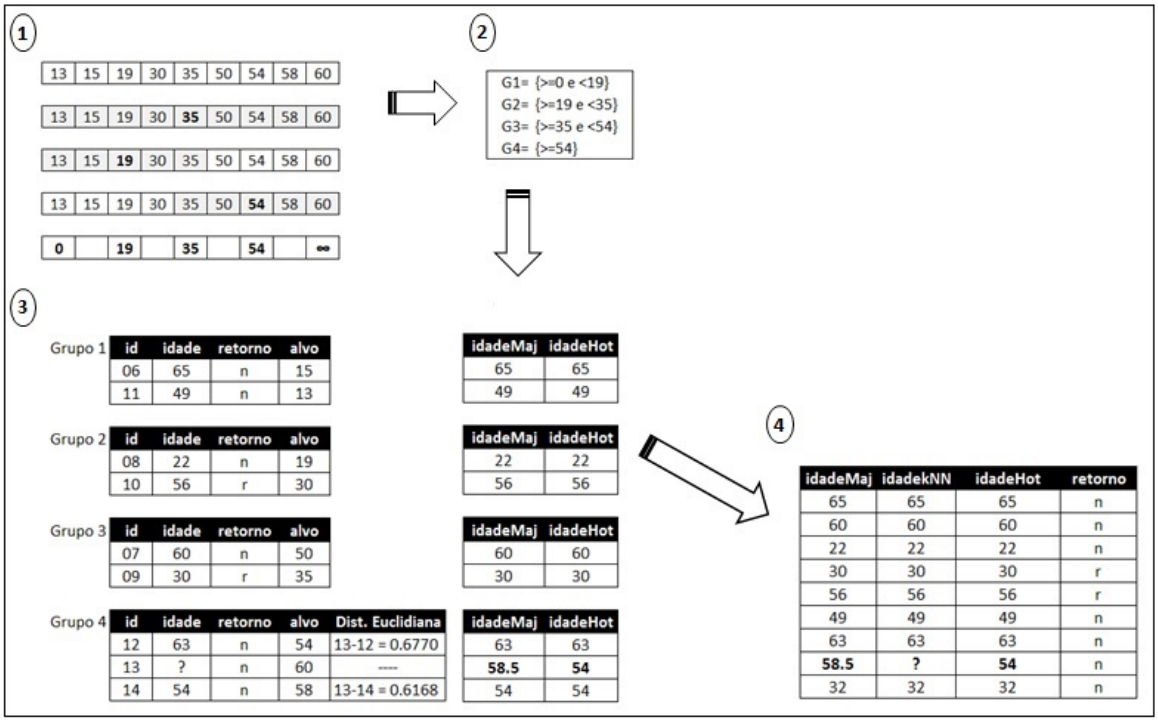


Figura 5.2 – Imputações *Majority* e *Hot-Deck*.

Diferentemente dos métodos *Majority* e *Hot-Deck*, o método k-NN independe da obtenção de grupos, pois fundamenta-se numa estratégia de busca global pelos *k* elementos mais similares àquele com valor ausente. Na solução aqui proposta, *k*=3 é usado para ava-

liação inicial, deixando sua alteração como uma opção parametrizável no algoritmo. Para o exemplo em questão, sua aplicação tem o valor 59 como resultado, dado que as instâncias mais similares são as de id 7, 12 e 14. Na Figura 5.3, o processo de imputação através do método de k-NN é apresentado.

id	idade	retorno	alvo	Dist. Euclidiana	idadekNN
06	65	n	15	13-06 = 0.6770	65
11	49	n	13	13-11 = 0.7071	49
08	22	n	19	13-08 = 0.6168	22
10	56	r	30	13-10 = 0.8388	56
07	60	n	50	13-07 = 0.1504	60
09	30	r	35	13-09 = 0.8009	30
12	63	n	54	13-12 = 0.0902	63
13	?	n	60	----	**59**
14	54	n	58	13-14 = 0.0300	54

Figura 5.3 – Imputação através do método de k-NN.

Nas próximas seções, apresentaremos as demais etapas do algoritmo AltIvo, abordando para tanto, as questões inerentes ao tratamento de dados ausentes.

5.3 Geração da População Inicial

Para geração da população inicial, é utilizada a mesma abordagem empregada em [BBdC+09, BBR+10], realizando, no entanto, alguns aperfeiçoamentos para o tratamento de dados ausentes e controle de inconsistências (ex: limiares incorretos). Nesta abordagem, o processo de geração de cada indivíduo é dividido em duas etapas. Na primeira etapa são geradas árvores básicas com particionamentos binários para cada atributo do *dataset*. Cada árvore básica é composta por um nodo raiz e dois nodos folhas. A quantidade de árvores básicas produzidas por atributo é variável conforme a tipagem do atributo.

Nos atributos categóricos, esta quantidade é definida pelo cálculo da média do atributo alvo para cada uma das *k* categorias do atributo em questão. Esses valores são então ordenados, e utilizados para criação de *k-1* árvores básicas binárias. Neste procedimento, o problema de dados ausentes é contornado por meio do uso do conjunto de treino imputado, eliminando assim, a necessidade de qualquer cálculo adicional. Como consequência lógica deste processo, o espaço de busca é ampliado em até três vezes (no pior caso). Isto deve-se à criação de três novas versões para cada atributo com dado ausente.

Para melhor compreensão, na Figura 5.4 é ilustrado o processo de criação de árvores básicas para atributos categóricos, considerando, para tanto, o seguinte exemplo. Estação é um atributo categórico utilizado na previsão de volumes de chuva, cujo número de categorias é igual a 4 (Outono, Inverno, Primavera e Verão). A partir do atributo alvo, AltIvo calcula a média do volume de chuvas, agrupando as instâncias do *dataset* pelas categorias

do atributo Estação. Como resultado desta operação, são gerados os seguintes valores: Outono=980mm, Inverno=1.230mm, Primavera=725mm e Verão=780mm. Esses valores são então ordenados, produzindo-se a seguinte enumeração: Primavera, Verão, Outono e Inverno. Dado que o número de categorias seja igual a quatro, tem-se considerado 3 possíveis particionamentos binários (*k-1*). Para os casos onde o atributo categórico seja imputado, multiplica-se por 3 o número de árvores básicas resultantes para o atributo em questão.

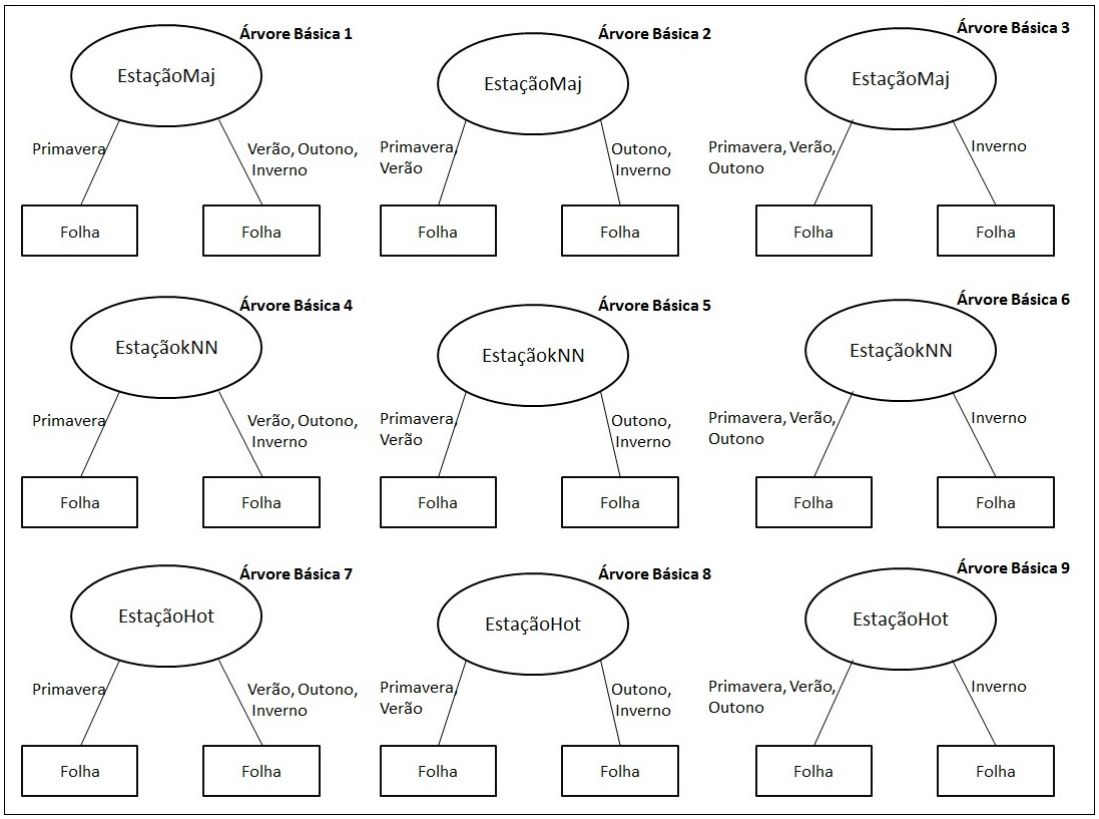


Figura 5.4 – Criação de árvores básicas para atributos categóricos.

Nos atributos numéricos, o conjunto de treino imputado é dividido em diferentes pedaços, calculando-se o SDR (*Standard Deviation Reduction*) para obtenção dos limiares. Esta estratégia de agrupamento pode ser compreendida da seguinte forma: primeiramente, todo conjunto de treino imputado é utilizado para o cálculo do SDR. Em seguida, este mesmo conjunto é dividido aleatoriamente em quatro pedaços, calculando-se novamente o SDR para cada um deles. Finalmente, os valores de SDR são utilizados para definição de cinco novas árvores básicas com limiares potencialmente diferentes. De acordo com [BBR+10], esta abordagem tem duas principais vantagens: i) os limiares são definidos de uma maneira dirigida pelos dados (ex: usando o SDR para selecionar valores de limiares interessantes), em vez de selecionar aleatoriamente valores, que é o caso da maior parte das abordagens evolutivas; e ii) um certo grau de heterogeneidade é alcançado pelo particionamento do conjunto de treino (nesta solução, imputado) em diferentes pedaços, aumentando as chances de selecionar um bom valor de limiar.

Na segunda etapa, as árvores básicas são combinadas aleatoriamente conforme uma estratégia de crescimento do tipo *grow* (ver Figura 5.5). No algoritmo proposto, é adotada uma avaliação empírica para definição da quantidade de indivíduos da população, assim como, para definição de sua profundidade máxima. Embora tenha-se definido inicialmente estes valores com 100 (tamanho da população) e 2 (profundidade), essa configuração é mantida como uma opção parametrizável no algoritmo.

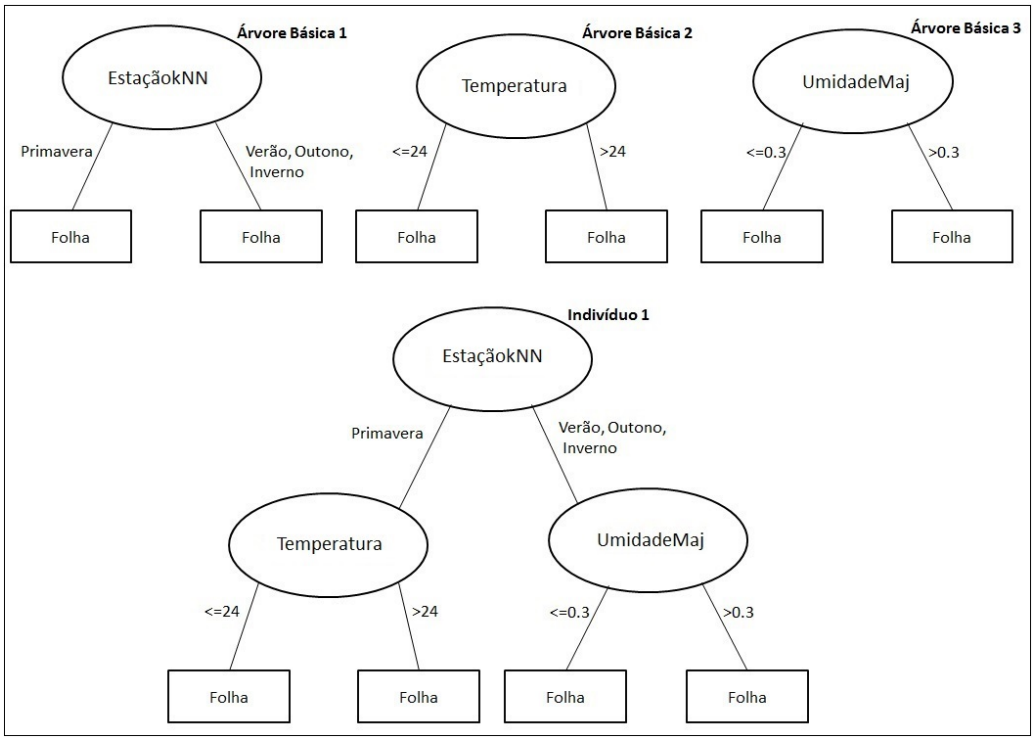


Figura 5.5 – Formação do indivíduo na geração da população inicial.

Uma outra característica do algoritmo AltIvo, diz respeito à verificação de inconsistências durante o processo de geração da população inicial (ver Figura 5.6). Este procedimento é implementado em tempo de execução na criação da árvore, evitando que um mesmo nodo categórico seja adicionado em um mesmo ramo, assim como, a ocorrência de inconsistências de limiares entre nodos numéricos. Para tanto, é mantido um *array* de restrições para cada nodo adicionado na árvore. Neste *array* são armazenadas informações sobre os nodos e limiares já percorridos, evitando assim, inclusões indesejadas. Contudo, não foi possível reproduzir este mesmo processo para as demais etapas do AE, uma vez que as restrições armazenadas dizem respeito apenas aos nodos pais na árvore. Neste caso, seria praticamente inviável controlar as consistências de subárvores adicionadas após a aplicação dos operadores de *crossover* e mutação. Contudo, esse problema é resolvido pelo uso de um procedimento de poda durante a distribuição das instâncias de treino em cada indivíduo (maiores detalhes são fornecidos na Seção 5.4).

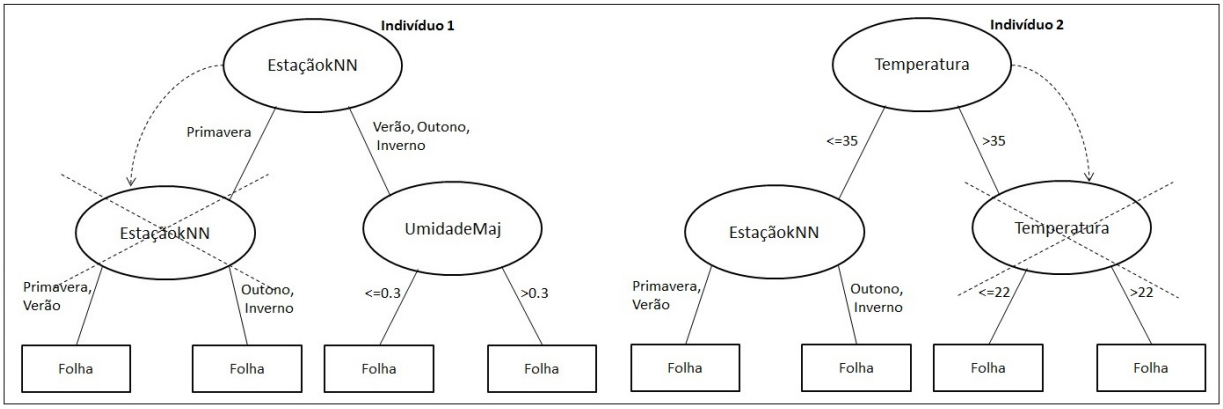


Figura 5.6 – Controle de inconsistências na geração da população inicial.

5.4 Indução dos modelos e controle de inconsistências

Depois de gerada a população inicial, a próxima etapa no algoritmo consiste na obtenção dos valores previstos para cada indivíduo. Na solução proposta, este processo se dá por meio da distribuição das instâncias de treino imputadas na árvore até que um nodo folha seja alcançado. Para obtenção do valor previsto, considera-se a média do atributo alvo para as instâncias presentes em cada nodo folha.

Com base nas predições realizadas, a qualidade de cada indivíduo é avaliada, considerando-os posteriormente para uma possível troca de material genético. Nesse ponto, uma nova população é produzida, e novas previsões são calculadas, iniciando-se assim, um novo ciclo evolutivo. Contudo, este é um processo que tende a trazer algumas situações problemáticas à medida que a aplicação de operadores genéticos tipicamente promove novas combinações de nodos e limiares (não necessariamente válidas). Para prevenir a ocorrência deste tipo de situação, uma espécie de poda é aplicada em todos os ramos cujos nodos folhas estejam vazios.

Adicionalmente, são calculados novos limiares para os casos onde um mesmo nodo com atributo numérico se repita. Desta forma, evita-se a ocorrência de qualquer inconsistência de limiar. Na Figura 5.7, são ilustradas as situações de inconsistências aqui mencionadas.

5.5 Função de Avaliação

Uma vez criada a população inicial, o algoritmo AltIvo tem implementado uma avaliação multiobjetiva para mensuração da qualidade de cada indivíduo. Nesta avaliação, é levada em consideração duas medidas quantitativas e uma qualitativa. Como medidas quantitativas, são utilizadas o RMSE e MAE (ilustradas nas equações 5.1 e 5.2), as quais

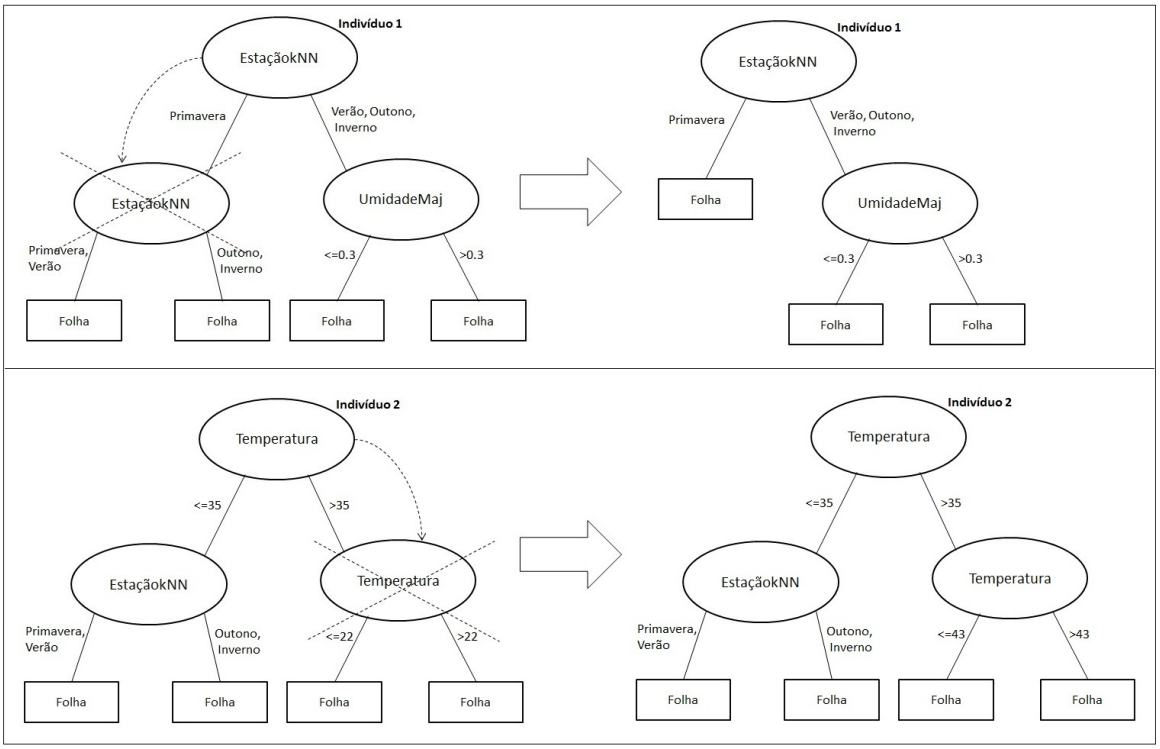


Figura 5.7 – Controle de inconsistências após a aplicação de operadores genéticos.

são comumente aplicadas em trabalhos de AM. Em última instância, essas medidas visam quantificar o erro produzido para cada indivíduo na predição de casos de validação e de teste. No entanto, alguns autores têm argumentado que este tipo de avaliação não é suficiente por si só, especialmente em casos no qual o modelo induzido não seja interpretável. A partir desta argumentação, optou-se por avaliar a compreensibilidade de cada árvore gerada, utilizando para tanto, o número de nodos da mesma.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Vpred_i - Vreal_i| \tag{5.1}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (Vpred_i - Vreal_i)^2} \tag{5.2}$$

onde, N é o número total de instâncias, $Vpred_i$ é o i-ésimo valor predito para o atributo alvo, e $Vreal_i$ é o i-ésimo valor real deste mesmo atributo.

Outro aspecto a ser levado em conta, diz respeito à distribuição de instâncias de validação na árvore. Na maior parte dos casos, essas instâncias são normalmente distribuídas. No entanto, para as instâncias de validação com valores ausentes, é verificado o tipo de imputação aplicada no nodo e posteriormente distribuído a instância na árvore. Para uma melhor compreensão, considere o seguinte exemplo: seja *idadeMaj* um nodo imputado com o método *Majority*, cujo valor da instância de validação também é desconhecido. Para a dis-

tribuição desta instância na árvore, a média das instâncias do conjunto de treino imputado para o atributo em questão (*IdadeMaj*) é utilizada. Este mesmo procedimento é aplicado para caso de árvores com mais de uma versão de imputação para um mesmo atributo (*IdadeMaj* e *IdadekNN*). Por final, esse procedimento é aplicado recursivamente até que todas instâncias alcancem os nodos folhas com suas respectivas previsões. Neste ponto, são coletadas informações sobre o tamanho da árvore, MAE e RMSE que serão posteriormente empregados dentro de uma estratégia lexicográfica de avaliação multi-objetiva.

- **Análise Lexicográfica:** O princípio de funcionamento da abordagem lexicográfica baseia-se na comparação de todos os indivíduos a partir de uma ordem de prioridades previamente definida no algoritmo. Neste processo, os indivíduos são comparados dois a dois, e o melhor é declarado caso a diferença entre eles seja considerada significativa para a medida em questão. Isto, por via de regra, significa que a diferença entre os valores da medida está acima de um determinado limiar. Na solução desenvolvida, o RMSE foi definida como medida prioritária, seguida pelo MAE e pelo tamanho da árvore. Assim, para o caso do RMSE, por exemplo, a diferença entre dois indivíduos é avaliada, e caso não seja maior que o limiar estabelecido, passa-se a comparar os indivíduos pela próxima medida na ordem de prioridades. Esse mesmo raciocínio é repetido para o MAE até que se compare o tamanho da árvore. Se todas as diferenças caírem dentro dos limiares atribuídos para cada medida, o menor valor absoluto de RMSE indicará o vencedor. Se ambos indivíduos tiverem o mesmo valor de RMSE, o valor absoluto do MAE é utilizado e assim por diante. Caso eles apresentem os mesmos valores para todas as medidas, qualquer um dos indivíduos poderá ser selecionado. Ao final, todos os indivíduos são classificados do melhor ao pior.

  Com relação à definição dos limiares, adotamos duas configurações: *i)* para as medidas de RMSE e MAE, é assumida uma tolerância de 5% do valor médio de cada medida para indicar qual solução era superior. *ii)* para o tamanho da árvore, é adotada uma tolerância de 20% sob o tamanho médio das árvores na população.

- **Fórmula Ponderada:** Além da abordagem lexicográfica, AltIvo também implementa a tradicional abordagem de fórmula ponderada, cujo princípio básico fundamenta-se na atribuição de pesos para cada uma das medidas consideradas, obtendo-se assim, um único valor como resultado. Na solução desenvolvida, optou-se inicialmente por priorizar as medidas de erro em detrimento do tamanho da árvore. Para tanto, os pesos de 40% são adotados para ambas medidas de erro, e 20% para o tamanho da árvore. Contudo, é importante ressaltar que este tipo de decisão tende a ser problemática, uma vez que a escolha do peso ideal sofre do problema do "número mágico". Desde então, esses valores são definidos empiricamente, conforme ilustrado na Equação 5.3.

$$
\begin{aligned}
Fitness(I_x) = 0,4 \times (\frac{RMSE(I_x) - argmin(RMSE)}{argmax(RMSE) - argmin(RMSE)}) + \\
0,4 \times (\frac{MAE(I_x) - argmin(MAE)}{argmax(MAE) - argmin(MAE)}) + \\
0,2 \times (\frac{Size(I_x) - argmin(Size)}{argmax(Size) - argmin(Size)})
\end{aligned}
\tag{5.3}
$$

onde I_x é um dado indivíduo da corrente população. O maior valor de uma determinada medida é dado por *argmax(medida)* e o menor valor por *argmin(medida)*.

Finalmente, dentre as estratégias implementadas, optou-se por avaliar a abordagem lexicográfica, já que esta é capaz de tratar cada critério separadamente, reconhecendo que cada critério mede um diferente aspecto de qualidade da solução candidata [Fre04]. Este tipo de discernimento é comumente ignorado na abordagem de fórmula ponderada à medida que esta tende a misturar critérios que nem sempre são comensuráveis.

5.6 Seleção

A seleção de indivíduos no algoritmo AltIvo ocorre por meio do tradicional método de Torneio. Para tanto, *n* indivíduos da população atual são randomicamente selecionados (por padrão, *n*=3), sendo escolhido o melhor desses para uma possível troca de material genético com um outro indivíduo igualmente selecionado. Nessa troca de material genético (*crossover*) é gerado um novo par de descendentes que os substituirão em uma nova população. Esse processo é então repetido, armazenando-se os descendentes em uma estrutura temporária até que o número total de indivíduos da população esteja completa. Neste sentido, AltIvo também implementa a técnica de elitismo, preservando 1% dos melhores indivíduos da população atual para a nova população sendo produzida. Obviamente, esses indivíduos não são considerados para a possível aplicação dos operadores de *crossover* e mutação. O método Torneio aplicado na solução é ilustrado na Figura 5.8.

5.7 Operadores Genéticos

Uma vez selecionados os pares de indivíduos na etapa de seleção, o próximo passo no algoritmo consiste na possível aplicação do operador de *crossover*. Essa aplicação é

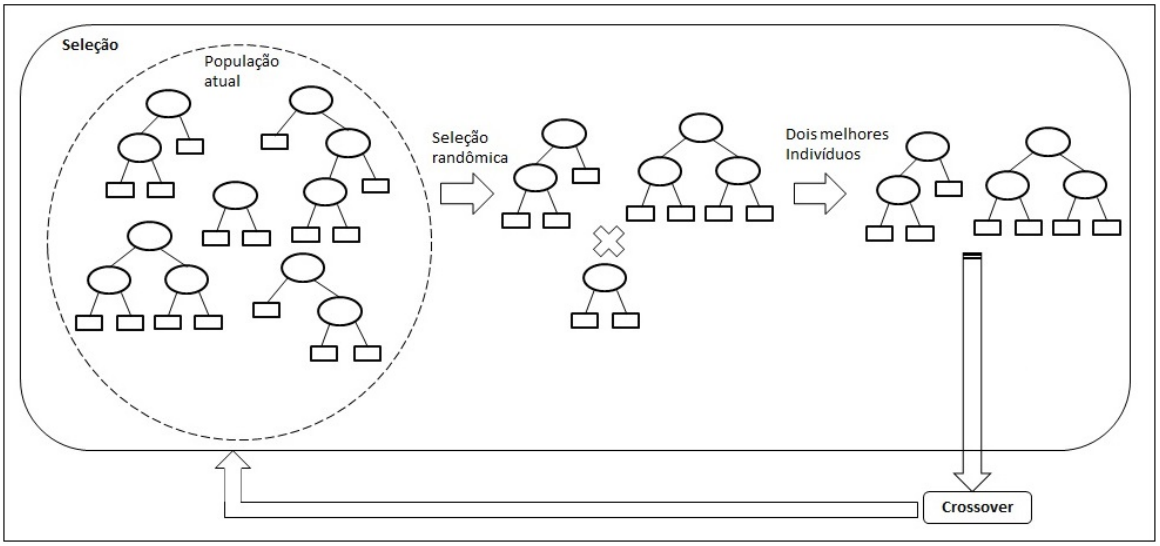


Figura 5.8 – Seleção no algoritmo AltIvo.

realizada conforme uma determinada probabilidade que é previamente configurada no algoritmo (por padrão, 90%). Para os casos onde não seja aplicado o referido operador, realiza-se uma cópia de ambos indivíduos, adicionando-os na estrutura temporária da nova população. Depois de preenchida (desconsiderando-se o(s) indivíduos do elitismo), a estrutura é percorrida, considerando-se cada um de seus indivíduos para a possível aplicação do operador de mutação. Neste caso, a probabilidade definida é de 5%. Ao final do ciclo evolutivo, é obtida uma nova população que substituirá a população atual. Nas próximas subseções, são apresentados maiores detalhes sobre a aplicação dos operadores de *crossover* e mutação.

5.7.1 Crossover

A operação de *crossover* implementada no algoritmo AltIvo consiste basicamente na seleção de nodos em cada par de indivíduos selecionados, e posteriormente na troca desses entre ambos. Para tanto, os nodos são selecionados randomicamente a partir de um número que varia de 1 até o número máximo de nodos para cada indivíduo em questão. Com base nesse número, AltIvo realiza uma busca pré-ordenada, visitando recursivamente o nodo raiz e seus filhos da esquerda para a direita.

Depois de identificado os nodos em ambos os indivíduos, AltIvo troca subárvores inteiras representadas por estes, gerando assim, dois novos indivíduos que substituirão seus pais em uma nova população. Esse processo é então repetido até que a estrutura temporária com a nova população esteja completa, e pronta para a aplicação do operador de mutação. O *crossover* implementado é ilustrado na Figura 5.9.

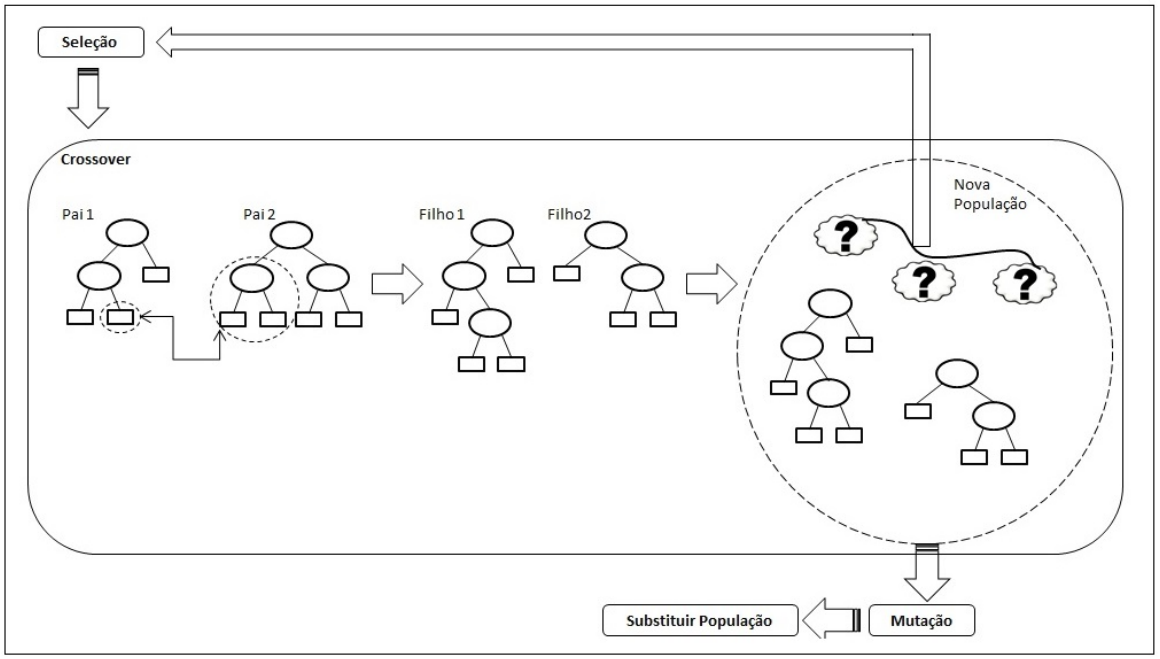


Figura 5.9 – Crossover no algoritmo AltIvo.

Entre os possíveis efeitos gerados pelo *crossover*, são identificados a geração de regras e limiares inconsistentes. Essas situações eram inicialmente evitadas durante a geração da população inicial, mas aqui, são preservadas para posterior tratamento (ver Seção 5.3). Sob o ponto de vista do tratamento de dados ausentes, não se tem identificado qualquer efeito negativo.

5.7.2 Mutação

Para cada indivíduo armazenado na estrutura temporária, AltIvo tem considerado a aplicação de duas diferentes estratégias de mutação. Na primeira, uma subárvore inteira é selecionada randomicamente no indivíduo, e posteriormente trocada por um nodo folha. Na segunda, repete-se o mesmo procedimento randômico, no entanto, para seleção de um nodo folha, trocando esse por uma das árvores básicas geradas durante a criação da população inicial. Essas estratégias objetivam aumentar ou diminuir o tamanho do indivíduo, aumentando a heterogeneidade da população e evitando a convergência em um local ótimo [BBR+10, BBdC+09]. A Figura 5.10 ilustra a operação de mutação.

Ao final da mutação, a população atual é substituída pelos indivíduos armazenados na estrutura temporária e por aquele(s) obtidos através da técnica de elitismo, repetindo-se esse processo por um número de gerações previamente definido no algoritmo.

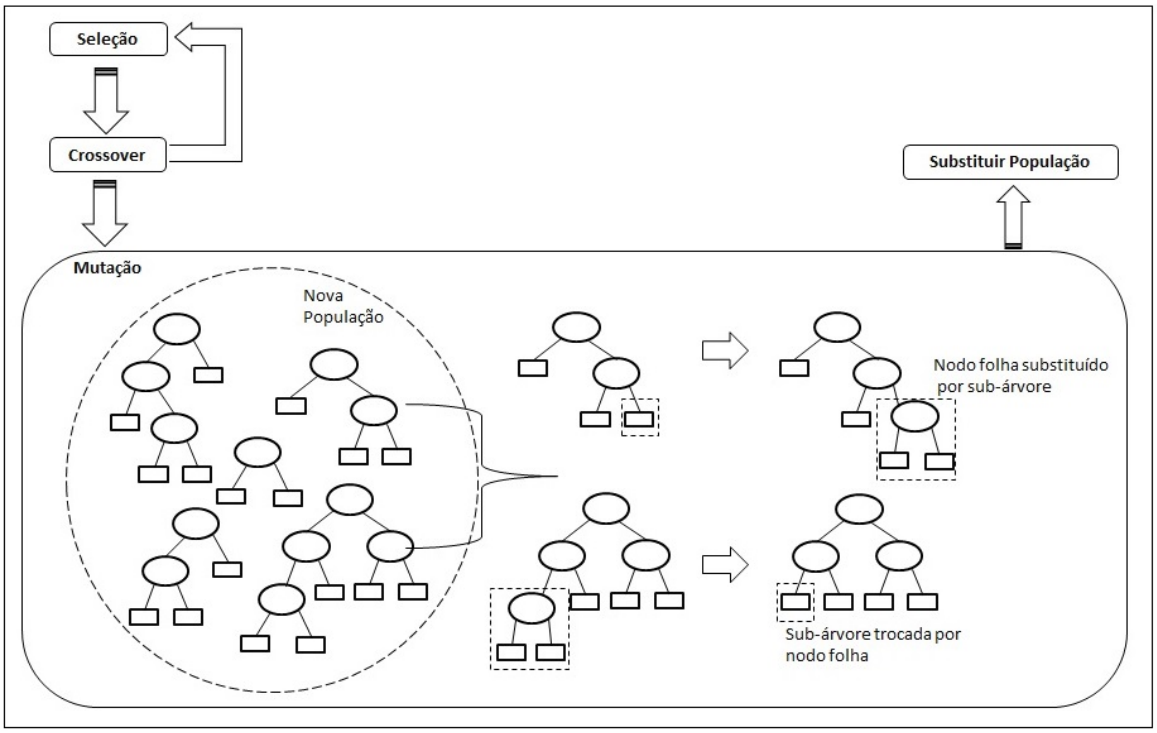


Figura 5.10 – Mutação no algoritmo AltIvo.

5.8 Critérios de Parada

Depois de iniciado o ciclo evolutivo do algoritmo, AltIvo utiliza o número máximo de gerações como critério de parada. Nos testes realizados neste trabalho, foram consideradas 100 gerações como valor limite. Para trabalhos futuros, planeja-se a inclusão de novos critérios de parada, como por exemplo, um número máximo de gerações sem evolução na melhor solução.

5.9 Trabalhos Relacionados

Algoritmos Evolutivos têm sido um tema em ascensão nos últimos anos. Isto se deve principalmente a sua eficiência para lidar com problemas que envolvam um amplo espaço de busca. Essa característica, em particular, tem motivado o seu uso para as mais diversas aplicações em mineração de dados. Entre os possíveis exemplos, a indução de modelos preditivos baseados em árvores de decisão mostra-se especialmente útil para os casos onde se deseja obter uma boa relação entre desempenho preditivo e compreensibilidade do modelo.

Árvore de regressão, por exemplo, é um tipo especial de árvore de decisão onde o valor armazenado no atributo alvo é um valor contínuo. Esse valor contínuo é a média dos valores para o atributo alvo de todas as instâncias que atingiram um determinado nodo.

Levando em consideração os algoritmos evolutivos para indução de árvores de regressão, recentes abordagens têm sido verificadas na literatura e então analisados seus respectivos métodos para o tratamento de valores ausentes.

Fan e Gray [FG05] propõem um algoritmo para indução de árvores de regressão chamado TARGET. Essa solução foi avaliada sob diferentes tipos de *datasets* (reais e simulados), e posteriormente comparada aos algoritmos CART, Bayesian CART e Random Forest. Resultados da análise comparativa mostram que o algoritmo TARGET apresenta um desempenho superior aos dois primeiros, e inferior ao Random Forest. No entanto, nenhuma menção é feita sobre a forma com que o algoritmo trata dados ausentes.

GRT (Global Induction of Regression Tree) [KC10] é uma outra alternativa para indução de árvores de regressão cujo processo de avaliação foi baseado na análise de *datasets* sintéticos e reais. Os resultados da avaliação realizada mostram que o algoritmo GRT apresenta um desempenho preditivo competitivo ao RepTree com a vantagem de apresentar uma árvore menos complexa do que as produzidas por esse. Com relação ao tratamento de dados ausentes, não foi encontrada qualquer menção sobre o método adotado. Pelo que se tem observado, os *datasets* utilizados nesse estudo são prioritariamente completos. Essa mesma situação também é verificada em um trabalho mais recente de ambos os autores [CK13] onde árvores de regressão e árvores modelo são induzidas.

Assim como árvores de regressão, árvores modelo são estruturas hierárquicas para predição de variáveis dependentes contínuas. A única diferença entre ambas é que, para a última, cada nodo é substituído por um plano de regressão em vez de um valor constante.

GPMCC [PE08] é um algoritmo evolutivo projetado para evoluir árvores modelos com modelos não-lineares em seus nodos folhas. GPMCC foi comparado com outras duas abordagens, utilizando para tanto, *datasets* obtidos do repositório da UCI e outros criados artificialmente. Pelo que se sabe, com exceção do Auto-mpg, nenhum outro *dataset* empregado possui dados ausentes. No trabalho também não é feita nenhuma menção sobre o tema.

E-Motion [BBR+10], por sua vez, é um algoritmo multiobjetivo para indução de árvores modelo que foi projetado inicialmente para predição de esforço em projetos de *software*. Para lidar com dados ausentes, E-Motion usa a técnica de *surrogate splitting* que consiste em encontrar um outro atributo para fazer o particionamento da ávore no lugar daquele com dado ausente. Durante o treinamento, E-Motion usa o valor da classe do atributo "substituto" na crença de que este seja o atributo mais correlacionado com aquele usado para o particionamento. Quando o procedimento de particionamento termina, todos os valores ausentes são substituídos pelos valores médios dos atributos correspondentes dos exemplos do conjunto de treino que chegaram até o nodo folha. Essa mesma técnica é utilizada pelo algoritmo M5 [Q+92].

5.10 Considerações do Capítulo

Neste capítulo foi apresentado AltIvo, um novo algoritmo evolutivo para indução de árvores de regressão. No intuito de otimizar o desempenho preditivo, AltIvo tem agregado em sua implementação uma abordagem multi-estratégia para o tratamento de dados ausentes. Essa abordagem consiste na inclusão de um módulo de imputação dentro do ciclo evolutivo do algoritmo. Neste módulo, são considerados os métodos *Majority*, k-NN e *Hot-Deck*.

Uma vez definidos os métodos, as estimativas são realizadas sob os dados do conjunto de treino, gerando assim, uma versão imputada desse mesmo conjunto. Essa nova versão, por sua vez, é utilizada como base para escolha dos atributos de cada árvore, assim como, para a estimativa dos valores faltantes nos conjuntos de validação e de teste. Esses conjuntos, como se sabe, são essenciais para o cálculo das medidas de avaliação do indivíduo, e consequentemente para a qualificação desse como solução do problema em questão. Ao final, esse processo é repetido para cada ciclo do método de validação cruzada de 10 *folds*.

A ideia por trás desta solução baseia-se no princípio de que diferentes problemas exigem diferentes soluções. Assim, acredita-se ter um diferencial em relação a algoritmos tradicionais, os quais na maioria dos casos utilizam-se de uma única estratégia para lidar com o tratamento de dados ausentes.

Finalmente, uma revisão da literatura é apresentada, abordando o uso de algoritmos evolutivos para indução de árvores de regressão e árvores modelo. Com exceção do algoritmo E-Motion [BBR+10], nenhum outro trabalho faz menção explícita sobre a estratégia utilizada para o tratamento de dados ausentes durante o ciclo evolutivo do algoritmo.

6. ANÁLISE EXPERIMENTAL

O uso de testes estatísticos tem sido uma prática comum para validação de experimentos e comparação de resultados. Em razão de prover maior garantia sobre a validade dos testes realizados, apresenta-se nesse capítulo a análise experimental conduzida para avaliação do desempenho preditivo do algoritmo AltIvo. Para melhor organização, um fluxo de atividades é ilustrado na Figura 6.1.

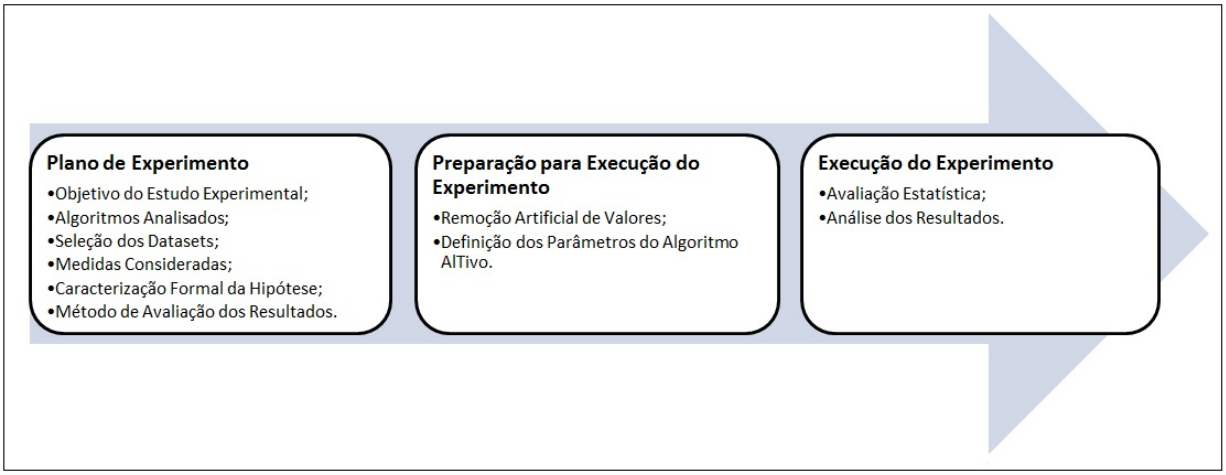


Figura 6.1 – Fluxo de atividades da análise experimental.

Uma vez realizada a análise experimental, também é avaliada a compreensibilidade dos modelos gerados por AltIvo, comparando-os com aqueles obtidos por outros algoritmos para indução de árvores de regressão. Para finalizar o capítulo, os resultados gerados são apresentados a partir de um estudo na área de saúde bucal envolvendo dados ausentes.

6.1 Plano de Experimento

6.1.1 Objetivo do Estudo Experimental

O objetivo deste estudo experimental é avaliar o desempenho preditivo do algoritmo AltIvo, especialmente sob bases de dados com altos percentuais de valores ausentes. Ao final, pretende-se comparar seus resultados com aqueles obtidos por outros algoritmos de regressão.

6.1.2 Algoritmos Analisados

Nesta avaliação, foram analisados 6 tradicionais algoritmos de regressão, comumente utilizados em aplicações de mineração de dados. O critério de escolha para seleção dos algoritmos é baseado em uma avaliação empírica, buscando-se reunir diferentes métodos de regressão. Esses algoritmos são ilustrados na Tabela 6.1.

Tabela 6.1 – Algoritmos utilizados na análise experimental.

Algoritmo	Método de Regressão	Compreensibilidade
M5P	Árvore Modelo	Caixa-transparente [1]
RepTree	Árvore de Regressão	Caixa-transparente [1]
SMOReg	Máquina de Vetores de Suporte	Caixa-preta [2]
IBK	k-Vizinhos mais Próximos	Caixa-transparente [1]
LinearRegression	Regressão Linear	Caixa-transparente [1]
MultiLayerPerecptron	Redes Neurais	Caixa-preta [2]

1 Caracterizado por revelar a estrutura do modelo de um modo compreensível [WF11].
2 O detalhamento do modelo é efetivamente incompreensível [WF11].

6.1.3 Seleção dos *Datasets*

Para análise dos algoritmos, foram utilizados 10 *datasets* de regressão públicos coletados no repositório da UCI (University of California, Irvine). Essas bases foram posteriormente preparadas para manterem 15 diferentes níveis de dados ausentes. Ao total, foram produzidas 150 variações das versões completas. Para finalizar, avaliamos os resultados obtidos a partir de um estudo real na área de saúde bucal envolvendo dados ausentes. As características das versões originais dos *datasets* analisados são ilustradas na Tabela 6.2.

6.1.4 Medidas Consideradas

Para avaliar o desempenho preditivo de cada algoritmo foi escolhida a medida RMSE (*Root Mean Square Error*). Essa medida foi utilizada na análise lexicográfica do algoritmo AltIvo e tem sido comumente empregada em aplicações de mineração de dados. Maiores informações sobre a medida são fornecidas no Capítulo 5.

Tabela 6.2 – *Datasets* considerados no experimento.

Datasets	Instâncias	Atributos	Tipagem
Pyrim	74	27	Numérico
Cloud	108	6	Numérico,Nominal
Veteran	137	7	Numérico,Nominal
Wisconsin	194	32	Numérico
CPU	209	7	Numérico,Nominal
BreastTumor	286	9	Numérico,Nominal
Sensory	576	11	Nominal
Strike	625	6	Numérico,Nominal
Servo	168	5	Numérico,Nominal
ForestFires	517	13	Numérico,Nominal
Odonto	490	9	Numérico,Nominal

6.1.5 Caracterização Formal da Hipótese

Conforme exposto neste documento, a incidência de altos percentuais de dados ausentes tem afetado o desempenho preditivo de diversos algoritmos na área de AM. Para lidar com esse problema, diferentes tipos de estratégias têm sido propostas na literatura. Neste trabalho, é proposto um novo algoritmo evolutivo para indução de árvores de regressão, cujo método de tratamento de dados ausentes é baseado no conceito de imputação múltipla. Desta forma, os algoritmos selecionados são comparados, levantando-se inicialmente as seguintes hipóteses:

1. **Hipótese Nula,** H_0**:** Não há diferença significativa entre os resultados obtidos por todos algoritmos. Neste caso, infere-se que os desempenhos obtidos são equivalentes ($\Delta RMSE_{AltIvo} = \Delta RMSE_{M5P} = \Delta RMSE_{RepTree} = \Delta RMSE_{SMOReg} = \Delta RMSE_{IBK} = \Delta RMSE_{LinearRegression} = \Delta RMSE_{MultiLayerPerceptron}$).

2. **Hipótese Alternativa,** H_1**:** Há pelos menos uma diferença significativa entre os resultados obtidos por todos algoritmos.

Caso a hipótese nula seja refutada em favor de H_1, é aplicado um pós-teste para comparação dos algoritmos par a par. Ao final desta avaliação, deseja-se saber se o desempenho preditivo do algoritmo AltIvo, o qual incorpora múltiplas estratégias para lidar com dados ausentes é IGUAL ao desempenho preditivo dos algoritmos que implementam apenas uma estratégia.

Maiores detalhes sobre a aplicação de métodos estatísticos são fornecidos na próxima seção.

6.1.6 Método de Avaliação dos Resultados

Em razão de verificar a significância estatística das diferenças dos resultados de cada algoritmo, os testes de Friedman [Fri37, Fri40] e Nemenyi [Nem63] são aplicados, conforme abordagem proposta por Demsar [Dem06]. De acordo com o referido autor, esses testes são mais adequados para comparação de múltiplos algoritmos sob várias bases de dados, ao contrário do Teste-t, que se mostra melhor para comparação de dois algoritmos. Nesse trabalho, 7 diferentes algoritmos são aavaliados (incluindo nossa solução) para problemas de regressão. Desde então, optou-se pela aplicação dos dois métodos inicialmente mencionados.

O teste de Friedman é um teste não paramétrico, que utiliza um *ranking* dos algoritmos para cada base de dados, separadamente. O algoritmo que obtiver melhor desempenho em uma base de dados recebe a posição 1 do *ranking*, o segundo melhor recebe a posição 2 e assim por diante. Considerando um total de N bases de dados, o teste de Friedman compara o *ranking* médio dos algoritmos, dado por $R_j = \frac{1}{N}\sum_i r_i^j$. Sob a hipótese nula, que considera que todos os algoritmos sendo comparados tem seus valores de R_j iguais, a estatística de Friedman, dada pela Equação 6.1, é distribuída de acordo com X_F^2 com k-1 graus de liberdade [Dem06].

$$X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{6.1}$$

Contudo, Iman e Davenport [ID80] argumentam que a estatística de Friedman X_F^2 é muito conservadora, propondo assim, uma melhor estatística que é distribuída de acordo com k-1 e (k-1) (N-1) graus de liberdade. Esta nova estatística derivada de Friedman é descrita na Equação 6.2.

$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2} \tag{6.2}$$

Desta forma, se a hipótese nula é rejeitada (ou seja, há diferença estatística entre os N algoritmos avaliados), o teste de Nemenyi é aplicado para verificar entre quais algoritmos ocorre essa diferença. Assim, cada par de algoritmos é comparado, e a significância estatística entre ambos é verificada caso seus respectivos *rankings* médios diferirem, no mínimo, de um valor crítico CD, conforme ilustrado na Equação 6.3. Os valores de $q\alpha$ são obtidos da tabela 6.3.

$$CD = q\alpha\sqrt{\frac{k(k+1)}{6N}} \tag{6.3}$$

Tabela 6.3 – Valores críticos para o teste de Nemenyi bicaudal [Dem06].

Classificadores	2	3	4	5	6	7	8	9	10
$q_{0,05}$	1,960	2,343	2,569	2,728	2,850	2,949	3,031	3,102	3,164

6.2 Preparação para Execução do Experimento

Uma vez definido o plano de experimento, partiu-se para a preparação dos *datasets* e definição dos parâmetros do algoritmo AltIvo. Maiores detalhes sobre estas duas etapas são fornecidos nas próximas seções.

6.2.1 Remoção Artificial de Valores

Com o objetivo de simular diferentes níveis de dados ausentes, foi desenvolvido um programa em Java, usando a biblioteca jxl (Java Excel). Essa biblioteca foi escolhida em razão de manipular arquivos CSV (*Comma-Separated Values*), formato aceito pela maior parte das ferramentas de aprendizado de máquina. A partir do *dataset* original, foram derivados mais 15 planilhas com percentuais de dados ausentes em um intervalo de 1% a 70%. Estes percentuais foram gerados de maneira randômica conforme o método *Math.Random()*.

No intuito de obter um maior controle sobre o processo de remoção de valores, optou-se por selecionar as três variáveis independentes mais correlacionadas com o atributo alvo. Desta forma, garantiu-se que independente do percentual, os atributos seriam sempre os mesmos. O filtro supervisionado *CfsSubsetEval* com o método de busca *BestFirst* foram empregados para seleção dos atributos mais correlacionados com o alvo. Ambos os métodos são disponíveis na ferramenta WEKA 3.7.0.

Depois de selecionados, os índices desses atributos eram passados por parâmetro para o programa em Java. Para melhor compreensão, o pseudocódigo produzido para remoção de valores é ilustrado abaixo.

Como é possível verificar no Algoritmo 6.1, a primeira etapa do processo de remoção de valores consiste na passagem por parâmetro do nome do *dataset*, dos três índices dos atributos considerados para remoção, e finalmente, do número *N* de percentuais produzidos. Para produção desses diferentes percentuais, um arquivo Excel era gerado, tendo como primeira planilha a versão completa do *dataset*.

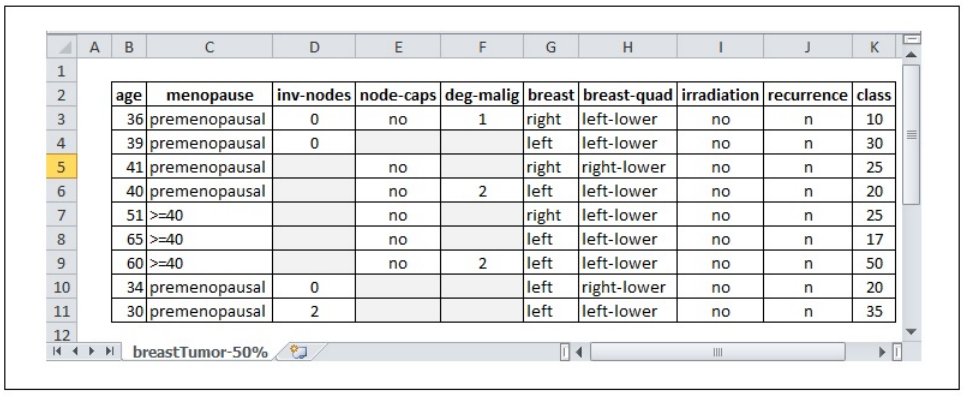
A partir desta primeira planilha, diferentes percentuais de dados ausentes eram gerados incrementalmente, incorporando-se para tanto o percentual de dados ausentes da

Algoritmo 6.1 Pseudocódigo para remoção artificial de valores.

```
1: Receber parâmetros
2: Obter planilha completa
3: Repita
4:     Criar nova planilha
5:     Repita
6:         Faça
7:             Gerar referência LxC a ser removida
8:             Verifica existência da referência LxC
9:         Equanto (existência=verdadeira)
10:        Remover valores da referência LxC na planilha
11:    Até(atingir o percentual de dados ausentes para a coluna C )
12: Até(Número N de planilhas ser atingido)
```

planilha anterior em uma nova até que o número N de planilhas fosse alcançado. Ao final, as referências necessárias eram produzidas de acordo com o número de campos (referências) calculados(as) para o respectivo percentual. Nesse processo, cada referência era gerada através da concatenação dos valores de linha e coluna. Os valores das linhas, por sua vez, eram obtidos a partir da aplicação da função *Math.Random()* do Java, e as colunas através dos valores dos índices passados por parâmetro no algoritmo.

Para evitar a repetição de uma referência pré-existente, criamos uma matriz auxiliar com as mesmas dimensões de linha e coluna do *dataset* analisado. Essa matriz foi totalmente populada com a constante 99999 para indicar a disponibilidade da referência a ser removida. O valor da referência (linha *x* coluna) era utilizado para indicar a posição da matriz a ser consultada. Desta forma, se o valor da referência na matriz fosse igual a 99999, a remoção de valor da planilha era realizada. Caso contrário, um novo valor era gerado até que o resultado do teste fosse positivo. A Figura 6.2 fornece uma visão geral da planilha após o processo de remoção de valores.



age	menopause	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiation	recurrence	class
36	premenopausal	0	no	1	right	left-lower	no	n	10
39	premenopausal	0			left	left-lower	no	n	30
41	premenopausal		no		right	right-lower	no	n	25
40	premenopausal		no	2	left	left-lower	no	n	20
51	>=40		no		right	left-lower	no	n	25
65	>=40		no		left	left-lower	no	n	17
60	>=40		no	2	left	left-lower	no	n	50
34	premenopausal	0			left	right-lower	no	n	20
30	premenopausal	2			left	left-lower	no	n	35

Figura 6.2 – Remoção artificial de valores.

6.2.2 Definição dos Parâmetros do Algoritmo AltIvo

Os algoritmos M5P, RepTree, LinearRegression, MultiLayerPerceptron, SMOReg e IBK foram executados com os valores *default* para seus respectivos parâmetros. Para definição dos parâmetros de AltIvo, foram assumidos valores com base no conhecimento prévio do autor. Esses parâmetros são ilustrados na Tabela 6.4.

Tabela 6.4 – Parâmetros utilizados pelo algoritmo AltIvo nos experimentos.

Parâmetro	Valor
Número de indivíduos da população	100
Número de execuções	10
Número de máximo de gerações	100
Profundidade das árvores geradas na população inicial	2
Função de *fitness* - limiar inicial do RMSE	5%
Função de *fitness* - limiar inicial do MAE	5%
Função de *fitness* - limiar tamanho da árvore	20%
Tratamento de dados ausentes	*Majority*, k-NN, Hot-Deck
Número de indivíduos do torneio	3
Tipo de *fitness*	Lexicográfico
Taxa de elitismo	1%
Taxa de crossover	90%
Taxa de mutação	5%
Poda	não
Critério de Parada	Número máximo de gerações

6.3 Execução do Experimento

6.3.1 Avaliação Estatística

A avaliação estatística é feita em diferentes percentuais (de 0% a 70%). Para uma melhor compreensão, apresentamos os ranqueamentos dos algoritmos para cada percentual de dados ausentes, e os resultados dos testes de Friedman e Nemenyi, conforme ilustrado nas Figuras 6.3 e 6.4, respectivamente.

Ranking para 0% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	LinearRegression	2,6
2	AltIvo	3,3
3	SMOReg	3,6
4	RepTree	3,9
5	M5P	4,1
6	IBK	5,0
7	MultiLayerPerceptron	5,5

Ranking para 1% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	LinearRegression	2,4
2	AltIvo	3,2
3	SMOReg	3,4
4	RepTree	4,2
5	M5P	4,3
6	IBK	5,0
7	MultiLayerPerceptron	5,5

Ranking para 5% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	LinearRegression	2,3
2	AltIvo	2,8
3	SMOReg	3,3
4	RepTree	3,9
5	M5P	4,3
6	IBK	5,5
7	MultiLayerPerceptron	5,9

Ranking para 10% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,5
2	LinearRegression	2,5
3	SMOReg	3,0
4	M5P	4,1
5	RepTree	4,1
6	IBK	5,7
7	MultiLayerPerceptron	6,1

Ranking para 15% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,4
2	LinearRegression	2,5
3	SMOReg	3,2
4	M5P	4,1
5	RepTree	4,2
6	IBK	5,7
7	MultiLayerPerceptron	5,9

Ranking para 20% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,4
2	LinearRegression	2,4
3	SMOReg	3,2
4	M5P	4,0
5	RepTree	4,4
6	IBK	5,7
7	MultiLayerPerceptron	5,9

Ranking para 25% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	LinearRegression	2,4
2	AltIvo	2,7
3	SMOReg	3,4
4	M5P	3,6
5	RepTree	4,3
6	IBK	5,7
7	MultiLayerPerceptron	5,9

Ranking para 30% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,3
2	LinearRegression	2,3
3	SMOReg	3,2
4	M5P	4,0
5	RepTree	4,4
6	IBK	5,9
7	MultiLayerPerceptron	5,9

Ranking para 35% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	LinearRegression	2,3
2	AltIvo	2,5
3	SMOReg	3,2
4	M5P	3,9
5	RepTree	4,5
6	IBK	5,8
7	MultiLayerPerceptron	5,8

Ranking para 40% de dados ausentes.

Posição	Algoritmo	Ranking Médio
1	LinearRegression	2,0
2	AltIvo	2,7
3	SMOReg	3,2
4	M5P	3,7
5	RepTree	4,2
6	IBK	5,8
7	MultiLayerPerceptron	6,4

Ranking para 45% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,3
2	LinearRegression	2,6
3	SMOReg	3,5
4	M5P	3,5
5	RepTree	4,5
6	IBK	5.7
7	MultiLayerPerceptron	5,9

Ranking para 50% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,1
2	LinearRegression	2,5
3	SMOReg	3,5
4	M5P	3,8
5	RepTree	4,2
6	IBK	5,5
7	MultiLayerPerceptron	6,4

Ranking para 55% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,1
2	LinearRegression	2,5
3	M5P	3,0
4	SMOReg	3,8
5	RepTree	4,8
6	IBK	5,4
7	MultiLayerPerceptron	6,4

Ranking para 60% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,1
2	LinearRegression	3,0
3	M5P	3,3
4	SMOReg	3,6
5	RepTree	4,5
6	IBK	5,5
7	MultiLayerPerceptron	6,0

Ranking para 65% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,0
2	LinearRegression	2,9
3	M5P	3,2
4	SMOReg	3,5
5	RepTree	4,2
6	IBK	5,8
7	MultiLayerPerceptron	6,4

Ranking para 70% de dados ausentes

Posição	Algoritmo	Ranking Médio
1	AltIvo	2,0
2	LinearRegression	2,9
3	M5P	3,1
4	SMOReg	3,8
5	RepTree	4,1
6	IBK	5,8
7	MultiLayerPerceptron	6,3

Figura 6.3 – Ranqueamentos de 0% a 70%.

Percentual	Teste de Friedman				Teste de Nemenyi (Algoritmos significativamente inferiores ao AltIvo)						
	X_F^2	F_F	F(6,54)	Rejeita H_0	$CD_{0.05}$	M5P	RepTree	SMOReg	LinearRegression	IBK	MultiLayerPerceptron
0	12,59	2,39	2,27	sim	2,84	não	não	não	não	não	não
1	14,87	2,96	2,27	sim	2,84	não	não	não	não	não	não
5	23,10	5,63	2,27	sim	2,84	não	não	não	não	sim	sim
10	27,47	7,60	2,27	sim	2,84	não	não	não	não	sim	sim
15	25,71	6,75	2,27	sim	2,84	não	não	não	não	sim	sim
20	26,61	7,17	2,27	sim	2,84	não	não	não	não	sim	sim
25	24,34	6,14	2,27	sim	2,84	não	não	não	não	sim	sim
30	29,57	8,74	2,27	sim	2,84	não	não	não	não	sim	sim
35	26,83	7,27	2,27	sim	2,84	não	não	não	não	sim	sim
40	33,13	11,09	2,27	sim	2,84	não	não	não	não	sim	sim
45	25,93	6,84	2,27	sim	2,84	não	não	não	não	sim	sim
50	30,43	9,26	2,27	sim	2,84	não	não	não	não	sim	sim
55	32,70	10,78	2,27	sim	2,84	não	sim	não	não	sim	sim
60	25,20	6,51	2,27	sim	2,84	não	não	não	não	sim	sim
65	32,44	10,59	2,27	sim	2,84	não	não	não	não	sim	sim
70	31,28	9,80	2,27	sim	2,84	não	não	não	não	sim	sim

Figura 6.4 – Resultados dos testes de Friedman e Nemenyi.

6.3.2 Análise dos Resultados

Com base nos resultados apresentados na seção anterior, é possível verificar que AltIvo tem se mantido sempre entre os dois melhores algoritmos, sendo 9 vezes ranqueado em primeiro lugar em um total de 16 avaliações (0% a 70%). Assim, AltIvo tem se mostrado superior especialmente para a faixa de 45% a 70% de dados ausentes, embora a diferença verificada não seja estatisticamente significativa em relação aos algoritmos LinearRegression, M5P, RepTree e SMOReg. Essa diferença, contudo, é relevante em comparação aos algoritmos IBK e MultiLayerPerceptron a partir dos 5% de dados ausentes a uma significância de 95%.

LinearRegression, por sua vez, mostra-se superior (não significativamente) em 6 avaliações, especialmente na faixa de 0% a 40% onde AltIvo é o segundo colocado. Outro aspecto a ser levado em consideração é a oscilação de desempenho entre os resultados gerados pelos algoritmos RepTRee, M5P e SMOReg que tem ocorrido em diferentes faixas de percentuais. Para a faixa de 0% a 20% de dados ausentes, por exemplo, os algoritmos SMOReg e RepTree estão melhores colocados, assumindo a terceira e quarta posição no *ranking*. Este ranqueamento, contudo, é alterado a partir dos 25% e 50% de dados ausentes, quando o M5P supera RepTRee e SMOReg respectivamente. MultiLayerPerceptron é o algoritmo com piores resultados, sendo seguido pelo IBK.

No restante da análise, os desempenhos preditivos são avaliados separadamente por *dataset*, considerando para tanto, apenas os algoritmos melhores ranqueados ou com alto grau de compreensibilidade (ver Figura 6.5). De um modo geral, não é possível se identificar uma relação linear entre perda de desempenho preditivo e o aumento do percentual de dados ausentes.

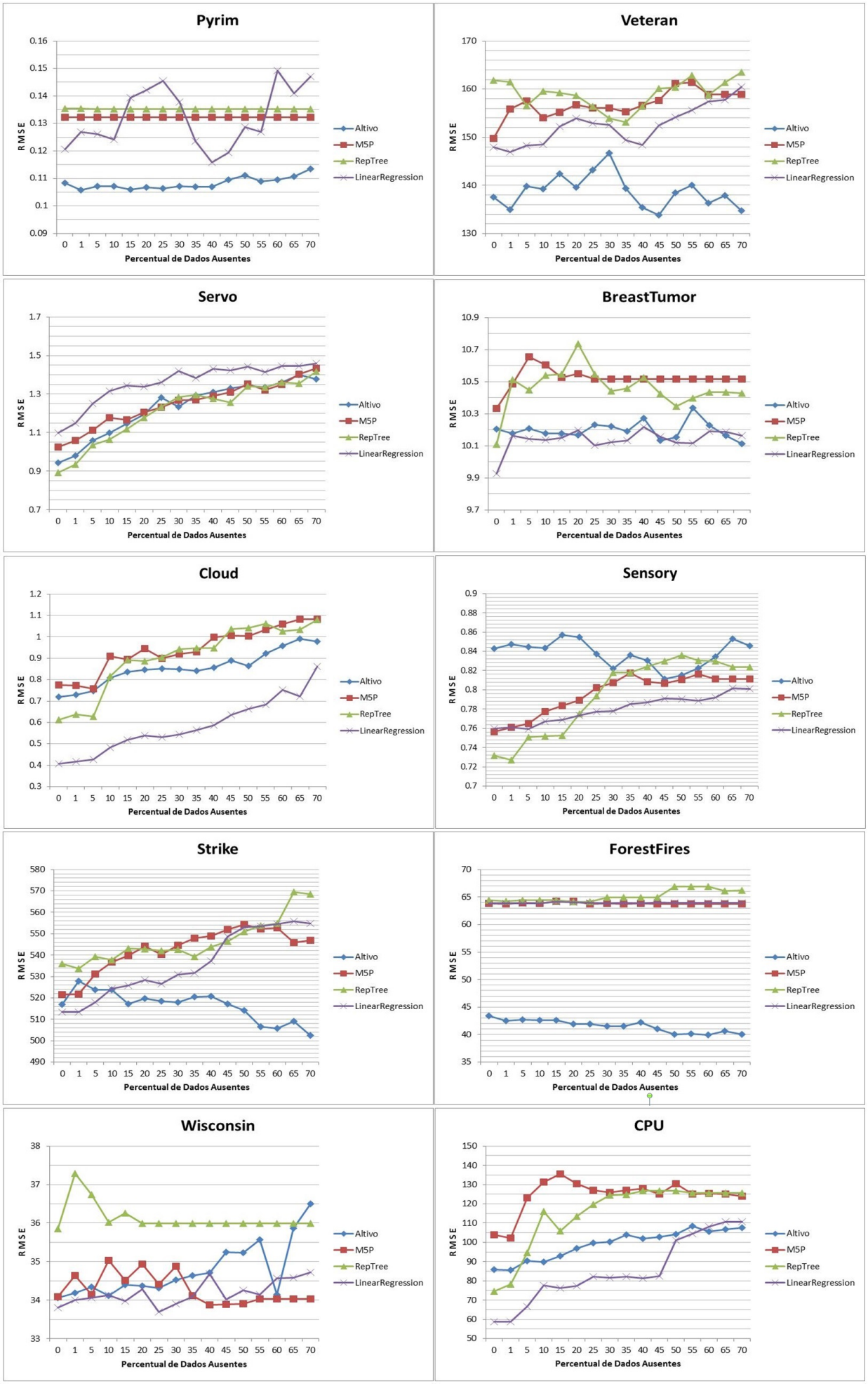


Figura 6.5 – Avaliação do desempenho preditivo por *dataset*.

É possível se constatar sim, uma grande aleatoriedade neste processo, embora, na maior parte dos casos haja uma relação predominantemente crescente na perda de desempenho quando comparado aos resultados dos estados iniciais (0% e 1%). Isto pode ser verificado, mais nitidamente, nas análises realizadas nos *datasets*: Cloud, Servo, CPU, Strike e Sensory.

Com relação ao desempenho dos algoritmos, há um certo equilíbrio entre os resultados obtidos por AltIvo e LinearRegression, que na maior parte dos casos são superiores àqueles obtidos pelos M5P e RepTree (exceto no *dataset* Servo, onde o LinearRegression apresenta o pior desempenho). Enquanto o algoritmo AltIvo destaca-se nos *datasets* Strike, ForestFires, Veteran e Pyrim, LinearRegression apresenta melhores resultados nos datasets Cloud, CPU e Sensory. Para os demais *datasets* (Wisconsin, BreastTumor e Servo), não é possível evidenciar qualquer superioridade, havendo assim, uma grande oscilação de ganho e perda entre ambos.

Nos Apêndices A e B, os resultados são apresentados na forma de diagramas críticos. Nesse tipo de diagrama, os relacionamentos entre os algoritmos são destacados por percentual onde a diferença do desempenho preditivo não é estatisticamente significativa. Com é possível verificar, AltIvo, LinearRegression e M5P são os algoritmos com o maior número de casos onde a diferença é estatisticamente significativa, o que acontece principalmente em relação aos algoritmos IBK e MultiLayerPerceptron.

Para avaliação da compreensibilidade dos modelos, a análise realizada é focada sob os algoritmos que trabalham com a indução de árvores. Essa forma de representação tem sido comumente aplicada em outras áreas de domínio, principalmente devido ao seu alto grau de comprensibilidade. Deste modo, os resultados gerados pelos algoritmos AltIvo, M5P e RepTree são analisados, sendo apresentadps na Figura 6.6.

Para fins de comparação, o número total de nodos gerados pelos modelos induzidos em cada algoritmo é considerado, e comparado em função do percentual de dados ausentes. De modo semelhante a análise anterior, não é possível identificar uma relação direta entre o aumento do número de nodos e o aumento do percentual de dados ausentes. Dentre os algoritmos avaliados, RepTRee é o que apresenta a maior variação no tamanho das árvores. Esse comportamento pode ser verificado em praticamente todos os *datasets*, com a exceção do Pyrim, onde o tamanho das árvores geradas pelo RepTree tem se mantido constantemente com o tamanho 1. Em contrapartida, AltIvo é aquele com o comportamento mais estável, independentemente do percentual de dados ausentes analisado.

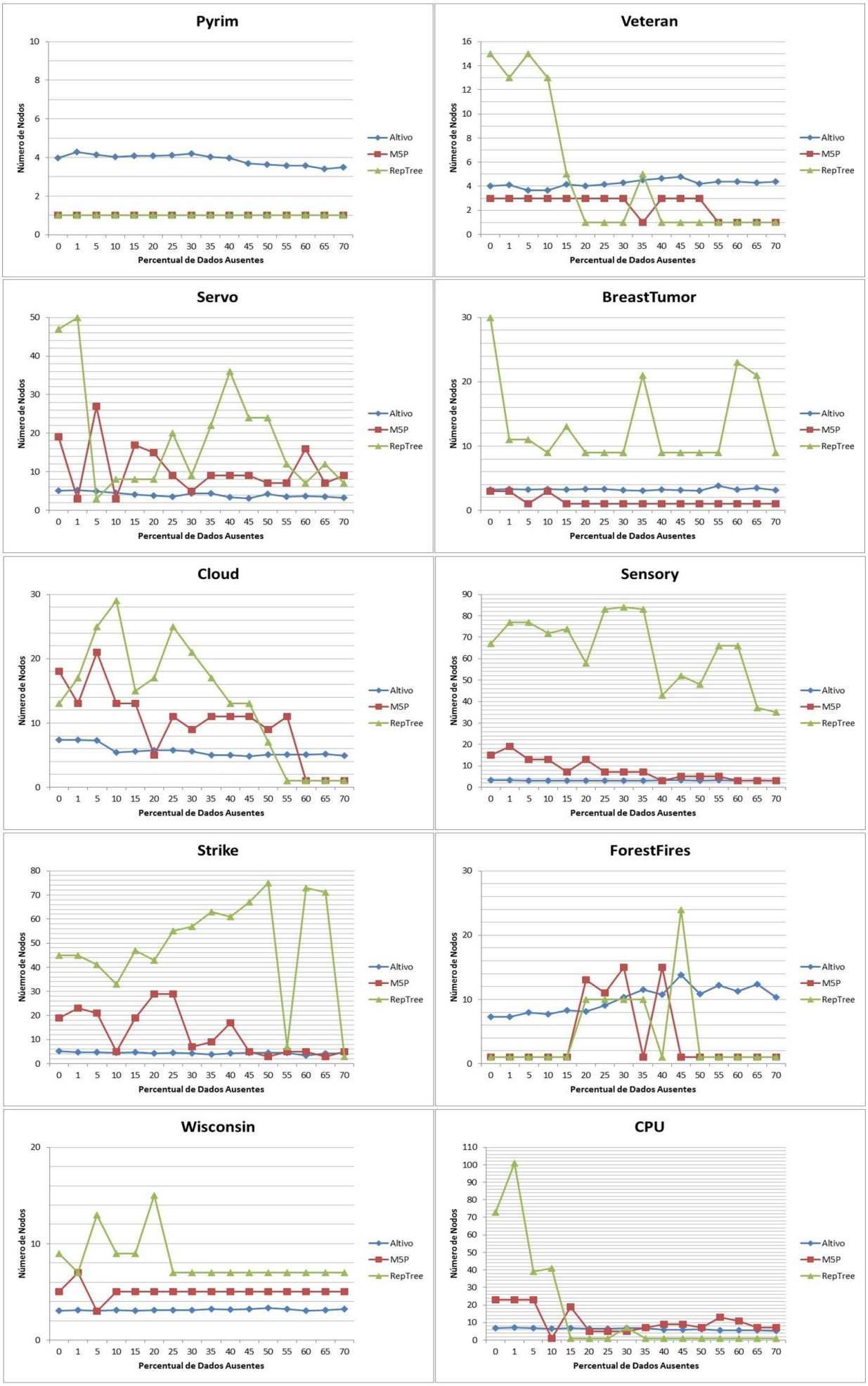


Figura 6.6 – Avaliação da compreensibilidade dos modelos por *dataset.*

6.4 Uma Avaliação Empírica sob Dados de Saúde Bucal

Nesta seção, são relatados os resultados obtidos a partir de uma avaliação empírica envolvendo dados de saúde bucal. Estes dados foram apreendidos durante o período de 2009 a 2011 junto ao Centro de Extensão Universitária Vila Fátima.

O Centro de Extensão Universitária Vila Fátima é uma unidade acadêmica descentralizada vinculada à PUCRS (Pontifícia Universidade Católica do Rio Grande do Sul) e que presta serviços nas áreas de assistência jurídica, assistência social, saúde e educação a uma população de baixa renda situada em uma vila da zona leste de Porto Alegre-RS.

De acordo com os levantamentos feitos durante a pesquisa de [BBFR13], a base de dados apreendida é formada basicamente por dados de anamnese, diagnóstico, odontograma e diversos tipos de tratamentos. Como principal característica desta base de dados, verificou-se um alto percentual de dados ausentes em diferentes atributos do *datasets*.

Ao final dessa pesquisa, foram coletadas 1.594 fichas odontológicas, abrangendo dados de 1.500 pacientes. No entanto, para o presente estudo, temos considerado apenas 490 destas fichas, que são aquelas com algum preenchimento no odontograma. Esta escolha é justificada mediante a imprescindibilidade do odontograma na predição da quantidade de cáries (atributo a ser predito). Na Tabela 6.5 apresentamos maiores informações sobre o *dataset* utilizado.

Tabela 6.5 – *Dataset* de saúde bucal.

Atributo	Tipagem	Valores	Dados Ausentes%
idade	Numérico	0-92	18%
diabetes	Nominal	S,N	0%
hipertensão	Nominal	S,N	0%
foiaodentista	Nominal	S,N	17%
usacremedental	Nominal	S,N	19%
utilizafluoreto	Nominal	S,N	23%
frequenciafluoreto	Nominal	0,1,2,3,4,5	23%
consumoaçucar	Nominal	3,5,7	33%
quantdentescariados	Numérico	0-32	0%

De modo semelhante a análise experimental, dados de RMSE, MAE e Tamanho médio da árvore foram coletados, utilizando-se para tanto, o tradicional método de validação cruzada de 10 folds. Para o algoritmo AltIvo, no entanto, optou-se por calcular a média de 10 execuções, o que tem se mostrado apropriado para abordagens não-determinísticas.

Com base nos resultados ilustrados na Tabela 6.6, é possível verificar uma certa similaridade em relação aos desempenhos obtidos na análise experimental. Neste sentido, AltIvo tem se mantido como a melhor solução em termos de RMSE, e a segunda melhor

Tabela 6.6 – Resultados do estudo em saúde bucal.

Algoritmo	RMSE	MAE	Tamanho médio da Árvore
AltIvo	4,82	3,37	4
M5Ps	4,98	3,43	5
RepTree	5,00	3,46	1
LinearRegression	4,95	3,42	—
SMOReg	5,20	3,21	—
IBK	6,40	4,20	—
MultiLayerPercepTron	7,10	4,34	—

pelo MAE. LinearRegression e M5P são os próximos no ranqueamento de desempenho preditivo. IBK e MultiLayerPerceptron são os piores qualificados para ambas medidas.

Levando em consideração o grau de interpretabilidade dos modelos, não é possível identificar uma maior oscilação entre os tamanhos de árvores geradas pelos algoritmos RepTRee e M5P. Como é possível se verificar, estes tamanhos têm se mantido na faixa de 1 a 5 nodos, o que não ocorreu na maior parte dos casos na análise experimental. Contudo, AltIvo continua apresentando uma boa relação entre interpretabilidade e desempenho preditivo.

6.5 Considerações do Capítulo

Neste capítulo, foram apresentadas diferentes análises para avaliar o desempenho do algoritmo AltIvo. Inicialmente, foi proposto a realização de um estudo experimental, considerando para tanto, um fluxo de atividades compreendido pelo planejamento do experimento, preparação, e execução do mesmo.

Em razão de realizar uma análise comparativa, foram utilizados os testes de Friedman e Nemenyi para avaliação do algoritmo AltIvo juntamente com outros 6 tradicionais algoritmos de regressão. Nesta avaliação, foram considerados 10 datasets públicos da UCI, modificados artificialmente para manterem diferentes níveis de dados ausentes. Os resultados da análise experimental apontam AltIvo entre os dois melhores algoritmos, sendo superior na faixa de 45% a 70%, embora a diferença entre os resultados não seja estatisticamente significativa em relação aos algoritmos LinearRegression, M5P, RepTree e SMOReg. Contudo, dentre algoritmos avaliados, AltIvo é aquele que apresenta a melhor relação entre desempenho preditivo e compreensibilidade dos modelos. Esses primeiros resultados têm motivado o prosseguimento da pesquisa e a exploração de novas possibilidades para otimização do algoritmo.

Em uma avaliação posterior, o grau de interpretabilidade dos modelos baseados em árvore foram analisados. Os resultados desta análise mostram que os modelos gerados

por AltIvo são mais estáveis em relação a variação do número de nodos das árvores. Em contrapartida, os algoritmos M5P e RepTree tem apresentado uma grande oscilação entre árvores grandes e árvores pequenas.

Na parte final do capítulo, apresentamos os resultados produzidos a partir de uma aplicação real de Saúde Bucal. Assim como já fora verificado na análise experimental, AltIvo tem alcançado os melhores resultados em uma relação de desempenho preditivo e compreensibilidade dos modelos gerados.

7. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Com a evolução da tecnologia de banco de dados nas últimas duas décadas, grandes volumes de dados têm sido acumulados, tornando a sua compreensão um processo extremamente complexo e dependente do uso de ferramentas de análise. Esse problema, contudo, toma maiores dimensões, já que uma parte significativa destes dados tende a ser incompleta.

Na área de AM, por exemplo, os efeitos produzidos por dados ausentes têm gerado sérios problemas na extração de conhecimento, ocultando importantes informações sobre o *dataset*, enviesando resultados e afetando a acurácia dos modelos induzidos [JW04, LLW05, NSZ07, ZWZ10].

De acordo com Acuna e Rodrigues [AR04], taxas menores do que 1% de dados ausentes são consideradas triviais, e 1% a 5% gerenciáveis. No entanto, taxas entre 5% e 15% requerem métodos sofisticados para manipulá-los, e mais do que 15% podem impactar severamente em qualquer tipo de interpretação.

Para lidar com essa situação, pode-se verificar dois maiores tipos de estratégias na literatura. O primeiro tipo compreende a aplicação de métodos de pré-processamento, tais como a imputação, onde um valor plausível é estimado e utilizado para preencher o campo com valor ausente. Essa estratégia, contudo, é tipicamente laboriosa, fazendo com que o usuário invista um grande tempo na escolha de um método ótimo. O segundo tipo de estratégia, por sua vez, consiste em tratar internamente no algoritmo os dados ausentes, o que é igualmente problemático, uma vez que, neste caso, o método de tratamento é invariável. Algoritmos como M5P, SMOReg e LinearRegression, por exemplo, substituem globalmente todos valores ausentes pela média/moda do atributo.

Independentemente do tipo de estratégia escolhida, não há um consenso na literatura sobre um método universal para todos os tipos de cenários envolvendo dados ausentes. Esse fato deve-se possivelmente a variabilidade de características dos *datasets* (ex: mecanismos de distribuição dos dados, correlação dos dados, percentual de dados ausentes, etc.), fazendo com que diferentes conformações de dados exijam diferentes soluções.

Com o objetivo de otimizar o desempenho preditivo em cenários com alta incompletude de dados, propõe-se neste trabalho um novo algoritmo evolutivo para indução de árvores de regressão, agregando em seu ciclo evolutivo uma estratégia baseada no conceito de imputação múltipla. Desta forma, os métodos *Majority*, k-NN e *Hot-Deck* são inicialmente considerados e combinados para estimativa de valores ausentes.

Uma vez definidos os métodos, as estimativas são feitas sob os dados do conjunto de treino, gerando assim, uma versão imputada desse mesmo conjunto. Esta nova versão, por sua vez, é utilizada como base para escolha dos atributos de cada árvore, assim como, para a estimativa dos valores faltantes nos conjuntos de validação e de teste. Esses conjuntos,

como se sabe, são essenciais para o cálculo das medidas de avaliação do indivíduo, e consequentemente para a qualificação desse como solução do problema em questão. Ao final, esse processo é repetido para cada ciclo do método de validação cruzada de 10 *folds*.

Em razão de estabelecer uma análise comparativa com a solução proposta, foram avaliados 6 tradicionais algoritmos de regressão, utilizando-se para tanto, 10 *datasets* públicos da UCI. Esses *datasets* foram modificados artificialmente para manterem diferentes níveis de dados ausentes.

Resultados da análise experimental apontam o AltIvo entre os dois melhores algoritmos, sendo superior na faixa de 45% a 70%, embora a diferença entre os resultados não seja estatisticamente significativa em relação aos algoritmos LinearRegression, M5P, RepTree e SMOReg. Contudo, dentre os algoritmos avaliados, AltIvo é aquele que apresenta a melhor relação entre desempenho preditivo e compreensibilidade dos modelos. Esses primeiros resultados têm motivado o prosseguimento da pesquisa e exploração de novas possibilidades para otimização do algoritmo.

Como trabalhos futuros, pretende-se ampliar o número de testes com o algoritmo AltIvo, buscando-se avaliar os efeitos produzidos pela escolha de diferentes parâmetros de inicialização do algoritmo. Nesse trabalho, essa escolha foi conduzida conforme o conhecimento prévio do autor.

Outro ponto a ser explorado futuramente, diz respeito ao uso de novos métodos para lidar com dados ausentes, o que pontencialmente contribuirá para dar uma maior cobertura no tratamento adequado de diferentes conformações de *datasets*.

Finalmente, entende-se que o estudo de temas como meta-aprendizado possa ser extremamente relevante para determinar em quais cenários AltIvo se mostra melhor ou pior.

REFERÊNCIAS BIBLIOGRÁFICAS

[AL10] Andridge, R. R.; Little, R. J. “A review of hot deck imputation for survey non-response”, *International Statistical Review*, vol. 78–1, 2010, pp. 40–64.

[AR04] Acuna, E.; Rodriguez, C. “The treatment of missing values and its effect on classifier accuracy”. In: *Classification, Clustering, and Data Mining Applications*, Springer, 2004, pp. 639–647.

[BBdC+09] Basgalupp, M. P.; Barros, R. C.; de Carvalho, A. C.; Freitas, A. A.; Ruiz, D. D. “Legal-tree: a lexicographic multi-objective genetic algorithm for decision tree induction”. In: Proceedings of the 2009 ACM symposium on Applied Computing, 2009, pp. 1085–1090.

[BBDCF12] Barros, R. C.; Basgalupp, M. P.; De Carvalho, A.; Freitas, A. A. “A survey of evolutionary algorithms for decision-tree induction”, *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42–3, 2012, pp. 291–312.

[BBFR13] Blomberg, L. C.; Barros, R. C.; Figueiredo, J. A.; Ruiz, D. D. “Applying a kdd process in an oral public health unit at rio grande do sul, brazil”. In: Proceedings of the IADIS e-Society Conference, 2013.

[BBR+10] Barros, R. C.; Basgalupp, M. P.; Ruiz, D. D.; de Carvalho, A. C.; Freitas, A. A. “Evolutionary model tree induction”. In: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1131–1137.

[BHR13] Blomberg, L. C.; Hemerich, D.; Ruiz, D. D. “Evaluating the performance of regression algorithms on datasets with missing data”, *International Journal of Business Intelligence and Data Mining*, vol. 8–2, 2013, pp. 105–131.

[BM03] Batista, G. E.; Monard, M. C. “An analysis of four missing data treatment methods for supervised learning”, *Applied Artificial Intelligence*, vol. 17–5-6, 2003, pp. 519–533.

[BRB11] Barros, R. C.; Ruiz, D. D.; Basgalupp, M. P. “Evolutionary model trees for handling continuous classes in machine learning”, *Information Sciences*, vol. 181–5, 2011, pp. 954–971.

[CK13] Czajkowski, M.; Kretowski, M. “An evolutionary algorithm for global induction of regression and model trees”, *International Journal of Data Mining, Modelling and Management*, vol. 5–3, 2013, pp. 261–276.

[CS06] Chang, W.-D.; Shin, J. “Missing data handling in multi-layer perceptron”. In: Proceedings of the 10th WSEAS international conference on Computers, 2006, pp. 640–645.

[Dem06] Demšar, J. “Statistical comparisons of classifiers over multiple data sets”, *The Journal of Machine Learning Research*, vol. 7, 2006, pp. 1–30.

[FG05] Fan, G.; Gray, J. B. “Regression tree analysis using target”, *Journal of Computational and Graphical Statistics*, vol. 14–1, 2005, pp. 206–218.

[FLGC11] Faceli, K.; Lorena, A. C.; Gama, J.; Carvalho, A. C. “Inteligência Artificial: uma abordagem de aprendizado de máquina”. LTC, 2011.

[FOW66] Fogel, L. J.; Owens, A. J.; Walsh, M. J. “Artificial intelligence through simulated evolution”. John Wiley Sons, 1966.

[Fre02] Freitas, A. A. “Data mining and knowledge discovery with evolutionary algorithms”. Springer, 2002.

[Fre04] Freitas, A. A. “A critical review of multi-objective optimization in data mining: a position paper”, *ACM SIGKDD Explorations Newsletter*, vol. 6–2, 2004, pp. 77–86.

[Fre08] Freitas, A. A. “A review of evolutionary algorithms for data mining”. In: *Soft Computing for Knowledge Discovery and Data Mining*, Springer, 2008, pp. 79–111.

[Fri37] Friedman, M. “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”, *Journal of the American Statistical Association*, vol. 32–200, 1937, pp. 675–701.

[Fri40] Friedman, M. “A comparison of alternative tests of significance for the problem of m rankings”, *The Annals of Mathematical Statistics*, vol. 11–1, 1940, pp. 86–92.

[GDG08] Ghosh, A.; Dehuri, S.; Ghosh, S. “Multi-objective evolutionary algorithms for knowledge discovery from databases”. Springer, 2008, vol. 98.

[HKP11] Han, J.; Kamber, M.; Pei, J. “Data Mining: Concepts and Techniques”. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011, 3rd ed..

[Hol75] Holland, J. H. “Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.” U Michigan Press, 1975.

[ID80] Iman, R. L.; Davenport, J. M. “Approximations of the critical region of the Friedman statistic”, *Communications in Statistics - Theory and Methods*, vol. 9–6, Jan 1980, pp. 571–595.

[JMGL+10] Jerez, J. M.; Molina, I.; García-Laencina, P. J.; Alba, E.; Ribelles, N.; Martín, M.; Franco, L. “Missing data imputation using statistical and machine learning methods in a real breast cancer problem”, *Artificial intelligence in medicine*, vol. 50–2, 2010, pp. 105–115.

[JW04] Jonsson, P.; Wohlin, C. “An evaluation of k-nearest neighbour imputation using likert data”. In: Software Metrics, 2004: Proceedings of the 10th International Symposium on, 2004, pp. 108–118.

[KBR84] Kononenko, I.; Bratko, I.; Roskar, E. “Experiments in automatic learning of medical diagnostic rules”, Relatório Técnico, Jozef Stefan Institute, Ljubljana, Yugoslavia, 1984.

[KC10] Kretowski, M.; Czajkowski, M. “An evolutionary algorithm for global induction of regression trees”. In: *Artifical Intelligence and Soft Computing*, Springer, 2010, pp. 157–164.

[KJ13] Kuhn, M.; Johnson, K. “Applied predictive modeling”. Springer, 2013.

[Koz92] Koza, J. R. “Genetic programming: on the programming of computers by means of natural selection”. MIT press, 1992, vol. 1.

[LCL12] Liu, C.-F.; Chen, T.-T.; Lee, S.-J. “A comparison of approaches for dealing with missing values”. In: Machine Learning and Cybernetics (ICMLC), 2012 International Conference on, 2012, pp. 1576–1582.

[LGH12] Luengo, J.; García, S.; Herrera, F. “On the choice of the best imputation methods for missing values considering three groups of classification methods”, *Knowledge and information systems*, vol. 32–1, 2012, pp. 77–108.

[Lin08] Linden, R. “Algoritmos genéticos: uma importante ferramenta da inteligência computacional”. BRASPORT, 2008, 2 ed..

[Liu05] Liu Peng, L. L. “A review of missing data treatment methods”, *Int. Journal of Intel. Inf. Manag. Syst. and Tech.*, vol. 1–3, 2005, pp. 412–419.

[LLW05] Liu, P.; Lei, L.; Wu, N. “A quantitative study of the effect of missing data in classifiers”. In: Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on, 2005, pp. 28–33.

[LR87] Little, R. J.; Rubin, D. B. “Statistical analysis with missing data”. Wiley, 1987.

[LR02] Little, R. J.; Rubin, D. B. “Statistical analysis with missing data”. Wiley, 2002.

[Mag04] Magnani, M. “Techniques for dealing with missing data in knowledge discovery tasks”, *Department of Computer Science, University of Bologna*, 2004.

[Mel99] Melanie, M. “An introduction to genetic algorithms”, *Cambridge, Massachusetts London, England, Fifth printing*, vol. 3, 1999.

[Nem63] Nemenyi, P. “Distribution-free multiple comparisons”, Tese de Doutorado, New Jersey, USA, 1963.

[NSZ07] Nogueira, B. M.; Santos, T. R.; Zárate, L. E. “Comparison of classifiers efficiency on missing values recovering: application in a marketing database with massive missing data”. In: Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on, 2007, pp. 66–72.

[Osb12] Osborne, J. W. “Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data”. Sage, 2012.

[PE08] Potgieter, G.; Engelbrecht, A. P. “Evolving model trees for mining data sets with continuous-valued classes”, *Expert Systems with Applications*, vol. 35–4, 2008, pp. 1513–1532.

[PNSK05] Pang-Ning, T.; Steinbach, M.; Kumar, V. “Introduction to Data Mining, (First Edition)”. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[Q+92] Quinlan, J. R.; et al.. “Learning with continuous classes”. In: Proceedings of the 5th Australian joint Conference on Artificial Intelligence, 1992, pp. 343–348.

[Qui93] Quinlan, J. R. “C4. 5: programs for machine learning”. Morgan kaufmann, 1993, vol. 1.

[Rec73] Rechenberg, I. “Evolution Strategies-ES”. Stuttgart: Frommann-Holzboog, 1973.

[Rub76] Rubin, D. B. “Inference and missing data”, *Biometrika*, vol. 63–3, 1976, pp. 581–592.

[Rub87] Rubin, D. B. “Multiple imputation for nonresponse in surveys. New York: J”. Wiley & Sons, 1987.

[SD08] Sivanandam, S.; Deepa, S. “Genetic Algorithm Optimization Problems”. Springer, 2008.

[SKG08] Su, X.; Khoshgoftaar, T. M.; Greiner, R. “Using imputation techniques to help learn accurate classifiers”. In: Tools with Artificial Intelligence, 2008. ICTAI’08. 20th IEEE International Conference on, 2008, pp. 437–444.

[SRP+11] Sridevi, S.; Rajaram, S.; Parthiban, C.; SibiArasan, S.; Swadhikar, C. “Imputation for the analysis of missing values and prediction of time series data”. In: Recent Trends in Information Technology (ICRTIT), 2011 International Conference on, 2011, pp. 1158–1163.

[SS04] Smola, A. J.; Schölkopf, B. “A tutorial on support vector regression”, *Statistics and computing*, vol. 14–3, 2004, pp. 199–222.

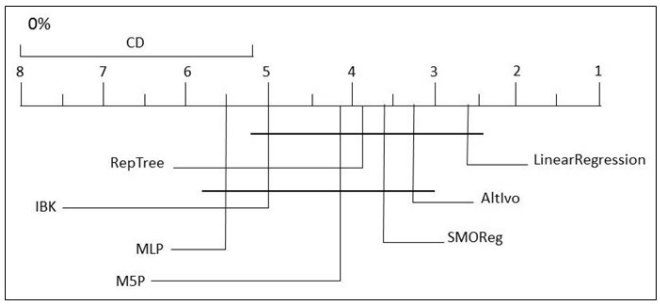
[SSCL08] Song, Q.; Shepperd, M.; Chen, X.; Liu, J. “Can k-nn imputation improve the performance of c4. 5 with small software project data sets? a comparative evaluation”, *Journal of Systems and Software*, vol. 81–12, 2008, pp. 2361–2370.

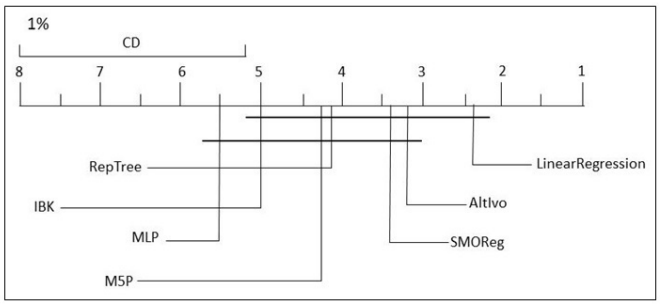
[TCS06] Twala, B.; Cartwright, M.; Shepperd, M. “Ensemble of missing data techniques to improve software prediction accuracy”. In: Proceedings of the 28th international conference on Software engineering, 2006, pp. 909–912.

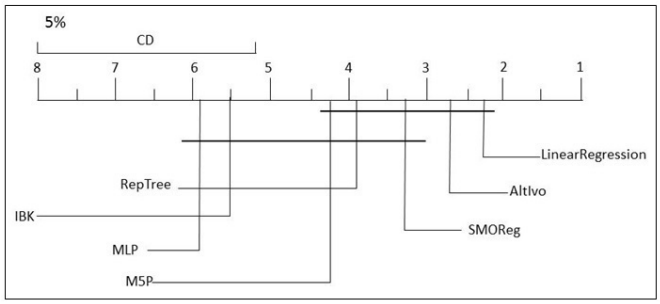
[WF11] Witten, I. H.; Frank, E. “Data Mining: Practical machine learning tools and techniques”. Morgan Kaufmann, 2011.

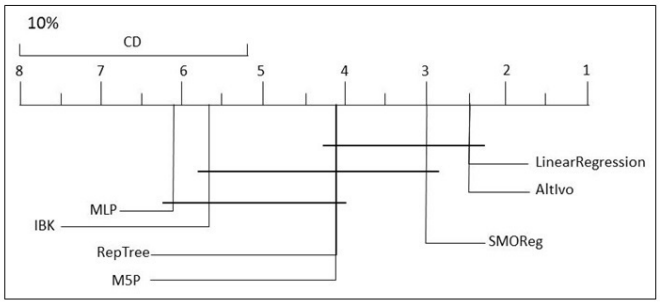
[ZWZ10] Zhang, S.; Wu, X.; Zhu, M. “Efficient missing data imputation for supervised learning”. In: Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on, 2010, pp. 672–679.

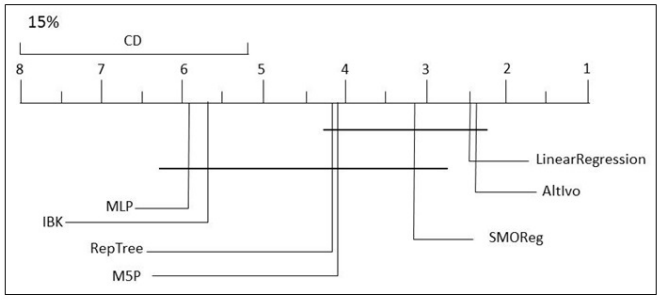
APÊNDICE A – DIAGRAMAS CRÍTICOS - 0% A 30%

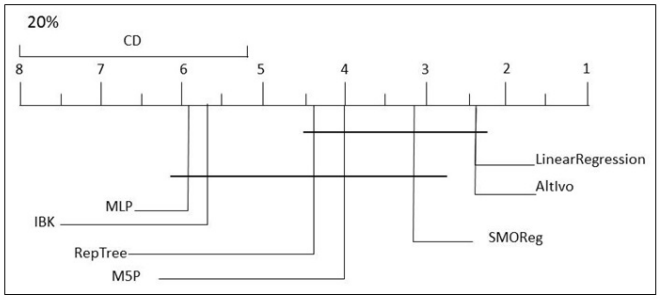


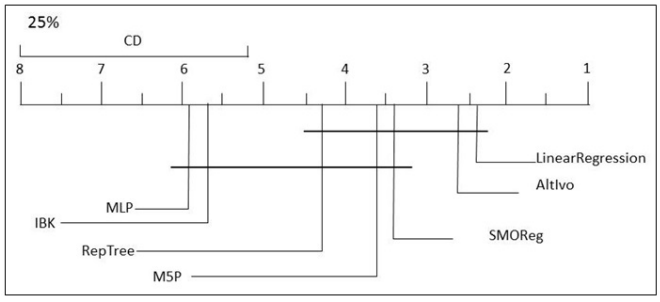


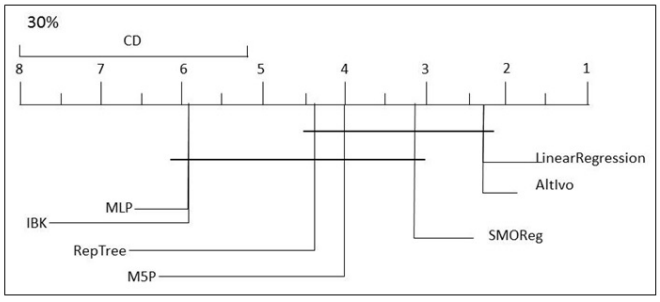












APÊNDICE B – DIAGRAMAS CRÍTICOS - 35% A 70%