CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

IMPUTAÇÃO DE DADOS *HOT-DECK*: UMA COMPARAÇÃO ENTRE COMITÊS DE REGRESSÃO

Thiago da Silva Pereira

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientador(a): Jorge de Abreu Soares
Coorientador(a): Eduardo Bezerra da Silva

Rio de Janeiro,
Agosto 2020

IMPUTAÇÃO DE DADOS *HOT-DECK*: UMA COMPARAÇÃO ENTRE COMITÊS DE REGRESSÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Thiago da Silva Pereira

Banca Examinadora:

Prof. Jorge de Abreu Soares, D.Sc. (Orientador(a))

Prof. Eduardo Bezerra da Silva, D.Sc. (Coorientador(a))

Prof. Diego Nunes Brandão, D.Sc. (CEFET/RJ)

Prof. Ronaldo Ribeiro Goldschmidt, D.Sc. (IME)

Rio de Janeiro,

Agosto 2020

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

P436 Pereira, Thiago da Silva
Imputacão de dados Hot-Deck: uma comparacão entre comitês de regressão / Thiago da Silva Pereira — 2020.
91f : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca , 2020.
Bibliografia : f. 86-91
Orientador: Jorge de Abreu Soares
Coorientador: Eduardo Bezerra da Silva

1. Algorítmos computacionais. 2. Estruturas de dados (Computação). 3. Processamento eletrônico de dados. 4. Redes neurais (Computação). I. Soares, ,Jorge de Abreu (Orient.). II. Silva, Eduardo Bezerra da (Coorient.) Título.

CDD 005.1

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

AGRADECIMENTOS

Ao CEFET/RJ, seu corpo docente e funcionários que ofereceram ambiente, estrutura e conhecimento necessário para o meu crescimento acadêmico.

Ao meu orientador Prof. Jorge de Abreu Soares e coorientador Prof. Eduardo Bezerra da Silva, pelas valiosas orientações.

Em especial, ao Prof. Jorge de Abreu Soares pelo apoio, por ter confiado em mim e me orientado nesta pesquisa.

Aos meus pais, Jorge Pereira e Janaína da Silva por me proporcionarem, com tanto esforço e apoio, chegar a este momento.

À minha esposa Mayara Manso da Silva e meu filho João Pedro Manso Pereira, por estarem comigo em todos os momentos dessa caminhada.

Aos meus irmãos Phelipe da Silva Pereira e Rafael da Costa Pereira por entenderem minha ausência nestes últimos tempos.

A todos os meus amigos, por entenderem minha ausência, meu estresse e por torcerem por mim e ao meu amigo Vinícius Barreto, em especial, pelas ajudas nos momentos mais complicados.

RESUMO

Imputação de dados *Hot-Deck*: uma comparação entre comitês de regressão

O pré-processamento de dados enfrenta uma questão importante relacionada ao tratamento de dados ausentes. Uma solução possível para resolver esse problema é a imputação *hot-deck*. Essa técnica possui duas etapas: agrupar registros semelhantes e executar a imputação propriamente dita. Selecionar o melhor algoritmo para imputação é um desafio, diversos algoritmos de aprendizado de máquina são estudados para isso, porém poucos estudos comparam métodos comitês para a etapa de imputação. Este estudo propõe uma solução baseada na imputação *hot-deck* comparando quatro comitês regressores: *Bagging*, *Adaboost*, *Gradientboost* e *Stacked Generalization*. Para verificar sua eficácia, usamos três conjuntos de dados, variando as taxas de ausências de 10% a 30%. Os resultados indicam que o *Gradientboost* apresenta melhor precisão em um tempo de processamento razoável.

Palavras-chave: Imputação Hot-Deck; Dados Ausentes; Bagging; Boosting; AdaBoost; GradientBoost; Stacked Generalization; Comitês

ABSTRACT

Hot-Deck Data Imputation: a comparison among ensemble methods

Preprocessing data faces an important question related to deal with missing data. A possible solution to resolve this problem is *hot-deck* imputation. This technique has two steps: group similar records and performs imputation. Selecting the best algorithm for imputation is a challenge. Several machine learning algorithms are studied for data imputation, however few studies compare ensemble methods for the imputation stage. This study proposes a solution based on hot-deck imputation comparing four ensemble regressors: *Bagging*, *Adaboost*, *Gradientboost*, and *Stacked Generalization*. To ascertain effectiveness, we have used three datasets, varying missing rates from 10% to 30%. Results measuring the precision of imputed data by both techniques indicate that the Gradientboost reveals better precision in reasonable processing time.

Keywords: Imputation Hot-Deck; Missing Data; Bagging; Boosting; AdaBoost; GradientBoost; Stacked Generalization; Ensemble

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ANN	Artificial Neural Networks
CART	Classification And Regression Tree
CONDBAGGING	Conditional Bagging
CONDRF	Conditional Random Forests
CONDTREE	Conditional Inference Tree
CWD	Casewise Deletion
EM	Expectation Maximization
ETL	Extract Transform Load
K-NN	*K*-Nearest Neighbor
KDD	Knowledge Discovery In Databases
LR	Linear Regression
MICE	Multiple Imputation By Chained Equations
MIST	Multiple Imputation By Sequential Regression Trees
MLLIB	Machine Learning Library
MLP	Multilayer Perceptron
MS	Mean Substitution
MVE	Multivariate Ensemble Method
MVO	Multivariate Ensemble Method With Output Augmentation
QOS	Quality Of Service
RBF	Radial Basis Function Network
REP	Reduced Error Pruning Tree
RF	Random Forest
RMSE	Root Mean Squared Error
SMOREG	Support Vector Machine For Regression
SOM	Self-Organizing Map
SVR	Support Vector Regression
UVE	Univariate Ensemble Method

SUMÁRIO

1- Introdução

Novas tecnologias e arquiteturas de software foram projetadas para extrair valor de um grande volume de dados [Manyika et al., 2011]. Uma dessas tecnologias é chamada *Knowledge Discovery in Databases* (KDD), que apoia este processo fornecendo meios para extrair conhecimento [Fayyad et al., 1996]. Inerente a este contexto, pode ser observado que essas bases possuem inconsistências e um dos fatores que contribuem para isso é o fato de os dados virem de diversas fontes diferentes, com formas diferentes e em tempos diferentes. Falhas em equipamentos como sensores de transmissão de dados, campos não preenchidos em formulários e falhas em rotinas de carga também contribuem para este cenário. Outra situação comum para este cenário está na integração de bases por meio da tecnologia conhecida como *Extract Transform Load (ETL)*.

Um tipo de inconsistência e o objeto de estudo nesta dissertação é a questão da ausência de dados. Dados ausentes podem prejudicar em diferentes níveis a busca de conhecimento em bases de dados levando a conclusões enviesadas e consequentemente incorretas Hoskin et al. [2019]. Ausência de dados é um problema relevante e presente nas diversas áreas de conhecimento existentes, e o estudo da complementação desses valores ausentes por um valor próximo ao real é conhecido por imputação de dados.

A imputação de dados é necessária em diversas áreas. Na área médica, por exemplo, sistemas de diagnósticos e avaliação de risco de pacientes são alguns dos casos onde um tratamento com relação aos dados ausentes se faz presente. Um estudo feito em um conjunto de dados de registros médicos, que tem como objetivo a previsão de risco de readmissão e casos que evoluam para emergência em pacientes, realizado por Saha et al. [2017], mostra uma melhora em suas previsões após realizar o trabalho de imputação de dados. Eles concluem que, após realizar a técnica de imputação proposta, tiveram ganhos em suas categorias de previsões.

Existem diversas técnicas para realizar a imputação de dados. As mais simples envolvem substituir valores ausentes pela média ou pela moda Maheswari et al. [2019]. Outras induzem algum modelo de aprendizado de máquina, como a imputação *hot-deck* Christopher et al. [2019]. Nela, os registros do conjunto de dados que possuem valores ausentes para algum atributo são agrupados por algum critério de similaridade [Ford,

1983; Christopher et al., 2019]. Em seguida, os valores ausentes são preenchidos em cada subgrupo.

A execução desse processo por meio da aprendizagem de máquina é complexa, despertando dúvidas em relação a qual algoritmo utilizar. Na área médica, sistemas de diagnósticos e avaliação de risco de pacientes são alguns dos exemplos onde um tratamento de dados ausentes se faz presente. Quinlan et al. [2019] em seus estudos sobre detecção de câncer cervical em mulheres, trataram os dados ausentes completando os dados com a média. Saha et al. [2017] mostraram uma melhora em suas previsões após realizar o trabalho de imputação de dados, utilizando aprendizado de máquina, em um conjunto de dados extraído de um aparelho de detecção de tumores. Na área de meteorologia esta situação também é um desafio. Gad et al. [2017] compararam os resultados da imputação de dados realizada em um conjunto de dados meteorológicos. Para condução dos testes, foram feitas as imputações com os algoritmos *Linear Regression*, *Random Forest*, *SVM* e *k-NN*.

Comitês de regressores (*ensemble regressors*) representam uma categoria de algoritmos onde a tomada de decisão é feita pela integração das respostas de vários modelos que o compõem. Vários regressores, conhecidos como *weak regressors*, com vieses relativamente similares são criados, e seus resultados são combinados por meio da média ou votação de maioria [Zhang and Ma, 2012].

Os algoritmos *Bagging* 2.9, *Adaboost* 2.10.1, *Gradientboost* 2.10.2 e *Stacked Generalization* 2.10.3 são exemplos categorizados como comitês Zhang and Ma [2012]. Em *Bagging*, T regressores independentes são treinados em paralelo, cada um com amostragem de dados gerada por meio da técnica de *bootstrap*. Nessa técnica, os registros são selecionados de forma aleatória e com reposição. Para obter o resultado final, os T regressores são combinados utilizando a média simples sobre seus resultados Zhang and Ma [2012]. Em *Adaboost*, os T regressores componentes do comitê são treinados de forma sequencial. Pesos são atrelados aos registros e, a cada iteração, são recalculados de acordo com os resultados certos ou errados na execução anterior. No caso de acerto, o peso é diminuído e, na ocorrência de erro, o peso é aumentado Freund et al. [1999]. *Gradientboost* se diferencia de *Adaboost* pelo fato de utilizar o gradiente descendente para minimizar a função de perda Friedman [2001]. Já em *Stacked Generalization*, T regressores independentes são treinados, em paralelo, de modo similar ao algoritmo *Bagging* e ao final um conjunto de dados com os resultados obtidos por

cada regressor é treinado finalmente por um regressor Wolpert [1992].

Este trabalho tem como objetivo comparar a realização da imputação *hot-deck* utilizando abordagens de aprendizado de máquina por meio do conceito de *comitês de regressão* com a imputação simples. Foi realizada a comparação dos resultados obtidos com quatro abordagens de construção de comitês: *Bagging*, *Adaboost*, *Gradientboost* e *Stacked Generalization*. A utilização de comitês de regressão na imputação *hot-deck* se mostrou uma alternativa promissora tanto no ponto de vista da precisão da reconstituição dos dados ausentes, quanto do ponto de vista do tempo de processamento.

Este trabalho ficou estruturado da seguinte forma: o capítulo 2 introduz os conceitos relacionados a pré-processamento, imputação de dados e comitês; o capítulo 3 aborda os trabalhos relacionados ao tema estudado; o capítulo 4 detalha a proposta para a realização dos testes da imputação *hot-deck*; o capítulo 5 descreve a avaliação experimental utilizada, os conjuntos de dados, as evoluções no *framework* e os resultados obtidos; e o capítulo 6 apresenta as considerações finais e as direções para trabalhos futuros.

2- Referencial Teórico

Neste capítulo são apresentados os conceitos referentes ao KDD (Seção 2.1), os métodos de imputação de dados (Seções 2.4, 2.5, 2.6) e os algoritmos comitês (Seção 2.8).

2.1- Visão Geral sobre o KDD

No processo de descoberta de conhecimento se aflora a busca por padrões ou tendências nos dados, busca essa que não se consegue observar sem o advento de ferramentas computacionais, uma vez que se trata de análises em giga, peta ou até zetabytes de dados. A área de estudo chamada Descoberta de Conhecimento em Bases de Dados (Knowledge Discovery in Databases (KDD)) apoia este processo fornecendo meios para que se possa extrair conhecimento em um conjunto de dados [Fayyad et al., 1996]. O KDD é dividido em etapas chamadas operacionais e classificadas como: pré-processamento, mineração de dados e pós-processamento. A Figura 1 ilustra as etapas operacionais do processo de KDD que são descritas a seguir.

Figura 1 – Ilustração das etapas operacionais no processo de KDD. Fonte: Adaptado de Goldschmidt et al. [2015]

- **Pré-Processamento** - esta etapa tem como objetivo preparar os dados para que os algoritmos utilizados na etapa de Mineração de Dados sejam executados da melhor forma;
- **Mineração de Dados** - principal etapa do processo de KDD e onde a busca por

conhecimento é efetivamente feita;

- **Pós-Processamento** - esta última etapa tem como objetivo tratar os dados resultantes da etapa de Mineração de Dados para facilitar a sua interpretação, apreciação e utilização pelos especialistas de domínio.

Objeto central deste projeto de pesquisa, a imputação de dados ausentes se faz presente na etapa de pré-processamento, portanto esta etapa terá uma atenção maior que as outras e será apresentada na próxima seção.

2.2- Pré-Processamento

Mensurar a qualidade dos dados é subjetivo e está diretamente ligado às necessidades de seu uso, ou seja, os dados devem satisfazer a premissas conhecidas de quem lhes querem utilizar. Citanto a temperatura climática do planeta, em condições normais, como exemplo de dado, é notório que valores acima de 70º graus célsius são incorretos. Neste sentido podemos afirmar que a qualidade deste dado (temperatura) é baixa. Existem alguns fatores que constituem a qualidade dos dados, são eles: precisão, integridade, consistência, pontualidade, credibilidade e interpretabilidade [Han et al., 2011].

Visando melhorar a qualidade dos dados, entre outros objetivos, a etapa de pré-processamento cobre assuntos relacionados com a captação, a organização e o tratamento dos dados e tem como objetivo preparar os dados para que sejam utilizados de forma mais apropriada por algoritmos na etapa de Mineração de Dados. As principais funções de pré-processamento de dados são [Goldschmidt et al., 2015]:

- Seleção de Dados - um subconjunto das bases de dados é identificado para ser considerado no processo de KDD, e a seleção dos dados pode ser feita por meio da escolha de atributos ou registros que serão utilizados.
- Limpeza de Dados - envolve qualquer atividade sobre os dados com objetivo de garantir a qualidade dos fatos representados pela base de dados, como por exemplo correções em dados ausentes, errôneos ou inconsistentes existente na mesma.
- Codificação dos Dados - os dados são codificados para que possam ser utiliza-

dos como entrada nos algoritmos de mineração. As codificações utilizadas são: numérica - categórica (transformação de valores reais em categorias ou intervalos) ou categórica - numérica (representação numérica para categorias).

- Enriquecimento dos Dados - tem como objetivo conseguir mais informações para serem agregadas aos registros existentes gerando valor ao processo de KDD. Umas das técnicas mais comuns é fazer consultas a bases de dados externas.

Os dados no mundo real, em sua natureza, são sujos, incompletos e inconsistentes e, como já dito, as técnicas de pré-processamento tendem a melhorar a sua qualidade, ajudando em uma melhor precisão na manipulação pela etapa seguinte do processo de KDD. Detectar *outliers* e anomalias, retificá-las precocemente e reduzir os dados a serem analisados pode levar a grandes retornos para a tomada de decisões Han et al. [2011]. Uma outra tarefa pertencente a limpeza de dados e tema central deste trabalho de pesquisa é o preenchimento de valores ausentes. Na próxima seção este assunto será detalhado bem como seus principais conceitos.

2.3- Dados Ausentes

A ausência de dados pode se fazer presente em diversas etapas, desde a origem do dado até sua manipulação. Este problema se torna cada vez mais relevante gerando um desafio a mais para a análise de dados tanto na ciência quanto nas empresas.

Uma razão bastante importante para isto é o fato de a maioria dos algoritmos utilizados para as análises serem baseados em matrizes completas, ou seja, sem valores ausentes. Sendo assim, todos os estudos necessitam que os dados estejam completos para que os algoritmos possam ser executados. Rubin [1976] apresentou teorias para a análise de dados ausentes baseada em verossimilhança e sistematizou esses métodos fornecendo uma base em que trabalhos futuros pudessem se apoiar.

2.3.1 Tipos de Ausência

Segundo Graham [2012] existem dois tipos de ausências de dados descritos na literatura: itens de dados ausentes (*item nonresponse*) e levas de dados ausentes (*wave nonresponse*).

Itens de dados ausentes ocorrem quando, em uma pesquisa de opinião, somente parte da mesma está completa, ou seja, parte da pesquisa está respondida e outra parte em branco. Dentre as causas deste tipo de ausência pode-se citar a não percepção de alguma questão em uma pesquisa por parte do entrevistado, o fato do entrevistado não saber a resposta de alguma questão ou não querer mesmo responder à questão por algum outro motivo. Outro exemplo é a perda de respostas ligadas a questões técnicas como falha na coleta ou no processamento dos dados.

Na situação de levas de dados ausentes, a pesquisa ocorre em levas ou ondas de perguntas, e a ausência ocorre quando, uma ou mais levas de perguntas não são respondidas, ou seja, o indivíduo deixa de responder uma leva de perguntas. Em alguns casos o indivíduo retorna a posteriori para responder as questões, mas em outros casos o mesmo nunca retorna e esse caso é denominado como atrito [Graham et al., 2012].

Além do tipo, as ausências possuem outras duas características que serão descritas nas duas próximas seções: **padrões de ausência e mecanismo de ausência**.

2.3.2 Padrões de Ausência

Quando se analisa um conjunto de dados em busca de algum conhecimento não facilmente perceptível, esses dados são observados em um formato tabular (retangular ou matriz). Alguns padrões de ausência de dados podem ser observados. Schafer and Graham [2002] definem que o reconhecimento desses padrões de ausência são um passo importante para que se possa escolher a melhor técnica de preenchimento desses valores e categorizam esses padrões em *geral* ou *aleatório* e *específico*.

Na ausência de dados de padrão geral ou aleatória, como a própria classificação sugere, a ausência está dispersa em qualquer registro do conjunto de dados. Com

relação aos padrões de ausência específicos, são divididos em dois subtipos *univariados* e *monotônicos* [Schafer and Graham, 2002].

Nos padrões classificados como univariados, observa-se uma distribuição de ausência dirigida a uma única variável. Ocorre em um item *Y* mas o conjunto dos *p* outros itens $X_1, ..., X_p$ continuam completos. Este padrão também deve incluir situações onde *Y* representa um grupo de itens completamente observados ou completamente ausentes e em suma estão restritos a uma única variável do conjunto de dados. Nos padrões monotônicos, a ausência é observada em mais de uma variável de tal forma que um grupo de itens $Y_1, ..., Y_p$ quando ordenados, se Y_j contém valores ausentes então $Y_j + 1, ..., p$ também conterão [Schafer and Graham, 2002]. A Figura 2 ilustra esses conceitos.

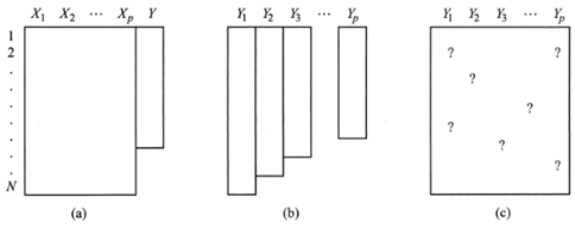

Figura 2 – Padrões de ausência em conjuntos de dados retangulares: *(a)* padrão univariado, *(b)* padrão monotônico e *(c)* padrão arbitrário. Em cada caso, as linhas correspondem a unidades observáveis e colunas correspondem a variáveis. Fonte: Adaptado de [Schafer and Graham, 2002].

2.3.3 Mecanismos de Ausência

Além de levar em consideração a observação dos padrões de ausência existente no conjunto de dados, outra característica que se deve observar é o mecanismo de ausência dos dados, ou seja, o que causou a ausência. Diversos trabalhos envolvendo a complementação de dados levam em consideração o mecanismo de ausência de dados existente no conjunto em questão.

Schafer and Graham [2002] destacam que, de acordo com a literatura estatística, é sugerido que mecanismo pode ser um processo pelo qual alguns dados são gerados e outros são perdidos. Little and Rubin [2019] sugerem que mecanismo de ausência é relacionado ao processo pelo qual os dados são perdidos e citam a importância de saber

se a ausência dos dados estão relacionadas aos valores subjacentes das variáveis do conjunto de dados.

Estatisticamente Little and Rubin [2019] os definem da seguinte forma: sendo *Y* um conjunto de dados completo e *M* o conjunto de ausências, o mecanismo de ausência é caracterizado pela distribuição condicional de *M* dado *Y*, $f(M|Y,\phi)$ onde, ϕ significa parâmetros desconhecidos.

Rubin [1976], define três tipos de classificações para mecanismos de ausência: completamente aleatório *(MCAR – Missing Completely At Random)*, aleatório *(MAR – Missing At Random)* e não aleatório *(NMAR – Not Missing At Random, ou IM – Ignorable Missing)*.

O mecanismo de ausência de dados é classificado como completamente aleatório quando o seu real motivo é desconhecido, não havendo qualquer relação com outro(s) atributo(s) do conjunto de dados, ou seja, quando *M* não depende dos valores de *Y*, sendo assim, se $f(M|Y,\phi) = f(M|\phi)$ *para todo* Y.

Se os valores ausentes dependem de algum(s) atributo(s) do conjunto de dados, este é classificado como aleatório. Seja Y_{obs} um subconjunto observado completo de *Y*, e Y_{mis} um subconjunto observado de dados ausentes, para que o mecanismo de ausência seja aleatório, Y_{mis} deve ser dependente de Y_{obs}, ou seja, $f(M|Y,\phi) = f(M|Y_{obs},\phi)$ *para todo* Y_{mis}, ϕ.

Existe um terceiro mecanismo de ausência chamado não aleatório, neste, a distribuição de ausência *M* depende dos valores ausentes representados por Y_{mis}. Graham and Donaldson [1993] definem este mecanismo de ausência como **inacessível** pelo fato de o mesmo não ser mensurável e, portanto, não disponível para avaliação.

Algumas ações podem ser feitas para tratar a ausência dos dados e ignorar essas ausências é uma ação possível, porém a substituição dos valores ausentes por valores próximos aos reais surge como uma solução viável para este problema Rubin [1988]. A próxima seção deste trabalho irá dissertar sobre este tema.

2.4- Imputação Simples de Dados

O objetivo de qualquer analista de dados é, entre outros, obter resultados não enviesados sobre suas análises. Análises em conjuntos de dados com valores ausentes podem enviesar seus resultados, não refletindo um panorama real. Sendo assim, existem duas maneiras de lidar com essa situação: ignorar a falta dos dados e assumir os riscos dos problemas relacionados com a ausência de dados ou realizar a imputação desses dados em seus respectivos lugares afim de minimizar os problemas decorrentes de sua falta [Schafer and Graham, 2002].

Segundo Little and Rubin [2019], imputações de dados podem ser definidas como médias ou extrações de uma distribuição preditiva dos valores ausentes, requerem um método de criação de uma distribuição preditiva para a imputação com base nos dados observados e definem duas abordagens genéricas para geração dessa distribuição: modelagem explícita e modelagem implícita.

Na modelagem explícita a distribuição preditiva é baseada em um modelo estatístico formal, por exemplo normal multivariada, e, portanto, as suposições são explícitas. São exemplos de modelagem explícita: imputação por média, imputação por regressão, imputação por regressão estocástica.

Na modelagem implícita o foco está em um algoritmo, o que implica em um modelo subjacente. As suposições estão implícitas, mas ainda precisam ser cuidadosamente avaliadas para garantir que sejam razoáveis. São exemplos de modelagem implícita: imputação *Hot Deck*, imputação por substituição e imputação *Cold Deck* [Little and Rubin, 2019].

Soares [2007] classifica a imputação em dois métodos: simples ou híbrido. No método simples, apenas um valor proposto é considerado para cada valor ausente. Já no método híbrido, dois ou mais métodos de imputação simples são considerados no processo de imputação como um todo visando a sua melhora [Little and Rubin, 2019].

2.5- Imputação Múltipla de Dados

Um exemplo do método de imputação híbrida é a imputação múltipla [Raghunathan, 2004]. Neste método o conjunto de dados ausentes é substituído por mais de um conjunto de dados e cada um destes, junto com o restante dos dados formam um conjunto de dados completo, que por sua vez, é analisado em separado por meio de estimativas estatísticas [Raghunathan, 2004].

A imputação simples substitui um valor ausente por um único valor e, em seguida, o trata como se fosse um valor verdadeiro [Rubin, 1988]. Como resultado, a imputação simples ignora a incerteza e quase sempre subestima a variância. A imputação múltipla supera esse problema, levando em conta tanto a incerteza inerente a imputação simples quanto a incerteza entre as imputações subsequentes.

O método de imputação múltipla produz n sugestões para cada valor ausente. Cada um desses n valores é atribuído como um valor escolhido e n novos conjuntos de dados são criados como se uma imputação simples tivesse ocorrido em cada conjunto de dados. Dessa forma, uma única coluna de uma tabela gera novos conjuntos de dados, que são analisados caso a caso usando métodos específicos. Essas análises são combinadas em um segundo passo, gerando ou consolidando resultados desse conjunto de dados. A figura 3 ilustra este conceito.

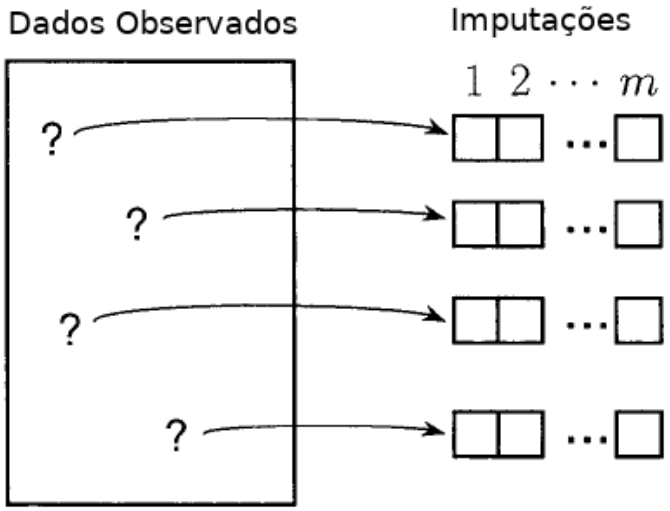

Figura 3 – Ilustração do processo de imputação múltipla. Fonte: Adaptado de Schafer and Graham [2002]

O processo de imputação múltipla se dá da seguinte forma:

1. Para cada atributo que possui valores ausentes um conjunto de n valores a serem imputados são gerados;
2. Uma análise estatística é realizada em cada conjunto de dados gerado a partir do uso de uma das n sugestões de substituição geradas no item anterior;
3. Os resultados das análises realizadas são combinados para produzir um conjunto de resultados.

Não encontrei na literatura um consenso ao escolher o número de n. Escolher um número alto para n pode não ser uma boa escolha para o processo como um todo devido ao grande número de conjuntos de dados gerados para cada novo valor.

2.6- Imputação Hot Deck

A imputação *hot-deck* é uma das técnicas mais utilizadas. Esta técnica consiste em realizar a complementação dos dados em subgrupos completos dos mesmos [Ford, 1983]. Neste sentido, os dados são agrupados por algum critério de similaridade e então os valores ausentes são preenchidos em cada subgrupo. Dentro de cada grupo os registros tendem a ser homogêneos, porém é importante que exista correlação entre os atributos classificadores (que serviram de base para a geração dos grupos) e os atributos ausentes. Caso contrário os resultados podem não ser satisfatórios. A imputação *Hot Deck* também é definida como um procedimento em que a imputação vem de outros registros nos mesmos dados, e estes são escolhidos com base no registro incompleto [Ford, 1983].

A imputação *Hot Deck* é dividida em duas etapas: dividir os dados em grupos similares e realizar a imputação em cada grupo. Porém o critério de agrupamento não é definido. Em alguns casos um outro algoritmo de aprendizado de máquina, como o *k-Means*, é utilizado na etapa de agrupamento, em outros casos, como em formulários de pesquisas, o agrupamento é feito por alguma coluna alvo como ”idade”, então quando algum existir algum valor ausente para este grupo o registro é preenchido com o valor do registro anterior.

Magnani [2004] argumenta que a imputação *Hot Deck* é um procedimento abstrato, que tem sido utilizado, na prática, de maneiras muito diferentes e não se baseia de uma teoria forte. Existem algumas razões que fazem da utilização desta técnica uma boa escolha segundo Magnani [2004]. São elas:

- Consegue-se uma redução de desvio sem a imposição de um modelo rígido; Reduzindo o tamanho do grupo, os dados tendem a se tornar homogêneos;
- Produção de um conjunto de dados limpo, sem valores ausentes;
- Preservação da distribuição da população representada pela amostra;
- Para alguns valores ausentes, nenhuma informação sobre imputação pode ser encontrada. Isto permite que outras técnicas de imputação possam ser usadas em conjunto;
- Pode-se usar uma técnica diferente para cada grupo gerado;
- Não precisa de um modelo robusto para prever valores ausentes;
- Não assume nenhuma distribuição em particular.

2.7- Aprendizado de Máquina

O Aprendizado de Máquina é a ciência que estuda como os computadores podem aprender com os dados existentes. Mitchell [1997] define o aprendizado de máquina da seguinte maneira: um programa de computador aprende a experiência E com relação a alguma classe de tarefas T e medida de desempenho P e se seu desempenho nas tarefas em T, medido por P, melhora com a experiência E. Ele cita como exemplo carros autônomos.

- Tarefa T: dirigir em rodovias públicas de quatro pistas usando sensores de visão;
- Medida de Desempenho P: distância média percorrida antes de um erro (erro definido por um ser humano);

- Experiência de Treino *E*: uma sequência de imagens e comandos de direção gravados observados de um motorista humano.

De acordo com a definição descrita acima, a medida que a experiência de treino melhora, a medida de desempenho também melhora e consequentemente a tarefa a ser aprendida também.

Como resultado da experiência de treino obtém-se uma predição, como uma decisão se um carro autônomo irá frear ou acelerar, se um e-mail é spam ou não ou a descoberta de qual animal está em uma foto. Nem sempre essa decisão é acertada e uma maneira de levar em consideração o resultado de várias experiências de treino para tomar uma melhor decisão, ou seja, tomar uma decisão em conjunto, é utilizando os *Comitês* que podem ser formados por componentes tanto de regressão quanto classificação [Zhang and Ma, 2012]. Na literatura este termo é referenciado, geralmente, como classificador. Para este trabalho foi usado comitês com componentes de regressão, porém este será referenciado somente como componente pois os conceitos são, na maioria dos casos, os mesmos. Particularidades de componentes de regressão serão mencionadas quando necessário.

2.8- Comitês

Tomadas de decisão em conjunto estão presente em nosso cotidiano, por viver em um país democrático votamos em conjunto e escolhemos um presidente de quatro em quatro anos, o sistema jurídico de diversos países também se baseiam em decisões tomadas em conjunto. Seja na busca de opiniões médicas de diferentes profissionais, uma leitura em recomendações de um produto específico ou em recomendações de profissionais para uma importante vaga de trabalho, nós tendemos a buscar opiniões diferentes antes de uma tomada de decisão importante. Por essas mesmas razões, a tomada de decisão em conjunto é feita no aprendizado de máquina [Zhang and Ma, 2012].

A precisão da decisão tomada por cada componente possui uma variabilidade diferente de zero e qualquer erro de regressão é composto por duas propriedades controláveis: viés, que é a precisão do componente e a variância, que é a precisão do componente quando treinado em diferentes conjunto de treinamentos. Muitas vezes,

esses dois componentes têm uma relação de compromisso: os componentes com baixo viés tendem a ter alta variância e vice-versa. Por outro lado, também se sabe que a média tem um efeito de suavização (redução de variância). Portanto, o objetivo dos sistemas de tomadas de decisão em conjunto é criar vários componentes, chamados *weak classifiers or regressors*, com vieses relativamente similares e então combinar suas saídas pela média (regressão) ou votação de maioria (classificação) para reduzir a variância [Zhang and Ma, 2012]. A figura 4 ilustra estes conceitos.

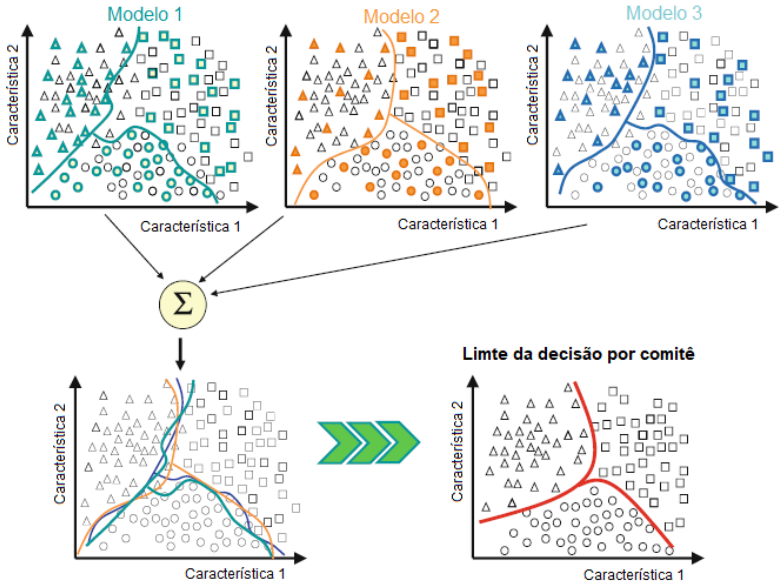

Figura 4 – Redução da variabilidade usando comitês. Fonte: Adaptado de [Zhang and Ma, 2012]

No contexto de tomadas de decisão em conjunto, um componente preciso é aquele que tem uma taxa de erro melhor que uma adivinhação aleatória para novos valores, como lançar uma moeda ao ar e adivinhar se o resultado será cara ou coroa, e dois componentes são diversos quando suas taxas de erros são diferentes para uma determinada predição [Dietterich, 2000]. Neste sentido, supondo que se tenha três componentes idênticos, ou seja, baixa diversidade, então quando $h_1(x)$ está incorreto, $h_2(x)$ e $h_3(x)$ também estarão. Porém quando se tem uma alta diversidade nas taxas de erros entre os componentes, a probabilidade de um acerto em conjunto aumenta, ou seja, para $h_1(x)$ incorreto, $h_2(x)$ e $h_3(x)$ poderão estar corretos melhorando a precisão na predição [Dietterich, 2000].

Como já descrito, a diversidade é uma propriedade crucial para o sucesso da predição, portanto, a seleção dos dados que serão levados em consideração para o

processamento, precisa apoiar este fator. Zhang and Ma [2012] descrevem três etapas importantes para o sucesso de um sistema de tomada de decisão em conjunto: seleção de dados (amostragem); componentes de dados de treinamentos e combinação dos componentes.

A seleção dos dados precisa apoiar a diversidade para que a tomada de decisão em conjunto seja melhor, uma estratégia para isto é a seleção de diferentes conjuntos de dados utilizando uma técnica chamada *Bootstrap*, nesta técnica os dados são sorteados aleatoriamente com repetição para geração de N outros conjuntos de dados. A predição realizada a partir dos dados de treinamento é a finalidade da tomada de decisão em conjunto. Esta etapa é onde os modelos são treinados individualmente. Inúmeros algoritmos foram desenvolvidos para esta etapa, este trabalho irá se concentrar na comparação entre quadro (*Bagging*, *Adaboost*, *Gradientboost* e *Stacked Generalization*). A última etapa, a combinação dos modelos, lida com o mecanismo usado para combinar os N resultados dos componentes individuais, mecanismo este que varia de acordo com os algoritmos utilizados na etapa de predição dos dados de treinamento. Alguns componentes, como o *k-NN*, retornam valores discretos, neste caso o mecanismo utilizado é a votação do valor em maioria, em outros casos como em redes neurais ou regressão linear, um valor contínuo é retornado, neste caso o mecanismo utilizado é uma operação aritmética como a média.

2.9- Algoritmo Bagging

O algoritmo *Bagging* é um dos primeiros e mais simples categorizados como *Comitês*. Dado um conjunto de dados de treinamento S de cardinalidade N, *Bagging* treina T modelos independentes, cada um com amostragem de dados com repetição, com N instâncias de dados retirados de S [Zhang and Ma, 2012]. A diversidade entre os conjuntos de treinamento a serem processados está garantido pela técnica de *Bootstrap* na seleção dos dados. Nesta técnica de seleção de amostragens são gerados N subconjuntos com base nos dados do conjunto de dados em questão, os dados são sorteados aleatoriamente, com reposição, mantendo a probabilidade $1/N$ de cada dado ser selecionado. Neste sentido cada amostra contém aproximadamente $63,2\%$ dos dados

originais Breiman [1996a]. A figura 5 ilustra como funciona o processo do algoritmo *Bagging*.

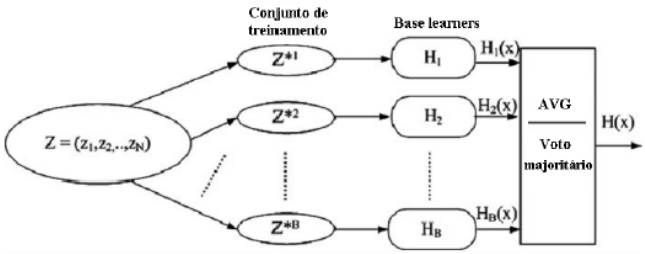

Figura 5 – Abordagem de Bagging. Utilização de *Bootstrap* para geração de amostras de treino. A partir de cada amostra é gerado um *base learner*. Ao final a decisão em conjunto é tomada. Fonte: Adaptado de [Zhang and Ma, 2012]

A utilização de *Bagging* melhora os resultados na tomada de decisão em conjunto mesmo que os resultados dos componentes individuais sejam instáveis, essa é uma das principais razões para utilização de *Bagging*. Vale lembrar que qualquer algoritmo de aprendizado de máquina pode ser utilizado como componente, pode-se utilizar também mais de um algoritmo diferente como componente. O algoritmo 1 ilustra *Bagging* por meio de um pseudocódigo.

Algoritmo 1 – ALGORITMO BAGGING

Entrada: Conjunto de dados $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$;
Algoritmo de aprendizado $\mathcal{E}$;
Número de *base learners* T.

início

para $t = 1, ..., T$ **faça**
$h_t = \mathcal{E}(D, D_bs)$ % D_bs é a distribuição *bootstrap*

Saída: $H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \mathbb{I}(h_t(x) = y)$

Na próxima seção serão discutidos os conceitos relacionados ao algoritmo de *Boosting* em específico os algoritmos *AdaBoosting*, *Gradient Boosting* e *Sacking*.

2.10- Algoritmos Boosting

Boosting possui uma abordagem iterativa para geração do chamado componente forte (*strong classifier or regressor*), capaz de atingir baixas taxas de erro de treinamento a partir de componentes fracos (*weak classifiers or regressors*), cada um sendo levemente mais preciso que uma escolha aleatória (probabilidade de acerto pouco maior que 50%) [Zhang and Ma, 2012]. Diferentemente da técnica de *Bagging*, a seleção de dados de treinamento em *Boosting* não utiliza *Bootstrap*. Os modelos são treinados sequencialmente e com o próximo treinamento focando nos valores gerados erroneamente pelo treinamento anterior, fazendo com que o aprendizado seja incremental.

Zhang and Ma [2012] definem que *Boosting*, projetado para resolver um problema de classificação binária, cria conjuntos de três classificadores fracos de cada vez. O primeiro classificador H_1 é gerado a partir de um subconjunto aleatório do conjunto de dados de treinamentos. O segundo classificador H_2 é gerado a partir de um subconjunto diferente do primeiro, neste caso constituído por metade dos dados classificados corretamente por H_1 e a outra metade classificados errados. O terceiro classificador H_3 é gerado a partir dos dados nos quais tanto H_1 quanto H_2 discordam. Ao final esses têes classificadores são combinados em uma votação de maioria para a definição da classificação vencedora [Freund et al., 1999]. A figura 6 ilustra estes conceitos.

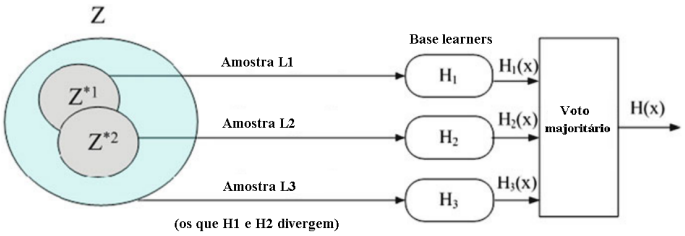

Figura 6 – Ilustração do algoritmo *Boosting*. Fonte: Adaptado de [Zhang and Ma, 2012]

Em *Boosting*, conjunto de dados de treinamento é dividido em três partes, Z_1^*, Z_2^* e Z_3^*. Para uma dada instância se a classificação atribuída a ela nos dois primeiros classificadores (H_1 e H_2) forem corretas, esta é a classificação final. O conjunto com os dados classificados erroneamente por H_1 e H_2 definem Z_3^* que é usada para gerar o classificador H_3. O algoritmo 2 ilustra o pseudocódigo de *Boosting*.

Algoritmo 2 – ALGORITMO BOOSTING

Entrada: Conjunto de dados $Z = \{z_1, z_2, ..., z_n\}$, com $z_i = (x_1, y_i)$, onde $x_i \in \mathcal{X}$ e $y_1 \in \{-1, +1\}$

início

1: Selecionar aleatoriamente sem reposição, $L_1 < n$ dados de Z para obter Z_1^*.

2: Executar o aprendiz fraco em Z_1^*, gerando classificador H_1.

3: Selecionar $L_2 < n$ dados de Z, com parte dos dados classificados erroneamente por H_1 para obter Z_2^*

4:Executar o aprendiz fraco em Z_2^*, gerando classificador H_2.

5: Selecionar todos os dados de Z no qual H_1 e H_2 classificaram erroneamente, produzindo Z_3^*.

6: Executar o aprendiz fraco em Z_3^*, gerando classificador H_3.

7: Classificação final por voto majoritário: $H(x) =$ sign $(\sum_{b=1}^{3} H_b(x))$

Saída: Classificador $H : \mathcal{X} \to \{-1, +1\}$

Esta é a ideia central do aprendizado de máquina realizado pelo algoritmo de *Boosting*, onde um aprendizado forte (*strong learning*) pode ser realizado a partir de uma combinação de aprendizagens fracas (*weak learning*) [Schapire, 1990]. Com sua evolução, *Boosting* deixou de ser apenas uma solução específica para uma categoria de algoritmos [Schapire and Freund, 2012]. Nas próximas seções serão apresentados alguns desse algoritmos que evoluíram a partir de *Boosting* e que serão utilizados neste trabalho de pesquisa.

2.10.1 AdaBoosting

Uma nova abordagem sobre *Boosting* é proposta por Freund et al. [1999] e foi denominada de *Adaptative Boosting* (*AdaBoost*). A ideia deste algoritmo é utilizar versões do mesmo conjunto de dados de treinamento com pesos associados a cada registro a fim de induzir os modelos, que são obtidos de forma sequencial. A cada iteração os pesos são recalculados de acordo com as predições mais próximas e mais distantes dos valores reais gerados na execução anterior. No caso de valores mais próximos o peso

é diminuído e no caso de valores mais distantes o peso é aumentado. A figura 7 ilustra estes conceitos.

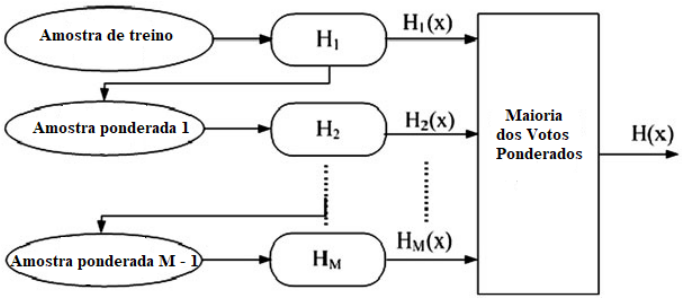

Figura 7 – Abordagem de *Adaboost*. Cada componente fraco é gerado a partir de amostras ponderadas do conjunto de dados de treinamento. Fonte: Adaptado de [Zhang and Ma, 2012]

Uma questão fundamental na utilização de pesos para os registros, é que na execução seguinte, o algoritmo de aprendizado de máquina terá uma amostra com os dados mais enriquecidos pois estarão destacados, por meio de pesos altos, os dados que deverão ter uma melhor atenção durante o aprendizado. Esta situação não acontece com *Bagging* e *Boosting*. A tabela 1 resume as diferenças entre *Boosting* e *Adaboost*.

Tabela 1 – Diferenças entre Boosting e AdaBoost

	Boosting (Algoritmo 2)	AdaBoost (Figura 7)
Uso do Dado	Seleção aleatória, sem reposição	Pesos
Numero de Amostras	Três	Uma
Numero de Classificadores	Três	Até *M*
Decisão em Conjunto	Maioria de Votos	Maioria de Votos (Classificação) ou Média (Regressão)

Uma ponto relevante é que ao final das M iterações os valores atribuídos com pesos mais altos, ou seja de mais difícil aprendizado, podem ser *outliers*. O algoritmo *AdaBoost* também é utilizado para tarefas de regressão. Neste caso, para combinarem as predições geradas, são utilizadas operações matemáticas como médias ou médias ponderadas no resultado das predições [Zhang and Ma, 2012]. Nas predições erradas pode-se calcular o “tamanho do erro”, a diferença entre o valor correto e o valor errado que pode ser calculado utilizando a *média dos erros quadrados (mean squared errors)* por exemplo. Na próxima seção será apresentado o algoritmo, que leva em consideração a medida de erro de cada classificador, chamado *Gradient Boosting*.

2.10.2 Gradient Boosting

Não muito diferente que o *AdaBoost*, *Gradient Boosting* utiliza o mesmo conceito combinando componentes fracos de forma iterativa para a geração de um componente forte. A diferença está na utilização do gradiente descendente para minimizar a função de perda. A ideia central deste algoritmo é construir um componente forte que maximize a correlação com o gradiente negativo da função de perda do comitê como um todo [Natekin and Knoll, 2013]. A escolha da função de perda é feita de forma arbitrária, a média dos erros quadrados *(mean squared errors)* é muito comum mas outras funções podem ser utilizadas como o erro absoluto médio *(mean absolute error)* ou até mesmo uma outra função construída pelo pesquisador.

Zhang et al. [2018] argumentam que um componente fraco construído pelo *Gradient Boosting* pode ser melhorado conforme as iterações vão acontecendo e os pesos vão sendo atualizados. Após uma iteração T_{m+1} um modelo é construído por $m + 1$ árvores (caso *random forest* seja o algoritmo de aprendizado de máquina escolhido) por exemplo e este novo modelo melhora o modelo anterior T_m. Consequentemente T_{m+1} é melhorado por T_{m+2} e assim sucessivamente [Zhang et al., 2018]. Como o próprio nome sugere, o algoritmo Gradiente Descendente é utilizado para minimizar a função de perda. Neste sentido, a árvore gerada $m + 1$ é a que minimizou a função de perda do modelo T_m. A figura 8 ilustra estes conceitos.

De acordo com o exemplo da figura 8, na primeira iteração uma linha vertical é formada para classificar os símbolos. O modelo T_1 é a versão mais simples, e ele classifica como círculos os símbolos do eixo x com valores menores que 1 e os valores maiores que 1 como quadrados. Este primeiro modelo classificou erroneamente 3 círculos, representando erro do modelo. Para a próxima iteração pesos maiores são associados a estas classificações erradas e o como resultado aparece o modelo T_2. Pode-se afirmar que este novo modelo é a combinação de T_1 com T_2. Este processo se repete até que todo o conjunto de dados seja processado.

Natekin and Knoll [2013] em seu artigo detalha estes conceitos por meio do algoritmo 3.

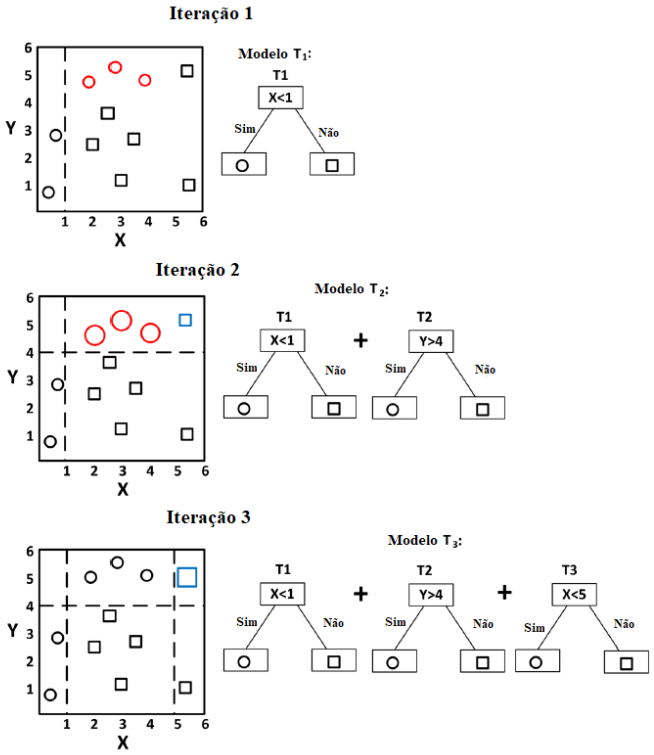

Figura 8 – Exemplo do processo do *Gradient Boosting*. Fonte: Adaptado de [Zhang et al., 2018]

Algoritmo 3 – ALGORITMO GRADIENT BOOSTING

Entrada: Conjunto de dados $(x_i, y_i)_{i=1}^{N}$

Numero de iterações M

Função de perda $\psi(y, f)$

Algoritmo base $h(x, \theta)$

início

1: Inicializar o modelo $\hat{f}_0$ com valores constantes

2: **Para** $t = 1, ..., M$ **faça**

3: Computar o gradiente negativo $g_t(x)$

4: Executar algoritmo base $h(x, \theta)$

5: achar o melhor gradiente descendente ρ_t:

$\rho_t = \arg\min_{\rho} \sum_{i=1}^{N} \psi\left[y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)\right]$

6: Atualização dos pesos: $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \theta)$

Saída: Classificador/Regressor forte

2.10.3 Stacked Generalization

O algoritmo *Stacked Generalization* tem como principal característica a combinação de múltiplos modelos gerados a partir de algoritmos de aprendizado de máquina anteriores, é utilizado para tarefas tanto de classificação quanto regressão e ainda aprendizado não supervisionado [Wolpert, 1992] [Breiman, 1996b] [Smyth and Wolpert, 1998].Ting and Witten [1999] detalham o algoritmo da forma apresentada no próximo parágrafo.

Dado um conjunto de dados $\mathcal{L} = \{(y_n, x_n), n = 1, ...N\}$, onde y_n é o valor da classe e x_n é o vetor que representa os valores dos atributos da $n$ésima instância, dividir os dados aleatoriamente em J conjuntos de dados de partes quase iguais $\mathcal{L}, ...\mathcal{L}_j$. Definir $\mathcal{L}_j$ e $\mathcal{L}^{(-j)}$ para serem respectivamente o conjunto de dados de teste e de treino do $j$ésimo *fold* do *j-fold cross-validation*. Dado K algoritmos de aprendizado de máquina, no qual são chamados de *level-0 generalizations*, executar o $k$ésimo algoritmo com o conjunto de dados de treino $\mathcal{L}^{(-j)}$ para induzir o modelo $\mathcal{M}_k^{(-1)}$, para $k = 1, ..., K$. Esses modelos são denominados *level-0 models*. Para cada instância x_n em $\mathcal{L}_j$, o conjunto de treinamento para o $j$ésimo *cross-validation fold*, z_{kn} define a predição do modelo $\mathcal{L}^{(-j)}$ em x_n. Ao final do processo de validação cruzada *cross-validation* o conjunto de dados gerado com as saídas dos K modelos é definido como $\mathcal{L}_{CV}$ = $\{(y_n, z_{1n}, ..., z_{Kn}), n = 1, ..., N\}$. Esse conjunto de dados é chamado de *level-1 data*.

A segunda etapa do algoritmo, também chamada de meta-aprendizado, é a utilização de algum algoritmo de aprendizado de máquina chamado *level-1 generalizer* para gerar, a partir dos dados criados na primeira etapa, um modelo $\tilde{\mathcal{M}}$ para y em função de $(z_1, ..., z_k)$. A figura 9 ilustra este processo.

Os algoritmos *AdaBoost* e *Gradient Boosting*, descritos anteriormente, utilizam em seu processo, o conjunto de dados de treinamento completo em cada iteração, e a definição dos pesos são estabelecidos no momento em que os conjuntos de dados são treinados por um algoritmo de aprendizado de máquina e atualizados a cada iteração [Zhang and Ma, 2012]. Essa situação não permite determinar que componente aprendeu qual parte do espaço do atributo. Com o algoritmo *stacked generalization* é possível determinar que um subconjunto do conjunto de dados de treinamento será executado por um algoritmo de aprendizado de máquina específico e outro subconjunto por outro algoritmo específico. Esta situação, além de apoiar um dos conceitos principais do

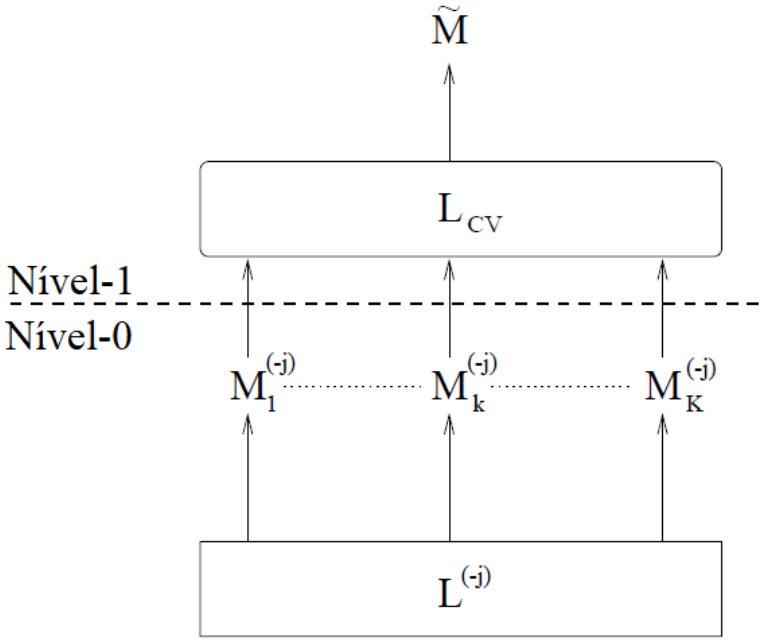

Figura 9 – Processo de validação cruzada no level-0; o conjunto de dados de teste level-1 $\mathcal{L}_{CV}$; e ao final o modelo level-1 $\tilde{\mathcal{M}}$. Fonte: Adaptado de Ting and Witten [1999]

processo de tomadas de decisão em conjunto, que é a diversidade dos dados, dá ao pré-processamento o uma dinâmica maior no sentido em que parte dos dados pode ser tratados de maneira específica para algoritmos específicos.

Também categorizado como *super learning*, *Stacked Generalization* possui um alto custo computacional visto precisar executar K diferentes algoritmos de aprendizado de máquina e combiná-los em um super modelo $\tilde{\mathcal{M}}$ posteriormente. Neste sentido, um dos principais desafios é a melhoria de desempenho do processo como um todo [LeDell, 2015].

3- Trabalhos Relacionados

Nesse capítulo serão abordados trabalhos relacionados ao tema de imputação de dados referente à aplicação de algoritmos comitês. Serão citados também trabalhos referentes à utilização de componentes e comparações de desempenho utilizados nesse trabalho.

Foi realizada uma pesquisa, apresentada no Apêndice A.1, na base de dados bibliográfica *Scopus*, e na área de imputação de dados. Nesta consulta, somente foram considerados trabalhos que aprimoram o processo de imputação em geral. Observamos que a maioria dos trabalhos são voltados para imputação simples, sendo poucos relacionados a imputação múltipla (Seção 2.4). Após a revisão de 445 artigos, 34 foram descartados por não estudar efetivamente a imputação de dados e somente utilizá-la; 203 foram classificados como aprimoramentos do processo, sendo 159 abordando imputação simples, 44 imputação múltipla e apenas cinco utilizando métodos ensemble. Com objetivo de refinar os resultados, foi realizada uma nova pesquisa, apresentada no Apêndice A.2, na mesma base de dados, considerando desta vez a utilização de métodos comitês. Após a revisão de 63 artigos, nove foram considerados relevantes, totalizando 14 artigos como diretamente correlatos ao tema desta pesquisa.

3.1- Imputação de Dados

Como citado na seção 2.4, a imputação de dados pode ser definida como o processo de substituição de valores ausentes por valores próximos aos reais em um conjunto de dados. Na técnica *Hot-Deck*, a tarefa de agrupamento é realizada antes da imputação em si. Souza et al. [2018] realizaram um estudo utilizando esta técnica e, em seus experimentos, foram combinados dois algoritmos de aprendizado de máquina, o *k-means* [Han et al., 2011] e o *k*-Nearest Neighbor (k-NN) [Han et al., 2011], comparando seus resultados com a média simples para imputação. Os experimentos foram executados em três conjuntos de dados com percentuais de ausência que variavam entre 10% e 50%.

Com 40% de ausência obteve um erro médio de 9,32% com a utilização do k-NN. Em outro conjunto de dados, com 10% de ausência, foi percebido 37,46% de erro médio. Em ambos os experimentos, os resultados foram melhores com a utilização do k-NN.

Inspirado na imputação *hot-deck*, Soares [2007] propôs um estudo baseado em estratégias de imputação de dados, onde cada estratégia refletia a aplicação, de forma sequencial, das tarefas de seleção de atributos e agrupamento de dados, precedendo a tarefa de imputação. Em seus estudos, Soares [2007] realizou testes nos conjuntos de dados *Iris Plants*, *Pima Indians Diabetes* e *Wisconsin Breast Cancer*, variando os percentuais de ausências de 10% a 50% com saltos de 10%. A imputação foi feita com os algoritmos *k-NN*, *Redes Neurais Back Propagation* e a média. Soares [2007] conclui que em todas as situações, a estratégia composta de agrupamento de dados seguida da seleção dos atributos mais importantes é a que mostrou melhores resultados e o algoritmo *k-NN* mostrou os melhore resultados em todos os conjuntos de dados.

Influenciada pelos resultados propostos por Soares [2007], Ferlin [2008] propõe um processo alternativo de imputação *hot-deck*. Nele, o conjunto de dados é dividido de acordo com sua morfologia de ausência [1] antes do processo de imputação. O conjunto de dados completo C_1 imputado anteriormente é considerado na imputação seguinte, gerando o conjunto de dados C_2. Este processo recorre até que todo o conjunto esteja completo. Em seus experimentos, Ferlin [2008] utilizou dados de cinco bases de dados pertencentes a *UCI Machine Learning Repository* [Lichman, 2018]. Os percentuais de ausência variavam de 10% a 30%, e os algoritmos utilizados foram o *Self-Organizing Map (SOM)* para agrupamento e o *k-NN* para imputação.

Em outro estudo, Silva-Ramírez et al. [2015] compararam diferentes técnicas de imputação de dados. Inicialmente, foi proposta uma abordagem de imputação simples utilizando o algoritmo *multilayer perceptron* empregando diversas regras de aprendizado. Em seguida, uma outra abordagem, formada pela combinação de *multilayer perceptrons* de múltiplas camadas e o k-NN, foram utilizadas. Técnicas clássicas como a imputação por média simples, moda, regressão e *hot-deck* também foram utilizadas. Todos estes algoritmos foram executados em dezoito conjuntos de dados com percentuais de ausência definidos aleatoriamente. Os resultados apontam o IMLP e o MIMLP como melhores modelos em 14 dos 18 conjuntos de dados.

Utilizando as informações da *UCI Machine Learning Repository*, Poulos and

[1] Conceito proposto por Ferlin [2008] para descrever a distribuição de valores presentes e ausentes nos atributos de um conjunto de dados.

Valle [2018] analisaram o processo de imputação de dados ausentes para aprendizado supervisionado [Lichman, 2018]. Foram realizadas comparações em relação à qualidade do modelo gerado a partir do conjuntos de dados com valores ausentes e dados imputados pelos seguintes algoritmos: k-NN, regressão logística, *random forest*, *support vector machines*, moda, e atualização aleatória. Em cada conjunto foram definidas ausências que variavam de 10% a 40%, utilizando os algoritmos *Artificial Neural Networks (ANN)*, *Decision Trees* e *Random Forest* para gerar modelos preditivos. Os resultados apontaram que a estratégia de imputação depende da natureza e da proporção dos valores ausentes, sendo o k-NN com o menor erro.

Em outro estudo, Krause et al. [2018] analisaram o uso da imputação múltipla em dados de rede [2]. Para a realização dos experimentos, foram criadas redes de dados artificiais. Os autores identificaram que as principais causas dos dados serem simulados são: i) controle das fronteiras; ii) complexidade dos relacionamentos; iv) controle dos processos de geração de dados; e iv) garantia de não haver dados nulos para realização dos experimentos. Foram definidas ausências respeitando três mecanismos (MAR, MCAR e MNAR), com percentuais variando entre 10% a 50%, concluindo que a utilização do algoritmo *Bayesian ERGM* para imputação múltipla é superior em relação aos algoritmos tradicionalmente empregados.

3.2- Imputação com Classificadores Ensemble

Kumar et al. [2016] realizaram um estudo relacionado à imputação de dados Quality of Service (QoS), utilizando técnicas ensemble. O objetivo era comparar a imputação pelo algoritmo *Bagging* e *Additive Regression (boosting)*. Os algoritmos utilizados como *base learning* foram: *Random Forest (RF)*; *K-Nearest Neighbor (k-NN)*; *Radial Basis Function Network (RBF)*; *Linear Regression (LR)*; *Reduced Error Pruning Tree (REP)*; *Support Vector Machine for Regression (SMOreg)*; e *Multilayer Perceptron (MLP)*. No experimento, o conjunto de dados foi dividido entre 70% para treino e 30% para teste. Para analisar o desvio, foram utilizadas técnicas como o coeficiente de correlação, média do erro absoluto e média dos erros quadrados. Os resultados indicaram melhor

[2]Esses dados são caracterizados por possuírem nós e relacionamentos. Os nós podem se pessoas, empresas ou países e relacionamentos podem ser qualquer forma de interação entre os nós.

acurácia do algoritmo *Random Forest* na predição dos valores ausentes.

No estudo realizado por Valdiviezo and Van Aelst [2015] foram comparadas diversas estratégias de predição baseadas em árvores. A utilização de árvores se deve por apresentarem bons resultados com diversos tipos de dados (categóricos, discretos e contínuos), e a aplicação de *Random Forest* pela facilidade de manipular problemas de alta dimensão. Na pesquisa, foram combinados os algoritmos *Classification and Regression Tree (CART)*, *Conditional Inference Tree (CondTree)*, *RF*, *Conditional Random Forests (CondRF)*, *Multiple Imputation by Chained Equations (MICE)*, *Multiple Imputation by Sequential Regression Trees (MIST)*, *Bagging* e *Conditional Bagging (CondBagging)* empregando estratégias de imputação simples, imputação múltipla e substituição e os algoritmos: média; matriz de proximidade; k-NN; MICE e MIST. Nos experimentos, foram definidos percentuais de ausências que variavam entre 10% a 40%. Os autores concluíram que, para quantidades significativas de dados ausentes, é aconselhável a imputação múltipla por MICE ou MIST seguida por CondRF. Embora essas técnicas demandem tempo computacional, o algoritmo CondBagging com substituição apresenta melhor desempenho e menor tempo. Para quantidades de dados ausentes menores, qualquer método conjunto com substituição ou precedido por imputação simples é suficiente para obter um bom desempenho.

Gad et al. [2017] compararam diversos algoritmos para imputação de dados como *Linear Regression*, *Random Forest*, *SVM* e *k-NN*. Foram utilizados conjuntos de dados reais e métricas de desvio padrão, variância, erro absoluto médio, erro quadrático médio, e coeficientes de determinação R2. No modelo preditivo, foi considerado apenas um subconjunto completo de dados (retirando as instâncias com dados ausentes). Segundo os autores, os resultados variam de acordo com a técnica utilizada, não havendo diferenças significativas com percentuais de ausência menores. Apesar de todos os métodos de imputação apresentarem desempenhos satisfatórios, pode-se observar que o algoritmo *Random Forest* apresenta melhores resultados.

Lu et al. [2011] realizaram um estudo de imputação múltipla com modelos de predição ensemble. Foi utilizado um conjunto de dados com 359.829 registros providos pelo governo chinês e com percentual de ausência de 51,3%. As técnicas de *Bagging* e *Boosting* foram aplicadas no modelo, sendo a primeira superior em relação a outra. Em outro estudo, Hassan et al. [2007] empregaram métodos ensemble tendo redes neurais como base (Seção 2.8). Foram realizados experimentos através de duas abordagens

para a distribuição de dados ausentes: abordagem univariadas (utilizando distribuição incondicional); e multivariadas (utilizando distribuição condicional).

Os experimentos foram realizados em dois conjuntos de dados bancários (Bank Dataset e Housing Dataset)[3], com percentuais de ausências variando entre 5% à 50%. Os autores realizaram uma comparação de três métodos ensemble (*Univariate Ensemble Method (UVE)*, *Multivariate Ensemble Method (MVE)* e *Multivariate Ensemble method with Output Augmentation (MVO)*) com os métodos de imputação *Casewise Deletion (CWD)*, *Mean Substitution (MS)* e *Expectation Maximization (EM)*. Os métodos UVE, MVE e MVO tiveram menores taxas de erros, sendo o MVO superior em relação aos outros.

Twala and Cartwright [2010] propuseram uma abordagem de ensemble utilizando *Bagging* com *Decision Trees* como classificador e percentuais de ausências iguais a 15%, 30% e 50%. Para avaliar a precisão, os autores utilizaram valores contínuos que variavam de zero (totalmente acertado) até um (totalmente errado). Twala and Cartwright [2010] apontam que esta abordagem melhora a qualidade de resposta, diminuindo a variância do erro. A precisão do método foi comparado com outros que não utilizavam ensemble e, concluíram, que a abordagem ensemble tem um desempenho superior em relação à precisão do dado imputado.

Saitoh [2016] apresenta uma abordagem de imputação com métodos ensemble utilizando um algoritmo chamado *Self-Organizing Map (SOM)*. O autor realiza uma comparação entre o chamado *SOM* convencional, *K-Means* e o algoritmo proposto. Seus experimentos foram realizados nos conjuntos de dados *Iris* e *Seeds* pertencentes ao *UCI Machine Learning Repository* [Lichman, 2018]. A proporção de ausência dos dados variaram entre 10% até 80% com saltos de 10%, também foi removido o atributo classe dos conjuntos de dados e para calcular a precisão do dado imputado utilizou-se o cálculo da diferença absoluta entre o valor real e o valor imputado. O método proposto pelo autor teve um desempenho melhor, com relação à precisão do dado imputado, comparado com as abordagens do *SOM* convencional e o *K-Means*.

Karanikola and Kotsiantis [2019] realizaram um estudo de imputação de dados onde comparações foram feitas entre resultados obtidos após a execução da imputação, tanto em valores categóricos quanto em valores contínuos. Em um primeiro momento foi realizado a imputação utilizando algoritmos de regressão linear e regressão logística propostos em Templ et al. [2011]. Estes resultados foram comparados com os resultados

[3]http://www.niaad.liacc.up.pt/ ltorgo/Regression/DataSets.html

obtidos após realizar a imputação de dados com alguns algoritmos *ensemble*. Para realizar as comparações propostas no estudo eles implementaram dois algoritmos ensemble:

- Logitboost para imputação de dados categóricos [Friedman et al., 2000] [Schapire and Singer, 1999]
- M5P Regression Trees para a imputação de valores contínuos [Quinlan et al., 1992]

Os experimentos foram realizados em 30 conjuntos de dados públicos disponíveis no *UCI Machine Learning Repository*. Os testes foram feitos nos conjuntos de dados com percentuais de ausências variando de 20% até 50% com saltos de 5%. As ausências foram realizadas a partir de remoções aleatórias dos dados. Os resultados obtidos foram comparados não só com os algoritmos de regressão linear e logística como também com o K-NN, média e moda. Não foi realizado nenhuma forma de comparação entre algoritmos ensemble. Eles concluem que os resultados obtidos pelos métodos ensemble utilizados superaram todas as outras formas de imputação comparadas mesmo com um percentual de ausência grande (50%).

Já Madhu et al. [2019] buscando melhorar os resultados na imputação de dados utilizando o algoritmo baseado em árvores de decisão *missForest*, realizaram um estudo de imputação de dados com o algoritmos *XGBoost*. Em seus estudos, Madhu et al. [2019] procuraram demonstrar uma análise geral e o impacto do algoritmo *XGBoost* em relação a outros métodos de imputação como o *missForest*, a imputação interativa e a imputação utilizando o *k-NN*. Os testes foram realizados em um grupo de 6 conjuntos de dados voltados para área médica e presente no repositório *KEEL* [Alcalá-Fdez et al., 2011]. A imputação foi realizada nos percentuais de ausência já existente nos conjuntos de dados. As características do conjunto de dados são apresentadas na tabela 2.

Tabela 2 – Sumário dos Conjuntos de Dados

Conjunto de Dados	Quantidade de Colunas	Quantidade de Registros	Classes	Percentual de Ausência
Cleveland	13	303	5	1,98%
Diabetes	8	768	2	50,65%
Dermatology	34	366	6	2,19%
Hepatitis	19	155	2	48,39%
Mammographic	5	961	2	16,63%
Wisconsin	9	699	2	2,29%

Após análise dos resultados dos experimentos Madhu et al. [2019] concluíram

que a precisão dos resultados no método de imputação proposto são melhores do que os outros métodos tradicionais de imputação realizados na comparação. Apesar de existir a comparação entre os algoritmos ensemble *XGBoost* e *missForest* (variação do árvores aleatórias) o foco do estudo não foi a comparação entre algoritmos ensemble.

Abnane et al. [2019] tiveram uma abordagem um pouco diferente com relação aos métodos ensemble. Em seus experimentos o algoritmo *k-NN* foi utilizado como componente, o teste se deu diferenciando os parâmetros do valor do *k* e do método de mensuração de similaridade em cada execução. Para cada método de mensuração de similaridade utilizado (*Euclidean, Manhattan, Minkowski e Chebyshev*) o valor do *k* variou de 1 a 5.

Os testes foram realizados em seis conjuntos de dados diferentes, com percentuais de ausências gerados artificialmente que variaram de 10% até 90% com saltos de 10%, e com três diferentes mecanismos de ausência (MCAR, MAR e NMAR). Após a execução deste algoritmo os resultados foram comparados com outros dois: *E-k-NN* e *GS-k-NN* (*Grid Search k-NN*). GS-k-NN é uma variação do *k-NN* que se diferencia por executá-lo com diversos parâmetros até achar o melhor resultado, diferentemente do *E-k-NN* proposto que combina os resultados, por meio da média dos obtidos nas diversas execuções, caracterizando *Bagging*.

A utilização do *E-k-NN* apresentou melhores resultados que o *k-NN* independente do mecanismo de ausência. No entanto, *E-k-NN* e *GS-k-NN* tiveram resultados similares quando executados nos mecanismos de ausência MCAR e MAR. Com relação ao mecanismo de ausência NMAR, *E-k-NN* obteve melhores resultados que *GS-k-NN*. Eles concluíram também que, no geral, *E-k-NN* supera *GS-k-NN* pelo fato de ser executado com menos esforço. Não foi apresentado nenhum tipo de comparação entre métodos ensemble.

Xi et al. [2018] realizaram um estudo em um conjunto de dados coletados de *Guangzhou Women and Children Medical Center* que cobria características de 2532 casos da doença Hand foot and mouth (Pé mão boca) e classificados como severo ou não. Este conjunto de dados era utilizado no treinamento de modelos de aprendizado de máquina para realizar a predição da forma severa da doença e dos 2532 registros, 365 (14,41%) eram classificados como severo. Eles criaram um algoritmo de imputação para o conjunto de dados onde após a detecção do tipo de dado da coluna alvo (categórico ou contínuo) a imputação é realizada com o algoritmo adequado. Nos dados categóricos foi

utilizado o *XGBClassification*, já nos dados contínuos foi utilizado o *XGBRegression*. Para a execução dos experimentos o conjunto de dados foi dividido em 70% para treino e 30% para o teste e comparações de desempenho foram feitas com os algoritmos *Random Forest* (Florestas Aleatórias) e Regressão Logística utilizando precision, recall, F1-score e ROC.

Visando melhorar o resultado da imputação em um conjunto de dados relativo a qualidade da água, Ratolojanahary et al. [2019] realizaram uma modificação em um algoritmo chamado MICE [Azur et al., 2011]. O conjunto de dados incompleto (aproximadamente 80% de ausência) usado neste trabalho é retirado de uma análise de amostras de água, feita em Oursbelille, na planície de Adour, no sudoeste da França, de 1991 a 2017. Em seu projeto inicial o algoritmo MICE utilizou regressão linear para realizar a etapa de imputação dos dados, porém, os autores nesta nova versão, resolveram utilizar alguns algoritmos Ensemble para a etapa de imputação como *Random Forest* e *Gradient Boosting*, além de utilizar o k-NN e o Support Vector Regression (SVR). Os autores concluíram que a versão do algoritmo MICE que utiliza SVR foi a melhor pois convergiu mais rapidamente que os outros 3 (*Random Forest, Gradient Boosting e k-NN*) e ofereceu o melhor desempenho em termos do percentual de erro médio.

Suresh et al. [2019] realizaram um estudo comparativo de imputação de dados com diversos algoritmos diferentes. Seu trabalho se deu através da revisão, seleção e comparação de 12 algoritmos com investigações mais detalhadamente sobre seus benefícios e limitações em várias dimensões. Eles concluem, de acordo com os resultados de que os algoritmos *Ensemble* são mais adequados para gerar dados sintéticos que refletem com uma certa precisão a verdade básica. Seus estudos se deram no conjunto de dados *BLADE* que pertence ao governo australiano e contém dados estatísticos que combina informações sobre impostos comerciais e pesquisa com dados sobre o uso de programas governamentais em todas as empresas australianas ativas do FY2001-02 até FY2015-16. Este conjunto de dados contém 28 colunas categóricas e contínuas e no FY2015-16 existem 8.094.618 linhas. Dentre os 12 algoritmos testados, seus resultados mostram um melhor desempenho nos algoritmos *Bagging*, *Extra Trees*, *Random Forest*. Apesar de o foco do trabalho não ser a comparação entre algoritmos Ensemble este trabalho realiza algumas comparações entre algoritmos deste tipo.

Nenhum dos trabalhos citados apresentam uma ampla comparação entre algoritmos que utilizam métodos ensemble, e os que realizaram essa comparação utilizaram

apenas *Bagging* e *Boosting*, não variando entre versões mais atuais de *Boosting* como *Adaboost* e *Gradient Boosting*. Também nenhum dos trabalhos apresentaram experimentos utilizando o algoritmo *Stacked Generalization*. Com isso, espera-se que a abordagem apresentada neste trabalho permita que novas conclusões sejam obtidas por meio dos resultados.

A Tabela 3 consolida as técnicas ensemble citadas por autores.

Tabela 3 – Autores e abordagens ensemble de imputação utilizadas.

Autor	Abordagem
Hassan et al. [2007]	Outros (Ensemble Network)
Twala and Cartwright [2010]	Bagging
Lu et al. [2011]	Boosting e Bagging
Valdiviezo and Van Aelst [2015]	Bagging
Saitoh [2016]	Outros (Randomized Decision Trees)
Kumar et al. [2016]	Boosting e Bagging
Gad et al. [2017]	Random Forest
Xi et al. [2018]	Gradientboost e Random Forest
Souza [2019]	Bagging
Karanikola and Kotsiantis [2019]	Boosting e Random Forest
Madhu et al. [2019]	Gradientboost e Random Forest
Abnane et al. [2019]	Bagging
Ratolojanahary et al. [2019]	Gradientboost e Random Forest
Suresh et al. [2019]	Bagging

4- Metodologia

Neste capítulo são abordados os métodos de imputação realizadas neste trabalho, que consiste na utilização de quatro algoritmos comitês combinados à técnica de imputação *hot-deck*. Além disso, será abordado junto com as adaptações necessárias para a realização dos experimentos, o algoritmo Motor de Imputação, desenvolvido por Souza [2019].

4.1- Imputação Hot-Deck e Comitês de Regressão

Na técnica de Imputação Hot-Deck (Seção 2.6), uma tarefa de agrupamento é executada precedendo a tarefa de imputação. Comitês de regressão (Seção 2.8) têm a capacidade de combinar o resultado de várias experiências de treino em aprendizagem de máquina, proporcionando uma melhor precisão no valor gerado.

Este trabalho tem como propósito comparar o resultado de quatro comitês regressores diferentes (*Adaboost*, *Bagging*, *Gradient Boosting* e *Stacked Generalization*) utilizando o Motor de Imputação desenvolvido por Souza [2019]. Em linhas gerais, este trabalho executa comitês de regressão na tarefa de imputação de uma estratégia.

4.1.1 Algoritmo Bagging

O algoritmo de *Bagging* treina T classificadores independentes, cada um com amostragem de dados gerada através da técnica de *bootstrap*, cujo os dados são sorteados de forma aleatória e com reposição (seção 2.9). Para obter o resultado final os T classificadores são combinados utilizando a média simples sobre seus resultados. O regressor base utilizado para realizar os treinos foi o Arvore de Decisão. O método funciona da seguinte forma:

1. De acordo com o valor do parâmetro *T* (regressor base), são criados *T* subconjuntos do conjunto original, utilizando *bootstrap*.

2. As instâncias do plano de imputação são executadas nos *T* subconjuntos gerados a partir do conjunto de dados original. Este é um ponto de paralelismo proporcionado pelas características do algoritmo, onde cada modelo pode ser treinado de forma independente um do outro.

3. Após a execução, o valor escolhido será a média simples dos resultados obtidos pelos *T* regressores base.

A figura 10 ilustra o funcionamento do algoritmo.

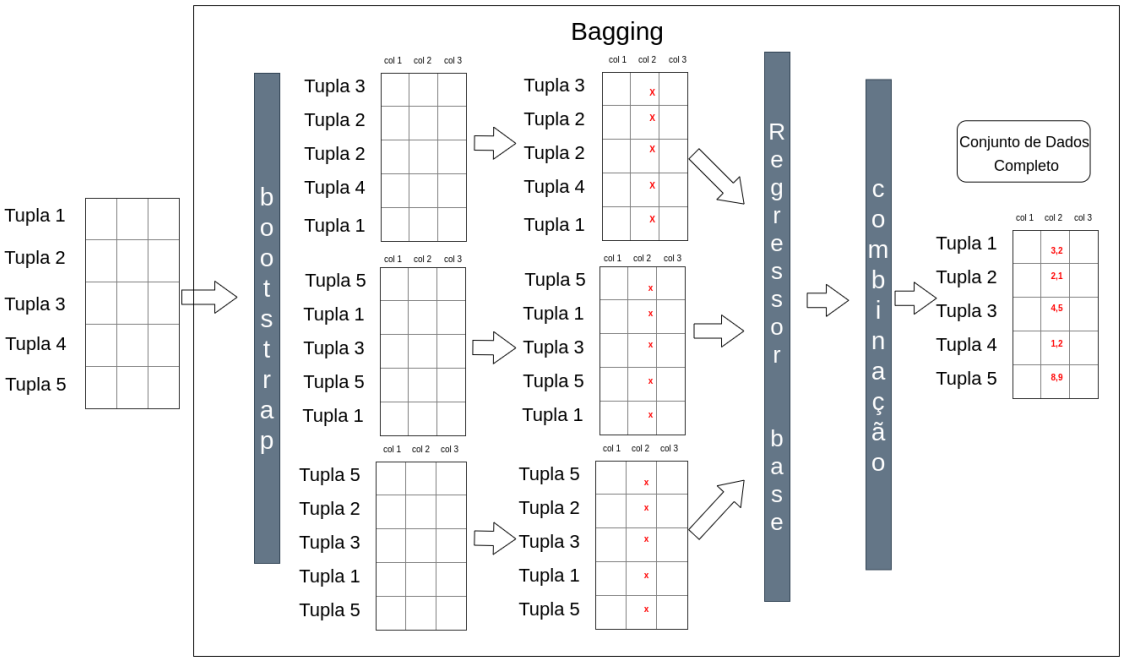

Figura 10 – Imputação de dados com o algoritmo *Bagging*.

4.1.2 Algoritmo Adaboost

O algoritmo de *Adaboost* treina *T* regressores, porém, diferentemente de *Bagging*, estes treinos são realizados de forma sequencial. Pesos são atrelados aos registros e a cada iteração esses pesos são recalculados de acordo com as classificações acertadas e erradas na execução anterior. No caso de acerto, o peso é diminuído e no caso de erro o peso é aumentado. O classificador base utilizado para realizar os treinos foi Árvode de Decisão. O método funciona da seguinte forma:

1. Pesos iguais são associados aos registros do conjunto de dados de treino.
2. As instâncias do plano de imputação são executadas sequencialmente, uma de *T* vezes.
3. Para esta iteração é computado o erro através da função de custo Média dos Erros Quadrados (*Mean Squared Error*).
4. Todos os pesos são atualizados com base nos acertos e erros mensurados pela função de custo.
5. O fluxo volta para a $T^{i-esima}$ execução e o passo a passo se repete até que tudo seja executado *T* vezes.

A figura 11 ilustra o funcionamento do algoritmo.

4.1.3 Algoritmo Gradient Boosting

O algoritmo *Gradient Boosting* treina *T* regressores de forma muito similar ao *Adaboost*, porém utilizando o gradiente descendente para minimizar a função de custo. Os demais passos são os mesmos. O classificador base utilizado para realizar os treinos foi o Árvore de Decisão. O método funciona da seguinte forma:

1. Pesos iguais são associados aos registros do conjunto de dados de treino.

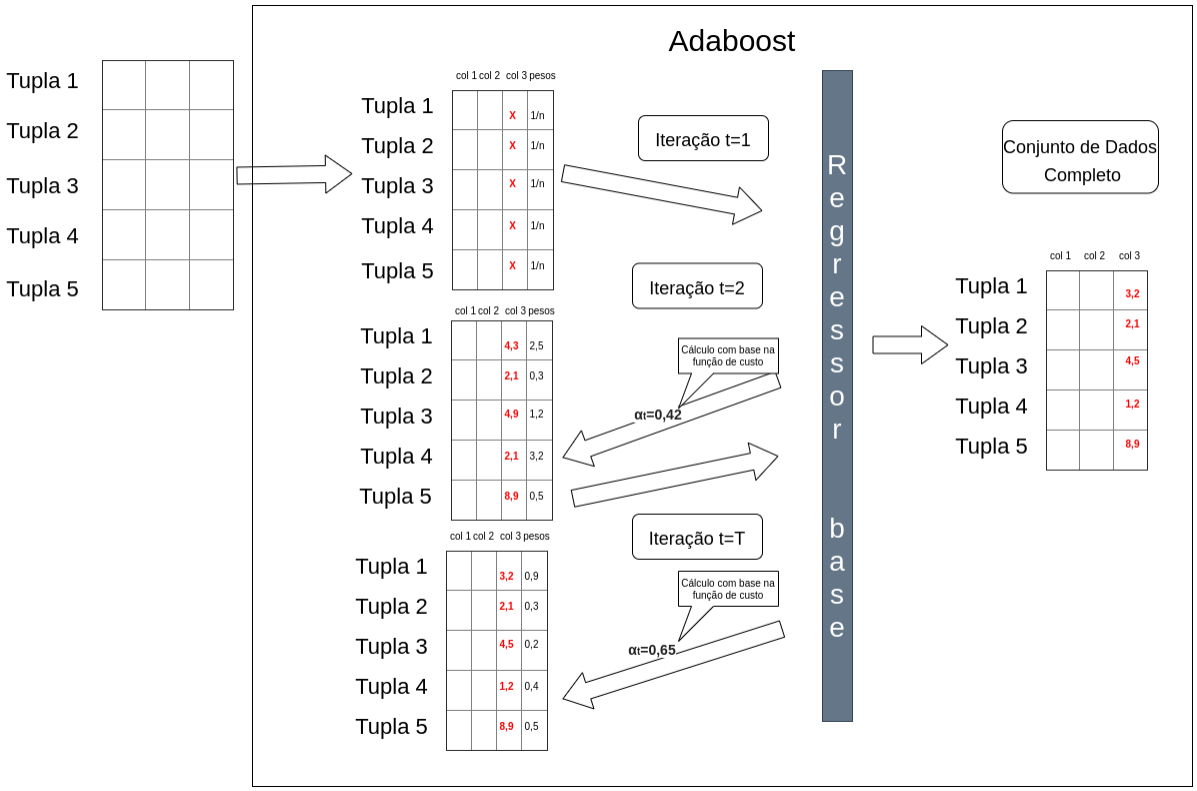

Figura 11 – Imputação de dados com o algoritmo *Adaboost*.

2. As instâncias do plano de imputação são executadas sequencialmente, uma de *T* vezes.

3. Para esta iteração é computado o menor erro através da utilização do gradiente descendente com a função de custo Média dos Erros Quadrados (*Mean Squared Error*).

4. Todos os pesos são atualizados com base nos acertos e erros mensurados pela função de custo.

5. O fluxo volta para a $T^{i-esima}$ execução e o passo a passo se repete até que tudo seja executado *T* vezes.

A figura 12 ilustra o funcionamento do algoritmo.

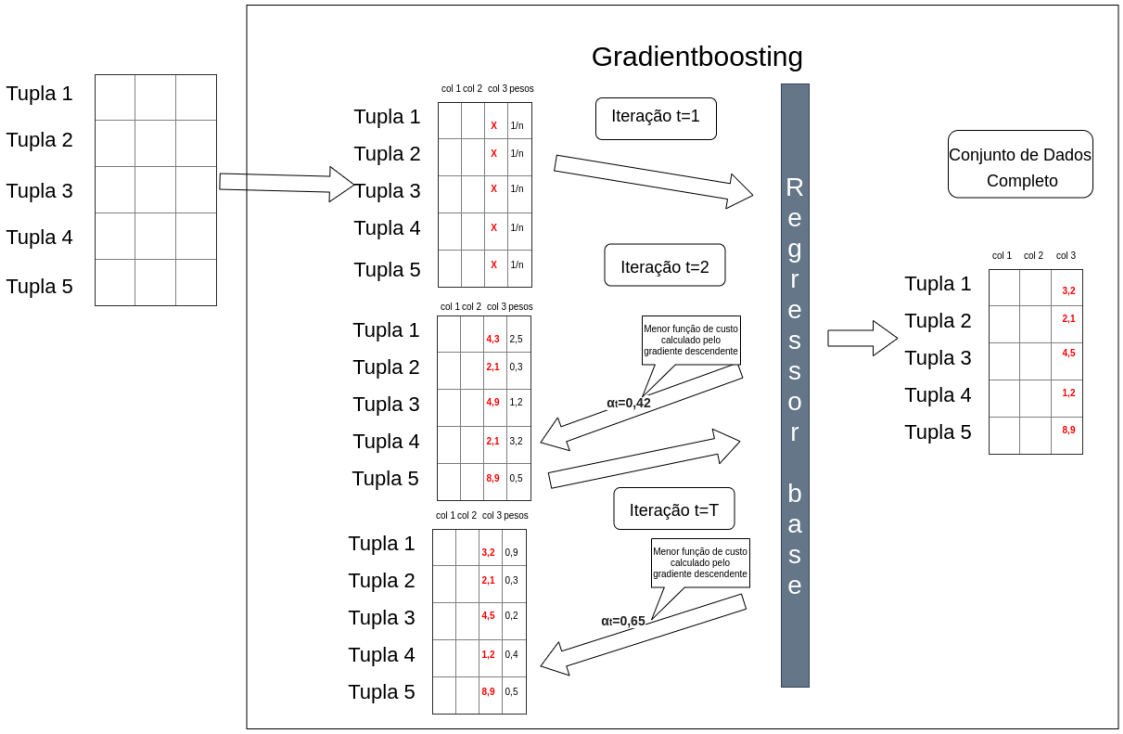

Figura 12 – Imputação de dados com o algoritmo *Gradient Boosting*.

4.1.4 Algoritmo Stacked Generalization

O algoritmo *Stacked Generalization* (Seção 2.10.3) é um caso particular de regressores *ensemble*. *Bagging* adota o *boostrap* para gerar os T subconjuntos do conjunto de dados e assim treinar os T modelos. *Adaboost* associa pesos aos registros do conjunto de dados e a cada iteração aumenta e diminui esses pesos de acordo com erros e acertos, fazendo com que a cada iteração o foco do aprendizado fique nos dados classificados de maneira equivocada anteriormente. *Stacked Generalization* é caracterizado por utilizar uma técnica chamada meta aprendizado, onde os regressor base são chamados de regressores de primeiro nível e um outro regressor chamado de segundo nível é utilizado para treinar a combinação dos regressores de primeiro nível. O regressor base utilizado para induzir os modelos foi *Árvore de Decisão*. O método funciona da seguinte forma:

1. Treine classificadores de primeiro nível com base no conjunto de dados de treinamento original. Neste passo pode-se utilizar diversos algoritmos como regressor base, inclusive outros comitês como *Bagging* e *Adaboost*

2. Construa um novo conjunto de dados com base na saída dos regressores de

primeiro nível. Aqui, os resultados previstos de saída dos regressores de primeiro nível são considerados no novo conjunto de dados. Suponha que cada exemplo em D é $\{x_i, y_i\}$, logo, é construído um exemplo correspondente $\{x'_i, y_i\}$ no novo conjunto de dados, em que $x_i = \{h_1(x_i), h_2(x_i), ..., h_T(x_i)\}$.

3. Treine um regressor de segundo nível com base no conjunto de dados recém-construído. Qualquer algoritmo de aprendizado de máquina pode ser aplicado para induzir um modelo a partir do regressor de segundo nível.

A figura 13 ilustra o funcionamento do algoritmo.

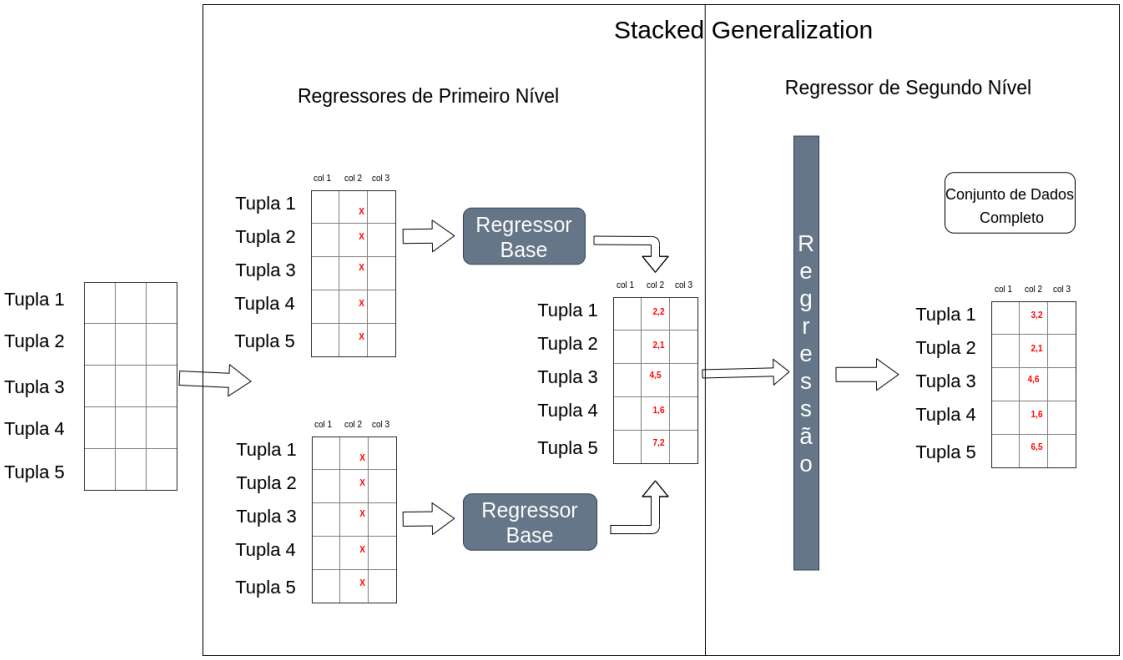

Figura 13 – Imputação de dados com o algoritmo *Stacked Generalization*.

4.2- Motor de Imputação

Souza [2019], desenvolveu o algoritmo Motor de Imputação que tem como objetivo compor e executar os algoritmos comitês no contexto proposto por [Soares, 2007]. Este algoritmo possui dois *pipelines* de processamento: o de imputação, onde são definidos, por exemplo, o regressor base e alguns parâmetros como o valor de *T* e a função de custo; e o de resultado onde ficam armazenados os resultados do processamento.

Segue abaixo o passo a passo do algoritmo para cada Comitê executado neste trabalho:

1. São inicializados os *pipelines* de imputação e resultado.
2. É verificado se o plano de imputação possui a estratégia de *ensemble*.
3. Caso algoritmo seja Bagging.
    3.1. O parâmetro *T* é obtido.
    3.2. *T* novos conjuntos de treinamento são criados utilizando a técnica *Bootstrap* e são associados a *T* cópias do classificador base escolhido.
    3.3. A função de custo configurada é obtida.
    3.4. A função de combinação configurada é obtida.
    3.5. O *pipeline* de imputação é executado e seus resultados são adicionados ao *pipeline* de resultado.
4. Caso algoritmo seja Adaboost ou Gradient Boosting.
    4.1. O parâmetro *T* é obtido.
    4.2. A coluna com os pesos é adicionada ao conjunto de treinamento.
    4.3. A função de custo configurada é obtida.
    4.4. O *pipeline* de imputação é executado e seus resultados são adicionados ao *pipeline* de resultado.
5. Caso algoritmo seja Stacked Generalization.

5.1. Os classificadores de primeiro nível são obtidos com os devidos parâmetros configurados.

5.2. O classificador de segundo nível é obtido com os devidos parâmetros configurados.

5.3. O *pipeline* de imputação é executado e seus resultados são adicionados ao *pipeline* de resultado.

A utilização da biblioteca *Spark-Ensemble* simplificou a utilização do algoritmo Motor de Imputação deixando-o apenas com as responsabilidades de parametrização dos componentes dos comitês, a execução e a obtenção dos resultados do mesmo. Todos os detalhes relacionados ao funcionamento do regressor em si ficaram em seus respectivos algoritmos.

Segue abaixo os respectivos pseudocódigos:

Algoritmo 4 – ALGORITMO MOTOR DE IMPUTAÇÃO

```
Entrada: PI
Saída: RESIMP
PIPEIMP ← inicializaPipelineImp();
PIPERES ← inicializaPipelineRes();
se PI.estrategiaEnsemble() então
    EE ← PI.recuperaEstrategiaEnsemble();
    PAR ← EE.recuperaParametros();
    AE ← EE.recuperaAlgoritmo();
    CT ← PI.recuperaConjuntoTreinamentoOriginal();
    CTA ← SIMULAAUSENCIA(CT, PI.percentualAusencia());
    PIPEIMP.adicionar(AE(CTA, PAR))
senão
    PIPEIMP.adicionar(PI)
fim
RESIMP ← inicializaResultadoImputacao();
RESIMPPL ← PIPEIMP.execImputacao();                    ◁ Paralelismo
 PIPERES.adicionar(RESIMPPL);
RESIMP ← PIPERES.result()
retorna RESIMP
```

A pseudo-codificação do algoritmo Motor de Imputação demonstra como a tarefa de imputação presente no pipeline é composta em tempo de execução, porém ao acionar o método *execImputacao()* do pipeline, o plano de imputação é realizado tendo suas tarefas, estratégias, algoritmos e parâmetros executados na ordem definida pelo cientista de dados. Caso a tarefa de agrupamento seja escolhida, o conjunto de treinamento é

dividido em subconjuntos de acordo com os parâmetros definidos. Este algoritmo foi projetado para executar mais de uma tarefa antecedendo a imputação. Neste sentido, outros algoritmos além do *k-means* podem ser executados. Para os testes deste trabalho, e de acordo com a técnica *hot-deck*, foi utilizado somente o *k-means* precedendo a imputação. O próximo capítulo dissertará acerca das características dos conjuntos de dados utilizados, a arquitetura da solução, e os experimentos realizados.

4.3- Appraisal-Spark

O *Appraisal-Spark* é um *framework* para Apache Spark desenvolvido por Souza [2019] e que tem como objetivo a criação, a execução e o gerenciamento de planos de imputação em larga escala. Este *framework* foi criado baseado no *Appraisal*, solução idealizada por Soares [2007].

Para que a solução pudesse executar os algoritmos *Bagging*, *Adaboost*, *Gradient Boosting* e *Stacked Generalization*, foi necessário desenvolver algumas extensões dentro do *Appraisal-Spark*. O framework *Apache Spark* disponibiliza uma biblioteca de aprendizado de máquina chamada *Machine Learning Library (MLlib)* com diversas implementações de algoritmos diferentes, porém nesta biblioteca não existem todos os algoritmos executados neste trabalho. Desta maneira, visando facilitar a execução dos experimentos e evitar o retrabalho, foi utilizada a biblioteca chamada *Spark-Ensemble* [1]. Esta biblioteca é uma abstração criada em cima da *MLlib* e nela contém os comitês de regressão testados neste trabalho. A utilização deste pacote irá respeitar a arquitetura já existente na solução como um todo, neste sentido, as novas classes implementarão a interface *EnsembleAlgorithm* e após o processo de imputação irá retornar um objeto da classe *EnsembleResult* com os resultados consolidados. A figura 14 ilustra o redesenho da arquitetura do *Appraisal-Spark*. Para medir o desempenho de acerto dos algoritmos será utilizada a técnica conhecida como *Root Mean Squared Error (RMSE)*.

[1] Disponível em: https://github.com/pierrenodet/spark-ensemble

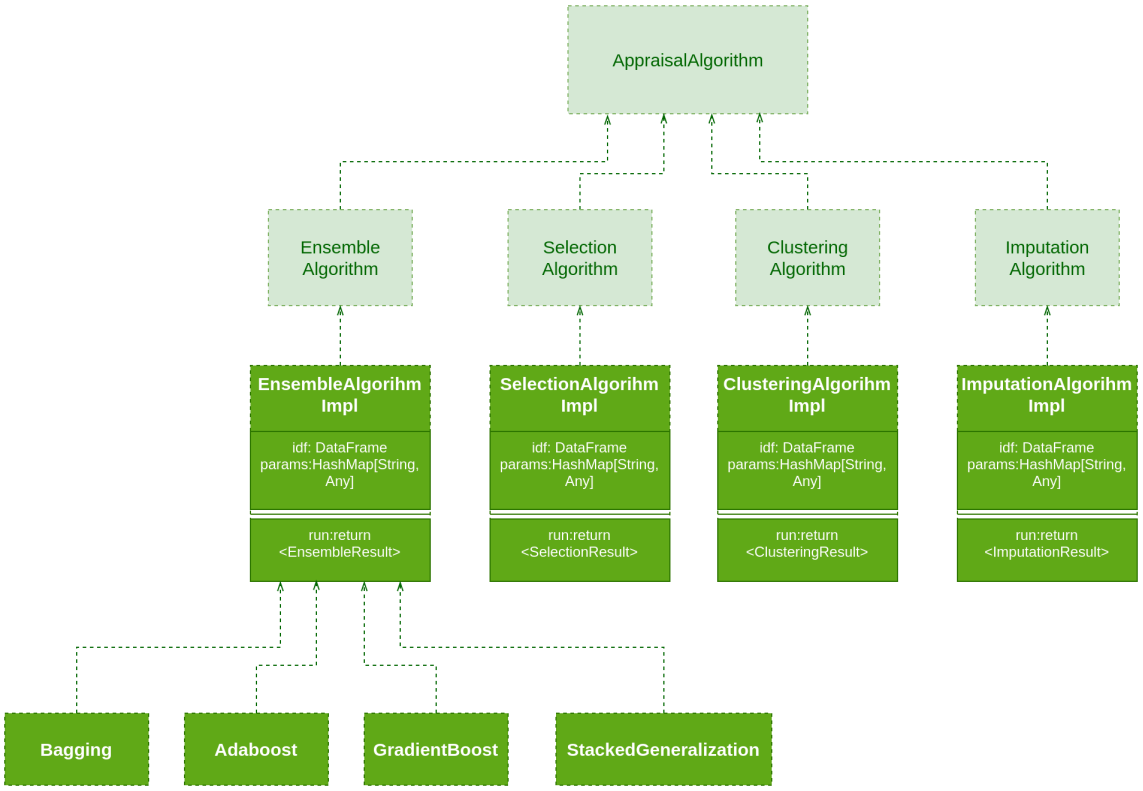

Figura 14 – Redesenho da arquitetura do Appraisal-Spark com as implementações propostas.

5- Avaliação Experimental

Neste capítulo são abordados os assuntos relacionados à avaliação experimental utilizada para a condução da pesquisa, o *framework* Appraisal-Spark, a parametrização dos algoritmos utilizados e a configuração do ambiente computacional utilizado.

5.1- Metodologia

Os testes foram realizados em três bases de dados com características distintas. A ideia foi realizar a imputação hot-deck (Seção 2.6) utilizando os algoritmos *Bagging*, *Adaboost*, *Gradient Boosting* e *Stacked Generalization* (Seções 2.9; 2.10.1; 2.10.2; 2.10.3 respectivamente) com os percentuais de ausências de dados variando de 10% até 30% com saltos de 10%. A configuração das estratégias de imputação será mostrada na seção 5.1.2. Todos os experimentos foram realizados utilizando o *framework* construído por Souza [2019] que é uma adaptação para a solução criada por Soares [2007]. Foi preciso realizar algumas evoluções na solução para que a mesma pudesse suportar os novos experimentos (Seção 5.1.2).

5.1.1 Conjuntos de Dados

Levar em consideração a análise da correlação entre os atributos é de extrema importância para o processo de imputação. Altas correlações tendem a favorecer o resultado da imputação [Soares, 2007]. Neste sentido, os testes foram realizados em três conjuntos de dados com diferentes graus de correlação entre seus atributos. Dois deles disponíveis no repositório da Universidade da Califórnia, Irvine, e o terceiro coletado a partir da Comissão Nacional de Saúde e Planejamento Familiar da China. São eles: *Breast Cancer Wisconsin (Original) Data Set*, *Pima Indians Diabetes* e *AIDS Deaths* -

National Health and Family Planning Commission of China. O conjunto de dados *Breast Cancer Wisconsin (Original) Data Set* possui registros referentes ao diagnóstico de câncer de mama e foram coletados no hospital de *Winscosin*. No conjunto de dados *Pima Indians Diabetes* estão registros referentes aos integrantes de uma tribo indígena conhecida por grande parte dos seus integrantes possuírem diabetes *mellitus*. Já em *AIDS Deaths - National Health and Family Planning Commission of China* os dados foram coletados a partir do relatório mensal da Comissão Nacional de Saúde e Planejamento Familiar da China [Nan and Gao, 2018]. A tabela 4 apresenta o sumário dos conjuntos de dados.

Tabela 4 – Sumário dos Conjuntos de Dados

Conjunto de Dados	Quantidade de atributos	Quantidade de registros	Quantidade de registros após remoção dos valores ausentes
Pima Indians Diabetes	9	768	336
Breast Cancer	11	699	683
Aids	33	78	78

A análise das matrizes de correlação dos atributos dos conjuntos de dados escolhidos (Figuras 15, 16, 17) para os experimentos indica que *Breast Cancer* possui alto percentual de correlação, pois seus atributos apresentam graus maiores do que 50%. Já as bases *Pima Indians Diabetes* e *Aids* apresentam níveis baixos de correlação. A tabela 5 apresenta estes números.

Tabela 5 – Consolidado dos coeficientes de correlações por conjunto de dados.

Conjunto de Dados	Percentual de Correlações Maiores que 50%	Correlação
Pima Indians Diabetes	9.38%	Baixa
Breast Cancer	78.00%	Alta
Aids	71.04%	Alta

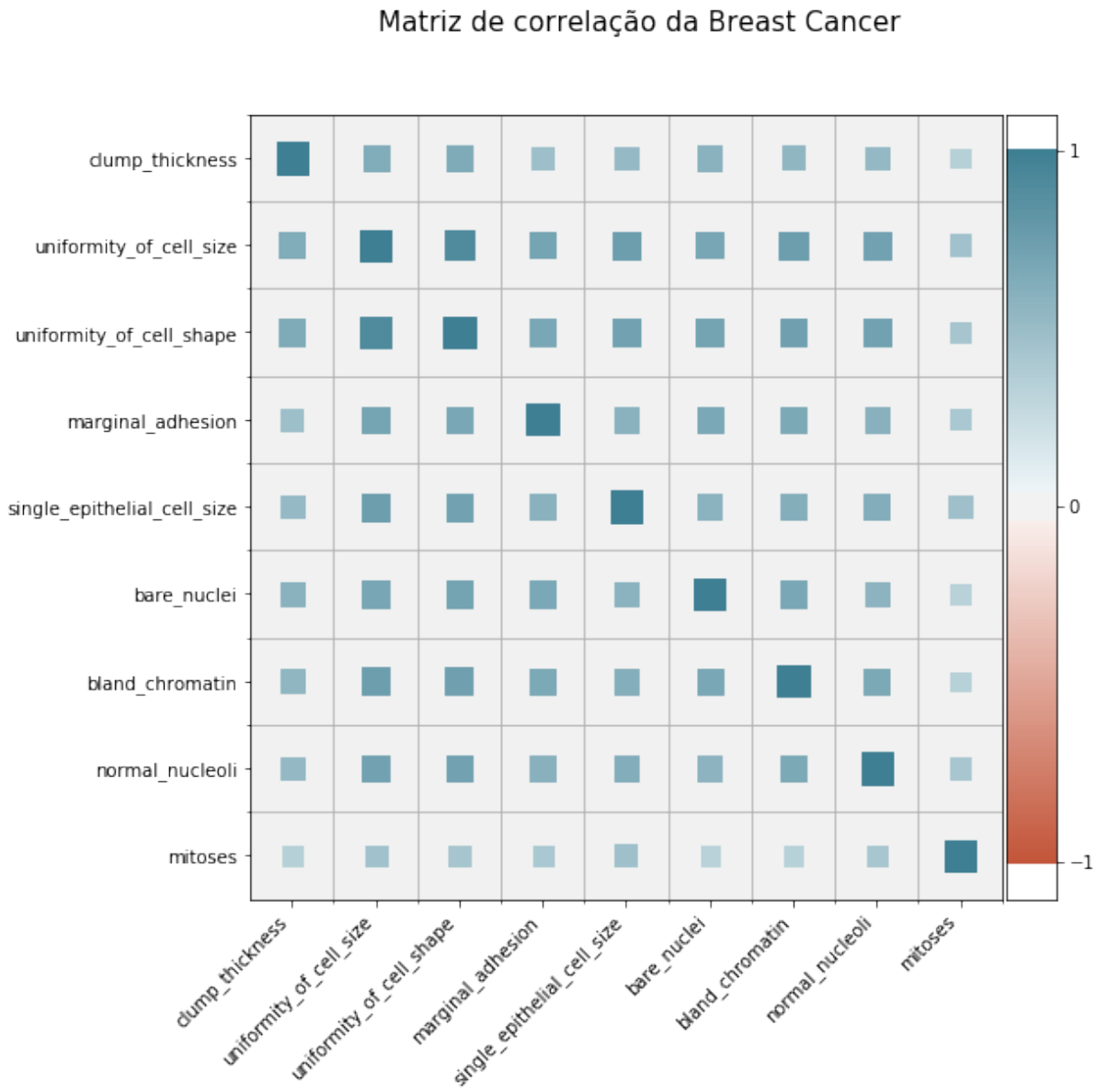

Figura 15 – Matriz de Correlação Breast Cancer.

Pré-Processamento dos dados

Para que os experimentos pudessem ser executados foi preciso realizar alguns tratamentos nos conjuntos de dados. Nos dados da *Breast Cancer* foram observadas 16 instâncias com valores ausentes, essas foram removidas utilizando a técnica *listwise deletion*, totalizando 683 instâncias e o conjunto de dados completo para execução dos experimentos. Com relação ao conjunto de dados *Pima Indians Diabetes* foram observados 432 instâncias que possuíam alguma coluna com valor zero, neste sentido,

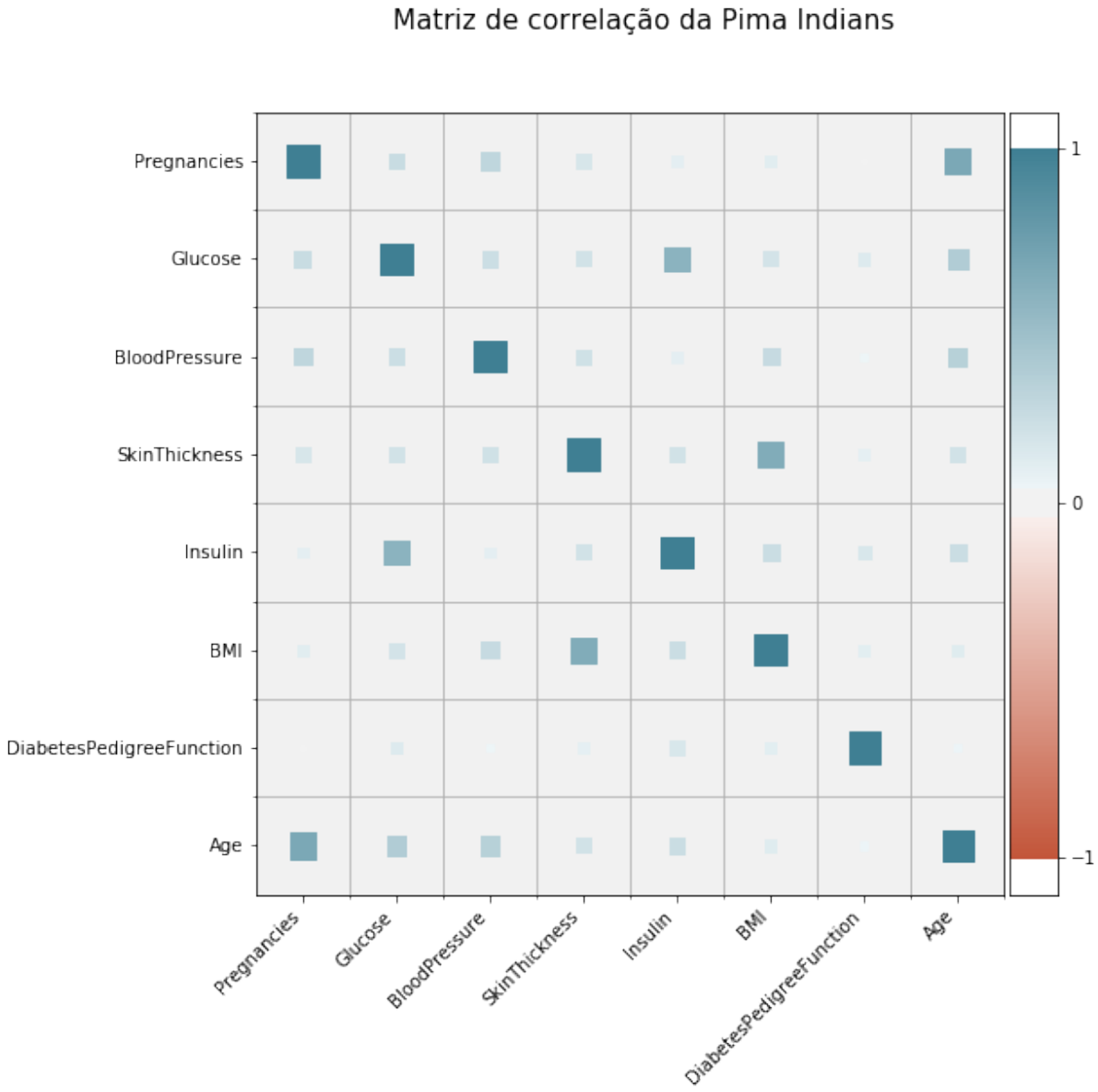

Figura 16 – Matriz de Correlação Pima Indians Diabetes.

esses registros foram removidos também utilizando a técnica *listwise deletion*. Já o conjunto de dados *Aids* não possui valores ausentes. Para que os resultados fossem apresentados em uma mesma ordem de grandeza, foi realizada a técnica de normalização *min-max normalization*:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

Figura 17 – Matriz de Correlação Aids.

5.1.2 Estratégias de Imputação Executadas

Todos os experimentos seguiram os conceitos da imputação simples e *hot-deck*, com isso foram gerados planos de imputação somente com a tarefa de imputação e com a tarefa de agrupamento precedendo a de imputação. Foram realizados os seguintes experimentos:

1. Agrupamento $\longrightarrow$ Imputação[Bagging]: AI[Bagging]

2. Agrupamento $\longrightarrow$ Imputação[Adaboost]: AI[Adaboost]

3. Agrupamento $\longrightarrow$ Imputação[Gradient Boosting]: AI[Gradient Boosting]
4. Agrupamento $\longrightarrow$ Imputação[Stacked Generalization]: AI[Stacked Generalization]
5. Imputação[Bagging]: I[Bagging]
6. Imputação[Adaboost]: I[Adaboost]
7. Imputação[Gradient Boosting]: I[Gradient Boosting]
8. Imputação[Stacked Generalization]: I[Stacked Generalization]

Em todas as estratégias de imputação os percentuais de ausências variaram entre 10%, 20% e 30% cada. Este cenário proporcionou uma quantidade grande de testes.

5.2- Algoritmos

Os algoritmos desta versão do *Appraisal-Spark* utilizam não só a biblioteca *MLlib* como a biblioteca *Spark-Ensemble*. Todos os algoritmos implementados seguem a arquitetura descrita na figura 14 e foram desenvolvidos na linguagem de programação *Scala*. Nesta linguagem, interfaces são chamadas de *traits*, porém para facilitar o entendimento as *traits* serão referidas como interface.

Os algoritmos foram desenvolvidos seguindo os padrões idealizados por Souza [2019], *Bagging*, *Adaboost*, *Gradient Boosting* e *Stacked Generalization* implementam a interface *EnsembleAlgorithm*, esperam como parâmetro de entrada o conjunto de dados de tratamento e a *hash* de parâmetros de configuração. O método *run* retorna a interface *EnsembleResult*. Não houve necessidade de modificações no *K-Means*.

5.2.1 Parametrização dos Algoritmos

Com a finalidade de deixar as comparações mais fidedignas foi utilizado o algoritmo *Decision Tree* como regressor base dos algoritmos comitês executados. Os parâmetros ficaram organizados da seguinte maneira:

- *Bagging*: Decision Tree como Regressor Base; Profundidade da árvore igual a 10; Quantidade de Regressores Base igual a 5;
- *Adaboost*: Decision Tree como Regressor Base; Profundidade da árvore igual a 10; Quantidade de Regressores Base Variável de Acordo com o Valor dos Pesos não ultrapassando a quantidade de 5;
- *Gradient Boosting*: Decision Tree como Regressor Base; Profundidade da árvore igual a 10; Quantidade de Regressores Base Variável de Acordo com o Valor dos Pesos;
- *Stacked Generalization*: Decision Tree como Regressor Base; Profundidade da árvore igual a 10; Quantidade de Regressores Base igual a 5 na aprendizagem de primeiro nível; Decision Tree como regressor na aprendizagem de segundo nível.

5.3- Infraestrutura e Configuração

A implementação da solução foi realizada com o *framework Spark*, versão 2.4.5 em um *cluster* composto por um nó *master* e 3 nós *workers*. Cada nó formado por um processador *Intel Xeon Platinum série 8000*, *quad-core* com 3,1 GHz e 16GB de memória RAM. O sistema operacional utilizado foi o *Linux Red Hat* 7.3.1-6 com *Kernel* versão 4.14.173.

Todos os experimentos foram executados utilizando o paralelismo nativo do *framework* Spark.

5.4- Resultados

Foram realizados diversos testes, cujo escopo de execução se concentrou na imputação simples e *hot-deck* em cada coluna de cada conjunto de dados. Levando em consideração todas as coluna de cada conjunto de dados, todos os percentuais de ausência, técnicas de imputação e cada algoritmo testado, foram realizadas 1176

predições em 173204 segundos. A tabela 6 ilustra estes dados.

Tabela 6 – Quantidades de testes realizados e tempo gasto por conjunto de dados

Conjunto de dados	Técnica	Quantidade de Testes	Tempo de Execução (segundos)
Breast Cancer	Imputação Simples	108	6666
	Imputação Hot-Deck	108	50156
Diabetes	Imputação Simples	96	3746
	Imputação Hot-Deck	96	27579
Aids	Imputação Simples	384	23792
	Imputação Hot-Deck	384	61265
Total		1176	173204

Os resultados indicam que, em geral, a técnica *hot-deck* supera a imputação simples no sentido de gerar valores mais próximos aos reais. Isso indica, que a redução dos desvios de similaridades entre os valores, por meio da divisão dos valores em subgrupos completos que possuam relação de similaridade com os dados ausentes, melhoram a qualidade do valor gerado.

Em todos os testes realizados é possível observar que o algoritmo *Gradientboost* obteve os melhores resultados independente da estratégia de imputação, com exceção em Aids onde foram observados os piores resultados na imputação simples com *Gradientboost*. Com relação ao tempo de processamento, foram observados os menores tempos em *Bagging* e os maiores em *Gradientboost*.

Para facilitar a ilustração dos resultados serão apresentados, em forma de gráficos, as médias de erro e de tempo por conjunto de dados. A próxima seção irá apresentar os resultados obtidos por meio da imputação simples.

5.5- Imputação Simples

Nos testes realizados com a imputação simples, foram observados os melhores tempos (Figuras 21, 22, 23) visto não ser realizado nenhuma tarefa antes da imputação. Porém os resultados não foram tão satisfatórios como os de *hot-deck* (Seção 5.6).

Os melhores resultados, no geral, foram observados nos conjuntos de dados com altos coeficientes de correlação (*Breast Cancer* e *Aids*), na imputação com o algoritmo

Gradientboost, exceto no conjunto de dados Aids, onde os melhores resultados foram observados em *Adaboost*. As seções 5.5.1, 5.5.2, 5.5.3 apresentam esses resultados.

5.5.1 Breast Cancer

No conjunto de dados *Breast Cancer*, pôde-se observar uma baixa média de erro em todos os algoritmos testados, todos abaixo de 0,25 (*RMSE*), independente do percentual de ausência. A diferença de erro variou pouco entre os percentuais de ausências testados em todos os algoritmos. *Gradientboost* aparece com os melhores resultados. O melhor resultado observado foi 0,12 em *Gradientboost* com 20% de ausência e o pior foi 0,25 em *Adaboost* também com 20% de ausência (Figura 19).

Em relação ao tempo de processamento, *Bagging* supera os demais com os menores tempos. Destaque para o teste de imputação com 10% de ausência (Figura 18) onde foi observado um tempo médio de 18 segundos.

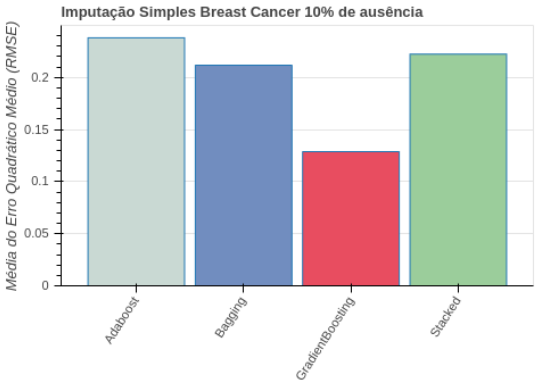

Figura 18 – Média do erro (RMSE), 10% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

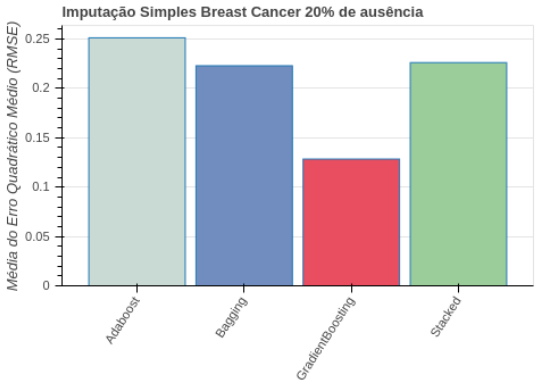

Figura 19 – Média do erro (RMSE), 20% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

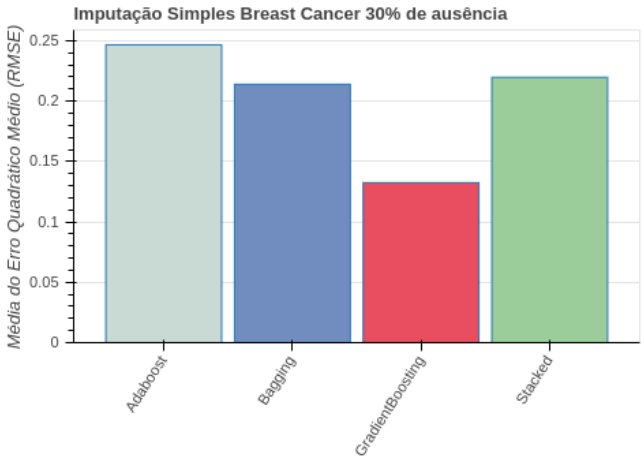

Figura 20 – Média do erro (RMSE), 30% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

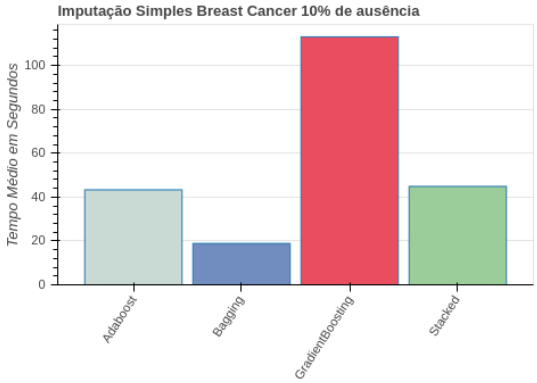

Figura 21 – Tempo médio (segundos) em *Breast Cancer* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

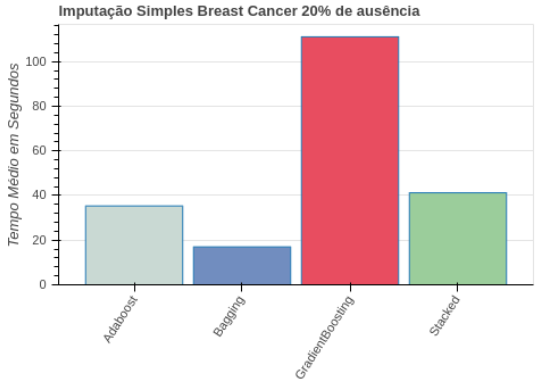

Figura 22 – Tempo médio (segundos) em *Breast Cancer* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

5.5.2 Diabetes

No conjunto de dados Diabetes, pôde-se observar os resultados menos satisfatórios em todos os algoritmos testados, todos abaixo de 0,40 (*RMSE*), independente do percentual de ausência. A diferença de erro também variou pouco entre os percentuais de ausências testados em todos os algoritmos. *Gradientboost* aparece com os melhores resultados. O melhor resultado observado foi 0,16 em *Gradientboost* com 10% de ausência e o pior foi 0,37 em *Adaboost* com 20% de ausência (Figuras 24 e 25).

Em relação ao tempo de processamento, *Bagging* também, supera os demais

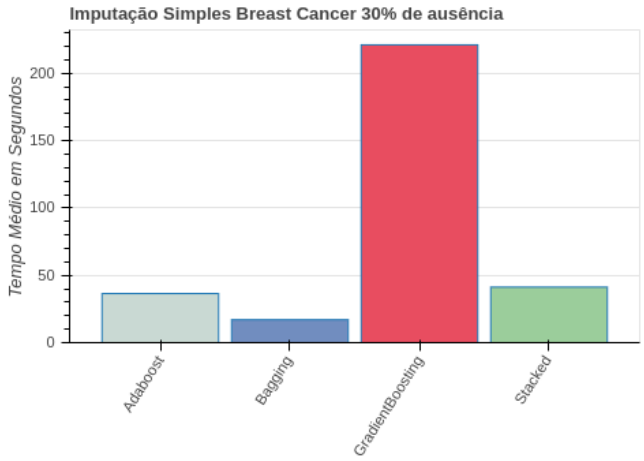

Figura 23 – Tempo médio (segundos) em *Breast Cancer* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

com os menores tempos. Destaque para o teste de imputação com 30% (Figura 29) de ausência onde foi observado um tempo médio de 14,5 segundos.

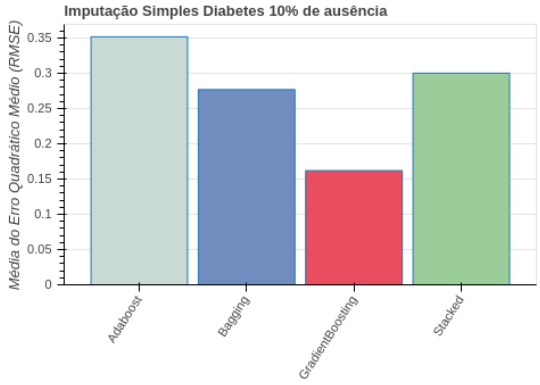

Figura 24 – Média do erro (RMSE), 10% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

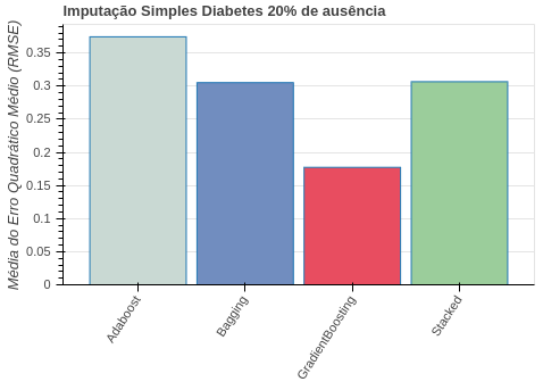

Figura 25 – Média do erro (RMSE), 20% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

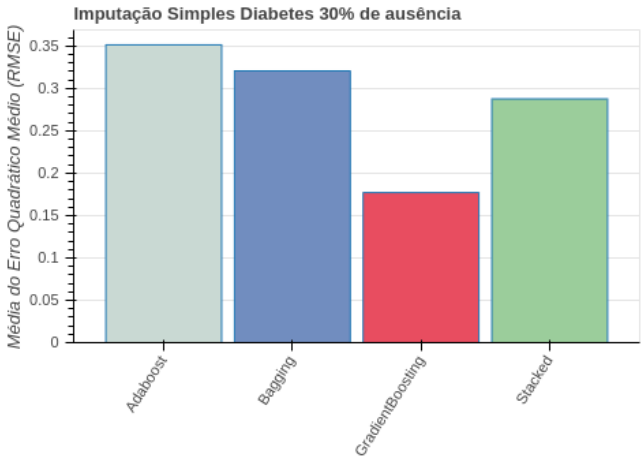

Figura 26 – Média do erro (RMSE), 30% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

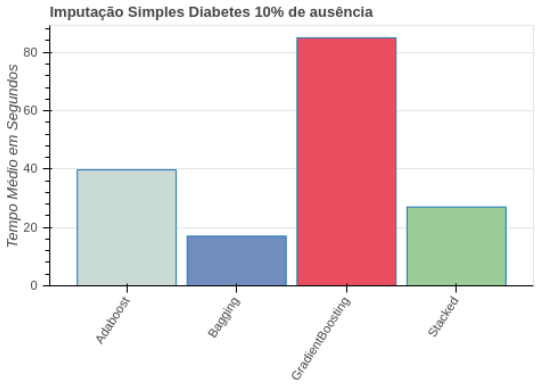

Figura 27 – Tempo médio (segundos) em *Diabetes* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

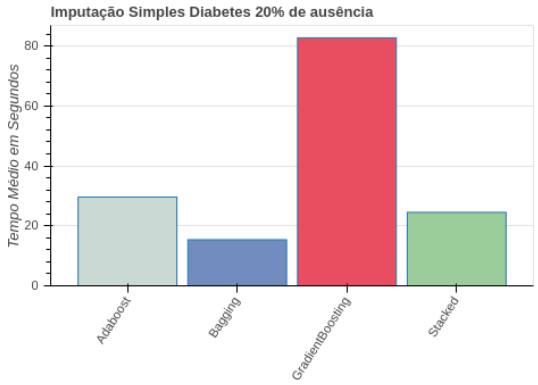

Figura 28 – Tempo médio (segundos) em *Diabetes* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

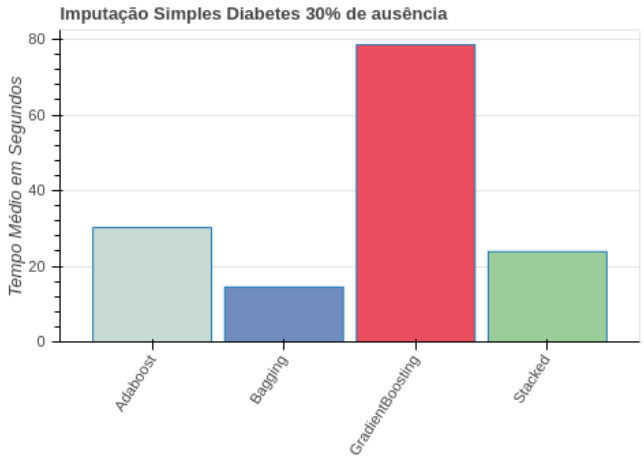

Figura 29 – Tempo médio (segundos) em *Diabetes* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

5.5.3 Aids

No conjunto de dados Aids, observa-se os resultados mais satisfatórios em todos os algoritmos testados, todos abaixo de 0,20 (*RMSE*), independente do percentual de ausência. A diferença de erro também variou pouco entre os percentuais de ausências testados em todos os algoritmos. Diferentemente dos outros conjuntos de dados, *Adaboost* aparece com os melhores resultados. O melhor resultado observado foi 0,14 em *Adaboost* com 10% de ausência e o pior foi 0,19 em *Gradientboost* com 30% de ausência (Figuras 30 e 32).

Apesar de ser o conjunto de dados com a menor quantidade de tuplas, *Aids* obteve, em seu menor tempo médio, um valor maior que os demais algoritmos 29 segundos na imputação com 30% de ausência em *Bagging* (Figura 35).

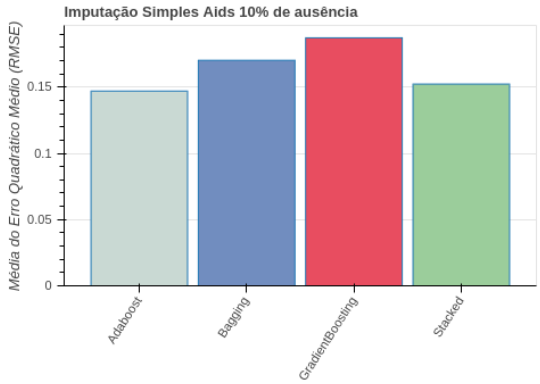

Figura 30 – Média do erro (RMSE), 10% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

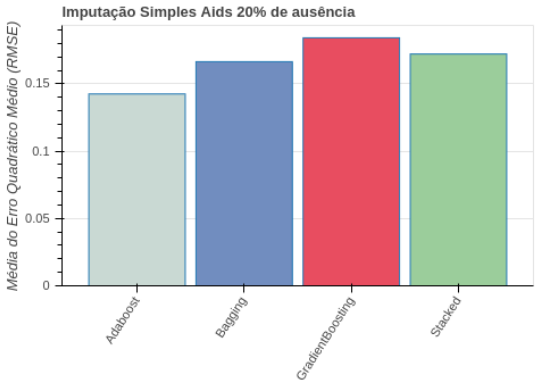

Figura 31 – Média do erro (RMSE), 20% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

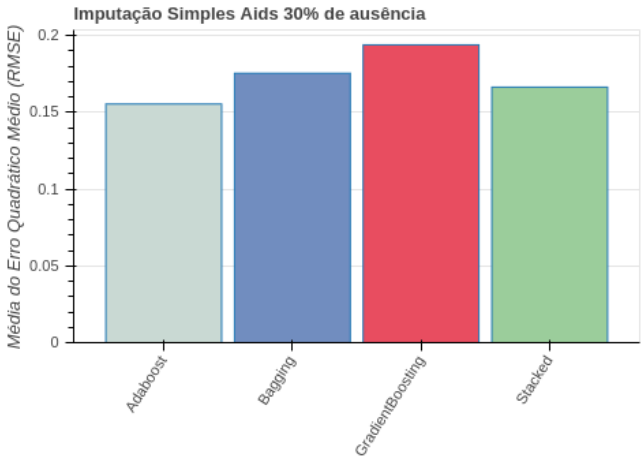

Figura 32 – Média do erro (RMSE), 30% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

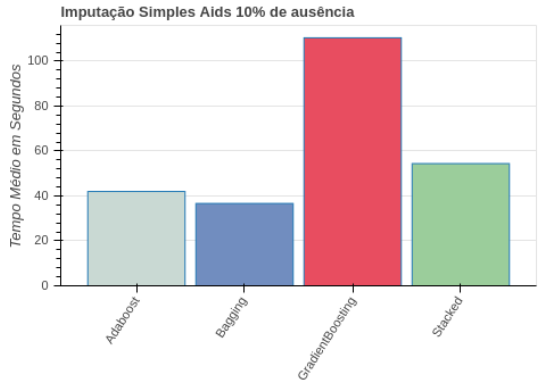

Figura 33 – Tempo médio (segundos) em *Aids* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

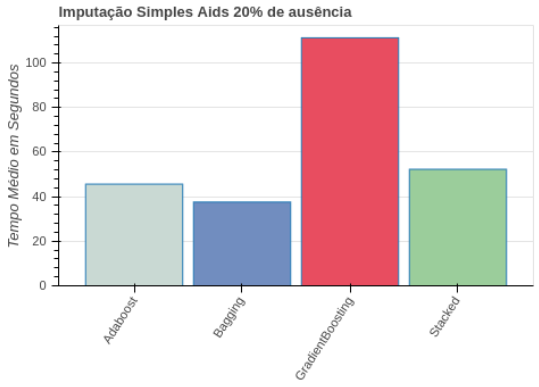

Figura 34 – Tempo médio (segundos) em *Aids* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

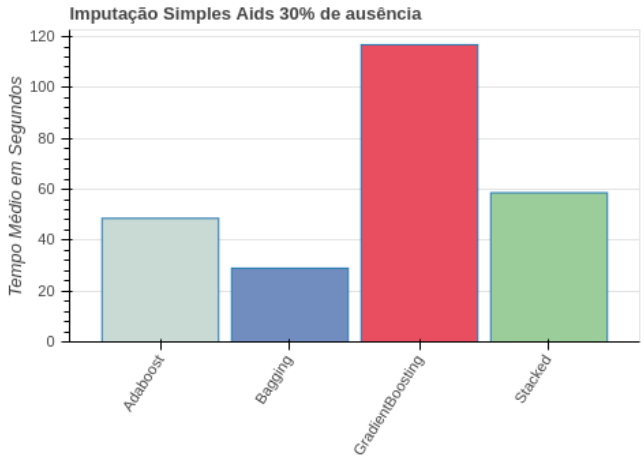

Figura 35 – Tempo médio (segundos) em *Aids* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

5.6- Imputação Hot-Deck

Nos testes realizados com a imputação *hot-deck*, foram observados as menores médias de erro, porém em um tempo de processamento mais elevado. Os melhores resultados, no geral, foram observados também nos conjuntos de dados com altos coeficientes de correlação (*Breast Cancer* e *Aids*), na imputação com o algoritmo *Gradientboost*, em todos os conjuntos de dados. No geral, os resultados foram melhores que a imputação simples.

A imputação *hot-deck* implica na utilização da tarefa de agrupamento precedendo a imputação e apesar dos ganhos relacionados à geração de valores mais próximos aos reais, os tempos de processamento são maiores em relação a imputação simples. As seções 5.6.1, 5.6.2, 5.6.3 apresentam esses resultados.

5.6.1 Breast Cancer

No conjunto de dados *Breast Cancer*, pôde-se observar uma baixa média de erro em todos os algoritmos testados, todos abaixo de 0,22 (*RMSE*), independente do percentual de ausência. A diferença de erro variou pouco entre os percentuais de ausências testados em todos os algoritmos. *Gradientboost* aparece com os melhores resultados. O melhor resultado observado foi 0,11 em *Gradientboost* com 20% de ausência e o pior foi 0,22 em *Bagging* também com 20% de ausência (Figura 37).

Em relação ao tempo de processamento, *Bagging*, supera os demais com os menores tempos. Destaque para o teste de imputação com 30% (Figura 41) de ausência onde foi observado um tempo médio de 154 segundos.

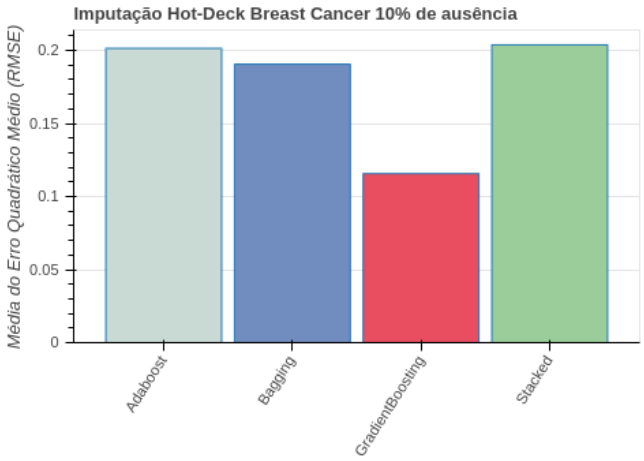

Figura 36 – Média do erro (RMSE), 10% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

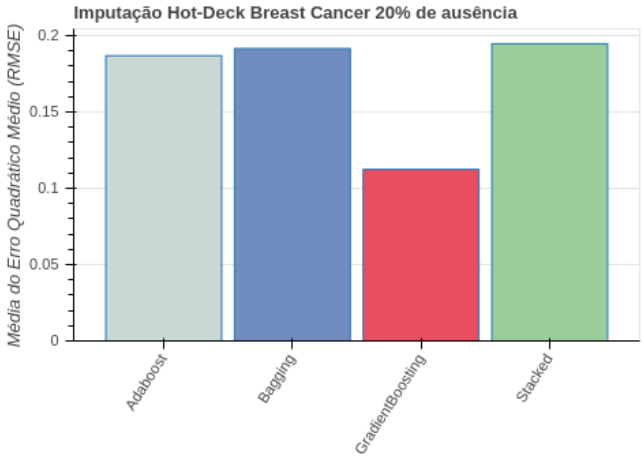

Figura 37 – Média do erro (RMSE), 20% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

5.6.2 Diabetes

No conjunto de dados Diabetes, pôde-se observar uma alta média de erro em relação aos demais conjuntos de dados, em todos os algoritmos testados, independente do percentual de ausência. A diferença de erro variou pouco entre os percentuais de ausências testados em todos os algoritmos. *Gradientboost* aparece com os melhores resultados. O melhor resultado observado foi 0,16 em *Gradientboost* com 10% de ausência e o pior foi 0,26 em *Bagging* com 20% de ausência (Figura 42 e 43).

Em relação ao tempo de processamento, *Bagging* também supera os demais

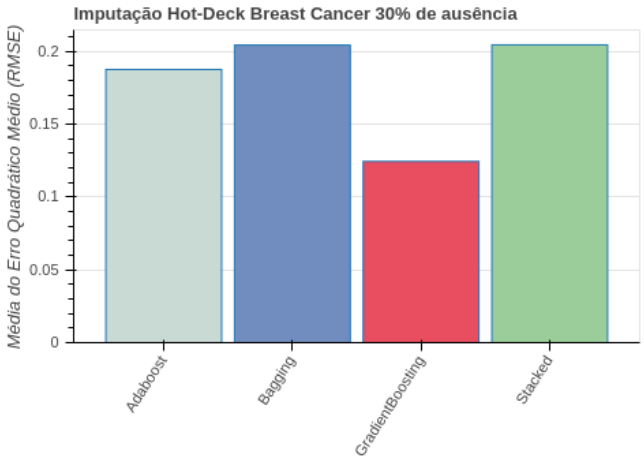

Figura 38 – Média do erro (RMSE), 30% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

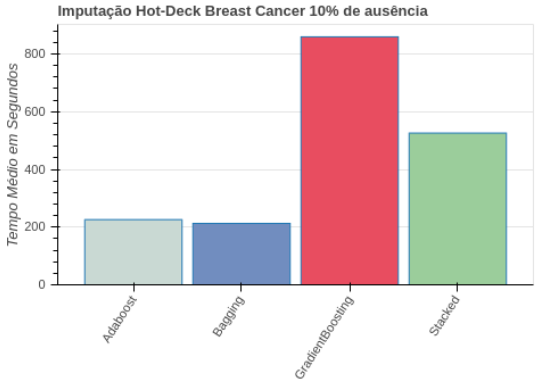

Figura 39 – Tempo médio (segundos) em *Breast Cancer* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

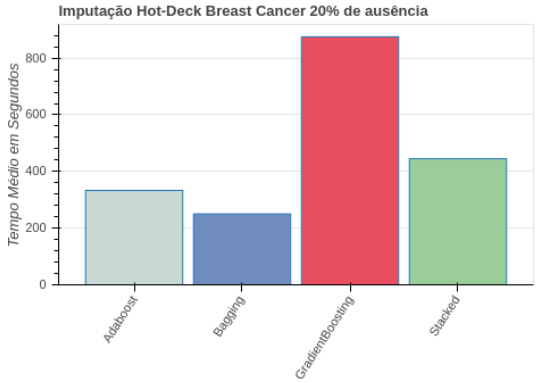

Figura 40 – Tempo médio (segundos) em *Breast Cancer* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

com os menores tempos. Destaque para o teste de imputação com 30% (Figura 47) de ausência onde foi observado um tempo médio de 111 segundos.

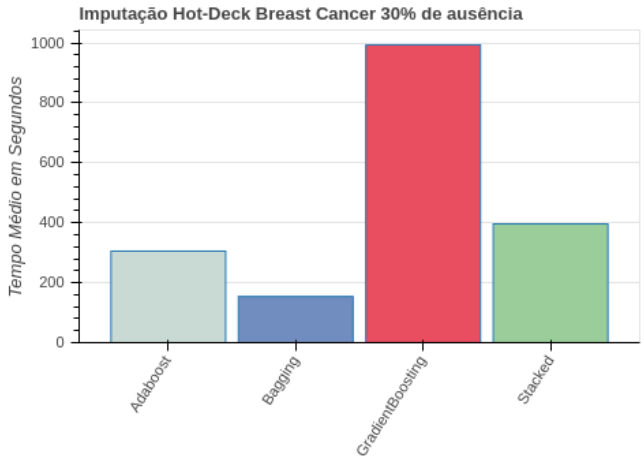

Figura 41 – Tempo médio (segundos) em *Breast Cancer* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

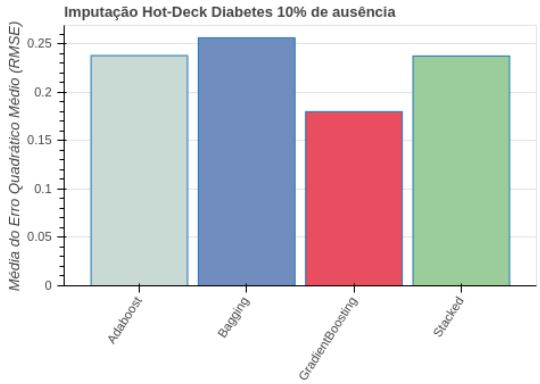

Figura 42 – Média do erro (RMSE), 10% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

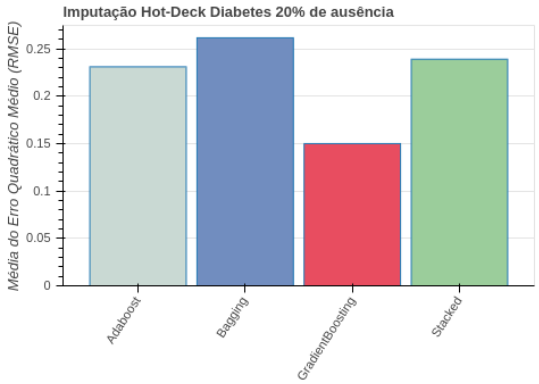

Figura 43 – Média do erro (RMSE), 20% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

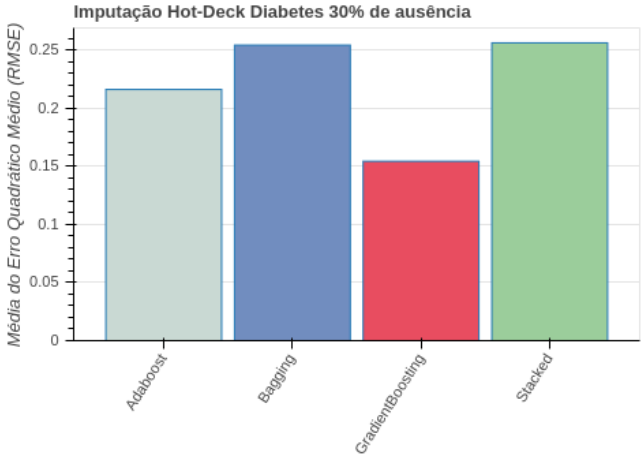

Figura 44 – Média do erro (RMSE), 30% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

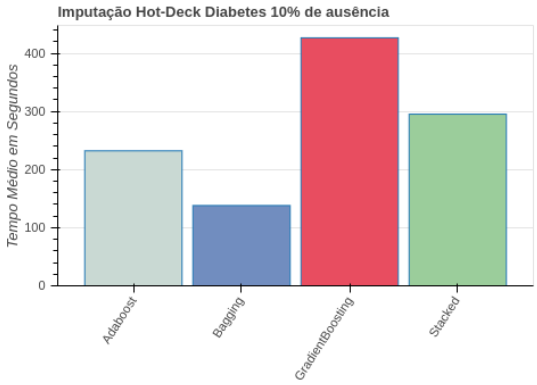

Figura 45 – Tempo médio (segundos) em *Diabetes* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

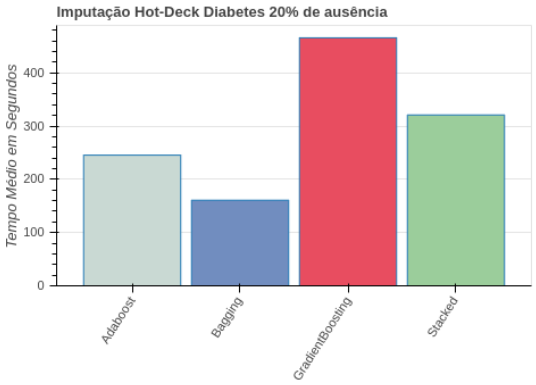

Figura 46 – Tempo médio (segundos) em *Diabetes* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

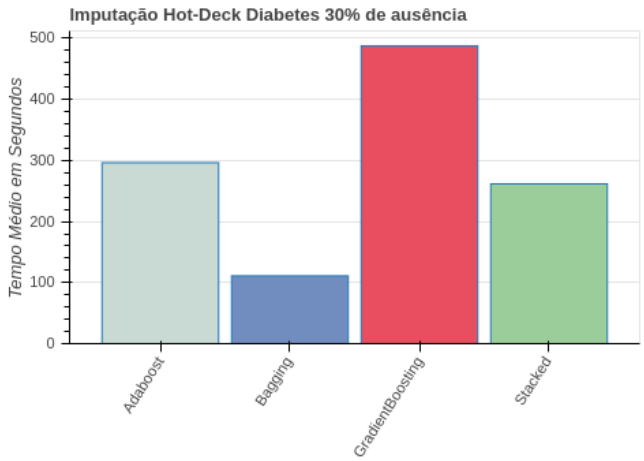

Figura 47 – Tempo médio (segundos) em *Diabetes* com imputação Adaboost, Bagging, Gradientboost e Stacked Generalization

5.6.3 Aids

No conjunto de dados Aids, apesar de possuir poucos registros, observa-se os resultados mais satisfatórios em todos os algoritmos testados, todos abaixo de 0,16 (*RMSE*), independente do percentual de ausência. A diferença de erro também variou pouco entre os percentuais de ausências testados em todos os algoritmos. *Gradientboost* aparece com os melhores resultados. O melhor resultado observado foi 0,11 em *Gradientboost* com 10% de ausência e o pior foi 0,15 em *Bagging* com 30% de ausência (Figuras 48 e 50).

Em relação ao tempo de processamento, diferentemente dos demais conjuntos de dados, em Aids, *Adaboost* supera os demais com os menores tempos. Destaque para o teste de imputação com 30% (Figura 47) de ausência onde foi observado um tempo médio de 111 segundos.

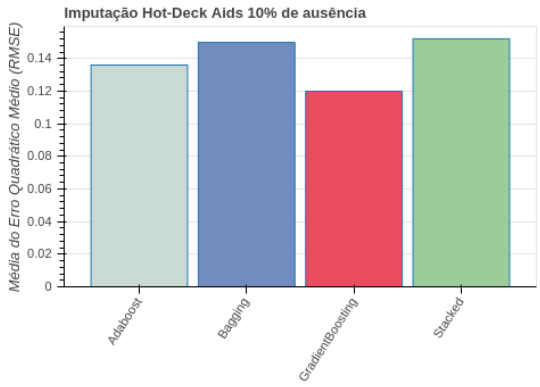

Figura 48 – Média do erro (RMSE), 10% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

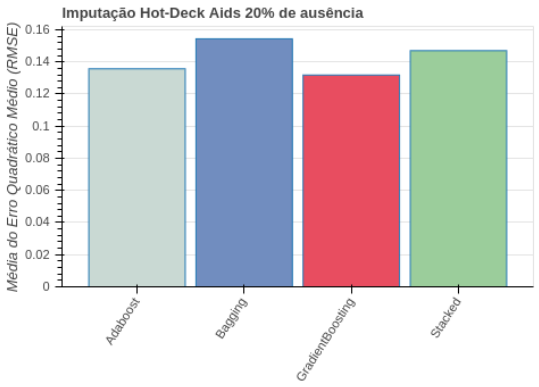

Figura 49 – Média do erro (RMSE), 20% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

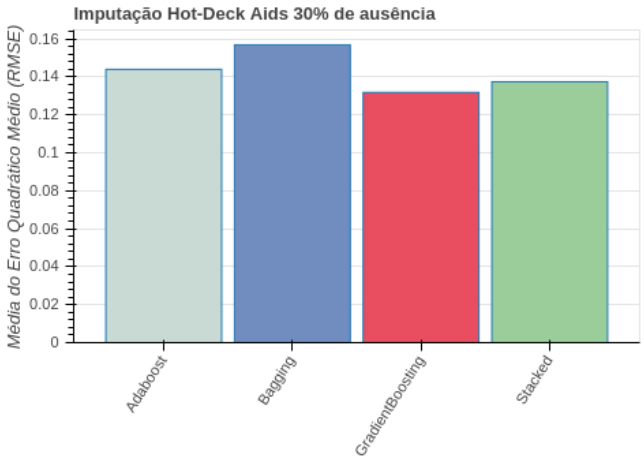

Figura 50 – Média do erro (RMSE), 30% de ausência, com imputação simples utilizando *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*.

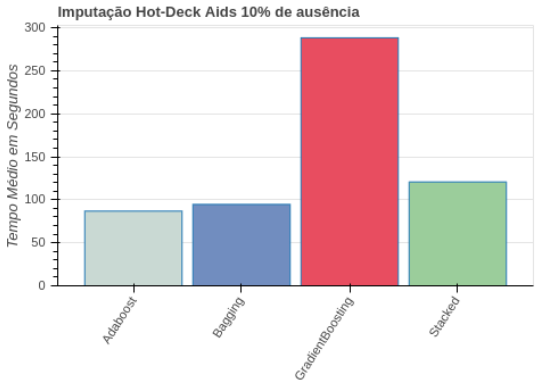

Figura 51 – Tempo médio (segundos) em *Aids* com imputação *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

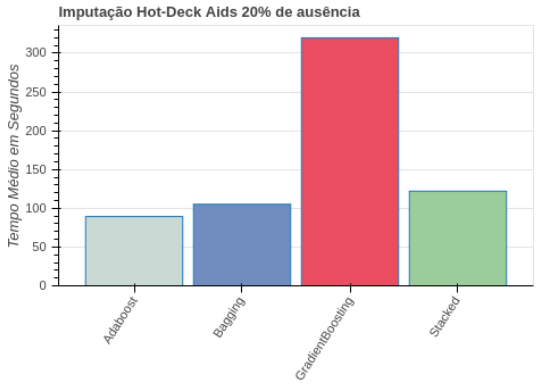

Figura 52 – Tempo médio (segundos) em *Aids* com imputação *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

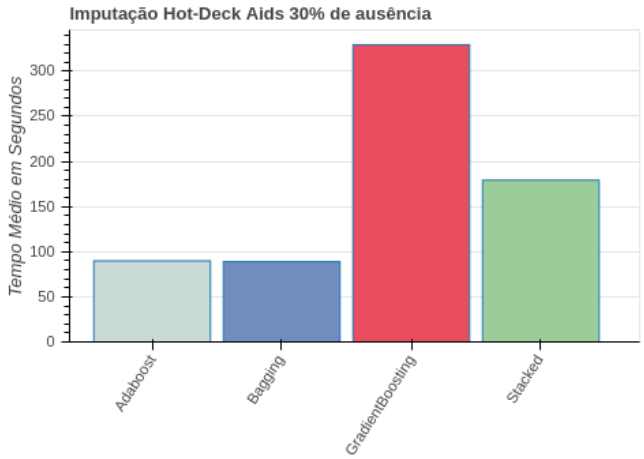

Figura 53 – Tempo médio (segundos) em *Aids* com imputação *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

5.7- Imputação Simples x Hot-Deck

Nesta seção serão apresentados os resultados comparativos entre a imputação simples e a imputação *hot-deck*. Os resultados indicam que a tendência de melhores resultados da imputação *hot-deck* se faz presente. No geral, a técnica *hot-deck* superou a imputação simples em relação a média de erro do valor imputado, exceto em *bagging* com 10% de ausência na *Breast Cancer* (Figura 54) e em *Gradientboost* com 10% de ausência na Diabetes (Figura 57).

5.7.1 Breast Cancer

Os maiores ganhos da imputação *hot-deck* em relação a imputação simples são observados no algoritmo *Adaboost* com destaque no teste com 20% de ausência, gráfico 55, onde o ganho foi mais significativo. Porém *Gradientboost* supera os demais algoritmos em todos os percentuais de ausência nas duas estratégias utilizadas gráficos 54, 55, 56.

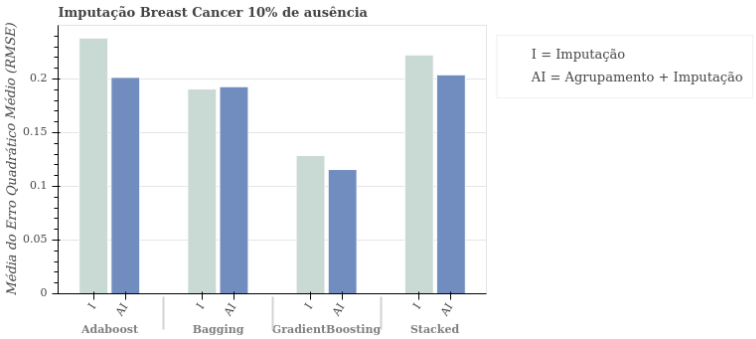

Figura 54 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

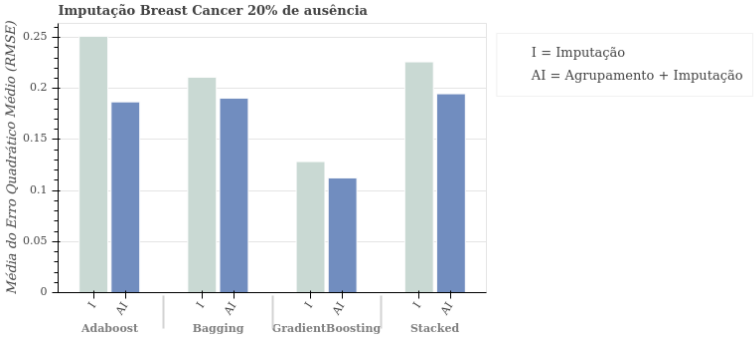

Figura 55 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

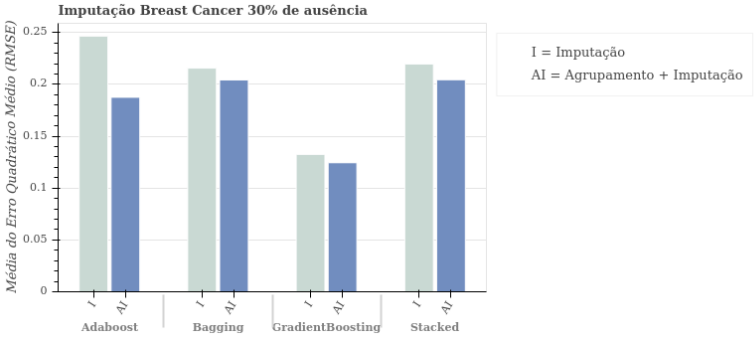

Figura 56 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

5.7.2 Diabetes

Os resultados da imputação no conjunto de dados *Diabetes* não foram tão satisfatórios quanto os observados na *Breast Cancer*.No comparativo entre a imputação *hot-deck* e a imputação simples, *hot-deck* obteve os menores erros.

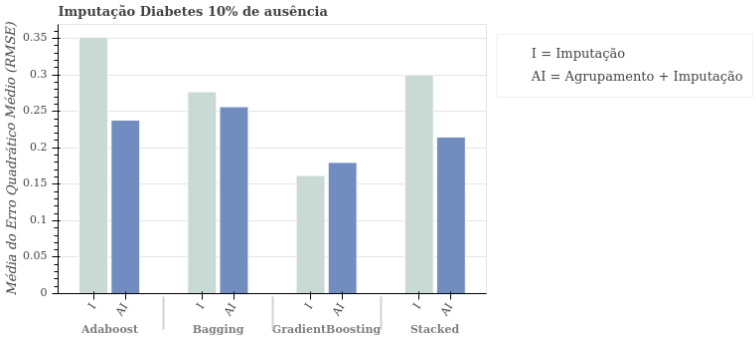

Figura 57 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

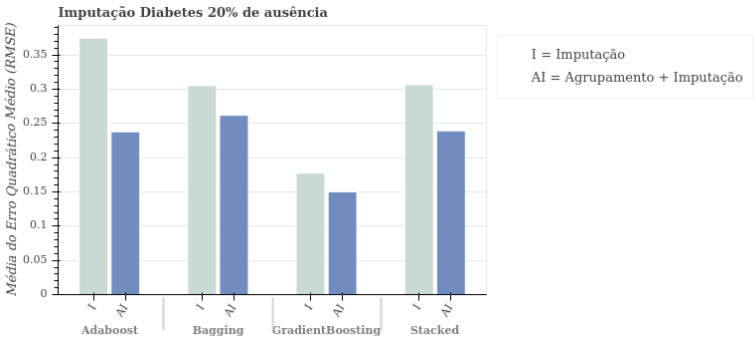

Figura 58 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

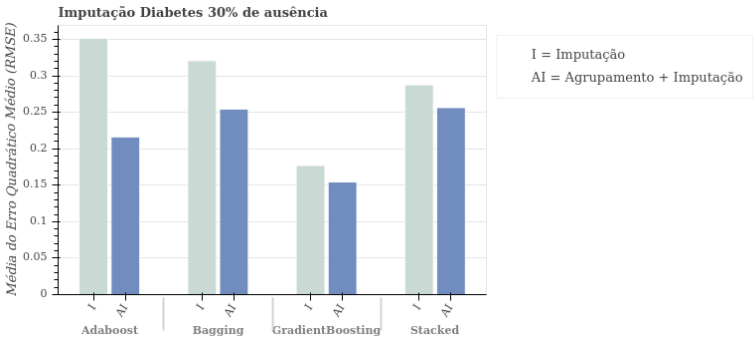

Figura 59 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

5.7.3 Aids

No conjunto de dados Aids, apesar de possuir poucos registros, pôde-se observar os resultados mais satisfatórios em todos os algoritmos testados. A técnica *hot-deck* superou a imputação simples em todos os algoritmos testados com a exceção do algoritmo *Stacked Generalization* onde houve um empate em 0,15 na média de erro.

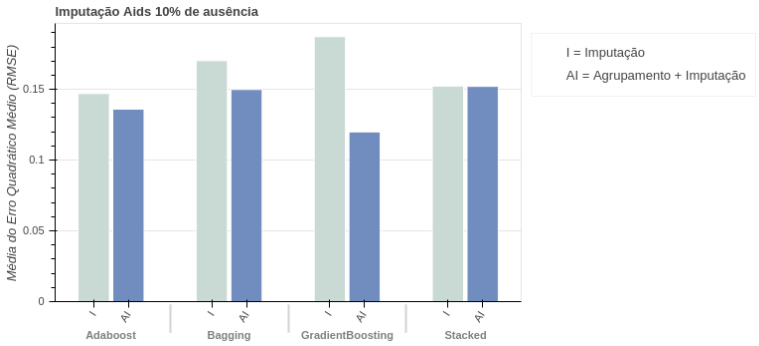

Figura 60 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

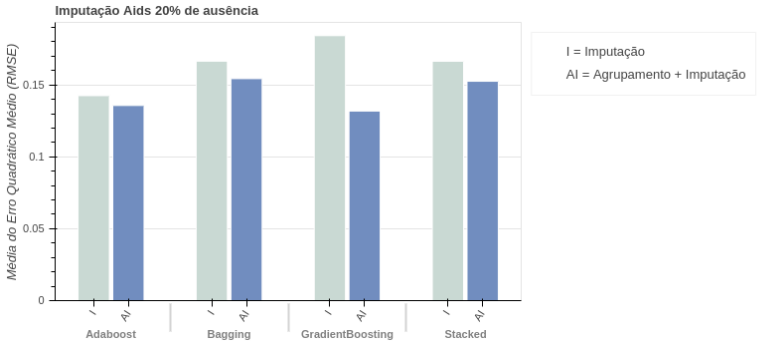

Figura 61 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

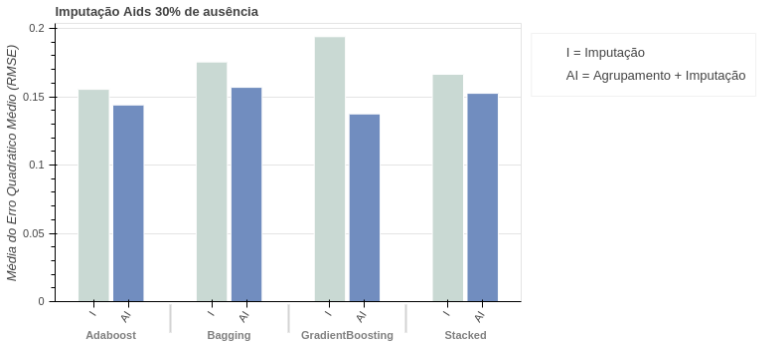

Figura 62 – Comparativo da média de erro (RMSE) entre imputação simples e *hot-deck* com os algoritmos *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*

6- Considerações Finais

Nesta seção são expressas as considerações finais, um compilado dos resultados, as contribuições e os próximos passos para condução de trabalhos futuros.

6.1- Análise Restrospectiva

Neste trabalho, foi apresentado um estudo experimental comparativo sobre o processo de imputação de dados utilizando as técnicas imputação simples e *hot-deck* com comitês regressores *Adaboost*, *Bagging*, *Gradientboost* e *Stacked Generalization*. Foram realizadas medições do erro quadrático médio e tomada de tempo, em segundos, em três conjuntos de dados diferentes (*Breast Cancer, Diabetes e Aids*) e com três percentuais de ausências diferentes (10%, 20%, 30%).

6.1.1 Síntese dos Resultados

Do ponto de vista da precisão do dado imputado, os resultados revelam que *hot-deck* supera a imputação simples na maioria dos cenários testados. Com exceções em: *Breast Cancer* com *Bagging* e 10% de ausência; Diabetes com *Gradientboost* e 10% de ausência; *Aids* com *Stacked Generalization* e 10% de ausência. Com relação as comparações entre *hot-deck*, *Gradientboost* apontou resultados melhores que os demais algoritmos.

Comparando os resultados entre todos os conjunto de dados, *Aids* mostrou os melhores resultados, seguido de *Breast Cancer* e *Pima Indians Diabetes*. Em relação ao tempo de execução, *Bagging* superou os demais algoritmos, na média, na maioria dos cenários testados, exceto nas imputações com 10% e 20% de ausência em *Aids*. Os processamentos mais lentos foram observados em *Gradientboost* seguindo de *Stacked*

Generalization.

Apesar do alto tempo de processamento, *Gradientboost* se mostra uma alternativa razoável, com uma média de tempo de 994 segundos, observados na imputação com 30% de ausência no conjunto com a maior quantidade de dados: *Breast Cancer*.

A utilização de comitês de regressão na imputação *hot-deck* se mostrou uma alternativa promissora tanto no ponto de vista da precisão da reconstituição dos dados ausentes, quanto do ponto de vista do tempo de processamento.

6.1.2 Síntese sobre Comitês

Adaboost e *Gradient Boosting*, por possuírem uma característica de treinamento iterativa e sequencial, não são facilmente escaláveis do ponto de vista computacional. Dado sua natureza de aprendizagem com o erro da iteração anterior somado a utilização do gradiente descendente para achar a menor função de custo, *Gradient Boosting* surge como uma boa opção de comitê apesar do alto tempo de processamento.

Bagging e *Stacked Generalization* se assemelham pelo fato dos K componentes não serem dependentes entre si, porém a grande diferença entre as duas abordagens está no fato de *Bagging* combinar os K resultados das predições por meio de votação de maioria ou média ponderada. Essa situação apoia *Bagging* em obter os menores tempos de processamento. Como visto nos resultados (Seção 5.4). *Stacked Generalization* realiza a predição final por meio de um algoritmo de aprendizado (meta-aprendizado) de máquina, além dos K componentes, implicando no aumento do tempo de processamento.

6.2- Trabalhos Futuros

Este estudo comparativo, poderá guiar a escolha de qual algoritmo comitê de regressão escolher em novas pesquisas, tanto de imputação de dados, quanto de aprendizado de máquina em geral. Os próximos passos da pesquisa consistem na utilização de comitês com outros regressores base, explorar o *Stacked Generalization*, com outro

comitê como base e realizar os testes com imputação composta.

A- Apêndice A - Strings de Busca no Scopus

A.1- Busca realizada para imputação de dados em geral

TITLE (”imputation”OR ”missing data”) AND NOT (”microarray”OR ”RNA”OR ”gene”OR ”genotype”OR ”biology”OR ”proteomics”OR ”game”OR ”image”OR ”signal”OR ”audio”OR ”web”OR ”text”) AND (LIMIT-TO (SRCTYPE , ”j”) OR LIMIT-TO (SRCTYPE , ”p”)) AND (LIMIT-TO (SUBJAREA , ”COMP”)) AND (LIMIT-TO (DOCTYPE , ”cp”) OR LIMIT-TO (DOCTYPE , ”ar”)) AND (LIMIT-TO (LANGUAGE , ”English”))

A.2- Busca realizada especificamente para imputação de dados utilizando métodos ensemble

TITLE-ABS-KEY ((”imputation”OR ”multiple imputation”OR ”missing data”) AND (”ensemble”OR ”stacking”OR ”stacked”OR ”bagging”OR ”random forest”OR ”boost”OR ”boosting”OR ”adaboosting”OR ”gradient boosting”OR ”xgboosting”OR ”catboost”OR ”lightgbm”)) AND NOT (”microarray”OR ”RNA”OR ”gene”OR ”gens”OR ”genotype”OR ”biology”OR ”proteomics”OR ”game”OR ”image”OR ”signal”OR ”audio”OR ”web”OR ”text”) AND (LIMIT-TO (SRCTYPE , ”j”) OR LIMIT-TO (SRCTYPE , ”p”)) AND (LIMIT-TO (SUBJAREA , ”COMP”)) AND (LIMIT-TO (DOCTYPE , ”cp”) OR LIMIT-TO (DOCTYPE , ”ar”)) AND (LIMIT-TO (LANGUAGE , ”English”))

Referências Bibliográficas

Abnane, I., Hosni, M., Idri, A., and Abran, A. (2019). Analogy software effort estimation using ensemble knn imputation. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 228–235. IEEE.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17.

Azur, M. J., Stuart, E. A., Frangakis, C., and Leaf, P. J. (2011). Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49.

Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L. (1996b). Stacked regressions. *Machine learning*, 24(1):49–64.

Christopher, S. Z., Siswantining, T., Sarwinda, D., and Bustaman, A. (2019). Missing value analysis of numerical data using fractional hot deck imputation. In *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)*, pages 1–6. IEEE.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Ferlin, C. (2008). Imputação multivariada: Uma abordagem em cascata. *Rio de Janeiro, RJ*.

Ford, B. L. (1983). An overview of hot-deck procedures. *Incomplete data in sample surveys*, 2(Part IV):185–207.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Gad, I., Manjunatha, B., et al. (2017). Performance evaluation of predictive models for missing data imputation in weather data. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1327–1334. IEEE.

Goldschmidt, R., Bezerra, E., and Passos, E. (2015). *Data Mining: Conceitos, técnicas, algoritmos, orientações e aplicações*.

Graham, J. W. (2012). *Missing data: Analysis and design*. Springer Science & Business Media.

Graham, J. W., Cumsille, P. E., and Shevock, A. E. (2012). Methods for handling missing data. *Handbook of Psychology, Second Edition*, 2.

Graham, J. W. and Donaldson, S. I. (1993). Evaluating interventions with differential attrition: the importance of nonresponse mechanisms and use of follow-up data. *Journal of Applied Psychology*, 78(1):119.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hassan, M. M., Atiya, A. F., El-Gayar, N., and El-Fouly, R. (2007). Regression in the presence missing data using ensemble methods. In *2007 International Joint Conference on Neural Networks*, pages 1261–1265. IEEE.

Hoskin, T. L., Boughey, J. C., Day, C. N., and Habermann, E. B. (2019). Lessons learned regarding missing clinical stage in the national cancer database. *Annals of surgical oncology*, 26(3):739–745.

Karanikola, A. and Kotsiantis, S. (2019). A hybrid method for missing value imputation. In *Proceedings of the 23rd Pan-Hellenic Conference on Informatics*, pages 74–79.

Krause, R. W., Huisman, M., Steglich, C., and Sniiders, T. A. (2018). Missing network data a comparison of different imputation methods. In *2018 IEEE/ACM International*

Conference on Advances in Social Networks Analysis and Mining (ASONAM), pages 159–163. IEEE.

Kumar, S., Pandey, M. K., Nath, A., and Subbiah, K. (2016). Performance analysis of ensemble supervised machine learning algorithms for missing value imputation. In *2016 2nd International Conference on Computational Intelligence and Networks (CINE)*, pages 160–165. IEEE.

LeDell, E. (2015). *Scalable Ensemble Learning and Computationally Efficient Variance Estimation*. PhD thesis, UC Berkeley.

Lichman, M. (2018). Uci machine learning repository. irvine, university of california, irvine, school of information and computer sciences.(2013).

Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data*, volume 793. Wiley.

Lu, X., Si, J., Pan, L., and Zhao, Y. (2011). Imputation of missing data using ensemble algorithms. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 2, pages 1312–1315. IEEE.

Madhu, G., Bharadwaj, B. L., Nagachandrika, G., and Vardhan, K. S. (2019). A novel algorithm for missing data imputation on machine learning. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 173–177. IEEE.

Magnani, M. (2004). Techniques for dealing with missing data in knowledge discovery tasks. *Obtido http://magnanim. web. cs. unibo. it/index. html*, 15(01):2007.

Maheswari, K., Priya, P. P. A., Ramkumar, S., and Arun, M. (2019). Missing data handling by mean imputation method and statistical. In *EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing: BDCC 2018*, page 137. Springer Nature.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

Mitchell, T. (1997). Machine learning, mcgraw-hill higher education. *New York*.

Nan, Y. and Gao, Y. (2018). A machine learning method to monitor china's aids epidemics with data from baidu trends. *PloS one*, 13(7):e0199697.

Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21.

Poulos, J. and Valle, R. (2018). Missing data imputation for supervised learning. *Applied Artificial Intelligence*, 32(2):186–196.

Quinlan, J. R. et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific.

Quinlan, S., Afli, H., and O'Reilly, R. (2019). A comparative analysis of classification techniques for cervical cancer utilising at risk factors and screening test results. In *AICS*, pages 400–411.

Raghunathan, T. E. (2004). What do we do with missing data? some options for analysis of incomplete data. *Annu. Rev. Public Health*, 25:99–117.

Ratolojanahary, R., Ngouna, R. H., Medjaher, K., Junca-Bourié, J., Dauriac, F., and Sebilo, M. (2019). Model selection to improve multiple imputation for handling high rate missingness in a water quality dataset. *Expert Systems with Applications*, 131:299–307.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

Rubin, D. B. (1988). An overview of multiple imputation. In *Proceedings of the survey research methods section of the American statistical association*, pages 79–84. Citeseer.

Saha, B., Gupta, S., Phung, D., and Venkatesh, S. (2017). Effective sparse imputation of patient conditions in electronic medical records for emergency risk predictions. volume 53, pages 179–206. Springer.

Saitoh, F. (2016). An ensemble model of self-organizing maps for imputation of missing values. In *2016 IEEE 9th International Workshop on Computational Intelligence and Applications (IWCIA)*, pages 9–14. IEEE.

Schafer, J. L. and Graham, J. W. (2002). Missing data: our view of the state of the art. *Psychological methods*, 7(2):147.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227.

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

Silva-Ramírez, E.-L., Pino-Mejías, R., and López-Coello, M. (2015). Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, 29:65–74.

Smyth, P. and Wolpert, D. (1998). Stacked density estimation. In *Advances in neural information processing systems*, pages 668–674.

Soares, J. (2007). *Pré-Processamento em mineração de dados: Um Estudo Comparativo em Complementação. Tese de Doutorado. Engenharia de Sistemas e Computação. UFRJ 2007.* PhD thesis.

Souza, R. T. (2019). Appraisal-spark: Uma abordagem para imputação em larga escala. Master's thesis, CEFET/RJ - PPCIC.

Souza, R. T., Castaneda, R., Ferlin, C., Goldschmidt, R., Alfredo, L. V. C., and Soares, J. d. A. (2018). Apoiando o processo de imputação com técnicas de aprendizado de máquina. In *33rd Brazilian Symposium on Databases (SBBD)*, pages 259–264.

Suresh, M., Taib, R., Zhao, Y., and Jin, W. (2019). Sharpening the blade: Missing data imputation using supervised machine learning. In *Australasian Joint Conference on Artificial Intelligence*, pages 215–227. Springer.

Templ, M., Kowarik, A., and Filzmoser, P. (2011). Iterative stepwise regression imputation using standard and robust methods. *Computational Statistics & Data Analysis*, 55(10):2793–2806.

Ting, K. M. and Witten, I. H. (1999). Issues in stacked generalization. *Journal of artificial intelligence research*, 10:271–289.

Twala, B. and Cartwright, M. (2010). Ensemble missing data techniques for software effort prediction. *Intelligent Data Analysis*, 14(3):299–331.

Valdiviezo, H. C. and Van Aelst, S. (2015). Tree-based prediction on incomplete data using imputation or surrogate decisions. *Information Sciences*, 311:163–181.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

Xi, Y., Zhuang, X., Wang, X., Nie, R., and Zhao, G. (2018). A research and application based on gradient boosting decision tree. In *International Conference on Web Information Systems and Applications*, pages 15–26. Springer.

Zhang, C. and Ma, Y. (2012). *Ensemble machine learning: methods and applications*. Springer.

Zhang, Z., Mayer, G., Dauvilliers, Y., Plazzi, G., Pizza, F., Fronczek, R., Santamaria, J., Partinen, M., Overeem, S., Peraita-Adrados, R., et al. (2018). Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from european narcolepsy network database with machine learning. *Scientific reports*, 8(1):10628.