CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

APPRAISAL-SPARK: UMA ABORDAGEM PARA IMPUTAÇÃO EM LARGA ESCALA

Rodrigo Tavares de Souza

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

Orientador:
Jorge de Abreu Soares

Rio de Janeiro,

Fevereiro de 2019

Appraisal-Spark: uma abordagem para imputação em larga escala

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Rodrigo Tavares de Souza

Aprovada por:

Presidente, Prof. Jorge de Abreu Soares, D.Sc. (orientador)

Prof. Eduardo Soares Ogasawara, D.Sc.

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

Rio de Janeiro,

Fevereiro de 2019

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

S168 de Souza, Rodrigo Tavares

Appraisal-Spark: uma abordagem para imputação em larga escala / Rodrigo Tavares de Souza.—2019.
xiii, 58f. : il. (algumas color.) , grafs. , tabs. ; enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca , 2019.
Bibliografia : f. 51-58
Orientador : Jorge de Abreu Soares

1. Imputação 2. Processamento Distribuído 3. Computação em Larga Escala 4. Dados ausentes 5. Comitês de Classificação 6. Bagging, I. Soares, Jorge de Abreu (Orient.). II. Título.

CDD 004

DEDICATÓRIA

Dedico este trabalho a memória dos meus avôs:

Francisco e Helcio.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ), que financiou parcialmente esta pesquisa. Agradeço também a CAPES, ao CNPq e a FAPERJ, pelo incentivo a pesquisa científica de forma geral.

Agradeço ao Prof. Jorge Soares, por ter acreditado em mim e me orientado no âmbito desta pesquisa. Obrigado por compartilhar seus conhecimentos comigo ao longo dessa jornada, sempre de maneira tão precisa e generosa.

Agradeço a todos os professores do PPCIC, vocês contribuíram para que eu me tornasse um profissional melhor. Em especial gostaria de agradecer ao Prof. Eduardo Ogasawara, pela dedicação na condução do programa e incentivo constante para que nos tornemos melhores pesquisadores. Agradeço também ao Prof. Ronaldo Goldschmidt pela disponibilidade em participar da banca examinadora

Agradeço a minha família: minha mãe Vera, meu pai Luiz e meus irmãos Rafael e Rogerio; vocês são a base de tudo.

RESUMO

Appraisal-Spark: uma abordagem para imputação em larga escala

Rodrigo Tavares de Souza

Orientador:
Jorge de Abreu Soares

Resumo da Dissertação submetida 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.

Cresce continuamente o volume de dados armazenados e a demanda por integração entre os mesmos. Esse cenário aumenta a ocorrência de um problema bastante conhecido dos cientistas de dados: as diversas possibilidades de inconsistências. Um de seus tipos comuns, a ausência de dados, pode prejudicar a análise e resultado de qualquer técnica produtora de informação. A imputação é a área que estuda métodos que buscam aproximar o valor imputado do real. A técnica de imputação composta aplica tarefas de aprendizado de máquina neste processo. Ela utiliza o conceito de plano de imputação, uma sequência lógica de estratégias e algoritmos utilizados na produção do valor imputado final. Neste trabalho, a utilização desta técnica é expandida, complementando sua utilização com o classificador *ensemble bagging*. Neste método, os dados são divididos em grupos aleatórios e atrelados a classificadores chamados *base learners*. Para os *subsets* gerados no *bagging* são retornadas as pontuações (percentual de assertividade) de cada plano de imputação. O plano com maior assertividade dentre todos os *subsets* é indicado como a sugestão de imputação para o conjunto completo. O trabalho é implementado em um *framework* desenvolvido para a ferramenta Apache Spark, denominado Appraisal-Spark, que tem como objetivo gerar valores com maior acurácia e desempenho preditivos para ambientes de larga escala. Através dele é possível compor diversos planos de imputação de alto desempenho, avaliando estratégias e comparando resultados. O Appraisal-Spark é utilizado para imputação em duas bases de dados, com diferentes características e níveis de correlação entre seus atributos. Os experimentos no Appraisal-Spark foram realizados de forma serial e paralela, explorando a utilização da ferramenta Apache Spark.

Palavras-chave:
Imputação; Processamento Distribuído; Computação em Larga Escala; Dados ausentes; Comitês de Classificação; Bagging.

Rio de Janeiro,

Fevereiro de 2019

ABSTRACT

Appraisal-Spark: an approach for large-scale imputation

Rodrigo Tavares de Souza

Advisor:
Jorge de Abreu Soares

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

The volume of stored data and the demand for integration between the two continually grow. This scenario increases the occurrence of a problem well known to the data scientists: the various possibilities for inconsistencies. One of its common types, the absence of data, may impair the analysis and outcome of any information-producing technique. Imputation is the area that studies methods that seek to approximate the imputed value to the real one. The composite imputation technique applies machine learning tasks in this process. It uses the imputation plan concept, a logical sequence of strategies and algorithms used to produce the final imputed value. In this work, the utilization of this technique is expanded, complementing its use with bagging, an ensemble classifier. In this method, the data are divided into random groups and attached to classifiers called base learners. For the generated subsets, the scores (percentage of assertiveness) of each imputation plan are returned. The plan with greater assertiveness of all the subsets is indicated as the suggestion of imputation for the complete set. The work is implemented in a framework developed for the Apache Spark tool, called Appraisal-Spark, which aims to generate better values of predictability and performance for large-scale environments. Through it, it is possible to compose several high performance imputation plans, evaluating strategies and comparing results. Appraisal-Spark was used for imputation in two databases, with different characteristics and levels of correlation between its attributes. Experiments in Appraisal-Spark are serially and parallelly performed, exploring the use of the Apache Spark tool.

Key-words:
Imputation; Distributed Processing; Large scale computing; Missing data; Ensemble Classifiers; Bagging.

Rio de Janeiro,

Fevereiro de 2019

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DAG	Directed Acyclic Graph	21
ERP	Enterprise Resource Planning	14
GRA	Grey Relational Analysis	24
IOT	Internet Of Things	1, 14
KDD	Knowledge Discovery In Databases	2, 4, 5, 6
MAR	Missing At Random	8, 9, 24
MCAR	Missing Completely At Random	8, 14
NMAR	Not Missing At Random	8, 9
PCA	Principal Components Analysis	10, 13, 26
SVM	Support-vector Machine	23
UML	Unified Modelling Language	21
WNN	Weighted Nearest Neighbours	26

Capítulo I Introdução

É cada vez maior a quantidade de dados gerados, processados e armazenados por sistemas de informação. Além disso, dispositivos de armazenamento tornaram-se mais acessíveis, apresentam melhor desempenho e maior capacidade. Tais fatos contribuíram para que o volume de dados armazenados e, consequentemente, processados, aumentasse massivamente nos últimos anos [Khan et al., 2014]. Soma-se a isso a significativa melhora da infraestrutura (redes e internet) e o aumento da demanda de utilização de sistemas distribuídos. A figura I.1 mostra o aumento da produção de HDs entre os anos de 1976 e 2013.

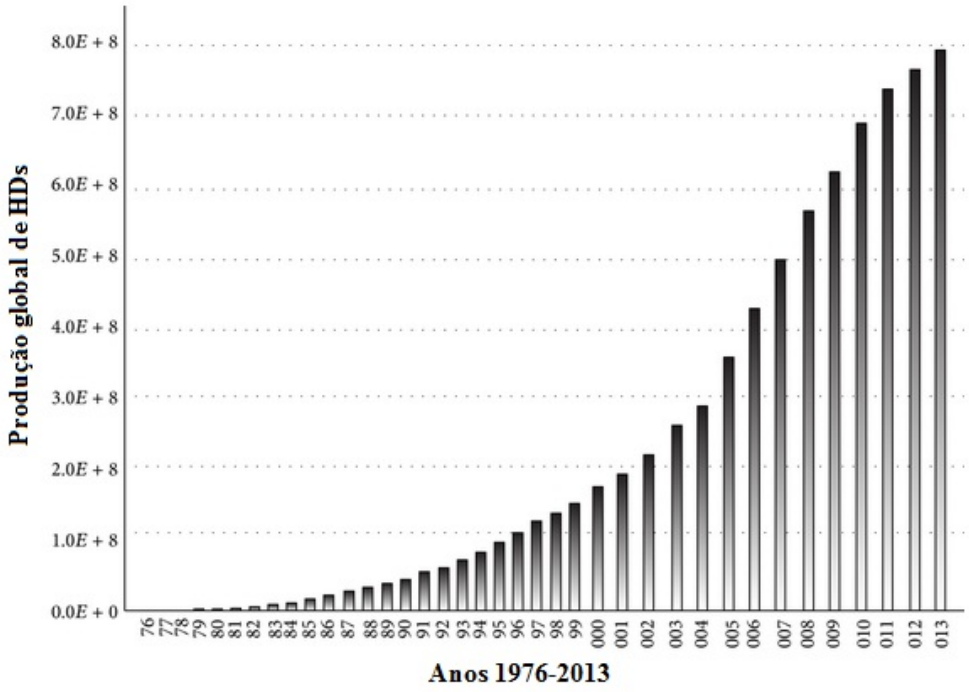

Figura I.1: Produção global de HDs. Fonte: adaptado de Khan et al. [2014].

Outro fato que contribui para o aumento de dados gerados em escala sem precedente é a criação de novos sensores e transmissores, capazes de prover diversos tipos de informação. Permitem que até mesmo eletrodomésticos sejam capazes de gerar dados. A *Internet of Things (IoT)* tende a transformar os campos físico e digital em um só [Atzori et al., 2010]. A integração entre dispositivos, o acesso a internet por parte destes e, consequentemente, a grandes data centers e nuvens, contribuem para o aumento da quantidade de dados armazenados.

O termo *Big Data* representa essa nova escala de volume de dados. Decisões anteriormente baseadas em suposições, ou sobre modelos de realidade meticulosamente trabalhada, são agora feitas através de modelos matemáticos [Jagadish et al., 2014]. A descoberta de informações nesse nível demanda uma estrutura de processamento de dados apoiada em arquiteturas de computação paralela e distribuída [Chen et al., 2014]. Devem compor uma plataforma de alto desempenho, processando a maior quantidade de dados possível em tempo adequado.

Grandes volumes de dados podem esconder informações valiosas capazes de definir o rumo estratégico das organizações. É possível criar um processo de descoberta de conhecimento em base de dados [M. Fayyad et al., 1996]. O termo originário *Knowledge Discovery in Databases (KDD)*, bem como as áreas mais recentes de Inteligência Artificial e Apredizado de Máquina, remetem a este processo de descoberta de conhecimento e apoio à decisão [Zhang and Ma, 2012]. O comportamento de clientes, tendências, características de negócio, entre outros, são muitas as informações que podem ser descobertas. Entretanto, para que dados sejam transformados em conhecimento existe um longo caminho a ser percorrido. A área de KDD procura descobrir e analisar as relações intrínsecas existentes em um conjunto de dados, gerando conhecimento [Fayyad et al., 1996].

O aumento no volume de dados por sua vez maximiza um problema conhecido dos administradores de dados: as inconsistências de dados. Inconsistências podem ocorrer por motivos tais como falhas de equipamentos, na transmissão da mensagem, erros de preenchimento não tratados, falhas em rotinas de carga, entre outros [Han et al., 2011]. Essas inconsistências podem também ser fruto primário dos diversos momentos onde bases de dados são integradas ou cujo processo de consolidação da base não recebeu o devido cuidado [Soares, 2007].

Ausências de dados, dependendo de sua natureza e incidência, podem prejudicar sobremaneira a análise de dados por qualquer técnica produtora de informação, tais como aprendizado de máquina, os armazéns de dados *(Data Warehouse)*, *Businees Inteligence* ou similares, e comprometer seus resultados [Enders, 2010]. Para isso, existe um campo de estudo denominado "Imputação" [Little and Rubin, 1986], que busca resolver esses problemas, complementando dados ausentes fundamentalmente com base em métodos estatísticos e de inteligência computacional [Little and Rubin, 2002]. Esses métodos podem ter abordagens simples, com resultados razoáveis, ou um pouco mais complexas, de acordo com o mecanismo de ausência apresentado [Gelman and Hill, 2007].

O crescimento do volume de dados, seja na na área industrial ou de pesquisa, representa enormes oportunidades, bem como grandes desafios computacionais [Deelman et al., 2009]. Como os tamanhos de dados ultrapassaram as capacidades das máquinas individuais, os cientistas de dados precisam de novos sistemas para ampliar o ambiente computacional, utilizando múltiplos nós de processamento [Zaharia et al., 2016a]. A evolução dos supercomputadores, bem como de soluções de processamento paralelo e distribuído, impulsionam a capacidade de processamento computacional

e o avanço científico [Deelman et al., 2009].

No que tange à pesquisa científica, é necessário que todo o trabalho, as diversas etapas e repetições de tarefas dentro de um experimento, sejam gerenciadas de forma organizada [Deelman et al., 2009]. Gravar corretamente os resultados, acompanhar a evolução do processo como um todo e garantir a reprodutibilidade dos experimentos são características de sistemas de *workflow* científico [Davidson and Freire, 2008]. Essa categoria de sistemas apoia a experimentação, possibilitando que cientistas analisem diversos tipos de dados, técnicas e algoritmos, criando um ambiente agradável e, preferencialmente, de fácil utilização [Deelman et al., 2009].

Nas últimas décadas, pesquisadores das áreas de inteligência computacional e aprendizado de máquina têm estudado técnicas que utilizam um processo de decisão articulado e combinado [Zhou, 2012]. Essas técnicas são geralmente referidas como classificadores *ensemble*, conhecidas por reduzir a variação dos classificadores originais e melhorar a robustez e precisão de sistemas de decisão [Zhang and Ma, 2012].

Este trabalho tem por objetivo avaliar a utilização do classificador *ensemble Bagging* em conjunto a técnica de Imputação Composta [Soares, 2007], com o objetivo de produzir valores imputados com menor erro absoluto. Para materialização do trabalho foi desenvolvido o framework Appraisal-Spark, utilizando a plataforma Apache Spark, para a composição e execução de planos de imputação em larga escala. Foram geradas ausências em duas bases de dados, de diferentes características, de forma que diversos planos de imputação foram executados em modo serial e paralelo, em um *cluster* Spark. Os resultados dos experimentos são analisados quanto ao desempenho computacional e qualidade dos valores imputados produzidos.

A estrutura do restante deste documento está organizada da seguinte forma: o capítulo II introduz os conceitos de imputação, classificadores *ensemble* e computação paralela e distribuída, fundamentais para este trabalho; o capítulo III aborda trabalhos relacionados aos temas citados; o capítulo IV detalha o método proposto para imputação em larga escala com classificadores *ensemble*; o capítulo V descreve a metodologia utilizada na condução dos experimentos, as bases de dados, o *framework* desenvolvido para a implementação da técnica proposta e os resultados obtidos; e finalmente o capítulo VI apresenta as considerações finais e direções para trabalhos futuros.

Capítulo II Fundamentação Teórica

Neste capítulo serão introduzidos os conceitos de imputação e o posicionamento da técnica no processo de descoberta de conhecimento em bases de dados. Além disso, serão abordados os algoritmos utilizados na técnica proposta, bem como conceitos de computação distribuída e a utilização de classificadores *ensemble* no contexto de aprendizado de máquina. Por fim, será abordada a importância da utilização de sistemas de *workflow* no processo de pesquisa científica.

II.1 Descoberta de Conhecimento em Bases de Dados

A necessidade de ampliar as capacidades de análise humana para inferir conhecimento sobre grandes volumes de dados possui motivações econômicas e científicas [Jagadish et al., 2014]. Empresas utilizam dados para ganhar vantagem competitiva, aumentar a eficiência e fornecer serviços mais valiosos para os clientes[Chen et al., 2014]. A ciência, por sua vez, busca interpretar dados que geram informações sobre o meio ambiente, o espaço, fenômenos naturais, entre outros. Essas informações ajudam na formulação de teorias e modelos que apoiam o desenvolvimento científico [Han et al., 2011].

A descoberta de conhecimento em base de dados tem origem no termo *KDD*, que consiste num processo de várias etapas, não trivial, interativo e iterativo, para identificação de padrões compreensíveis, válidos, novos e potencialmente úteis, a partir de grandes conjuntos de dados [M. Fayyad et al., 1996]. As aplicações de *KDD* são diversas e buscam desenvolver métodos e técnicas que criem sentido para os dados.

O processo possui dois objetivos: verificação e descoberta [Han et al., 2011]. No de verificação, o sistema limita-se a avaliar hipóteses pré-estabelecidas. No de descoberta, o sistema é capaz de identificar automaticamente padrões ocultos nos dados. A descoberta pode ser subdividida em preditiva – onde o sistema descobre padrões que indicam um comportamento ou acontecimento futuro – e descritiva, que encontra padrões a serem apresentados de forma inteligível[Fayyad et al., 1996].

O processo de descoberta de conhecimento em bases de dados é composto pelas seguintes etapas: seleção, pré-processamento, transformação, mineração de dados, avaliação de padrões e apresentação do conhecimento [Han et al., 2011]. A figura II.1 mostra o caminho percorrido desde os dados

originais, passando pelas tarefas de *KDD*, até a geração do conhecimento.

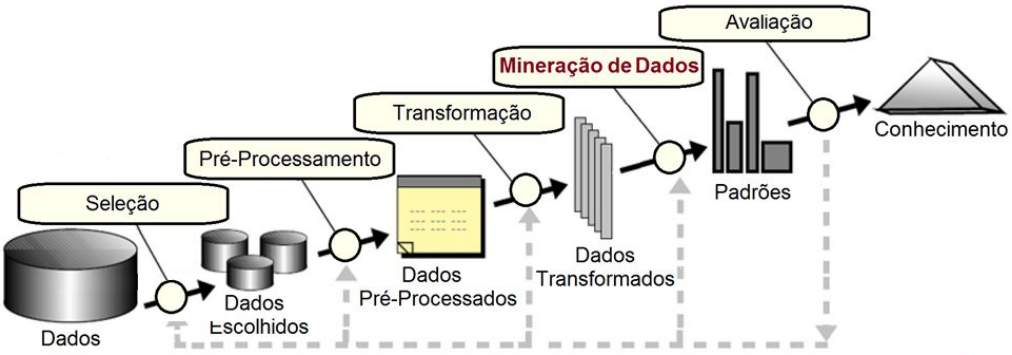

Figura II.1: O processo de KDD. Fonte: adaptado de Han et al. [2011].

A mineração de dados é o processo de descoberta de padrões e geração de conhecimento sobre grandes volumes de dados [Han et al., 2011]. As fontes de dados podem incluir bancos de dados, armazéns de dados, a Web, repositórios de informação, entre outros. É durante esta etapa que as relações entre os elementos da base de dados são descobertas[Goldschmidt et al., 2015].

Hand et al. [2001] denota um conceito interessante, sob a ótica da representatividade dos resultados gerados na etapa de mineração de dados:

> "A análise de (quase sempre grandes) conjuntos dados observados para descobrir relações escondidas e para consolidar os dados de uma forma tal que sejam inteligíveis e úteis aos seus donos"[Hand et al., 2001].

Tornar os dados inteligíveis significa apresentá-los de forma que informações relevantes possam ser obtidas sobre os mesmos. Informações essas que gerem conhecimento e agreguem valor científico ou estratégico [Chen et al., 2014]. Para tal, uma série de técnicas podem ser utilizadas, dependo do tipo de análise que se deseja realizar. As tarefas do processo de mineração de dados mais utilizadas são [Han et al., 2011]:

• Regras de associação: baseia-se nos conceitos de suporte e confiança para inferir associações entre os atributos de um conjuntos de dados. Identifica objetos como antecedentes e consequentes, quando é observado o acontecimento de um fenômeno em consequência de outro.

• Classificação: extrai modelos que descrevem classes de dados. Tais modelos, chamados classificadores, predizem rótulos de classe para novos valores inseridos na amostra.

• Agrupamento: tem como objetivo reunir em um mesmo grupo objetos que mantenham algum grau de afinidade. Busca maximizar a similaridade de objetos do mesmo grupo e minimizar a similaridade entre elementos de grupos distintos.

• Previsão de Séries Temporais: uma série temporal é um conjunto de observações de um fenômeno ordenadas no tempo. Sua análise consiste no processo de identificação das características,

padrões e propriedades, utilizados para descrever o fenômeno gerador. Tem como principal objetivo a geração de modelos voltados à previsão de valores futuros.

• Detecção de Desvios (*Outliers*): um outlier é um objeto cujo valor desvia significativamente dos demais na amostra, como se fosse gerado por um mecanismo diferente do usual.

É comum observar o processo de *KDD* sendo referido simplesmente como Mineração de Dados, já que esta é, por diversas vezes, considerada a etapa mais importante do trabalho.

Dados podem possuir diferentes formas e tipos. A sessão II.1.1 aborda os tipos de atributos existentes em conjuntos de dados.

II.1.1 Natureza dos dados

Os atributos de uma tabela podem ser divididos em duas classes: numéricos e categóricos [Han et al., 2011]. Atributos numéricos são quantitativos, apresentam ou não limites inferior e superior. Subdividem-se em atributos contínuos e discretos. Atributos contínuos são aqueles que podem ser medidos em uma escala contínua, como peso e temperatura, por exemplo. Atributos discretos são os que representam um número finito ou uma quantidade enumerável, como o número de membros de uma equipe, por exemplo.

Atributos categóricos ou nominais indicam o conceito ao qual o objeto se enquadra, revelando a categoria em que o mesmo se encontra[Han et al., 2011]. Neste tipo de atributo não existe a noção de ordem entre os valores. Podem ser representados por tipos alfanuméricos.

Para que tarefas de mineração de dados gerem resultados completos, é necessário que os dados estejam íntegros [Goldschmidt et al., 2015]. Dados com erros, incompletos, redundantes ou desnivelados, podem comprometer a qualidade da análise [Soares, 2007]. Esses problemas são tratados na etapa de pré-processamento.

II.1.2 Pré-Processamento

Dados possuem qualidade se satisfazem os requisitos do uso pretendido. Há muitos fatores que compõem a qualidade dos dados, entre eles: precisão, integridade, consistência, pontualidade, credibilidade e interpretabilidade [Han et al., 2011].

A etapa de pré-processamento possui como principais atividades [Goldschmidt et al., 2015]:

- Agrupamento de dados
- Coleta e Integração
- Codificação
- Construção de Atributos
- Correção de Prevalência
- Discretização

- Enriquecimento
- Limpeza dos Dados
- Normalização de Dados
- Partição dos Dados
- Redução de Dados
- Seleção de Dados

A figura II.2 ilustra algumas atividades da etapa de pré-processamento de dados.

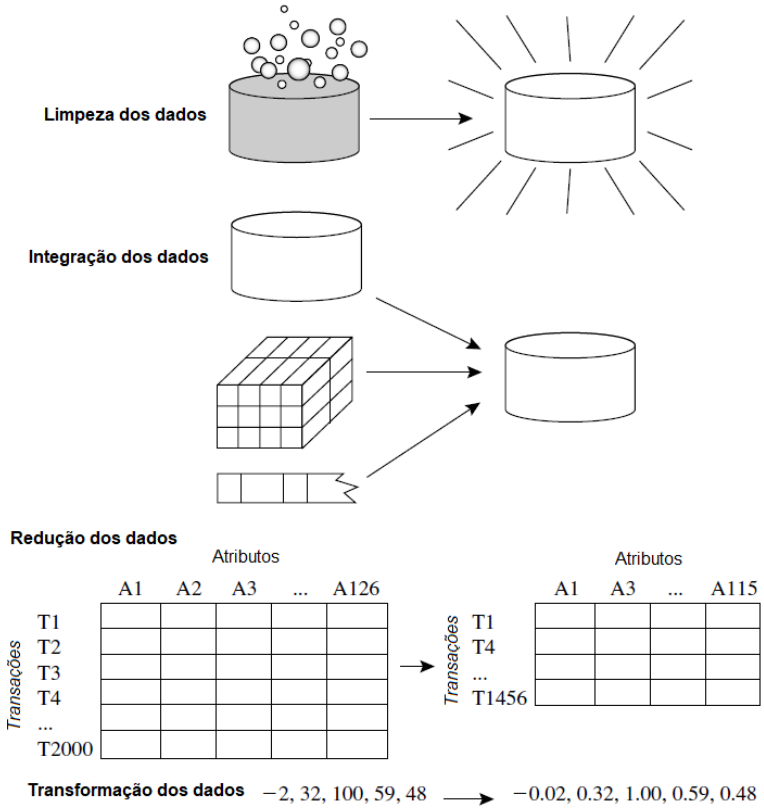

Figura II.2: A etapa de pré-processamento de dados. Fonte: adaptado de Han et al. [2011].

A tarefa de limpeza de dados tem como objetivo [Han et al., 2011]:

- Complementar dados ausentes.
- Suavizar ruídos através da identificação de *outliers*.
- Corrigir possíveis inconsistências nos dados.

A tarefa de complementação de dados ausentes será abordada com mais detalhes na proxima seção, pois é a que compreende a subtarefa de imputação, no contexto de descoberta do conhecimento.

II.1.3 Complementação de Dados Ausentes

Tanto em tarefas de treinamento como em testes de modelos de KDD, o significado da análise depende de duas premissas: a precisão do banco de dados e a qualidade dos dados da amostra. De acordo com Brown and Kros [2003], a ausência de dados deve ser corretamente tratada, pois ignorar esse problema pode introduzir uma polarização nos modelos avaliados, levando a conclusões incorretas no processo de mineração de dados.

Existem métodos mais avançados para o tratamento de ausências de dados, como a imputação, explicada em mais detalhes na seção III.1. Entretanto, existem métodos convencionais, menos eficientes, que tratam a ausência de dados simplesmente removendo-a da amostra. Magnani [2004] classificou-os como:

a) Remoção completa de casos *(Listwise Deletion)* - É a maneira mais fácil de obter um conjunto de dados completo. Somente registros completos são mantidos. Este método é de fácil implementação, mas possui a desvantagem de que, em situações reais, muitos dados podem ser perdidos.

b) Remoção em pares *(Pairwise Deletion)* - É uma variante da remoção completa de casos. Utiliza um registro incompleto somente quando a variável analisada não é ausente. A vantagem desta técnica é poder utilizar todos os dados quando os valores analisados não apresentam dependência de valores ausentes. Como desvantagem, sua implementação é mais complexa, pois demanda a análise da relação entre os atributos do conjunto de dados.

c) Remoção de colunas com valores ausentes - Qualquer atributo que apresente valores ausentes é inteiramente removido. Sua aplicação na maioria das vezes não é recomendada, já que a perda de informação com a remoção de atributos é significativa.

II.1.4 Mecanismos de Ausência de Dados

Ausências de dados normalmente seguem um mecanismo de distribuição, mas também podem ocorrer de forma intermitente ou simplesmente ao acaso [Enders, 2010]. Os motivos de ocorrência são vários [Han et al., 2011]: integrações entre bases de dados, falhas de equipamentos, na transmissão de mensagens, erros de preenchimento não tratados, falhas em rotinas de carga, entre outros.

Trabalhos envolvendo a complementação de dados ausentes levam inevitavelmente em conta o mecanismo que causou a ausência dos dados. Little and Rubin [1986] classificaram esses mecanismos em três tipos: completamente aleatório - Missing Completely at Random (MCAR), aleatório - Missing at Random (MAR) e não aleatório - Not Missing at Random (NMAR).

- MCAR: é o nível mais alto de aleatoriedade. Ocorre quando a probabilidade de uma instância possuir valores ausentes, para um determinado atributo, não depende dos valores conhecidos e nem dos ausentes. Nesse nível de aleatoriedade, qualquer tratamento pode ser aplicado sem

risco de enviesar os dados.

- MAR: ocorre quando a probabilidade de uma instância possuir valores ausentes depende de valores conhecidos, ou seja, de atributos diferentes do que possui a ausência. Por exemplo, um atributo que registra os custos de reparos realizados em uma oficina, não possui alguns valores quando o carro é da marca Ford (outro atributo).

- NMAR: nesse caso a ausência é relacionada ao próprio atributo que possui valor ausente. Por exemplo, um atributo que registra a velocidade máxima dos carros em uma via não possui registros para velocidades maiores que 180 km/h.

As características da ausência de dados devem ser analisadas para a escolha do tratamento adequado [Enders, 2010]. Dados com alto nível de ausência, ou cujos valores ausentes não sigam uma aleatoriedade, devem ser tratados por métodos robustos que sigam um modelo preditivo [Batista and Monard, 2001]. Em um modelo preditivo, o atributo com valores ausentes é utilizado como classe (atributo de saída) e os demais atributos como entrada, para a composição de valores mais precisos para a substituição da ausência [Enders, 2010].

II.2 Imputação

Existem diversos métodos para tratar a ausência de dados, ou seja, substituir valores ausentes por valores reais [Rubin, 1988]. A tarefa de imputação tem como objetivo recuperar valores ausentes de maneira mais precisa, através de técnicas que variam desde a média simples, regressão linear, modelos preditivos específicos, até a utilização de algoritmos de aprendizado de máquina [Enders, 2010]. O método de imputação utilizado depende primordialmente da natureza do atributo imputado [Soares, 2007].

Métodos de imputação são classificados em dois grupos [Soares, 2007]: métodos simples e métodos híbridos. Métodos simples consideram apenas um valor proposto para cada valor ausente. Métodos híbridos combinam dois ou mais métodos de imputação simples, com o objetivo de melhorar a qualidade do processo de imputação como um todo [Soares, 2007].

Um exemplo de método híbrido é a imputação múltipla, onde o conjunto de valores ausentes é substituído por mais de um conjunto plausível de valores [Raghunathan, 2004]. A criação de mais de um, no caso n plausíveis conjuntos de substituição para valores ausentes, consequentemente gerando n conjuntos completos. A variação em todos os conjuntos completos, medida através do desvio padrão, reflete a incerteza da imputação. C Yuan [2005] propõe uma abordagem parecida; entretanto, após a identificação do método com melhor desempenho, novas inferências estatísticas são realizadas de forma a ajustar os hiperparâmetros dos modelos de imputação. A utilização de múltiplos classificadores tende a minimizar erros de imputação [Jordanov et al., 2018].

Métodos de imputação podem obter melhores resultados ao restringir o subconjunto de dados utilizado no processo de imputação. A imputação local, ou *hot-deck*, busca reduzir desvios do processo de imputação, classificando a amostra em um primeiro momento e utilizando apenas grupos de objetos completos que possuam relação de similaridade com o dado ausente [A. Fuller and Kim, 2005]. Tem como principal objetivo reduzir desvios através da classificação da amostra [Soares, 2007].

As relações entre os atributos de amostra podem revelar informações valiosas para o processo de imputação. O experimento de Agrawal et al. [1993], por exemplo, mostrou que, numa padaria, noventa por cento das vendas que envolviam pão e manteiga também envolviam leite. Imagine então uma situação hipotética, um conjunto de dados que possui atributos com tuplas completas para as compras de pão e manteiga, mas alguns registros de compras de leite estão ausentes, para essas mesmas tuplas. A probabilidade dos registros ausentes confirmarem a compra de leite é, no mínimo maior, do que a de não confirmarem. Seguindo este raciocínio, imputar dados que confirmem essa afirmação parece ser a ação mais correta. Entretanto, a existência de relações entre os atributos de uma amostra nem sempre pode ser assumida como verdade absoluta para o processo de imputação. É necessário avaliar a priori o nível de correlação entre os atributos da amostra [Nanni et al., 2012]. Presumir que os valores ausentes de um atributo estão relacionados a todos os demais atributos pode levar a um modelo de imputação mal especificado [Austin and Escobar, 2005]. É necessário inferir um nível maior de confiança a respeito dessas relações. Soares [2007] obteve bons resultados utilizando o algoritmo Principal Components Analysis (PCA) para analisar a correlação de atributos na etapa de seleção do processo de imputação composta, explicada em mais detalhes na próxima seção.

A utilização de tarefas de mineração de dados e técnicas de aprendizado de máquina, precedendo a etapa de imputação, contribui para geração de valores imputados com maior acurácia [Tavares et al., 2018][Tran et al., 2018] [Soares, 2007].

II.3 Imputação Composta

A imputação composta é uma técnica proposta por Soares [2007], baseada no conceito de estratégias e planos de imputação de dados. Combina uma ou mais tarefas usadas na etapa de mineração de dados, como por exemplo, agrupamento e seleção de colunas, para gerar valores imputados mais precisos.

O processo de imputação composta descrito por Soares [2007] baseia-se na definição dos seguintes elementos:

- T_i: uma tarefa do processo de Descoberta de Conhecimento em Bases de Dados (KDD).

Exemplos: T_1 = seleção de atributos, T_2 = agrupamento, T_3 = criação de regras de associação, T_4 = imputação, entre outros.

- $\rightarrow$: operador que define uma ordem de precedência de tarefas de KDD. A expressão $\rightarrow$ significa que a tarefa precede a tarefa . Exemplo: agrupamento $\rightarrow$ imputação significa que a tarefa de agrupamento precederá a de imputação.

- E(v,B): estratégia utilizada no processo de imputação de um atributo v de uma base de dados B. E(v,B) é representada por $T_1 \rightarrow T_2 \rightarrow ... \rightarrow T_m$, onde T_m é necessariamente uma tarefa de imputação.

- A_i : algoritmo utilizado no processo de imputação. Exemplos: A_1 = média, A_2 = algoritmo dos k vizinhos mais próximos.

- $\Rightarrow$: operador que define uma ordem de precedência de aplicação de algoritmos. A expressão $A_i \Rightarrow A_j$ significa que o algoritmo A_i é aplicado antes do algoritmo A_j.

- P(v,B): plano de imputação utilizado no processo de imputação de um atributo de uma base de dados B. P(v,B) é representada por $A_1 \Rightarrow A_2 \Rightarrow ... \Rightarrow A_p$, onde A_p é necessariamente um algoritmo de imputação. Exemplo: $A_1 \Rightarrow A_2 \Rightarrow A_3$ representa a aplicação sequenciada dos algoritmos A_1 = algoritmo dos K centroides, A_2 = análise de componentes principais e A_3 = algoritmo dos k vizinhos mais próximos.

- Ψ_i: instância da aplicação de um algoritmo A_i, segundo parâmetros Θ_i = { Θ_{i1}, Θ_{i2}, ... , Θ_{iq} }. $\Psi_i = f(\mathrm{A}_1, \Theta_i)$

- I(v,B): instância de um plano de imputação de um atributo de uma base de dados , representada por uma sequência ordenada de q instâncias de aplicações de algoritmos. $\Psi_1 \Rightarrow \Psi_2 \Rightarrow .... \Psi_q$, onde Ψ_q é necessariamente uma instância de aplicação de algoritmo de imputação.

- ε(I(v,B)): medida do erro na execução de uma instância de um plano de imputação do atributo .

A figura II.3 ilustra a composição de um plano de imputação.

O conjunto de valores assumidos por um plano de imputação é composto pelos valores da instância que apresentar o menor erro médio de todas as instâncias do referido plano [Soares, 2007].

II.4 Agrupamento de Dados com o Algoritmo K-Means

O K-Means é um algoritmo apresentado por MacQueen [1967], que busca particionar os dados em k grupos cujas instâncias apresentem semelhanças entre si. É um algoritmo consolidado e com ampla utilização em tarefas de agrupamento.

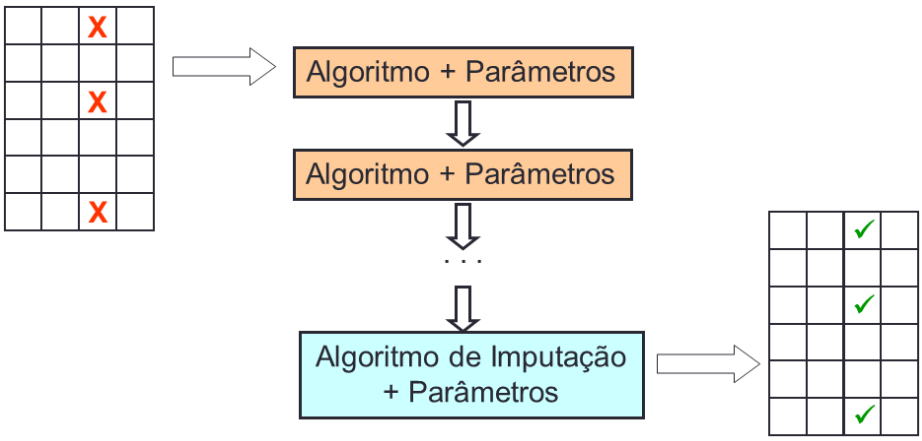

Figura II.3: Imputação Composta. Fonte: adaptado de Soares [2007].

O algoritmo funciona da seguinte forma:

1) São passados como entrada o conjunto de dados D e o número k de grupos que se deseja encontrar.

2) Arbitrariamente, são selecionados k objetos de D, denominados centroides.

3) Para cada objeto de D, é verificada a distância do mesmo em relação aos centroides. O objeto é agrupado junto ao centróide mais próximo.

4) Ao fim da verificação de todos os objetos de D, é calculada a média simples entre os objetos de cada grupo, definindo os valores dos novos centroides.

5) O processo reinicia no passo 3, até que não ocorra mais mudança de grupo entre os objetos ou um número máximo definido de iterações seja alcançado.

A figura II.4 denota o funcionamento do algoritmo:

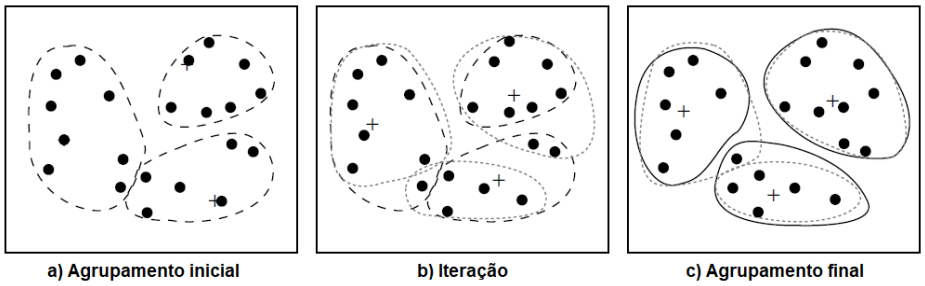

Figura II.4: O algoritmo K-Means. Fonte: adaptado de Han et al. [2011].

O algoritmo K-Means pressupõe que cada tupla seja agrupada de forma que pertença a um único grupo. Entretanto, em casos reais, é possível observar situações em que um objeto pode pertencer a mais de um grupo [Han et al., 2011].

II.5 Seleção de Atributos com o Algoritmo PCA

O algoritmo PCA é um método de redução de dimensionalidade. Suponha que em uma amostra, os dados a serem reduzidos consistem em tuplas ou vetores de dados descritos por n atributos ou dimensões. O algoritmo busca por vetores ortogonais n-dimensionais que possam melhor representar os dados, onde [k <= n] [Han et al., 2011]. Os dados originais são projetados em um espaço menor, resultando em redução de dimensionalidade. É um algoritmo muito utilizado para compressão de imagens.

Além da redução de dimensionalidade, o algoritmo PCA é uma maneira de identificar padrões em dados e expressá-los de forma a destacar suas semelhanças e diferenças [Smith, 2005]. Neste trabalho, será utilizado para inferir a correlação entre os valores dos atributos em um conjunto de dados. Dessa forma, permite selecionar os mais representativos para utilização no processo de imputação.

Funcionamento do algoritmo:

1. Os dados de entrada são normalizados, para evitar que valores de grandezas maiores enviesem os cálculos.

2. São computados k vetores ortonormais, que servem como base para os dados de entrada normalizados. Vetores unitários são gerados, de forma que cada ponto em uma direção é perpendicular aos demais. Esses vetores são denominados componentes principais.

3. Os componentes principais são ordenados em ordem decrescente de significância ou força. Servem como um novo grupo de eixos representando os dados, provendo informação sobre a variância. A matriz de componentes possui os eixos ordenados da esquerda para a direita em ordem de representatividade.

4. Como os componentes são classificados em ordem decrescente de significado, o tamanho dos dados pode ser reduzido eliminando os componentes menos representativos.

II.6 Imputação com o Algoritmo K-NN

O algoritmo *K-Nearest-Neighbor* (K-NN) tem como base o aprendizado por analogia, isto é, compara tuplas que sejam similares a tupla analisada [Han et al., 2011]. Em uma amostra com n atributos, cada tupla representa um ponto em um espaço n-dimensional. As tuplas são armazenadas em um espaço de padrão de tamanho n. Dada uma tupla desconhecida, o algoritmo busca o espaço de padrão para as k tuplas mais próximas da tupla desconhecida. Essas k tuplas mais próximas são os vizinhos mais próximos da tupla desconhecida.

O método de imputação baseado no algoritmo K-NN é amplamente utilizado e pode apresentar bons resultados para valores desconhecidos que sigam distribuição MCAR [Batista, 2003]. Valores imputados com o algoritmo K-NN são geralmente mais bem comportados, uma vez que são relacionados a valores de outros atributos. Normalmente, quanto mais atributos com valores desconhecidos, e quanto maior a quantidade de valores desconhecidos, mais simples são os classificadores induzidos [Batista and Monard, 2003b]. Dessa forma, a imputação deve ser cuidadosamente aplicada, sob o risco de simplificar demasiadamente o problema em estudo.

Descobrir o melhor valor k para imputação não é uma tarefa objetiva. Uma estratégia que apresenta bons resultados, é considerar a substituição de valores ausentes pelo correspondente valor de atributo da instância completa mais semelhante no conjunto de dados [Hruschka et al., 2005]. Em outras palavras, utilizar o algoritmo K-NN com $[k = 1]$ e uma função de distância, como a euclidiana, para identificar os vizinhos mais próximos. Entretanto, Jonsson and Wohlin [2004] sugerem que, para alguns casos, como em dados de pesquisas do tipo *likert data*, utilizar k como a raiz quadrada do número de casos completos (arredondado para o número inteiro mais próximo) pode apresentar melhores resultados. Logo, no contexto de imputação, o valor ótimo de k pode variar dependendo das características dos dados.

A principal desvantagem do algoritmo K-NN é que, sempre que os vizinhos mais semelhantes são procurados, o algoritmo pesquisa todas as tuplas do conjuntos de dados. Isso pode ser um ponto crítico para a execução em grandes volumes de dados [Batista and Monard, 2003a].

II.7 Computação Paralela e Distribuída

Nos últimos vinte anos, o volume de dados gerados por sistemas informatizados cresceu em larga escala. De acordo com um informativo do grupo *International Data Corporation (IDC)*, em 2011 o volume de dados gerados no mundo cresceu quase nove vezes em relação aos anos anteriores [Chen et al., 2014]. A projeção é de que o volume de dados dobrará a cada dois anos em um futuro próximo. O planejado telescópio SKA (Square Kilometre Array), por exemplo, irá produzir até um milhão de terabytes de dados brutos por dia a partir de 2020 [Jagadish et al., 2014].

Vultuosos volumes de dados podem ser gerados tanto por sistemas de business convencional, como os encontrados em sistemas Enterprise Resource Planning (ERP), quanto por sistemas mais complexos, como os biomoleculares, georreferenciados ou espaciais, que demandam grandes esforços de análise e processamento.

A chamada IoT também contribui para o aumento do volume de dados produzidos. Tem como idéia básica a presença generalizada de uma variedade de coisas ou objetos, como tags, sensores, atuadores, telemóveis, entre outros, que são capazes de interagir uns com os outros para atingir objetivos comuns [Atzori et al., 2010]. Praticamente qualquer dispositivo é capaz de gerar dados,

normalmente transferidos para grandes *data centers* e nuvens.

Esses imensos volumes de dados remetem ao termo *Big Data*. O modelo dos cinco Vs [Chen et al., 2014] é amplamente utilizado para a caracterização de Big Data: volume (grandes volumes), variedade (diversas modalidades), velocidade (geração rápida), veracidade (dados verdadeiros) e valor (valor enorme, porém de baixa densidade) são características de bases *big data*.

Métodos para descobrir conhecimento em *big data* são fundamentalmente diferentes dos utilizados na análise estatística tradicional ou em pequenas amostras [Jagadish et al., 2014]. Dados da escala *big data* normalmente apresentam ruído, são dinâmicos, heterogêneos e inter-relacionados [Chen et al., 2014]. Contudo, possuem maior valor significativo, pois a mineração de dados nesses ambientes ignora flutuações individuais, identificando padrões ocultos e gerando conhecimento de maneira mais confiável [Jagadish et al., 2014].

A descoberta de conhecimento nesse nível demanda uma estrutura de processamento de dados apoiada em arquiteturas de computação distribuída [Chen et al., 2014]. Devem compor uma plataforma de alto desempenho, capaz de processar grandes volumes de dados em tempo satisfatório, além de conseguir processar tarefas de alto custo computacional de forma eficiente, através da utilização de recursos distribuídos [McPhillips et al., 2009]. Para processar dados dessa magnitude são utilizados *clusters* de computadores [Jagadish et al., 2014], unidades compostas por múltiplos nós de processamento, muitas vezes em nuvem, possibilitando a divisão do trabalho computacional e o processamento paralelo de tarefas.

II.7.1 Apache Spark

Apache Spark é uma ferramenta unificada para processamento de dados distribuídos [Zaharia et al., 2016a]. Possui um modelo de programação similar ao consagrado *map-reduce*, otimizado por uma abstração de compartilhamento de dados chamada *Resilient Distributed Datasets (RDD)*.

Através dos RDDs Spark consegue atuar em uma ampla faixa de processamento, de forma que processamentos que anteriormente necessitariam de *engines* separadas, como SQL, *streaming*, *machine learning*, entre outros, possam ser executados de maneira otimizada. Dentre os componentes principais da arquitetura Spark, se destacam: o *Driver Program* (executa o programa), o *Cluster Manager* (gerencia a utilização dos recursos disponíveis) e os *Worker Nodes* (executam as tarefa computacionais) [Zaharia et al., 2016a]. A figura II.5 ilustra a arquitetura Spark:

A ferramenta Spark possui diversos componentes com propósitos especifícios, dentre os quais se destacam como mais utilizados: Spark SQL, MLlib e Spark Streaming.

- Spark SQL é uma biblioteca complementar ao Apache Spark, a qual provê uma abordagem de utilização próxima a linguagens de bancos de dados relacionais. Integra o processamento relacional e procedural através de uma *interface* declarativa chamada *DataFrame* [Armbrust

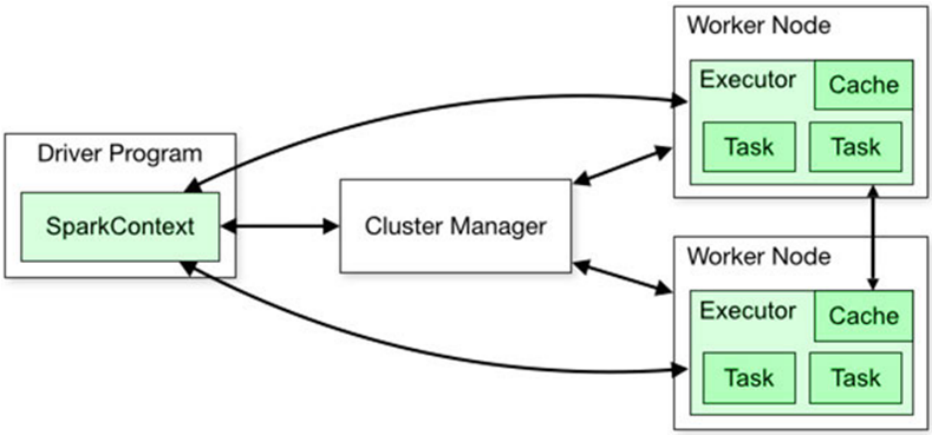

Figura II.5: Arquitetura Spark. Fonte: adaptado de Zaharia et al. [2016b].

et al., 2015]. Além disso, possui um otimizador extensível chamado *Catalyst*. Desenvolvido na linguagem *Scala*, facilita a composição de regras, o controle de geração de código e a definição de pontos de extensão. Tem como objetivo criar planos de execução que favoreçam o paralelismo e a distribuição de tarefas computacionais. Spark SQL é capaz de gerar ganhos significativos em comparação com o próprio Spark [Armbrust et al., 2015].

- MLlib é a biblioteca de aprendizado de máquina da ferramenta Spark. Oferece implementações eficientes de algoritmos de aprendizado de máquina para diferentes tarefas como: classificação, regressão, filtragem colaborativa, agrupamento e redução de dimensionalidade [Meng et al., 2016b]. Além disso, provê uma variedade de métodos estatísticos, de álgebra linear e otimização. A biblioteca foi desenvolvida em Scala, tendo C++ como linguagem nativa. Oferece APIs para as linguagens Java, Scala, Python e R.

- Spark Streaming é uma extensão da API Spark que permite que engenheiros de dados e cientistas de dados processem dados em tempo real [Lee and Damji, 2016]. Suporta diversas fontes de dados, desde arquivos *flat* até Kafka, Flume, Amazon Kinesis, entre outros.

II.8 Aprendizado de Máquina e Classificadores Ensemble

Aprendizado de máquina é um subárea da inteligência artificial, investiga como os computadores podem aprender (ou melhorar seu desempenho) com base em dados, sem que tenham sido especificamente programados para tal [Han et al., 2011]. Algoritmos de aprendizado de máquina podem reconhecer padrões complexos e tomar decisões inteligentes com base em dados.

O processo de geração de modelos a partir de dados é chamado de aprendizado ou treinamento e o modelo estabelecido é chamado de hipótese, denotada pela letra $\mathbf{H}$, $\mathbf{H(x)}$ quando aplicada a um exemplo de treinamento $\mathbf{x}$ [Zhou, 2012]. De forma geral, modelos de aprendizado de máquina

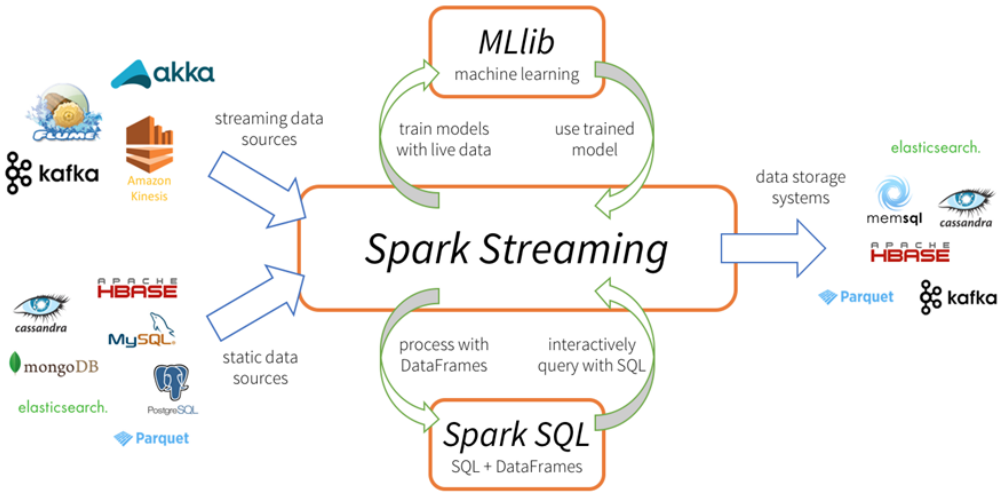

Figura II.6: Apache Spark: MLlib, Spark Streaming e Spark SQL. Fonte: adaptado de Lee and Damji [2016].

são classificados em dois tipos: supervisionados e não supervisionados [Han et al., 2011]. Na aprendizagem supervisionada, o objetivo primário é a classificação dos dados. A aprendizagem tem como base os exemplos rotulados no conjunto de dados de treinamento, para a previsão de um novo valor (rótulo) aferido pelo modelo. A aprendizagem não supervisionada não depende de rótulos conhecidos. O objetivo é descobrir informações inerentes a distribuição dos dados. Uma tarefa típica desse tipo de aprendizado é o agrupamento, ou seja, identificar grupos cujas instâncias (vetores de características) apresentem similaridade entre si.

Métodos *ensemble* consistem no treinamento de mais de um modelo de aprendizado para a solução do mesmo problema [Zhou, 2012]. Em comparação com abordagens comuns de aprendizado de máquina, tendem a apresentar melhores resultados, pois reduzem a variância e melhoram a acurácia do processo de decisão através da utilização de múltiplos classificadores, chamados *base learners* [Zhang and Ma, 2012].

Classificadores *ensemble* tem sido utilizados na resolução de diversos problemas de aprendizado de máquina, como a seleção ou ausência de características, aprendizado incremental, entre outros [Zhang and Ma, 2012]. A capacidade de generalização dos métodos *ensemble* geralmente é maior do que a de modelos individuais, chamados *weak learners* [Zhou, 2012]. Uma vez que os classificadores sejam independentes entre si, a combinação de seus resultados tende a apresentar menor probabilidade de erro do que a produzida por classificadores individuais [Zhang and Ma, 2012]. A figura II.7 ilustra a utilização de classificadores *ensemble*.

Métodos *ensemble* tem se mostrado efetivos quando utilizados em conjunto com diferentes técnicas para o treinamento de classificadores [Duda et al., 2000]. Como exemplo de métodos para estimativa e comparação de modelos de classificadores *ensemble*, pode-se citar como mais utilizados [Zhou, 2012]: *Bootstrapping*, *Bagging*, *Boosting* e *Adaboost*.

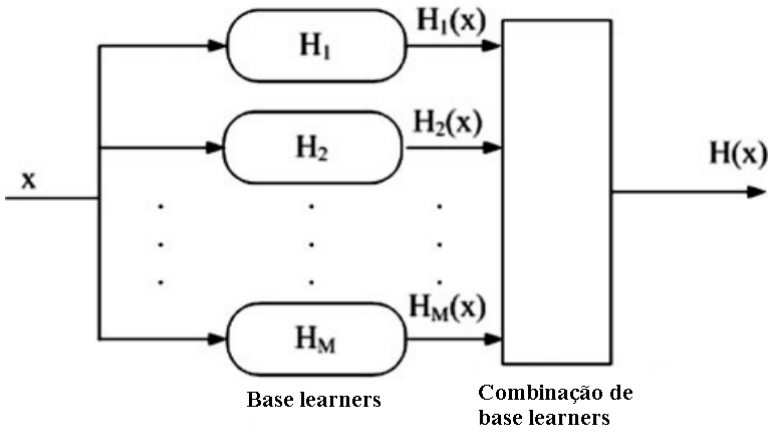

Figura II.7: Classificadores ensemble. Fonte: adaptado de Zhang and Ma [2012].

Primeiramente são abordadas as técnicas de Bootstrapping e Bagging, de maneira mais detalhada, pois são imprescindíveis para a compreensão deste trabalho. Posteriormente as técnicas Boosting e Adaboost são apresentadas de forma mais resumida.

Bootstrapping é um método de origem na área da estatística. Tem como base a criação de amostras de propósito geral, onde vários conjuntos de treinamento (sem intervalo) são obtidos aleatoriamente com a substituição, a partir do conjunto de dados original [Duda et al., 2000]. Em um conjunto de dados com **N** amostras, cada instância é selecionada com probabilidade 1 / **N**; consequentemente, após **N** sorteios (com **N** máximo), a probabilidade de que uma determinada instância não seja selecionada é denotada na equação II.8, mostrando que se trata de um método bastante preciso, mesmo que **N** seja moderadamente grande [Zhang and Ma, 2012]. Dessa forma é possível afirmar que cada amostra contém aproximadamente 63,2 % das instâncias originais.

$$\left(1 - \frac{1}{N}\right)^{N} \approx \exp(-1) \approx 0.368;$$

Figura II.8: Probabilidade de um *base learner* conter as instâncias originais. Fonte: adaptado de Zhang and Ma [2012].

Bagging (que significa agregação bootstrap) é uma técnica que utiliza a amostragem bootstrap para reduzir a variação e/ou melhorar a precisão de modelos preditivos (pode ser usado para tarefas de classificação e regressão) [Zhang and Ma, 2012]. Zhou [2012] detalha o funcionamento do método *Bagging* da seguinte forma:

> "Dado um conjunto de treinamento contendo um número **m** de exemplos, uma amostra com **m** exemplos de treinamento será gerada por amostragem com substituição. Alguns exemplos originais aparecerão mais de uma vez, enquanto alguns exemplos originais não estarão presentes na amostra. Aplicando o processo **T** vezes, **T** amostras de **m** exemplos serão obtidas. Então a partir de cada subamostra um base learner pode ser treinado

através de um algoritmo de aprendizado"[Zhou, 2012].

A figura II.9 ilustra o funcionamento da técnica de *Bagging*:

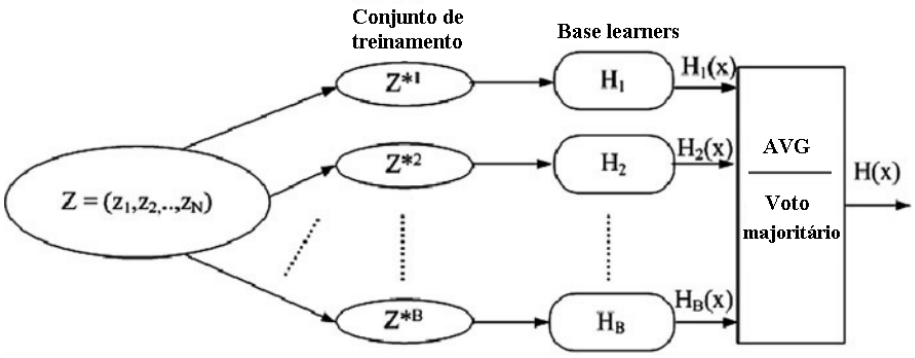

Figura II.9: Classificador ensemble Bagging. Fonte: adaptado de Zhang and Ma [2012].

A etapa de combinação dos valores aferidos pelos *base learners* utiliza a média simples, para tarefas de regressão, e o voto majoritário (cada classificador sugere um valor, o valor sugerido mais vezes é escolhido), para tarefas de classificação [Zhang and Ma, 2012]. Zhang and Ma [2012] afirma ainda que a combinação de múltiplos classificadores diminui o erro, pois reduz o componente de variância da decomposição viés-variância, sendo a redução da variância, proporcional ao número de classificadores utilizados na combinação.

Em comparação com o processo de aprendizagem com um classificador, de forma convencional, isto é, utilizando apenas o conjunto completo de treinamento, *Bagging* apresenta duas vantagens principais [Zhang and Ma, 2012]:

- aumenta a estabilidade e precisão do classificador;
- reduz a variação do classificador, em termos da decomposição de viés-variância;

Em geral, modelos com *Bagging* melhoram o reconhecimento de classificadores instáveis, uma vez que efetivamente amenizam descontinuidades [Duda et al., 2000]. O algoritmo *Bagging* é detalhado na figura II.10.

Entrada: Conjunto de dados $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Algoritmo de aprendizado $\mathfrak{L}$;
Número de *base learners* T.

Processo:
1. **para** $t = 1, \ldots, T$:
2. $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$ % $\mathcal{D}_{bs}$ é a distribuição bootstrap
3. **fim**

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

Figura II.10: O algoritmo Bagging. Fonte: adaptado de Zhang and Ma [2012].

A estratégia do método *Boosting* é diferente, consiste em treinar múltiplos *weak learners* e combiná-los de forma a criar um único *strong learner*, capaz de inferir classificações com menor probabilidade de erro [Zhang and Ma, 2012]. A ideia central é: treinar múltiplos classificadores simples e gerar um classificador mais complexo é mais fácil do que aprender com um único classificador complexo [Zhou, 2012]. A figura II.11 ilustra o funcionamento do método *Boosting*:

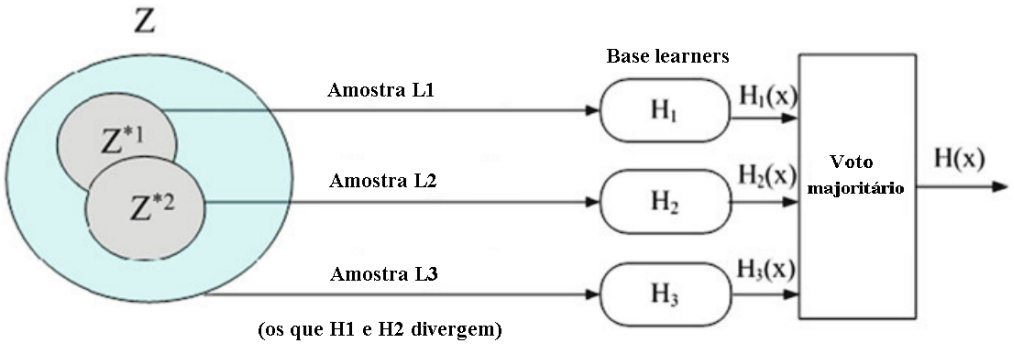

Figura II.11: Classificador ensemble Boosting. Fonte: adaptado de Zhang and Ma [2012].

Essencialmente, *Boosting* consiste no uso repetitivo dos algoritmos de aprendizado dos *weak learners*, com diferentes pesos do conjunto de treinamento, gerando uma sequência de *weak learners* combinados entre si [Zhou, 2012]. O peso atrelado a cada instância do conjunto de treinamento, em cada rodada do algoritmo, depende da precisão dos classificadores anteriores, permitindo assim que o algoritmo concentre atenção nas amostras que ainda estão incorretamente classificadas [Zhang and Ma, 2012].

A figura V.3 compara os métodos *Bootstrapping, Bagging e Boosting* quanto a utilização do conjunto de treinamento:

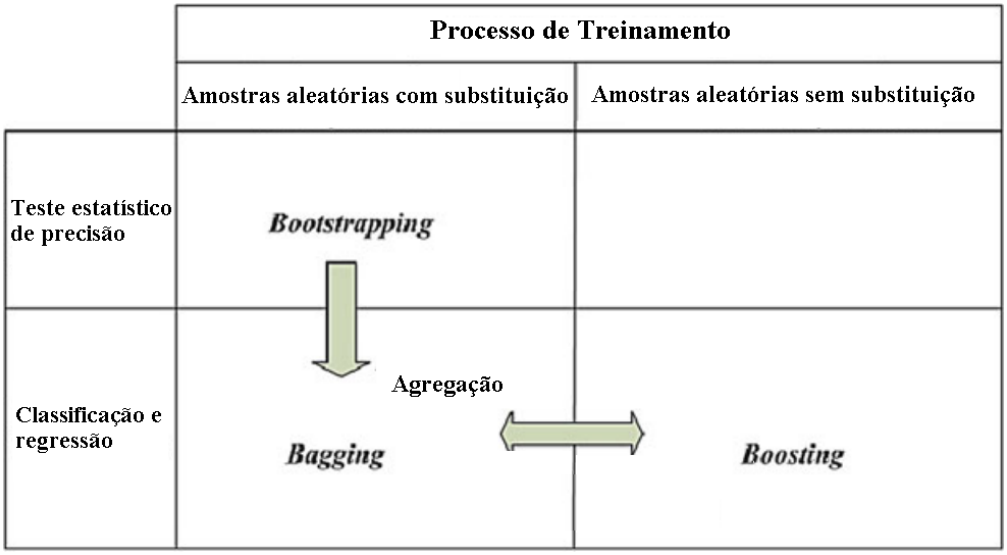

Figura II.12: Classificadores Bootstrapping, Baging, Boosting e a utilização do conjunto de treinamento. Fonte: adaptado de Zhang and Ma [2012].

As variantes do algoritmo de *Boosting* diferem na escolha dos *base learners* e no critério de atu-

alização dos pesos nos conjuntos de treinamento [Duda et al., 2000]. *AdaBoost* (*Adaptive Boosting*) possui duas diferenças fundamentais em relação ao *Boosting* [Zhang and Ma, 2012]:

- instâncias são criadas com base em conjuntos subsequentes provenientes de distribuições amostrais, iterativamente atualizadas, do conjunto de treinamento.
- os classificadores são combinados através de votação majoritária, atrelada a pesos, compostos pelos erros dos classificadores de treinamento, os quais por si são atrelados a pesos da distribuição amostral.

II.9 Workflow Cientifíco

Atualmente a computação tem se estabelecido como uma das bases do avanço científico [Deelman et al., 2009]. Supercomputadores são capazes de simular modelos complexos envolvendo diversos passos computacionais. A execução de experimentos envolve tarefas repetitivas, com movimentos cíclicos dos dados [McPhillips et al., 2009]. São gerados resultados de entrada e saída durante a realização de tarefas isoladas e ao longo de toda a execução da repetição. Esse processo se repete até a obtenção dos resultados finais. Sistemas de *Workflow* Científico automatizam tarefas cíclicas, de forma que os cientistas possam se concentrar somente nas pesquisas e não no gerenciamento computacional [Deelman et al., 2009].

A área de *e-Science* remete à utilização de métodos de obtenção de resultados científicos através da utilização de computação intensiva. De acordo com [Deelman et al., 2009], sistemas de workflows de *e-Science* tem como principal objetivo oferecer ambientes de programação especializados, que simplifiquem o esforço de programação despendido por cientistas para orquestrar experimentos científicos em ambientes computacionais.

A composição é uma etapa importante na elaboração de workflows. Nessa etapa o cientista especifica os passos e as dependências envolvidas desde o início da execução até o final da realização do experimento [Davidson and Freire, 2008]. Uma forma de simplificar a elaboração de workflows é fornecer ferramentas para a composição dos experimentos. De acordo com Deelman et al. [2009], em um nível abstrato, workflows são compostos por séries de unidades funcionais, sejam componentes, tarefas, serviços ou dependências, executados em uma ordem definida. Podem ser representados por grafos *Directed Acyclic Graph (DAG)*, redes de *petri*, *Business Process Model and Notation* (BPMN), *Unified Modelling Language (UML)*, entre outros.

Sistemas de workflow são classificados de duas formas:

a) *Data centric* - o foco é a análise e transformação dos dados.

b) *Process centric* - o foco é a definição do fluxo de negócio.

Cientistas tendem a ter uma visão centrada nos dados para suas análises [McPhillips et al.,

2009]. Passos computacionais são importantes, mas a obtenção, análise e criação de informação através dos dados tendem a ser mais relevantes.

Uma questão fundamental na temática de workflows é a capacidade de proveniência do sistema. Proveniência permite interpretar e entender resultados, através da análise da sequência de passos executados desde o início do processo até o resultado final [Freire et al., 2008]. Dessa forma, é possível obter informações sobre o encadeamento das tarefas, avaliando se estas foram executadas seguindo os procedimentos esperados. Outro ponto fundamental a ser garantido pela proveniência é a capacidade de reprodutibilidade dos experimentos [Davidson and Freire, 2008]. Sistemas de workflows devem ser capazes de salvar os dados da execução para que possam ser reproduzidos posteriormente.

Quando se trata de tarefas computacionais, existem duas formas de proveniência [Freire et al., 2008]: prospectiva e retrospectiva. A proveniência prospectiva captura a especificação de uma tarefa computacional (seja um script ou um workflow) e corresponde às etapas que devem ser seguidas para gerar um determinado resultado. A proveniência retrospectiva capta as etapas executadas, bem como informações sobre o ambiente usado para derivar o resultado - em outras palavras, é uma espécie de *log* detalhado da execução das tarefas computacionais.

Capítulo III Trabalhos Relacionados

Nesse capítulo serão abordados trabalhos relacionados ao tema imputação, no que tange a utilização de algoritmos adaptados para essa finalidade, seja na etapa de pré-processamento ou na imputação propriamente dita. Além disso, serão citados também trabalhos relacionados aos temas de utilização de classificadores *ensemble* e computação paralela e distribuída, fundamentais para a composição deste trabalho.

III.1 Imputação

Tran et al. [2018] propõem a utilização de tarefas de agrupamento e seleção de atributos, integradas ao processo de imputação, para melhorar e eficiência e evitar a perda de precião dos valores imputados.

Tavares et al. [2018] estudaram o impacto da utilização do algoritmo dos k-vizinhos mais próximos em comparação ao uso da média no processo de imputação global, bem como explora o uso da técnica de imputação hot-deck com o algoritmo de agrupamento k-Means precedendo a imputação com k-NN. Os resultados revelam interessante redução da margem de erro obtida em simulações em três bases de dados com diferentes características.

O trabalho de García-Laencina et al. [2015] avaliou a utilização de quatro diferentes métodos de imputação: K-NN, árvores de classificação, regressão logística e Support-vector Machine (SVM). Os melhores resultados foram obtidos com o algoritmo K-NN.

Silva and Zárate [2014] avaliam a importância da complementação de dados ausentes no processo de KDD, os métodos de imputação mais utilizados e a importância do mecanismo gerador da ausência na escolha do método adequado.

Cheema [2014] realizou uma análise da literatura referente ao tema de complementação de dados ausentes. Diversos métodos de imputação foram avaliados considerando o impacto do tamanho da amostra, da proporção de dados ausentes, o desempenho computacional e a acurácia dos dados imputados.

Luengo et al. [2012] analisaram a combinação de 23 métodos de classificação com 14 abordagens diferentes de imputação para o tratamento de dados ausentes.

Medina [2012] obteve bons resultados em experimentos utilizando a técnica de imputação com-

posta. A tarefa de seleção foi implementada com algoritmos genéticos e a de agrupamento com Redes de *Kohonen*.

Wohlrab and Fürnkranz [2011] avalia a utilização de algoritmos de divisão e conquista adaptados para a tarefa de imputação, combinando-os com a seleção de atributos para a composição do valor imputado. Alguns métodos apresentaram bons resultados para bases de dados com elevados percentuais de ausência.

Zhang et al. [2010] apresentam um *framework* de imputação que transforma valores completos em micro recursos de estruturas de dados. Além disso, utiliza uma variação do algoritmo K-NN adaptado para buscas nessas estruturas. O método apresentou bons resultados em bases de dados com elevados percentuais de dados ausentes.

García-Laencina et al. [2009] apresentam uma variação do algoritmo K-NN para classificação e imputação simultânea de valores ausentes. Utiliza o conceito de informação mútua, que é uma medida natural para calcular distâncias entre variáveis aleatórias. A distância entres os atributos que possuem valores ausentes é considerada para criação de classificadores, melhorando o valor imputado.

Ferlin [2008] apresenta uma abordagem em cascata para tratar a imputação multivariada, através do reuso. Valores previamente imputados são reutilizados para a imputação de diferentes grupos de objetos, criando um efeito em cascata. O método utiliza redes neurais *self-organizing map* (SOM) na etapa de agrupamento.

Castaneda et al. [2008] estuda um aprimoramento do processo de imputação multivariada, através da composição de um workflow de imputação iterativa, capaz de automatizar a configuração, execução, e avaliação de técnicas de imputação.

Saar-Tsechansky and Provost [2007] apresentam um método de imputação híbrida com fatores de ajuste quanto ao custo computacional, a complexidade do modelo e a precisão do valor imputado.

Huang and Lee [2004] implementaram uma modificação no algoritmo K-NN que utiliza *Grey Relational Analysis (GRA)* como métrica de distância para determinar os vizinhos mais próximos de instâncias com valores ausentes.

Li et al. [2004] avaliaram a utilização do algoritmo Fuzzy K-Means como tarefa de agrupamento predecessora a de imputação. Nessa versão do algoritmo, o centroide é calculado como a média de todos os pontos, ponderada pelo grau de inclusão do objeto em cada grupo.

Métodos de imputação normalmente pressupõem que a distribuição da ausência seja MAR. Allison [1999] aborda uma técnica de imputação múltipla em que essa premissa não é necessária. O método identifica relações de linearidade entre os atributos e utiliza um modelo de regressão linear para gerar o valor imputado. Entretanto, essa técnica não obteve um nível alto de precisão para todas bases de dados testadas.

A tabela III.1 apresenta de forma objetiva os autores e as abordagens de imputação utilizadas.

Tabela III.1: Autores e abordagens de imputação utilizadas.

Autor	Abordagem
Allison [1999]	Imputação múltipla | regressão linear
Li et al. [2004]	Fuzzy K-Means | média ponderada
Huang and Lee [2004]	K-NN | GRA
Saar-Tsechansky and Provost [2007]	Imputação híbrida
Soares [2007]	Imputação composta | imputação *hot-deck* | imputação múltipla | K-NN | K-Means | PCA | redes *backpropagation*
Castaneda et al. [2008]	Imputação multivariada e iterativa
Ferlin [2008]	Imputação em cascata | redes SOM
García-Laencina et al. [2009]	K-NN
Zhang et al. [2010]	K-NN
Wohlrab and Fürnkranz [2011]	Divisão e conquista
Medina [2012]	Imputação composta | algoritmos genéticos | redes de *Kohonen*
Luengo et al. [2012]	K-NN | WK-NN | K-Means | K-means | SVMI | SVDI | BPCA | LLSI |
Cheema [2014]	AVG | regressão linear | imputação múltipla | imputação *hot-deck*
Silva and Zárate [2014]	Imputação *hot-deck* | imputação múltipla | imputação composta | K-NN | redes SOM
García-Laencina et al. [2015]	K-NN | árvore de classificação | regressão logística | SVM
Tavares et al. [2018]	Imputação *hot-deck* | K-NN | K-Means | AVG
Tran et al. [2018]	Imputação múltipla | K-Means | C4.5 | *naive-bayes*

III.2 Computação em Larga Escala

Um dos requisitos principais de sistemas de workflow é conseguir processar tarefas de alto custo computacional de forma eficiente, através da utilização de recursos distribuídos [McPhillips et al., 2009]. A ferramenta Apache Spark possui recursos nesse sentido, tendo apresentado bons resultados para processamento de dados em larga escala [Zaharia et al., 2016a].

O experimento de Gopalani and Arora [2015] é um bom exemplo do ganho de desempenho de algoritmos executados em Spark frente a outras soluções de computação distribuída. O Apache Mahout é uma ferramenta otimizada para execução de algoritmos de aprendizado de máquina. As tabelas III.2 e III.3 apresentam, respectivamente, os resultados da execução do algoritmo K-Means, no mesmo ambiente e com a mesma quantidade de dados, utilizando Spark e Mahout.

Como exemplo de utilização de Spark para tarefas de imputação, Qu et al. [2016] apresentam

Tabela III.2: Spark (pacote MLib). Fonte: adaptado de Gopalani and Arora [2015].

Tamanho dos dados	Número de *cores*	Tempo (s)
62MB	1	18
1240MB	1	149
1240MB	2	85

Tabela III.3: Apache Mahout. Fonte: adaptado de Gopalani and Arora [2015].

Tamanho dos dados	Número de *cores*	Tempo (s)
62MB	1	44
1240MB	1	291
1240MB	2	163

uma modificação no algoritmo Apriori para imputação baseada em regras de associação. O algoritmo Apriori original é aplicado a dados sob uma abordagem boleana e não se aplica a dados multidimensionais. Logo os dados foram convertidos para o formato binário-simétrico, para possibilitar a utilização de regras de associação.

Trabalhos recentes buscam otimizar o funcionamento do algoritmo PCA para ambientes *Big Data*, com foco na melhora de desempenho. Elgamal et al. [2015] implementaram mudanças no algoritmo PCA desenvolvido para Apache Spark. Através da alavancagem da esparsidade de matrizes, foi observado ganho de desempenho em operações matemáticas.

[Wu et al., 2016] apresenta uma versão do algoritmo PCA otimizada para computação paralela e distribuída, de forma a ajustar a utilização da matriz de correlação ao modelo *map-reduce*. O algoritmo foi executado em Spark e apresentou ganhos significativos de *speedup* no experimento realizado.

Existem adaptações do algoritmo K-NN que apresentam resultados melhores para dados com características específicas. Por exemplo, para séries temporais *Big Data*, Talavera-Llames et al. [2016] propõem um novo método de previsão com base na técnica *Weighted Nearest Neighbours (WNN)*, que utiliza vetores correlacionados para identificar vizinhos mais próximos.

O método K-NN-IS apresenta melhorias para execução do algoritmo K-NN em ambientes *Big Data* com Spark [Maillo et al., 2017]. A tarefa de *map* computa a distância dos vizinhos mais próximos em diferentes particionamentos de dados. Múltiplos *reducers* processam os vizinhos da lista gerada na tarefa de *map*.

III.3 Classificadores Ensemble

Nanni et al. [2012] avalia o desempenho de diferentes métodos estatísticos e de aprendizado de máquina, aplicados a imputação, incluindo uma nova abordagem de imputação múltipla combinada com classificadores *ensemble*. O método proposto obteve bons resultados, inclusive em percentuais

de ausência elevados (maiores que 30%).

O trabalho de Nanni et al. [2012] foi o único encontrado na literatura a avaliar a utilização de classificadores *ensemble* aplicados especificamente a problemas de imputação. Dada a obtenção de resultados irregulares com a imputação múltipla e a utilização de comitês de complementação de dados ausentes [Soares, 2007], optou-se nesse trabalho por utilizar como base a técnica de imputação composta.

Capítulo IV Métodos Ensemble e Imputação em Larga Escala

Esse capítulo discorrerá sobre o método de imputação proposto neste trabalho, que consiste em criar uma tarefa de *ensemble* combinada à técnica de imputação composta. Além disso, será abordado o algoritmo Motor de Imputação, que tem como objetivo a aplicação da referida técnica possibilitando a execução de planos de imputação em larga escala.

IV.1 Imputação Composta e Ensemble

Na técnica de Imputação Composta criada por Soares [2007] e detalhada na seção II.3, um plano de imputação é a representação de uma sequência de tarefas de mineração de dados. Essas tarefas definem uma *estratégia*, a qual é associada à execução de algoritmos em uma ordem predefinida. O valor sugerido é gerado na última tarefa da sequência, a tarefa de imputação.

Conforme explicado na seção II.8, *Bagging* é um método de classificação *ensemble* que utiliza *Bootstrapping* com substituição. Dessa forma, tuplas do conjunto original podem existir mais de uma vez, ou até mesmo não existir, nos subconjuntos de dados de cada *base learner*. O método possui ainda uma etapa de combinação, a qual é sugerida a utilização da média simples para tarefas de regressão [Zhang and Ma, 2012].

A proposta deste trabalho é um método que busca gerar um valor imputado mais preciso, através da criação de uma tarefa de *ensemble*, utilizando *Bagging*, aplicada à imputação composta. Dessa forma, instâncias de um mesmo plano de imputação são criadas para cada subconjunto de dados, compondo *base learners* de imputação. Posteriormente, os valores gerados pelos *base learner* de imputação são combinados através da média simples. O método funciona da seguinte forma:

1. De acordo com o parâmetro T (número de *base learners*) informado, são criados T subconjuntos do conjunto original, com substituição.

2. Instâncias do plano de imputação são associadas aos subconjuntos de dados, configurando os *base learners* de imputação. Essa etapa tem suas iterações executadas de forma independente, apresentando oportunidade de paralelização.

3. Os *base learners* de imputação são executados e seus valores combinados através da média simples. O resultado é sugerido como valor imputado. Essa etapa também favorece a computação

paralela e distribuída, uma vez que os *base learners* de imputação são independentes.

A figura IV.1 ilustra o funcionamento do método.

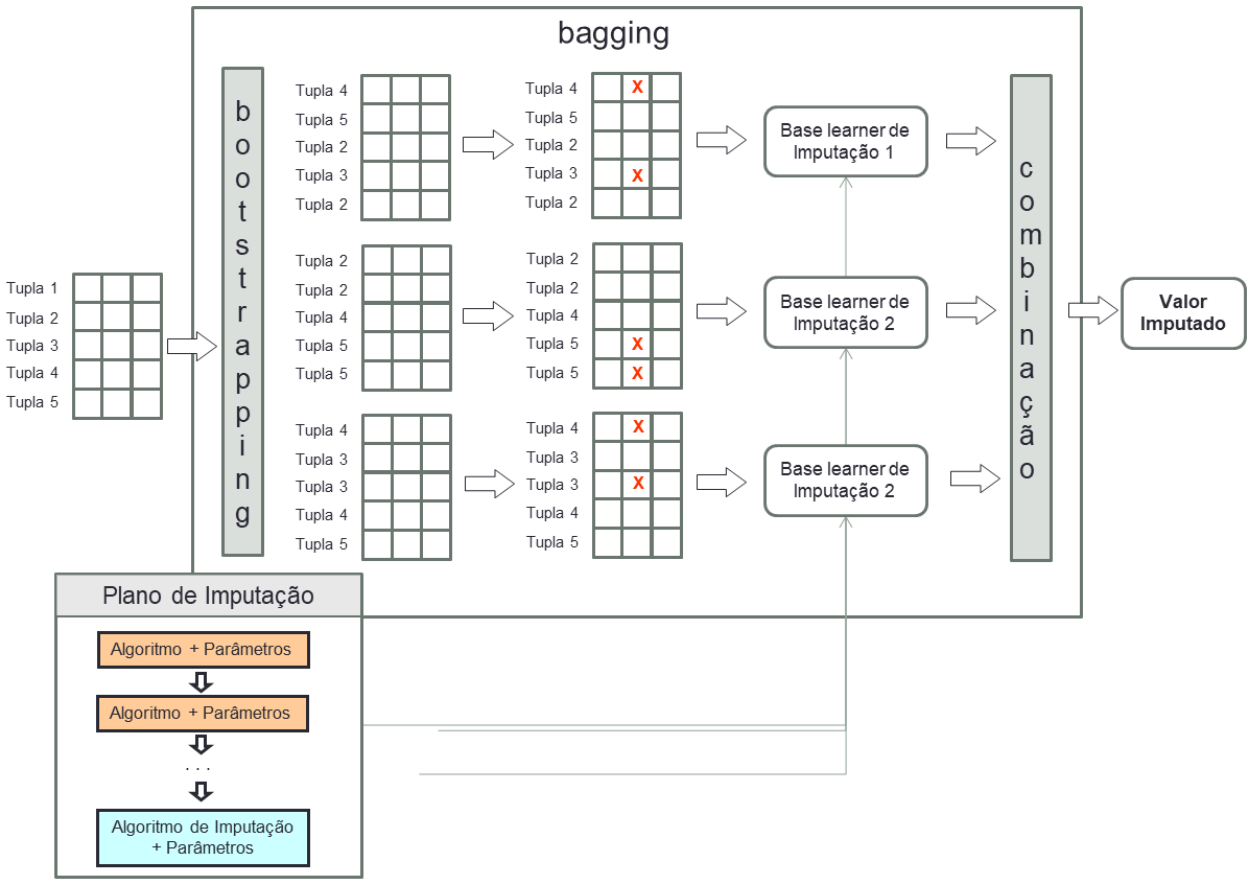

Figura IV.1: Imputação Composta utilizando a tarefa de *ensemble* com *Bagging*.

A utilização de ferramentas que favoreçam a eficiência computacional podem possibilitar o processamento de planos de imputação em larga escala. Além dos pontos citados no método proposto, a própria imputação composta utiliza algoritmos que naturalmente apresentam oportunidades de paralelismo, como o K-Means e o K-NN [Han et al., 2011].

A imputação composta apresentou resultados melhores do que imputação múltipla utilizando comitês de complementação de dados ausentes [Soares, 2007]. O método proposto busca obter valores ainda mais precisos através da característica de generalização natural de classificadores *ensemble*.

Na próxima seção é abordado o algoritmo Motor de Imputação, criado com o intuito de integrar a utilização de classificadores *ensemble* com a imputação composta, considerando as prerrogativas de paralelismo citadas.

IV.2 Motor de Imputação

O algoritmo Motor de Imputação tem como objetivo possibilitar que a composição e e execução de *base learners* de imputação possam ser processados de maneira eficiente, favorecendo a paralelização de tarefas computacionais. O algoritmo possui dois *pipelines* de processamento: o de imputação e o de resultado. O de imputação corresponde a coleção onde são armazenados os *base learners* de imputação; o de resultado contém o retorno da execução de cada um destes. A implementação sugerida considera planos de imputação que possuam, ou não, tarefas de *ensemble*. Dessa forma, pode ser utilizado também para o processamento de planos exclusivamente de imputação composta. Segue abaixo o ordenamento dos passos do algoritmo:

1. Primeiro são inicializados os *pipelines* de imputação e resultado.
2. Posteriormente é verificado se o plano de imputação possui a estratégia de *ensemble*.
3. Caso possua, é obtido o algoritmo implementado para a referida estratégia.
4. Caso não possua, o plano é adicionado diretamente ao *pipeline de imputação*.
5. Caso o algoritmo seja o de *Bagging*, o parâmetro T é obtido.
6. T novos conjuntos de treinamento são criados com *Bootstrapping* e associados a T cópias do plano de imputação original, formando os *base learners* de imputação.
7. Todos os T planos são adicionados ao *pipeline* de imputação.
8. A função de combinação implementada no algoritmo *Bagging* é configurada no *pipeline* de resultado.
9. O *pipeline* de imputação é executado e seus resultados (erro aferido e vetor de valores imputados) são adicionados ao *pipeline* de resultado.
10. Por fim, caso esteja configurada, é executada a etapa de combinação para a composição do vetor consolidado de valores imputados e desempenho médio do plano. Caso se trate da execução de um plano exclusivamente de imputação composta, não existe a etapa de combinação.

A pseudo-codificação do algoritmo motor de imputação é apresentada na figura IV.2.

Para a etapa de imputação composta é adotado o seguinte modelo de implementação: o plano de imputação é desmembrado, tendo suas tarefas, estratégias, algoritmos e parâmetros executados na ordem definida pelo cientista de dados; se a tarefa for de seleção, os atributos utilizados no conjunto de treinamento são restringidos ao percentual de seleção definido pelo usuário, por ordem

Figura IV.2 Algoritmo Motor de Imputação.

```
    Entrada: PI
    Saída: RESIMP
 1: PIPEIMP ← inicializaPipelineImp()
 2: PIPERES ← inicializaPipelineRes()
 3: se PI.estrategiaEnsemble() então
 4:     EE ← PI.recuperaEstrategiaEnsemble()
 5:     AE ← EE.recuperaAlgoritmo()
 6:     se AE.algoritmoBagging() então
 7:         T ← AE.recuperaParametroT()
 8:         para i ← 1 até T faça                                  ▷ Paralelismo
 9:             CT ← PI.recuperaConjuntoTreinamentoOriginal()
10:             CTBL ← AE.executaBootstrapping(CT)
11:             CTA ← SIMULAAUSENCIA(CTBL, PI.percentualAusencia())
12:             PIBL ← PI.copiarPlanoImputacao(CTA)
13:             PIPEIMP.adicionar(PIBL)
14:         fim para
15:         PIPERES.algoritmoCombinacao(AE.algoritmoCombinacao())
16:     fim se
17: senão
18:     PIPEIMP.adicionar(PI)
19: fim se
20: RESIMP ← inicializaResultadoImputacao()
21: para i ← 1 até PIPEIMP.tamanho() faça                          ▷ Paralelismo
22:     PIA ← PIPEIMP.obter(i)
23:     CTA ← PIA.recuperaConjuntoTreinamentoAusencia()
24:     RESIMPPL ← IMPUTACAOCOMPOSTA(CTA, PIA)                     ▷ Paralelismo
25:     PIPERES.adicionar(RESIMPPL)
26: fim para
27: se PIPERES.combinacao() então
28:     RESIMP ← PIPERES.executaCombinacao()
29: senão
30:     RESIMP ← PIPERES.primeiro()
31: fim se
32: retorna RESIMP
```

de relevância, de acordo com a parametrização do algoritmo de seleção utilizado; caso a tarefa seja de agrupamento, o conjunto de treinamento é dividido em subconjuntos, de acordo com os parâmetros e o algoritmo de agrupamento utilizado; quando a tarefa de agrupamento é utilizada, a complexidade do plano de imputação aumenta na grandeza $O(N^2)$, pois todas as tarefas subsequentes serão executadas para cada um dos subconjuntos criados; se a tarefa for de imputação, a entrada da mesma é um vetor de *base learners* de imputação, onde ao final da execução é gerado o vetor de valores imputados correspondente; por fim, é aferida a precisão do valor imputado, comparando-o com o valor do conjunto original (completo); o percentual de erro é associado ao plano de imputação executado.

O próximo capítulo aborda os objetivos, a metodologia, a implementação e os resultados dos experimentos realizados utilizando o método proposto.

Capítulo V Avaliação Experimental

Esta seção detalha a metodologia utilizada na condução dos experimentos realizados, bem como o *framework* Appraisal-Spark, a parametrização dos algoritmos utilizados e configurações do ambiente computacional. Por fim são apresentados os resultados obtidos.

V.1 Objetivos

Os experimentos se deram em duas bases de dados, de características distintas, com o objetivo de avaliar o método proposto neste trabalho (utilização da técnica de imputação composta associada a uma tarefa de *ensemble*). Foram criados planos de imputação com diferentes configurações quanto ao percentual de simulação de ausência e de seleção de atributos, além da variação do parâmetro T (*Bagging*). Todos os experimentos foram realizados utilizando um *framework* construído para implementação da técnica proposta. Os experimentos foram executados em modo de execução serial e paralelo, de forma que pudessem ser avaliados em ambiente de processamento de larga escala.

V.2 Metodologia

A metodologia utilizada foi a mesma proposta por Soares [2007] e Castaneda et al. [2008], que consiste em gerar ausências de dados em conjuntos completos de forma a aferir a acurácia do valor sugerido pelo método de imputação.

V.2.1 Bases de Dados

Foram utilizadas duas bases de dados numéricas nos experimentos: AIDS Deaths - National Health and Family Planning Commission of China[1] e Breast Cancer Wisconsin (Original) [2].

A base de dados AIDS Deaths - National Health and Family Planning Commission of China, referida neste trabalho como Aids, possui dados relacionados ao número de mortes por AIDS na população chinesa. Os dados foram coletados originalmente por Nan and Gao [2018], a partir do relatório mensal do Comissão Nacional de Saúde e Planejamento Familiar da China [3]. Posterior-

[1]Disponível em: `https://datadryad.org/resource/doi:10.5061/dryad.f45s8/1`
[2]Disponível em: `http://archive.ics.uci.edu/ml/index.php`
[3]Disponível em: `http://www.moh.gov.cn/`

mente, esses dados foram cruzados por com relatórios de termos pesquisados no principal buscador web da China, chamado Baidu. A base de dados consolidada encontra-se no repositório científico Dryad[4].

Utilizando a referida base de dados, Nan and Gao [2018] apresentaram um um método de aprendizado de máquina utilizando redes neurais artificiais, capaz de prever e monitorar a incidência de mortes por AIDS, dada a frequência de procura de determinados termos (relacionados ao tema) no buscador Baidu. A base Aids possui 33 atributos, sendo estes: o período (mês e ano), o número de mortes no referido período e os demais campos são os termos buscados no Baidu, com suas respectivas frequências. A base possui 78 tuplas, onde cada linha representa as frequências de um mês, começando de janeiro de 2011 até junho de 2017. Seguem abaixo os atributos da base Aids com seus respectivos identificadores:

1. Time
2. AIDS Death,
3. Regulation on the Prevention and Treatment of AIDS,
4. AIDS prevention knowledge,
5. AIDS awareness,
6. Handwritten AIDS newspaper,
7. Handwritten anti-AIDS newspaper,
8. AIDS virus,
9. How to prevent AIDS,
10. Route of transmission of AIDS,
11. AIDS prevention,
12. What is AIDS,
13. Which day is World AIDS Day,
14. The origins of the AIDS,
15. The origins of the AIDS,
16. AIDS,

[4]Disponível em: `https://datadryad.org/`

17. AIDS Day,
18. The origins of AIDS,
19. How to prevent AIDS,
20. AIDS awareness day,
21. World AIDS Day,
22. How to prevent AIDS,
23. AIDS awareness slogan,
24. AIDS/HIV,
25. Initial symptoms of AIDS,
26. Images of AIDS skin rashes,
27. How long will one survive once he/she contracts HIV,
28. How long can AIDS patients survive,
29. HIV/AIDS prevention,
30. AIDS village in Henan province,
31. How does one contract HIV,
32. Number of AIDS patients in China,
33. Symptoms of AIDS in incubation period

A base de dados Breast Cancer consolida dados do hospital de Wiscosin sobre o diagnóstico de câncer de mama, relativos a pacientes que tiveram a doença nos meses de janeiro e outubro de 1989; fevereiro, abril e agosto de 1990; e janeiro, junho e novembro de 1991. A base possui 699 tuplas e 11 atributos, sendo eles o código identificador do paciente, o atributo classificador e os demais campos são características da doença. Seguem abaixo os atributos da referida base com seus respectivos identificadores:

1. Code number
2. Clump thickness
3. Uniformity of cell size

4. Uniformity of cell shape
5. Marginal adhesion
6. Single epithelial cell size
7. Bare nuclei
8. Bland chromatin
9. Normal nucleoli
10. Mitoses
11. Class

Os dados utilizados foram os originais dos dois *data sets*, sem nenhum processo de normalização dos mesmos. A base Breast Cancer possuía 16 tuplas com registros incompletos, removidos com *listwise deletion*, totalizando 683 tuplas utilizadas nesta base. A base Aids não possuía registros incompletos.

A escolha desses dois *datasets* teve como objetivo avaliar a técnica proposta em uma base de dados referência para *benchmark* em experimentos de aprendizado de máquina, a base Breast Cancer, utilizada nos trabalhos de Nanni et al. [2012], Castaneda et al. [2008], Soares [2007], entre outros, em comparação com a base Aids, utilizada recentemente nos experimentos de Nan and Gao [2018].

A correlação entre os atributos das bases de dados é de suma importância para o processo de imputação e tendem a favorecer medições baseadas em análises de similaridade [Soares, 2007]. As tabelas V.1, V.2, V.3 e V.4 representam as matrizes de correlação (*pearson*) dos *datasets* Aids e Breast Cancer. Dada a quantidade de atributos, a matriz da base Aids foi dividida em três tabelas para facilitar a visualização.

Tabela V.1: Aids: matriz de correlação, parte 1.

	3	4	5	6	7	8	9	10	11
3	1	0,79	0,732	0,723	0,735	0,777	0,697	0,708	0,715
4	0,79	1	0,877	0,846	0,851	0,657	0,888	0,744	0,906
5	0,732	0,877	1	0,904	0,876	0,656	0,914	0,689	0,906
6	0,723	0,846	0,904	1	0,982	0,744	0,885	0,688	0,937
7	0,735	0,851	0,876	0,982	1	0,756	0,866	0,682	0,92
8	0,777	0,657	0,656	0,744	0,756	1	0,677	0,803	0,705
9	0,697	0,888	0,914	0,885	0,866	0,677	1	0,812	0,959
10	0,708	0,744	0,689	0,688	0,682	0,803	0,812	1	0,751
11	0,715	0,906	0,906	0,937	0,92	0,705	0,959	0,751	1
12	0,728	0,833	0,854	0,829	0,807	0,782	0,937	0,852	0,913
13	0,708	0,779	0,771	0,788	0,815	0,7	0,745	0,648	0,763
14	0,641	0,735	0,614	0,647	0,675	0,619	0,663	0,652	0,635
15	0,667	0,517	0,473	0,612	0,626	0,878	0,554	0,746	0,578
16	0,662	0,587	0,612	0,676	0,659	0,887	0,721	0,852	0,702
17	0,602	0,815	0,858	0,8	0,772	0,622	0,924	0,754	0,902
18	0,674	0,611	0,66	0,733	0,733	0,867	0,721	0,767	0,732
19	0,617	0,838	0,893	0,807	0,789	0,582	0,936	0,661	0,907
20	0,703	0,84	0,903	0,813	0,802	0,526	0,851	0,563	0,845
21	0,576	0,784	0,825	0,733	0,726	0,517	0,853	0,664	0,811
22	0,599	0,792	0,84	0,763	0,746	0,572	0,925	0,689	0,872
23	0,538	0,773	0,894	0,75	0,727	0,408	0,803	0,504	0,767
24	0,473	0,673	0,677	0,663	0,692	0,581	0,739	0,627	0,689
25	0,568	0,408	0,342	0,561	0,595	0,823	0,427	0,623	0,494
26	0,583	0,308	0,246	0,434	0,494	0,666	0,252	0,388	0,326
27	0,519	0,306	0,226	0,436	0,498	0,702	0,257	0,458	0,343
28	0,463	0,11	0,169	0,304	0,322	0,64	0,204	0,431	0,198
29	0,426	0,713	0,769	0,555	0,503	0,231	0,763	0,497	0,672
30	0,451	0,585	0,563	0,404	0,378	0,441	0,595	0,562	0,53
31	0,51	0,384	0,444	0,515	0,529	0,738	0,502	0,612	0,48
32	0,526	0,645	0,553	0,372	0,404	0,315	0,607	0,596	0,51
33	0,596	0,523	0,475	0,543	0,541	0,787	0,633	0,753	0,603

Tabela V.2: Aids: matriz de correlação, parte 2.

	12	13	14	15	16	17	18	19	20
3	0,728	0,708	0,641	0,667	0,662	0,602	0,674	0,617	0,703
4	0,833	0,779	0,735	0,517	0,587	0,815	0,611	0,838	0,84
5	0,854	0,771	0,614	0,473	0,612	0,858	0,66	0,893	0,903
6	0,829	0,788	0,647	0,612	0,676	0,8	0,733	0,807	0,813
7	0,807	0,815	0,675	0,626	0,659	0,772	0,733	0,789	0,802
8	0,782	0,7	0,619	0,878	0,887	0,622	0,867	0,582	0,526
9	0,937	0,745	0,663	0,554	0,721	0,924	0,721	0,936	0,851
10	0,852	0,648	0,652	0,746	0,852	0,754	0,767	0,661	0,563
11	0,913	0,763	0,635	0,578	0,702	0,902	0,732	0,907	0,845
12	1	0,746	0,622	0,677	0,856	0,935	0,834	0,893	0,768
13	0,746	1	0,667	0,514	0,594	0,719	0,597	0,708	0,78
14	0,622	0,667	1	0,544	0,531	0,585	0,42	0,588	0,593
15	0,677	0,514	0,544	1	0,879	0,53	0,876	0,41	0,312
16	0,856	0,594	0,531	0,879	1	0,719	0,917	0,618	0,474
17	0,935	0,719	0,585	0,53	0,719	1	0,701	0,9	0,782
18	0,834	0,597	0,42	0,876	0,917	0,701	1	0,635	0,51
19	0,893	0,708	0,588	0,41	0,618	0,9	0,635	1	0,869
20	0,768	0,78	0,593	0,312	0,474	0,782	0,51	0,869	1
21	0,834	0,722	0,58	0,405	0,565	0,892	0,564	0,856	0,782
22	0,894	0,633	0,551	0,471	0,653	0,883	0,683	0,946	0,797
23	0,668	0,612	0,497	0,178	0,358	0,7	0,417	0,821	0,845
24	0,688	0,509	0,546	0,505	0,582	0,626	0,649	0,718	0,521
25	0,553	0,485	0,44	0,864	0,768	0,405	0,763	0,272	0,179
26	0,384	0,538	0,427	0,711	0,577	0,264	0,58	0,165	0,223
27	0,386	0,458	0,411	0,792	0,636	0,273	0,654	0,142	0,127
28	0,378	0,302	0,275	0,738	0,698	0,227	0,641	0,099	0,06
29	0,65	0,483	0,445	0,031	0,284	0,715	0,266	0,813	0,792
30	0,576	0,472	0,441	0,252	0,431	0,562	0,379	0,592	0,533
31	0,649	0,56	0,422	0,756	0,799	0,541	0,759	0,451	0,336
32	0,587	0,491	0,48	0,272	0,374	0,59	0,36	0,595	0,559
33	0,766	0,478	0,49	0,797	0,874	0,615	0,807	0,521	0,394

Tabela V.3: Aids: matriz de correlação, parte 3.

	21	22	23	24	25	26	27	28	29	30	31	32	33
3	0,576	0,599	0,538	0,473	0,568	0,583	0,519	0,463	0,426	0,451	0,51	0,526	0,596
4	0,784	0,792	0,773	0,673	0,408	0,308	0,306	0,11	0,713	0,585	0,384	0,645	0,523
5	0,825	0,84	0,894	0,677	0,342	0,246	0,226	0,169	0,769	0,563	0,444	0,553	0,475
6	0,733	0,763	0,75	0,663	0,561	0,434	0,436	0,304	0,555	0,404	0,515	0,372	0,543
7	0,726	0,746	0,727	0,692	0,595	0,494	0,498	0,322	0,503	0,378	0,529	0,404	0,541
8	0,517	0,572	0,408	0,581	0,823	0,666	0,702	0,64	0,231	0,441	0,738	0,315	0,787
9	0,853	0,925	0,803	0,739	0,427	0,252	0,257	0,204	0,763	0,595	0,502	0,607	0,633
10	0,664	0,689	0,504	0,627	0,623	0,388	0,458	0,431	0,497	0,562	0,612	0,596	0,753
11	0,811	0,872	0,767	0,689	0,494	0,326	0,343	0,198	0,672	0,53	0,48	0,51	0,603
12	0,834	0,894	0,668	0,688	0,553	0,384	0,386	0,378	0,65	0,576	0,649	0,587	0,766
13	0,722	0,633	0,612	0,509	0,485	0,538	0,458	0,302	0,483	0,472	0,56	0,491	0,478
14	0,58	0,551	0,497	0,546	0,44	0,427	0,411	0,275	0,445	0,441	0,422	0,48	0,49
15	0,405	0,471	0,178	0,505	0,864	0,711	0,792	0,738	0,031	0,252	0,756	0,272	0,797
16	0,565	0,653	0,358	0,582	0,768	0,577	0,636	0,698	0,284	0,431	0,799	0,374	0,874
17	0,892	0,883	0,7	0,626	0,405	0,264	0,273	0,227	0,715	0,562	0,541	0,59	0,615
18	0,564	0,683	0,417	0,649	0,763	0,58	0,654	0,641	0,266	0,379	0,759	0,36	0,807
19	0,856	0,946	0,821	0,718	0,272	0,165	0,142	0,099	0,813	0,592	0,451	0,595	0,521
20	0,782	0,797	0,845	0,521	0,179	0,223	0,127	0,06	0,792	0,533	0,336	0,559	0,394
21	1	0,806	0,703	0,558	0,272	0,245	0,201	0,124	0,74	0,511	0,432	0,689	0,448
22	0,806	1	0,736	0,789	0,319	0,151	0,148	0,134	0,754	0,567	0,506	0,598	0,608
23	0,703	0,736	1	0,596	0,058	-0,012	-0,026	-0,06	0,82	0,545	0,168	0,488	0,223
24	0,558	0,789	0,596	1	0,433	0,125	0,228	0,142	0,481	0,492	0,556	0,474	0,562
25	0,272	0,319	0,058	0,433	1	0,757	0,802	0,71	-0,182	0,164	0,713	0,072	0,761
26	0,245	0,151	-0,012	0,125	0,757	1	0,894	0,794	-0,218	-0,048	0,622	0,124	0,49
27	0,201	0,148	-0,026	0,228	0,802	0,894	1	0,771	-0,278	-0,021	0,618	0,108	0,522
28	0,124	0,134	-0,06	0,142	0,71	0,794	0,771	1	-0,249	-0,021	0,693	0,014	0,567
29	0,74	0,754	0,82	0,481	-0,182	-0,218	-0,278	-0,249	1	0,604	0,062	0,648	0,191
30	0,511	0,567	0,545	0,492	0,164	-0,048	-0,021	-0,021	0,604	1	0,223	0,45	0,433
31	0,432	0,506	0,168	0,556	0,713	0,622	0,618	0,693	0,062	0,223	1	0,273	0,732
32	0,689	0,598	0,488	0,474	0,072	0,124	0,108	0,014	0,648	0,45	0,273	1	0,382
33	0,448	0,608	0,223	0,562	0,761	0,49	0,522	0,567	0,191	0,433	0,732	0,382	1

Tabela V.4: Breast Cancer: matriz de correlação.

	2	3	4	5	6	7	8	9	10
2	1	0,642	0,653	0,488	0,524	0,593	0,554	0,534	0,351
3	0,642	1	0,907	0,707	0,754	0,692	0,756	0,719	0,461
4	0,653	0,907	1	0,686	0,722	0,714	0,735	0,718	0,441
5	0,488	0,707	0,686	1	0,595	0,671	0,669	0,603	0,419
6	0,524	0,754	0,722	0,595	1	0,586	0,618	0,629	0,481
7	0,593	0,692	0,714	0,671	0,586	1	0,681	0,584	0,339
8	0,554	0,756	0,735	0,669	0,618	0,681	1	0,666	0,346
9	0,534	0,719	0,718	0,603	0,629	0,584	0,666	1	0,434
10	0,351	0,461	0,441	0,419	0,481	0,339	0,346	0,434	1

Com o objetivo de analisar o nível de correlação dos atributos das bases de dados, foram verificadas as quantidades de correlações maiores que 50%, de modo a aferir se os *datasets* apresentam alta ou baixa correlação. A tabela V.5 detalha comparativamente os resultados.

V.2.2 Ausências de Dados e Seleção de Atributos

No que tange à simulação da ausência, adotou-se neste trabalho uma abordagem em largura [Soares, 2007]: foram geradas ausências em todos os atributos de todas as bases, à exceção dos atributos identificadores e classificadores. Soares [2007] propõe a utilização dos percentuais de ausência 10%, 20%, 30%, 40% e 50%. Entretanto, para cada base e atributo, optou-se por gerar ausências de dados nas proporções de 10%, 20% e 30%, dado o número elevado de simulações caso fosse considerada a proposta original; também pelo fato de o fenômeno que se deseja observar, variação do parâmetro T, apresentar resultados interessantes na configuração proposta. As ausências foram geradas seguindo o mecanismo MCAR.

Em relação à seleção de atributos, o objetivo foi verificar se a tarefa de seleção com o algoritmo PCA contribui para a geração de valores imputados com maior precisão. Foram utilizadas as proporções de 10%, 20% e 30% de redução na quantidade de atributos selecionados, com base na matriz de covariância gerada pelo algoritmo PCA [Soares, 2007]. Isso significa que foram utilizados, respectivamente, 90%, 80% e 70% dos atributos mais relevantes das bases de dados. Como a redução é percentual e não representa valores inteiros (quantidade de atributos em cada base), foi utilizado sempre a porção inteira inferior (piso) do valor sugerido pela tarefa de seleção. Por exemplo, 10% de redução no número de atributos da base Aids significa utilizar 29,7 atributos, logo neste caso 29 atributos foram utilizados.

V.3 Appraisal-Spark

O Appraisal-Spark é um *framework* desenvolvido para Apache Spark que tem como objetivo geral a criação, a execução e o gerenciamento de planos de imputação em larga escala. Inspirado no sistema Appraisal desenvolvido por Soares [2007], o Appraisal-Spark é uma implementação totalmente nova, na linguagem Scala, sendo a materialização da técnica proposta neste trabalho: a

Tabela V.5: Análise comparativa do número de correlações maiores que 50% nas bases Aids e Breast Cancer.

Base de dados	Nºde atributos	Nºde correlações	Nºde correlações $> 50\%$	Correlação
Aids	31	961	236 (25%)	Baixa
Breast Cancer	9	81	63 (78%)	Alta

utilização da imputação composta, combinada com a tarefa de *ensemble* (Bagging), para a geração de valores imputados com maior acurácia.

A linguagem Scala foi a escolhida para a implementação do *framework*, uma vez que o próprio Apache Spark é desenvolvido nativamente em Scala, fazendo com que códigos nesta linguagem apresentem maior desempenho[5]. Além disso, apesar de suportar Python, Java e R, novas implementações do Apache Spark são disponibilizadas primeiramente em Scala.

V.3.1 Arquitetura

A arquitetura do Appraisal-Spark tem como base os principais módulos do sistema Appraisal original:

1. O módulo *Crowner*, que executa os planos de imputação;
2. O módulo *Reviewer*, que verifica a qualidade das sugestões de imputação produzidas pelo módulo *Crowner*;
3. O módulo *Eraser*, que simula valores ausentes em base de dados, seguindo o mecanismo e percentual de ausência definidos pelo usuário.

A figura V.1 ilustra a interação entre os módulos do sistema:

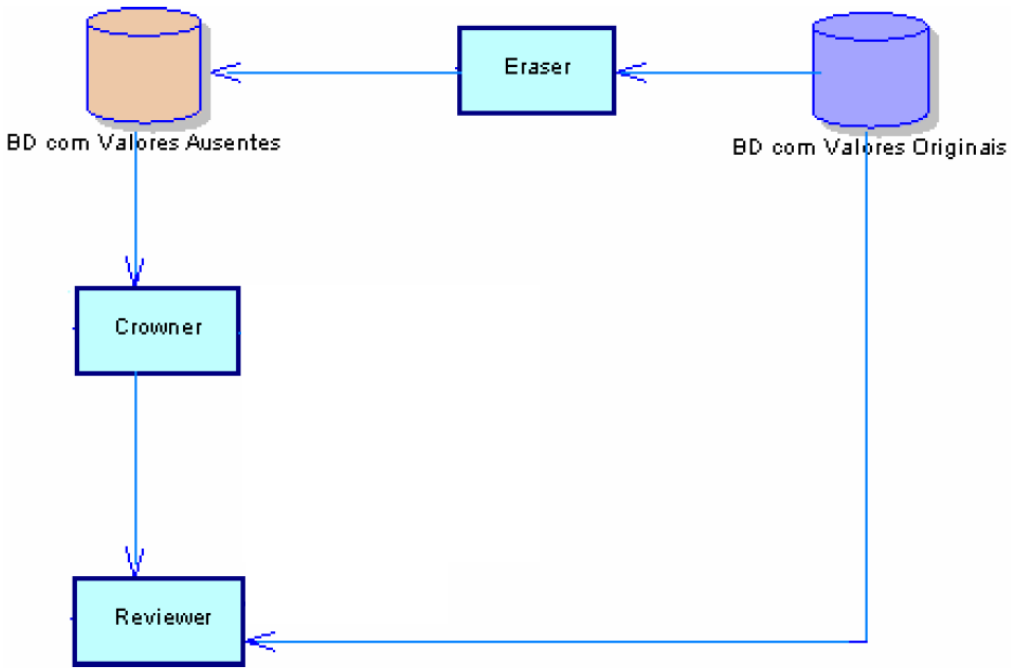

Figura V.1: Módulos do sistema Appraisal. Fonte: adaptado de Soares [2007].

Soares [2007] propôs um quarto módulo chamado *Committee*, o qual criava comitês de complementação de dados ausentes, combinando resultados de diferentes planos de imputação. Entretanto,

[5]Disponível em: `https://www.dezyre.com/article/scala-vs-python-for-apache-spark/213`

a utilização deste módulo não aumentou a precisão dos valores imputados. Além disso, na estratégia adotada, são combinadas execuções de um mesmo plano de imputação e não de planos de imputação diferentes. Por esses motivos optou-se por não utilizar o referido módulo.

No Appraisal-Spark, além das tarefas de seleção, agrupamento e imputação, originárias da imputação composta, foi criada a tarefa de *ensemble*. A tarefa de *ensemble* compõe uma estratégia, que por sua vez é associada ao algoritmo *Bagging*. Se utilizada, a tarefa de *ensemble* deve ser obrigatoriamente a primeira tarefa do plano de imputação. Isso ocorre pois é a partir dessa tarefa que se originam os *base learners* de imputação, conforme explicado na seção IV.1. O *core* do Appraisal-Spark implementa o algoritmo motor de imputação, detalhado na seção IV.2, para a execução de tarefas de *ensemble* e imputação composta em ambiente Apache Spark.

Seguindo o conceito de proveniência retrospectiva, todos os planos de imputação, bem como suas estratégias, algoritmos e parâmetros são detalhados no *log* de execução do *framework*. São controlados o tempo total de execução (janela) e o tempo por plano de imputação. Como instrumentos de proveniência prospectiva, é possível verificar no *log* as próximas tarefas a serem executadas, dentro de um mesmo plano de imputação, bem como os próximos planos a serem executados. É possível acompanhar a quantidade de planos de imputação finalizados e o percentual de completude geral dos experimentos. Ao fim do experimento, são registrados no *log* os resultados de todos os planos de imputação realizados. Por fim, o plano com menor erro absoluto é sugerido para imputação na referida base de dados.

Tais funcionalidades do Appraisal-Spark, no sentido de permitir a composição de diversos planos de imputação, acompanhar suas execuções e gerenciar a proveniência retrospectiva e prospectiva, remetem a características de *workflows* científicos.

V.3.2 Funcionalidades

No que tange ao modo de execução, o Appraisal-Spark foi construído no sentido de utilizar toda a capacidade de paralelismo do Apache Spark. Entretanto, com o objetivo de possibilitar a comparação de desempenho entre planos de imputação, com e sem paralelismo, o modo de execução pode ser configurado. O *framework* possui um parâmetro chamado *single*, informado na instrução de execução do experimento. Se o parâmetro for informado, o experimento é executado em modo serial, se não for informado, é executado em paralelo.

Na modalidade serial, os planos de imputação, bem como suas tarefas e algoritmos, são executados na ordem em que foram programadas pelo cientista de dados. O modo paralelo tem como base os conceitos *map-reduce* e *parallelized collections* [Zaharia et al., 2016c], de forma que tantos os planos de imputação, como suas tarefas e algoritmos são executadas em paralelo. O Apache Spark otimiza esse processo, sem comprometer o ordenamento das tarefas do plano de imputação. Buscou-

se, sempre que possível, implementar as instruções em Spark SQL, uma vez que estas apresentam ganho de desempenho em comparação com a utilização tradicional de RDDs [Armbrust et al., 2015].

O *framework* possui nativamente as seguintes funcionalidades:

- Eraser: possibilita a simulação de ausências em bases de dados seguindo o mecanismo MCAR. Tem como entrada o atributo onde serão geradas as ausências e o percentual de ausência desejado.
- Composição de Planos de Imputação: permite a composição de planos de imputação por meio das interfaces *ImputationPlan*, *Strategy*, *Ensemble*, *Selection*, *Clustering* e *Imputation*. Essas interfaces permitem a criação de planos seguindo o modelo de imputação composta.
- Classificadores *ensemble*: utiliza o algoritmo *Bagging* para a geração de *base learners* de imputação e posterior combinação, para a geração dos valores imputados.
- Seleção de atributos: implementa o algoritmo PCA para identificar os atributos de maior correlação em um conjunto de dados.
- Agrupamento: utiliza os algoritmos K-Means e K-MeansPlus, para identificar grupos em conjuntos de dados.
- Imputação: os algoritmos AVG e K-NN são utilizados para a composição de valores imputados.

Os algoritmos implementados serão abordados com mais detalhes na próxima seção.

V.3.3 Algoritmos

Alguns algoritmos do Appraisal-Spark utilizam como base a biblioteca MLlib [Meng et al., 2016b]; outros foram desenvolvidos integralmente no *framework*. Entretanto, todos os algoritmos implementados seguem a arquitetura detalhadas na figura V.2. Na linguagem Scala, interfaces são chamadas *traits*. As classes, por sua vez, estendem *traits*. Para facilitar o entendimento, *traits* serão referidos como interfaces no texto.

Os algoritmos Bagging, PCA, K-Means, K-MeansPlus, AVG e K-NN foram implementados da seguinte forma:

- O algoritmo Bagging implementa a interface EnsembleAlgorithm, recebendo como entrada o conjunto de treinamento e a *hash* de parâmetros de configuração. O método *run* retorna a interface EnsembleResult.
- O algoritmo PCA implementa a interface SelectionAlgorithm, recebendo como entrada o conjunto de treinamento e a *hash* de parâmetros de configuração. O método *run* retorna a

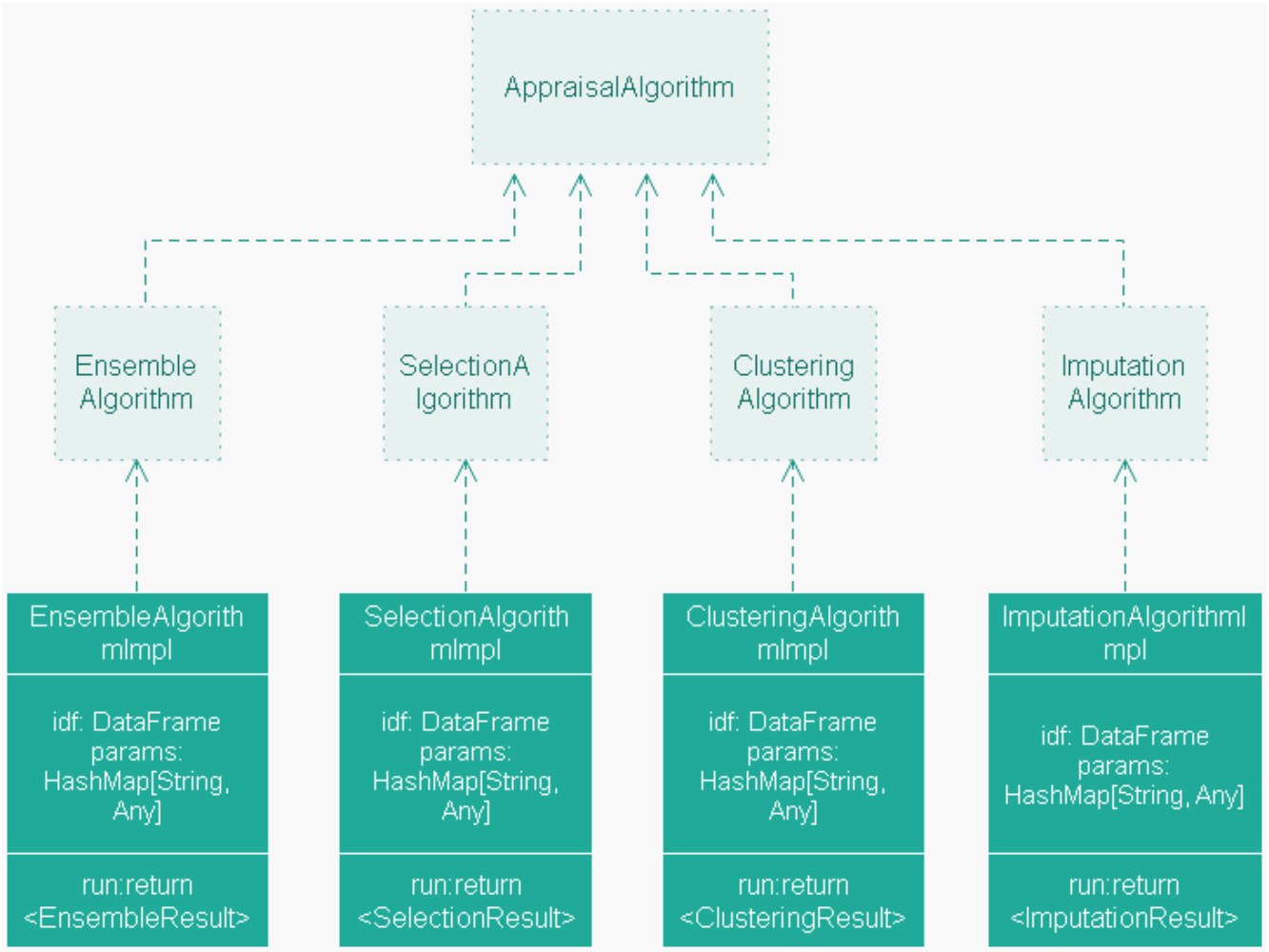

Figura V.2: Diagrama de classes: interfaces do *framework* Appraisal-Spark.

interface SelectionResult. Utiliza a classe *org.apache.spark.mllib.linalg.distributed.RowMatrix* da biblioteca MLlib, para computar matrizes de covariância.

- O algoritmo K-Means implementa a interface AgrupamentoAlgorithm, recebendo como entrada o conjunto de treinamento e a *hash* de parâmetros de configuração. O método *run* retorna a interface ClusteringResult. Utiliza a classe *org.apache.spark.mllib.clustering.KMeans* da biblioteca MLlib, para identificar grupos com base no parâmetro k informado.

- Além da versão tradicional do algoritmo, foi implementada uma versão chamada K-MeansPlus, a qual adota o procedimento descrito na documentação da API [Meng et al., 2016a], que consiste em verificar o valor do atributo WSSSE (*Within Set Sum of Squared Error*) a cada iteração do K-Means. A documentação sugere que, geralmente, o número ótimo k de grupos é obtido na iteração anterior a qual o valor de WSSSE aumenta. A versão KMeansPlus foi a utilizada na execução dos experimentos.

- O algoritmo AVG implementa a interface ImputationAlgorithm, recebendo como entrada o conjunto de treinamento e a *hash* de parâmetros de configuração. O método *run* retorna a

interface ImputationResult.

- O algoritmo K-NN implementa a interface ImputationAlgorithm, recebendo como entrada o conjunto de treinamento e a *hash* de parâmetros de configuração. O método *run* retorna a interface ImputationResult.

Os algoritmos implementados no Appraisal-Spark não necessitam que suas execuções ocorram dentro de um contexto de imputação, ou seja, obrigatoriamente atreladas a um plano de imputação. Todos os algoritmos desenvolvidos podem ser executados de maneira independente. Além disso, o *framework* foi desenvolvido com o intuito de suportar implementações personalizadas. Logo, qualquer algoritmo que estenda as interfaces descritas na figura V.2, pode ser utilizado na composição de planos de imputação.

V.3.4 Estrutura de Pacotes

O Appraisal-Spark é organizado de acordo com a seguinte estrutura de pacotes:

- *appraisal.spark.interfaces* contém as interfaces utilizadas ao longo do *framework*. É nesse pacote que estão os elementos para composição de planos de imputação.
- *appraisal.spark.entities* contém entidades utilizadas ao longo das implementações, inclusive tipos imutáveis, como os retornos dos métodos *run* das interfaces do sistema.
- *appraisal.spark.strategies* contém as estratégias utilizadas na composição dos planos de imputação.
- *appraisal.spark.engine* contém as classes responsáveis pelo *core* do AppraisalSpark, o motor de imputação, o qual executa e gerencia planos de imputação.
- *appraisal.spark.algorithm* contém as implementações dos algoritmos do *framework*.
- *appraisal.spark.statistic* contém a classe com métodos estatísticos para aferir a precisão dos valores imputados.
- *appraisal.spark.util* contém a classe com implementações utilitárias reutilizadas ao longo do *framework*.

Juntamente com o Appraisal-Spark, foi desenvolvido um segundo projeto chamado Appraisal-Spark-Executor. Este projeto contém exemplos de como utilizar o Appraisal-Spark como uma biblioteca. Além disso, possui classes com demonstrações de como compor planos de imputação e utilizar as funcionalidades do *framework*. Os experimentos deste trabalho estão implementados no projeto.

V.4 Parametrização de Algoritmos

Adotou-se a proposta de ampla variação do parâmetro k para os algoritmos K-Means e K-NN Soares [2007]. Dessa forma, considerando o número de tuplas em cada base de dados, o parâmetro k teve como variação máxima os valores descritos na tabela V.6.

Tabela V.6: Variação do parâmetro k nos algoritmos K-Means e K-NN.

Base de dados	Algoritmo	Percentual de ausência		
		10	20	30
Aids	k-NN	1-70	1-62	1-54
	k-Means	2-70	2-62	2-54
Breast Cancer	k-NN	1-614	1-546	1-478
	k-Means	2-614	2-546	2-478

V.5 Infraestrutura e Configuração

Os experimentos foram realizados em um *cluster* Apache Spark composto por quatro máquinas com a seguinte configuração:

- Processador: AMD PRO A10-67508 R7, 3.60 GHz.
- Quantidade de *cores*: 4, sendo todos utilizados pelo *cluster* Spark.
- Memória (RAM): 8,00 GB, sendo 5,00 GB utilizados pelo *cluster* Spark.
- No total, o *cluster* Spark dispunha de 16 *cores* e 20,00 GB de memória (RAM).

Os experimentos realizados em modo serial foram executados em único *core* com 5,00 GB de memória (RAM). Os experimentos com paralelismo foram executados em 1 *core*, com 1,00 GB de memória (RAM) alocada ao *driver memory*, e quatro *executors* com 1 GB de memória (RAM) cada, totalizando os mesmos 5,00 GB de memória (RAM) utilizados no experimento serial.

V.5.1 Experimentos Realizados

Seguindo a estratégia de experimentação proposta por Soares [2007], para cada base de dados, atributo, percentual de ausência e percentual de seleção, foram executados os seguintes planos de imputação:

1. Seleção→Agrupamento→Imputação[Avg]: SAI[Avg]
2. Seleção→Agrupamento→Imputação[Knn]: SAI[Knn]
3. Agrupamento→Seleção→Imputação[Knn]: ASI[Knn]
4. Agrupamento→Imputação[Avg]: AI[Avg]
5. Agrupamento→Imputação[Knn]: AI[Knn]
6. Seleção→Imputação[Knn]: SI[Knn]
7. Imputação[Avg]: I[Avg]
8. Imputação[Knn]: I[Knn]

Considerando os experimentos com *ensemble*, foi utilizada a variação do parâmetro T de um a três, de forma a promover a geração de no mínimo um e no máximo três *base learners* de imputação. O propósito dessa estratégia é verificar se a afirmação de Zhou [2012], de que o erro com *Bagging* diminui conforme o número de *base learners* aumenta, também se repete para o método proposto.

V.6 Resultados

Foram realizados oito experimentos, cujas técnicas, modos de execução, quantidade de planos de imputação e tempos de processamento são detalhados na tabela V.7.

Tabela V.7: Número de planos de imputação executados e tempo gasto por experimento.

Base de dados	Técnica	Modo de execução	Quantidade de PIs	Tempo (h)
Aids	IC	Serial	2232	50
		Paralelo		24
Breast Cancer	IC	Serial	648	22
		Paralelo		12
Aids	IC + Ensemble	Serial	6696	260
		Paralelo		119
Breast Cancer	IC + Ensemble	Serial	1944	115
		Paralelo		69
Total			23040	671

V.6.1 Modo de Execução (Serial x Paralelo)

Em relação ao modo de execução, todos os experimentos obtiveram ganho de desempenho com paralelismo, conforme informado abaixo:

- O ganho de desempenho utilizando paralelismo foi de 51,88% na base de dados Aids, para o experimento apenas com imputação composta.
- O ganho de desempenho com paralelismo foi de 45,20% na base de dados Breast Cancer, para o experimento apenas com imputação composta.
- O ganho de desempenho utilizando paralelismo foi de 54,09% na base de dados Aids, para o experimento com imputação composta e *ensemble*.
- O ganho de desempenho com paralelismo foi de 39,59% na base de dados Breast Cancer, para o experimento com imputação composta e *ensemble*.

A figura V.3 ilustra o ganho de desempenho com o modo de execução paralela e distribuída.

V.6.2 Resultados por Experimento

Nesta seção serão analisados graficamente os resultados para cada uma das bases de dados utilizadas e seus respectivos experimentos. Os resultados dos experimentos estão consolidados de acordo com o erro médio do plano de imputação, ou seja, a média de erro para imputações realizadas em

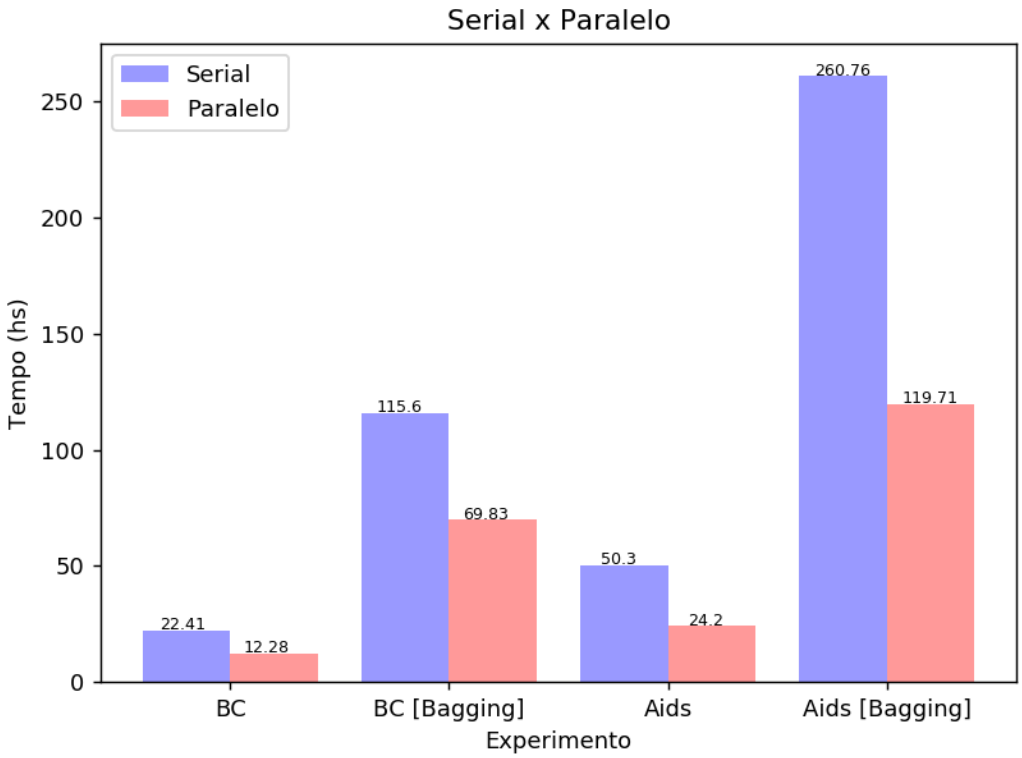

Figura V.3: Comparação dos experimentos quanto ao modo de execução: serial x paralelo.

todos atributos das bases de dados, em cada um dos percentuais de ausência, seleção e quantidade de *base learners* de imputação T (no caso dos experimentos com *ensemble*). A priori, serão analisados os resultados somente com imputação composta e posteriormente com imputação composta e *ensemble*.

Foi possível observar que o plano de imputação que utiliza apenas a média simples apresentou erro elevado em todos os experimentos realizados. Tal fato confirma a afirmação de Soares [2007] e Tavares et al. [2018], de que esta não é uma boa estratégia de imputação. Dessa forma, este plano teve sua visualização restrita a um limite, ajustado para facilitar a visualização gráfica. Entretanto, os valores absolutos dos erros de todos os planos de imputação são apresentados. Os experimentos foram numerados para facilitar a organização e entendimento.

Imputação Composta - Base de Dados AIDS

1) No experimento com 10% de ausência e 10% de redução na seleção de atributos, gráfico V.4, o plano de imputação que obteve o melhor resultado foi: AI[Knn]. É possível observar que a utilização da tarefa de seleção afetou pouco o erro gerado para o plano SI[Knn], em comparação com o plano I[Knn]. É possível observar também que a tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

2) No experimento com 10% de ausência e 20% de redução na seleção de atributos, gráfico V.5,

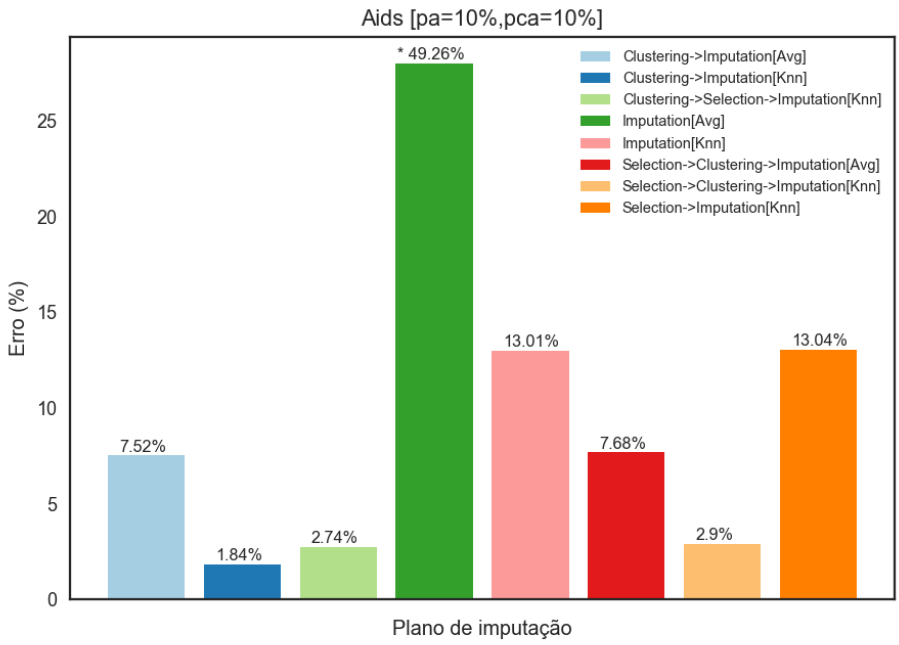

Figura V.4: Experimento com 10% de ausência de dados e redução de 10% na etapa de seleção de atributos.

o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

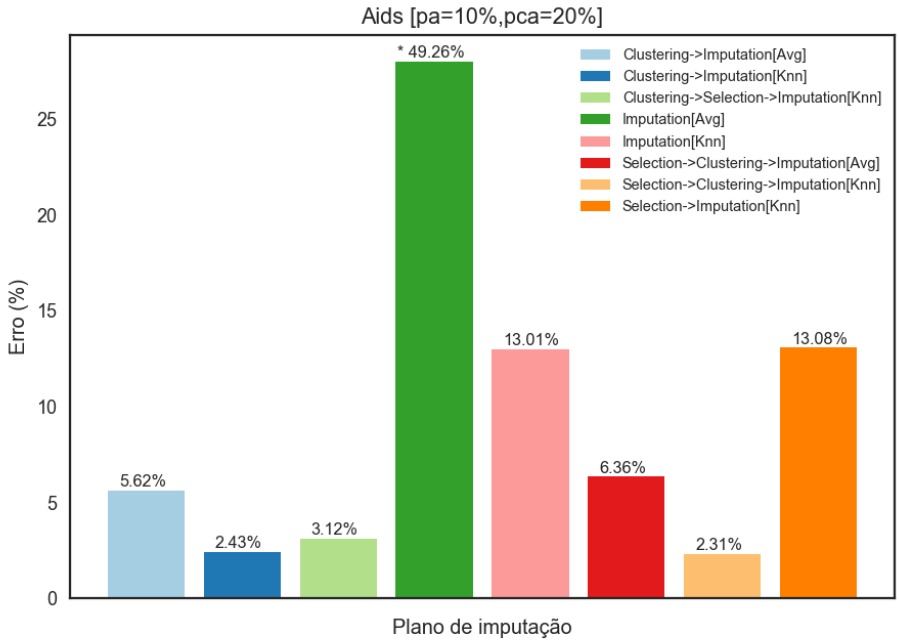

Figura V.5: Experimento com 10% de ausência de dados e redução de 20% na etapa de seleção de atributos.

3) No experimento com 10% de ausência e 30% de redução na seleção de atributos, gráfico V.6,

o plano de imputação que obteve o melhor resultado foi: ASI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

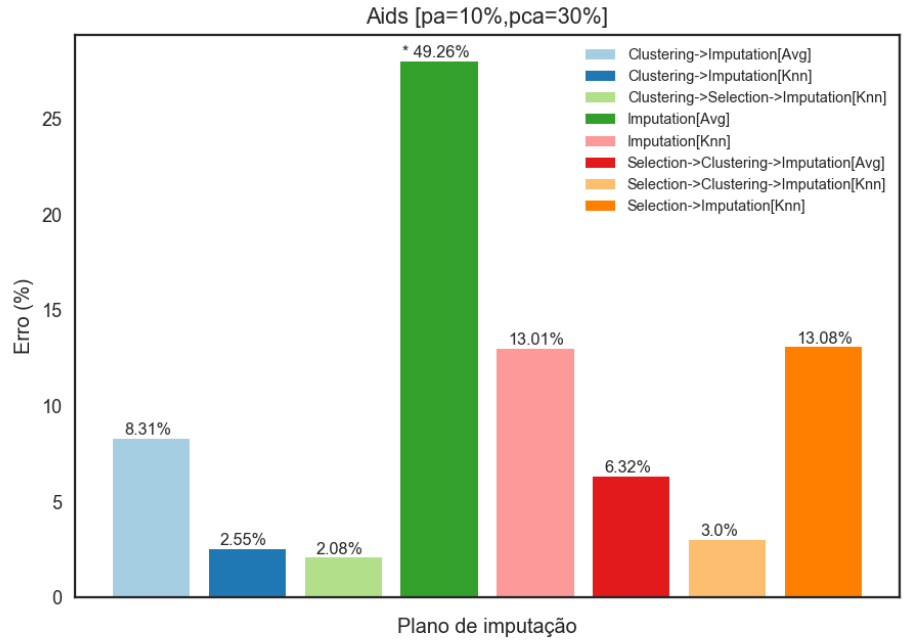

Figura V.6: Experimento com 10% de ausência de dados e redução de 30% na etapa de seleção de atributos.

4) No experimento com 20% de ausência e 10% de redução na seleção de atributos, gráfico V.7, o plano de imputação que obteve o melhor resultado foi: AI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

5) No experimento com 20% de ausência e 20% de redução na seleção de atributos, gráfico V.8, o plano de imputação que obteve o melhor resultado foi novamente: AI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

6) No experimento com 20% de ausência e 30% de redução na seleção de atributos, gráfico V.9, o plano de imputação que obteve o melhor resultado foi: ASI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

7) No experimento com 30% de ausência e 10% de redução na seleção de atributos, gráfico V.10, o plano de imputação que obteve o melhor resultado foi novamente: ASI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

8) No experimento com 30% de ausência e 20% de redução na seleção de atributos, gráfico V.11, o plano de imputação que obteve o melhor resultado foi novamente: ASI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo

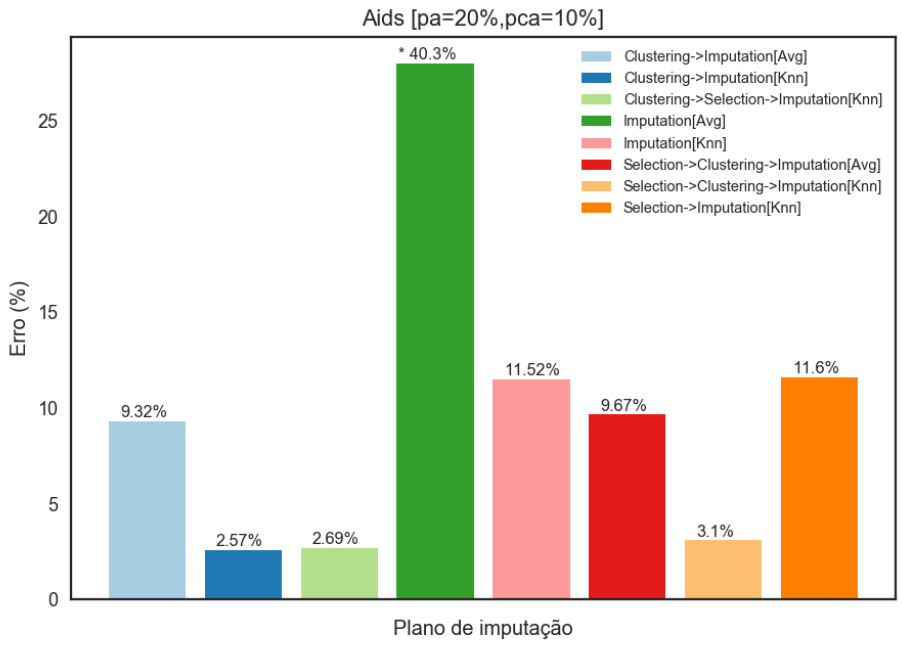

Figura V.7: Experimento com 20% de ausência de dados e redução de 10% na etapa de seleção de atributos.

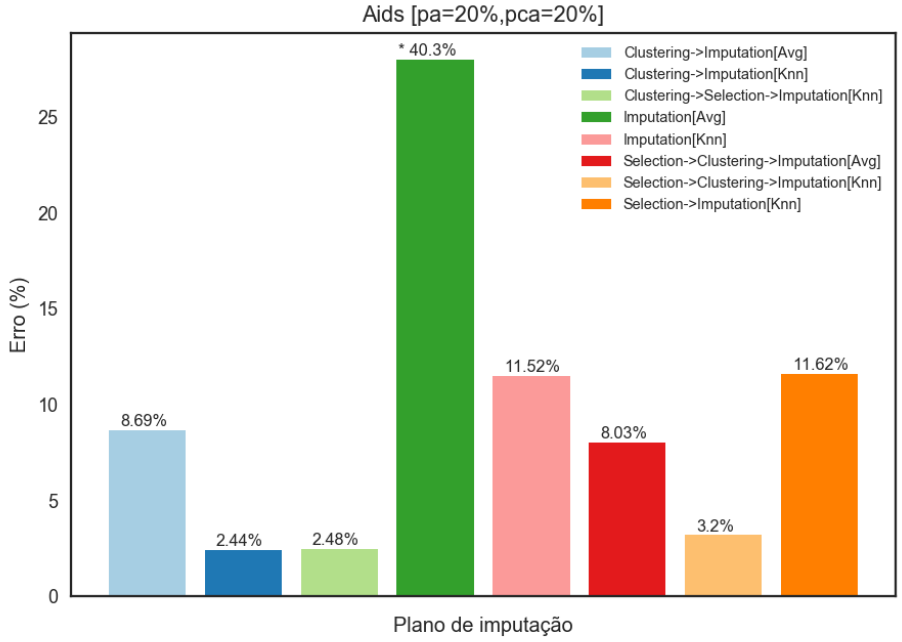

Figura V.8: Experimento com 20% de ausência de dados e redução de 20% na etapa de seleção de atributos.

a de agrupamento.

9) No experimento com 30% de ausência e 30% de redução na seleção de atributos, gráfico V.12, o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo

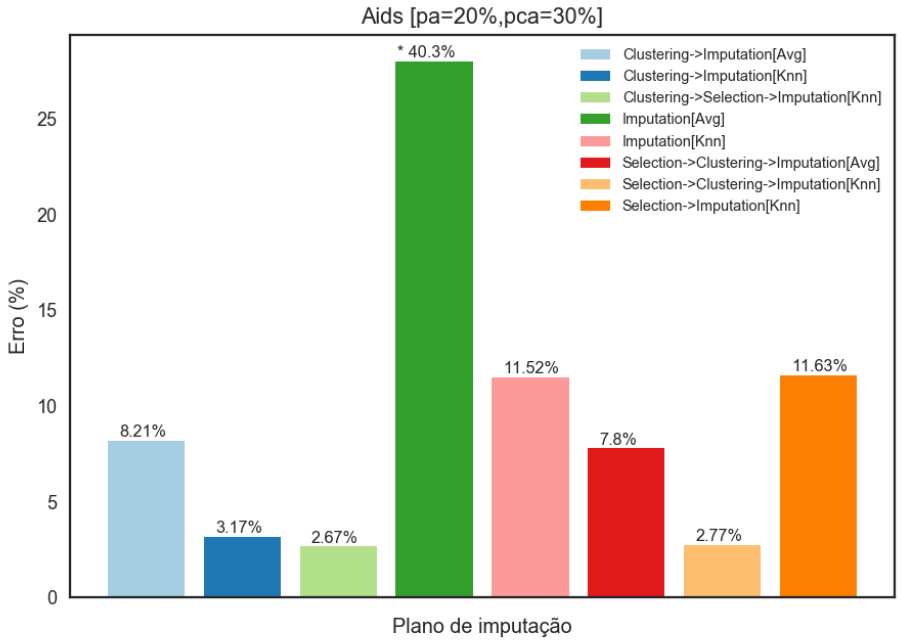

Figura V.9: Experimento com 20% de ausência de dados e redução de 30% na etapa de seleção de atributos.

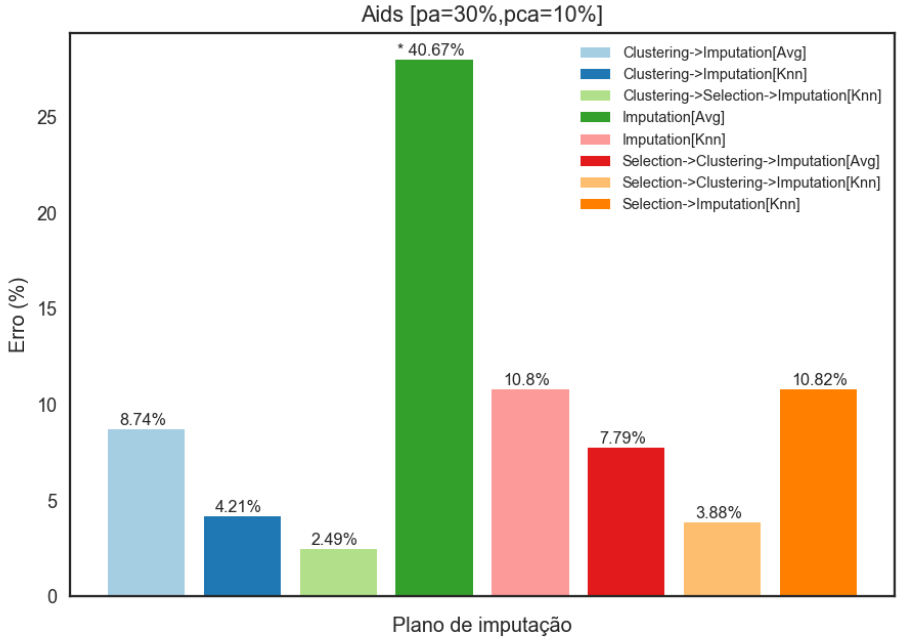

Figura V.10: Experimento com 30% de ausência de dados e redução de 10% na etapa de seleção de atributos.

a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

É possível observar na tabela V.8 o erro médio de todos os planos de imputação executados no experimento. O gráfico V.13 detalha a evolução do plano ASI[Knn], que apresentou a maior precisão

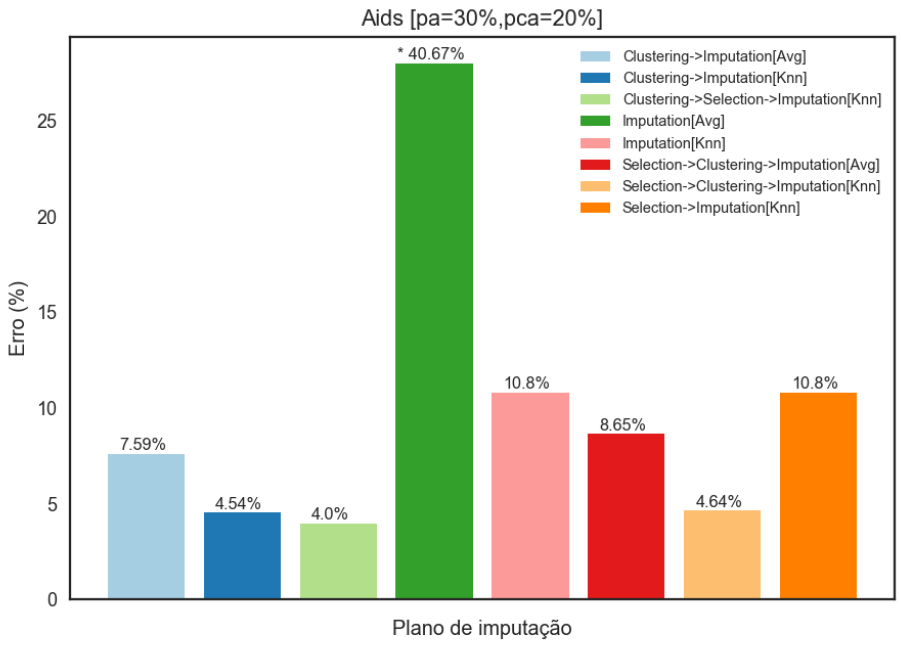

Figura V.11: Experimento com 30% de ausência de dados e redução de 20% na etapa de seleção de atributos.

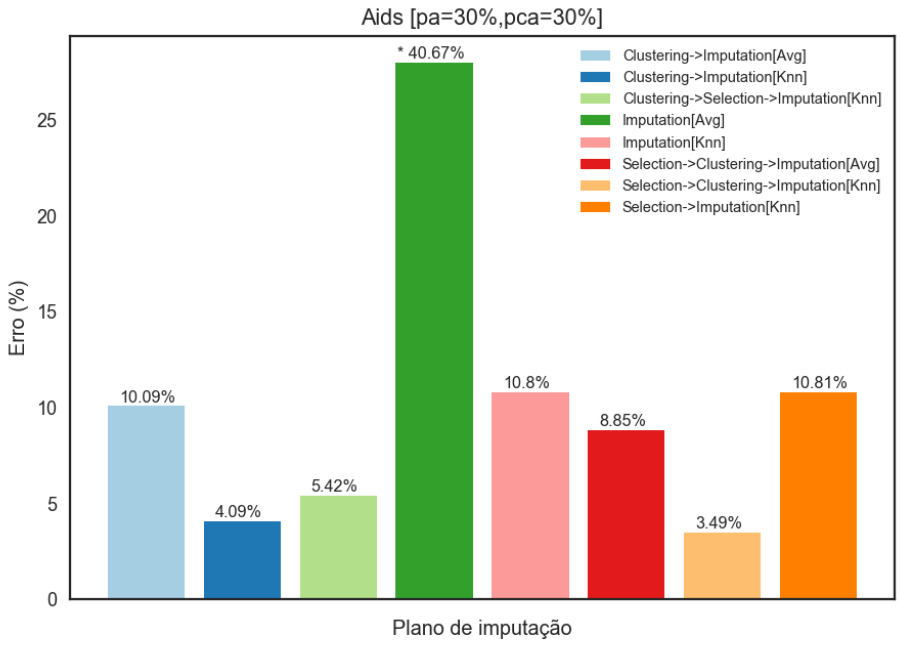

Figura V.12: Experimento com 30% de ausência de dados e redução de 30% na etapa de seleção de atributos.

ao longo dos experimentos realizados. O plano obteve o melhor resultado na configuração com o menor percentual de ausência (10%), quando existem menos dados a serem imputados. Reduzir o número de atributos utilizados na tarefa de seleção diminuiu o erro em dois experimentos, nos

demais o erro do plano aumentou.

Tabela V.8: Aids: erro médio dos experimentos com imputação composta.

Plano de imputação	Erro médio(%)
AI[Avg]	8.23
AI[Knn]	3.09
ASI[Knn]	3.07
I[Avg]	43.40
I[Knn]	11.77
SAI[Avg]	7.90
SAI[Knn]	3.25
SI[Knn]	11.83

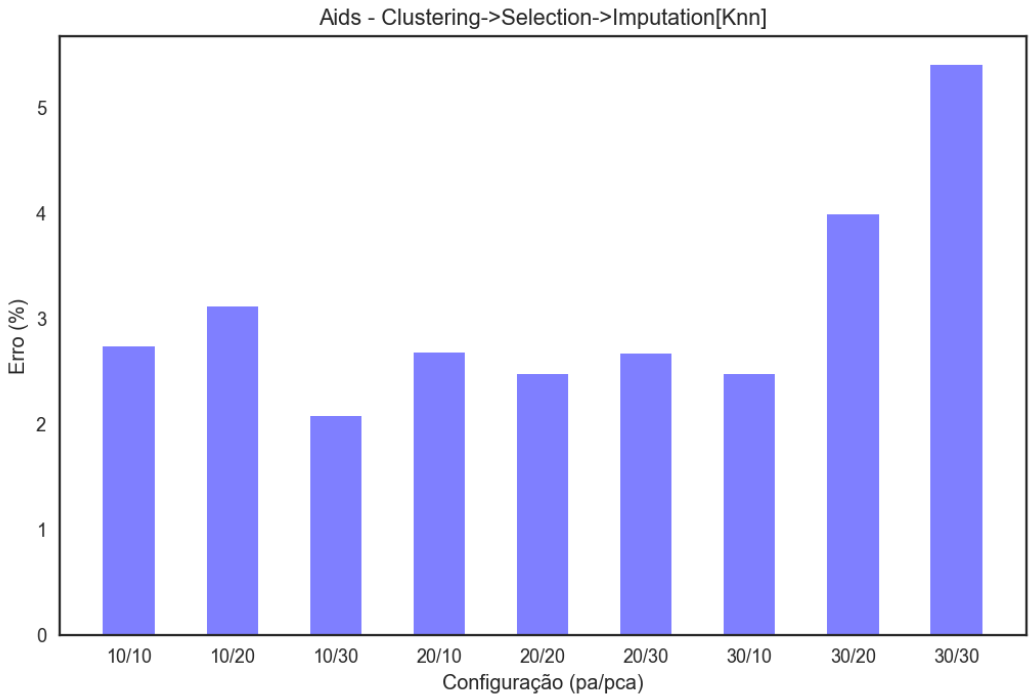

Figura V.13: Desempenho do plano de imputação ASI[Knn] ao longo dos experimentos realizados.

Imputação Composta - Base de Dados Breast Cancer

1) No experimento com 10% de ausência e 10% de redução na seleção de atributos, gráfico V.14, o plano de imputação que obteve o melhor resultado foi: AI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

2) No experimento com 10% de ausência e 20% de redução na seleção de atributos, gráfico V.15, o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

3) No experimento com 10% de ausência e 30% de redução na seleção de atributos, gráfico V.16, o plano de imputação que obteve o melhor resultado foi: ASI[Knn]. A tarefa de agrupa-

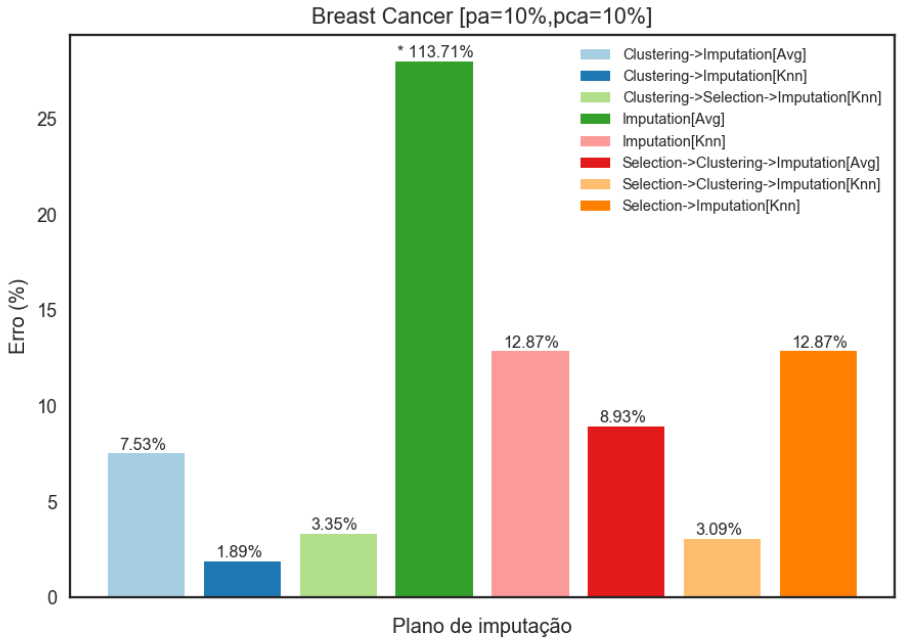

Figura V.14: Experimento com 10% de ausência de dados e redução de 10% na etapa de seleção de atributos.

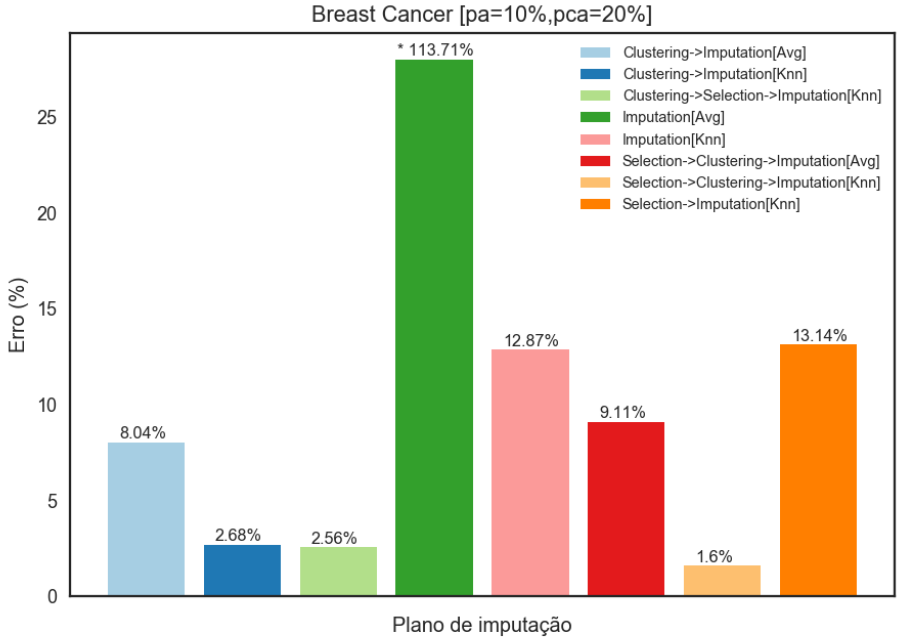

Figura V.15: Experimento com 10% de ausência de dados e redução de 20% na etapa de seleção de atributos.

mento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

4) No experimento com 20% de ausência e 10% de redução na seleção de atributos, gráfico V.17,

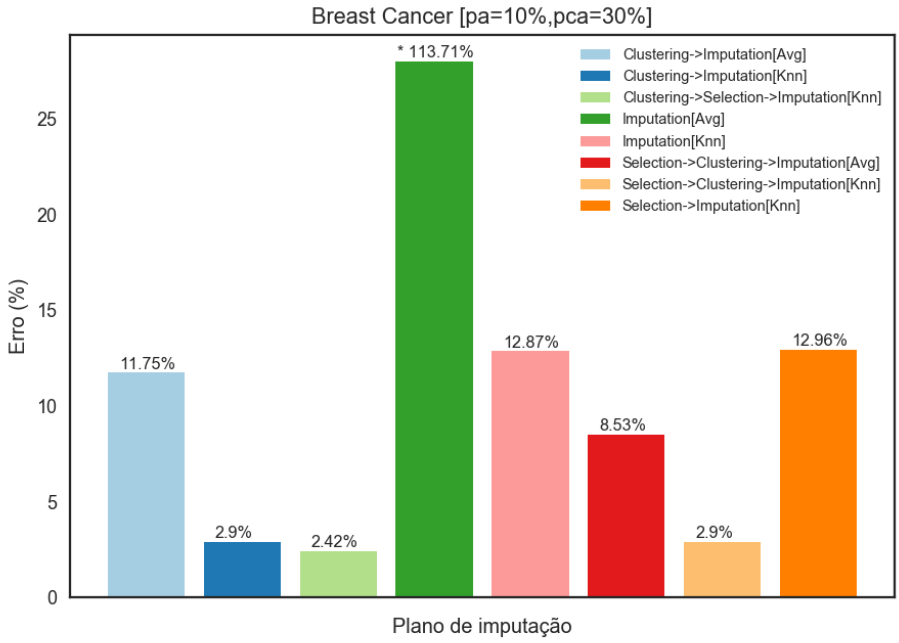

Figura V.16: Experimento com 10% de ausência de dados e redução de 30% na etapa de seleção de atributos.

o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

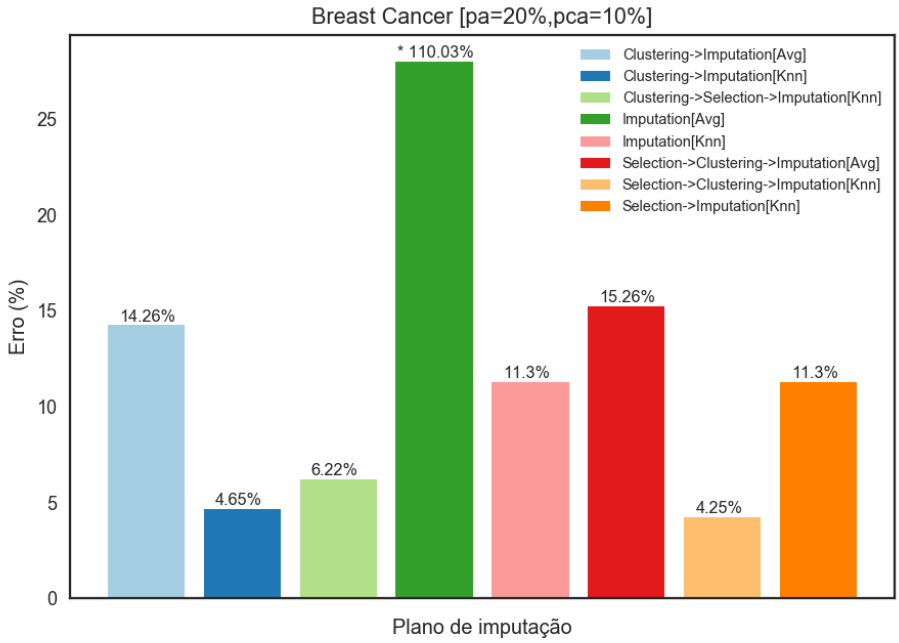

Figura V.17: Experimento com 20% de ausência de dados e redução de 10% na etapa de seleção de atributos.

5) No experimento com 20% de ausência e 20% de redução na seleção de atributos, gráfico V.18,

o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

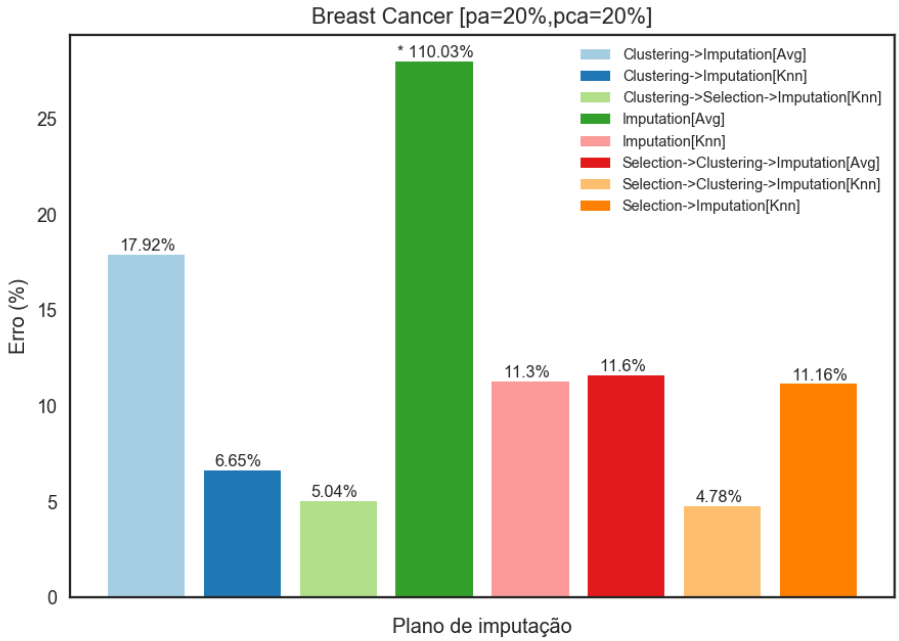

Figura V.18: Experimento com 20% de ausência de dados e redução de 20% na etapa de seleção de atributos.

6) No experimento com 20% de ausência e 30% de redução na seleção de atributos, gráfico V.19, o plano de imputação que obteve o melhor resultado foi: AI[Knn]. A tarefa de agrupamento precedendo a de seleção obteve um resultado melhor do que a de seleção precedendo a de agrupamento.

7) No experimento com 30% de ausência e 10% de redução na seleção de atributos, gráfico V.20, o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

8) No experimento com 30% de ausência e 20% de redução na seleção de atributos, gráfico V.21, o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

9) No experimento com 30% de ausência e 30% de redução na seleção de atributos, gráfico V.22, o plano de imputação que obteve o melhor resultado foi: SAI[Knn]. A tarefa de seleção precedendo a de agrupamento obteve um resultado melhor do que a de agrupamento precedendo a de seleção.

É possível observar na tabela V.9 o erro médio de todos os planos de imputação executados no experimento. O gráfico V.23 detalha a evolução do plano SAI[Knn], que apresentou a maior precisão ao longo dos experimentos realizados. O plano obteve o melhor resultado na configuração com o menor percentual de ausência (10%), quando existem menos dados a serem imputados. Reduzir o número de atributos utilizados na tarefa de seleção diminuiu o erro em dois experimentos, nos

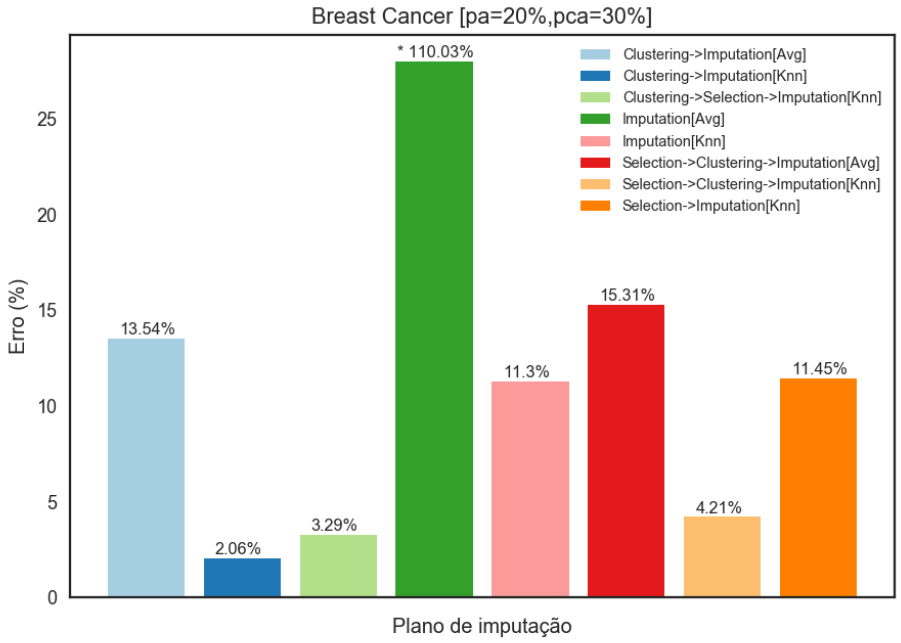

Figura V.19: Experimento com 20% de ausência de dados e redução de 30% na etapa de seleção de atributos.

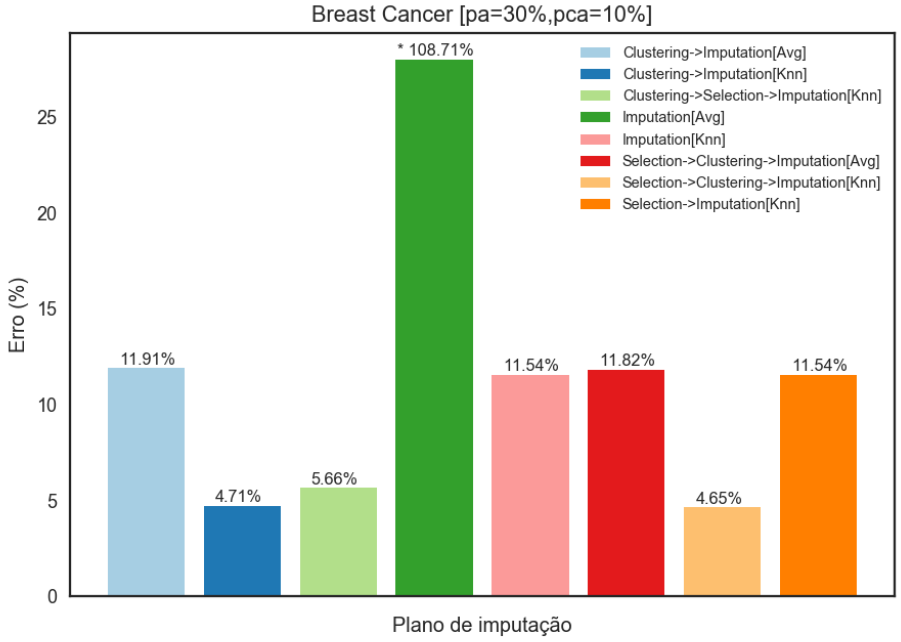

Figura V.20: Experimento com 30% de ausência de dados e redução de 10% na etapa de seleção de atributos.

demais o erro do plano aumentou.

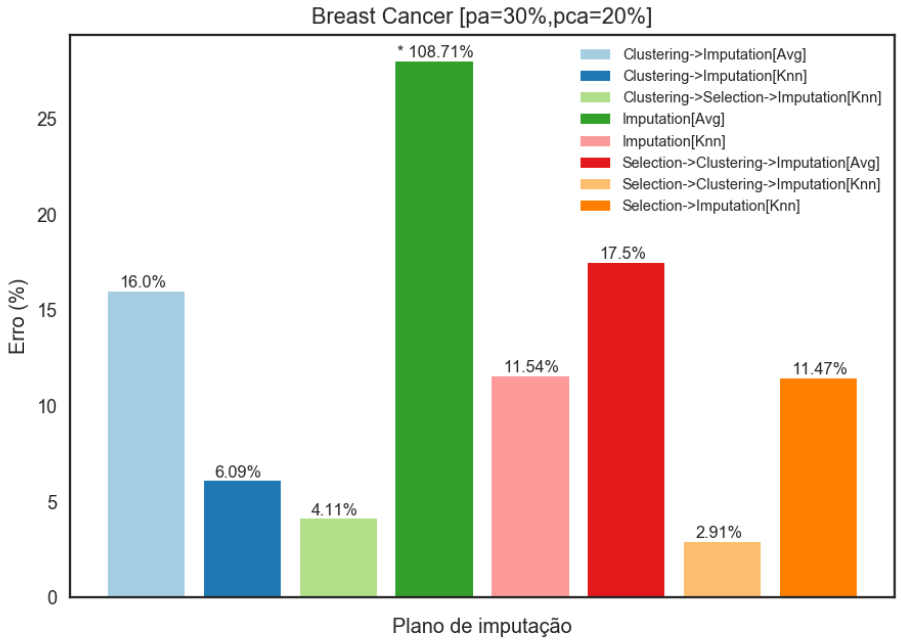

Figura V.21: Experimento com 30% de ausência de dados e redução de 20% na etapa de seleção de atributos.

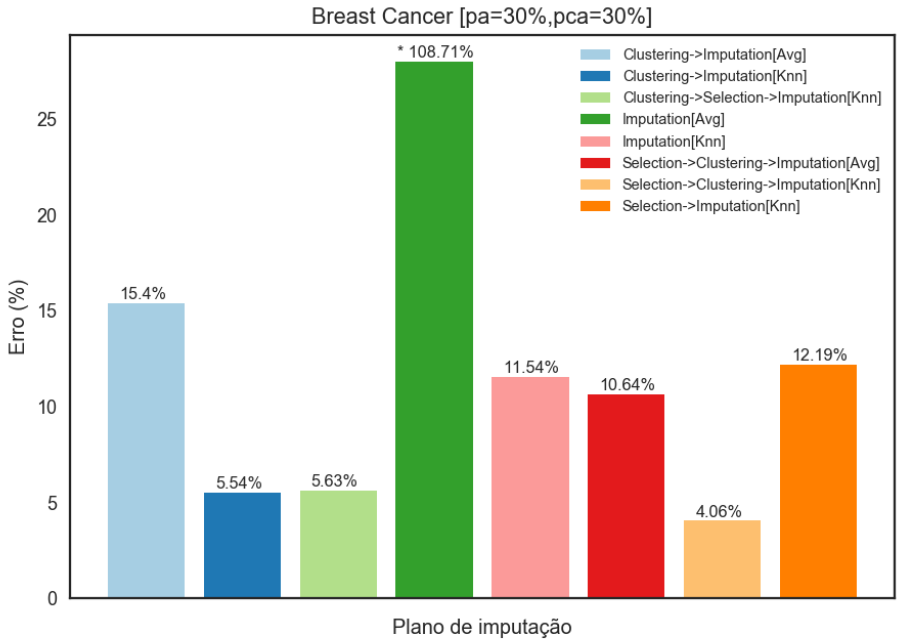

Figura V.22: Experimento com 30% de ausência de dados e redução de 30% na etapa de seleção de atributos.

Imputação Composta e Ensemble - Base de Dados Aids

Conforme explicado anteriormente, optou-se variar o parâmetro T de um a três, cabe ressaltar que a execução com T igual a um é a propria imputação composta, uma vez que com apenas um

Tabela V.9: Breast Cancer: erro médio dos experimentos com imputação composta.

Plano de imputação	Erro médio(%)
AI[Avg]	12.92
AI[Knn]	4.13
ASI[Knn]	4.25
I[Avg]	110.81
I[Knn]	11.90
SAI[Avg]	12.07
SAI[Knn]	3.60
SI[Knn]	12.01

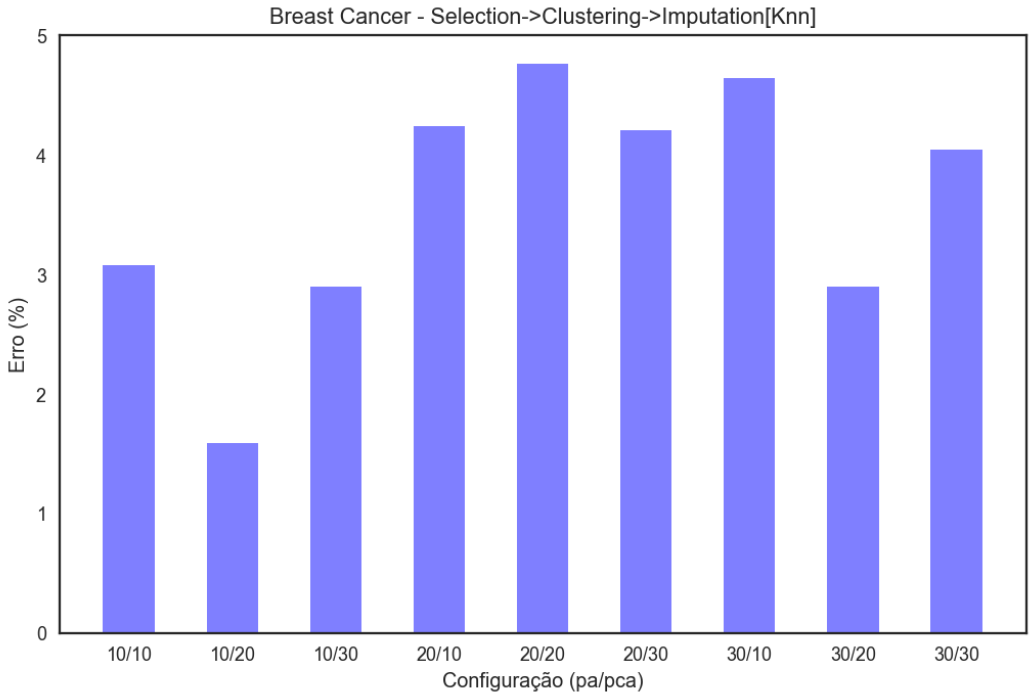

Figura V.23: Desempenho do plano de imputação SAI[Knn] ao longo dos experimentos realizados.

base learner não existe etapa de combinação no algoritmo Bagging.

1) No experimento com 10% de ausência e 10% de redução na seleção de atributos, gráfico V.24, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Nestes planos o erro diminuiu continuamente de T1 a T3. Dois planos apresentaram aumento de erro de T1 para T2. Três planos apresentaram aumento de erro de T2 para T3.

2) No experimento com 10% de ausência e 20% de redução na seleção de atributos, gráfico V.25, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Nestes planos o erro diminuiu continuamente de T1 a T3 em três experimentos. Dois planos apresentaram aumento de erro de T1 para T2. Três planos apresentaram aumento de erro de T2 para T3.

3) No experimento com 10% de ausência e 30% de redução na seleção de atributos, gráfico V.26, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em seis

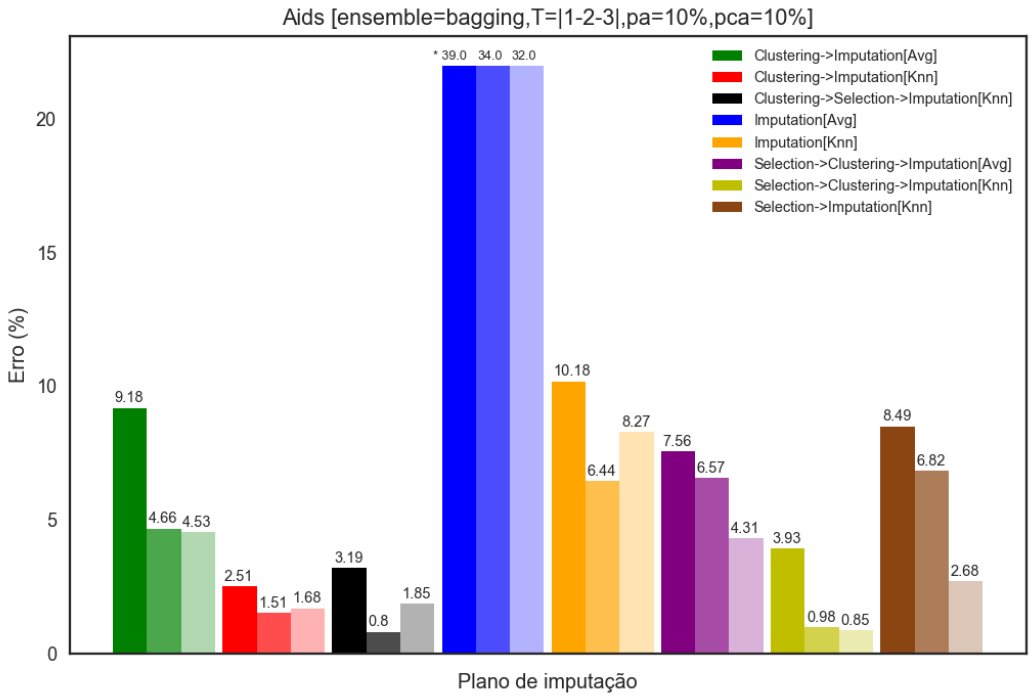

Figura V.24: Experimento com 10% de ausência de dados, redução de 10% na etapa de seleção de atributos e utilização de classificadores bagging.

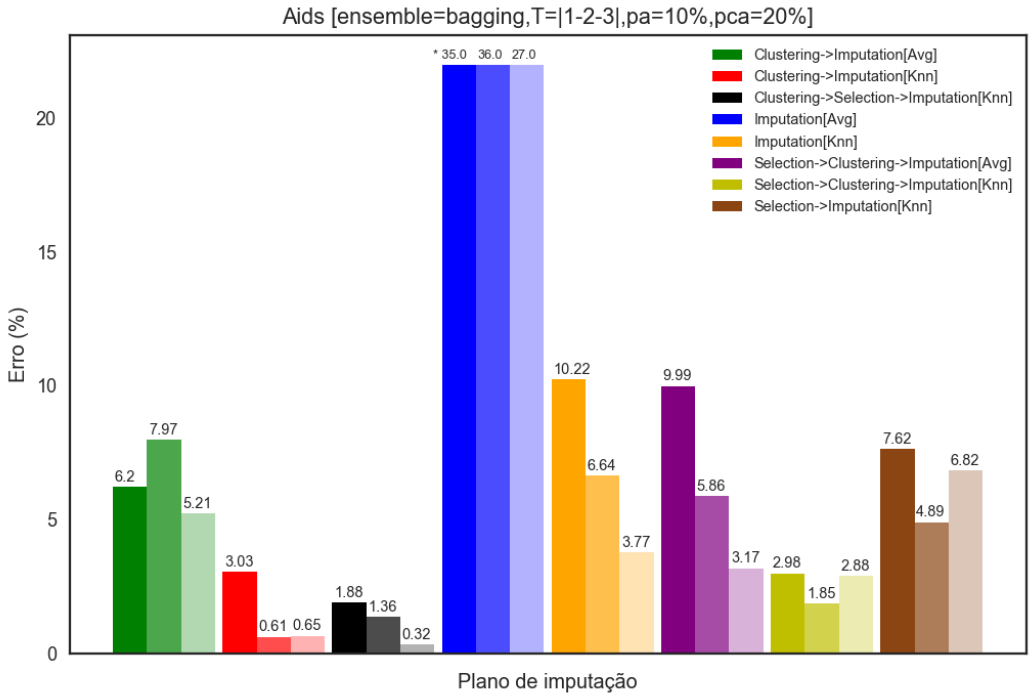

Figura V.25: Experimento com 10% de ausência de dados, redução de 20% na etapa de seleção de atributos e utilização de classificadores bagging.

destes o erro diminuiu continuamente de T1 a T3. Nenhum plano apresentou aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3.

4) No experimento com 20% de ausência e 10% de redução na seleção de atributos, gráfico V.27,

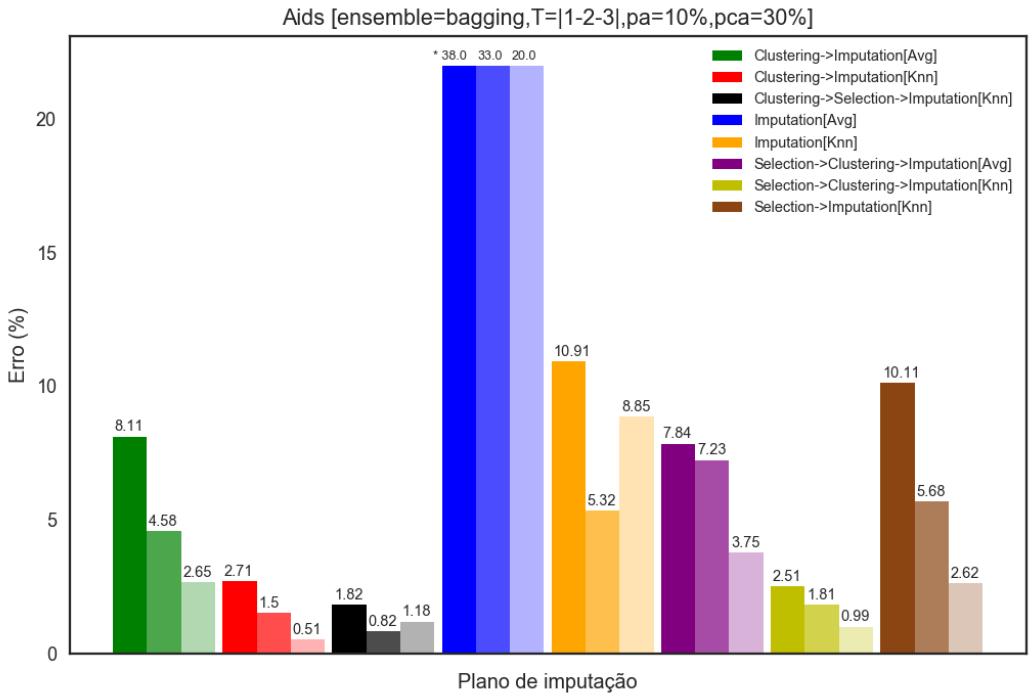

Figura V.26: Experimento com 10% de ausência de dados, redução de 30% na etapa de seleção de atributos e utilização de classificadores bagging.

é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em seis planos o erro diminuiu continuamente de T1 a T3. Nenhum plano apresentou aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3.

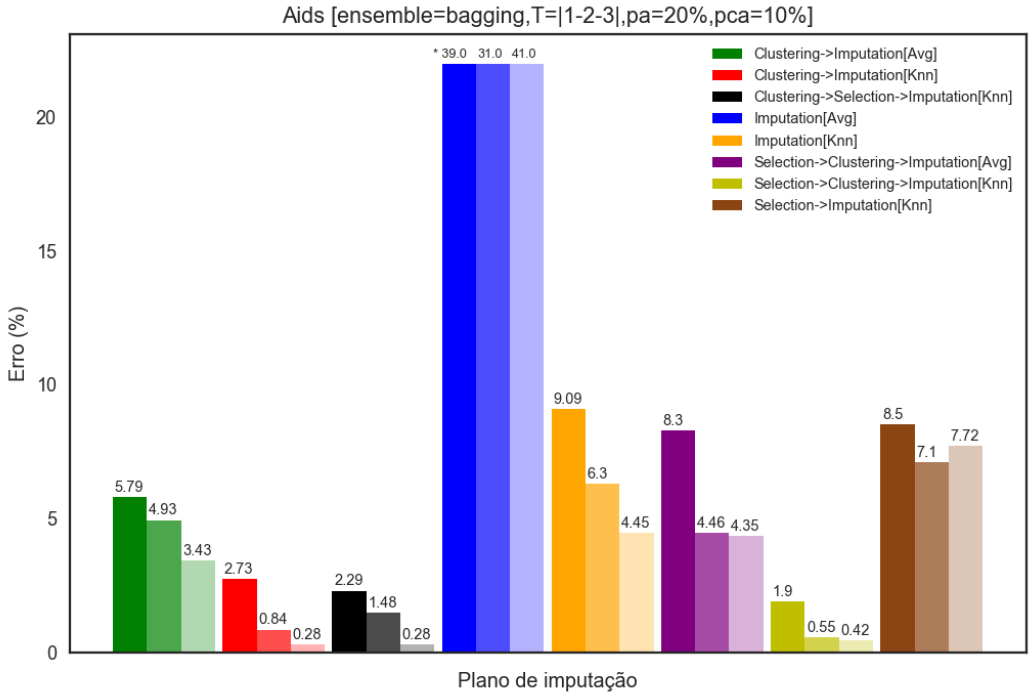

Figura V.27: Experimento com 20% de ausência de dados, redução de 10% na etapa de seleção de atributos e utilização de classificadores bagging.

5) No experimento com 20% de ausência e 20% de redução na seleção de atributos, gráfico V.28, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em sete planos o erro diminuiu continuamente de T1 a T3. Nenhum plano apresentou aumento de erro de T1 para T2. Um plano apresentou aumento de erro de T2 para T3.

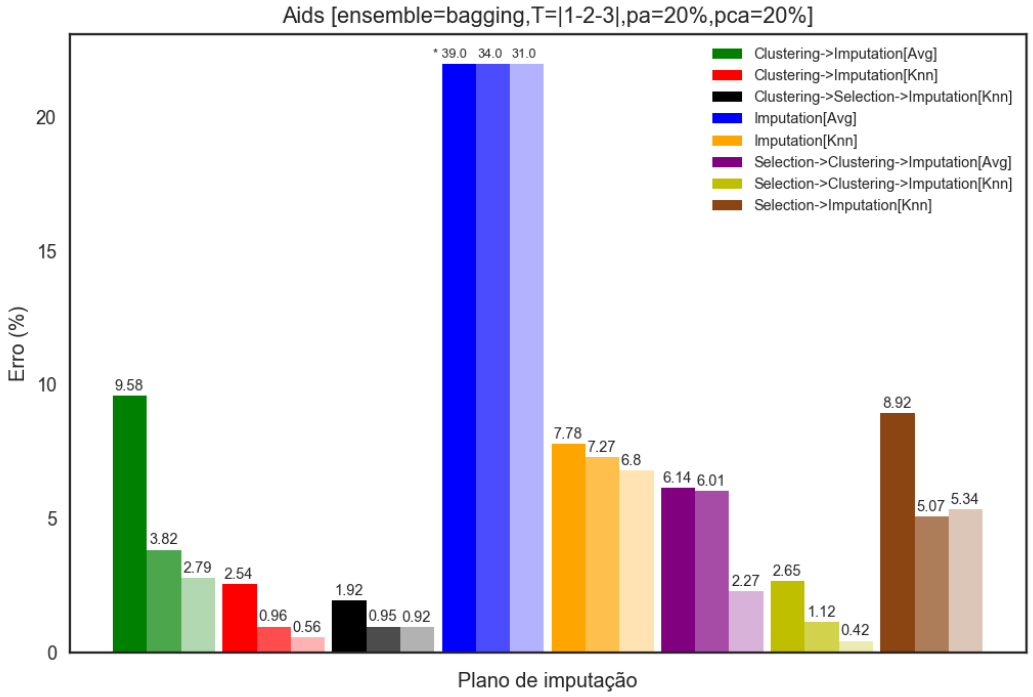

Figura V.28: Experimento com 20% de ausência de dados, redução de 20% na etapa de seleção de atributos e utilização de classificadores bagging.

6) No experimento com 20% de ausência e 30% de redução na seleção de atributos, gráfico V.29, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em quatro planos o erro diminuiu continuamente de T1 a T3. Um plano apresentou aumento de erro de T1 para T2. Três planos apresentaram aumento de erro de T2 para T3.

7) No experimento com 30% de ausência e 10% de redução na seleção de atributos, gráfico V.30, é possível observar que dois experimentos não obtiveram ganhos de precisão com *ensemble*. Em quatro planos o erro diminuiu continuamente de T1 a T3. Dois planos apresentaram aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3.

8) No experimento com 30% de ausência e 20% de redução na seleção de atributos, gráfico V.31, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em cinco planos o erro diminuiu continuamente de T1 a T3. Nenhum plano apresentou aumento de erro de T1 para T2. Três planos apresentaram aumento de erro de T2 para T3.

9) No experimento com 30% de ausência e 30% de redução na seleção de atributos, gráfico V.32, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em sete planos o erro diminuiu continuamente de T1 a T3. Nenhum plano apresentou aumento de erro de

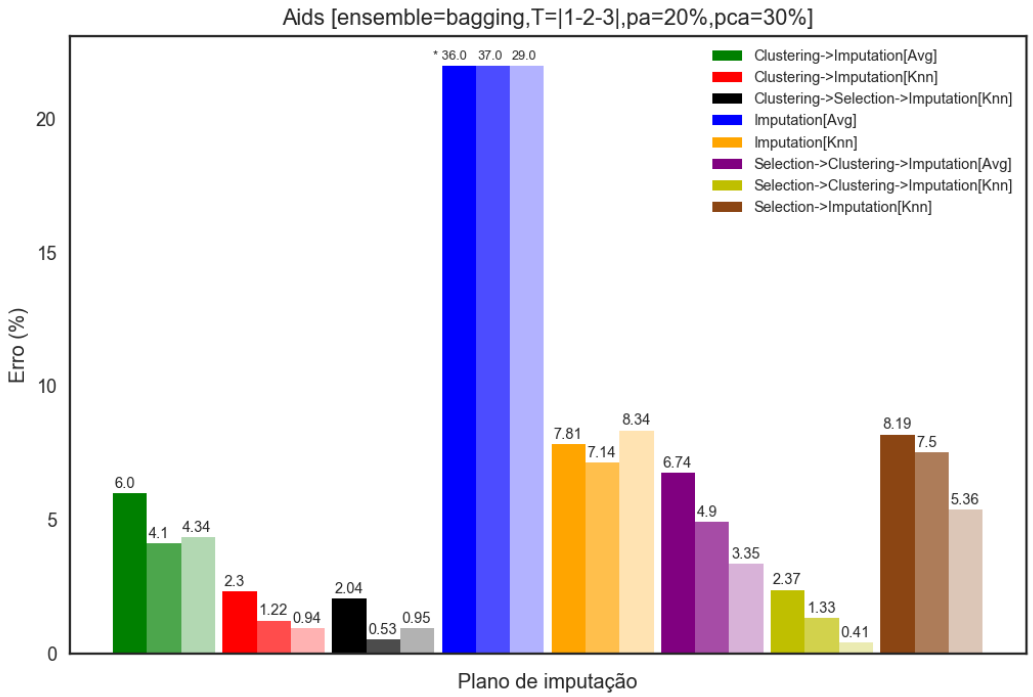

Figura V.29: Experimento com 20% de ausência de dados, redução de 30% na etapa de seleção de atributos e utilização de classificadores bagging.

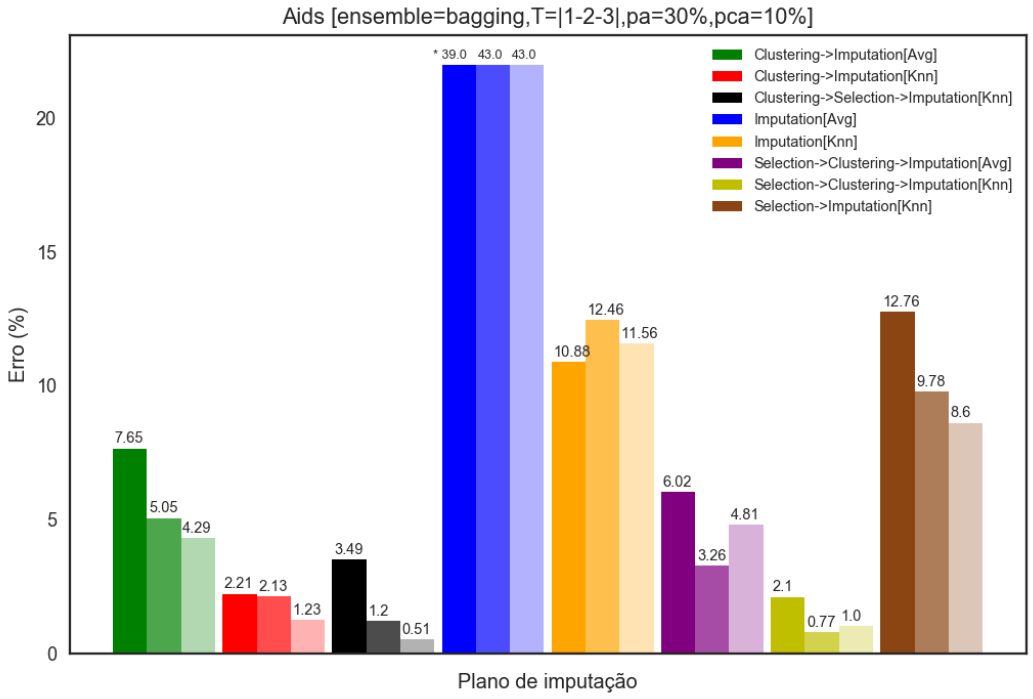

Figura V.30: Experimento com 30% de ausência de dados, redução de 10% na etapa de seleção de atributos e utilização de classificadores bagging.

T1 para T2. Um plano apresentou aumento de erro de T2 para T3.

É possível observar na tabela V.10 o erro médio de todos os planos de imputação executados no experimento. O gráfico V.33 detalha a evolução do plano ASI[Knn] →, que apresentou a maior

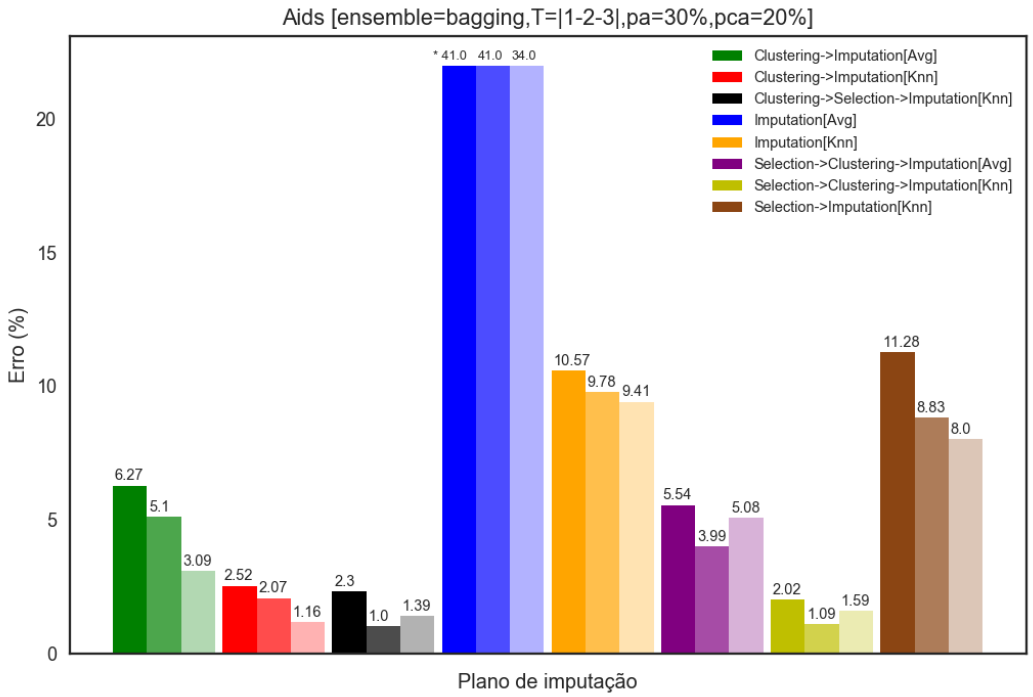

Figura V.31: Experimento com 30% de ausência de dados, redução de 20% na etapa de seleção de atributos e utilização de classificadores bagging.

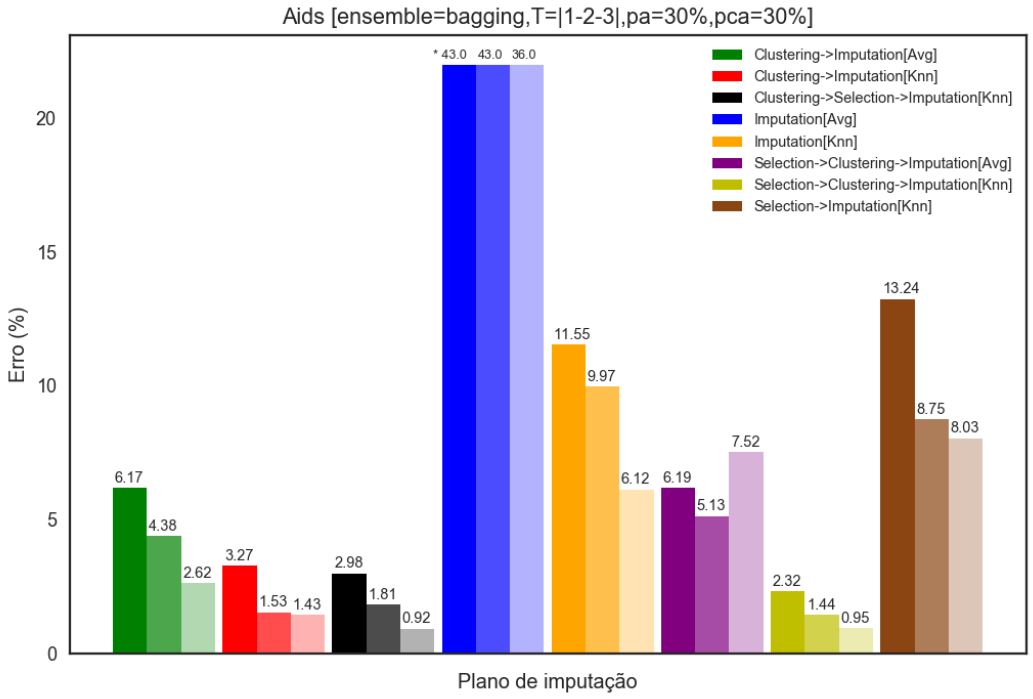

Figura V.32: Experimento com 30% de ausência de dados, redução de 30% na etapa de seleção de atributos e utilização de classificadores bagging.

precisão ao longo dos experimentos realizados. O plano obteve o melhor resultado com a seguinte configuração: 20% de percentual de ausência, 10% de redução na tarefa de seleção de atributos e *Bagging* com T = 3.

Tabela V.10: Aids: erro médio dos experimentos com imputação composta e ensemble.

Plano de imputação	Erro médio(%)
AI[Avg][T-1]	7.21
AI[Avg][T-2]	4.95
AI[Avg][T-3]	3.66
AI[Knn][T-1]	2.64
AI[Knn][T-2]	1.37
AI[Knn][T-3]	0.93
ASI[Knn][T-1]	2.43
ASI[Knn][T-2]	1.10
ASI[Knn][T-3]	0.92
I[Avg][T-1]	38.80
I[Avg][T-2]	36.93
I[Avg][T-3]	32.74
I[Knn][T-1]	9.88
I[Knn][T-2]	7.92
I[Knn][T-3]	7.50
SAI[Avg][T-1]	7.14
SAI[Avg][T-2]	5.26
SAI[Avg][T-3]	4.28
SAI[Knn][T-1]	2.53
SAI[Knn][T-2]	1.21
SAI[Knn][T-3]	1.05
SI[Knn][T-1]	9.89
SI[Knn][T-2]	7.15
SI[Knn][T-3]	6.13

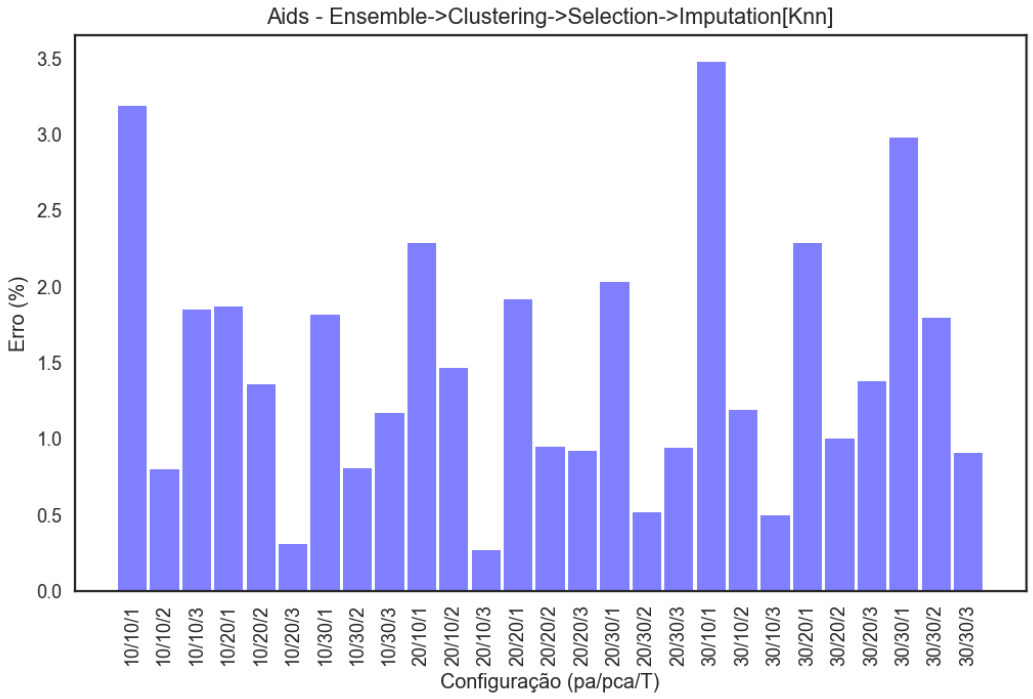

Figura V.33: Desempenho do plano de imputação ASI[Knn] ao longo dos experimentos realizados.

Imputação Composta e Ensemble - Base de Dados Breast Cancer

1) No experimento com 10% de ausência e 10% de redução na seleção de atributos, gráfico V.34, é possível observar que seis experimentos obtiveram ganhos de precisão com *ensemble*. Em três planos o erro diminuiu continuamente de T1 a T3. Quatro planos apresentaram aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3. Dois experimentos apresentaram erro igual a zero utilizando *ensemble*.

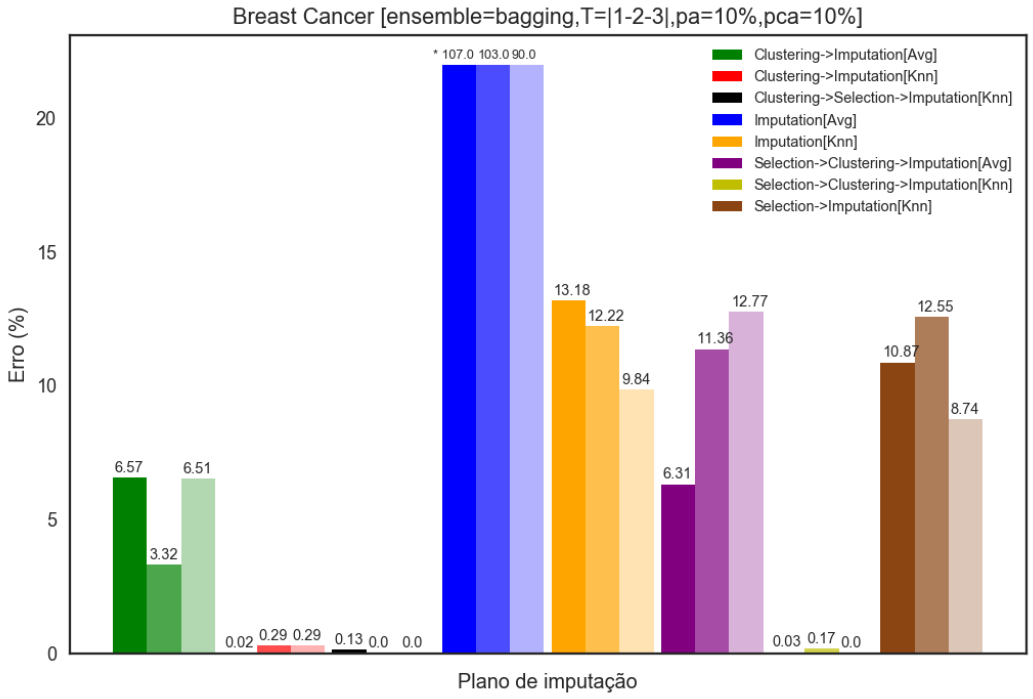

Figura V.34: Experimento com 10% de ausência de dados, redução de 10% na etapa de seleção de atributos e utilização de classificadores bagging.

2) No experimento com 10% de ausência e 20% de redução na seleção de atributos, gráfico V.35, é possível observar que sete experimentos obtiveram ganhos de precisão com *ensemble*. Em três planos o erro diminuiu continuamente de T1 a T3. Três planos apresentaram aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3. Dois experimentos apresentaram erro igual a zero utilizando *ensemble*.

3) No experimento com 10% de ausência e 30% de redução na seleção de atributos, gráfico V.36, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em quatro planos o erro diminuiu continuamente de T1 a T3. Nenhum plano apresentou aumento de erro de T1 para T2. Quatro planos apresentaram aumento de erro de T2 para T3. Três experimentos apresentaram erro igual a zero utilizando *ensemble*.

4) No experimento com 20% de ausência e 10% de redução na seleção de atributos, gráfico V.37, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em

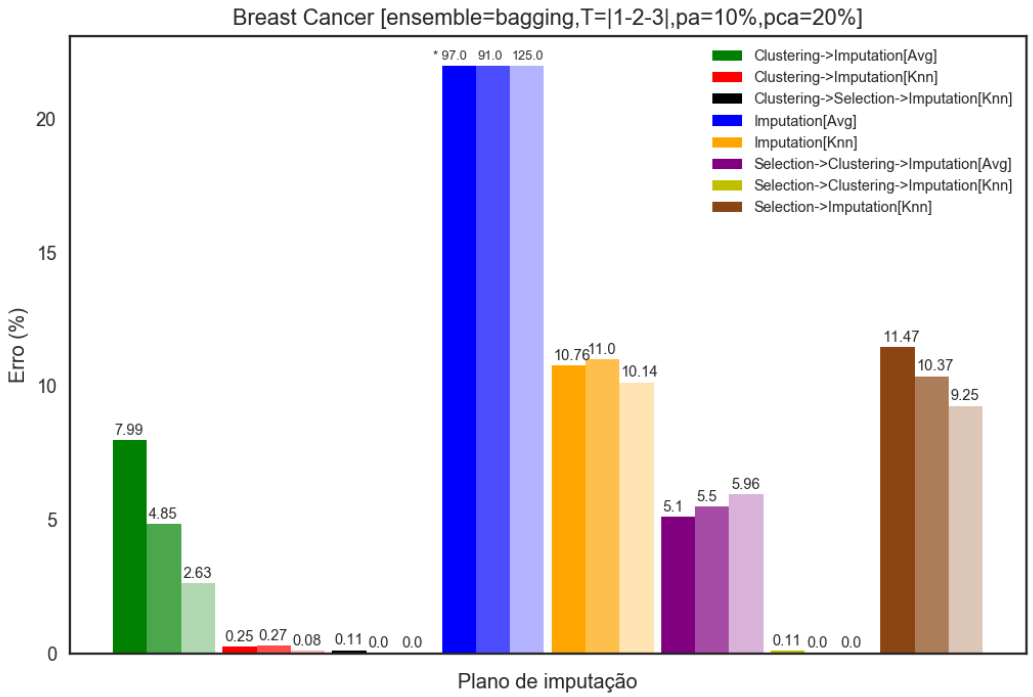

Figura V.35: Experimento com 10% de ausência de dados, redução de 20% na etapa de seleção de atributos e utilização de classificadores bagging.

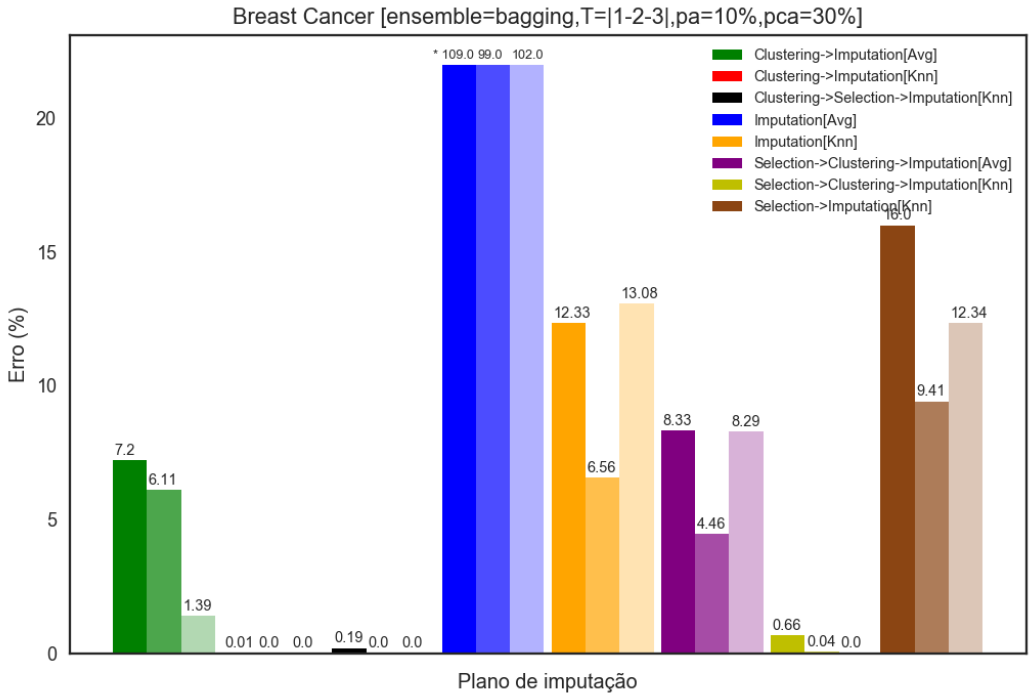

Figura V.36: Experimento com 10% de ausência de dados, redução de 30% na etapa de seleção de atributos e utilização de classificadores bagging.

quatro planos o erro diminuiu continuamente de T1 a T3. Dois planos apresentaram aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3. Quatro experimentos apresentaram erro igual a zero utilizando *ensemble*.

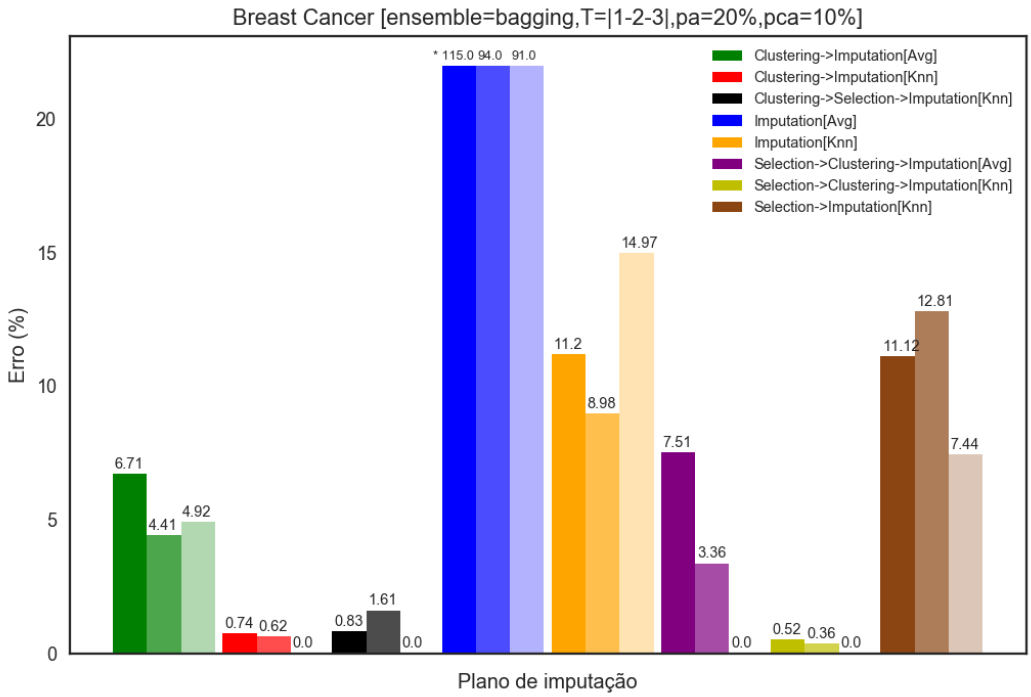

Figura V.37: Experimento com 20% de ausência de dados, redução de 10% na etapa de seleção de atributos e utilização de classificadores bagging.

5) No experimento com 20% de ausência e 20% de redução na seleção de atributos, gráfico V.38, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em quatro planos o erro diminuiu continuamente de T1 a T3. Dois planos apresentaram aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3. Dois experimentos apresentaram erro igual a zero utilizando *ensemble*.

6) No experimento com 20% de ausência e 30% de redução na seleção de atributos, gráfico V.39, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em quatro planos o erro diminuiu continuamente de T1 a T3. Dois planos apresentaram aumento de erro de T1 para T2. Dois planos apresentaram aumento de erro de T2 para T3. Dois experimentos apresentaram erro igual a zero utilizando *ensemble*.

7) No experimento com 30% de ausência e 10% de redução na seleção de atributos, gráfico V.40, é possível observar que sete experimentos obtiveram ganhos de precisão com *ensemble*. Em três planos o erro diminuiu continuamente de T1 a T3. Três planos apresentaram aumento de erro de T1 para T2. Três planos apresentaram aumento de erro de T2 para T3. Um experimento apresentou erro igual a zero utilizando *ensemble*.

8) No experimento com 30% de ausência e 20% de redução na seleção de atributos, gráfico V.41, é possível observar que sete experimentos obtiveram ganhos de precisão com *ensemble*. Em dois planos o erro diminuiu continuamente de T1 a T3. Um plano apresentou aumento de erro de T1 para T2. Seis planos apresentaram aumento de erro de T2 para T3. Nenhum experimento

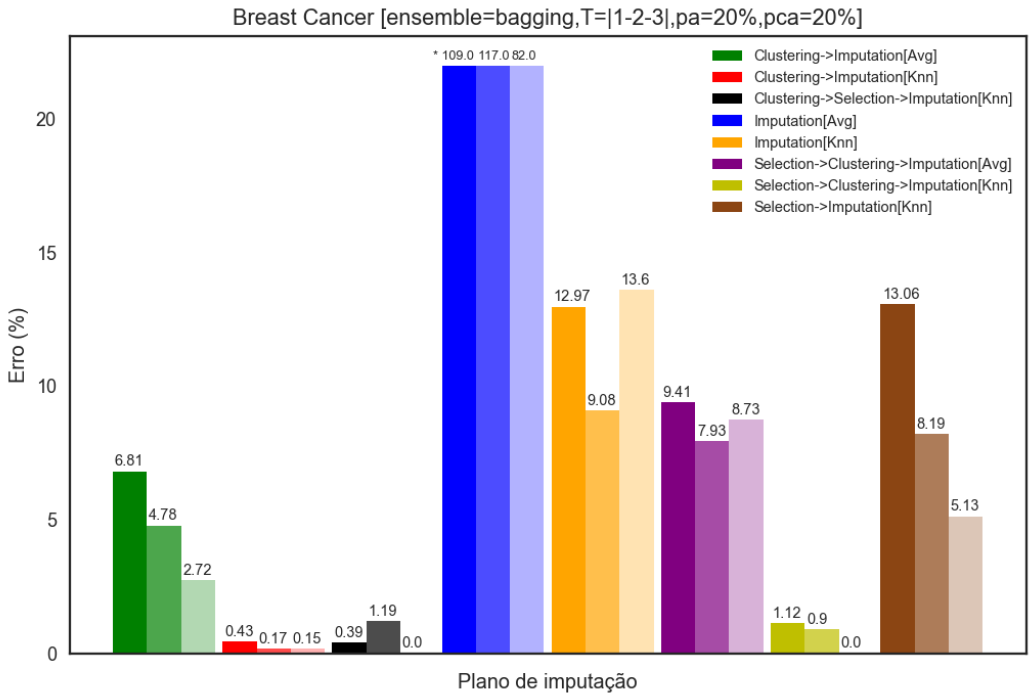

Figura V.38: Experimento com 20% de ausência de dados, redução de 20% na etapa de seleção de atributos e utilização de classificadores bagging.

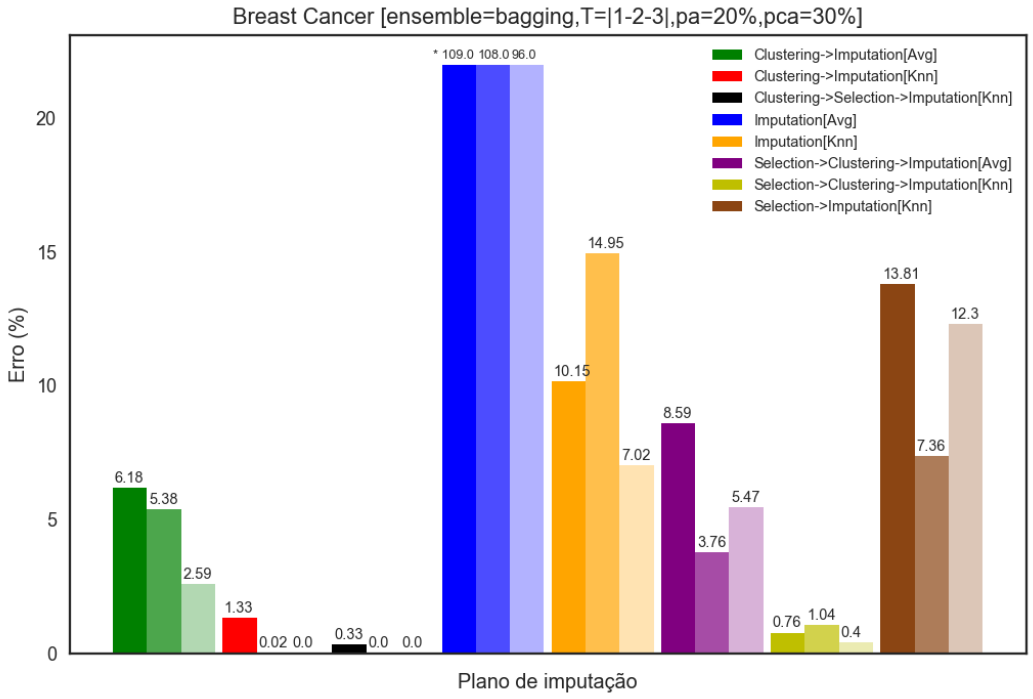

Figura V.39: Experimento com 20% de ausência de dados, redução de 30% na etapa de seleção de atributos e utilização de classificadores bagging.

apresentou erro igual a zero.

9) No experimento com 30% de ausência e 30% de redução na seleção de atributos, gráfico V.42, é possível observar que todos os experimentos obtiveram ganhos de precisão com *ensemble*. Em

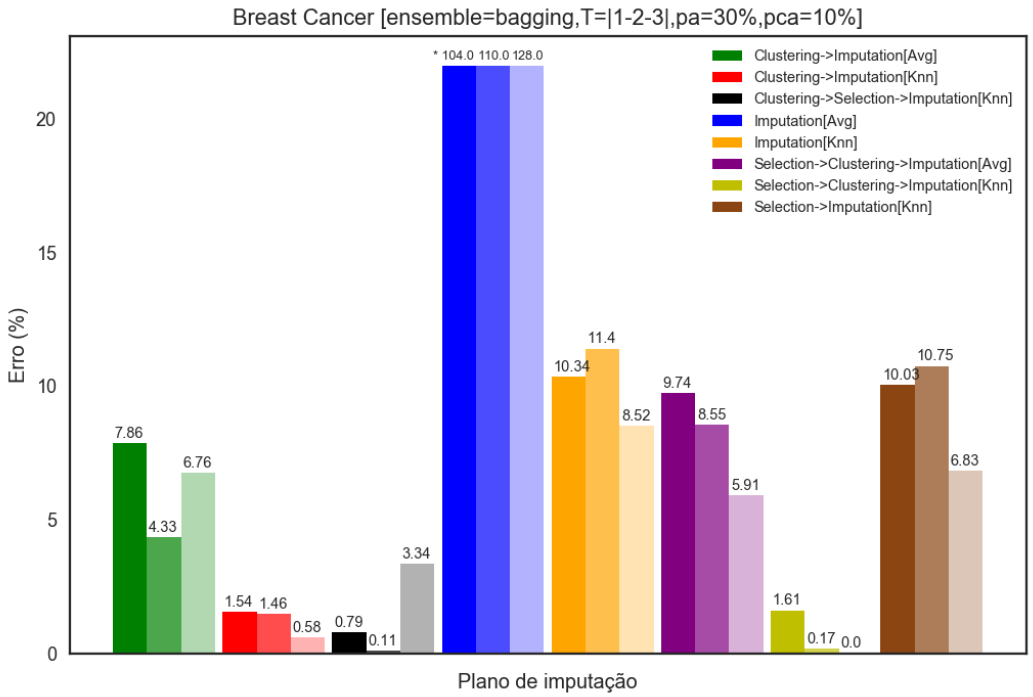

Figura V.40: Experimento com 30% de ausência de dados, redução de 10% na etapa de seleção de atributos e utilização de classificadores bagging.

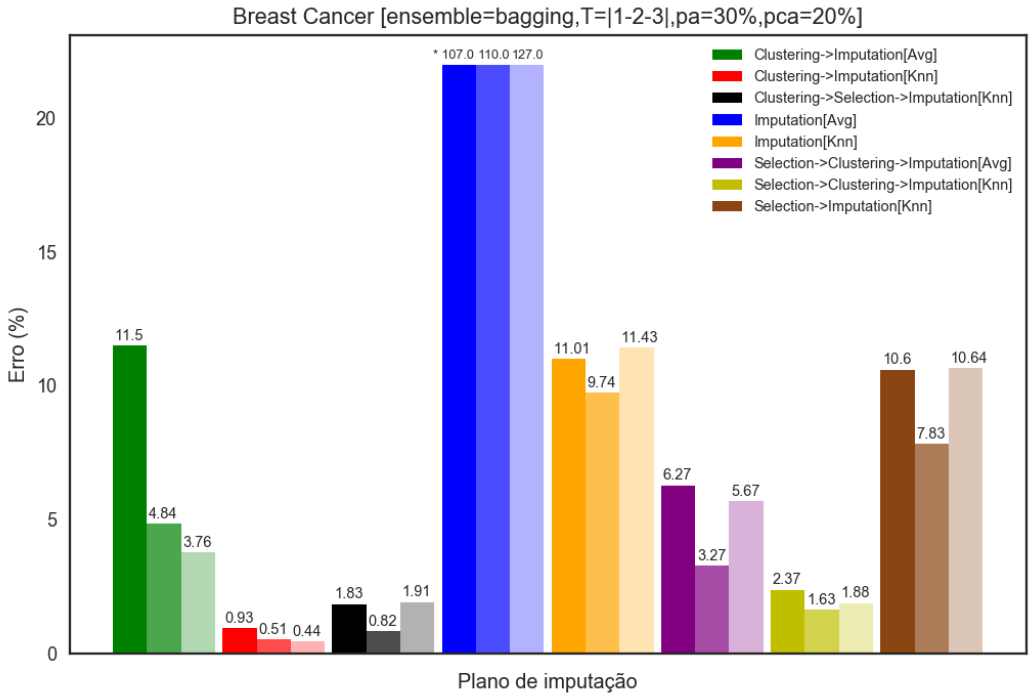

Figura V.41: Experimento com 30% de ausência de dados, redução de 20% na etapa de seleção de atributos e utilização de classificadores bagging.

dois planos o erro diminuiu continuamente de T1 a T3. Dois planos apresentaram aumento de erro de T1 para T2. Quatro planos apresentaram aumento de erro de T2 para T3. Um experimento apresentou erro igual a zero utilizando *ensemble*.

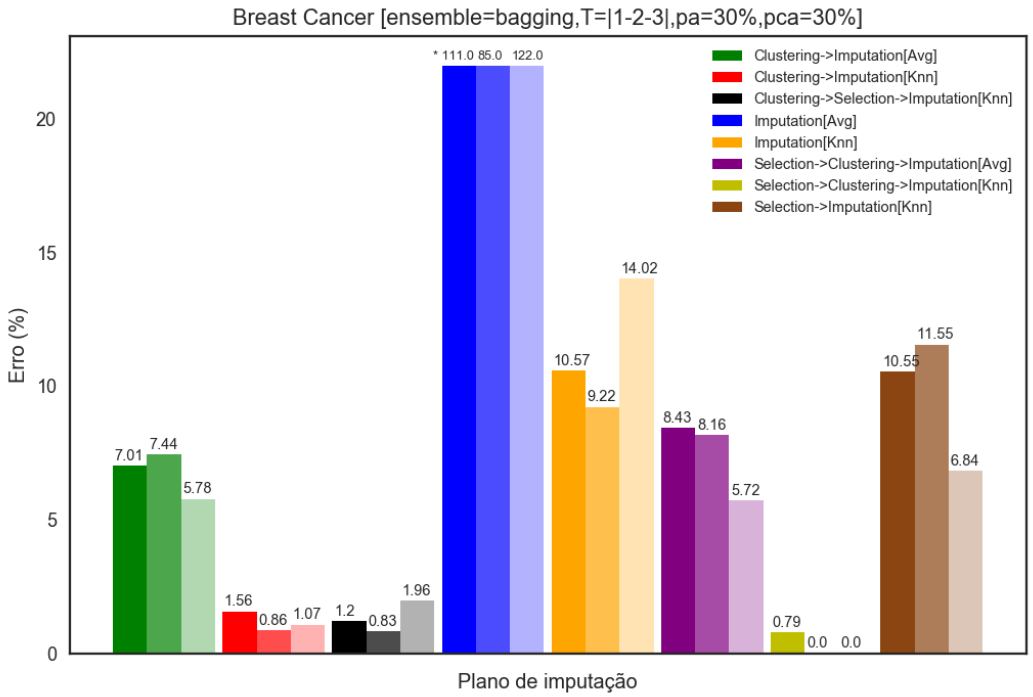

Figura V.42: Experimento com 30% de ausência de dados, redução de 30% na etapa de seleção de atributos e utilização de classificadores bagging.

É possível observar na tabela V.11 o erro médio de todos os planos de imputação executados no experimento. O gráfico V.43 detalha a evolução do plano SAI[Knn] →, que apresentou a maior precisão ao longo dos experimentos realizados. O plano obteve o melhor resultado com o parâmetro T igual a 3. O erro foi igual a zero em 9 experimentos, todos utilizando *Bagging* (em dois casos com o parâmetro T igual a 2 e nos demais com T igual a 3). Além disso, em 6 experimentos o erro diminuiu de forma contínua ao longo da variação do parâmetro T.

A tabela V.12 apresenta os erros médios dos planos de imputação com melhor desempenho em cada um dos experimentos realizados, nas duas bases de dados.

Os planos de imputação ASI[Knn] e SAI[Knn] apresentaram-se, respectivamente, como as melhores estratégias de imputação para as bases de dados Aids e Breast Cancer. A alternâncias entre as tarefas de seleção e agrupamento quanto a ordem de precedência nos planos citados provavelmente se dê pelas características intrínsecas das duas bases de dados; como o nível correlação de seus atributos (para a tarefa de seleção) e o nível de similaridade entre suas tuplas (agrupamento). O método proposto apresentou melhor resultado na base Breast Cancer, possivelmente pela elevada correlação entre os atributos da mesma.

Tabela V.11: Breast Cancer: erro médio dos experimentos com imputação composta e ensemble.

Plano de imputação	Erro médio(%)
AI[Avg][T-1]	7.53
AI[Avg][T-2]	5.05
AI[Avg][T-3]	4.11
AI[Knn][T-1]	0.75
AI[Knn][T-2]	0.46
AI[Knn][T-3]	0.29
ASI[Knn][T-1]	0.64
ASI[Knn][T-2]	0.50
ASI[Knn][T-3]	0.80
I[Avg][T-1]	107.51
I[Avg][T-2]	101.71
I[Avg][T-3]	106.94
I[Knn][T-1]	11.38
I[Knn][T-2]	10.35
I[Knn][T-3]	11.40
SAI[Avg][T-1]	7.74
SAI[Avg][T-2]	6.26
SAI[Avg][T-3]	6.50
SAI[Knn][T-1]	0.88
SAI[Knn][T-2]	0.47
SAI[Knn][T-3]	0.25
SI[Knn][T-1]	11.94
SI[Knn][T-2]	10.09
SI[Knn][T-3]	8.83

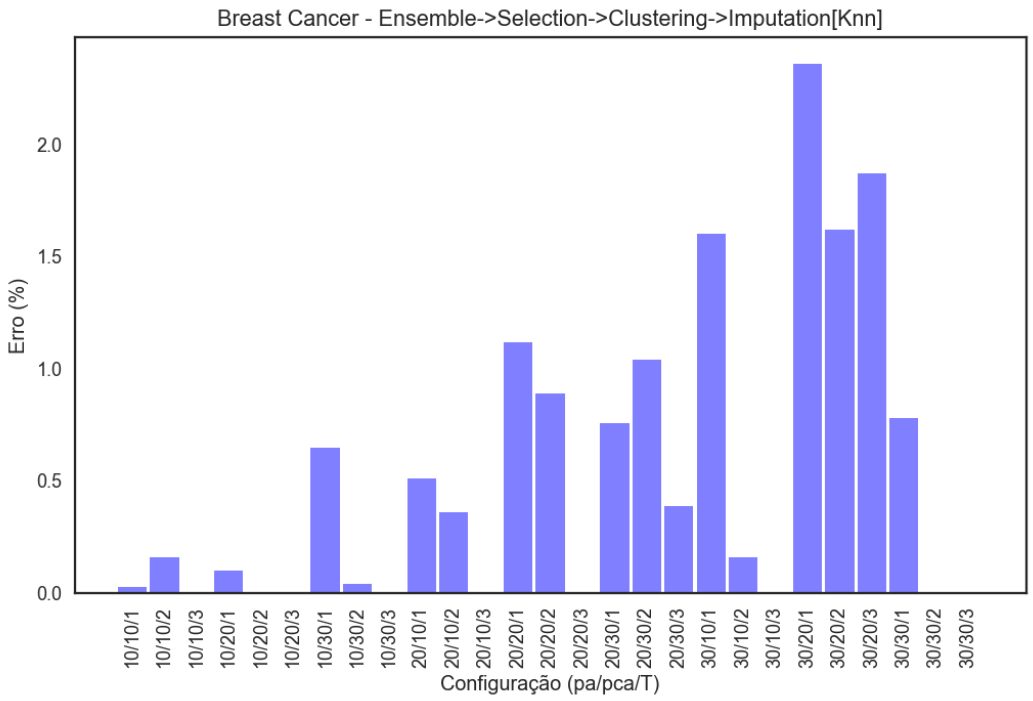

Figura V.43: Desempenho do plano de imputação SAI[Knn] ao longo dos experimentos realizados.

Tabela V.12: Erro médio dos planos de imputação com melhor desempenho ao longo dos experimentos realizados. Comparação por base de dados e técnica utilizada.

Base de dados	Técnica	Plano de imputação	Erro médio(%)
Aids	IC	ASI[Knn]	3.07
	IC+Ensemble	ASI[Knn][T-3]	<u>0.92</u>
Breast cancer	IC	SAI[Knn]	3.60
	IC+Ensemble	SAI[Knn][T-3]	<u>0.25</u>

Capítulo VI Considerações Finais

Neste capítulo são apresentadas as considerações finais, um resumo dos resultados e as principais contribuições e direções da pesquisa realizada.

VI.1 Proposta

A tarefa de complementação de dados ausentes é importante para o processo de descoberta de conhecimento em bases dados. Caso não seja tratada de forma adequada, pode enviesar os resultados gerados por qualquer técnica produtora de informação.

Neste trabalho foi proposto um método de imputação que utiliza a tarefa de classificação *ensemble*, através do algoritmo *Bagging*, combinada a técnica de imputação composta. Para atingir esse objetivo, o modelo proposto foi materializado através da implementação do *framework* Appraisal-Spark, possibilitando o processamento de planos de imputação em larga escala.

VI.2 Síntese dos Resultados

De forma a avaliar o método proposto, um total de 23.040 experimentos foram conduzidos, totalizando 671 horas de processamento, para a simulação de percentuais de ausência em duas bases de dados, onde posteriormente os valores imputados foram comparados com os originais.

As imputações foram realizadas através do *framework* Appraisal-Spark, em modo de execução serial e paralelo. O método proposto com a utilização de classificadores *ensemble* combinada a imputação composta foi capaz de aumentar a precisão dos valores imputados em quase todos os experimentos realizados, em comparação com os experimentos que utilizaram a imputação composta sem a tarefa de *ensemble*.

VI.3 Contribuições

A utilização da tarefa de *ensemble* combinada com a imputação composta aumentou a precisão do dado imputado em 95,83% dos experimentos realizados. O processo de imputação em larga escala com Appraisal-Spark obteve ganho médio de desempenho da ordem de 47,69%, em comparação com a execução em modo serial. O método proposto apresentou bons resultados quanto a

identificação do melhor plano de imputação para cada base de dados e cenário de ausência estudados. Ao longo deste trabalho foi elaborado um artigo com foco em imputação *hot-deck*, utilizando técnicas de aprendizado de máquina [Tavares et al., 2018], sendo este uma das inspirações da técnica apresentada.

VI.4 Trabalhos Futuros

O *framework* Appraisal-Spark possibilita a implementação de outros algoritmos de aprendizado de máquina nas tarefas de *ensemble*, seleção, agrupamento e imputação. Seria interessante, por exemplo, realizar experimentos utilizando *Boosting* na tarefa de *ensemble*, comparando os resultados obtidos com os deste trabalho.

É possível explorar também a utilização do *Appraisal-Spark* no que tange a configuração do *cluster* Spark utilizado. Aumentar a quantidade de mémória e o número de *cores* disponíveis poderá ampliar a quantidade de *base learners* criados com o algoritmo *Bagging* e, possivelmente, aumentar a acurácia dos valores imputados.

Referências Bibliográficas

A. Fuller, W. and Kim, J.-k. (2005). Hot deck imputation for the response model. *Surv. Methodol.*, 31. 10

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216. 10

Allison, P. (1999). Multiple imputation for missing data: A cautionary tale. 28. 24, 25

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1383–1394, New York, NY, USA. ACM. 15, 16, 42

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805. 1, 14

Austin, P. C. and Escobar, M. D. (2005). Bayesian modeling of missing data in clinical research. *Computational Statistics Data Analysis*, 49(3):821 – 836. 10

Batista, G. (2003). *Pré-Processamento de Dados em Aprendizado de Máquina Supervisionado*. PhD thesis, USP. 14

Batista, G. and Monard, M.-C. (2001). A study of k-nearest neighbour as a model-based method to treat missing data. In *Proceedings of the ASAI-01 - Argentine Symposium on Artificial Intelligence*. 9

Batista, G. and Monard, M.-C. (2003a). An analysis of four missing data treatment methods for supervised learning. 17. 14

Batista, G. and Monard, M.-C. (2003b). Um estudo sobre a efetividade do método de imputação baseado no algoritmo k-vizinhos mais próximos. In *Proceedings of IV Workshop on Advances Trends in AI for Problem Solving*. 14

Brown, M. L. and Kros, J. F. (2003). Data mining. chapter The Impact of Missing Data on Data Mining, pages 174–198. IGI Global, Hershey, PA, USA. 8

C Yuan, Y. (2005). Multiple imputation for missing data: Concepts and new development. 9

Castaneda, R., Ferlin, C., Goldschmidt, R., de Abreu Soares, J., de Carvalho, L. A. V., and Choren, R. (2008). Aprimorando processos de imputaÇÃo multivariada de dados com workflows. In *Proceedings of the 23rd Brazilian Symposium on Databases*, SBBD '08, pages 238–252, Porto Alegre, Brazil, Brazil. Sociedade Brasileira de Computação. 24, 25, 33, 36

Cheema, J. R. (2014). A review of missing data handling methods in education research. *Review of Educational Research*, 84(4):487–508. 23, 25

Chen, M. ., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209. 2, 4, 5, 14, 15

Davidson, S. B. and Freire, J. (2008). Provenance and scientific workflows: Challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1345–1350, New York, NY, USA. ACM. 3, 21, 22

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528 – 540. 2, 3, 21

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience, New York, NY, USA. 17, 18, 19, 21

Elgamal, T., Yabandeh, M., Aboulnaga, A., Mustafa, W., and Hefeeda, M. (2015). spca: Scalable principal component analysis for big data on distributed platforms. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 79–91, New York, NY, USA. ACM. 26

Enders, C. (2010). *Applied Missing Data Analysis*. Methodology in the social sciences. Guilford Publications. 2, 8, 9

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34. 2, 4

Ferlin, C. (2008). *Imputação Multivariada: Uma abordagem em cascata*. PhD thesis, COPPE/UFRJ. 24, 25

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science Engineering*, 10(3):11–21. 22

García-Laencina, P. J., Abreu, P. H., Abreu, M. H., and Afonoso, N. (2015). Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values. *Computers in Biology and Medicine*, 59:125 – 133. 23, 25

García-Laencina, P. J., Sancho-Gómez, J.-L., Figueiras-Vidal, A. R., and Verleysen, M. (2009). K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing*, 72(7):1483 – 1493. Advances in Machine Learning and Computational Intelligence. 24, 25

Gelman, A. and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press. 2

Goldschmidt, R., Passos, E., and Bezerra, E. (2015). *Data Mining: Conceitos, técnicas, algoritmos, orientações e aplicações*. 5, 6

Gopalani, S. and Arora, R. (2015). Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means. *International Journal of Computer Applications*, 113(1):8–11. , 25, 26

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques, Third Edition*. Morgan Kaufmann, Waltham, Mass., 3 edition edition. 00000. , 2, 4, 5, 6, 7, 8, 12, 13, 16, 17, 29

Hand, D. J., Smyth, P., and Mannila, H. (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA, USA. 5

Hruschka, E. R., Hruschka, E. R., and Ebecken, N. F. F. (2005). *Missing Values Imputation for a Clustering Genetic Algorithm*, pages 245–254. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

Huang, C.-C. and Lee, H.-M. (2004). A grey-based nearest neighbor approach for missing attribute value prediction. *Applied Intelligence*, 20(3):239–252. 24, 25

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., and Shahabi, C. (2014). Big data and its technical challenges. *Commun. ACM*, 57(7):86–94. 2, 4, 14, 15

Jonsson, P. and Wohlin, C. (2004). An evaluation of k-nearest neighbour imputation using likert data. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 108–118. 14

Jordanov, I., Petrov, N., and Petrozziello, A. (2018). Classifiers accuracy improvement based on missing data imputation. *Journal of Artificial Intelligence and Soft Computing Research*, 8(1):31 – 48. 9

Khan, N., Yaqoob, I., Hashem, I., Inayat, Z., Kamaleldin, W., Alam, M., Shiraz, M., and Gani, A. (2014). Big data: Survey, technologies, opportunities, and challenges. 2014:18. , 1

Lee, D. and Damji, J. (2016). Apache spark key terms, explained. [Online; accessed January 2, 2019]. , 16, 17

Li, D., Deogun, J., Spaulding, W., and Shuart, B. (2004). *Towards Missing Data Imputation: A Study of Fuzzy K-means Clustering Method*, pages 573–579. Springer Berlin Heidelberg, Berlin, Heidelberg. 24, 25

Little, R. J. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, 2 edition edition. 2

Little, R. J. A. and Rubin, D. B. (1986). *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA. 2, 8

Luengo, J., García, S., and Herrera, F. (2012). On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and Information Systems*, 32(1):77–108. 23, 25

M. Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. 17:37–54. 2, 4

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press. 11

Magnani, M. (2004). Techniques for dealing with missing data in knowledge discovery tasks. 8

Maillo, J., Ramírez, S., Triguero, I., and Herrera, F. (2017). knn-is: An iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, 117(Supplement C):3 – 15. Volume, Variety and Velocity in Data Science. 26

McPhillips, T., Bowers, S., Zinn, D., and Ludäscher, B. (2009). Scientific workflow design for mere mortals. *Future Generation Computer Systems*, 25(5):541 – 551. 15, 21, 25

Medina, J. (2012). *Algoritmos genéticos e redes de Kohonen na complementação de dados ausentes*. PhD thesis, IME/UERJ. 23, 25

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2016a). Clustering - rdd-based api. [Online; accessed January 4, 2019]. 43

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2016b). Mllib: Machine learning in apache spark. *J. Mach. Learn. Res.*, 17(1):1235–1241. 16, 42

Nan, Y. and Gao, Y. (2018). Data from: A machine learning method to monitor china's aids epidemics with data from baidu trends. 33, 34, 36

Nanni, L., Lumini, A., and Brahnam, S. (2012). A classifier ensemble approach for the missing feature problem. *Artificial Intelligence in Medicine*, 55(1):37 – 50. 10, 26, 27, 36

Qu, Z., Yan, J., and Yin, S. (2016). Association-rules-based data imputation with spark. In *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 145–149. 25

Raghunathan, T. E. (2004). What do we do with missing data? some options for analysis of incomplete data. *Annual Review of Public Health*, 25(1):99–117. PMID: 15015914. 9

Rubin, D. B. (1988). An overview of multiple imputation. In *In Proceedings of the Survey Research Section, American Statistical Association*, pages 79–84. 9

Saar-Tsechansky, M. and Provost, F. (2007). Handling missing values when applying classification models. *J. Mach. Learn. Res.*, 8:1623–1657. 24, 25

Silva, L. O. and Zárate, L. E. (2014). A brief review of the main approaches for treatment of missing data. *Intell. Data Anal.*, 18(6):1177–1198. 23, 25

Smith, L. I. (2005). A Tutorial on Principal Component Analysis. 13

Soares, J. (2007). *Pré-processamento em Mineração de Dados: um Estudo Comparativo em Complementação*. PhD thesis, COPPE/UFRJ. , 2, 3, 6, 9, 10, 11, 12, 25, 27, 28, 29, 33, 36, 39, 40, 45, 46, 48

Talavera-Llames, R. L., Pérez-Chacón, R., Martínez-Ballesteros, M., Troncoso, A., and Martínez-Álvarez, F. (2016). *A Nearest Neighbours-Based Algorithm for Big Time Series Data Forecasting*, pages 174–185. Springer International Publishing, Cham. 26

Tavares, R., Castaneda Ribeiro, R., Ferlin, C., Goldschmidt, R., Carvalho, L., and Soares, J. (2018). Apoiando o processo de imputação com técnicas de aprendizado de máquina. In *Proceedings of the 33rd Brazilian Symposium on Databases*, SBBD '18. Sociedade Brasileira de Computação. 10, 23, 25, 48, 76

Tran, C. T., Zhang, M., Andreae, P., Xue, B., and Bui, L. T. (2018). Improving performance of classification on incomplete data using feature selection and clustering. *Applied Soft Computing*, 73:848 – 861. 10, 23, 25

Wohlrab, L. and Fürnkranz, J. (2011). A review and comparison of strategies for handling missing values in separate-and-conquer rule learning. *Journal of Intelligent Information Systems*, 36(1):73–98. 24, 25

Wu, Z., Li, Y., Plaza, A., Li, J., Xiao, F., and Wei, Z. (2016). Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(6):2270–2278. 26

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016a). Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65. 2, 15, 25

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016b). Apache spark architecture. [Online; accessed January 1, 2019]. , 16

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016c). Spark programming guide. [Online; accessed January 6, 2019]. 41

Zhang, C. and Ma, Y. (2012). *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated. , 2, 3, 17, 18, 19, 20, 21, 28

Zhang, P., Zhu, X., Tan, J., and Guo, L. (2010). Skif: A data imputation framework for concept drifting data streams. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1869–1872, New York, NY, USA. ACM. 24, 25

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition. 3, 16, 17, 18, 19, 20, 46