PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

SILVIA CRISTINA NUNES DAS DÔRES

UBERBAND: META-APRENDIZADO E OTIMIZAÇÃO BASEADA EM BANDIDOS MULTI-ARMADOS PARA SELEÇÃO EFICIENTE E EFETIVA DE PROCESSOS COMPLETOS

Porto Alegre
2019

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

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

UBERBAND: META-APRENDIZADO E OTIMIZAÇÃO BASEADA EM BANDIDOS MULTI-ARMADOS PARA SELEÇÃO EFICIENTE E EFETIVA DE PROCESSOS COMPLETOS

SILVIA CRISTINA NUNES DAS DÔRES

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

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz
Co-Orientador: Prof. Dr. Carlos Manuel Milheiro de Oliveira Pinto Soares

Porto Alegre
2019

Ficha Catalográfica

D999u Dôres, Silvia Cristina Nunes das

Uberband Meta-Aprendizado e Otimização Baseada em Bandidos Multi-Armados para Seleção Eficiente e Efetiva de Processos Completos / Silvia Cristina Nunes das Dôres . – 2019.

202.

Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz.

Co-orientador: Prof. Dr. Carlos Manuel Milheiro de Oliveira Pinto Soares.

1. aprendizado de máquina automático. 2. meta-aprendizado. 3. otimização baseada em bandidos multi-armados. 4. seleção de processo completo. I. Ruiz, Duncan Dubugras Alcoba. II. Soares, Carlos Manuel Milheiro de Oliveira Pinto. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Silvia Cristina Nunes das Dôres

Uberband: Meta-Aprendizado e Otimização Baseada em Bandidos Multi-Armados para Seleção Eficiente e Efetiva de Processos Completos

Tese apresentada como requisito parcial para obtenção do grau de Doutor em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 29 de Abril de 2019.

BANCA EXAMINADORA:

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho (ICMC/USP)

Profa. Dr. Ricardo Bastos Cavalcante Prudêncio (CIn/UFPE)

Prof. Dr. Felipe Rech Meneguzzi (PPGCC/PUCRS)

Prof. Dr. Carlos Manuel Milheiro de Oliveira Pinto Soares (FEUP - Co-Orientador)

Prof. Dr. Duncan Dubugras Alcoba Ruiz (PPGCC/PUCRS - Orientador)

DEDICATÓRIA

À Leandro, Samuel, Eva e Dora.

“A determinação das pessoas é como água, ninguém pode parar. Rodeia montanhas, cai em cascatas. Mas chega aonde tem que chegar.”
(Andoni Aduriz)

AGRADECIMENTOS

Ao meu amado marido Leandro, que ressignificou na minha vida as definições de companheirismo, confiança e amor. Esse trabalho leva o meu nome, mas não sem dúvida nenhuma ele só existe por causa do teu apoio irrestrito. O lar, a família e os laços que criamos nesses últimos anos são meu bem mais precioso. Nenhum agradecimento é suficiente. Eu te amo.

À Eva e à Dora, os seres de maior sensibilidade que existem no mundo, companheiras fiéis de todos os momentos. A vida não seria a mesma sem vocês.

Ao meu bebê Samuel, que "surgiu"aos 45 do segundo tempo deste trabalho. Obrigada por dividir os seus primeiros meses de "existência"com os últimos meses que precisei para terminar a tese, não existe no mundo bebê mais compreensivo.

À minha querida família paraense, Pai, irmã, tios, tias e primos, pelo amor e torcida incondicional. Obrigada por me incentivarem a seguir meus sonhos e por serem a casa para onde eu sempre poderei voltar. E me perdoem por eu ter ido tão poucas vezes a Belém nesse período.

Ao meu orientador, professor Duncan Ruiz, por acreditar no meu trabalho, desde 2013. Ouvi várias vezes durante esses anos, quando brincava que minhas sessões de orientação eram mais curtas que a dos meus colegas, que ele tratava cada aluno conforme necessário. Para mim, coube exatamente o que eu precisava: confiança e espaço para desenvolver as minhas ideias, alguns puxões de orelha diante das ideias muito mirabolantes, e uma porta sempre aberta para resolver qualquer problema. Obrigada por me proporcionar o melhor ambiente de trabalho possível.

Ao meu co-orientador, Carlos Soares, que esteve presente no meu doutorado desde do primeiro artigo que li sobre Meta-Aprendizado, para escrever meu plano de doutoramento. Nessa época, nunca imaginei que eu viria a conhecer uma das maiores referências da área, e muito menos ser orientada por ele. Obrigada por ter me recebido de braços abertos no Porto, por cada hora dedicada ao meu trabalho, por cada ensinamento.

Ao Rodrigo Barros, por me apresentar ao mundo do Aprendizado de Máquina e me fazer não só entender os conceitos, mas gostar da área a ponto de querer fazer doutorado nisso.

Aos colegas do GPIN, que tornaram a caminhada mais alegre, seja nos nossos momentos de confraternização ou nos debates calorosos sobre os mais diversos temas. Mantenham as tradições!

Aos amigos de Porto Alegre, Porto e Belém, por sempre me darem a certeza que eu teria com quem contar.

À HP, Dell e CAPES, pelo suporte financeiro que tornaram possível a realização deste trabalho.

UBERBAND: META-APRENDIZADO E OTIMIZAÇÃO BASEADA EM BANDIDOS MULTI-ARMADOS PARA SELEÇÃO EFICIENTE E EFETIVA DE PROCESSOS COMPLETOS

RESUMO

Na medida em que tecnologias para gerenciamento e armazenamento de dados se tornam amplamente disponíveis, torna-se um desafio fornecer aos usuários sistemas eficazes de análise e compreensão desses dados. Descoberta de Conhecimento em Bases de Dados (DCBD) é o processo não-trivial de extração de padrões interessantes, válidos e úteis a partir de dados. Este processo inclui desde a seleção de dados, até a interpretação dos padrões identificados. Especialmente para usuários não-especialistas, a definição e gestão de um processo de DCBD são atividades complexas, pois é exigido o conhecimento sobre como escolher as operadores adequados dentre a gama disponível para cada etapa, como configurá-los e como interpretar sua saída. Seleção Automática de Processos Completos (SPC) objetiva auxiliar usuários de DCBD na tarefa onerosa de escolher o processo completo de operadores, que inclui métodos de pré-processamento, algoritmos de aprendizado de máquina e suas configurações de hiper-parâmetros mais adequados a um determinado problema. Embora diversas soluções já existam para esta tarefa, tais soluções são limitadas do ponto de vista da avaliação do processo completo i) algumas soluções não realizam experimentação do processo e se baseiam apenas em estimativas de desempenho de problemas similares, o que pode levar a recomendações não-precisas e ii) as demais soluções avaliam os processos completos repetidas vezes sobre o conjunto de treinamento inteiro até encontrar a melhor opção. Estas últimas soluções geralmente obtém resultados mais precisos, porém, se tornam computacionalmente custosas em termo de tempo, à medida em que os conjuntos de dados aumentam e novos algoritmos são desenvolvidos. Neste sentido, esta pesquisa propõe e analisa um novo algoritmo para SPC, denominado Uberband, que combina Meta-Aprendizado, para a estimar a probabilidade de

amostragem de operadores, e otimização baseada em bandidos multi-armados, para realização de alocação adaptativa de instâncias do conjunto de treinamento durante o processo de otimização. Resultados da análise experimental comparativa com soluções estado-da-arte em SPC, indicaram que Uberband proporciona uma SPC com desempenho similar e em um tempo expressivamente menor do que as soluções atuais.

Palavras-Chave: aprendizado de máquina automático, meta-aprendizado, otimização baseada em bandidos multi-armados, seleção de processo completo.

UBERBAND: COMBINING METALEARNING AND BANDIT-BASED OPTIMIZATION FOR EFFICIENT AND EFFECTIVE WORKFLOW SELECTION

ABSTRACT

Since data management and storage technologies become widely available, it becomes a challenge to provide users with effective systems for analyzing and understanding these data. Knowledge Discovery on Databases (KDD) is the non-trivial process of extracting interesting, valid, and useful patterns from data. This process ranges from data selection to interpretation of the identified patterns. Especially for non-expert users, the definition and management of KDD process are complex activities, since it requires knowledge on how to choose the appropriate operators from the available range, how to configure them and how to interpret their output. Automatic Workflow Selection (AWS) aims to assist users of KDD in the onerous task of choosing the workflow, which includes preprocessing methods, machine learning algorithms and their hyper-parameter configurations, more suitable for a given problem. Although several solutions already exist for this task, such solutions are limited from the point of view of experimental evaluation of candidate workflows: i) some solutions do not perform workflow experimentation and are based on performance predictions in similar problems, which can lead to non-precise recommendations and ii) other solutions evaluate the workflows configurations over the entire training set until the best option is found. These latter solutions usually get more accurate results, however, they become computationally time-consuming as the datasets increase and new algorithms are developed. In this sense, this research proposes and investigates a new algorithm for AWS, named Uberband, that combines metalearning and multi-armed bandit optimization to perform adaptive allocation of the training data set during the optimization process. Results of the comparative experimental analysis with state-of-the-art solutions in AWS indicated that Uberband provides a AWS with good performance and in a significantly speedup over the current solutions.

Keywords: automatic workflow selection, bandit-based optimization, knowledge discovery in databases, meta-learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

ACO – *Ant Colony Optimization*

AG – Algoritmo Genético

AM – Aprendizado de Máquina

AUTOAM – Aprendizado de Máquina Automático

AUC – *Area Under ROC Curve*

BA-FMS – *Bat Algorithm for Full Model Selection*

CA – Configuração de Algoritmos

CARS – *Collaborative Analytics Recommender System*

CASH – *Combined Algorithm Selection and Hiperparameter Optimization*

DCBD – Descoberta de Conhecimento em Base de Dados

DMER – *Data Mining Experiment Repository*

DMOP – *Data Mining OPtimization ontology*

DMWF – *Data Mining Ontology for Workflows*

FF – *Fast-Forward*

GAD – Gráfico Acíclico Direcionado

GAPSO-FMS – *Genetic Algorithm and Particle Swarm Optimization for Full Model Selection*

HMDA – *Hybrid Data Mining Assistant*

HTN – *Hierarchical Task Network*

IDA – *Intelligent Discovery Assistant*

IDEA – *Intelligent Discovery Eletronic Assistant*

K-NN – *k- Nearest Neighbours*

KDDVM – *Knowledge Discovery in Databases Virtual Mart*

LOO – *Leave-One-Out*

MA – Meta-Aprendizado

MCPS – *Multicomponent Predictive System*

MD – Mineração de Dados

METAL – *Meta-Learning Assistant for Providing User Support in Machine Learning Mining*

MH – Meta-Heurística

NEXT – *Next-generation Experiment Toolbox*

NSGA2FMS – *NSGA-II for Full Model Selection*

OB – Otimização Bayesiana

OWL – *Ontology Web Language*

PA – Planejamento Automático

PDDL – *Planning Domain Definition Language*

PG – Programação Genética

PMD – *Planning for Data Mining*

PMML – *Predictive Model Markup Language*

PSMS – *Particle Swarm Model Selection*

PSO – *Particle Swarm Optimization*

RBC – Raciocínio Baseado em Casos

SA – Seleção de Algoritmos

SC – Seleção de Características

SMAC – *Sequential Model Algorithm Configuration*

SPC – Seleção de Processo Completo

SVM – *Support Vector Machine*

TPE – *Tree Parzen Estimator*

TPOT – *Tree Based Pipeline Optimization*

LISTA DE SÍMBOLOS

SUMÁRIO

1. INTRODUÇÃO

O rápido crescimento da capacidade de gerar e coletar dados possibilitou que muitas empresas e instituições armazenem enormes volumes de dados. Porém, a obtenção de conhecimento a partir desses dados ainda é um grande desafio. Descoberta de Conhecimento em Bancos de Dados (DCBD) é definida como o processo não-trivial de identificar padrões válidos, novos, potencialmente úteis e compreensíveis em Bancos de Dados [Fayyad et al., 1996a]. O processo de DCBD é um processo altamente complexo, iterativo e interativo, com uma natureza orientada por objetivos, dependente de domínio [Diamantini et al., 2009] e tipicamente envolve muitos operadores, ou seja, métodos de pré-processamento e algoritmos de Aprendizado de Máquina (AM).

A grande variedade de operadores disponíveis em cada fase do processo de DCBD faz com que a seleção dos operadores mais adequados a um determinado problema possa acarretar na realização de diversos experimentos, muitas vezes conduzidos por tentativa e erro, demandando muito esforço e tempo, e sem garantia de sucesso no final [Vilalta e Drissi, 2002] [de Sá et al., 2017]. Os primeiros sistemas de descoberta de conhecimento, como Weka [Witten e Frank, 2016], KNIME [Berthold et al., 2009] e as primeiras versões do RapidMiner [RapidMiner, 2018], contêm implementações de vários operadores de DCBD, mas fornecem apenas suporte limitado em seu uso, porque a seleção, combinação e configuração, isto é, a definição dos hiper-parâmetros, desses operadores são feitas pelo usuário. Como a demanda por algoritmos de AM cresce mais rapidamente que a oferta de especialistas na área, existe uma necessidade crescente de tecnologias para selecionar, combinar e configurar automaticamente algoritmos para a tarefa em questão.

Aprendizado de Máquina Automático (AutoAM) é a sub-área de AM que trata desse problema. Esta área inclui diferentes esforços de pesquisa sobre automação de AM, como seleção e configuração de algoritmos [Guyon et al., 2015]. Os sistemas de AutoAM para Seleção automática de Processos Completos (SPC) integram essas diversas tarefas, ou seja, projetam e recomendam uma combinação otimizada de operadores, incluindo a configuração de hiper-parâmetros, para tarefas de aprendizado específicas, com dependência mínima do conhecimento do usuário.

Nos últimos anos pesquisadores buscaram melhorar a experiência de usuários de DCBD, a partir do desenvolvimento de diversas soluções para SPC. O objetivo dessas soluções é encontrar rapidamente, dentro de um limite de recursos pré-estabelecido, uma combinação de operadores de DCBD, e seus respectivos hiper-parâmetros, que maximizam uma medida de desempenho no problema de aprendizado de máquina e sobre o conjunto de dados fornecidos. Essas soluções envolvem a aplicação de diversas abordagens baseadas em Inteligência Artificial (IA), tais como Raciocínio Baseado em Casos

(RBC) [Charest et al., 2008], Planejamento Automático (PA) [Fernández et al., 2010], Meta-heurística (MH) [Sun, 2014] e Otimização Bayesiana (OB) [Thornton et al., 2013].

Embora Raciocínio Baseado em Casos esteja apto a oferecer uma seleção precisa do processo completo adequado a um determinado problema, este resultado está intrinsecamente relacionado com a quantidade e qualidade de casos de sucesso, presentes na base de casos. Logo, se não houver na base casos que possam ser utilizados para guiar a seleção de processo completo para um dado problema, ou os casos existentes são sejam suficientes para realizar uma seleção precisa, a qualidade do processo recomendado ficará comprometida [Sun et al., 2013]. Alternativamente, métodos baseados em otimização como Meta-Heurísticas e Otimização Bayesiana têm como vantagem a seleção de processos completos ótimos, ou próximos de ótimos, avaliados diretamente nos dados de entrada. Em contrapartida, por se tratarem de uma busca que avalia todo o amplo espaço de possibilidades, estas abordagens podem vir a serem mais custosas computacionalmente quanto maior o conjunto de dados e o número de operadores de DCBD a serem combinados.

Na tentativa de contornar o desafio de explorar um grande espaço de busca aleatoriamente, [Nguyen et al., 2014, Feurer et al., 2015a, Chong et al., 2013] combinam Planejamento Automático, Otimização Bayesiana e Meta-Heurística, respectivamente, com um estratégia de Meta-Aprendizado, de maneira a inicializar a estratégia de busca com um conjunto de processos completos que obtiveram bom desempenho preditivo em problemas anteriores similares ao problema atual.

Embora a restrição do espaço de busca proporcione vantagens, estas soluções ainda recaem em uma limitação fundamental: o grande tempo de execução necessário para testar um processo completo candidato no conjunto de dados de treinamento, levando em consideração que muitas possibilidades de processo são testadas até que uma boa combinação seja encontrada. Dado que a aplicação DCBD vem incorporando problemas cada vez mais complexos, bem como novos algoritmos de AM continuam sendo desenvolvidos, a questão do tempo de busca se tornará mais severa no futuro. Assim, surge a necessidade de se utilizar métodos que otimizem a avaliação dos processos candidatos, tornando o processo de SPC mais eficiente.

Neste contexto, a partir da análise de trabalhos recentes aplicados a tarefa de otimização de hiper-parâmetros de algoritmos, verificou-se a possibilidade de explorar resultados intermediários de dados treinamento. Especificamente, o trabalho de Li *et al.* [Li et al., 2017] apresenta uma solução para identificar a melhor configuração de hiper-parâmetros utilizando um algoritmo de otimização baseado em bandidos multi-armados. Este algoritmo, denominado Hyperband, tem como foco a otimização da avaliação de configurações, a partir da alocação adaptativa de recursos para configurações promissoras, enquanto as menos promissoras são eliminadas. Em seus resultados comparativos, Hyperband superou soluções baseadas em Otimização Bayesiana, até então estado-da-arte na tarefa de otimização de hiper-parâmetros, em diversos experimentos. Porém, ao ser

avaliado para o problema de SPC, embora tenha convergido mais rapidamente, obteve um desempenho inferior às demais abordagens.

Uma vez que Hyperband foi desenvolvido para otimizar a configurações de algoritmo, o espaço de busca é restrito aos hiper-parâmetros do algoritmo em questão e uma única escolha deve ser otimizada: a combinação desses hiper-parâmetros. Porém, configuração de algoritmos é apenas uma das tarefas da SPC, que também inclui a seleção desses algoritmos. Neste contexto, quando não se tem conhecimento prévio que direcione a melhores possibilidades de escolha, o algoritmo de otimização deve trabalhar a partir da seleção aleatória de processos completos, até determinar as regiões de melhor desempenho, o que é computacionalmente custoso, mesmo levando em consideração a alocação de adaptativa de recursos.

1.1 Objetivo da Tese

A partir da limitações existentes nas soluções atuais, a seguinte questão de pesquisa pode ser estabelecida: "é possível melhorar o desempenho e o tempo de execução da Seleção de Processos Completos a partir do conhecimento proveniente de problemas anteriores e sem a necessidade de experimentar todas as soluções candidatas no conjunto completo de dados de treinamento?".

Com base nesta questão, esta tese tem como objetivo principal propor e avaliar um algoritmo para Seleção de Processos Completos que utiliza Meta-Aprendizado e alocação adaptativa de recursos, a partir da utilização de métodos baseados em bandidos multi-armados.

Como resultado, espera-se que, do ponto de vista do usuário, dado um conjunto de dados referente a um problema de DCBD, o algoritmo proposto retorne o processo completo, isto é, uma combinação de operadores (métodos de pré-processamento e algoritmo de aprendizado de máquina) e suas configurações de hiper-parâmetros, que obteve o melhor desempenho preditivo em um determinado espaço de busca. E do ponto de vista científico, que este resultado possua um desempenho preditivo igual ou superior às soluções já existentes, com redução significativa do tempo de execução.

1.2 Principais Contribuições da Tese

A principal contribuição desta tese é um novo algoritmo para seleção de processo completo, denominado Uberband (Capítulo 5), que combina Meta-Aprendizado como forma de favorecer amostragem de operadores com maior desempenho estimado para o problema em questão, com uma estratégia de avaliação de processos que conduz experimentos com

pequenas porções dos dados de entrada para eliminar progressivamente processos não promissores, enquanto aloca maiores recursos de dados para ajustar configurações de melhor desempenho.

Como contribuições específicas acerca de SPC, temos:

- Revisão Sistemática da Literatura sobre Seleção de Processos Completos: apresentada no Capítulo 4, onde os trabalhos relacionados foram detalhados, organizados em uma taxonomia e discutidos em termos de suas vantagens e desvantagens;

- Arquitetura Genérica de Componentes de uma Soluções de Seleção de Processos Completos: A análise dos trabalhos relacionados também permitiu a elaboração de uma arquitetura geral que categoriza senão todas, mas boa parte das soluções existentes, tanto do ponto de vista técnico como de contribuições científicas. Esta arquitetura é apresentada no Capítulo 3;

- Estratégia de Meta-Aprendizado para Predição e Desempenho de Operadores: embora esta estratégia seja utilizada neste trabalho como entrada para o otimizador, ela demonstrou ser capaz de selecionar combinações de operadores com qualidade, o que indica que pode ser utilizada individualmente para esta tarefa. A avaliação individual da estratégia de MA é apresentada no Capítulo 6;

- Avaliação Empírica em grande volume de dados: O Capítulo 7 apresenta a avaliação da abordagem proposta em 60 conjuntos de dados no repositório OpenML [Vanschoren et al., 2014].

1.3 Metodologia

A Figura 1.1 apresenta o desenho da pesquisa apresentada nesta Tese, incluindo as fases da pesquisa e métodos que são aplicados para concepção desta solução.

- **FASE 1:** nesta fase é realizado o estudo de base teórica da área. Fazem parte da base teórica o estudo do processo de Descoberta de Conhecimento em Base de Dados, Aprendizado de Máquina, que inclui AM Supervisionado, AM automático e Meta-Aprendizado, bem como os métodos de Otimização, em especial a otimização baseada em bandidos multi-armados (Capítulos 2, 3). Durante este processo é realizada a Revisão Sistemática da Literatura (Capítulo 4) com o objetivo de identificar os trabalhos existentes em Seleção de Processo Completo, bem como as possibilidades de pesquisa na área.

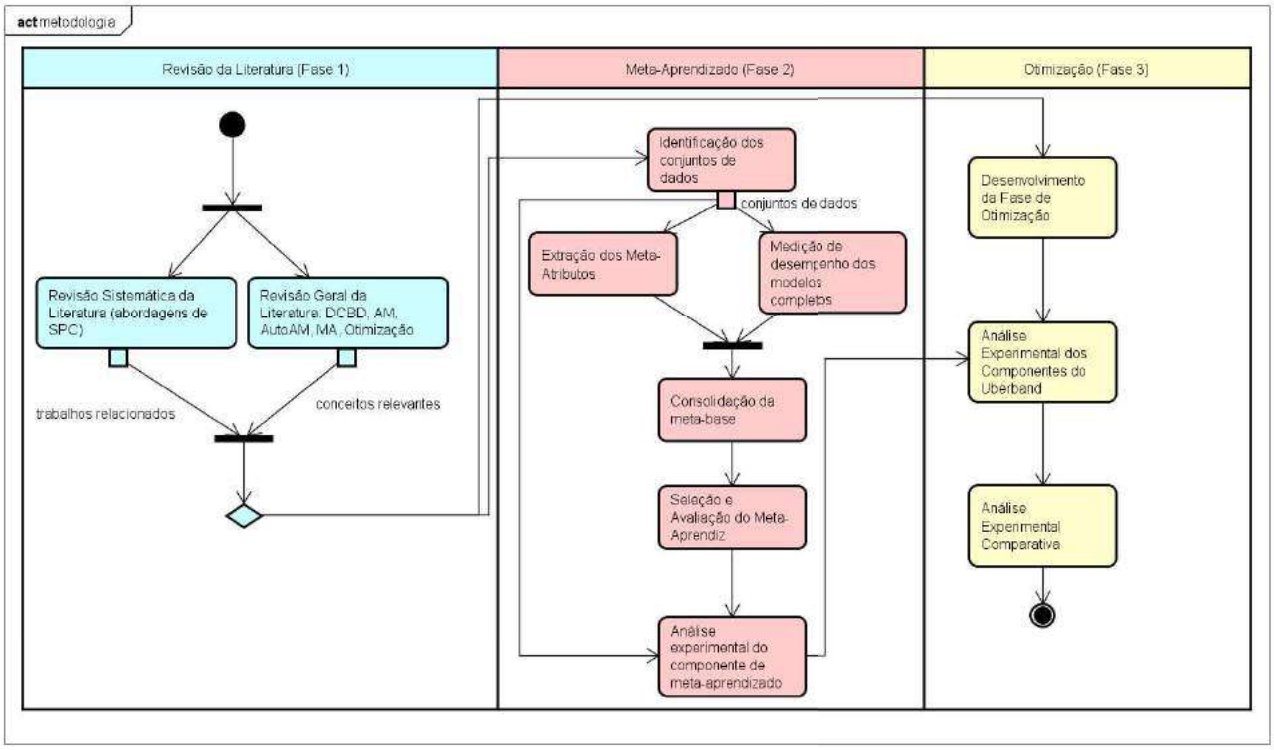

Figura 1.1 – Desenho da Pesquisa

- **FASE 2:** nesta fase foi desenvolvida e avaliada a abordagem de Meta-Aprendizado (Capítulo 4). Esta fase incluiu: a identificação dos conjuntos de dados (representativos de problemas de DCBD reais) em repositórios de dados públicos, a definição dos meta-atributos (características dos dados) para caracterizar as etapas do processo de DCBD e a medição do desempenho do conjunto de processos completos a serem avaliados, sendo que estas duas últimas fases constituem os dados a serem registrados na meta-base de conhecimento. Além disso, foi selecionado experimentalmente o meta-aprendiz (algoritmo de AM responsável por construir o modelo de predição de desempenho dos processos completos com base na meta-base de conhecimento). Por fim, é feita a análise experimental do componente de MA.

- **FASE 3:** nesta fase é desenvolvido e avaliado o algoritmo para seleção de processo completo. Por fim, é realizada uma análise comparativa entre a solução proposta e os trabalhos relacionados existentes mais relevantes, identificados na revisão sistemática da literatura (FASE 1).

1.4 Organização da Tese

Esta tese está estruturada em 8 capítulos. O restante do volume está organizado da seguinte maneira:

- O Capítulo 2 apresenta o processo de Descoberta de Conhecimento em Base de Dados e a área de Aprendizado de Máquina, com foco em Aprendizado de Máquina Supervisionado;
- O Capítulo 3 apresenta os conceitos relacionados à Aprendizado de Máquina Automático, incluindo, Seleção de Algoritmos, com foco em Meta-Aprendizado, Configuração de Algoritmos, com foco em Bandidos Multi-Armados e Seleção de Processo Completo;
- O Capítulo 4 apresenta os trabalhos relacionados, em forma de uma Revisão Sistemática da Literatura sobre Seleção de Processo Completo;
- O Capítulo 5 apresenta a visão geral da solução desenvolvida nesta pesquisa;
- O Capítulo 6 apresenta o detalhamento e a avaliação do componente de MA;
- O Capítulo 7 apresenta o detalhamento e a avaliação do processo de otimização para SPC, bem como a avaliação comparativa da solução;
- Por fim, o Capítulo 8 apresenta as considerações finais, avaliando as contribuições da pesquisa, limitações e possibilidades de trabalhos futuros.

2. DESCOBERTA DE CONHECIMENTO EM BASE DE DADOS E APRENDIZADO DE MÁQUINA

Este capítulo tem como objetivo apresentar a primeira parte da base teórica para a pesquisa proposta neste trabalho e está organizado da seguinte maneira: na Seção 2.1 é apresentada uma visão geral do processo de Descoberta de Conhecimento em Bases de Dados e na Seção 2.2 é apresentada a área de Aprendizado de Máquina, com ênfase em Aprendizado de Máquina Supervisionado.

2.1 Descoberta de Conhecimento em Bases de Dados e Mineração de Dados

Descoberta de Conhecimento em Bases de Dados (DCBD, do inglês - *Knowledge Discovery in Databases*) é uma área que estuda como extrair informações não-triviais de grandes quantidades de dados com o objetivo de descobrir padrões desconhecidos, tendências inesperadas ou outras relações presentes nesses dados [Fayyad et al., 1996b]. O processo de Descoberta de Conhecimento em Bases de Dados é ilustrado na Figura 2.1.

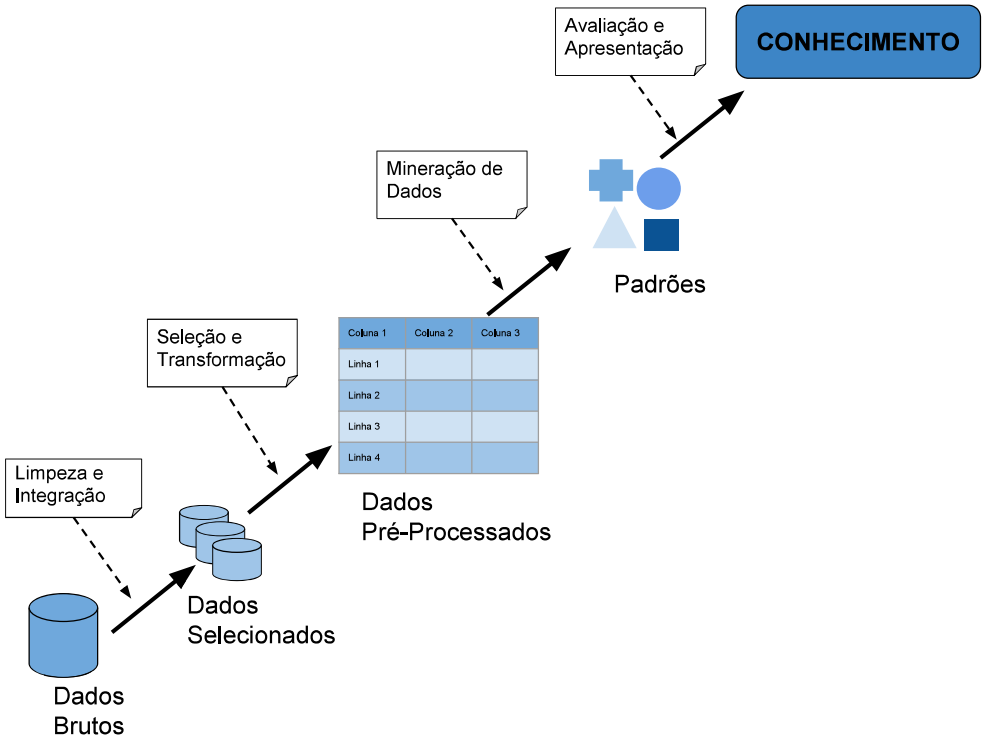

Figura 2.1 – Processo de Descoberta de Conhecimento em Bases de Dados [Adaptada de [Fayyad et al., 1996b]]

De acordo com [Fayyad et al., 1996b], o processo de DCBD inclui as etapas de:

- **Limpeza dos Dados:** remover o ruído e as inconsistências;
- **Integração dos Dados:** combinar múltiplas fontes de dados;
- **Seleção dos Dados:** selecionar dados relevantes para a análise a partir da base de dados;
- **Transformação de Dados:** transformar e consolidar dados de uma forma apropriada para mineração através da realização de operações de sumarização e agregação;
- **Mineração de Dados:** aplicar métodos inteligentes para extração de padrões;
- **Avaliação de Padrões:** identificar padrões verdadeiramente interessantes de representação do conhecimento, com base em medidas de interesse, como suporte e confiança;
- **Apresentação do Conhecimento:** apresentar o conhecimento minerado aos usuários, através de técnicas de visualização e representação de conhecimento.

Dentre essas etapas, Mineração de Dados (MD) destaca-se por ser onde efetivamente os padrões úteis podem ser descobertos, ou "minerados"[Tan et al., 2005]. De fato, muitos trabalhos utilizam o termo Mineração de Dados como sinônimo de Descoberta de Conhecimento em Base de Dados, tamanha a importância desta etapa [Han et al., 2011] [Witten e Frank, 2016].

Mineração de Dados é uma área interdisciplinar, que inclui Estatística, Técnicas de Modelagem, Banco de Dados, e teorias de aprendizado provenientes de Inteligência Artificial, como Reconhecimento de Padrões e Aprendizado de Máquina [Tan et al., 2005]. Seu principal objetivo é a extração de conhecimento inovador, ou seja, ainda não conhecido e que tenha valor para o domínio em que é aplicado [Faceli et al., 2011]. Neste sentido, técnicas de Aprendizado de Máquina (AM) se destacam neste processo, e aplicações utilizando AM são continuamente propostas, tais como, modelos para análise de crédito, modelos para diagnóstico médico.

2.2 Aprendizado de Máquina

Aprendizado é qualquer mudança em um sistema que melhore o seu desempenho na segunda vez que ele repetir a mesma tarefa ou outra tarefa tirada da mesma população [Simon, 1981]. A partir dessa definição, Mitchell [Mitchell, 1997] define Aprendizado de Máquina como sendo o estudo de algoritmos e sistemas que melhoram automaticamente com a experiência.

Diversas estratégias de aprendizado podem ser utilizadas para desenvolver algoritmos de AM. A mais estudada atualmente baseia-se no conceito de indução, segundo o qual é possível obter-se conclusões genéricas a partir de fatos ou observações particulares [Witten e Frank, 2016]. Esse tipo de inferência lógica caracteriza-se por extrapolar a informação contida nos dados a fim de modelar conceitos mais gerais.

O aprendizado indutivo tem sido tradicionalmente empregado para a concepção de abordagens de AM segundo três vertentes básicas: supervisionada, não-supervisionada e semi-supervisionada.

Na primeira, o objetivo é induzir descrições gerais de conceitos utilizando exemplos rotulados por um supervisor. O algoritmo de aprendizado utiliza essas informações para aprender a associação entre características dos exemplos e seus rótulos. Se os rótulos forem discretos, o problema é caracterizado como sendo de classificação, se forem contínuos, como regressão. Na segunda, a meta é descobrir padrões e regras gerais capazes de explicar as observações. Comumente, a tarefa de algoritmos não-supervisionados é analisar os exemplos e tentar identificar, caso haja, estruturas de grupos nos dados. Por fim, no aprendizado de máquina semi-supervisionado o algoritmo aprende tanto a partir de dados rotulados quanto de não rotulados.

A subseção a seguir apresenta uma visão geral de aprendizado de máquina supervisionado, que é o foco desta pesquisa.

2.2.1 Aprendizado de Máquina Supervisionado

Um conceito pode ser entendido como uma regra que particiona os objetos de um domínio, de acordo com a obediência (ou não) deles a ela [Utgoff, 1984]. A tarefa do AM supervisionado é induzir tais conceitos a partir de exemplos específicos dos mesmos.

No AM supervisionado, os objetos (também denominados exemplos ou instâncias) são rotulados por um supervisor ou professor, que detém o conhecimento do domínio e conhece a definição do conceito a ser aprendido. O supervisor fornece os objetos na forma de pares (entrada e saída desejada) [Haykin, 1998] e o algoritmo aprende, então, uma relação entre as características (também conhecidas como atributos ou variáveis) das entradas e das saídas, que seja consistente com os objetos considerados e que possa ser utilizado para predizer saídas corretas para entradas não vistas anteriormente.

No geral, os problemas tratados em AM supervisionado podem ser divididos em duas categorias: classificação e regressão, cuja principal diferença está no rótulo atribuído ao objeto, que pode ser categórico ou contínuo. As subseções a seguir apresentam uma visão geral sobre as tarefas de classificação e regressão.

Classificação

Classificação é a tarefa de atribuir objetos a uma dentre várias categorias pré-estabelecidas. É um problema que engloba diversas aplicações, tais como categorização de células cancerígenas, detecção de *spams* em *e-mails*, detecção de fraudes bancárias, dentre outros.

Em tarefas de classificação, dado um objeto $(\hat{x}, y)$, em que $\hat{x}$ é um vetor de m valores dos atributos preditivos $\hat{x} = (x^1, x^2, ..., x^m)$, o objetivo é predizer o rótulo do atributo alvo y, que assume valores discretos, e também pode ser denominado como **atributo classe**. Se as classes tiverem diferentes números de objetos, a classe com maior número é denominada classe majoritária, e a com menor número, minoritária. O processo de classificação se resume a construir um modelo (também denominado classificador ou hipótese), a partir dos dados observados, que seja capaz de classificar objetos com rótulos desconhecidos. Tal processo é formalmente definido por Tan *et al.* [Tan et al., 2005] como:

"Tarefa de aprendizado de uma **função alvo** f que mapeia cada conjunto de atributos $\hat{x}$ para um dos rótulos de classe pré-definidos y."

A Figura 2.2 ilustra o processo de classificação. Com base em um conjunto de dados de treinamento, que possui objetos rotulados, um algoritmo de AM é utilizado para construir um modelo (classificador). Após construído, este classificador é aplicado para predizer classes para objetos de um conjunto de testes, cujos rótulos de classe são desconhecidos.

Uma característica importante do algoritmo de AM nesta tarefa é construir modelos com boa capacidade de generalização, ou seja, modelos capazes de predizer, com precisão, rótulos de classe para objetos não apresentados anteriormente [Tan et al., 2005]. Exemplos de algoritmos de classificação incluem Árvores de Decisão, Redes Neurais Artificiais, Máquinas de Vetores de Suporte (SVM - do inglês, *Support Vector Machines*), classificadores Bayesianos, dentre outros, onde cada um se diferencia dos demais pelo viés indutivo empregado.

Avaliação de Classificadores

A avaliação do desempenho dos modelos de classificação é baseada na contagem do número de objetos do conjunto de teste que são corretamente e incorretamente preditos pelo modelo. A tabulação desta contagem é feita em uma tabela conhecida como **matriz de confusão**.

A Tabela 2.1 ilustra uma matriz de confusão para um problema de classificação de duas classes, **positiva** e **negativa**. As siglas VP, VN, FP e FN significam:

- Verdadeiro Positivo (VP): total de objetos da classe positiva corretamente classificados como positivos;

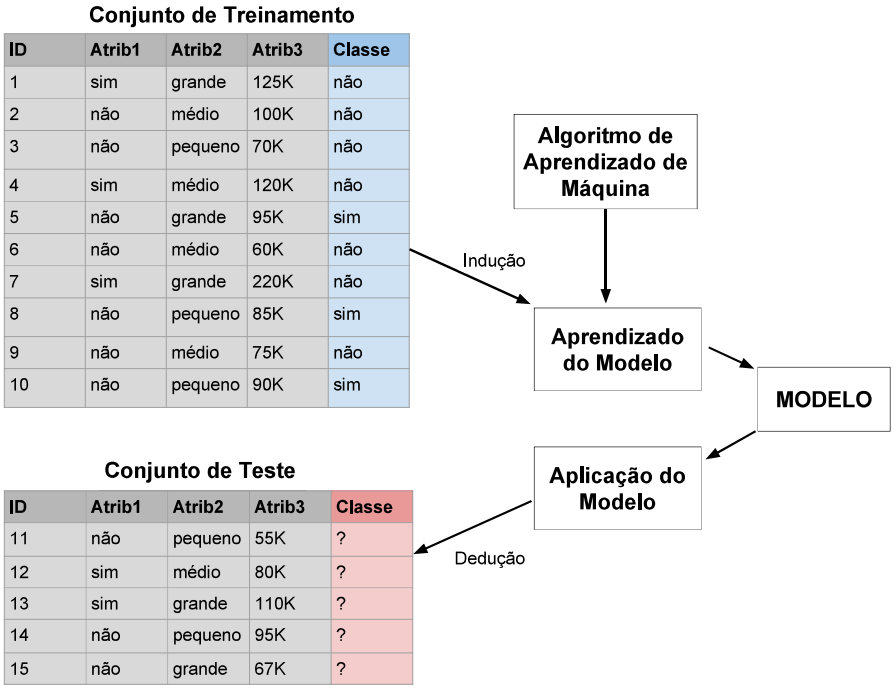

Figura 2.2 – Abordagem geral para construção de um modelo de classificação [Adaptada de [Tan et al., 2005]]

- Verdadeiro Negativo (VN): total de objetos da classe negativa corretamente classificados como negativos;
- Falso Positivo (FP): total de objetos da classe negativa incorretamente classificados como positivos;
- Falso Positivo (FN): total de objetos da classe positiva incorretamente classificados como negativos.

Tabela 2.1 – Matriz de Confusão para um problema de classificação binária

		Classe Predita	
		Positivo	Negativo
Classe Real	Positivo	**VP**	**FN**
	Negativo	**FP**	**VN**

Diversas medidas de desempenho podem ser derivadas partir da matriz de confusão. Trabalhos como [Ferri et al., 2009] buscam identificar as medidas mais comumente usadas na comunidade de AM. Porém, a escolha da medida mais adequada para um determinado contexto dependerá principalmente das características do problema sob análise e do objetivo final da avaliação [Rossi, 2013].

A seguir são descritas algumas das mais populares medidas de desempenho derivadas da matriz de confusão, de acordo com Faceli *et al.* [Faceli et al., 2011], dado um conjunto de teste com N objetos:

- **Acurácia:** também conhecida como taxa de acerto, é a soma dos elementos da diagonal principal da matriz, dividida pela soma de todos os elementos da matriz.

$$ac = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.1}$$

- **Precisão:** proporção de objetos positivos classificados corretamente, dentre todos os objetos preditos como positivo.

$$prec = \frac{VP}{VP + FP} \tag{2.2}$$

- **Sensibilidade:** também denominada revocação ou taxa de verdadeiros positivos (TVP), corresponde a taxa de acertos da classe positiva.

$$sens = rev = \frac{VP}{VP + FN} \tag{2.3}$$

- **Especificidade:** corresponde a taxa de acertos da classe negativa.

$$esp = \frac{VN}{VN + FP} \tag{2.4}$$

- **Medida-F:** uma vez que a precisão pode ser vista como uma medida de exatidão do modelo e a revocação como uma medida de completude, a análise dessas medidas é geralmente feita de maneira combinada. Esta combinação, conhecida como medida-F, nada mais é que é média harmônica ponderada de precisão e revocação.

$$F_m = \frac{(w + 1) * rev * prec}{rev + w * prec} \tag{2.5}$$

Adicionalmente, uma outra forma de se avaliar classificadores em problemas binários é com o uso de curvas ROC (do inglês, *Receiving Operating Characteristics*). O gráfico ROC é um gráfico bidimensional desenhado em um espaço denominado ROC , onde o eixo X representa a taxa de falsos positivos (TFP) e o eixo Y representa a taxa de verdadeiros positivos (TVP). A Figura 2.3 apresenta um exemplo do espaço ROC: a linha pontilhada diagonal representa classificadores com predições aleatórias. Assim, qualquer classificador abaixo desta linha é considerado pior que o aleatório. O ponto (1,0) representa as classificações perfeitas, onde todos as instâncias são classificadas corretamente; já o ponto (0,1) representa uma classificação completamente incorreta; o ponto (0,0) indica que todos os exemplos foram classificados como negativos; e o ponto (1,1) representa as classificações sempre positivas.

O procedimento usual para se comparar classificadores no espaço ROC é a geração de uma curva, denominada curva ROC. A linha em azul na Figura 2.3 apresenta um

exemplo de curva ROC para um classificador. A partir desta curva pode-se calcular a área abaixo da curva ROC (AUC, do inglês, *Area Under ROC Curve*). A medida AUC produz valores entre 0 e 1, sendo os valores próximos de 1 considerados melhores.

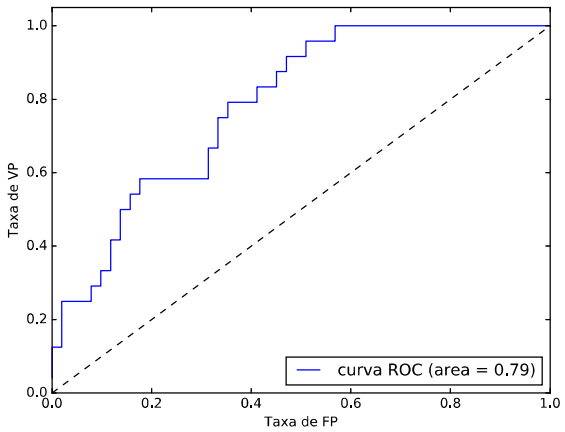

Figura 2.3 – Exemplo de Curva ROC

Regressão

Enquanto que na classificação o atributo alvo y é um conjunto de categorias, na tarefa de regressão este atributo é um valor contínuo.

Exemplos de aplicações de regressão incluem estimativa de custo de projetos de desenvolvimento de software, projeção de vendas totais de uma companhia, predição de índices no mercado de ações, dentre outras aplicações.

Em uma definição formal, de acordo com Tan *et al.* [Tan et al., 2005], dado um conjunto de dados D que contém N objetos,

$$D = (\hat{x}_i, y_i) | i = 1, 2, ..., N$$

onde cada $\hat{x}_i$ corresponde ao conjunto de atributos do i-ésimo objeto (também conhecido como **variáveis independentes** ou **atributos preditivos**, que podem assumir tantos valores discretos quanto contínuos) e y_i corresponde ao atributo alvo. A regressão pode ser descrita como:

"Tarefa de aprendizado de uma **função alvo** f que mapeia cada conjunto de atributos $\hat{x}$ em uma saída de valor contínuo y"

Dentre os exemplos de algoritmos de AM para regressão temos a Regressão Linear Simples, Regressão Linear Multivariada, Regressão dos Mínimos Quadrados, dentre outros.

Avaliação de Regressores

O objetivo da tarefa de regressão é encontrar uma função alvo, também chamada de **regressor**, que se ajuste aos dados de entrada com um **erro mínimo**. Para calcular este erro, uma gama de funções de erro é proposta na literatura, sendo que as principais, destacadas por Witten e Frank [Witten e Frank, 2016] são listadas na Tabela 2.2.

Dado um conjunto de testes com N objetos, onde $\mathbf{p} = (p_1, p_2, ..., p_N)$ são os valores preditos e $\mathbf{r} = (r_1, r_2, ..., r_N)$ são os valores reais dos objetos de teste, temos as seguintes medidas:

Tabela 2.2 – Medidas de Desempenho para Modelos de Regressão

Medida	Fórmula
Erro Quadrático Médio	$MSE = \frac{(p_1 - r_1)^2 + ... + (p_n - r_n)^2}{N}$
Raiz do Erro Quadrático Médio	$RMSE = \sqrt{\frac{(p_1 - r_1)^2 + ... + (p_n - r_n)^2}{N}}$
Erro Absoluto Médio	$MAE = \frac{\lvert p_1 - r_1)\rvert + ... + \lvert p_N - r_N\rvert}{n}$
Erro Quadrático Relativo	$RSE = \frac{(p_1 - r_1)^2 + ... + (p_N - r_N)^2}{(r_1 - \bar{r})^2 + ... + (r_N - \bar{r})^2}$, onde $\bar{r} = \frac{1}{N}\sum_{i=0}^{N} r_i$
Erro Absoluto Relativo	$RAE = \frac{\lvert p_1 - r_1)\rvert + ... + \lvert p_N - r_N\rvert}{\lvert p_1 - \bar{r}\rvert + ... + \lvert p_N - \bar{r}\rvert}$

O **erro quadrático médio** (MSE, do inglês - *Mean Squared Error*) é a medida principal e mais comumente usada, pois tende a ser a mais fácil de se manipular matematicamente. A partir desta medida, pode-se tirar a raiz quadrada para manter a mesma unidade dos valores preditos, ao que denomina-se **raiz do erro quadrático médio** (RMSE, do inglês - *Root Mean Squared Error*).

O **erro absoluto médio** (MAE, do inglês - *Mean Absolute Error*) é uma medida alternativa à MSE, que calcula uma média da magnitude dos erros individuais, sem considerar os seus sinais. Esta característica é relevante uma vez que o MSE tende a aumentar o efeito de *outliers* (valores discrepantes presentes nos dados), enquanto que o MAE não possui este efeito, por tratar os erros igualmente de acordo com sua magnitude.

Em alguns casos, o erro relativo pode ser mais relevante que o erro absoluto e, portanto, medidas adequadas devem ser empregadas. O **erro quadrático relativo** (RSE, do inglês - *Relative Squared Error*) é a razão do erro quadrático do modelo induzido pelo algoritmo de regressão e do erro quadrático de um preditor simples, onde tal preditor utiliza a média do atributo alvo do conjunto de treinamento para predizer o atributo alvo dos objetos de teste.

Por fim, o **erro absoluto relativo** (RAE, do inglês - *Relative Absolute Error*) é o erro absoluto total, com o mesmo tipo de normalização do RSE, ou seja, normalizado por um preditor simples.

Técnicas de Amostragem de Dados

Na avaliação experimental dos algoritmos, tanto de classificação quanto de regressão, é necessário levar em consideração que os objetos utilizados no treinamento do modelo devem ser diferentes dos usados para testá-lo. Do contrário, o desempenho preditivo do modelo tende a ser otimista, uma vez que os algoritmos de AM tendem a melhorar de alguma forma o seu desempenho preditivo nesses objetos durante a fase indutiva [Faceli et al., 2011].

Técnicas de amostragem de dados atuam na definição de subconjuntos de treinamento e de teste, onde os dados de treinamento são empregados para induzir e ajustar o modelo, e os dados de teste simulam a apresentação de novos objetos ao preditor. Um exemplo de utilização desses subconjuntos (Treino e Teste) é ilustrado na Figura 2.2.

A Figura 2.4 ilustra os principais métodos de amostragem [Han et al., 2011]. A seguir, os métodos mais comumente aplicados, *Holdout* e Validação Cruzada, são descritos em maiores detalhes.

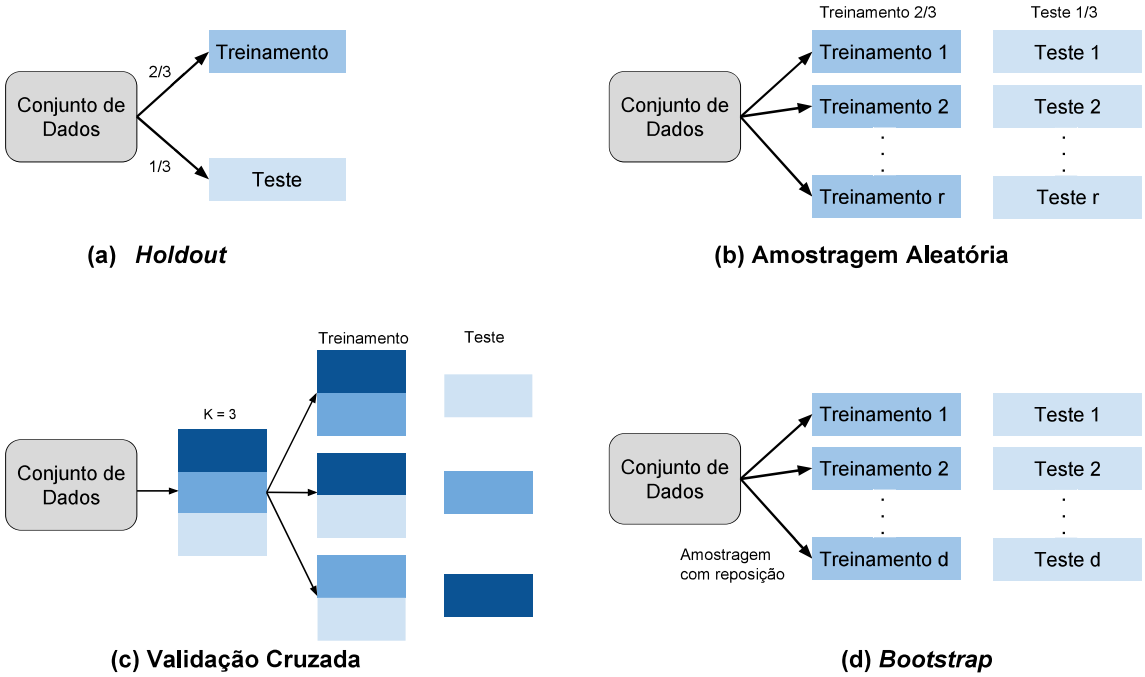

Figura 2.4 – Métodos de Amostragem de Dados [Adaptada de [Faceli et al., 2011]]

Holdout

Neste método, um conjunto de dados com N objetos é particionado aleatoriamente em subconjunto de treinamento e subconjunto de teste, na proporção de p objetos para treinamento e $p - 1$ para teste. Usualmente, utiliza-se $p = \frac{2}{3}$, conforme pode ser observado na Figura 2.4 (a).

A estimativa obtida por este método é dita pessimista, uma vez que apenas uma porção dos dados iniciais é utilizada para induzir o modelo. Além disso, o *Holdout* não

permite avaliar a variação de desempenho dos algoritmos mediante a apresentação diferentes combinações de objetos no treinamento [Faceli et al., 2011]. Uma forma contornar as desvantagens deste método é a utilização de **amostragem aleatória** (Figura 2.4 (b)). Essa variação do método faz com que o *Holdout* seja repetido r vezes, de maneira que o desempenho final do preditor é a média do desempenho obtido em cada iteração.

Validação Cruzada

Na Validação Cruzada (em inglês, *k-fold cross validation*), os N objetos iniciais são aleatoriamente particionados em k subconjuntos mutualmente exclusivos ($D_1, ..., D_k$), onde $N \geq k > 0$. Após isso, o treinamento e o teste são executados k vezes, sendo que em cada iteração i a partição D_i é reservada para teste, enquanto que as demais são utilizadas conjuntamente no treinamento. Este cenário é ilustrado na Figura 2.4 (c).

Ao contrário do *Holdout* e da amostragem aleatória, neste método, todos os objetos são utilizados o mesmo número de vezes para treinamento e uma vez para teste. O desempenho final do preditor é obtido pela média do desempenho em cada subconjunto de teste. Quando $k = N$ tem-se o método *leave-one-out*, onde um único objeto é utilizado para teste a cada iteração. Apesar de produzir uma estimativa mais fiel do desempenho do modelo, este método é computacionalmente caro e, geralmente, aplicado a um conjunto pequeno de dados.

2.3 Considerações Finais do Capítulo

Este Capítulo introduziu dois conceitos base para esta pesquisa: Descoberta de Conhecimento em Base de Dados e Aprendizado de Máquina. Estes conceitos serão explorados para definição de Aprendizado de Máquina Automático (Capítulo 3), para categorização das soluções existentes de Seleção de Processo Completo, de acordo com o processo de DCBD (Capítulo 4), bem como nas definições experimentais, onde serão referenciadas as medidas de desempenho, tanto de classificação quanto de regressão, bem como as técnicas de amostragem de dados.

3. APRENDIZADO DE MÁQUINA AUTOMÁTICO

Com o crescimento do interesse em Ciência de Dados, profissionais da área estão frequentemente enfrentando o desafio de decidir qual algoritmo, e valores de hiper-parâmetros associados, são melhores para determinada tarefa de Aprendizado da Máquina (AM). A seleção do algoritmo efetivo e a configuração de seus hiper-parâmetros são tarefas que requer amplo conhecimento em AM e/ou testes repetitivos. Isso não está apenas além da capacidade de usuários leigos, com pouca experiência em computação, mas, também, com frequência é uma tarefa não-trivial mesmo para especialistas em AM [Sparks et al., 2015].

Aprendizado de Máquina Automático (AutoAM) é o campo de AM que visa atender a essas necessidades através do desenvolvimento de soluções que permitem aos profissionais de Ciência de Dados, especialistas e não-especialistas, criarem eficientemente modelos preditivos ajustados a problemas específicos, com um mínimo de intervenção humana [de Sá et al., 2017].

Neste Capítulo serão abordados os principais conceitos de Aprendizado de Máquina Automático. A Seção 3.1 apresenta uma definição da área. A Seção 3.2 apresenta AutoAM aplicado à Seleção de Algoritmos, com ênfase na abordagem de Meta-Aprendizado. A Seção 3.3 apresenta AutoAM aplicado à Configuração de Algoritmos, com ênfase na abordagem de Otimização Baseada em Bandidos Multi-Armados. Por fim, a Seção 3.4 apresenta os conceito de AutoAM para Seleção de Processo Completo e a introduz uma Arquitetura Genérica de Componentes para soluções desse tipo, que será usada como base para categorizar os trabalhos relacionados, apresentados no Capítulo 4.

3.1 Definição de Aprendizado de Máquina Automático

Aprendizado de Máquina tem como objetivo melhorar o desempenho de uma classe de tarefas, com base na experiência adquirida anteriormente [Mitchell, 1997]. Pesquisas em AM tradicional visam o desenvolvimento e a análise de algoritmos de aprendizado que sigam essas diretrizes sem, no entanto, a preocupação do quão fácil é a utilização desses algoritmos. Um exemplo efetivo desse contexto é a tendência recente de Aprendizado Profundo (do inglês, *Deep Learning)* [LeCun et al., 2015], cujos algoritmos oferecem um excelente desempenho nas tarefas aos quais são aplicados, mas são muito difíceis de serem configurados.

Levando-se em consideração a quantidade de algoritmos de AM disponíveis (por exemplo, a biblioteca Weka [Witten e Frank, 2016] dispõe atualmente de cerca de 70 algoritmos entre métodos supervisionados e não-supervisionados) apenas a seleção do algoritmo

mais adequado ao problema já seria uma escolha trabalhosa. No entanto, cada algoritmo existente está geralmente associado com hiper-parâmetros, ou seja, entradas do algoritmo que determinam como ele irá atuar para realizar a generalização de dados não vistos (por exemplo, número *k* de vizinhos no k-NN, taxa de aprendizado em uma Rede Neural Artificial), sendo que qualidade do modelo predito criticamente depende da configuração desses hiper-parâmetros [Li et al., 2017, Thornton et al., 2013]. Sem nenhum tipo de ajuda externa, dado um problema de aprendizado, a seleção e a configuração de algoritmos é geralmente uma tarefa manual e, portanto, custosa por parte do usuário.

Com foco nesse desafio, AutoAM propõe a automatização desse processo laborioso, a partir da construção de soluções de alto-nível que atuem sob os algoritmos de aprendizado, de maneira a selecioná-los e configurá-los sem a interferência do usuário [Thornton et al., 2013].

Apesar do termo "Aprendizado de Máquina Automático"ter sido definido recentemente, no trabalho de [Thornton et al., 2013]. A busca pela automatização de partes do processo de AM vem sendo feita desde a década de 70 [Rice, 1976], seja em tarefas distintas ou na junção dessas tarefas. As tarefas envolvidas em AutoAM são apresentadas na Figura 3.1 e tem como base o processo de Descoberta de Conhecimento em Bases de Dados (DCBD), onde os algoritmos de AM são ferramentas essenciais.

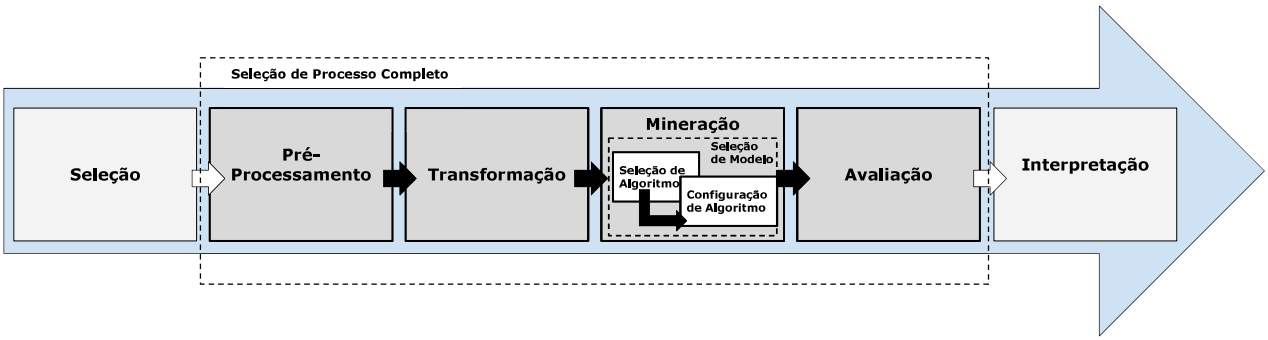

Figura 3.1 – Tarefas de Aprendizado de Máquina Automático

A tarefa de Mineração de Dados é tida como a mais importante do processo de DCBD, pois é onde efetivamente os padrões serão derivados, a partir da utilização dos algoritmos de AM [Fayyad et al., 1996b]. Com o foco nessa tarefa, a automatização mais simples e, portanto, a primeira a ser explorada pelos pesquisadores, foi a Seleção de Algoritmos de AM, que será detalhada na Seção 3.2. Essa tarefa é tida como mais simples pois, embora existam muitos algoritmos, esse espaço de busca ainda é limitado e finito se comparado ao espaço de busca de hiper-parâmetros, que frequentemente envolve valores reais.

A tarefa de Configuração de Algoritmos (ou otimização de hiper-parâmetros) começou a ser explorada em meados da década de 90 na medida em que AM começou a ser mais aplicado e observou-se o impacto que esses hiper-parâmetros tinham sob a qualidade

do modelo gerado. Assim, ao combinar a escolha do algoritmo com sua parametrização, AutoAM passou a oferecer a Seleção de Modelo.

No entanto, ao observarmos o processo de DCBD como um todo podemos notar que, embora MD seja uma tarefa chave, as demais também desempenham papel importante, senão essencial, uma vez que os algoritmos de AM são susceptíveis as características dos dados de entrada (desbalanceamento, dados ausentes, dimensionalidade, etc). Sendo assim, em um cenário real, as escolhas do usuário passam a envolver também esses métodos de pré-processamento, que por sua vez possuem seus próprios hiper-parâmetros, elevando o papel de AutoAM para escolha não somente do modelo (algoritmo de AM + hiper-parâmetros), mas de um processo completo de operadores (métodos de pré-processamento + algoritmo de AM + hiper-parâmetros).

Idealmente, um processo completo de operadores de DCBD, doravante chamado "processo completo", deve incluir a automatização de todo o processo. No entanto, as etapas de seleção de dados e interpretação, por definição, envolvem decisões que precisam da interferência humana. Sendo assim, AutoAM foca-se nas etapas onde as tarefas podem ser efetivamente automatizadas: pré- processamento, transformação, mineração e avaliação de dados.

Adicionalmente, a área de AutoAM também possui uma vertente que investiga a automatização de estruturas de Redes Neurais Convolucionais, tais como apresentado em [Liu et al., 2018] e [Zoph e Le, 2016]. Porém este tema está fora do escopo deste trabalho, maiores detalhes podem ser encontrados em [Elsken et al., 2018].

As próximas seções apresentam os processos de Seleção de Algoritmos 3.2, Configuração de Algoritmos 3.3 e Seleção de Processo Completo 3.4, que é o foco principal desta Tese.

3.2 Seleção de Algoritmos

Seleção de Algoritmos (SA) consiste em escolher, a partir de uma coleção de algoritmos, aquele que espera- se que resolva um determinado problema mais eficientemente [Lindauer et al., 2015].

Rice [Rice, 1976] foi o primeiro a formalizar a ideia de selecionar dentre diferentes algoritmos com base nas características do problema.

Definição 1: Dado

- Um conjunto P de problemas;
- um espaço de algoritmos A; e
- uma medida de desempenho $m : P \times A \rightarrow \mathbb{R}$

o problema de seleção de algoritmos consiste em encontrar um mapeamento $f : P \rightarrow A$ que otimize $m(p, a(p))$, isto é, o desempenho esperado para os problemas $p \in P$, atingido a partir da execução do algoritmo selecionado $a \in A$ para o problema p.

Na prática, o mapeamento f é frequentemente implementado usando características dos problemas $p \in P$. Essas características são então mapeadas para um algoritmo usando técnicas de AM. Essa estratégia pode envolver um modelo aprendido único ou uma combinação complexa de vários que, dado um novo problema a ser resolvido, é usada para decidir para qual algoritmo mapear um determinado problema [Bischl et al., 2016].

Os tipos de decisão que o processo de SA produz orientam a escolha dos modelos de aprendizado que realizam a seleção. Se apenas um único algoritmo deve ser executado, pode-se treinar um modelo de classificação que tome exatamente essa decisão. Existem alternativas para usar modelos de classificação para selecionar um único algoritmo, bem como existem modelos de regressão que predizem o desempenho de cada algoritmo do conjunto disponível [Mersmann et al., 2013] [Roberts et al., 2007]. Outras abordagens incluem o uso de técnicas de agrupamento para particionar instâncias de problemas em espaço de características e tomar decisões para cada partição separadamente [Kadioglu et al., 2010].

Para tomar decisões, sistemas de SA precisam de informações sobre os problemas a serem resolvidos e o desempenho de algoritmos neste conjunto de problemas. A extração dessas informações, características usadas por técnicas de AM para seleção, incorre em sobrecarga de processamento não necessária quando apenas um único algoritmo é usado para todos problemas, independentemente das características do problema. Portanto, é desejável extrair informações da forma mais barata possível, garantindo, assim, que os benefícios do desempenho de usar seleção de algoritmos não sejam superados por essa sobrecarga [Bischl et al., 2016].

Algumas abordagens usam apenas o desempenho anterior dos algoritmos disponíveis como base para a seleção daqueles que serão executados para um dado problema [Streeter et al., 2007] [Silverthorn e Miikkulainen, 2010]. Essas abordagens têm o benefício de que os dados necessários podem ser coletados com um mínimo de sobrecarga à medida que os algoritmos são executados. Elas podem funcionar bem se o desempenho dos algoritmos for semelhante em amplos intervalos de instâncias de problemas. No entanto, quando essa suposição não é satisfeita (como na maioria dos casos), são necessários mais recursos informativos [Bischl et al., 2016].

Por fim, a área de Meta-Aprendizado (MA) utiliza os chamados meta-atributos, que são características descritivas de conjuntos de dados, juntamente com o desempenho de algoritmos sobre estes conjuntos de dados, e busca descobrir o relacionamento entra estes meta-atributos e o desempenho dos algoritmos, de maneira a prever o desempenho desses mesmos algoritmos, dado um novo conjunto de dados [Brazdil et al., 2008]. A Seção a seguir detalha o processo de MA para Seleção de Algoritmos.

3.2.1 Meta-Aprendizado para Seleção de Algoritmos

Meta-aprendizado (MA) estuda a forma como os algoritmos de AM podem aumentar sua eficiência por meio da experiência [Vilalta e Drissi, 2002]. O objetivo é entender como o próprio processo de aprendizado pode se tornar flexível de acordo com a natureza da tarefa considerada. Esta abordagem difere do aprendizado convencional (ou aprendizado-base), no escopo de seu nível de adaptação: enquanto o AM convencional é baseado no acúmulo de experiências de uma tarefa em específico (tais como, diagnóstico médico, estimativa de custo, detecção de fraudes, dentre outras), o meta-aprendizado é baseado no acúmulo de experiência sobre o desempenho de múltiplas aplicações de um algoritmo de AM [Vilalta et al., 2010].

Com base no trabalho seminal de Rice [Rice, 1976], Smith-Miles [Smith-Miles, 2009] define os seguintes pré-requisitos para tratar o problema de seleção de algoritmos usando uma abordagem de meta-aprendizado:

- um grande conjunto de problemas, com vários níveis de complexidade (espaço de problemas P);
- diversos algoritmos para tratar os problemas (espaço de algoritmos A);
- métricas de desempenho para avaliar o desempenho dos algoritmos (espaço de desempenho Y); e
- um conjunto de características apropriadas para descrever as propriedades dos problemas (espaço de características C).

A Figura 3.2 apresenta uma visão geral do processo de MA para SA, conforme proposto por [Brazdil et al., 2008]. Neste processo, podemos observar que a associação de características dos problemas com as medidas de desempenho dos algoritmos cria um conjunto de meta-conhecimento (conhecimento extraído sobre o processo de aprendizado) sobre o desempenho de algoritmos, utilizando os conceitos de “aprender a aprender” como base do MA para seleção de algoritmos.

O processo inicia com a construção de um repositório que armazena os P conjuntos de dados referentes às instâncias de problemas diversos. As propriedades de cada instância são investigadas e descritas de acordo com características independentes da tarefa em particular (meta-atributos preditivos ou meta-características), em C. Paralelamente à extração de meta-atributos, os algoritmos em A são aplicados aos P problemas, e o desempenho preditivo destes algoritmos é avaliado (meta-atributo alvo), em Y. As informações de caracterização e o desempenho dos algoritmos são agrupados, para cada problema, em uma base de conhecimento (meta-base), onde cada linha, equivalente a um par meta-atributo/meta-atributo alvo, é denominada meta-exemplo. Essa meta-base é utilizada como

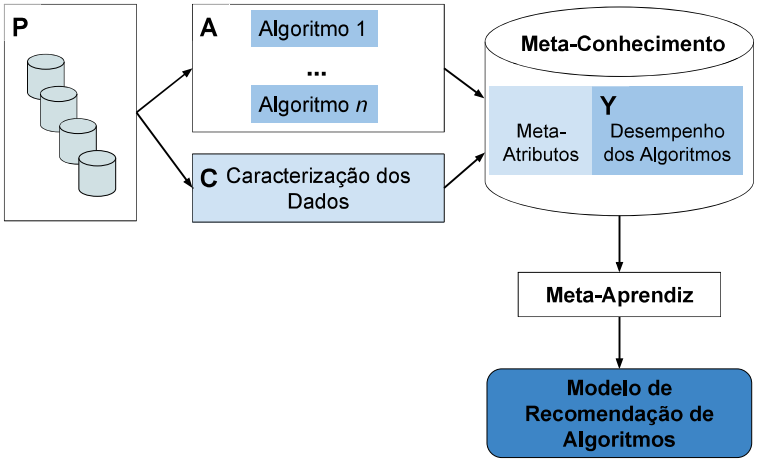

Figura 3.2 – Processo de Meta-Aprendizado para Seleção de Algoritmos [Adaptada de [Brazdil et al., 2008]

entrada para um algoritmo de AM, denominado meta-aprendiz, que induz um modelo capaz de relacionar os meta-atributos preditivos ao meta-atributo alvo, gerando a relação *f* da definição de seleção da algoritmos. Por fim, o modelo gerado pode ser utilizado para recomendar os algoritmos de AM mais adequados a um novo problema.

Com base nas definições apresentadas acima, temos que a base de uma abordagem de meta-aprendizado para recomendação de algoritmos é constituída por: i) caracterização dos problemas (representados por conjuntos de dados); ii) definição das medidas de avaliação de desempenho dos algoritmos; iii) formas de recomendação.

Caracterização da Base de Dados

Caracterizar bases de dados consiste em identificar e extrair propriedades que possivelmente afetem o desempenho preditivo dos algoritmos de aprendizado [Souza, 2010]. Essas características, que são usadas como meta-atributos de entrada, devem enfatizar os aspectos importantes do domínio, provendo conhecimento útil capaz de diferenciar o desempenho preditivo de um dado conjunto de algoritmos de AM. Dessa forma, a qualidade das recomendações realizadas é dependente da caracterização dos dados e, por isso, esse componente é essencial para o sucesso dos métodos baseados em meta-aprendizado [Rossi, 2013].

De acordo com Brazdil *et al.* [Brazdil et al., 2008], o desenvolvimento desses meta-atributos deve levar em consideração os seguintes pontos:

- Poder discriminativo: o conjunto de meta-atributos deve conter informações que discriminem diferentes algoritmos de AM em termos de desempenho.
- Complexidade computacional: os meta-atributos não podem ser muito complexos computacionalmente. Do contrário, os ganhos obtidos em não realizar uma busca

por força bruta pelo melhor algoritmo podem ser mitigados pela caracterização dos dados.

- Dimensionalidade: o número de meta-atributos não deve exceder muito a quantidade de bases de dados disponíveis. Do contrário, pode ocorrer um super-ajuste no modelo.

Por ser uma etapa crucial no processo de seleção de algoritmos, diversos estudos tratam da definição, categorização e seleção de meta-atributos. De maneira geral, há três principais tipos de meta-atributos: i) diretos; ii) baseados em *landmarking*; e iii) baseados em modelo.

1. **Diretos**: os meta-atributos diretos podem ser ser obtidos por três grupos de medidas: simples, estatísticas e baseadas em teoria da informação. Grande parte dessas medidas foi definida pelos projetos STALOG [Michie et al., 1994] e METAL (em inglês, *A Meta-Learning Assistant for Providing User Support in Machine Learning and Data Mining*) (ESPRIT Nr. 26.357). Porém, outros trabalhos como [Kalousis, 2002] [Sohn, 1999] [Soares, 2004], também contribuíram neste sentido.

    - Simples: incluem informações gerais sobre um conjunto e dados (por exemplo, número de atributos, número de objetos e número de classes);
    - Estatísticas: são aplicadas para atributos numéricos das bases de dados e calculam grandezas estatísticas, tais como curtose, variância e assimetria;
    - Baseadas em teoria da informação: são aplicadas para caracterizar atributos nominais, por exemplo, entropia e informação mútua.

2. ***Landmarking***: são estimativas rápidas do desempenho de um algoritmo em uma determinada base de dados, propostos inicialmente por [Pfahinger et al., 2000]. O uso de *landmarking* baseia-se no conceito de que cada algoritmo opera de maneira satisfatória em uma área de competência, definida pela propriedade dos dados. Dessa forma, utilizando-se algoritmos simples com vieses indutivos diversos, chamados *landmarkers*, é possível obter informações importantes sobre a natureza do domínio em que eles são aplicados. Nesse contexto, o *landmarking* é utilizado para determinar a proximidade de um conjunto de dados em relação a outros, por meio da similaridade de desempenho dos algoritmos;

3. **Baseados em Modelo**: Assim como na abordagem baseada em *landmarking*, a baseada em modelo caracteriza dos dados de maneira indireta. Neste caso, ao invés de utilizar as medidas de desempenho do classificador induzido, usa-se a estrutura do modelo em si [Bensusan, 1998] [Peng et al., 2002].

A Tabela 3.1 apresenta exemplos de meta-atributos diretos (DIR), *landmarking* e baseados em modelo (BM).

Tabela 3.1 – Exemplos de Meta-Atributos Diretos (Simples, Estatísticos e Baseados em Teoria da Informação (BTI)), *Landmarking* e Baseados em Modelo

Tipo	Medida	Referência
DIR - Simples	Número de objetos Número de atributos Número de classes Número de atributos binários	[Michie et al., 1994]
DIR - Estatístico	Correlação média absoluta entre atributos, por classe Assimetria média absoluta dos atributos Curtose média dos atributos	
DIR - BTI	Entropia média dos atributos Informação mútua média entre classes e atributos Razão sinal/ruído	
Landmarking	Acurácia do 1-NN (k-NN com k=1) Acurácia do Naïve Bayes	[Bensusan e Giraud-Carrier, 2000]
BM	Altura (número de níveis) de uma Árvore de Decisão Número de nós de uma Árvore de Decisão Número de folhas de uma Árvore de Decisão	[Peng et al., 2002]

Medidas de avaliação

Para determinar qual algoritmo de AM utilizar em um determinado conjunto de dados, é necessário especificar as medidas de desempenho, para que uma lista de preferência dos algoritmos possa ser estabelecida [Kalousis, 2002]. A princípio, qualquer medida de desempenho empregada para avaliar os algoritmos candidatos pode ser utilizada [Faceli et al., 2011], sendo assim, por Seleção de Algoritmos ser uma tarefa de AM supervisionado, as principais medidas utilizadas são as mesmas apresentadas na Seção 2.2.1.

Além de medidas relacionadas com as predições feitas pelos modelos, outras medidas podem ser interessantes tais como, o tempo requerido para o algoritmo de AM construir um modelo, o tempo requerido para o modelo rotular um objeto, a quantidade de memória requerida pelo algoritmo e a simplicidade e interpretabilidade dos modelos construídos [Souza, 2010].

Formas de apresentação de recomendação

A forma de recomendação disponibilizada por um sistema de meta-aprendizado deve ser selecionada de acordo com o uso desejado para o sistema. Por exemplo, o usuário pode desejar saber qual o melhor algoritmo para uma determinada tarefa, ou pode querer um conjunto de recomendações.

A forma de recomendação determina o tipo de **meta-atributo alvo** e pode ser sintetizada em quatro abordagens [Brazdil et al., 2008]:

1. Recomendação do melhor algoritmo;

2. Recomendação de uma lista de algoritmos;

3. Recomendação de um *ranking* de algoritmos;

4. Estimativa de desempenho.

A primeira forma de recomendação consiste em fornecer um único algoritmo, isto é, aquele que produza o melhor modelo para uma dada tarefa, segundo o critério de desempenho utilizado [Kalousis, 2002]. A principal vantagem dessa abordagem é que o problema de meta-aprendizado pode ser visto como um problema de classificação, onde o atributo classe é o melhor algoritmo. O principal problema em aplicar essa abordagem é a ausência de alternativas para o algoritmo recomendado quando ele não puder ser utilizado, deixando o usuário sem opções de escolha [Faceli et al., 2011].

A segunda abordagem é mais flexível que a primeira, indicando, dentre os algoritmos considerados, um conjunto de algoritmos (geralmente pequeno) que espera-se que obtenha bom desempenho na tarefa em consideração. Nesse conjunto, além do melhor algoritmo, estão presentes os algoritmos que não possuem desempenho estatisticamente inferior ao melhor [Kalousis, 2002]. Esta abordagem foi inicialmente proposta no projeto STATLOG [Michie et al., 1994] e posteriormente adotada em trabalhos como [Brazdil et al., 1994] [Lindner e Studer, 1999]. A vantagem desta forma de recomendação é a maior gama de opções dada ao usuário, em comparação com a primeira abordagem. Porém, essa recomendação é feita de maneira desordenada, ou seja, não fornece indicações de qual algoritmo tentar como primeiro, segundo, e assim por diante [Brazdil et al., 2008].

Nestas duas primeiras abordagens, os problemas de meta-aprendizado são formulados como problemas de classificação, sendo o modelo de recomendação gerado por um algoritmo de classificação (meta-classificador). As classes a serem previstas podem ser tanto o nome de um determinado algoritmo, como a informação de que um determinado algoritmo é aplicável ou não para um problema [Kalousis, 2002].

A terceira abordagem exibe os algoritmos em ordem de preferência com relação à base de dados, ou seja, gera um *ranking* de algoritmos, tal como em [Kalousis, 2002], [Soares e Brazdil, 2000]. O critério de ordenação pode ser simplesmente a acurácia dos modelos, ou medidas mais complexas, que envolvem múltiplos objetivos, tais como tempo de execução do algoritmo ou a interpretabilidade do modelo gerado [Souza, 2010]. *Rankings* são particularmente recomendados para seleção de algoritmos, pois o sistema de MA pode ser desenvolvido sem qualquer informação sobre quantos algoritmos o usuário vai tentar utilizar, uma vez que esta quantidade depende dos recursos computacionais disponíveis e da importância que o desempenho do algoritmo tem para a tarefa em questão [Brazdil et al., 2008]. Por exemplo, se o tempo é um fator crítico, poucas opções serão selecionadas. Por outro lado, se a acurácia é o fator crítico, um maior número de alternativas vai ser testado, na tentativa de elevar o desempenho.

Por fim, caso o usuário esteja interessado no desempenho real do algoritmo a ser utilizado, ao invés do desempenho relativo, como o dado por *rankings*, o sistema de meta-

aprendizado pode fornecer recomendações em forma de valores, indicando o desempenho estimado para cada algoritmo [Brazdil et al., 2008]. Essa abordagem transforma o problema de recomendação em vários problemas de regressão, um para cada algoritmo. Exemplos de aplicação podem ser encontrados em [Gama e Brazdil, 1995] [Sohn, 1999]. Como o desempenho de cada algoritmo será medido, individualmente, em um problema de regressão, esta abordagem é flexível para mudanças nos algoritmos, pois o acréscimo ou retirada de um algoritmo não afeta o desempenho dos demais.

Em abordagens de regressão os problemas de meta-aprendizado são formulados como problemas de regressão, e o modelo de recomendação é gerado por um algoritmo de regressão (meta-regressor). O resultado obtido por um meta-regressor para um novo conjunto de dados pode ser a estimativa do desempenho do algoritmo de AM associado ao meta-regressor para esse novo conjunto [Rossi, 2013].

A grande vantagem desta última abordagem é que, além de fornecer uma recomendação direta para o usuário, ela pode ser adaptada para obter as outras três abordagens apresentadas anteriormente, onde: o melhor algoritmo é o que obteve a maior estimativa de desempenho, a lista de algoritmos contém o subconjunto que estima-se que terá um bom desempenho com base no melhor, e o *ranking* é o conjunto de todos os algoritmos, ordenados por desempenho.

3.3 Configuração de Algoritmos

Modelos de aprendizado são compostos por parâmetros que fazem parte de sua formulação matemática e cujo valores são estimados a partir de um conjunto de dados de treinamento [Murphy, 2012]. Nestes modelos, existem ainda parâmetros denominados hiper-parâmetros, que se diferenciam dos primeiros por não serem estimados do mesmo modo e por necessitarem de uma definição de valores definitiva, antes mesmo que o treinamento se inicie. Os hiper-parâmetros definem propriedades como a complexidade do modelo e o quão rápido os parâmetros serão aprendidos [Bishop, 2006]. Ou seja, os hiper-parâmetros estão diretamente ligados ao desempenho do modelo treinado e ao número de operações computacionais necessárias no aprendizado, assim como no tempo de duração.

Em AutoAM, a tarefa de determinar os hiper-parâmetros é denominada Configuração de Algoritmos (CA), ou otimização de hiper-parâmetros, e é altamente relacionada com a Seleção de Algoritmos [Bischl et al., 2016]. A principal diferença entre as duas tarefas diz respeito ao fato de que SA opera em um conjunto finito (e geralmente factível) de algoritmos, enquanto a CA opera em um espaço combinatorial de configurações de parâmetros de algoritmos, que envolve valores inteiros e reais[Bischl et al., 2016].

Segundo [Hutter et al., 2009], podemos definir a otimização de hiper-parâmetros como sendo:

Definição 2: Dado um algoritmo $a \in A$, com h hiper-parâmetros. Denota-se o domínio do i-ésimo hiper-parâmetro por Λ_i e o espaço total de configurações de hiper-parâmetros por $\boldsymbol{\Lambda} = \Lambda_1, \Lambda_2, , ..., \Lambda_h$. Um vetor de hiper-parâmetros é denotado por $\lambda \in \boldsymbol{\Lambda}$, e a com seus hiper-parâmetros instanciados é denotado por a_λ. Dado um conjunto de dados D, o objetivo é encontrar:

$$\boldsymbol{\lambda}^* = \arg\min_{\lambda \in \boldsymbol{\Lambda}} \boldsymbol{V}(\mathfrak{L}, a_\lambda, D_{treino}, D_{validacao}) \tag{3.1}$$

onde $\boldsymbol{V}(\mathfrak{L}, a_\lambda, D_{treino}, D_{validacao})$ mede a perda do modelo gerado pelo algoritmo a com hiper-parâmetros λ no conjunto de dados de treinamento D_{treino} e avaliado no conjunto de dados de validação $D_{validacao}$. Geralmente o protocolo de validação $\boldsymbol{V}$ é *Holdout* ou Validação Cruzada (Seção 2.2.1) e a função de perda utilizada é definida pelo usuário, por exemplo, erro de classificação (Seção 2.2.1).

Os domínios de hiper-parâmetros podem ser valores reais (por exemplo, taxa de aprendizado), valores inteiros (por exemplo, número de camadas de uma Rede Neural Artificial), binários (por exemplo, se deve-se fazer poda em uma Árvore de Decisão ou não), ou categóricos (por exemplo, escolha do kernel de um SVM). Além disso, o domínio de configurações pode conter *condicionalidade*, isto é, um hiper-parâmetro pode ser relevante apenas se outro hiper-parâmetro (ou uma combinação de hiper-parâmetros) assume um determinado valor. Espaços condicionais assumem a forma de grafos acíclicos direcionados [Thornton et al., 2013].

No geral, qualquer método de otimização caixa-preta pode ser aplicado para otimização de hiper-parâmetros. Dentre estes métodos se destacam os métodos livre de modelo e os métodos de otimização Bayesiana. Além disso, novas estratégias como métodos de fidelidade múltipla (do inglês, *multi-fidelity*) estão sendo utilizados para modelos complexos, devido ao fato de explorarem a aproximação de desempenho, o que é mais barata do que a avaliação do modelo de aprendizado completo.

3.3.1 Métodos Livre de Modelo

Em relação aos métodos livres de modelo, a busca exaustiva (*Grid Search*) é a mais básica, e consiste na avaliação de um conjunto finito de hiper-parâmetros pré-definidos, sem a utilização de qualquer otimização durante o processo de pesquisa. Ela busca através de um subconjunto manualmente especificado do espaço de hiper-parâmetros guiada por um método de estimativa de desempenho preditivo, por exemplo, Validação Cruzada [Montgomery, 2017]. Este método sofre com a "maldição"da dimensionalidade [Yang e Wu, 2006], uma vez que o número de avaliação de funções aumenta exponencialmente com a dimensionalidade do espaço de busca.

Alternativamente à busca exaustiva, a busca aleatória amostra configurações aleatoriamente até que um determinado orçamento seja atingido [Bergstra e Bengio, 2012]. Como vantagens em relação à busca exaustiva, a busca aleatória apresenta a facilidade de paralelização e permite a alocação de flexível de recursos. Em termos experimentais, busca aleatória é considerada um bom método comparativo (*baseline*), pois não faz suposições sobre o algoritmo a ser otimizado e, dada uma quantidade suficiente de recursos pode atingir um desempenho arbitrariamente próximo ao ótimo. A busca aleatória também é um método útil para inicializar um processo de otimização, pois explora todo o espaço de configurações e, portanto, muitas vezes encontra configurações com desempenho razoável.

Meta-heurísticas baseadas em população, tais como Algoritmos Genéticos (AG) [Holland, 1992], Programação Genética (PG) [Koza, 1992] e Otimização por Enxame de Partículas (PSO, do inglês, *Particle Swarm Optimization*) [Kennedy, 2011] são algoritmos que mantém uma população, isto é, um conjunto de configurações, e melhora esta população aplicando perturbações locais e combinações de diferentes configurações para obter uma nova geração de melhores configurações [Boussaïd et al., 2013]. Esses métodos são conceitualmente simples, podem lidar com diferentes tipos de dados, e com paralelização.

3.3.2 Métodos Baseados em Otimização Bayesiana

A Otimização Bayesiana (OB) é um algoritmo iterativo com dois componentes principais: um modelo substituto probabilístico e uma função de aquisição para decidir qual ponto avaliar em seguida. Em cada iteração, o modelo é ajustado a todas as observações da função-alvo feitas até então. Então, a função de aquisição, que usa a distribuição preditiva do modelo probabilístico, determina a utilidade de diferentes pontos candidatos, balanceando exploração e aproveitamento. Embora outras opções estejam disponíveis, geralmente a otimização Bayesiana emprega *Expected Improvement* (EI) [Jones et al., 1998] como função de aquisição, e processos Gaussianos [Ma, 2012] ou o algoritmo Random Forest [Breiman, 1996] para modelar a função-alvo.

Otimização Bayesiana é atualmente considerada o estado-da-arte em métodos de otimização de caixa-preta para otimização de hiper-parâmetros por obter resultados significativos na configuração algoritmos de Aprendizado de Máquina Profundo [Snoek et al., 2015, Dahl et al., 2013]

Métodos de otimização de hiper-parâmetros baseados em otimização Bayesiana incluem ParamILS [Hutter et al., 2009] e SMAC (do, inglês, *Sequential Model Based Algorithm Configuration*) [Hutter et al., 2011].

3.3.3 Métodos de Fidelidade Múltipla

O aumento constante do volume de dados a ser processado, que leva a geração de modelos cada vez mais complexos pelos algoritmos de AM são um grande desafio da configuração de algoritmos, pois tornam a avaliação do desempenho dos modelos gerados por algoritmos de caixa-preta mais cara, uma vez que tais modelos são gerados sob o conjunto completo de treinamento. Assim, treinar uma única configuração de hiperparâmetro em grandes conjuntos de dados pode exceder várias horas e levar vários dias [Krizhevsky et al., 2012].

Técnicas comuns para acelerar a calibragem desses modelos incluem treinar configurações candidatas em um pequeno subconjunto de dados por algumas iterações, usar um subconjunto de atributos preditivos ou usar poucos *folds* na Validação Cruzada.

Os métodos de fidelidade múltipla usam essas heurísticas manuais em algoritmos formais, usando as chamadas aproximações de baixa fidelidade da função de perda a ser minimizada. Essas aproximações introduzem um *trade-off* entre o desempenho da otimização e o tempo de execução, mas, na prática, a aceleração obtida supera o erro de aproximação [Kandasamy et al., 2017]. Diversos métodos de fidelidade múltipla foram aplicados no contexto de otimização de hiper-parâmetros, tais como métodos baseados na predição de curvas de aprendizado [Kohavi e John, 1995] e métodos baseados em bandidos multi-armados [Jamieson e Talwalkar, 2016].

Recentemente, os métodos baseadas em bandidos multi-armados têm se destacado com resultados que superam a otimização Bayesiana, tida até então como estado-da-arte nesta tarefa, em diversas aplicações. Devido a esses resultados promissores, métodos baseados em bandidos multi-armados são aprofundados nas próximas seções.

3.3.4 Otimização Baseada em Bandidos Multi-Armados

Os problemas baseados em bandidos foram introduzidos em meados da década de 50 por [Thompson, 1933], em um estudo que investigava o processo de aprendizado animal. Inicialmente, o estudo foi executado com ratos que enfrentaram o dilema de escolher entre direita e esquerda, após entrarem em um labirinto em forma de T, sem saber em qual dos lados encontrariam comida. Já na aplicação do estudo com humanos, foram utilizadas máquinas com duas alavancas ("two-armed bandit"), onde cada alavanca oferecia uma recompensa aleatória, cuja distribuição era desconhecida pelo jogador. A máquina foi denominada "two-armed bandit"em homenagem à "one-armed bandit", uma antiga denominação para as máquinas caça-níquel operadas por uma alavanca. Nesse caso, a denominação bandido seria pelo fato das máquinas "roubarem dinheiro".

Lattimore et al. [Lattimore e Szepesvári, 2018] definem o problema como sendo:

Definição 3: O problema baseado em bandidos é um jogo sequencial entre um **aprendiz** e um **ambiente**. O jogo é realizado em *rd* rodadas onde $rd \in \mathbb{N}^+$ é denominado **horizonte**. Em cada rodada, o aprendiz seleciona uma ação ac_t de um conjunto de ações *AC* e o ambiente então revela a recompensa $rc_t \in \mathbb{R}$. A seleção da ação ac_t é baseada no **histórico** $H_{t-1} = (ac_1, rc_1, ..., ac_{t-1}, rc_{t-1})$. A **política** é um mapeamento do histórico para ações. O objetivo final do aprendiz é escolher ações que levem à acumulação da maior quantidade possível de recompensas ao final das *rd* rodadas, que é $\sum_{t=1}^{rd} rc_t$.

O principal desafio dos problemas baseados em bandidos é que o aprendiz não conhece o ambiente, apenas sabe que o ambiente verdadeiro leva à um conjunto ε chamado **classe do ambiente**.

Outro conceito importante é o de **arrependimento** do aprendiz. Ele é relativo a uma política π, e consiste na diferença entre o total de recompensa esperada ao usar π para *rd* rodadas, e o total de recompensas coletadas pelo aprendiz após *rd* rodadas. A recompensa relativa à um conjunto de políticas Π é o máximo arrependimento relativo a qualquer política $\pi \in \Pi$, onde o conjunto Π é conhecido como **classe do competidor**. De outra forma, pode-se dizer que o arrependimento mede o desempenho do aprendiz em relação à melhor política da classe do competidor. Costuma-se medir o arrependimento em relação a um conjunto de políticas Π - que é grande o suficiente para incluir a política ótima para todos os ambientes em ε. Nesse caso, o arrependimento mede a perda sofrida pelo aprendiz em relação à política ótima.

Para uma dada política e classe de competidor, a recompensa depende do ambiente. Os ambientes onde o arrependimento é grande são aqueles em que o aprendiz está se comportando pior. O caso ideal é que o arrependimento seja pequeno para todos os ambientes. O pior arrependimento é o arrependimento máximo em todos os ambientes possíveis.

Uma grande classe de ambiente corresponde a um menor conhecimento por parte do aprendiz. Já uma classe de competidor grande significa que o arrependimento é um critério mais exigente. Critérios para selecionar esses conjuntos apropriadamente devem levar em consideração que: (a) as garantias sobre o arrependimento sejam significativas e (b) existam políticas que tornem o arrependimento pequeno.

Aplicações comuns de soluções baseadas em bandidos incluem: AB Testing, Posicionamento de Anúncios, Sistemas de Recomendação, Busca em Árvores e Alocação de Recursos.

Exploração Pura

No geral, as políticas propostas para problemas de bandidos multi-armados visam maximizar a recompensa acumulada e minimizar o arrependimento, para isso, são designa-

das para balancear cuidadosamente exploração e aproveitamento. Porém, existem cenários onde é desejável identificar a "alavanca"com maior retorno médio, dado um orçamento fixo de recursos disponíveis, de maneira que a exploração pode ser realizada mais efetivamente. Problemas dessa natureza são conhecidos como "Exploração Pura". Tais problemas tem como objetivo minimizar o arrependimento simples, definido em função da distância para solução ótima, o mais rápido possível e em qualquer configuração [Li et al., 2017].

De acordo com [Lattimore e Szepesvári, 2018], problemas de exploração pura podem ser do tipo: arrependimento simples, identificação da melhor solução com uma confiança fixa e identificação da melhor solução com um orçamento fixo.

No contexto deste trabalho, utilizaremos último tipo, a identificação da melhor solução com um orçamento fixo. Esta variação da Exploração Pura consiste em identificar uma solução ótima ("best arm") tendo uma restrição no horizonte (número de rodadas que serão realizadas) e deve minimizar a probabilidade de selecionar uma solução sub-ótima.

Algoritmo Hyperband

O algoritmo Hyperband foi proposto por Li et al. [Li et al., 2017] para otimização de hiper-parâmetros. Esta estratégia modela o problema de otimização como um problema de Exploração Pura e tem como foco a otimização da **avaliação de configurações**. Para isso, realizam alocação adaptativa de recursos para configurações promissoras, enquanto as menos promissoras são eliminadas.

O algoritmo Hyperband utiliza como sub-rotina o algoritmo Successive Halving, proposto por [Jamieson e Talwalkar, 2016]. O procedimento do Successive Halving é simples: dado um número n de configurações pré-definido, aloca um orçamento para cada configuração, avalia o desempenho dessas configurações, elimina a metade com pior desempenho e repete até que reste uma configuração. A ideia principal é alocar exponencialmente mais recursos para configurações mais promissoras.

Dado um orçamento fixo B, B/n recursos são alocados em média dentre as configurações. Porém, não é claro, *a priori*, se deve-se: a) considerar mais configurações (n grande) com um orçamento médio pequeno; ou b) considerar um pequeno número de configurações com um orçamento grande.

O Algoritmo 3.1 apresenta o pseudo-código do Hyperband. Para endereçar o problema "n versus B/n", o Hyperband considera vários valores possíveis de n para um orçamento fixo B. Em essência, ele realiza uma busca exaustiva (Seção 3.3.1) sobre valores factíveis de n. Associada com cada valor de n está uma quantidade miníma de recursos r, que é alocada para todas as configurações antes de algumas serem descartadas: um grande valor de n corresponde a um pequeno r e leva a uma parada antecipada (*early stopping*) mais agressiva. De maneira geral, o Hyperband possuí dois componentes principais:

1: **Entrada** : R, η
2: **Inicializacao**: $s_{max} = \lfloor \log_{\eta}(R) \rfloor$, $B = (s_{max} + 1)R$
3: **for** s $\in \{s_{max}, ..., 0\}$ **do**
4: $n = \left\lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \right\rceil$, $r = R\eta^{-s}$
5: $T = obter\_hiper - parametros(n)$
6: **for** i $\in \{0, ..., s\}$ **do**
7: $n_i = \lceil n\eta^{-i} \rceil$
8: $r_i = r\eta^i$
9: $L = \{avaliacao\_de\_desempenho(t, r_i) : t \in T\}$
10: $T = top\_k(T, L, \left\lfloor \frac{n_i}{\eta} \right\rfloor)$
11: **end for**
12: **end for**
13: **return** Configuração com melhor desempenho encontrada

Algoritmo 3.1 – Algoritmo Hyperband, adaptado de [Li et al., 2017]

1. Laço Interno: executa o Successive Halving para valores fixos de n e r (linhas 6-11).

2. Laço Externo: itera sobre valores valores diferentes de n e r (linhas 3-12).

Cada execução do Successive Halving utiliza uma quantidade de recursos B e corresponde a um diferente balanceamento (*tradeoff*) entre n e B/n.

Hyperband requer dois parâmetros de entrada:

1. R, que corresponde a máxima quantidade de recursos que pode ser alocada para uma única configuração;

2. η, um parâmetro que controla a proporção de configurações descartadas em cada rodada do Successive Halving.

Esses dois parâmetros definem quantas execuções do Successive Halving serão realizadas, especificamente $s_{max} + 1$ valores diferentes para n são considerados com $s_{max} = \lfloor \log_{\eta}(R) \rfloor$.

O algoritmo inicia com uma execução mais agressiva do Successive Halving $s = s_{max}$, o que configura n para *maximizar a exploração*, sujeito a restrição de que, para pelo menos uma configuração, serão alocados R recursos. Cada execução subsequente do Successive Halving reduz n por um fator de aproximadamente η até que a execução final, $s = 0$, são alocados R recursos para todas as configurações. Essa última execução caracteriza uma pesquisa aleatória clássica 3.3.1.

Desta forma, Hyperband realiza uma pesquisa geométrica no orçamento médio por configuração, e remove a necessidade de selecionar n para um orçamento fixo, com um custo de aproximadamente $s_{max} + 1$ mais trabalho do que executar o Successive Halving para um único valor de n. Com isso, Hyperband está apto a explorar situações onde a

alocação adaptativa de recursos trabalha bem, enquanto faz ganho de situações onde uma alocação mais conservadora é requerida.

Hyperband utiliza três funções:

- *obter_hiper − parametros*(n) (linha 3): função que retorna um conjunto de n configurações amostradas de uma distribuição definida sobre o espaço de busca.
- *avaliacao_de_desempenho*(t, r): função que avalia uma configuração (t) utilizando os recursos de entrada (r) e retorna o desempenho de validação.
- *top_k*(T, L, k): função que recebe um conjunto de configurações (T), com seus respectivos desempenhos associados (L), e retorna as k melhores configurações.

Usando avaliações de baixa-fidelidade baratas, Hyperband demonstrou superioridade de desempenho a pesquisa aleatória tradicional e a otimização Bayesiana na avaliação de diferentes tipos de recursos, como atributos e épocas [Li et al., 2017].

3.4 Seleção de Processo Completo

Inicialmente, os trabalhos voltados para apoio ao processo de Aprendizado de Máquina Automático tinham como foco selecionar um de algoritmo aplicável para um dado conjunto de dados. Porém, é sabido que aplicações reais de Mineração de Dados frequentemente necessitam de pré-processamento de dados, o que inclui transformação de dados, seleção de atributos e amostragem, juntamente com as tarefas de classificação e/ou regressão, para construir o modelo final [Brazdil e Giraud-Carrier, 2018]. Assim, as pesquisas na área voltaram sua atenção para o desenvolvimento de abordagens de apoio a Seleção Automática de Processos Completos (SPC).

Um processo completo (em inglês, *workflow*, *pipeline* ou simplesmente *full-model*) pode ser entendido como uma combinação de operadores (algoritmos de aprendizado de máquina e métodos de pré-processamento em geral), selecionados a partir de um espaço de operadores disponíveis, com o objetivo de produzir um modelo preditivo com bom desempenho para um determinado problema. Além dos operadores, um processo completo deve incluir a configuração de hiper-parâmetros desses operadores.

Estendendo a definição de [Thornton et al., 2013] do problema de seleção combinada de algoritmos e otimização de hiper-parâmetros (CASH, do inglês, *Combined Algorithm Selection and Hiperparameter Optimization*) para selecionar um subconjunto de operadores, podemos definir a SPC como:

Definição 4: Dado W subconjuntos de operadores de DCBD $Op = \{Op^1, ..., Op^W\}$, com espaço de hiper-parâmetros associados $\mathbf{\Lambda} = \{\mathbf{\Lambda}^1, ..., \mathbf{\Lambda}^W\}$, um vetor de hiper-parâmetros

de um operador $op \in Op$ é denotador por $\lambda \in \mathbf{\Lambda}$, e op com seus hiper-parâmetros instanciados é denotado por op_λ. Um subconjunto de operadores é uma combinação válida de operadores de pré-processamento e um algoritmo de aprendizado, de modo a tornar sempre possível a fase de Mineração de Dados, restrito de acordo com a ordem das fases do processo de DCBD a qual cada operador de pré-processamento pertence e a compatibilidade de operadores.

O objetivo da SPC é encontrar processo completo, ou seja, subconjunto de operadores, com seus respectivos hiper-parâmetros associados, denotado por op^*_λ, de maneira que:

$$op^*_\lambda = \underset{op \in Op, \lambda \in \mathbf{\Lambda}}{\arg\min}\ \boldsymbol{V}(\mathfrak{L}, op_\lambda, D_{treino}, D_{validacao}) \tag{3.2}$$

onde $\boldsymbol{V}(\mathfrak{L}, op_\lambda, D_{treino}, D_{validacao})$ mede a perda do processo completo gerado pelo subconjunto de operadores op com hiper-parâmetros λ no conjunto de dados de treinamento D_{treino} e avaliado no conjunto de dados de validação $D_{validacao}$. Assim como na Configuração de Algoritmos, o protocolo de validação $\boldsymbol{V}$ é *Holdout* ou Validação Cruzada (Seção 2.2.1) e a função de perda utilizada $\mathfrak{L}$ é definida pelo usuário, por exemplo, erro de classificação (Seção 2.2.1).

As seções a seguir abordam uma breve discussão sobre o Espaço de Busca de problemas de SPC e uma descrição dos componentes principais de uma solução de SPC.

3.4.1 Espaço de Busca

Sun [Sun et al., 2013] define que um espaço de busca consiste de todos os operadores de DCBD que estão disponíveis para um determinado conjunto de dados, tais como um conjunto de métodos de seleção de características, um conjunto de técnicas de transformação de dados, um conjunto de algoritmos de AM, dentre outras.

Neste espaço, um processo completo é abstraído como um grafo acíclico direcionado (GAD) consistindo em operadores que são conectados com base em alguma relação. A Figura 3.3 apresenta uma ilustração deste conceito.

Levando em consideração o grande número de operadores existentes, bem como seus hiper-parâmetros, pode-se perceber que o espaço de busca de soluções é extremamente grande, claramente inviável de ser resolvido por força bruta [Sun et al., 2013]. Neste caso, embora uma solução ideal deva suportar todas as possibilidades de operadores existentes (que é finito) e seus hiper-parâmetros (que é potencialmente infinito), na prática este espaço é transformado em finito pela definição dos operadores que serão incluídos em uma determinada solução. Seja por limitações de recursos ou pelo foco da solução em si,

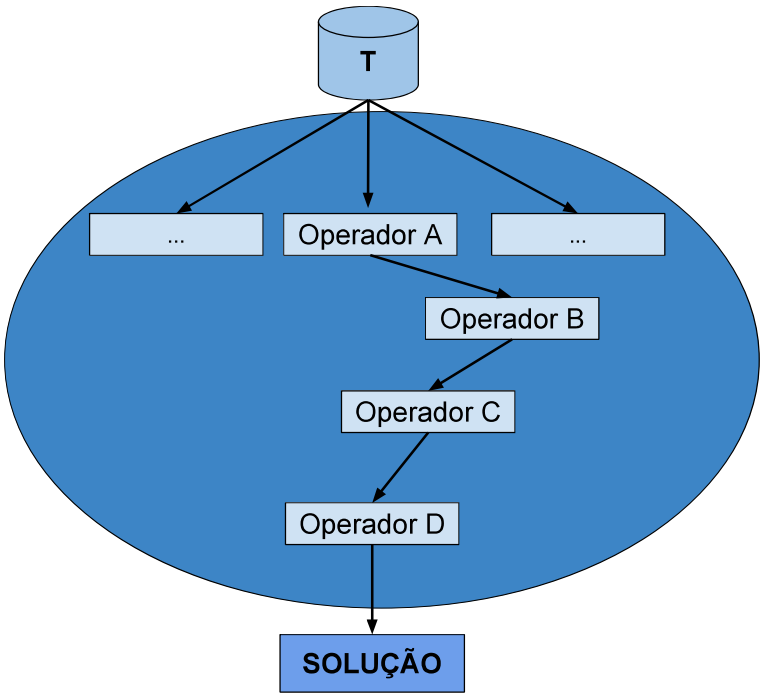

Figura 3.3 – Espaço de Operadores [Adaptada de [Sun et al., 2013]

que pode ser direcionada para uma única tarefa de AM ou limitada a algumas tarefas de pré-processamento.

3.5 Definição dos Componentes de uma Solução de Seleção de Processo Completo

Nos últimos 13 anos, diversas abordagens foram desenvolvidas para Seleção de Processo Completo. Estas serão apresentadas e analisadas no Capítulo 4. Nesta seção, é apresentada uma Arquitetura Genérica dos Componentes de uma solução de SPC, tanto os componentes técnicos como científicos. Essa arquitetura é útil para classificação dos trabalhos existentes e também pode ser usada como suporte para o desenvolvimento de novas soluções na área.

A Figura 3.4 apresenta os principais componentes em uma solução de SPC, as linhas pontilhadas indicam um componente ou conexão não obrigatório, ou não identificado em todas as soluções:

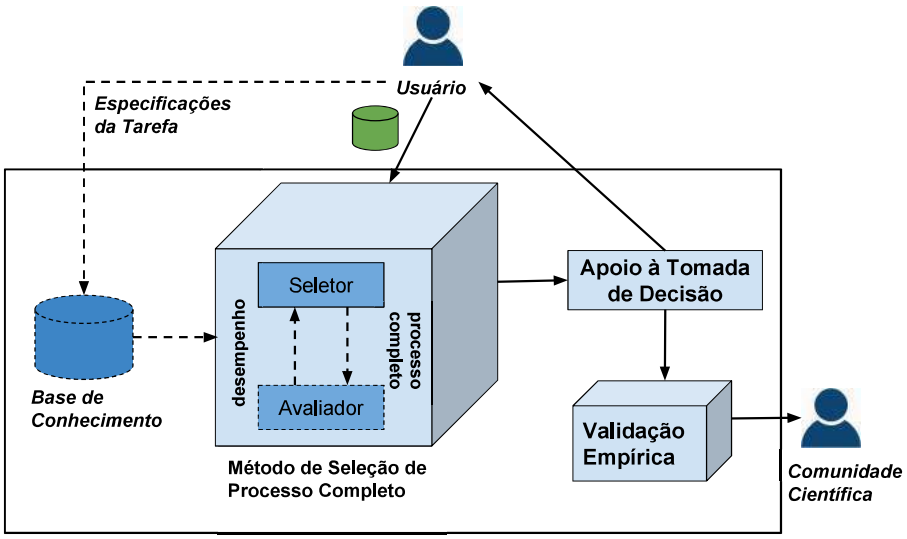

Figura 3.4 – Arquitetura Genérica de uma Solução para Apoio à Seleção de Processo Completo

3.5.1 Base de Conhecimento

Algumas soluções de SPC requerem especificações explícitas da tarefa, conhecimento de tarefas de aprendizado passado e/ou detalhes sobre o funcionamento de cada operador, enquanto outras soluções são capazes de realizar a recomendação do zero, apenas baseadas na definição do espaço de pesquisa e no conjunto de dados de entrada. Para o primeiro grupo, o componente Base de Conhecimento identifica os dados e as informações usados como entrada pelo método de SPC. Esta informação pode conter:

- Operadores: caracterização dos operadores disponíveis (métodos de pré-processamento e algoritmos de aprendizado). Esta informação descreve entradas, saídas, pré-condições e efeitos.

- Conjunto de Dados: caracterização dos dados disponíveis e da tarefa. Essas informações geralmente consistem em características de nível-meta que representam o quão difícil é resolver o problema para cada um dos operadores disponíveis.

- Desempenho: caracterização do desempenho de processos completos, sendo que as medidas de desempenho dependem da tarefa (classificação, regressão, etc.).

- Descrição da Tarefa: contém informações como tipo da tarefa, restrições, objetivo.

3.5.2 Método de Seleção de Processo Completo

É o principal componente de uma solução de SPC, consiste na técnica computacional aplicada para selecionar o processo completo. Esse componente é geralmente composto por dois sub-componentes: um seletor e um avaliador, e a recomendação se dá a partir da interação entre eles:

1. Seletor: dado o espaço de busca e o conjunto de dados, o seletor é responsável por determinar as configurações de processo completo que serão avaliadas. Dependendo do tipo de método aplicado, esta seleção pode levar ou não em consideração *feedbacks* oferecidos pelo otimizador ou informações externas provenientes da base de conhecimento, como detalhes dos operadores.

2. Avaliador: é responsável por medir o desempenho de uma configuração de processo completo escolhida pelo seletor e prover *feedback* para o mesmo. Geralmente isto é feito a partir do treinamento da configuração nos dados de entrada. Porém, há casos em que o método não avalia diretamente uma configuração, mas estima seu desempenho com base em informações contidas na base de conhecimento, como o desempenho dos operadores em problemas passados.

3.5.3 Apoio à Tomada de Decisão

A base de conhecimento e o método de SPC caracterizam as soluções de SPC do ponto de vista técnico dessas soluções. Já o componente Apoio à Tomada de Decisão avalia as soluções do ponto de vista do usuário, levando em consideração a abrangência das funcionalidades oferecidas. Assim, os tipos de apoio fornecidos podem ser categorizados em termos de:

- Fases do Processo de DCBD: esta categoria indica quais fases do processo de DCBD, além da fase de mineração de dados [Fayyad et al., 1996a], são suportadas;

- Otimização de Hiper-Parâmetros: embora possa ser considerada uma tarefa essencial para SPC, dado o impacto no desempenho final do processo, muitas soluções não realizam esta tarefa, logo ela é avaliada separadamente;

- Tarefa: as tarefas de Aprendizado de Máquina suportadas (tais como classificação, regressão, agrupamento);

- Recomendação: forma com que a recomendação é apresentada ao usuário, que varia de um único processo completo (o melhor), todas as opções possíveis ou um *ranking*

dos processos completos candidatos, classificados de acordo com algum critério de qualidade;

- Execução: algumas soluções geram um processo completo que pode ser executado diretamente, enquanto outras fazem uma recomendação que deve ser implementada pelo usuário;
- Implementação: indica se a solução é um sistema completo que suporta uma interação do usuário ou uma estrutura que o usuário pode editar e executar por conta própria (*framework*).

3.5.4 Validação

O componente de validação trata do processo de experimentação empírica da solução proposta. Embora isto não faça parte diretamente do núcleo de uma solução de SPC, é um aspecto de alta relevância quando se avalia a contribuição científica de uma solução do tipo.

Dada a natureza muito diversificada das soluções disponíveis para o problema de seleção de processo completo, não é possível compará-las apenas com base em suas características. Métodos alternativos podem, assim, ser usados para avaliar a aplicabilidade e a utilidade de uma solução.

A validação pode ser caracterizada em termos de:

- Tipo: como a validação é executada, por exemplo uma prova de conceito ou uma comparação experimental;
- Medidas: lista as medidas de desempenho usadas para avaliar a solução;
- Conjuntos de Dados: quais tipos de conjuntos de dados foram usados para validação e quantos foram;
- Operadores: quais tipos de operadores foram usados e quantos de cada tipo.

3.6 Principais Desafios da Seleção de Processo Completo

Seleção de Processo Completo enfrenta vários desafios que a tornam um problema difícil na prática:

1. Lidar com um espaço de busca crescente: a medida em que novos algoritmos (e hiper-parâmetros) vão sendo desenvolvidos, as soluções de SPC devem ser capazes

de suportar o acréscimo desses novos operadores, mantendo a premissa da automatização, ou seja, o mínimo de interferência;

2. Qualidade do Modelo Gerado: com a busca crescente por soluções de SPC é importante que as soluções se adequem as necessidades do usuário e ofereçam recomendações cada vez mais precisas. A maioria dos trabalhos garante o desempenho por experimentação direta, mas como mencionado anteriormente, nem todos o fazem. Nesse último caso, é preciso garantir a qualidade de outra maneira;

3. Avaliação dos Modelos Completos: funções de avaliação podem ser extremamente custosas para processos completos complexos (que envolvam diversos operadores e hiper-parâmetros) e/ou grandes conjuntos de dados.

No próximo Capítulo é apresentado como as soluções atuais lidam com esses desafios e quais as possibilidades em aberto.

3.7 Considerações Finais do Capítulo

Este Capítulo teve como foco a definição da área de Aprendizado de Máquina Automático, que é o conceito básico a ser explorado nesta tese. Embora o foco principal do trabalho seja Seleção de Processo Completo, a apresentação dos conceitos de Seleção e Configuração de Algoritmos permitiu com que o conceito principal pudesse ser melhor construído, a partir da composição dos demais, além de possibilitar o detalhamento das técnicas de Meta-Aprendizado e Otimização Baseada em Bandidos Multi-Armados.

O Capítulo 4 apresenta a análise completa dos trabalhos existentes voltados para apoio à SPC, identificados a partir de uma Revisão Sistemática da Literatura e avaliados de acordo com a Arquitetura Genérica de uma Solução para Apoio à Seleção de Processo Completo e discutidas em termos dos principais desafios da área, apresentados neste Capítulo.

4. SELEÇÃO AUTOMÁTICA DE PROCESSO COMPLETO: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Soluções para apoio à Seleção Automática de Processo Completo (SPC) visam tornar o processo de Descoberta de Conhecimento em Base de Dados (DCBD) mais acessível à usuários de Mineração de Dados (MD), em especial os não-especialistas na área, a partir a automatização da seleção e configuração do sub-conjunto de operadores (métodos de pré-processamento e algoritmos de aprendizado) com melhor desempenho esperado para um determinado problema. Dado aumento do volume de usuários dessas ferramentas e a constante evolução de suas necessidades, o desafio dos pesquisadores é aperfeiçoar tais soluções para que tenham resultados cada vez mais precisos. Com isso, torna-se necessário verificar o estado-da-arte nessa área de pesquisa, explicitando as técnicas já aplicadas e as possibilidades de melhoria.

Neste contexto, o objetivo deste Capítulo é identificar e analisar as soluções existentes para apoiar a SPC. As soluções identificadas foram caracterizadas de acordo com a Arquitetura Genérica de Componentes de Soluções de SPC, proposta no Capítulo 3. No geral, não apenas foi identificado o estado-da-arte nesta área de pesquisa, mas os desafios e possibilidades em aberto para novas pesquisas.

Para identificar as soluções que visam apoiar a SPC foi realizada uma Revisão Sistemática da Literatura [Kitchenham e Charters, 2007]. A principal diferença entre essa revisão e a publicação mais relacionada [Serban et al., 2013] é o foco da análise. Este último é focado em sistemas do tipo IDAs (do inglês, *Intelligent Discovery Assistants*). Por outro lado, este trabalho, concentra-se nas abordagens para desenvolver soluções para dar suporte à SPC.

O restante do Capítulo está organizado da seguinte forma: a Seção 4.1 apresenta o planejamento da revisão sistemática. A Seção 4.2 apresenta o processo de seleção de artigos. A Seção 4.3 avalia as soluções identificadas do ponto de vista de suas entradas e pré-requisitos. A Seção 4.4 apresenta as técnicas aplicadas para SPC. A Seção 4.5 detalha o apoio fornecido ao usuário por cada solução. A Seção 4.6 sintetiza a validação empírica realizada por cada trabalho. Por fim, a Seção 4.7 apresenta uma discussão geral dos trabalhos identificados.

4.1 Planejamento e Levantamento de Artigos por Revisão Sistemática da Literatura

A Revisão Sistemática da Literatura é caracterizada como uma maneira de identificar, avaliar e interpretar toda a pesquisa relevante disponível sobre uma questão de pes-

quisa em particular [Kitchenham, 2004]. É um estudo secundário que permite, dentre outras coisas, a sumarização das evidências existentes sobre uma determinada tecnologia e a identificação de *gaps* em uma área de pesquisa, que permitam a sugestão de novos temas a serem investigados nessa área, além de prover suporte para o posicionamento de novas atividades de pesquisa. O planejamento desta revisão sistemática está baseado no protocolo apresentado em [Kitchenham e Charters, 2007].

4.1.1 Objetivo e Questões de Pesquisa

O principal objetivo desta revisão é identificar as soluções existentes para apoio à Seleção de Processo Completo.

A Tabela 4.1 apresenta as questões de interesse da pesquisa, bem como a motivação para tais questões.

Tabela 4.1 – Questões de Pesquisa

QP	**Questão de Pesquisa**	**Motivação**
QP1	Quais são as soluções existentes voltadas para apoio à seleção de processo completo para DCBD?	Identificar as soluções de apoio à seleção de processo completo para DCBD
QP1.1	Quais são os tipos de abordagens utilizadas nas soluções existentes?	Identificar o tipo de tecnologia utilizada para construir a solução
QP1.2	Quais as fases do processo de DCBD apoiadas pela seleção?	Identificar as fases do processo de DCBD que são efetivamente atendidas pela solução, de acordo com as fases definidas por [Fayyad et al., 1996b]
QP2	Como é realizada a validação das soluções propostas?	Avaliar a evidência empírica obtida
QP2.1	Quais conjuntos de dados são utilizados?	Identificar se os conjuntos de dados utilizados são apropriados para avaliar a solução proposta
QP2.2	Quais as medidas de desempenho utilizadas para avaliar a solução?	Identificar os critérios objetivos de avaliação das soluções propostas
QP3	Quais os pontos fortes e fracos das soluções propostas?	Caracterizar as soluções propostas em termos de vantagens e desvantagens de utilização

A busca de artigos foi realizada em motores de busca e a partir da técnica de *snowballing* [Wohlin, 2014].

4.1.2 Artigo de Controle

Com o intuito de dar uma maior garantia aos resultados gerados pela *string* de busca e para iniciar o *snowballing*, foram selecionados artigos de controle, ou seja, artigos que devem aparecer nos resultados provenientes de todas as bases. Neste trabalho, tais artigos são [Escalante et al., 2009] [Bernstein et al., 2005] [Thornton et al., 2013]. Caso eles não apareçam nas buscas, a *string* de busca deve ser alterada. Estes artigos foram

escolhidos por serem amplamente referenciados como exemplos de solução de seleção de processo completo para DCBD e cuja técnica apresentada foi estendida, comparada e debatida em publicações no decorrer dos anos.

4.1.3 Busca em Motores de Busca

Os motores de busca selecionados para esta revisão foram: IEEEXplore, ACM Digital Library, Springer-Link, ScienceDirect, Web of Science e Scopus.

A *string* aplicada a estes motores de busca foi baseada na literatura sobre o tema e nas palavras-chave dos artigos de controle:

("machine learning"OR "classification"OR "regression"OR "knowledge discovery in databases"OR "knowledge discovery from data"OR "data mining"OR "knowledge mining"OR "knowledge mining from data"OR "Knowledge Discovery"OR "DCBD") AND ("algorithm"OR "approach"OR "solution"OR "process"OR "tool"OR "framework"OR "method") AND ("full model selection"OR "intelligent assistant"OR "intelligent discovery assistant"OR "workflow"OR "workflow planning"OR "knowledge discovery support system"OR "cash Problem"OR "automl"OR "machine learning pipeline"))

4.1.4 Busca por *snowballing*

Adicionalmente, a partir dos artigos de controle foi realizado um *snowballing*, com o objetivo de identificar trabalhos relacionados anteriores e posteriores que não tenham sido obtidos a partir da *string* de busca e que sejam relacionados à seleção de processo completo para DCBD. O procedimento adotado para esta fase está de acordo com o proposto por [Wohlin, 2014], e ilustrado na Figura 4.1. A fase de *Backward* foi realizada a partir da leitura direta dos metadados presentes nas referências dos artigos. Já a fase de *Forward* foi realizada com base no *Google Scholar* [1], que apresenta os artigos que citaram um determinado artigo.

4.2 Seleção de Artigos

A seleção de artigos foi dividida em duas fases. Na fase I, para a seleção dos estudos de interesse, os artigos foram avaliados de acordo com os dados do título, resumo e palavras-chave, sobre os quais foram aplicados os seguintes critérios de inclusão/exclusão:

[1] https://scholar.google.com.br/

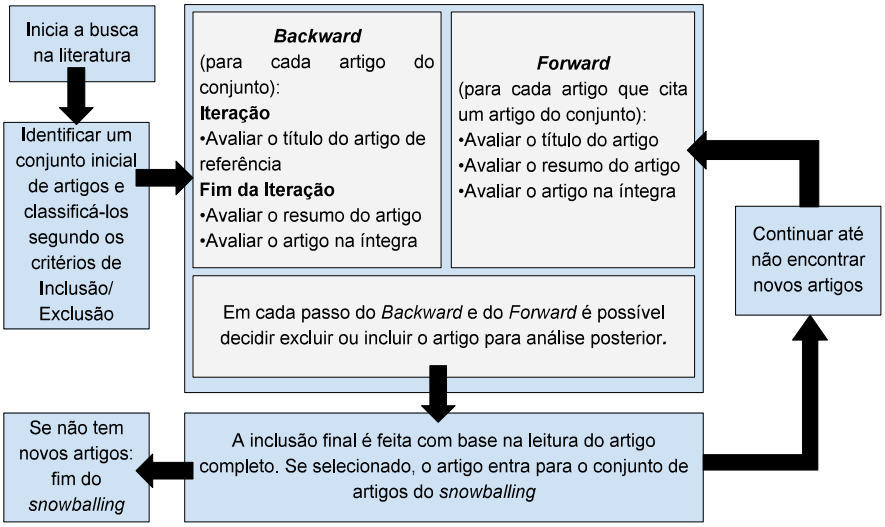

Figura 4.1 – Procedimento de *Snowballing*

1. Critério de inclusão dos estudos:

   (a) Artigos que apresentem soluções de seleção de processo completo para descoberta de conhecimento em bases de dados. As soluções propostas nos artigos podem incluir técnicas, métodos, modelos, estratégias, algoritmos ou qualquer outra iniciativa relacionada à SPC;

2. Critérios de exclusão dos estudos:

   (a) Trabalho que não estejam no formato de artigo completo (*full paper*);

   (b) Trabalhos relacionados a seleção de processo completo para DCBD, mas que não proponham soluções para realização de recomendação (por exemplo: trabalhos que façam apenas comparações entre soluções);

   (c) Trabalhos cujo conteúdo esteja relacionado ao apoio a uma única fase do processo de DCBD;

   (d) Trabalhos que unicamente indiquem os desafios e as direções a serem tomadas na área de pesquisa em questão (*roadmaps*).

Na fase II foi realizado o refinamento da revisão sistemática, a partir da leitura do texto completo dos artigos selecionados na fase I. Para cada artigo selecionado foram respondidos os critérios de avaliação de qualidade do estudo.

1. Abrangência do processo de DCBD: quantas/quais fases do processo de DCBD atende?
2. Caracterização da solução: classifica o tipo de abordagem (por exemplo, planejamento automático, meta-heurística, Meta-Aprendizado, outro)?
3. Validação: avalia o desempenho da solução (Sim / Não)?
4. Quantidade de processos completos gerados.

4.2.1 Resultado da Seleção de Artigos

A figura 4.2 mostra a quantidade de artigos selecionados a cada fase, tanto da busca por *string* quanto do *snowballing*, bem como a quantidade de artigos final.

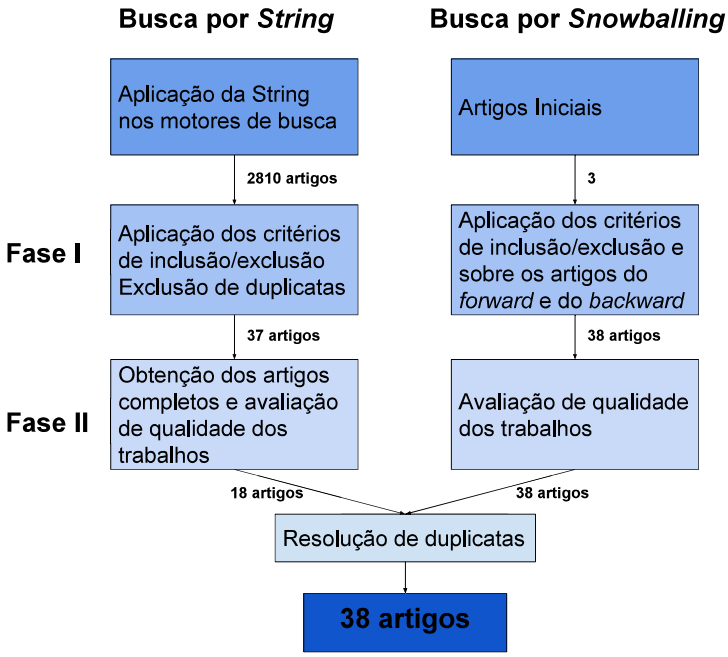

Figura 4.2 – Levantamento de artigos

Em relação a busca por *string*, o grande número de artigos encontrados na primeira fase deu-se pela nomenclatura empregada nos artigos de controle. Em um primeiro momento, foi elaborada uma busca com termos mais específicos, porém esta busca não retornou os artigos de controle. Após analisados, estes trabalhos levaram a termos mais genéricos (como "*model selection*", "*algorithm selection*", "*workflow*", dentre outros) que deixaram a *string* bastante ampla.

Já no *snowballing*, um ponto a se destacar foi o fato de que praticamente não houve interseção entre os artigos de referência/citação dos dois artigos de controle. Embora o problema atacado seja o mesmo, o que notou-se foi que os trabalhos com foco em planejamento automático, inspirados em [Bernstein et al., 2005], são desenvolvidos de forma independente dos artigos baseados em meta-heurística similares à [Escalante et al., 2009] e à otimização Bayesiana similares à [Thornton et al., 2013].

No geral, foram identificados 38 artigos que apresentam soluções para SPC. Uma vez que vários destes artigos tratam da mesma solução em diferentes estágios de desenvolvimento, os trabalhos identificados serão agrupados de acordo com as soluções. A Tabela 4.2 apresenta uma listagem das soluções, tipo de abordagem (MA - Meta-Aprendizado, MH-Meta-Heurística, PA - Planejamento Automático, RBC - Raciocínio Baseado em Casos, OB-Otimização Bayesiana, e possíveis combinações).

Tabela 4.2 – Soluções de Apoio à Seleção de Processo Completo

Sigla	Nome	Abordagem	Referências
HMDA	Hybrid Data Mining Assistant	RBC	[Charest et al., 2006, Charest e Delisle, 2006, Charest et al., 2008]
Auto-Weka	Auto-Weka	OB	[Thornton et al., 2013, Kotthoff et al., 2016]
Hyperopt	Automatic Hyperparameter Configuration for Scikit-Learn		[Komer et al., 2014]
MCPS	Automation of composition and optimisation of multicomponent predictive systems		[Salvador et al., 2016a, Salvador et al., 2016b]
PSMS	Particle Swarm Model Selection	MH	[Escalante et al., 2009]
BA-FMS	Bat Algorithm for Full Model Selection		[Bansal e Sahoo, 2015a, Bansal e Sahoo, 2015b]
GAPSO-FMS	Genetic Algorithm and Particle Swarm Optmization for Full Model Selection		[Sun et al., 2013, Sun, 2014]
NSGA2FMS	NSGA-II for Full Model Selection		[Pérez-Castro et al., 2016]
TPOT	Tree-Based Pipeline Optimization		[Olson e Moore, 2016]
IDEA	Intelligent Discovery Electronic Assistant	PA	[Bernstein et al., 2005]
NExT	Next-generation Experiment Toolbox		[Bernstein e Dänzer, 2007]
RMD	RMD		[Záková et al., 2008, Záková et al., 2009, Lavrac, 2008, Záková et al., 2011]
eProPlan/eIDA			[Kietz et al., 2009, Kietz et al., 2010b, Kietz et al., 2010a, Kietz et al., 2012, Kietz et al., 2014, Serban, 2010]
KDDVM	Knowledge Discovery in Databases Virtual Mart		[Diamantini et al., 2009]
PMD	Planning for Data Mining		[Fernández et al., 2009, Fernández et al., 2010]
Orange4WS	Orange4WS		[Podpečan et al., 2011]
e-LICO		MA-PA	[Nguyen et al., 2011, Hilario et al., 2011, Nguyen et al., 2012a, Nguyen et al., 2012b, Nguyen et al., 2014]
Auto-Sklearn		MA-OB	[Feurer et al., 2015a]
CARS	Collaborative Analytics Recommender System	MA-MH	[Chong et al., 2013]

As soluções serão avaliadas nas próximas seções com base nos componentes apresentados no arquitetura genérica dos componentes de soluções de SPC, na Seção 3.5.

4.3 Base de Conhecimento

A Tabela 4.3 mostra uma comparação entre as soluções existentes em termos da Base de Conhecimento usada.

Tabela 4.3 – Comparação entre as soluções - Dados de Entrada

Solução	Operadores	Conjunto de Dados	Desempenho	Descrição da Tarefa
HDMA	ESPE	66 meta-atributos (simples, estatísticos, baseados em TI e de negócio)	não	regras de especialistas
Auto-Weka	não	não	não	não
Hyperopt	não	não	não	não
MCPS	não	não	não	não
PSMS	não	não	não	não
BA-FMS	não	não	não	não
GAPSO-FMS	não	não	não	não
NSGA-II-FMS	não	não	não	não
TPOP v.03	não	não	não	não
IDEA	ESPE	não	não	tipo, objetivo
NExT	ESPE	não	não	objetivo
RDM	ESPE	não	não	objetivo
ePropPlan/eIDA	ESPE	não	não	tipo, objetivo
KDDVM	ESPE	não	não	tipo, objetivo
PDM	descrições de modelos	não	não	tipo, objetivo
Orange4WS	ESPE	não	não	tipo, objetivo
e-LICO	ESPE	150 meta-atributos (descrição simples, informação-teórica, estatística, processos completos)	Acurácia de 35 processos completos aplicados em 65 datasets	tipo, objetivo
Auto-Sklearn	não	38 meta-atributos (simples, teoria da informação e estatísticos)	Precisão do melhor processo completo para 140 conjuntos de dados	não
CARS	não	6 meta-atributos (*landmarking*)	Precisão de 40 processos completos para 7 conjuntos de dados	não

4.3.1 Operadores

As características dos operadores determinam como eles processam dados. Algumas dessas características são *propriedades* (por exemplo, capacidade de processar variáveis numéricas), enquanto outras são *conhecimento comportamental*, que emergem da combinação de fatores intrínsecos e extrínsecos, que são mais difíceis de identificar e expressar (por exemplo, capacidade de um algoritmo de aprendizagem de lidar com o ruído nos atributos-alvo). Ambos os tipos de características permitem a conexão entre as características dos conjuntos de dados e o comportamento dos operadores. Isso é válido para operadores de todas as fases do processo de DCBD.

O primeiro tipo de características, conhecido como propriedades, fornece diversas informações sobre os operadores, incluindo:

- Entrada: descreve os hiper-parâmetros envolvidos na execução do operador;

- Saída: descreve o tipo de saída dada a execução do operador;
- Pré-condições: descreve os requisitos e restrições para um operador ser utilizado;
- Efeito: descreve as mudanças nos dados após a aplicação do operador.

Este tipo de informação é usada em diferentes níveis de detalhe por diversas das soluções analisadas: HDMA, IDEA, NeXt, RDM, eProPlan/eIDA, KDDVM, PDM, Orange4WS e e-LICO.

Estas soluções fornecem ou usam ontologias onde os principais conceitos dos operadores são modelados manualmente, geralmente em OWL (do inglês, *Ontology Web Language*) [McGuinness et al., 2004]. No contexto da SPC, ontologias descrevem conceitos relacionados aos operadores, como: fase do processo de DCBD no qual o operador é aplicado, o conjunto de restrições e regras para representar conhecimento, tarefa-alvo (por exemplo, classificação, regressão), parâmetros, pré-condições e pós-condições.

O segundo tipo de características, conhecimento comportamental, geralmente resulta de uma análise empírica do comportamento dos operadores que compõe os processos completos. Nesse caso, algumas soluções [Bernstein et al., 2005] usam ontologias para modelar informações extras sobre os operadores, como desempenho relativo e tempo de execução, que são usadas para ordenar soluções para a exibição final, já que nem todas as soluções permitem a execução do processo completo recomendado para calcular o desempenho real.

4.3.2 Informações do Conjunto de Dados

As informações do conjunto de dados correspondem as informações extraídas do conjunto de dados de entrada para reunir descritores sobre a distribuição de dados que se correlacionam bem com o desempenho dos modelos de aprendizado. Esse tipo de informação é aplicada no contexto de soluções de Meta-Aprendizado [Brazdil et al., 2008] 3.2.1, onde é conhecida como meta-atributo.

Dentre os trabalhos analisados, HDMA e Auto-Sklearn utilizam meta-atributos diretos do tipo simples, estatísticos e baseados em teoria da informação. CARS faz uso de meta-atributos do tipo *landmarking*, neste caso, o pequeno número de características está relacionado ao custo computacional para calculá-las, que é superior aos outros tipos. e-LICO, faz uso da maior gama de informações do conjunto de dado, além de meta-atributos simples, estatísticos, baseados em teoria da informação e baseados em modelo, também são caracterizados os processos completos (meta-atributo alvo) com o objetivo de entender quais os aspectos de um modelo explicam seu desempenho esperado dado um conjunto de dados.

4.3.3 Desempenho do Operador

Poucas soluções usam informações sobre o desempenho passado dos processos completos(e-LICO, Auto-Sklearn, CARS e HDMA). Tais soluções armazenam esse tipo de informação em uma base de conhecimento ou repositório de casos.

4.4 Tipos de Métodos de Seleção de processo completo

Esta seção discute os métodos aplicados para oferecer suporte à SPC. Esses métodos podem ser organizados em quatro categorias principais, mostradas na Figura 4.3:

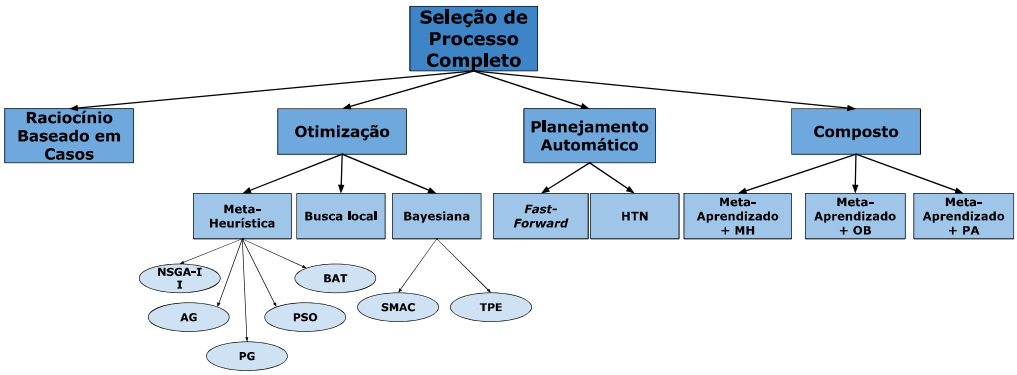

Figura 4.3 – Taxonomia das Abordagens de Seleção de Processo Completo

- Raciocínio Baseado em Casos (RBC): as soluções baseadas em RBC usam processos completos bem-sucedidos em problemas passados como recomendações para novos problemas. [Charest et al., 2006];
- Otimização: dado um espaço de busca pré-definido, as soluções de otimização visam identificar um processo completo que melhor minimize (ou maximize) uma determina função objetivo, geralmente relacionada com o desempenho deste processo sobre os dados de entrada;
- Planejamento Automático: soluções que abordam o problema da seleção de processo completo como um problema de planejamento, onde o resultado esperado é antecipado por um processo de deliberação explícito que escolhe e organiza ações;
- Compostas: soluções que combinam mais de uma abordagem. Atualmente, todas as abordagens compostas usam uma estratégia de Meta-Aprendizado para extrair o meta-conhecimento de uma coleção de conjuntos de dados e aplicar essas informações para recomendar processos completos semelhantes para um novo conjunto de

dados. Essas recomendações são usados como entrada para uma abordagem de otimização ou planejamento automático para acelerar o processo de pesquisa.

4.4.1 Raciocínio Baseado em Casos

Raciocínio Baseado em Casos (RBC) é uma área de pesquisa em aprendizado de máquina que aplica casos anteriores a problemas atuais, seguindo um processo de raciocínio analógico [Althoff, 2001]. O RBC é baseado em um modelo cognitivo centrado na memória. A ideia básica é que experiências passadas podem ser lembradas e adaptadas para orientar a resolução de problemas atuais. Em vez de criar uma solução do zero, casos semelhantes ao problema atual são recuperados da memória. A melhor correspondência é selecionada e adaptada para se adequar ao caso atual com base nas diferenças e semelhanças entre os dois casos [Kolodner, 1993]. A Figura 4.4 dá uma visão geral do processo RBC:

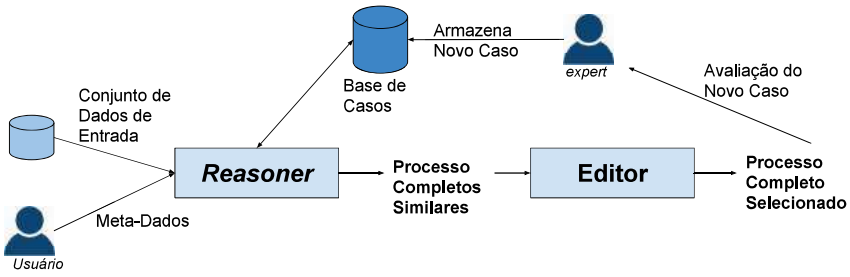

Figura 4.4 – Processo Geral de uma solução de Raciocínio Baseado em Casos

Na seleção de processo completo, um caso representa um processo completo bem-sucedido e é armazenado e mantido em uma *base de casos* por um especialista. O sistema *Hybrid Data Mining Assistant* (HDMA), proposto por Charest *et al.* [Charest et al., 2006, Charest e Delisle, 2006, Charest et al., 2008], é o único sistema RBC proposto para a SPC.

Entrada

A Seção 4.3 apresentou as entradas para cada solução de seleção de processo completo. Com relação aos sistemas RBC, a entrada mais importante é a base de casos.

Um sistema RBC [Kolodner, 1993] começa com um conjunto representativo de casos, que abrange as metas que surgem no raciocínio e tentativas bem-sucedidas e malsucedidas de atingir essas metas . Tentativas bem-sucedidas são usadas para propor soluções para novos problemas e tentativas mal-sucedidas são usadas para avisar sobre possíveis falhas. Essa base de casos é criada e mantida por especialistas, que avaliam os casos a serem inseridos na base continuamente, à medida que novos casos são criados. Essa

avaliação pode ser baseada na relevância do caso, qualidade da solução, distribuição sobre o espaço do problema, entre outros critérios. Além disso, esse tipo de sistema utiliza informações de entrada fornecidas pelo usuário, como caracterização da tarefa e detalhes sobre os operadores, que são utilizados para selecionar casos semelhantes.

A base de casos do sistema HDMA consiste em 3 casos, selecionados a partir de conjuntos de dados reais fornecidos por um departamento de apoio à decisão da universidade onde o sistema foi desenvolvido. Adicionalmente, o sistema HDMA utiliza como entrada 66 características do problema, extraídas do conjunto de dados de entrada (descritores simples, estatísticos, de teoria da informação e de negócios). O sistema HDMA também utiliza como entrada uma ontologia, denominada *Data Mining Ontology* (DM ontology), contendo tanto conhecimento declarativo quanto procedural para apoiar o processo de adaptação do caso. A ontologia define e gerencia conceitos de alto nível, como tarefas, tipos de atividades e algoritmos.

Reasoner

O componente *reasoner* recupera situações anteriores semelhantes à atual e as utiliza para ajudar a resolver o novo problema, ou seja, quando o usuário insere um novo problema, o sistema recupera os casos que possuem valores de similaridade relativamente altos. As técnicas de recuperação de casos incluem métodos matemáticos, busca por vizinhos mais próximos e métodos estatísticos de ponderação.

No sistema HDMA, 14 atributos discriminantes, selecionados a partir da caracterização do problema, são utilizados por um componente de recuperação (algoritmo k-NN, com $k = 5$) para identificar casos similares da base de casos.

Editor

Uma vez que o melhor processo completo é escolhido, entre casos semelhantes (por análise multi-objetiva, heurística de preferências ou análise de especialistas), é necessário adaptá-lo aos dados atuais. No problema de seleção de processo completo, esta adaptação permite adicionar ou substituir operadores DCBD. O novo processo completo é testado nos dados de entrada e, se aceito, é adicionado à base de casos.

No sistema HDMA, o usuário é responsável por selecionar o caso mais apropriado entre casos semelhantes sugeridos pelo sistema. Após isso, o sistema usa as características do problema atual para recomendar as adaptações que devem ser feitas no caso. Essa recomendação é apoiada pelas informações contidas na ontologia.

4.4.2 Planejamento Automático

O papel de uma solução baseada em Planejamento Automático (PA) é fornecer processos completos válidos por meio de raciocínio sobre a aplicabilidade dos operadores de DCBD em uma determinada etapa, de acordo com suas pré/pós-condições [Nguyen et al., 2012a]. As soluções baseadas em PA são capazes de construir processos completos autonomamente [Serban et al., 2013]. Um processo completo é exibido como um plano ou uma sequência de operadores para transformar o conjunto de dados inicial em resultados de análise de dados. Um sistema PA normalmente consiste em 3 componentes principais: as entradas (descrição do domínio e descrição da tarefa), um planejador e um classificador (Figura 4.5).

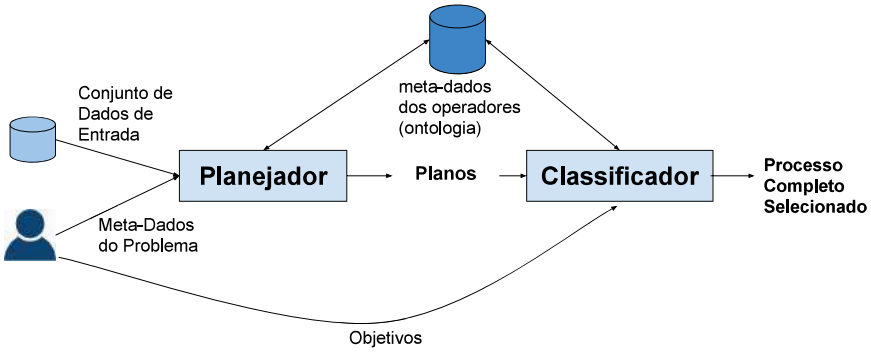

Figura 4.5 – Solução de Planejamento Automático para Seleção de Processo Completo. Adaptado de [Serban et al., 2013]

As soluções de PA para SPC são os sistemas: IDEA, NExT, RDM, eProPlan/eIDA, KDDVM, PDM e Orange4WS.

Descrição do Domínio

Além do conjunto de dados correspondente ao problema atual, algoritmos de planejamento requerem duas entradas principais [Záková et al., 2011]: i) a descrição do domínio, especificando os tipos disponíveis de objetos e ações, e ii) a descrição da tarefa, especificando o estado inicial, estado objetivo, os operadores disponíveis e objetivos do usuário, como medida de avaliação (precisão, tempo de execução, compreensibilidade, etc) e objetivo da tarefa (classificação, regressão, agrupamento, etc). A descrição da tarefa é detalhada na próxima seção.

A descrição do domínio é geralmente representada por uma ontologia e armazena os metadados de todos os operadores de DCBD disponíveis (métodos e algoritmos). Conforme descrito na Seção 4.3, essas informações podem ser: a fase do processo de DCBD no qual o operador é aplicado, o conjunto de restrições e regras para representar conhecimento, tarefa-alvo (por exemplo, classificação, regressão), parâmetros, pré-condições

e pós-condições (características específicas que uma entrada ou saída deve ter para ser usada por um método ou um algoritmo). Com exceção do sistema PDM, que usa um arquivo PMML (do inglês, *Predictive Model Markup Language*) de entrada contendo a descrição do domínio, as outras soluções usam ontologias com as informações dos operadores.

KDDONTO é a ontologia que suporta o sistema KDDVM, o conceito chave é o algoritmo. Outros conceitos fundamentais são: método, fase, tarefa, modelo, conjunto de dados, parâmetro, condições prévias, pós-condição, desempenho e função de otimização.

A ontologia utilizada pelo sistema IDEA contém, para cada operador, as pré- condições e pós- condições, efeitos, estimador dos efeitos do operador em atributos como velocidade, precisão, etc., e função de ajuda, com informações textuais sobre o operador. Os operadores são categorizados em operadores de pré-processamento, algoritmos de indução e pós-processamento.

KD Ontology, utilizada pelo sistema RDM e sistema Orange4WS, é composta por três conceitos principais: conhecimento (capturando os elementos declarativos no processo de DCBD), algoritmos (que servem para transformar uma instância de conhecimento em outra instância de conhecimento) e tarefa de descoberta de conhecimento (que implementa os conceitos relacionados aos processos completos).

O sistema eProPlan/eIDA usa duas ontologias: i) *Data Mining Ontology for Workflows* (DMWF) [Kietz et al., 2009] que contém dados dos operadores, objetivos, tarefas e métodos, bem como a decomposição de tarefas em métodos e operadores, e ii) a *Data Mining OPtimization ontology* (DMOP) [Hilario et al., 2009] que foca nas características dos operadores e é usada pelo planejador para otimizar os planos gerados.

Descrição da Tarefa

A descrição da tarefa é fornecida pelo usuário em um tipo de descrição do domínio de planejamento que é compreensível pelo planejador (por exemplo, PDDL (*Planning Domain Definition Language*) [McDermott et al., 1998]).

O sistema IDEA usa como descrição da tarefa os metadados, o objetivo da tarefa e os objetivos do usuário. Os autores não fornecem detalhes sobre como essas informações são descritas.

O sistema RDM é apresentado em duas versões, a versão 1 usa como entrada um arquivo PDDL com a descrição da tarefa baseada nos elementos da ontologia, a versão 2 usa apenas o conjunto de dados como arquivo de entrada.

As entradas do sistema KDDVM consistem no objetivo da tarefa e nas restrições do processo (restrições de tarefa) definidas pelo usuário.

No sistema PDM, assim como na descrição do domínio, a descrição da tarefa também é feita em um arquivo PMML fornecido como entrada, que é automaticamente traduzido em um problema de planejamento descrito em PDDL.

O sistema NExT usa uma base de conhecimento contendo todo o conhecimento de domínio e um sistema genérico de suporte à execução.

Por fim, o eProPlan/eIDA usa os objetivos do usuário como descrição da tarefa e o Orange4WS não usa arquivos de descrição de tarefa.

Planejador

Ao referenciar a descrição da tarefa, o planejador constrói processos completos válidos selecionando operadores apropriados com base em seus metadados (descrição do domínio).

O RDM, o Orange4WS e o PDM implementam algoritmos de planejamento baseados no sistema de planejamento Fast-Forward (FF) [Hoffmann e Nebel, 2001].

O sistema RDM implementa duas versões do algoritmo de planejamento FF: i) a primeira versão assume que todas as ações disponíveis são descritas através de um arquivo PDDL, criado em um estágio de pré-processamento usando a ontologia proposta e, neste caso, a ontologia não é usada durante o estágio de planejamento; ii) a segunda versão obtém estados vizinhos, combinando pré-condições de algoritmos disponíveis com condições atualmente satisfeitas, neste caso, *Pellet reasoner* [Sirin et al., 2007] é usado para consultar diretamente a ontologia.

O Orange4WS implementa o algoritmo de planejamento FF da mesma maneira que a versão 2 do sistema RDM, também usando o *Pellet reasoner* para consultar a ontologia.

O sistema PDM também usa uma solução baseada em pesquisa direta, denominada SAYPHI [De la Rosa et al., 2007].

O trabalho referente ao sistema IDEA não especifica o algoritmo de planejamento aplicado. Os autores descrevem apenas o processo de pesquisa direta em alto-nível: a sequência de operadores são construídas desde o estado inicial até o estado objetivo. Começando com um processo vazio, em cada estado, ele encontra os operadores aplicáveis usando as informações de compatibilidade fornecidas pela ontologia, adiciona cada operador ao processo parcial que o trouxe para o estado atual e transforma o estado usando os efeitos do operador. O planejador pára quando atinge o estado objetivo.

O sistema ePropPlan/ eIDA usa o algoritmo HTN (do inglês, *Hierarchical Task Network*) [Nau et al., 1998] como planejador. O problema de planejamento da HTN consiste em decompor a tarefa inicial, usando um método que contribui para as metas atuais, em operadores e sub-tarefas e, em seguida, decompor recursivamente cada sub-tarefa até

obter uma sequência de operadores aplicáveis. *Pellet reasoner* é usado para consultar a ontologia DMWF e traduzi-la para o planejamento.

O sistema KDDVM utiliza um algoritmo personalizado que começa no estado objetivo e adiciona iterativamente operadores usando um procedimento de correspondência de algoritmo, formando assim, um grafo direcionado até que o primeiro operador de cada processo completo seja compatível com o conjunto de dados fornecido.

Finalmente, o Sistema NExt aplica um planejamento e execução de iniciativa mista, ele usa as restrições do usuário para selecionar o caso mais similar, usando o *Pellet reasoner*. Depois disso, o sistema usa o conhecimento fornecido pela biblioteca de processos e pela biblioteca de entidades do domínio para suportar um processo de planejamento, que adapta o caso ao problema atual.

Classificador

O classificador ordena os processos completos de acordo com os objetivos do usuário (tempo de execução, precisão, etc.).

O sistema KDDVM fornece quatro critérios para apoiar o usuário na escolha entre os processos completos gerados: medição de similaridade, relaxamento de pré-condição, uso de módulos de link, avaliação de desempenho. Depois que o usuário seleciona um critério, o sistema recomenda o processo completo que atende a esse critério.

O sistema IDEA produz um classificador heurístico para classificar os processos completos adequados, em uma ordem baseada em escolhas do usuário.

O sistema PDM não possui um componente de classificação. Todos os planos gerados são convertidos em processos completos para serem executados em uma ferramenta de Mineração de Dados. Os arquivos de saída com as instâncias de processos completos e informações estatísticas, como tempo de execução e precisão, são retornados ao usuário.

O sistema RDM também não possui um componente de classificação. Em contraste com o PDM, no sistema RDM, os planos gerados (chamados processos completos abstratos e representados em um grafo acíclico direcionado) são apresentados ao usuário, que escolhe quais são instanciados para processos completos executáveis.

No sistema eProplan/eIDA, enquanto a ontologia DMWF é utilizada para fornecer informações para o processo de planejamento, a ontologia do DMOP é utilizada para suportar o processo de classificação, pois armazena informações detalhadas dos operadores.

O sistema NExT não aplica um classificador, porque seleciona apenas o caso mais similar para corrigir com o planejador.

4.4.3 Otimizadores

Soluções de otimização estudam como maximizar ou minimizar um determinada função objetivo.

A Figura 4.6 apresenta uma visão geral do processo de seleção de processo completo por um otimizador. Usando apenas o conjunto de dados de entrada, um componente de seleção (seletor) identifica as soluções candidatas, enquanto o componente de avaliação (avaliador) seleciona as melhores soluções entre as candidatas, com base em uma função de objetivo predefinida. Esse processo ocorre iterativamente até que um critério de parada seja satisfeito (limite de desempenho, número de iterações, etc.).

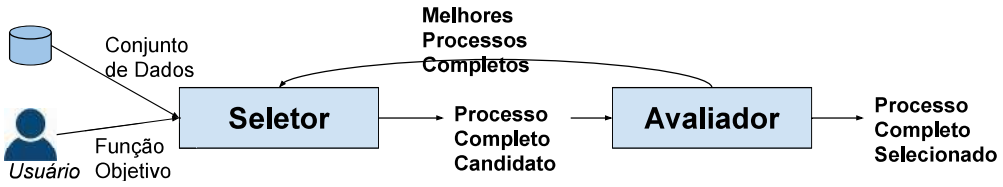

Figura 4.6 – Solução de Otimização para Seleção de Processo Completo

Meta-Heurísticas

Uma meta- heurística (MH) é um algoritmo geral projetado para encontrar soluções aproximadas para uma ampla gama de problemas de otimização difíceis e para os quais não há algoritmo específico satisfatório para resolvê-los[Boussaïd et al., 2013].

As meta-heurísticas visam equilibrar a exploração (*exploration*) e o aproveitamento (*exploitation*). A exploração é necessária para identificar partes do espaço de pesquisa com soluções de alta qualidade. O aproveitamento é importante para intensificar a busca em algumas áreas promissoras da experiência de pesquisa acumulada. As diferenças mais importantes entre as MH existentes dizem respeito à maneira particular pela qual elas tentam alcançar esse equilíbrio [Gendreau e Potvin, 2010].

No contexto de seleção de processo completo, cinco soluções foram identificadas usando meta-heurísticas: PSMS, GAPSO-FMS, N2FMS, BA-FMS e TPOP v0.3.

1. Seletor:

   - PSMS usa o algoritmo *Particle Swarm Optimization* (PSO) [Kennedy, 2011] para explorar o espaço de pesquisa do processo completo. Uma solução candidata (processo completo) é representada por dois componentes, posição e velocidade. O valor de velocidade é usado para explorar o espaço de pesquisa. O conjunto de soluções candidatas é inicializado aleatoriamente, o *fitness* de cada

solução é avaliado, o melhor local (melhor posição de uma solução) e o melhor global (melhor posição entre todas as soluções) são inicializados. Em cada iteração, a posição e a velocidade de cada solução são atualizadas e as melhores são identificadas.

- A solução GAPSO-FMS combina o Algoritmo Genético (AG) [Holland, 1992] com o PSO para seleção de processo completo. Nessa solução, o AG é usado para pesquisar a combinação de operadores ideais e o PSO é usado para otimização de hiper-parâmetros. Um conjunto inicial de processos completos (população) é gerado aleatoriamente. Em seguida, para cada iteração do AG (geração), o PSO é usado para pesquisar os hiper-parâmetros ideais para cada processo completo (semelhante ao PSMS). Após a avaliação do PSO, os operadores do AG, como cruzamento e mutação, são aplicados para gerar novos processos completos que são usados para substituir os processos completos com pontuações de avaliação mais baixas. O processo completo com a melhor pontuação de avaliação é retornado.
- O N2FMS utiliza o algoritmo Non-dominated Sorting Genetic Algorithm (NSGA-II) [Deb et al., 2002]. Uma população de indivíduos é gerada aleatoriamente. A função objetivo é calculada para cada indivíduo e é comparada com as funções objetivo de todos os indivíduos. Portanto, uma classificação é feita em diferentes frentes, o conjunto de soluções é conhecido como conjunto ótimo de Pareto, e seus valores de função de objetivo correspondentes são conhecidos como a frente de Pareto. Após a classificação, cada indivíduo recebe uma distância de superposição na frente. Um procedimento de seleção de torneio binário é usado para preencher a população pai. A seleção é baseada na classificação e, se dois indivíduos tiverem a mesma classificação, a seleção é baseada na distância de superlotação. A população de descendentes é criada pela realização de operações genéticas (cruzamento e mutação). Uma população intermediária é gerada, que consiste na união da população atual e população de descendentes da geração atual. Finalmente, uma substituição elitista é realizada para selecionar os indivíduos para formar a população para a próxima geração. Ele itera até que um número máximo de gerações seja atingido.
- O BAT-FMS usa o Algoritmo Bat (AB) [Yang, 2010] para seleção de processos completos. O AB é baseado no comportamento de ecolocalização do morcego (do inglês, *bat*) para sentir a distância dos alimentos ou obstáculos. Os morcegos liberam um som de pulso muito alto e avaliam o eco que retorna para calcular a distância do alvo a partir deles. Na implementação do algoritmo, um conjunto inicial de morcegos (processos completos) é inicializado aleatoriamente. Esta população é avaliada e classificada de acordo com a função objetivo. A população classificada contém frentes, com os indivíduos na primeira frente sendo total-

mente não dominantes em relação aos indivíduos na segunda frente. Um valor de classificação, 1 e 2, é atribuído aos indivíduos na primeira e segunda frente, respectivamente. Uma distância de superlotação (distância entre um indivíduo e seu vizinho) é calculada para cada indivíduo. Usando o valor de classificação e a distância de aglomeração, a seleção do torneio binário é feita para selecionar os pais da população, com base em dois critérios, a classificação é menor e o apinhamento é maior. A população é atualizada. A população recente e a população de descendentes são classificadas usando o tipo não dominante. Ele itera até que um número máximo de iterações seja atingido.

- TPOT v0.3 usa um algoritmo de Programação Genética (PG) [Koza, 1992]. Os operadores que compõem o processo são tratados como primitivas de PG e as árvores PG são construídas a partir deles. Para começar, o algoritmo de PG gera processos completos aleatórios baseados em árvore e avalia sua precisão realizando validação-cruzada no conjunto de dados de entrada. Para cada geração, o algoritmo seleciona os principais processos completos na população de acordo com o esquema de seleção NSGA-II [Deb et al., 2002]. Cada um dos processos completos selecionados produz cópias (ou seja, descendentes) na população da próxima geração, os procedimentos de cruzamento e mutação são feitos. A cada geração, o algoritmo atualiza uma frente de Pareto das soluções não dominadas descobertas em qualquer ponto da execução da PG. O algoritmo repete esse processo avaliar-selecionar-cruzar-mutar por gerações, quando o algoritmo seleciona o melhor processo completo da frente de Pareto.

2. Avaliador:

    - A função objetivo na solução PSMS é minimizar a taxa de erro balanceada (BER) do processo completo. Assim, em cada iteração, cada partícula é avaliada para verificar o quão distante está da solução ótima (BER mínimo) no espaço de busca;
    - A solução GAPSO-FMS utiliza como função objetivo a maximização do desempenho da área sob a curva ROC (AUC) na análise experimental, mas é adaptável a outras medidas como precisão, tempo de execução, etc;
    - Em N2FMS, BAT-FMS e TPOP v0.3 o problema de seleção de processo completo é definido como um problema de função multi-objetivo;
    - No N2FMS o CVER (*Cross Validation Error Rate*) e tempo de execução de cada processo candidato são considerados como objetivos a serem otimizados através do algoritmo NSGA-II. No final do processo de otimização multi-objetivo, um conjunto de soluções de Pareto é obtido, onde todas as soluções podem ser consideradas igualmente desejáveis em um sentido matemático;

- Na solução BAT-FMS o objetivo principal é aumentar o desempenho da classificação e o segundo objetivo é diminuir o número de atributos no conjunto de dados de entrada;
- Por fim, no TPOT v0.3 os processos completos são selecionados para maximizar simultaneamente a precisão da classificação no conjunto de dados de entrada enquanto minimiza o número de operadores.

Otimização Bayesiana

A Otimização Bayesiana (OB), também conhecida como otimização baseada em modelo sequencial, é um método iterativo para resolver problemas de otimização de caixa-preta [Brochu et al., 2010], e foi apresentada na Seção 3.3

Atualmente, três soluções abordam o problema de seleção de processo completo com algoritmos de otimização bayesiana: Auto-Weka, Hyperopt e MCPS.

Todas as soluções são baseados no SMBOA (do inglês, *Sequential Model- Based Optimization*) [Hutter et al., 2011], uma estrutura de otimização estocástica que pode trabalhar explicitamente com hiper-parâmetros categóricos e contínuos. O SMBO constrói um modelo probabilístico que captura a dependência da função objetivo nas configurações de processo completo. Em seguida, realiza uma iteração onde o modelo gerado é usado para determinar uma configuração candidata promissora para avaliar em seguida, avalia a configuração, e atualiza o modelo com o novo ponto de dados obtido.

Os algoritmos disponíveis no SMBO mais proeminentes são o *Sequential Model-Algorithm Configuration* (SMAC) [Hutter et al., 2011] e o *Tree- structured Parzen Estimator* (TPE) [Bergstra et al., 2011], que são os usados pelas soluções de SPC.

1. Seletor:
   - Auto-Weka usa o SMAC e o TPE para resolver o problema de seleção de processo completo. Como função de aquisição, usa *positive expected improvement* (EI). Para o SMAC, ele usa modelos Random Forest, pois eles funcionam bem com dados de entrada discretos e de alta dimensionalidade;
   - O Hyperopt usa o TPE, mas os autores apontam que a solução é flexível para suportar outros algoritmos (como o SMAC);
   - O MCPS usa os algoritmos SMAC e TPE. Inicialmente, eles apresentam o problema de seleção de processo completo como um problema de sistema preditivo de múltiplos componentes e representam o processo completo como uma Rede de Petri [Murata, 1989] denominada WA-WF-net (Rede de Petri de Fluxo de Trabalho Acíclico e Bem Tratado).
2. Avaliador:

- A função objetivo do Auto-Weka, Hyperopt-Sklearn e MCPS é minimizar a taxa de erro de classificação.

4.4.4 Sistemas Compostos

Soluções compostas combinam várias abordagens para resolver o problema de seleção de processo completo. A Figura 4.7 apresenta uma visão geral do processo de solução composta, em que todas as soluções incluem um estágio de Meta-Aprendizado.

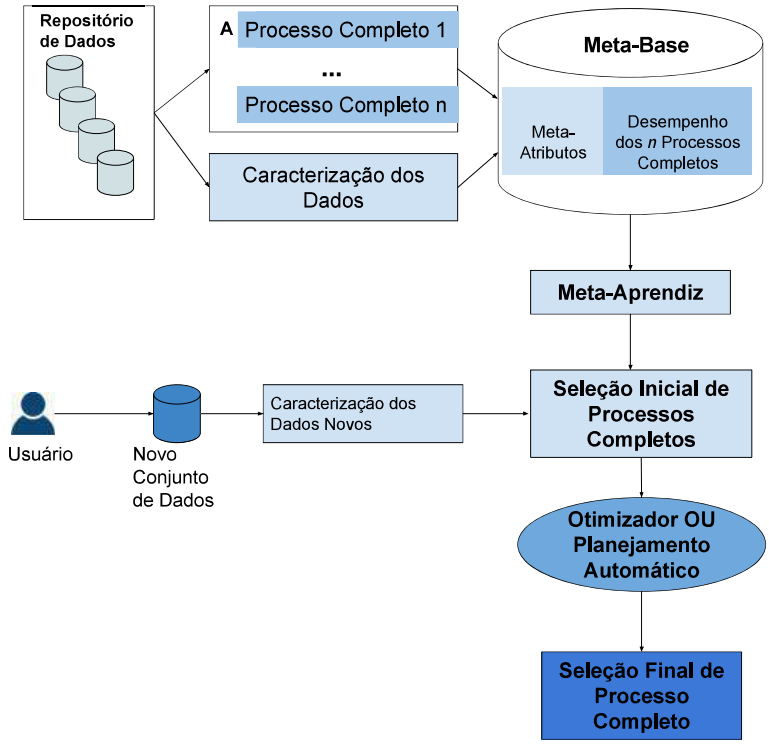

Figura 4.7 – Solução Composta para Seleção de Processo Completo

A Figura 4.7 fornece uma visão geral do processo de Meta-Aprendizado tal qual apresentado na Seção 3.2.1. Neste contexto, os processos completos recomendados pelo Meta-Aprendizado são usados como entrada para um algoritmo de planejamento automático, de meta-heurística ou otimização Bayesiana, que efetivamente farão a recomendação final do processo completo apropriado ao problema em questão. A recomendação inicial é útil para reduzir o espaço de busca de processos completos, uma vez que identifica bons candidatos, de acordo com o meta-conhecimento acumulado na meta-base.

Três soluções compostas foram identificadas: e-LICO, Auto-Sklearn e CARS. Essas soluções são descritas em termos de seus componentes de Meta-Aprendizado e otimização.

Fase de Meta-Aprendizado

No e-LICO IDA, um DMER (*Data Mining Experiment Repository*) armazena todos os meta-dados usados no processo de Meta-Aprendizado (conjuntos de dados de treinamento, seus meta-atributos, e processos completos com seus resultados de desempenho). Dado um novo problema (um conjunto de dados associado a uma tarefa), o componente de Meta-Aprendizado extrai as características dos dados e usa as informações fornecidas pelo DMER para aprender um modelo de meta-mineração que associa conjuntos de dados e características de processos completos, do ponto de vista de otimização de desempenho.

O Auto-Sklearn usa Meta-Aprendizado para inicializar um algoritmo de Otimização Bayesiana. Na meta-base cada problema é caracterizado por 38 meta-atributos, e pelo processo completo com maior desempenho empírico determinado por algoritmo de otimização Bayesiana. Dado um novo conjunto de dados, as meta-atributos são computados e os 25 problemas mais similares no repositório são identificados pela distância L1. Os processos completos associados a esses problemas são utilizados para inicializar o algoritmo de otimização.

A solução CARS implementa um mecanismo de recomendação que utiliza um repositório de processos completos de estoque e seus conjuntos de dados associados para buscar por modelos promissores para um dado conjunto de dados. A busca é feita por um algoritmo baseado em instâncias (k-NN) e a similaridade é definida em termos de características dos conjuntos de dados. Essas características são meta-atributos do tipo *landmarking*. Para cada conjunto de dados de entrada são definidos 3 vizinhos mais próximos, usando distância Euclidiana. Um SVM é aplicado para predizer a acurácia dos modelos para o conjunto de dados de entrada, e os k melhores modelos são selecionados para serem otimizados.

Fase de Otimização

A SPC do e-LICO é construída sobre um algoritmo de planejamento automático cooperativo que constrói planos seguindo a abordagem de planejamento HTN, apresentada na solução e-proplan/e-IDA (Seção 4.1). A cada etapa do planejamento, o planejador gera uma lista de ações válidas, e dos processos completos candidatos parciais, que é passada para ser classificada pelo componente de Meta-Aprendizado (ordenada pelo desempenho esperado). O componente de Meta-Aprendizado classifica a lista usando o meta-modelo e o planejamento continua com os melhores processos completos candidatos até a tarefa ser resolvida. Ao final, os k melhores modelos são apresentados ao usuário, ordenados pelo desempenho esperado.

Auto-Sklearn usa o algoritmo SMAC [Hutter et al., 2011] como o algoritmo de otimização bayesiana, similarmente as soluções Auto-Weka e TPOT. Além da Otimização Baye-

siana, os autores propõem a construção automática de *ensembles* usando os modelos descartados durante o processo de otimização.

O CARS aplica o Algoritmo Genético (AG) para melhorar os processos completos recomendados por meio da evolução iterativa. O AG é aplicado aos *k* melhores processos completos recomendados pela fase de Meta-Aprendizado. Estes *k* modelos passam por diversos ciclos de seleção, cruzamento e mutação, guiados na direção de maior adequação (alguma medida do desempenho de previsão), até que o algoritmo converge para uma função de adequação.

4.5 Apoio à Tomada de Decisão

A Tabela 4.4 apresenta uma comparação entre as soluções em termos do tipo de suporte ao usuário para seleção de processo completo.

4.5.1 Fases do Processo de DCBD

Uma solução ideal para Seleção de processo Completo deveria apoiar todas as fases do processo de DCBD. No entanto, a primeira e a última fases (Seleção de Dados e Interpretação de Dados) geralmente envolvem uma grande quantidade de informações subjetivas específicas do domínio, o que dificulta sua automação. Assim, as fases comumente suportadas são aquelas envolvendo métodos e algoritmos que podem ser aplicados automaticamente (ou seja, pré-processamento de dados, transformação de dados, mineração de dados e avaliação de dados).

Dentre as soluções identificadas, a tarefa mais apoiada é a redução de dimensionalidade (na fase de Transformação de Dados). Esta tarefa é subdividida em seleção de características (*Feature Selection*) e extração de características (*Feature Extraction*), e sua popularidade está relacionada com à maldição da dimensionalidade [Bellman, 1966], uma vez que dados com alta dimensionalidade apresentam um dos desafios mais proeminentes aos métodos de AM existentes [Yang e Wu, 2006]. Todas as soluções suportam a seleção de recursos e/ou a extração de recursos.

Como nenhuma solução suporta a fase de Seleção de Dados, também é interessante destacar que elas trabalham a partir de dados já estruturados, ou seja, não lidam com dados brutos.

Tabela 4.4 – Comparação de Soluções - Tipo de Apoio

Método	Solução	Fases DCBD	HP	Tarefa	Recomendação	Execução	Implementação
RBC	HMDA	PD/TD/MD	não	classificação, regressão	ranking	não	sistemas GUI
OB	Auto-Weka	TD/MD	sim	classificação, regressão	melhor	sim	framework
	Hyperopt	TD/MD	sim	classificação	melhor	sim	framework
	MCPS	PD/TD/MD	sim	classificação	melhor	sim	framework
MH	PSMS	TD/MD	sim	classificação	melhor	sim	framework
	BA-FMS	TD/MD	não	classificação	melhor	não	framework
	GAPSO-FMS	PD/TD/MD	sim	classificação	melhor	sim	framework
	NSGA-II-FMS	TD/MD	sim	regressão	melhor	sim	framework
	TPOP v.03	TD/MD	sim	classificação	melhor	sim	framework
PA	IDEA	TD/MD/DE	não	classificação, regressão	ranking	sim	sistemas GUI
	NExT	TD/MD	não	classificação	ranking	sim	sistemas GUI
	RMD	TD/MD	não	classificação	melhor	não	sistemas GUI
	ePropPlan/eIDA	TD/MD	não	classificação, regressão	ranking	sim	sistemas GUI
	KDDVM	TD/MD/DE	não	classificação, agrupamento	ranking	não	sistemas GUI
	PMD	TD/MD	não	classificação, regressão, agrupamento	todos	sim	sistemas GUI
	Orange4WS	TD/MD	não	classificação	todos	não	sistemas GUI
MA+PA	e-LICO	TD/MD	não	classificação	ranking	sim	sistemas GUI
MA+OB	Auto-Sklearn	TD/MD	sim	classificação	melhor	sim	framework
MA+MH	CARS	TD/MD	não	classificação	ranking	sim	framework

4.5.2 Otimização de Hiper-Parâmetros

Os hiper-parâmetros desempenham um papel importante na maioria dos algoritmos no processo de DCBD e defini-los manualmente consome tempo principalmente por dois motivos: (1) normalmente há vários hiper-parâmetros que podem assumir muitos valores (com hiper-parâmetros contínuos representando o caso extremo de um domínio infinito), e (2) eles são tipicamente validados usando Validação Cruzada [Salvador et al., 2016b].

Além disso, no contexto do problema de SPC, a contribuição de um único método e dos valores correspondentes do hiper-parâmetro para o desempenho do processo completo depende da escolha dos outros métodos no processo completo e de seus valores de hiper-parâmetro. Em outras palavras, a escolha dos componentes do processo completo e os

valores de seus hiper-parâmetros não podem ser feitos separadamente, o que faz com que o tamanho do espaço de busca cresça exponencialmente.

Devido ao tamanho do espaço de pesquisa, muitas abordagens ignoram o problema de selecionar valores para os hiper-parâmetros dos métodos no processo completo, usando um conjunto de parâmetros padrão. Notamos que apenas soluções baseadas em meta-heurísticas e em otimização Bayesiana trabalham com otimização de hiper-parâmetros. Isso se deve à capacidade dos otimizadores usados (PSO, GA, SMAC, TPE, PG) de lidarem com um grande espaço de busca em um tempo viável.

4.5.3 Tarefa

Assim como no contexto de AM em geral, a maioria das abordagens para a SPC está focada na tarefa de classificação.

As exceções incluem o NSGA-II-FMS, que trata da seleção de processo completo para problemas de estimativa em dados de séries temporais; HDMA, Auto-Weka, GAPSO, IDEA, eProPlan e PDM que suportam regressão; e PDM e KDDVM que também fornecem suporte para a tarefa de agrupamento.

No entanto, é importante observar que, em muitos casos, os métodos subjacentes são adequados para tarefas diferentes da tarefa usada na implementação da ferramenta ou na avaliação empírica.

4.5.4 Tipo de Recomendação

Algumas das abordagens propostas simplesmente identificam todos os processos completos compatíveis com as restrições do problema. Esta é uma recomendação muito completa, que deverá incluir a melhor alternativa possível. No entanto, ela não fornece nenhuma orientação sobre como o usuário final deve prosseguir com os experimentos. Dado o número de soluções geralmente fornecidas por essas abordagens, elas não são muito úteis. Eles incluem as soluções baseadas puramente em Planejamento Automático, PDM e Orange4WS.

Alternativamente, algumas abordagens ordenam os processos completos de acordo com o desempenho esperado (por exemplo, tempo de execução ou precisão). Assim, nesses casos, o usuário é apresentado a um *ranking* de alternativas, que pode ser usado para guiar os experimentos (por exemplo, começar com o primeiro, depois o segundo e assim por diante). HDMA, IDEA, NExT, ePropPlan, KDDVM, e-LICO e CARS oferecem várias possibilidades de processos completos ordenadas de acordo com algum critério de quali-

dade. Este critério é, em geral, o desempenho preditivo do modelo gerado. Alguns deles, como IDEA, eProPlan e e-LICO, também oferecem a possibilidade de alterar os critérios para outros, como o tempo de execução.

Finalmente, algumas abordagens recomendam um único processo completo, ou seja, espera-se que o único produza o melhor processo completo para um determinado problema, de acordo com um critério de qualidade. Por ser um método de recomendação simples, o principal problema associado é a ausência de alternativas ao processo completo recomendado, caso ele não tenha o desempenho esperado ou até mesmo se não for executado. Este é o tipo de recomendação fornecido pelas abordagens restantes.

4.5.5 Execução

Algumas abordagens fornecem uma recomendação em suporte para executá-la. Estas incluem a meta-heurística BA-FMS, bem como a solução de Planejamento Automático NExT.

Outras abordagens fornecem mecanismos para executar os processos completos recomendados. Essas soluções podem implementar os próprios operadores (métodos e algoritmos) ou podem usar uma biblioteca MD existente. As soluções restantes estão no último caso. Eles usam ferramentas de MD bem conhecidas, incluindo Weka (Auto-Weka, PDM, MCPS Automático, GAPSO-FMS, PSMS), Scikit-learn (Auto-Sklearn, hyperopt), Rapid Miner (CARS, e-ProPlan, e-LICO).

4.5.6 Tipo de Implementação da Solução

Em relação ao tipo de implementação, categorizamos as soluções em dois tipos: sistemas com interface gráfica para o usuário (sistemas GUI, do ingês, *Graphics User Interface*) ou *frameworks*.

No caso de um sistema GUI, a solução fornece uma interface gráfica completa para interação do usuário, que a configura de acordo com suas necessidades. A execução é transparente para o usuário.

Frameworks são geralmente projetos de código aberto. Eles são integrados com ferramentas existentes, o que significa que o usuário precisa ter as ferramentas de software necessárias disponíveis de acordo com a linguagem de programação (por exemplo, Java e Python) e a biblioteca de MD (como Weka e Scikit learn) usada. Além disso, é necessário configurar manualmente a ferramenta de acordo com a aplicação.

No geral, há um equilíbrio entre os tipos de solução. Podemos observar que todas as meta-heurísticas e otimização Bayesiana são *frameworks*, mesmo se combinadas com Meta-Aprendizado. As abordagens restantes (Planejamento Automático e RBC) estão disponíveis como um sistema completo.

4.6 Validação

A Tabela 4.5 apresenta um resumo da validação realizada pelas soluções.

4.6.1 Tipo de Validação

Em geral, os trabalhos apresentaram pouca validação das soluções. Nenhuma validação é apresentada para a solução Orange4ws. Entre os trabalhos que realizam comparação experimental, apenas alguns baseados em meta-heurísticas e Otimização Bayesiana fornecem comparações com outras soluções: PSMS, BA-FMS, NSGA-II-FMS são comparados com algoritmos que geram modelos simples (suportam apenas a fase MD no processo de DCBD) em termos de taxa de erro. Auto-Weka, Hyperopt e TPOP v0.3 apresentam uma comparação com *baselines* simples como seleção aleatória. O GAPSO-FMS, o MCPS e o Auto-Sklearn foram os únicos que foram comparados com outras soluções de SPC.

4.6.2 Medidas de Desempenho

Várias medidas de desempenho podem ser aplicadas para avaliação de processos completos. No entanto, a escolha da medida mais apropriada para um determinado contexto dependerá principalmente das características do problema em análise e do objetivo final da avaliação.

Apesar do leque de medidas de avaliação existentes, a maioria dos trabalhos utiliza apenas erro de classificação e/ou tempo de execução como forma de avaliar o desempenho. O PSMS, Auto-Sklearn e GAPSO levam em conta o desbalanceamento entre classes nos conjuntos de dados usados e aplicam medidas que suavizam esse efeito.

Tabela 4.5 – Comparação de Soluções - Validação

Abordagem	Solução	Validação	Medidas	Dados
RBC	HDMA	demonstração de usabilidade	-	1 conjunto de dados
OB	Auto-Weka	comparação experimental	erro de classificação	21 conjunto de dados públicos
	Hyperopt	demonstração de desempenho	medida-F	3 conjunto de dados públicos
	MCPS	comparação experimental	erro de classificação	21 conjunto de dados públicos; 7 indústria química
MH	PSMS	comparação experimental	taxa de erro balanceada	9 conjunto de dados públicos
	BA-FMS	comparação experimental	erro de classificação	5 Expressão gênica conjunto de dados
	GAPSO-FMS	comparação experimental	AUC	10 conjunto de dados públicos
	NSGA-II-FMS	comparação experimental	erro de validação cruzada	8 conjunto de dados de séries temporais
	TPOP v.03	comparação experimental	acurácia balanceada	150 conjunto de dados públicos
	IDEA	demonstração de desempenho	tempo de execução, erro de classificação	24 conjunto de dados públicos
PA	NExT	demonstração de usabilidade	-	-
	RDM	demonstração de desempenho	tempo de execução	2 conjunto de dados públicos
	ePropPlan/eIDA	demonstração de desempenho	tempo de execução	dados públicos: 117 classificação; 47 regressão
	KDDVM	use demonstration	-	-
	PDM	demonstração de desempenho	erro de classificação	20 conjunto de dados públicos
	Orange4WS	demonstração de usabilidade	-	-
MA+PL	e-LICO	demonstração de desempenho	erro de classificação	65 conjunto de dados biológicos
MA+OB	Auto-Sklearn	comparação experimental	erro de classificação, erro de teste balanceado	140 conjunto de dados públicos
MA+MH	CARS	demonstração de desempenho	erro de classificação	7 conjunto de dados públicos

4.6.3 Conjunto de Dados

A maioria das abordagens usa dados públicos disponíveis na Biblioteca UCI [2], com exceção do estudo empírico sobre o e-LICO, que usa os conjuntos de dados de expressões gênicas fornecidos pelo NCBI (*National Center for Biotechnology Information*) [3].

[2] http://archive.ics.uci.edu/ml/index.php
[3] https://www.ncbi.nlm.nih.gov/

O tamanho dos conjuntos de dados varia de 23 a 32.561 instâncias e 3 e 54.675 atributos. Alguns conjuntos de dados contêm valores ausentes.

4.7 Discussão

As seções anteriores apresentaram um detalhamento das soluções existentes para SPC separadamente. De maneira a realizar uma discussão geral dessas soluções, os seguintes aspectos são considerados: extensibilidade e custo benefício do processo completo gerado, onde será avaliada a qualidade da recomendação gerada em balanceamento com a demanda de recursos computacionais para fazê-lo. Esses aspectos estão alinhados com os principais desafios da área, apresentados na Seção 3.6.

4.7.1 Extensibilidade do Espaço de Busca

Toda solução suporta um determinado número de operadores para SPC. Com o desenvolvimento constante de novos métodos de pré-processamento e algoritmos de aprendizado, uma característica importante da solução é permitir que novos operadores sejam adicionados facilmente, com o mínimo de interferência humana.

Soluções baseadas em otimizadores são favorecidas em termos de extensibilidade, uma vez que não utilizam uma base de conhecimento no processo de recomendação. Assim, para aumentar o espaço de busca, basta acrescentar ou retirar os operadores no espaço de busca já suportado, e atualizar as restrições de combinações entre eles.

Já nas soluções baseadas em RBC, para cada novo operador adicionado devem ser adicionados casos de sucesso (processos completos) que utilizaram este método, o que é parcialmente dependente do usuário, que deve avaliar a validade/ sucesso dos casos antes de acrescentá-los. Da mesma forma, as soluções que envolvem Planejamento Automático suportado por ontologias (que descrevem os operadores) são extremamente dependentes da intervenção de especialistas, que devem fazer as anotações referentes a novos métodos/algoritmos na devida ontologia, uma vez que nenhuma delas realiza isto automaticamente. Neste último caso, mesmo a solução PMD, que não utiliza ontologia, necessita de uma descrição dos operadores e especificações da tarefa que é passada diretamente ao planejador, pelo usuário. Já a solução proposta por e-LICO, embora possua uma meta-base de conhecimento que pode ser atualizada automaticamente, também depende da descrição dos operadores anotada em uma ontologia, para suporte ao planejamento.

No caso das soluções compostas, para o componente de Meta-Aprendizado, a cada novo método/ algoritmo adicionado, basta executá-lo em relação aos conjuntos de dados existentes e adicionar o resultado desta execução na meta-base de conhecimento.

Assim, as soluções compostas de Meta-Aprendizado com otimizadores também são automaticamente extensíveis.

4.7.2 Custo Benefício do Modelo Gerado

Soluções baseadas em RBC utilizam as características dos dados de entrada para buscar por casos similares na base e adaptar os processos usados anteriormente para o problema em questão. Já no Planejamento Automático, um repositório (geralmente uma ontologia) armazena os metadados de todos os operadores disponíveis como blocos de construção. O usuário fornece um novo conjunto de dados e especifica a descrição ou condições do problema. Essas entradas do usuário são expressas em um tipo de descrição do domínio de planejamento que é compreensível pelo planejador. Referenciando essa descrição do problema, o planejador constrói processos completos válidos selecionando operadores apropriados com base em seus metadados.

Em ambos os casos a recomendação é geralmente rápida, pois não é realizada a execução dos processos completos candidatos sobre os dados de entrada, sendo o desempenho do processo recomendado estimado com base em informações previamente disponíveis na base de conhecimento. Por outro lado, essas soluções dependem estritamente das informações provenientes da base de conhecimento: no caso de RBC, esse resultado está intrinsecamente relacionado à quantidade e à qualidade dos casos armazenados na base de casos, para as soluções de PA, a qualidade do modelo gerado está associada com a qualidade do repositório de metadados e do planejador em si.

No caso das soluções baseadas puramente em otimização (meta-heurísticas e otimização Bayesiana), o desempenho de cada processo candidato é avaliado a partir do treinamento desse modelo no conjunto de treinamento completo e avaliação (por validação cruzada) no conjunto de validação, o que garante uma boa aproximação do desempenho real. Em contraponto, esta abordagem despende um maior tempo/recurso na busca por um processo completo ideal, já que as possíveis soluções são constantemente executadas e avaliadas no conjunto de entrada completo.

Por fim, as soluções parcialmente apoiadas por Meta-Aprendizado utilizam esta estratégia para recomendar um conjunto inicial de processos completos (com base em similaridade dos exemplos da meta-base), que são adaptados para o problema em questão utilizando uma estratégia de planejamento ou otimização. Como o custo do Meta-Aprendizado é relativamente pequeno, ao aplicarem essa estratégia, tais soluções tendem a economizar recursos da estratégia de otimização. Neste caso, a qualidade do processo completo recomendado está associada à qualidade do modelo de recomendação gerado pelo processo de MA, pois, do contrário, se esta recomendação inicial não for precisa, as modificações re-

alizadas pelo planejador ou otimizador podem não ser suficientes para obter um processo completo ideal.

4.7.3 Considerações Gerais

Com base nas características das soluções existentes apresentadas no decorrer do capítulo, pode-se avaliar que soluções baseadas em otimização ou que combinem essa estratégia com Meta-Aprendizado, são as demonstram maior aderência aos principais desafios gerais da área, de extensibilidade, qualidade e avaliação do modelo.

Por outro lado, mesmo para essas soluções, um obstáculo fundamental para a seleção de processo completo é o longo tempo necessário avaliar um processo completo candidato sob o conjunto completo de dados de entrada, levando em consideração que para encontrar uma combinação cujo desempenho satisfaça o usuário final, as soluções existentes testam muitas combinações. Na tentativa de mitigar esse tempo, algumas soluções utilizam Meta-Aprendizado como forma de determinar um conjunto inicial de processos completos para serem avaliados. Tais soluções tem demonstrado bons resultados.

Dado que os conjuntos de dados tendem a crescer constantemente e que novos algoritmos serão desenvolvidos, soluções que explorem possibilidades de avaliações de processos completos menos custosas, ainda são uma possibilidade em aberto na área.

4.8 Considerações Finais do Capítulo

Este capítulo apresentou os trabalhos existentes para Seleção de Processo Completo, identificados com base em uma Revisão Sistemática da Literatura, ressaltando os pontos fortes e fracos de cada abordagem. A partir desta revisão foi possível identificar lacunas nas soluções existentes, que devem ser tratadas afim de se obter uma solução de SPC efetiva.

Com base nisso, o próximo capítulo apresenta a solução proposta neste trabalho para seleção de processo completo de DCBD, que busca explorar pontos fortes e mitigar as algumas das limitações identificadas na literatura até então, com o objetivo de gerar recomendações mais precisas e menos custosas computacionalmente.

5. UBERBAND: SELEÇÃO DE PROCESSO COMPLETO UTILIZANDO META-APRENDIZADO E OTIMIZAÇÃO BASEADA EM BANDIDOS MULTI-ARMADOS

Nos últimos anos, pesquisadores buscaram melhorar a experiência de usuários de Mineração de Dados (MD) e Aprendizado de Máquina (AM), a partir do desenvolvimento de diversas soluções para Seleção de Processos Completos (SPC). O objetivo destas soluções é encontrar rapidamente, dentro de um limite de recursos pré-estabelecido, uma combinação de operadores de Descoberta de Conhecimento em Bases de Dados (DCBD). que correspondem a métodos de pré-processamento e algoritmos de aprendizado e seus respectivos hiper-parâmetros, que maximizam uma medida de desempenho no problema de aprendizado de máquina e no conjunto de dados fornecidos.

Um desafio fundamental e ainda em aberto na área é o grande tempo de execução necessário para testar um processo completo candidato no conjunto de dados de treinamento, levando em consideração que muitas possibilidades de processo são testadas até que uma boa combinação seja encontrada. No geral, esta tarefa pode levar vários dias em um conjunto de dados com um número moderado de instâncias e atributos [Thornton et al., 2013], visto que a execução de um algoritmo de aprendizado geralmente cresce superlinearmente em função no número de instâncias e pelo menos linearmente em função do número de atributos [Hutter et al., 2019]. Dada a aplicação de SPC a problemas mais complexos e o desenvolvimento constante de novos operadores, surge a necessidade de se utilizar métodos que otimizem a avaliação dos processos candidatos, tornando o processo de SPC mais eficiente.

A partir da análise de trabalhos recentes aplicados a tarefa de otimização de hiper-parâmetros, verificou-se a possibilidade de explorar resultados intermediários de treinamento a partir da utilização de métodos de fidelidade múltipla (Seção 3.3.3). Nos métodos de fidelidade múltipla são utilizadas heurísticas, como amostragem do conjunto de treinamento, amostragem dos atributos, para gerar aproximações de baixa fidelidade da função a ser otimizada, de forma a diminuir o tempo de execução das configurações avaliadas [Kandasamy et al., 2017].

Nesse contexto, o trabalho de [Li et al., 2017] se destaca por apresentar uma solução baseada em bandidos multi-armados para identificação da melhor configuração de hiper-parâmetros a partir da alocação adaptativa de recursos para configurações promissoras, enquanto as menos promissoras são eliminadas. Em seus resultados comparativos, esse algoritmo, denominado Hyperband, superou soluções baseadas em Otimização Bayesiana (OB) com resultados que apresentam desempenho equivalente e tempo de execução muito inferior as soluções de OB, em tarefas como regularização de *kernel* de Support Vector Machines (SVM) e para otimização de uma Rede Neural Convolucional.

Por outro lado, ao aplicar Hyperband em um tarefa de SPC, usando o mesmo espaço de busca que o Auto-Sklearn [Feurer et al., 2015a], [Li et al., 2017] reportaram que, embora Hyperband tenha convergido mais rapidamente, ele obteve erros de validação e teste superiores as abordagens de OB e à busca aleatória. A Figura 5.1 (a, b) apresenta os resultados reportados, considerando uma janela de tempo máximo de 1 hora (3600 segundos)

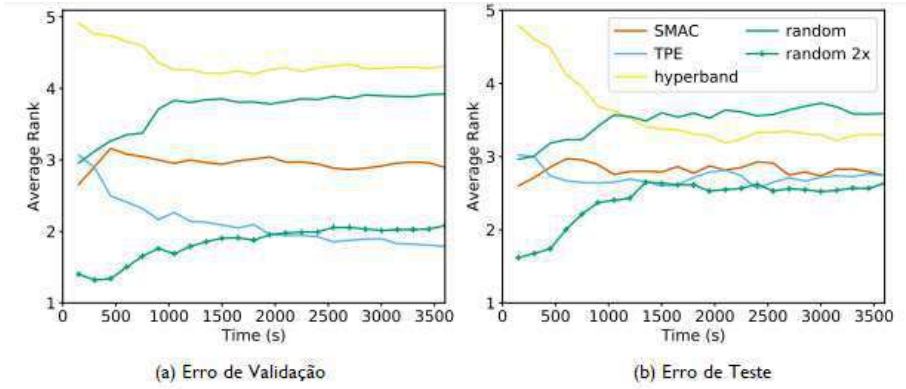

Figura 5.1 – Ranking médio da avaliação de Hyperband contra Estratégias de Otimização Bayesianas (SMAC e TPE) e amostragem aleatória, para o problema de SPC em 117 conjuntos de dados [Fonte [Li et al., 2017]]

Os resultados obtidos em 117 conjuntos de dados mostram que Hyperband supera a busca aleatória no teste, apesar de ter um desempenho pior na validação. Os métodos de otimização Bayesiana (SMAC e TPE) superam Hyperband e a busca aleatória no desempenho de teste, mas também exibem sinais de *overfitting* para o conjunto de validação, uma vez que superam a Hyperband por uma margem maior na validação.

A justificativa para estes resultados foi a de que os métodos de OB escalam melhor em espaços de busca de alta-dimensionalidade, como é o caso da SPC. Uma vez que Hyperband foi desenvolvido para otimizar a configurações de hiper-parâmetros de um único algoritmo por vez, o espaço de busca é restrito a este algoritmo e uma única escolha deve ser otimizada: a combinação desses hiper-parâmetros. Porém, configuração de algoritmos é apenas uma das sub-tarefas da seleção de processo completo, que primeiramente inclui a seleção desses algoritmos.

Tendo em vista o potencial oferecido pela alocação adaptativa de recursos no contexto de SPC e as limitações do algoritmo Hyperband para esta tarefa, esta tese propõe Uberband, que trata dos problemas de otimização da seleção dos operadores e da otimização dos hiper-parâmetros destes operadores, de maneira a oferecer uma cobertura completa ao processo de SPC.

Uberband faz uso de uma estratégia de Meta-Aprendizado para predição de desempenho dos operadores, onde estes valores são usados como peso para amostragem de operadores realizada pelo otimizador, de maneira a favorecer a seleção daqueles com maior desempenho previsto. Uma vez amostrados os operadores mais promissores, Uber-

band utiliza a alocação adaptativa de recursos para experimentar diversas configurações de hiper-parâmetros, até selecionar o processo completo com melhor desempenho para a tarefa em questão.

O restante do Capítulo está dividido da seguinte maneira, a Seção 5.1 apresenta a estrutura geral da solução proposta. A Seção 5.2 apresenta o componente de Meta-Aprendizado. A Seção 5.3 apresenta o componente de otimização. Por fim, a Seção 5.4 apresenta o Uberband com base na Arquitetura Genérica de uma Solução para Apoio à Seleção de Processo Completo, apresentada no Capítulo 3.

5.1 Uberband

A Figura 5.2 apresenta uma visão geral do Uberband. A partir desta ilustração, podemos identificar os dois componentes da solução:

1. Componente de Meta-Aprendizado (MA) - Fase 1, que prevê o desempenho dos operadores no espaço de busca, com base no conhecimento adquirido com problemas de aprendizado anteriores;

2. Componente de otimização para Seleção de Processo Completo (SPC) - Fase 2, baseado em bandidos multi-armados e que utiliza o desempenho predito pela estratégia de MA como probabilidade para amostragem de operadores, bem como atua na otimização das configurações de hiper-parâmetros desses operadores.

As próximas seções apresentam cada um desses componentes.

5.2 Fase 1: Abordagem de Meta-Aprendizado para Predição de Desempenho de Operadores

Na Seleção de Processo Completo proposta, inicialmente é aplicada uma estratégia de Meta-Aprendizado (MA) ("Fase 1", Figura 5.2), para determinar o desempenho previsto de cada subconjunto de operadores, que correspondem a todas as combinações válidas de métodos de pré-processamento e algoritmos de aprendizado disponíveis no espaço de busca, mas sem incluir as combinações de hiper-parâmetros.

Meta-Aprendizado é uma estratégia utilizada por soluções de SPC para pré-selecionar processos completos para inicializar algoritmos de otimização, tal como em [Feurer et al., 2015a] e [Chong et al., 2013], e algoritmos de planejamento automático [Nguyen et al., 2014], visando tornar a recomendação mais eficaz e menos custosa.

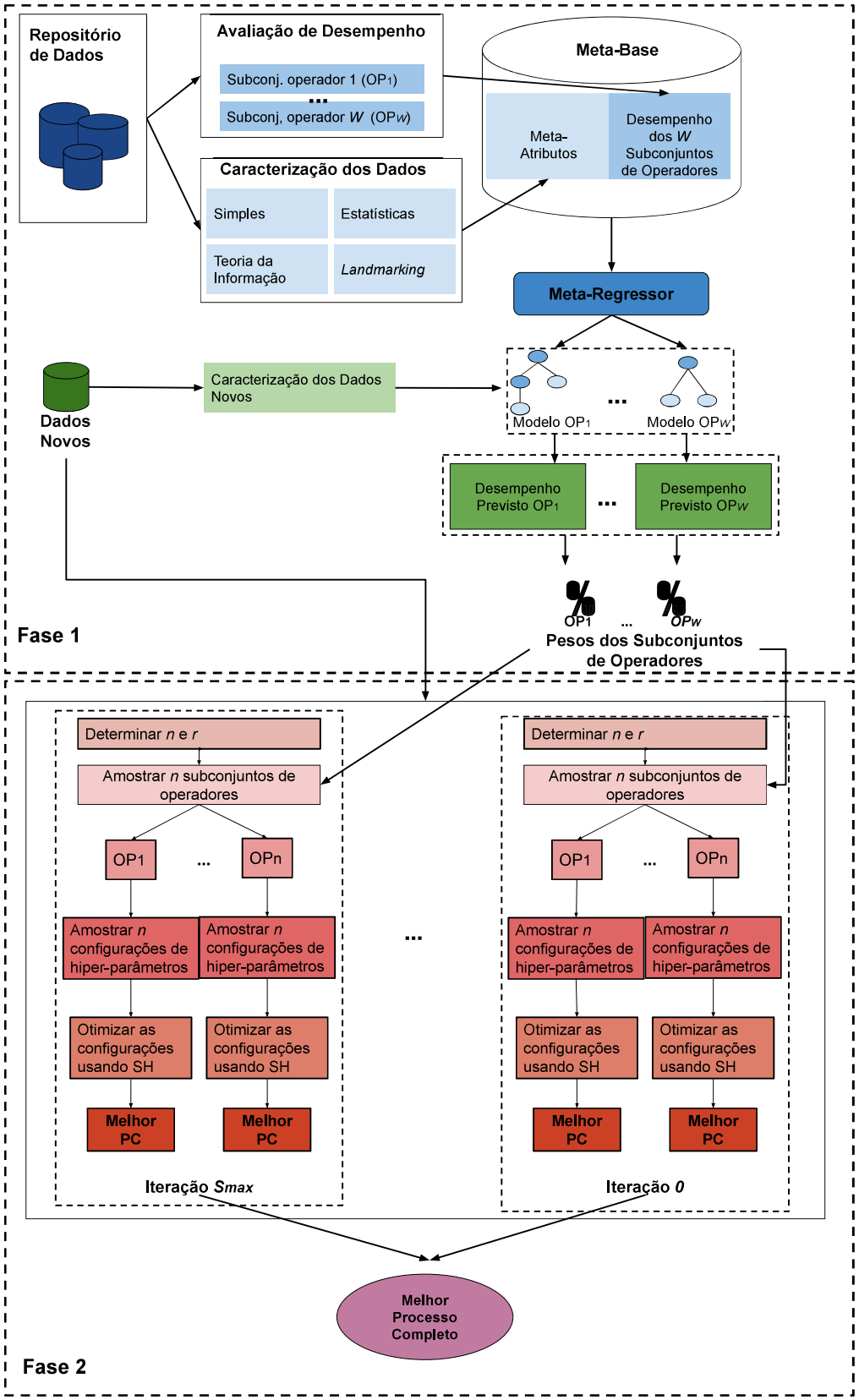

Figura 5.2 – Visão Geral do Uberband

Diferente dessas soluções, a abordagem de Meta-Aprendizado proposta em Uberband não tem como foco determinar um conjunto de processos completos para inicializar a fase de otimização, por dois motivos:

1. Embora um sistema de Meta-Aprendizado seja apto a fornecer uma recomendação precisa do processo completo de DCBD mais adequado a um determinado problema, este resultado está intrinsecamente relacionado com a quantidade e qualidade dos conjuntos de dados utilizados para geração da meta-base de conhecimento. Logo, se não houver na base exemplos que possam ser utilizados para guiar a recomendação para um dado problema, ou os exemplos existentes são sejam suficientes para realizar uma recomendação precisa, a qualidade do processo recomendado ficará comprometida [Sun et al., 2013];

2. Dada a influência dos hiper-parâmetros sobre o desempenho final dos métodos e algoritmos, para determinar efetivamente os processos completos a serem otimizados e, com isso, restringir o espaço de busca, a estratégia de Meta-Aprendizado deveria considerar as combinações desses hiper-parâmetros (além das combinações dos operadores). Do contrário, o espaço de busca poderia ser podado indevidamente. Porém, computar e armazenar todas as combinações de hiper-parâmetros acarretaria em maior complexidade na construção da meta-base.

Levando em consideração esses pontos, ao propor a utilização do desempenho previsto para gerar probabilidades de amostragem de subconjunto de operadores, e não um *ranking* ou o melhor processo completo, o Meta-Aprendizado auxilia o otimizador na busca por um processo completo ótimo, não realizando efetivamente a recomendação, o que faz com que a solução: i) favoreça subconjuntos de operadores com maior desempenho previsto, sem desconsiderar as demais, minimizando, com isso, as possibilidades de erro do modelo gerado pelo meta-aprendiz; ii) possa trabalhar no nível-meta apenas com as combinações de operadores, sem levar em consideração os hiper-parâmetros, tendo um menor custo computacional no cálculo da meta-base em relação às abordagens atuais.

De acordo com o conhecimento da autora da presente tese, esta é a primeira aplicação de Meta-Aprendizado que busca identificar uma distribuição de amostragem no espaço de busca de um otimizador.

5.2.1 Definição da Estratégia de Meta-Aprendizado

O processo de Meta-Aprendizado apresentado nesta tese é baseado nas definições de Brazdil et al. [Brazdil et al., 2008], e adaptado para o problema de seleção de processo completo. Ele inicia com um repositório de conjunto de dados $D = D^1, ..., D^P$ referente a P problemas de aprendizado de máquina.

Cada conjunto de dados D^i é descrito por um conjunto de M meta-atributos $\hat{m}^i = (m^i_1, ..., m^i_M)$. Simultaneamente, o desempenho preditivo dos W subconjuntos de operadores disponíveis no espaço de busca (definidos a partir de combinações válidas de métodos de

pré-processamento e algoritmos de aprendizado) é pré-computado, $\hat{w}^i = (w_1^i, ..., w_W^i)$, sobre D^i.

Os meta-atributos e o desempenho dos operadores são armazenados em uma base de conhecimento (meta-base) como tuplas $(\hat{m}, \hat{w})$, onde $\hat{m} \in \mathbb{R}^M$ e $\hat{w} \in \mathbb{R}^W$.

Então, para cada subconjunto de operadores $j = 1, ..., |W|$, um algoritmo de regressão é aplicado para construir um modelo φ_j capaz de predizer o desempenho do subconjunto $w_j' \in \mathbb{R}$ para um vetor de meta-atributos $\hat{m}'$ usando a função de predição $f(\varphi_j, \hat{m}')$.

Dado um novo conjunto de dados D^{P+1}, são computados os meta-atributos $\hat{m}^{P+1}$ e cada modelo gerado pelo meta-regressor é aplicado para estimar o desempenho do subconjunto de operadores correspondente:

$$w_j^{P+1} = f(\varphi_j, \hat{m}^{P+1}) \tag{5.1}$$

O desempenho estimado é utilizado para gerar a recomendação de pesos para os subconjunto de operadores, de maneira a proporcionar uma amostragem probabilística durante a fase de otimização. Para geração destes pesos, é utilizado o método de seleção de roleta viciada [Goldberg e Holland, 1988], que assume que a probabilidade de seleção é proporcional ao desempenho predito do subconjunto de operadores . Assim, para cada $j = 1, ..., |W|$ em $\hat{w}^{P+1}$ a probabilidade de seleção é determinada como:

$$v_j^{P+1} = \frac{w_j^{P+1}}{\sum_{j=1}^{W} w_j^{P+1}} \tag{5.2}$$

As probabilidades são então agrupadas em um vetor $\hat{v}^{P+1} = (v_1^{P+1}, ..., v_W^{P+1})$ que será utilizado para identificar a distribuição da qual amostrar subconjunto de operadores no espaço de pesquisa do otimizador.

O Capítulo 6 detalha os componentes da estratégia de Meta-Aprendizado, bem como a validação experimental desta estratégia.

5.3 Fase 2: Sistema de Seleção de Processo Completo

Nesta fase ("Fase 2", Figura 5.2) é realizada efetivamente a recomendação do processo completo.

Além de utilizar os pesos derivados da estratégia de Meta-Aprendizado, um diferencial desta solução em relação as já existentes é a condução de treinamento/validação de baixo custo em pequenas amostras do conjunto de dados para eliminar combinações pouco promissoras de processos completos o mais cedo possível, e dedicar mais recursos computacionais a ajustes finos de combinações promissoras. Mais especificamente, é utili-

zada uma amostra relativamente pequena de instâncias de treinamento para validar várias combinações e encontrar as mais promissoras. O processo é repetido por várias rodadas, aumentando a amostra e diminuindo o espaço de pesquisa. Na última rodada, todo conjunto de dados de treinamento é utilizado para encontrar bons processos completos apenas nos candidatos com melhor desempenho até então.

Por fim, a recomendação final é dada na forma do melhor processo completo encontrado, classificado de acordo com o desempenho.

5.3.1 Otimização de Processos Completos Baseada em Bandidos Multi-Armados

Uberband ataca o problema de otimização de processo completo em duas fases: otimização dos subconjuntos de operadores e otimização dos hiper-parâmetros dos subconjuntos de operadores. O pseudo-código do Uberband é apresentado em Algoritmo 5.1.

Entrada : novo conjunto de dados D^{P+1}, modelos preditos $\hat{\varphi} = (\varphi_1, ..., \varphi_W)$, R, η
$\hat{m}^{P+1} = (m_1^{P+1}, ..., m_{|M|}^{P+1})$
for j $\in \{0, ..., |W|\}$ **do**
    $w_j^{P+1} = f(\varphi_j, \hat{m}^{P+1})$
end for
$\hat{v}^{P+1} \leftarrow selecao\_por\_roleta(\hat{w}^{P+1})$
$n_{max} = \max\{9, R/1000\}$
$s_{max} = \lfloor \log_\eta(n_{max}) \rfloor$
$B = (s_{max} + 1)R$
for $s \in \{s_{max}, ..., 0\}$ **do**
    $n = \left\lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \right\rceil$, $r = R\eta^{-s}$
    $O = obter\_operadores(n, \hat{v}^{P+1})$
    for $j \in \{1, ..., |O|\}$ **do**
        $H = obter\_hiper\_parametros(n, j)$
        for $i \in \{0, ..., s\}$ **do**
            $n_i = \lceil n\eta^{-i} \rceil$
            $r_i = r\eta^i$
            $L = \{avaliacao\_desempenho(h, r_i) : h \in H\}$
            $H = top\_k(H, L, \lfloor \frac{n_i}{\eta} \rfloor)$
        end for
    end for
end for
return melhor processo completo identificado

Algoritmo 5.1 – Algoritmo Uberband

Uberband utiliza os parâmetros definidos no Hyperband para determinar quantas configurações de processo completo irá avaliar e qual o orçamento de recursos a ser uti-

lizado. Porém, a seleção aleatória de processos completos no início de cada iteração é substituída por duas etapas:

1. Seleção de subconjuntos de operadores, baseada nos pesos derivados na fase de MA;

2. Seleção aleatória configurações de hiper-parâmetros, para cada subconjunto de operadores selecionado anteriormente.

Então, o procedimento Successive Halving padrão é realizado para otimizar as configurações de hiper-parâmetros.

5.3.2 Recursos

Em termos de recursos o trabalho de [Li et al., 2017] sugere diversos elementos, tais como tempo, atributos, número de iterações, a depender do tipo de algoritmo cujos hiper-parâmetros estão sendo otimizados.

Neste trabalho, cujo foco é a otimização de processos completos, o que envolve a análise de diversos algoritmos e métodos de processamento simultaneamente, cada qual com seu respectivo viés indutivo, utilizamos como recurso as *instâncias* (exemplos) do conjunto de dados de entrada. Este recurso foi selecionado por ser um elemento em comum aos diversos tipos de algoritmos e métodos.

Assim, o número máximo de recursos, R, a ser utilizado por uma configuração é determinado em função do tamanho do conjunto de treinamento disponível.

5.3.3 Número Máximo de Configurações

No algoritmo Hyperband, apresentado em 3.1, considera-se que R é também o número máximo de configurações avaliadas na execução de Successive Halving que realiza maior exploração, isto é $s = s_{max}$. Neste caso, quando $i = 0$, $n = R$ e $r = 1$.

Porém, uma vez que o recurso a ser utilizado é o número de instâncias do conjunto de dados, seria impraticável que uma determinada configuração de processo completo fosse executada com apenas uma única instância. Sendo assim, neste trabalho foi realizada uma adaptação do algoritmo, sugerida por [Li et al., 2017], a partir da adoção de um atributo $n_{max} = \max\{9, R/1000\}$, de maneira que $n \leq n_{max}$. Com isso, o número máximo de configurações a serem avaliadas passa a ser restringido em função deste novo parâmetro e não mais em função de R: $s_{max} = \lfloor \log_{\eta}(n_{max}) \rfloor$.

5.3.4 Seleção de Processo Completo

As entradas do Uberband consistem no conjunto de dados D^{P+1}, representativo de um novo problema de MD a ser resolvido, os modelos de predição gerados pelo processo de Meta-Aprendizado $\varphi = \{\varphi_1, ..., \varphi_W\}$ para cada subconjunto de operadores disponível no espaço de busca, $j = \{1, ..., |W|\}$ e os hiper-parâmetros R, que define o número máximo de recursos a ser utilizado por uma única configuração, η, que controla a proporção de configurações que será descartada a cada execução do SuccessiveHalving.

Os modelos de predição são utilizados para estimar o desempenho de cada subconjunto de operadores para o novo conjunto de dados $\hat{w}^{P+1}$. Estas estimativas de desempenho são utilizadas para gerar os pesos $\hat{v}^{P+1} \leftarrow$ *selecao_por_roleta*$(\hat{w}^{P+1})$.

A partir de então, a fase de otimização efetivamente inicia, com a determinação do número de execuções do SuccessiveHalving que serão realizadas s_{max}, que é feita em função do número máximo de configurações n_{max}.

Uberband inicia com a maior execução do SuccessiveHalving, privilegiando o processo de exploração e avaliando o maior número de configurações (n).

Inicialmente são amostrados os subconjuntos de operadores que serão otimizados, usando os pesos determinados pelo processo de Meta-Aprendizado, n subconjuntos de operadores são amostrados proporcionalmente ao desempenho predito (linha 12). Então, para cada subconjunto de operadores, são amostrados n configurações de hiper-parâmetros (linha 14), que serão avaliadas de acordo com o procedimento do SuccessiveHalving (linhas 15-20). No geral, a cada iteração de s são avaliados n^2 processos completos candidatos.

Os recursos disponíveis r são adaptativamente alocados a medida que as melhores configurações avaliadas são determinadas. A cada iteração de s, a medida em que o número de configurações é podado pelo fator η, mais recursos são alocados.

A fase de otimização do Uberband utiliza quatro funções:

- *obter_operadores*$(n, \hat{v}^{P+1})$ (linha 12): função que retorna n subconjuntos de operadores, sendo que a probabilidade de amostragem de cada subconjunto é proporcional ao desempenho predito pelo Meta-Aprendizado;

- *obter_hiper_parametros*(n, O) (linha 14): função que retorna um conjunto de n configurações (hiper-parâmetros) para cada processo completo em O, amostradas no espaço de hiper-parâmetros de cada operador (método de pré-processamento ou algoritmo de aprendizado) que compõe o processo completo;

- *avaliacao_de_desempenho*(H, r) (linha 18): função que avalia cada configuração em H utilizando (r) instâncias do conjunto de treinamento como recurso e retorna o desempenho no conjunto de validação;

- *top_k*($H, L, \left\lfloor \frac{n_i}{\eta} \right\rfloor$) (linha 19): função que recebe um conjunto de configurações (H), com seus respectivos desempenhos associados (L), e retorna as k melhores configurações.

O Capítulo 7 apresenta a avaliação experimental do Uberband, incluindo a avaliação dos hiper-parâmetros.

5.3.5 Espaço de Busca

Atualmente, Uberband trabalha com um espaço de busca de operadores e hiper-parâmetros pré-definido, onde a ordem e as combinações válidas dos operadores de diferentes fases do processo de DCBD deve ser estabelecida previamente. A ordem dos operadores pode ser baseada no próprio processo de DCBD (Seção 2.1)

Na fase de Meta-Aprendizado, os operadores são combinados, de acordo com a ordem pré-estabelecida, e são executados utilizando hiper-parâmetros *default*. Esta é uma limitação imposta pela impossibilidade de executar experimentos com todas as configurações possíveis de cada operador, uma vez que diversos hiper-parâmetros possuem valores reais.

Já na fase de otimização, subconjuntos de operadores são amostrados de acordo com as mesmas combinações possíveis da fase de Meta-Aprendizado e os hiper-parâmetros são amostrados respeitando as possíveis condicionalidades determinadas.

Apesar de utilizar uma ordem pré-estabelecida de operadores e condicionalidade de hiper-parâmetros, Uberband permite que seu espaço de busca seja alterado com o mínimo de intervenção humana: caso uma nova fase do processo de DCBD seja inserida ou removida, a nova ordem de combinação de operadores deve ser estabelecida. Já no caso de inserção de operadores em uma fase já existente, os operadores podem ser apenas inseridos no espaço de busca com seus respectivos hiper-parâmetros. Para ambos os casos, a meta-base de conhecimento é atualizada com a adição das medidas de desempenho nos novos subconjuntos de operadores inseridos ou pela remoção de medidas de desempenho dos subconjuntos não mais disponíveis. Os modelos de predição são atualizados de acordo com as mesmas alterações, bem como o espaço de amostragem do otimizador.

5.4 Caracterização do Uberband com Base na Arquitetura Genérica de uma Solução para Apoio à Seleção de Processo Completo

A Figura 5.3 apresenta a caracterização do Uberband de acordo com a Arquitetura Genérica de uma Solução para Apoio à Seleção de Processo Completo, definida no Capítulo 3.

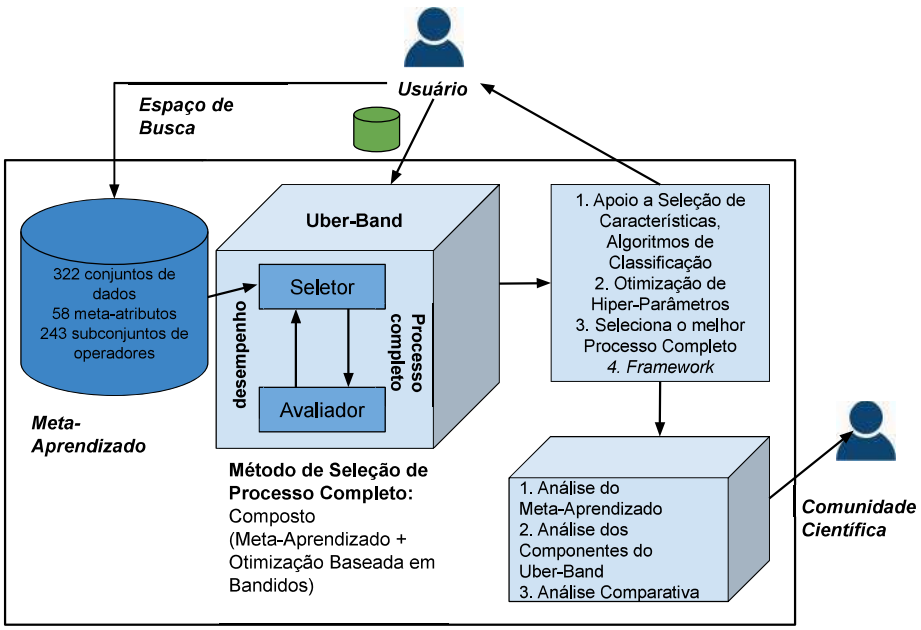

Figura 5.3 – Caracterização do Uberband com Base na Arquitetura Genérica de uma Solução para Apoio à Seleção de Processo Completo

A "Base de Conhecimento" é estruturada como uma meta-base de Meta- Aprendizado (Seção 3.2.1) e contém um repositório de problemas de AM dos quais foram extraídos meta-atributos que caracterizam estes problemas, além do desempenho previsto de subconjuntos de operadores de DCBD para cada problema.

Como "Método de Seleção de Processo Completo", foi combinado Meta- Aprendizado com Otimização Baseada em Bandidos Multi- Armados, de forma a combinar o conhecimento oferecido pela base de conhecimento na seleção de melhores opções a serem avaliadas pelo otimizador.

A versão atual do Uberband oferece apoio à Seleção Processos Completos que incluam algoritmos de Seleção de Características e a Seleção de Algoritmos para problemas de Classificação, bem como a Otimização de Hiper-Parâmetros.

Por fim, Uberband foi avaliado sob diversos aspectos, incluindo a avaliação individual do processo de Meta-Aprendizado, Avaliação de cada componente e Avaliação comparativa com soluções estado-da-arte em SPC.

5.5 Considerações Finais do Capítulo

Tendo como base as limitações nas soluções de Seleção de Processo Completo existentes, principalmente no que diz respeito à avaliação dos processos completos candidatos, neste capítulo foi apresentada a visão geral de Uberband, uma solução para SPC que utiliza Meta-Aprendizado e otimização baseada em bandidos multi-armados para recomendar um processo completo para um determinado problema de aprendizado.

Nos Capítulos 6 e 7 serão apresentados detalhadamente os componentes de MA e otimização via bandidos multi-armados, bem como a análise experimental realizada sobre cada componente.

6. META-APRENDIZADO PARA PREDIÇÃO DE DESEMPENHO DE SUBCONJUNTOS DE OPERADORES

Abordagens de Meta-Aprendizado (MA) podem ser empregadas para elaboração de sistemas de seleção de algoritmos, seleção de hiper-parâmetros, bem como para a seleção de processo completo (SPC). Neste último caso, MA é utilizado para definir um conjunto inicial de processos completos a serem avaliados por um algoritmo de otimização [Feurer et al., 2015a, Chong et al., 2013].

Diferente das soluções existentes, esta pesquisa propõe a utilização de Meta-Aprendizado como forma de guiar o processo de amostragem de processos completos feito por um algoritmo de otimização (Capítulo 5), a partir da estimativa de desempenho desses processos.

Neste capítulo é apresentado o processo experimental conduzido para avaliar a estratégia proposta de Meta-Aprendizado isoladamente, Figura 6.1. A avaliação em relação aos ganhos oferecidos para a solução completa proposta neste trabalho será apresentada no próximo capítulo.

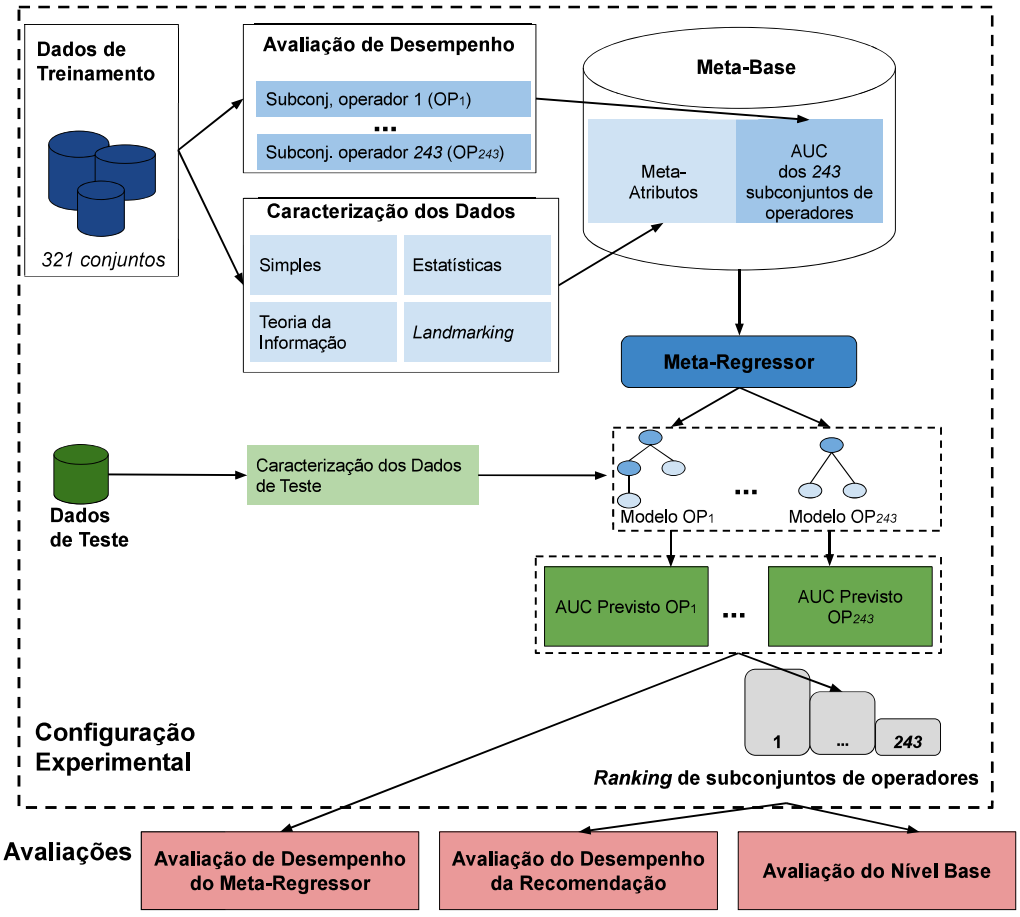

Figura 6.1 – Avaliação da Estratégia de Meta-Aprendizado

A partir da configuração experimental definida, são realizadas 3 tipos de avaliações:

1. Avaliação de desempenho do meta-regressor: avaliação do desempenho predito para cada sub-conjunto de operadores;

2. Avaliação do desempenho da recomendação: avaliar a acurácia do *ranking* predito em relação ao *ranking* ideal;

3. Avaliação do nível-base: avaliar o ganho real da recomendação em termos de resultados obtidos no problema de nível-base.

O restante do Capítulo está dividido da seguinte maneira: a Seção de Configuração Experimental (6.1) apresenta as bases de dados utilizadas, os operadores, medida de desempenho, os meta-atributos e o meta-regressor, além da recomendação por *ranking* utilizada para avaliar a estratégia de MA. Na Seção 6.2 é apresentada a metodologia de avaliação. Na Seção 6.3 é descrita a avaliação de desempenho do meta-regressor, na Seção 6.4 a avaliação de desempenho da recomendação e, por fim, na Seção 6.5 é apresentada a avaliação de nível-base.

6.1 Configuração Experimental

6.1.1 Bases de Dados

Neste trabalho, foram avaliados conjuntos de dados referentes a tarefa de classificação (Seção 2.2.1), por ser a tarefa com maior amostragem de dados públicos disponíveis para teste, maior número de algoritmos de Aprendizado de Máquina (AM) propostos e, especificamente do ponto de vista de Meta-Aprendizado, ter o maior número de meta-atributos propostos.

Ao todo foram utilizados 322 conjuntos de dados para construir a meta-base de conhecimento. Essas bases representam problemas de classificação binária de diversas naturezas. O número de instâncias varia entre 100 e 98.528 e o número de atributos entre 3 e 10.937.

Estes dados são provenientes do OpenML [Vanschoren et al., 2014], uma plataforma colaborativa para compartilhamento de conjuntos de dados e experimentos de AM. A lista completa das bases, com as principais características está disponível no Anexo A.

6.1.2 Operadores

Para projetar os experimentos da estratégia de Meta-Aprendizado e, posteriormente, da estratégia de otimização foram selecionados operadores da biblioteca de AM Weka [Witten e Frank, 2016], uma vez que esta é uma das mais proeminentes bibliotecas da área e contém a maior variedade de métodos de pré-processamento e algoritmos de aprendizado disponíveis.

Os algoritmos de aprendizado selecionados são heterogêneos para representar os diferentes métodos preditivos de AM, tais como baseado em distâncias (k-NN), probabilísticos (NaiveBayes, BayesNet), baseados em procura (HoeffdingTree, J48, REPTree, RandomTree, DecisionStump, LMT, JRip, OneR, ZeroR, Decision Table, PART), baseados em otimização (MultilayerPerceptron, SGD, SVM, Logistic, Simple Logistic, Voted Perceptron) e modelos múltiplos preditivos (AdaBoost M1, RandomForest, RandomSubset, RandomCommittee, Vote, Stacking, LogitBoost).

Como tarefa de pré-processamento, foi selecionada a Seleção de Características (SC) (*feature selection*) uma vez que lidar com dados de alta-dimensionalidade apresenta um dos maiores desafios do processo de DCBD [Parmezan et al., 2017]. Neste contexto, os métodos de seleção de características objetivam encontrar um subconjunto de atributos a partir dos dados de entrada, que maximize um determinado objetivo e, portanto, pode ser visto como um problema de pesquisa com um determinado método de pesquisa, métrica de avaliação e objetivo geral. Os métodos de SC são categorizados em diferentes abordagens, de acordo com a dependência em relação ao algoritmo de aprendizado: i) filtros, onde a seleção de características importantes é realizada em um passo de pré-processamento usando características gerais do conjunto de dados, em um processo que é independente do algoritmo de aprendizado; ii) *wrappers*, usa o modelo gerado por um algoritmo de aprendizado para avaliar cada subconjunto de características candidatas, baseado no seu desempenho preditivo; e iii) *embedded*, a tarefa de SC é realizada internamente no algoritmo de aprendizado, como parte de seu processo de construção do modelo. Além disso, os métodos de SC empregam diferentes medidas de importância para avaliar os subconjuntos de características, tais como correlação, ganho de informação e distância.

Nesta pesquisa, para obter resultados mais generalizáveis, foram avaliados métodos de SC baseados nas diferentes abordagens existentes e com diferentes medidas de importância.

Ao todo, foram utilizados 27 algoritmos de aprendizado e 8 métodos de pré- processamento, totalizando 243 subconjuntos de operadores: 8 métodos $\times$ 27 algoritmos de aprendizado +27 algoritmos de aprendizado.

A Tabela 6.1.2 apresenta a lista de algoritmos de aprendizado e a Tabela 6.1.2 dos métodos de pré-processamento.

Tabela 6.1 – Algoritmos de Aprendizado Usados para Avaliar o Uberband

Algoritmo	Tipo de Método	Parâmetro
AdaBoost M1	múltiplo	100 iterações
BayesNet	probabilístico	
DecisionStump	baseado em procura	
DecisionTable	baseado em procura	
HoeffdingTree	baseado em procura	
J48	baseado em procura	
JRip	baseado em procura	
IBk	baseado em distância	k = 1
LMT	baseado em procura	
LogitBoost	múltiplo	
Logistic	baseado em otimização	ridge = 0,00000001
MultilayerPerceptron	baseado em otimização	1 camada oculta
NaiveBayes	probabilístico	
OneR	baseado em procura	
PART	baseado em procura	
RandomForest	múltiplo	100 árvores
RandomTree	baseado em procura	
RandomSubset	múltiplo	
RandomCommitte	múltiplo	
REPTree	baseado em procura	
Stacking	múltiplo	
SimpleLogistic	baseado em otimização	
Stochastic Gradient Descent (SGD)	baseado em otimização	
Sequential Minimal Optimization (SMO)	baseado em otimização	
Vote	múltiplo	
VotedPerceptron	baseado em otimização	
ZeroR	baseado em procura	

Tabela 6.2 – Métodos de Pré-Processamento Usados para Avaliar o Uberband

Método	Tipo de Abordagem	Medida de Importância
CFSSubsetEval	Filtro	Correlação
Classifier Attribute Eval	Wrapper	
Gain Ratio Eval	Filtro	Ganho de Informação
Info Gain Eval	Filtro	Ganho de Informação
OneR Attribute Eval	Wrapper	
RELIEF Eval	Filtro	Distância
Symmetrical Uncert	Wrapper	
Wrapper Sub Eval	Wrapper	

6.1.3 Desempenho dos Operadores

Todos os 243 subconjuntos de operadores foram avaliados sobre os 322 conjuntos de dados usando Validação Cruzada de 10-*folds* (Seção 2.2.1).

Como medida de desempenho foi utilizada a área sob a curva ROC (AUC) (Seção 2.2.1). Esta medida foi selecionada como critério de avaliação em detrimento das comumente utilizadas em trabalhos de Meta-Aprendizado, como acurácia e precisão, por uma variedade de razões. Em primeiro lugar, a distribuição das classes é muito desbalanceada para a maior parte dos conjuntos de dados: neste caso, uma votação por maioria sim-

ples pode alcançar uma precisão muito alta, enquanto que na prática este pode não ser um modelo útil. Segundo, a classificação de falsos positivos e falsos negativos pode ter um custo diferente, mas esses custos não são conhecidos, por isso faz sentido avaliar o desempenho em toda a faixa de pontuação do modelo. Outras medidas de desempenho igualmente interessantes neste caso, para substituição de AUC seriam área sob a curva de precisão-revocação e acurácia balanceada.

6.1.4 Meta-Atributos

No Capítulo 3 foram apresentadas as principais abordagens para caracterização das bases de dados utilizadas em Meta-Aprendizado. Elas são empregadas para a geração de meta-atributos constituintes dos meta-exemplos. Neste trabalho, foram usadas medidas do tipo simples, estatísticas, baseadas em teoria da informação e *landmarkings*. A escolha dessas medidas deu-se pela ampla aplicação e, consequentemente, avaliação em trabalhos de Meta-Aprendizado.

Uma vez que a abordagem de MA é uma etapa dentro da solução final proposta, tinha-se interesse na utilização de medidas cuja complexidade computacional fosse relativamente baixa, de maneira a não sobrecarregar o tempo total para seleção de processo completo. Medidas específicas para caracterização de processo completo, tais como as propostas por Nguyen *et al.* [Nguyen et al., 2014], que envolvem Mineração de Padrões em árvores de decisão, e com foco em uma determinada etapa de pré-processamento, como as medidas de *landmarking* para seleção de características propostas por [Post et al., 2016] apresentam uma complexidade alta por envolverem execuções de *landmarkings* mais complexos.

A Tabela 6.3 apresenta as medidas de caracterização.

As medidas do tipo simples, estatísticas e baseadas em teoria da informação são adequadas tanto aos dados categóricos (no caso das medidas baseadas em teoria da computação) quanto aos numéricos (no caso das medidas estatísticas) utilizados aqui. As medidas foram calculadas segundo as definições apresentadas em [Kalousis, 2002].

6.1.5 Meta-Regressor

Meta-regressão têm sido aplicada tanto no contexto de seleção quanto de configuração de algoritmos. Neste trabalho, utiliza-se o algoritmo Random Forest [Breiman, 2001] devido as seguintes motivações:

Tabela 6.3 – Meta-Atributos Uberband

Categoria	Medida	Descrição
Simples	Número de Instâncias, Logaritmo do Número de Instâncias, Número de Atributos, Logaritmo do Número de Atributos, Dimensionalidade, Logaritmo da Dimensionalidade, Dimensionalidade Inversa, Logaritmo da Dimensionalidade Inversa, Número de Classes, Número de Atributos Categóricos, % de Atributos Categóricos, Número de Atributos Numéricos, % de Atributos Numéricos, Relação Numérico para Nominal, Relação Nominal para Numérico, Probabilidade de Classe Mínima, Máxima, Média e Desvio Padrão, Número de Atributos Binários,Número de Símbolos Mínimo, Máximo, Médio e Desvio Padrão, Número de Valores Ausentes, % de Valores Ausentes, Número de Instâncias com Valores Ausentes, % de Instâncias com Valores Ausentes, Número de Atributos com Valores Ausentes, % de Atributos com Valores AusentesTamanho da Classe Majoritária, % da Classe Majoritária	Descrevem propriedades obtidas a partir da representação de atributo-valor
Estatísticas	Curtose Mínima, Curtose Máxima, Curtose Média, Desvio Padrão da Curtose	Expressa o grau de achatamento da distribuição, que em muitos casos é considerado em relação à distribuição normal.
	Assimetria Mínima, Máxima, Média e Desvio Padrão da Assimetria	Expressa como e quanto a distribuição dos dados foge da condição de simetria.
	Média das Médias de Atributos Numéricos Desvio Padrão dos Desvios Padrões dos Atributos Numéricos	
Teoria da Informação	Entropia da Classe	Indica a quantidade aproximada de informação necessária para identificar o rótulo da classe de um exemplo a partir do conjunto de dados original
	Entropia Média dos Atributos	Estima a qualidade de informação que um atributo em particular tem a oferecer para a predição da classe
	Informação Mútua Média	Mede a redução da entropia causada pela partição dos exemplos de acordo com os valores de um dado atributo.
	Número de Atributos Equivalentes	Taxa entre a Entropia da Classe e a Informação Mútua Média
	Taxa Sinal/Ruído	Estima a quantidade de informação não útil do conjunto de dados
Landmarking	Acurácia Árvore de Decisão (gini) Acurácia Árvore de Decisão (entropia)	
	Acurácia 1-NN	

1. De maneira geral, comitês de algoritmos (*ensembles*) usualmente superam algoritmos base nas tarefas de AM [Rokach, 2010, Sagi e Rokach, 2018];

2. Considerando-se que o meta-regressor deve ser capaz de lidar com vários problemas de predição de desempenho, ou seja, múltiplos atributos-alvo, e que os atributos preditivos (meta-atributos) são os mesmos para todas as predições, o meta-regressor deve ser capaz de lidar com um grande número de meta-atributos e com a possibilidade de atributos irrelevantes em cada sub-problema de predição. Neste contexto, estudos sugerem que o desempenho preditivo do Random Forest não é significativamente afetado com atributos irrelevantes [Sun et al., 2013];

3. Random Forest demonstrou bons resultados quando aplicado como meta-regressor em uma meta-base com muitos atributos-alvo, no contexto de otimização de hiperparâmetros, em estudos como [Feurer et al., 2015b].

6.1.6 *Ranking* de Subconjuntos de Operadores

Para tornar possível a avaliação desta estratégia de Meta-Aprendizado separadamente a estratégia de otimização, aplicou-se os processos de avaliação comumente utilizados para este tipo de abordagem. Assim, os subconjuntos de operadores foram ordenados em forma de *ranking*, de acordo com o desempenho estimado.

Em geral, um *ranking* apresenta uma função de preferência sobre um conjunto de itens. Nestes experimentos, os itens são os subconjuntos de operadores, já a função de preferência, expressa o desempenho esperado dos subconjuntos de operadores em um conjunto de dados, tal que, se um subconjunto apresentar melhores resultados que outro, ele deve ser representado na posição mais alta do *ranking*.

A Seção 3.2.1 apresentou uma visão geral dos diversos tipos de *ranking* aplicáveis no contexto de Meta-Aprendizado. Nesta avaliação experimental, é utilizado o *ranking* por regressão, onde o desempenho de cada subconjunto de operadores é estimado separadamente e, posteriormente, é ordenado de maneira a estabelecer o *ranking*.

6.2 Aplicação e Avaliação de Meta-Aprendizado

Para comprovar a eficácia da estratégia de Meta-Aprendizado para predição de desempenho, é necessário demonstrar empiricamente que, de fato, a previsão do meta-regressor se aproxima da realidade e se o método de predição é capaz de produzir resultados que limitem o tempo gasto com experimentação, com deterioração reduzida na qualidade dos resultados. Para tanto, assim como na avaliação de algoritmos de Aprendizado de Máquina, é necessário definir medidas de avaliação de desempenho e uma estratégia para avaliar o método de predição.

No Meta-Aprendizado por regressão, que é o foco desta abordagem, os desempenhos esperados dos sub-conjuntos de operadores são inicialmente preditos pela aplicação do algoritmo Random Forest e combinados para a composição de *ranking*, de forma a prover a quantificação e comparação da abordagem de recomendação proposta.

As subseções a seguir apresentam as medidas de desempenho utilizadas em cada avaliação, a estratégia de avaliação e as estratégias de comparação (*baselines*).

6.2.1 Medida de Desempenho do Meta-Regressor

Na avaliação do meta-regressor, para cada elemento do conjunto de testes, um modelo de regressão é induzido para predizer a AUC para cada subconjunto de operadores. Para mensurar a qualidade das predições realizadas, calcula-se o seu desvio em relação ao AUC real obtido pelo subconjunto correspondente. A raiz quadrada do erro médio (RMSE, do inglês, *Root Mean Squared Error*) é empregada como medida de desempenho do Meta-Aprendizado, de modo que os valores menores de RMSE indicam uma menor discrepância entre os erros preditos e real e, portanto, estão associados aos melhores regressores.

6.2.2 Medida de Desempenho de *Ranking*

Os *rankings* gerados em Meta-Aprendizado são séries de dados pareados, isto é, dados onde existe uma conexão entre o número de membros correspondentes na amostra. Neste contexto, métodos de correlação de *ranking* são empregados para avaliar a monotonicidade entre *rankings*, ou seja, se suas variações estão relacionadas. Caso os valores nos *rankings* apresentem tendência a crescer ou a decrescer juntos, então há uma correlação positiva, com máximo de 1. Se os valores de um *ranking* crescem enquanto os do outro decrescem, então há uma correlação negativa com mínimo de -1. Uma correlação de 0 indica que os *rankings* não estão correlacionados.

A Correlação de Spearman é uma medida comumente empregada para medir a acurácia de *rankings* gerados por métodos de Meta-Aprendizado [Brazdil et al., 2008]. Esse coeficiente é calculado de acordo com a Equação 6.1, para rankings arbitrários $Ranking_A$ e $Ranking_B$, com n itens:

$$r_s = \frac{\sum_{i=1}^{n}(Ranking_{Ai} - \overline{Ranking_A}) * (Ranking_{Bi} - \overline{Ranking_B})}{\sqrt{\sum_{i=1}^{n}(Ranking_{Ai} - \overline{Ranking_A})^2 * (Ranking_{Bi} - \overline{Ranking_B})^2}} \quad (6.1)$$

onde $Ranking_{Ai}$ ($Ranking_{Bi}$) corresponde à posição do item i em $Ranking_A$ ($Ranking_B$). Enquanto $\overline{Ranking_A}$($\overline{Ranking_B}$) representa a média dessas posições.

Uma característica da correlação de Spearman é a atribuição de peso igual a todas as posições dos *rankings* considerados. Tal propriedade pode não ser suficientemente informativa em casos em que itens melhor posicionados devam ter maior importância no cálculo do coeficiente de correlação. Esta situação ocorre com frequência no contexto de recomendação de algoritmos, uma vez que o usuário tende a preferir executar algoritmos que estejam em posições superiores no *ranking* sugerido [Souza, 2010]. Assim, uma informação complementar para avaliar a qualidade de um *ranking* pode ser obtida com a utilização de medidas que realizem ponderação de posição, como Medida de Correlação de *Ranking* Ponderada (WRMC, do inglês, *Weighted Ranking Measure of Correlation*) [Pinto da Costa e Soares, 2005], apresentada na Equação 6.2.

$$r_w = 1 - \frac{6 \sum_{i=0}^{n} (Ranking_{Ai} - Ranking_{Bi})^2 * ((n - Ranking_{Ai} + 1) + (n - Ranking_{Bi} + 1))}{n^4 + n^3 + n^2 - n} \quad (6.2)$$

Neste caso, o primeiro termo do produto $(Ranking_{Ai} - Ranking_{Bi})^2$, assim como na correlação de Spearman, representa a distância entre $Ranking_A$ e $Ranking_B$. O segundo termo, $((n - Ranking_{Ai} + 1) + (n - Ranking_{Bi} + 1)$ representa não apenas a importância de $Ranking_{Ai}$, mas também a importância de $Ranking_{Bi}$. Os valores admissíveis para r_w também estão no intervalo $[-1, 1]$.

Nos experimentos realizados são adotadas ambas as medidas. A correlação de Spearman, por ser amplamente utilizada na literatura relacionada à Meta-Aprendizado, e a Medida de Correlação de *Ranking* Ponderada.

As medidas são aplicadas sobre o *ranking* ideal associado à um meta-exemplo e o *ranking* construído por um determinado método, afim de determinar a acurácia da predição realizada.

6.2.3 Método de Avaliação de Nível-Base

A acurácia do *ranking* (medida pelas correlações de Spearman e Correlação Ponderada) representa a similaridade entre o *ranking* recomendado e o ideal. Porém esta medida não contém informações sobre o valor, isto é, o resultado final, no caso da recomendação ser seguida pelo usuário [Brazdil et al., 2008]. No contexto do usuário, dado um *ranking* de operadores, a escolha de qual utilizar é ponderada em relação ao custo benefício. Neste caso, testar várias das opções sugeridas aumenta a chance de escolher a melhor e, geralmente, as primeiras posições do ranking são as escolhidas. Porém, o custo computacional aumenta a cada possibilidade executada.

Uma vez que a ordem do *ranking* tende a ser seguida, [Brazdil et al., 2008] sugerem a abordagem *Top-N* para avaliar quantos itens em particular um usuário deve seleci-

onar. Este método consiste em simular que os *top-n* itens do *ranking* serão selecionados, variando o valor de *n*. Nesta pesquisa, o método Top-N é utilizado para avaliar os *rankings* em termo do ganho de desempenho, sem levar em conta o tempo de execução, o Algoritmo 6.1 apresenta a adaptação do Top-N para este fim.

1: **entrada** : $Ranking_{rec} = \{r_1, ..., r_n\}$, $Desempenho_{teste}$
2: $tg_1 = d_{r_1}$
3: **for** i $\in \{2, ..., n\}$ **do**
4: Determinar o ganho de executar o algoritmo rankeado na i-ésima posição depois de executar os algoritmos melhor rankeados
5: $tg_i = \max(tg_{i-1}, d_{r_i})$
6: **end for**
7: **return tg**

Algoritmo 6.1 – Algoritmo de avaliação Top-N, adaptado de [Brazdil et al., 2008]

Onde $Ranking_{rec}$ é o *ranking* recomendado para o conjunto de testes, e $Desempenho_{teste}$ é o desempenho real obtido pela execução dos operadores no conjunto de dados de teste. A saída **tg** corresponde as estimativas dos Top-N desempenhos do *ranking* recomendado $Ranking_{teste}$ no conjunto de dados *teste*.

Esta estratégia é aplicada para cada conjunto de dados de teste e a média dos valores de desempenho é agrupada para avaliar a recomendação.

6.2.4 Estratégia de Avaliação

Uma vez definidas as medidas para avaliar o desempenho do meta-regressor e, posteriormente, do *ranking* predito, pode-se avaliar os métodos utilizados para sua construção. Tendo como base o *ranking* ideal, diz-se que um método é mais acurado que o outro se este for capaz de construir *rankings* mais similares ao ideal. Para realizar esta análise, emprega-se técnicas de amostragem (apresentadas no Capítulo 2).

Nestes experimentos, é utilizada a estratégia de avaliação *leave-one-out* (LOO), onde cada meta-exemplo é sucessivamente separado para teste e os demais para treinamento.

Para avaliação estatística dos resultados é utilizado o teste de Friedman seguido do pós-teste de Bonferroni-Dunn para múltiplas comparações [Demšar, 2006].

6.2.5 Métodos de Comparação

Os resultados obtidos utilizando Random Forest como meta-regressor foram comparados a 3 outros métodos de geração de *ranking*:

- Ranking padrão: também conhecido como *ranking* médio, consiste em sintetizar a informação dos desempenhos preditos de todos os meta-exemplos de treinamento em um único *ranking*, calculado a partir da média de cada posição;
- Ranking gerado utilizando k-NN como meta-regressor, com k = 1;
- Ranking gerado utilizando k-NN como meta-regressor, com k = 3;

6.3 Avaliação de Desempenho do Meta-Regressor

A avaliação do meta-regressor ocorre de acordo com LOO aplicado sobre os 322 meta-exemplos considerados. A Tabela 6.4 mostra os resultados da meta-regressão para os dados caracterizados utilizando-se os meta-atributos. Os valores apresentados correspondem a medida RMSE dos regressores Random Forest, 1-NN, 3-NN e *Ranking* Padrão quando utilizados para predizer o desempenho dos 243 subconjuntos de operadores considerados. Já a Figura 6.2 apresenta a distribuição do RMSE para cada regressor.

Tabela 6.4 – Desempenho médio e desvio padrão dos meta-regressores na predição de desempenho dos modelos-completos

	Random Forest	*Ranking* Padrão	1-NN	3-NN
RMSE médio	**0.1181**	0.1576	0.1823	0.1536
Desvio Padrão	**0.0376**	0.0529	0.0568	0.0491

Como pode-se notar, o meta-regressor Random Forest apresenta o melhor desempenho geral, sendo superior a todos os demais. Assim, espera-se que o erro médio de sua estimação em novos meta-exemplos seja inferior à 12% para os 243 subconjuntos de operadores.

A Figura 6.3 apresenta os resultados do teste estatístico de Friedman, com pós teste de Bonferoni-Dunn em um gráfico de Diferença Crítica. A hipótese nula considerada foi a de que não existe diferença estatística entre o Random Forest e cada um dos demais métodos. Foi considerada uma confiança de 95% no teste de Friedman e 90% no teste de Bonferroni-Dunn. Os meta-regressores foram ordenados de acordo com o erro médio (menor é melhor), e aqueles conectados com uma linha horizontal são estatisticamente equivalentes.

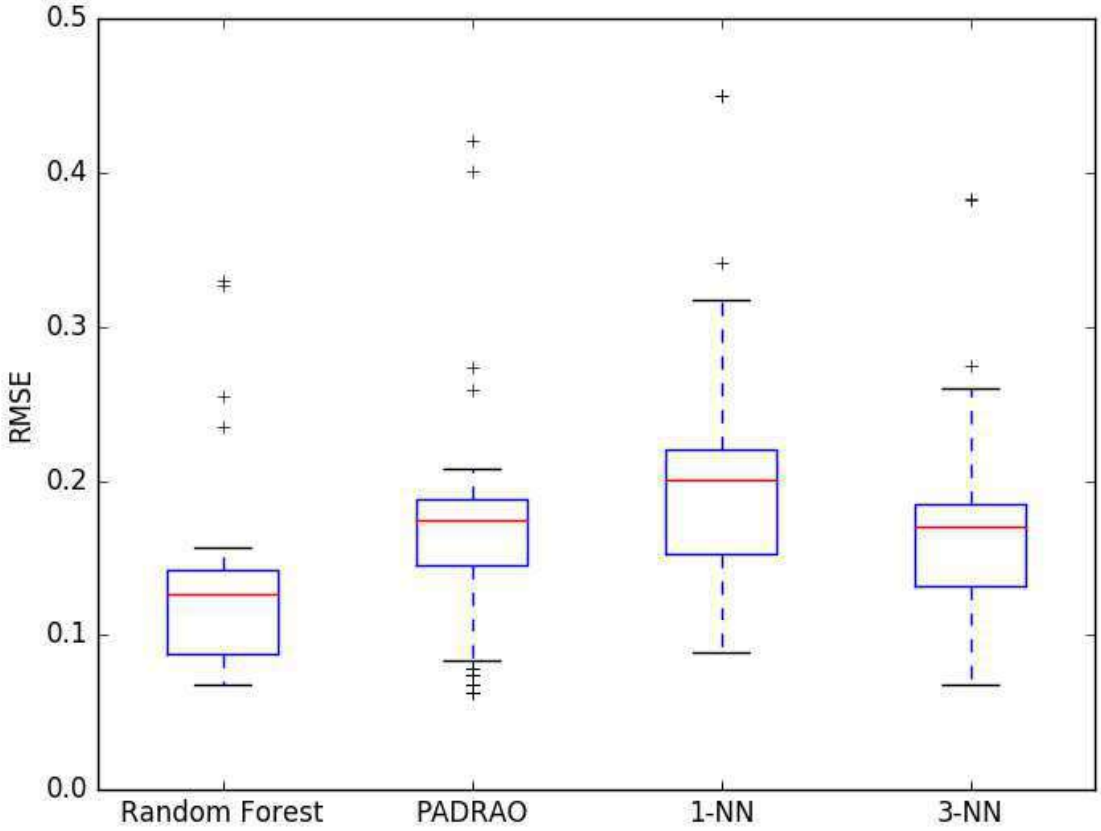

Figura 6.2 – Desempenho dos meta-regressores Random Forest, 1-NN, 3-NN e PADRAO na estimativa de erro de 243 subconjuntos de operadores

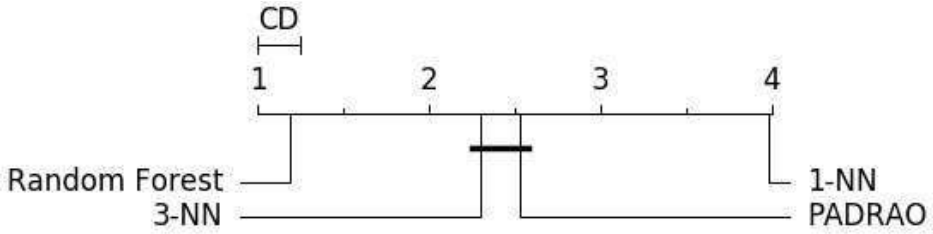

Figura 6.3 – Diagrama de Diferença Crítica entre os Meta-Regressores na avaliação de predição de desempenho dos subconjuntos de operadores

Percebe-se que a hipótese nula foi rejeitada, indicando que o desempenho do Random Forest é estatisticamente melhor aos demais, de acordo com o nível de confiança requerido. A partir desta análise também é possível perceber que não foi apresentada diferença estatística entre o 3-NN e o desempenho do *ranking* padrão. Isto, juntamente com o método 1-NN tendo sido avaliado com o pior desempenho, corroboram a deficiência do método k-NN como meta-regressor na tarefa de seleção de subconjuntos de operadores.

6.4 Avaliação da Construção de *Rankings*

Na seção anterior foi apresentada a avaliação do desempenho do meta-regressor, ou seja, em quão bom ele é para predizer o desempenho dos subconjuntos de operadores. Nesta seção é apresentada a qualidade da recomendação realizada.

As Tabelas 6.5 e 6.6 exibem os resultados da utilização dos métodos Random Forest, 1-NN e 3-NN para predição dos desempenhos utilizados na construção dos *rankings*. Suas entradas correspondem as médias de acurácia (utilizando o coeficiente de Spearman, r_s e a Medida de Correlação Ponderada, respectivamente) dos *rankings* preditos no decorrer do LOO para os 322 problemas considerados. Adicionalmente, exibe-se também as correlações resultantes da aplicação do *Ranking* Padrão.

Tabela 6.5 – Acurácia Média dos *Rankings* (r_s gerados pelos meta-regressores, calculada com a Correlação de Spearman)

	Random Forest	*Ranking* Padrão	1-NN	3-NN
Spearman médio	**0.7464**	0.7077	0.6770	0.7149
Desvio Padrão	**0.2322**	0.2416	0.2599	0.2341

Tabela 6.6 – Acurácia Média dos *Rankings* (r_w gerados pelos meta-regressores, calculada com a Medida de Correlação Ponderada)

	Random Forest	*Ranking* Padrão	1-NN	3-NN
Correlação Ponderada Média	**0.7464**	0.6829	0.6503	0.7265
Desvio Padrão	**0.2778**	0.2866	0.2713	0.2778

Através da aplicação dos testes de Friedman e pós-teste de Bonferroni-Dunn, com 95% e 90% de confiança, respectivamente, observou-se novamente a superioridade do Random Forest em relação às demais abordagens, uma vez que o teste apontou que há significância estatística nos seus resultados. Neste caso, uma vez que os resultados foram ranqueados de acordo com as medidas de correlação (Spearman e Correlação Ponderada), uma posição de *ranking* maior é melhor.

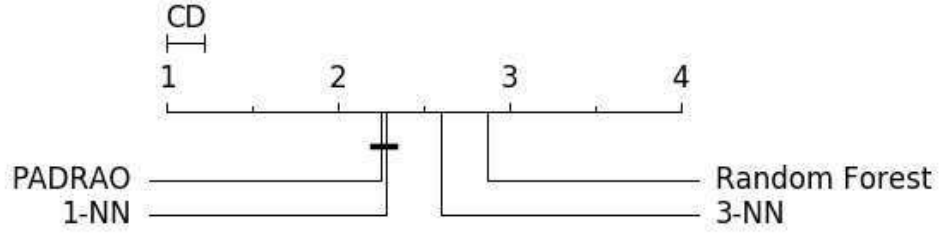

Figura 6.4 – Diagrama de Diferença Crítica entre os Meta-Regressores na avaliação da geração de *ranking* usando a Medida de Correlação Spearman

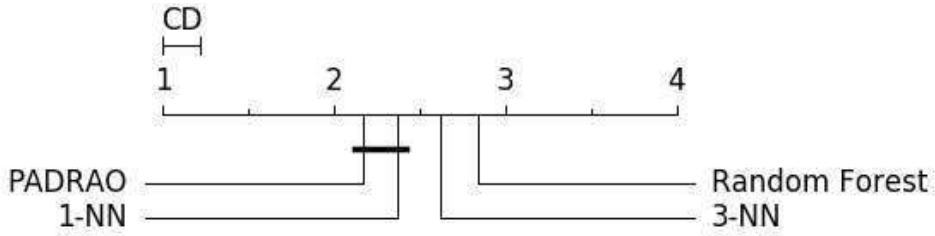

Figura 6.5 – Diagrama de Diferença Crítica entre os Meta-Regressores na avaliação da geração de *ranking* usando a Medida de Correlação Ponderada

Comparando os resultados desta avaliação para a construção de *rankings* com os resultados da avaliação de predição de desempenho dos subconjuntos de operadores percebemos que o Random Forest mantém-se consistente como o método mais adequado em ambos os casos.

6.5 Avaliação de Nível-Base

Esta seção apresenta os resultados da avaliação dos ganhos de desempenho em nível-base do *ranking* gerado pelo Random-Forest. A Figura 6.6 apresenta a média dos resultados do algoritmo Top-N aplicado aos 322 conjuntos de dados.

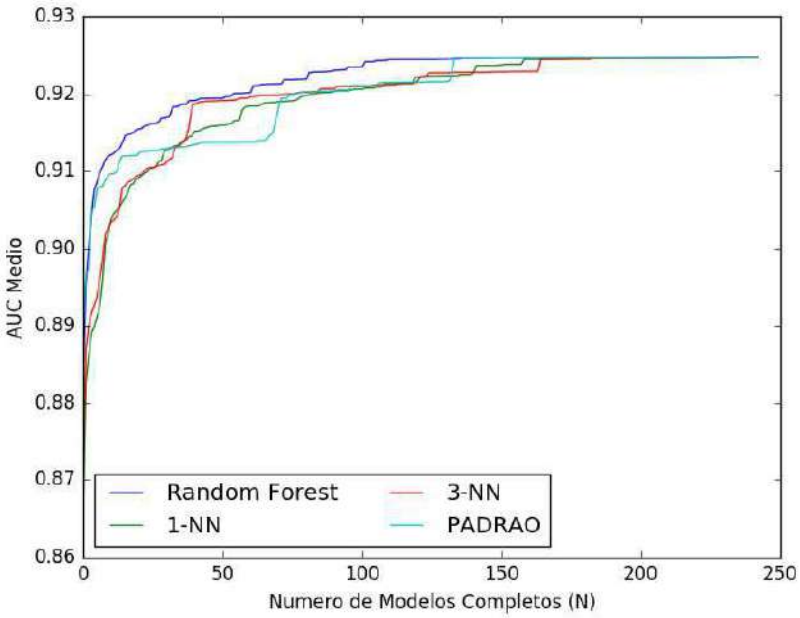

Figura 6.6 – Média do Desempenho Top-N entre os 322 conjuntos de dados de Random Forest, 1-NN, 3-NN e Ranking Padrão

Comparando com o desempenho obtido pelos meta-regressores 1-NN, 3-NN e Padrão, vemos que os ganhos obtidos pelo *ranking* gerado pelo Random-Forest são de

maneira geral consistentemente mais próximos ao desempenho real do que as demais abordagens.

Estes resultados corroboram os resultados das avaliações anteriores em relação a superioridade do Random Forest como meta-regressor.

6.6 Considerações Finais do Capítulo

Uberband inclui Meta-Aprendizado como parte da solução para Seleção de Processo Completo, mais especificamente, Meta-Aprendizado é utilizado para gerar pesos baseados no desempenho previsto dos subconjuntos de operadores, de maneira a influenciar a amostragem realizada pelo otimizador. Assim, subconjuntos com maior desempenho predito pela estratégia de Meta-Aprendizado serão favorecidos na fase de otimização, onde serão configurados, visando aumentar a qualidade da recomendação final.

Neste Capítulo foi apresentada a avaliação da estratégia de Meta-Aprendizado de forma isolada, de maneira a medir a qualidade da recomendação oferecida, isto é, o quanto o erro de estimativa de desempenho da estratégia de meta-aprendizado pode impactar na distribuição dos pesos dos subconjuntos de operadores.

7. AVALIAÇÃO EXPERIMENTAL UBERBAND

Neste Capítulo são apresentadas diversas análises empíricas que avaliam o desempenho do Uberband sob diferentes aspectos. As avaliações foram divididas em:

1. Análise dos componentes do Uberband;

2. Análise comparativa com soluções estado-da-arte de Seleção de Processo Completo;

O restante do Capítulo está dividido da seguinte maneira: A Seção 7.1.1 apresenta a configuração experimental, que inclui o espaço de busca de operadores e hiper-parâmetros utilizados nestes experimentos e os conjuntos de dados avaliados. A Seção 7.2 apresenta a avaliação dos componentes de Meta-Aprendizado e Otimização do Uberband. Por fim, a Seção 7.3 apresenta a avaliação do Uberband junto a trabalhos proeminentes da área de Seleção de Processo Completo.

7.1 Configuração Experimental

7.1.1 Definição do Espaço de Busca Experimental

Conforme apresentado no Capítulo 6, os operadores utilizados para avaliar o Uberband são relacionados à Seleção de Características e Algoritmos de Classificação.

A Figura 7.1 apresenta os componentes experimentais do Uberband. Foram utilizados os 243 subconjuntos de operadores da fase de Meta-Aprendizado. Porém, enquanto na fase de Meta-Aprendizado apenas os subconjuntos de operadores foram avaliados, na fase de otimização também são levados em consideração os hiper-parâmetros desses operadores. A parametrização de cada operador de seleção de características resultou em um espaço de 122 hiper-parâmetros. Alguns desses parâmetros são condicionais, isto é, só são ativados se seus respectivos componentes são selecionados.

A Tabela 7.1 lista os 27 algoritmos de aprendizado e a Tabela 7.2 lista os 8 métodos de pré-processamento, respectivamente, com a quantidade de hiper-parâmetros correspondentes de cada um, que foram distinguidos em condicionais (cond), categóricos (cat) e numéricos (num). A descrição dos hiper-parâmetros e algoritmos pode ser consultada no Anexo B.

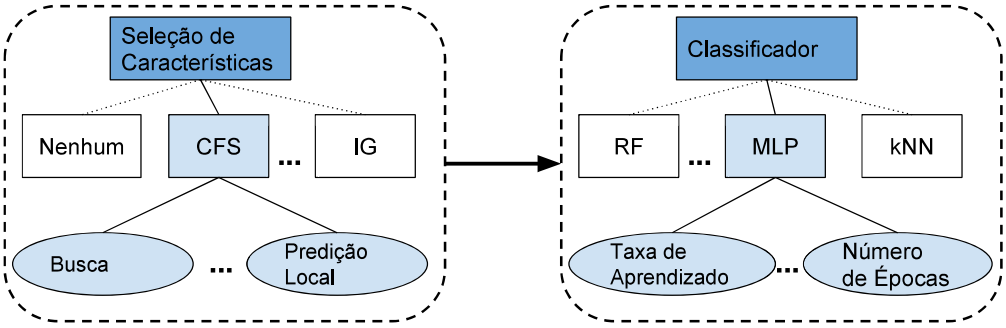

Figura 7.1 – Estrutura do espaço de busca experimental do Uberband. Retângulos denotam hiper-parâmetros pai, enquanto elipses denotam hiper-parâmetros folha. Formas coloridas denotam um exemplo de processo completo selecionado.

Tabela 7.1 – Algoritmos de AM e número de hiper-parâmetros correspondentes

nome	#cond	#cat	#num	nome	#cond	#cat	#num
AdaBoost M1	1	1	4	OneR	0	0	1
BayesNet	0	2	0	PART	1	2	2
DecisionStump	0	1	0	RandomForest	2	0	5
DecisionTable	0	4	0	RandomTree	3	1	7
HoeffdingTree	0	0	6	RandomSubSpace	0	0	3
J48	2	6	3	RandomCommitte	0	0	2
JRip	0	2	1	REPTree	1	1	4
IBk	1	4	1	Stacking	0	0	2
LMT	1	5	3	SimpleLogistic	1	2	2
LogitBoost	0	1	4	SGD	0	2	3
Logistic	0	0	1	SVM	0	3	2
MultilayerPerceptron	0	5	2	Vote	0	1	1
NaiveBayes	1	2	0	VotedPerceptron	0	0	3
				ZeroR	0	1	0

Tabela 7.2 – Métodos de Seleção de Características (Avaliadores (A) e Métodos de Busca (B)) e o número de hiper-parâmetros correspondentes

nome	Tipo	#cond	#cat	#num
CFSSubsetEval	A	1	2	1
Classifier Attribute Eval	A	1	1	0
Gain Ratio Eval	A	1	1	0
Info Gain Eval	A	1	2	0
OneR Attribute Eval	A	1	1	2
RELIEF Eval	A	1	1	3
Symmetrical Uncert	A	1	1	0
Wrapper Sub Eval	A	1	2	2
Best First	B	0	0	2
Greedy Stepwise	B	0	3	0
Ranker	B	0	0	1

7.1.2 Conjuntos de Dados

Para avaliar a solução proposta, foram selecionados 60 conjuntos de dados da coleção de conjuntos inicialmente utilizada no processo de Meta-Aprendizado, provenientes

da plataforma OpenML. Os conjuntos foram limitados aqueles com no mínimo 1000 instâncias e máximo 50.000, máximo de 1000 atributos e menos de 20% de valores ausentes, de maneira a acelerar os experimentos e excluir conjuntos de dados com poucos recursos (instâncias), o que inviabilizaria alocação adaptativa. Os 60 conjuntos de dados usados nos experimentos são apresentados na Tabela 7.3.

A meta-base de conhecimento utilizada para a etapa de Meta-Aprendizado é a mesma apresentada no Capítulo 6, contendo 322 conjuntos de dados. Para não avaliar novamente um conjunto de dados que já tenha sido utilizado na etapa de Meta-Aprendizado, foi realizado um *leave-one-out* com os 60 conjuntos desses experimentos, de forma que quando um desses conjuntos for avaliado, são utilizados os meta-dados dos outros 321.

Cada conjunto de dados foi inicialmente amostrado em treino (60%), teste (20%) e validação (20%). Durante a execução do algoritmo, o conjunto de treinamento foi amostrado de acordo com o procedimento de alocação de recursos adaptativo, apresentado na Seção 5.3. Conforme o número de recursos a ser usado para cada configuração, r, é definido, o número correspondente de instâncias é amostrado aleatoriamente do conjunto de treinamento. Esse subconjunto de instâncias é utilizado para treinar a configuração de processo completo e o desempenho do modelo gerado é avaliado no conjunto de validação. O conjunto de testes é usado apenas nos melhores processos completos identificados no processo de otimização.

Tabela 7.3 – Conjuntos de Dados de Teste para Avaliação Comparativa do Uberband

dados	#Inst	#Atrib	#Cat	#Num	dados	#Inst	#Atrib	#Cat	#Num
2dplanes	40768	11	1	10	houses	20640	9	1	8
ada_prior	4562	15	9	6	hypothyroid	3772	30	23	7
adult	32561	16	9	7	jm1	10885	22	1	21
adult-census	48842	15	13	2	kc1	2109	22	1	21
ailerons	13750	41	1	40	kr-vs-kp	3196	37	37	0
analcatdata_halloffame	1340	18	3	15	letter	20000	17	1	16
analcatdata_supreme	4052	8	1	7	mfeat-morphological	2000	7	1	6
balloon	2001	3	1	2	mfeat-pixel	2000	241	241	0
bank32nh	8192	33	1	32	mfeat-zernike	2000	48	1	47
bank8FM	8192	9	1	8	mozilla4	15545	6	1	5
car	1728	7	7	0	mushroom	8124	23	23	0
cmc	1473	10	8	2	nursery	12960	9	9	0
colleges_aaup	1161	17	4	13	optdigits	5620	65	1	64
colleges_usnews	1302	35	3	32	page-blocks	5473	11	1	10
cpu_act	8192	22	1	21	pc1	1109	22	1	21
credit-g	1000	21	14	7	pc2	5589	37	1	36
delta_ailerons	7129	6	1	5	pc3	1563	38	1	37
delta_elevators	9517	7	1	6	pc4	1458	38	1	37
electricity-normalized	45312	9	2	7	pendigits	10992	17	1	16
elevators	16599	19	1	18	pollen	3848	6	1	5

Tabela 7.3 – Conjuntos de Dados de Teste para Avaliação Comparativa do Uberband

dados	#Inst	#Atrib	#Cat	#Num	dados	#Inst	#Atrib	#Cat	#Num
fri_c0_1000_5	1000	6	1	5	puma8NH	8192	9	1	8
fri_c1_1000_10	1000	11	1	10	quake	2178	4	1	3
fri_c2_1000_25	1000	26	1	25	rmftsa_sleepdata	1024	3	2	1
fri_c2_1000_5	1000	6	1	5	scene	1024	3	2	1
fri_c3_1000_5	1000	6	1	5	segment	2310	20	1	19
fri_c4_1000_10	1000	11	1	10	sick	3772	30	23	7
fri_c4_1000_25	1000	26	1	25	spambase	4601	58	1	57
fried	40768	11	1	10	splice	3190	62	62	0
gina_agnostic	3468	971	1	970	thyroid_sick	3772	30	23	7
house_16H	22784	17	1	16	wind	6574	15	1	14

7.2 Análise dos Componentes do Uberband

Este primeiro conjunto de experimentos avalia o algoritmo Uberband quando deparado com diferentes configurações de hiper-parâmetros e em termos de qualidade dos seus componentes individuais.

7.2.1 Método de Avaliação

Para a Avaliação dos Componentes do Uberband, tanto de Meta-Aprendizado quanto de Otimização, foram realizados experimentos comparando o Uberband com versões sem o respectivo componente, da seguinte maneira:

1. Para avaliar o componente de Meta-Aprendizado, Uberband foi comparado com uma versão que faz apenas o processo de otimização, fazendo tanto a amostragem dos subconjuntos de operadores quanto a amostragem de hiper-parâmetros aleatoriamente;

2. Para avaliar o componente de Otimização, Uberband foi comparado com uma versão do algoritmo Hyperband (apresentado no Capítulo 3) incrementado do componente de Meta-Aprendizado;

3. Para uma avaliação conjunta dos componentes, Uberband foi comparado com o algoritmo Hyperband original.

Na avaliação de cada componente, os algoritmos são comparados, levantando-se inicialmente as seguintes hipóteses:

- Hipótese Nula, H_0: Não há diferença significativa de desempenho com e sem o uso do componente (Meta-Aprendizado, Otimização, Ambos).

- Hipótese Alternativa, H_1: Há diferença significativa entre os resultados obtidos com e sem o uso do componente (Meta-Aprendizado, Otimização, Ambos).

Ao final desta avaliação, deseja-se saber se o desempenho preditivo do algoritmo do Uberband com Meta-Aprendizado e Otimização, é superior ao desempenho preditivo sem algum desses componentes.

Por cada comparação se tratar de uma avaliação de dois algoritmos, foi utilizado o teste de Wilcoxon [Wilcoxon, 1945] conforme indicado por Demŷsar [Demšar, 2006]. O teste de Wilcoxon (*Wilcoxon signed-ranks test*) é um teste não-paramétrico que classifica as diferenças de desempenho de dois algoritmos para cada conjunto de dados, ignorando os sinais, e comparando os *ranks* para diferenças positivas e negativas. Neste teste, com α = 0.05 e 60 conjuntos de dados, a hipótese nula pode ser rejeitada se o coeficiente *z* é menor que -1.96.

7.2.2 Avaliação de Hiper-Parâmetros

Por ser um algoritmo de aprendizado, Uberband não é livre de hiper-parâmetros e seus efeitos devem ser analisados cuidadosamente. Por ser baseado no Hyperband, este algoritmo por definição possui os hiper-parâmetros *R* e η.

O parâmetro *R* representa o número máximo de recursos que uma configuração pode receber. Isto é importante quando são avaliados cenários onde diferentes tipos de recursos são disponíveis, como é o caso da otimização de hiper-parâmetros de um algoritmo. Conforme explicado no Capítulo 6, no caso da SPC, ao lidar com diferentes tipos de operadores, recursos específicos (por exemplo, número de iterações, número de épocas) deixam de estar disponíveis. Por conta disso, como recurso que pode ser usado em comum entre os diferentes operadores foi definido o número de instâncias do conjunto de treinamento. Logo, o número máximo de recursos que uma configuração de processo completo pode usar para o treinamento é o tamanho total do conjunto de treinamento. Em virtude disso, o parâmetro *R* não será avaliado.

Já o parâmetro η controla o quão agressivamente o SuccessiveHalving elimina as configurações com pior desempenho, ou seja, a proporção de descarte de configurações. Grandes valores de η correspondem a uma eliminação mais agressiva e, consequentemente uma conclusão mais rápida do processo de otimização, uma vez que são retidas $\frac{1}{\eta}$ configurações de um total de $\lfloor \log_\eta^n \rfloor + 1$ iterações de eliminação com *n* configurações. Li et al. [Li et al., 2017], sugerem a escolha de η = 3. Os resultados a seguir comparam esse

valor com os valores de $\eta = 2$ e $\eta = 4$, de maneira avaliar possibilidades menos e mais rígidas de eliminação de configurações de processo completo.

Para cada valor de η, Uberband foi executado 5 vezes em cada um dos 60 conjuntos de dados apresentados na Tabela 7.3. Os resultados foram então agregados de acordo com a média de desempenho (medido em AUC) e de tempo de execução (medido em segundos), tanto do conjunto de validação quanto do conjunto de testes. A Figura 7.2 apresenta o desempenho do Uberband com diferentes valores do parâmetro η medido em AUC.

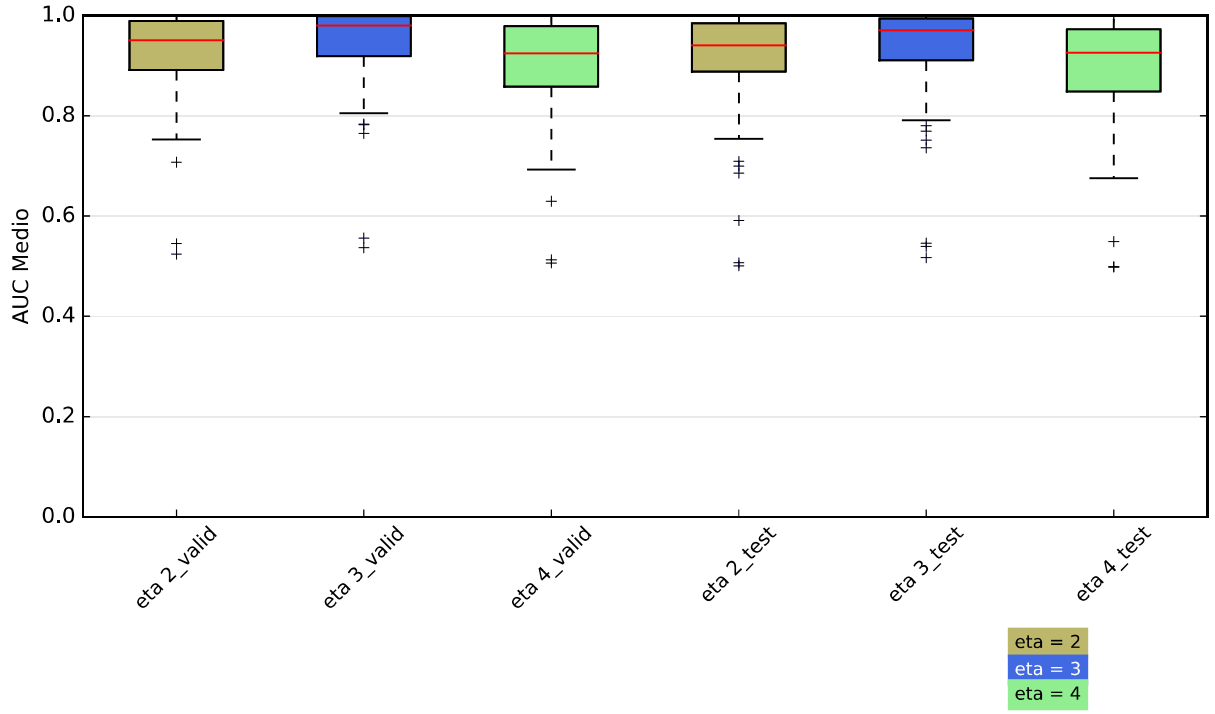

Figura 7.2 – Avaliação de Desempenho (em AUC) do Uberband variando o Hiper-Parâmetro η.

De acordo com esses valores, pode-se observar que a abordagem com $\eta = 3$ apresenta o melhor desempenho, embora muito próximo do desempenho de $\eta = 2$. Por outro lado, considerando que um valor maior de η corresponde a uma eliminação maior de configurações e a avaliação mais rápida, o $\eta = 3$ tende a ser amplamente uma abordagem mais promissora. Portanto, avaliando o custo benefício obtido com esses resultados, o valor $\eta = 3$ será utilizado como padrão do Uberband nos restantes dos experimentos apresentados neste Capítulo.

7.2.3 Avaliação da Estratégia de Meta-Aprendizado

Para avaliar a estratégia de Meta-Aprendizado em conjunto com o otimizador, foram executados experimentos com e sem esta estratégia. Para tanto, o algoritmo Uberband foi alterado de maneira a realizar apenas amostragens aleatórias, tanto dos subconjuntos de operadores, quanto dos hiper-parâmetros, essa versão é referida por Uberband sem Meta-Aprendizado (UB_sem_meta). A Figura 7.3 apresenta uma visão geral das abordagens comparadas, o pseudo-código de UB_sem_meta é apresentado no Apêndice A.

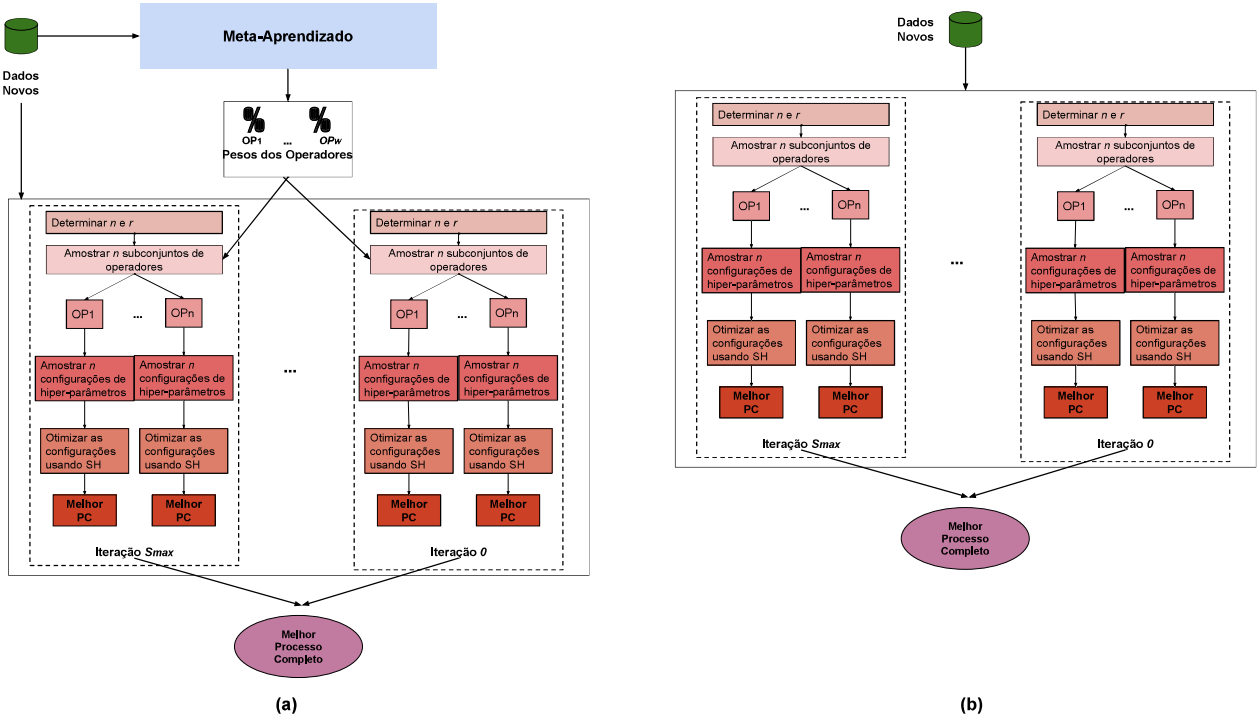

Figura 7.3 – Visão Geral do Uberband com Meta-Aprendizado (a) e sem Meta-Aprendizado (b) para auxílio no processo de otimização

Uberband e Uberband sem Meta-Aprendizado foram executados 5 vezes em cada um dos 60 conjuntos de dados. A Figura 7.4 apresenta o resultado da distribuição do desempenho do melhor processo encontrado (medido em AUC) por cada algoritmo, nas 5 execuções realizadas, tanto para o conjunto de validação quanto de testes e a Figura 7.5 apresenta a comparação de desempenho entre as estratégias. A Tabela 7.4 apresenta a média e o desvio padrão do desempenho das duas abordagens.

Conforme pode-se observar, Uberband com Meta-Aprendizado obteve o melhor desempenho, tanto na validação quanto no teste: comparado com o *baseline*, podemos notar que Uberband obteve a maior média de desempenho, bem como a menor variabilidade, principalmente nos dados de teste.

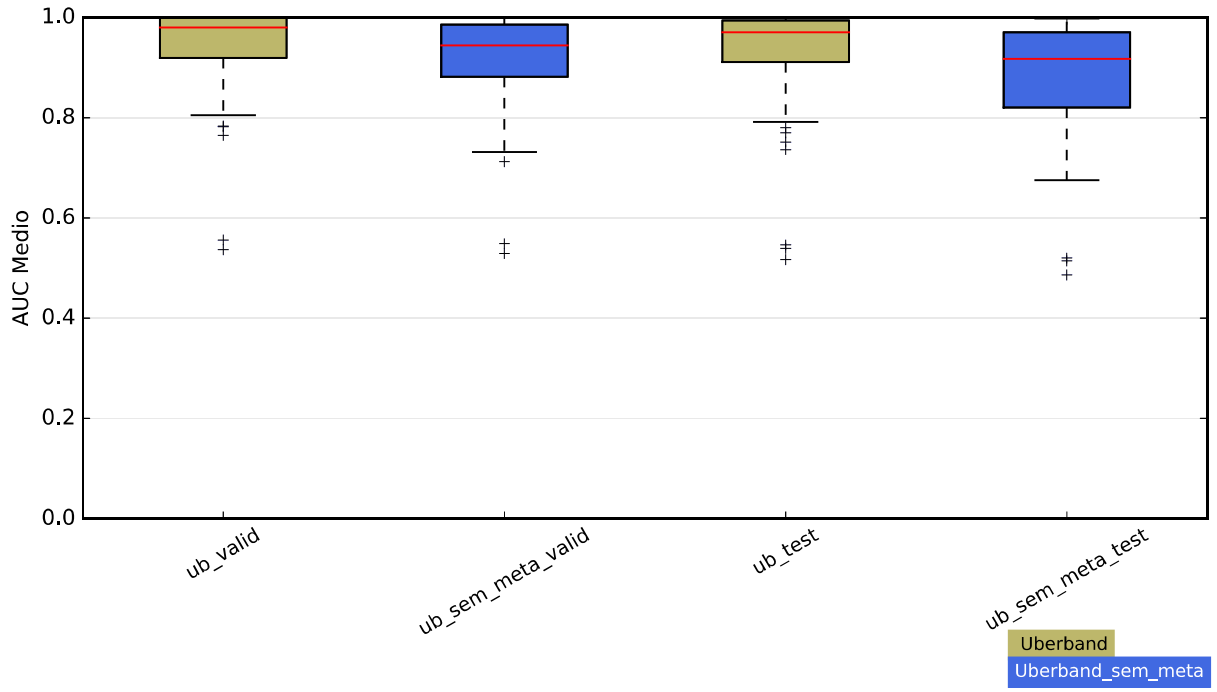

Figura 7.4 – Distribuição de Desempenho (em AUC) do Uberband comparado com Uberband sem Meta-Aprendizado.

Tabela 7.4 – Média e desvio-padrão do desempenho (AUC) do Uberband comparado com Uberband sem Meta-Aprendizado

	Validação		Teste	
	Uberband	Uberband sem Meta	Uberband	Uberband sem Meta
Média	**0.936**	0.908	**0.921**	0.878
Desvio Padrão	**0.098**	0.107	**0.115**	0.126

Nesta avaliação, deve-se levar em consideração que apenas a amostragem de subconjuntos de operadores é auxiliada pelo Meta-Aprendizado e, para ambas as estratégias avaliadas, a amostragem de hiper-parâmetros é feita aleatoriamente. Neste caso, é possível que um conjunto de hiper-parâmetros amostrado por um algoritmo seja melhor do que para o outro.

Para evidenciar o impacto destes resultados, aplicou-se o teste de Wilcoxon. Considerando Uberband como algoritmo 1 e Uberband sem meta-aprendizado como algoritmo 2, e o número de conjunto de dados igual à 60, obteve-se $z = -6.25$ para o conjunto de validação e $z = -6.43$ para o conjunto de testes. Como $z < -1.96$, em ambos os casos, esse resultado rejeita a hipótese nula de igualdade e corrobora a significância do componente de Meta-Aprendizado para o Uberband.

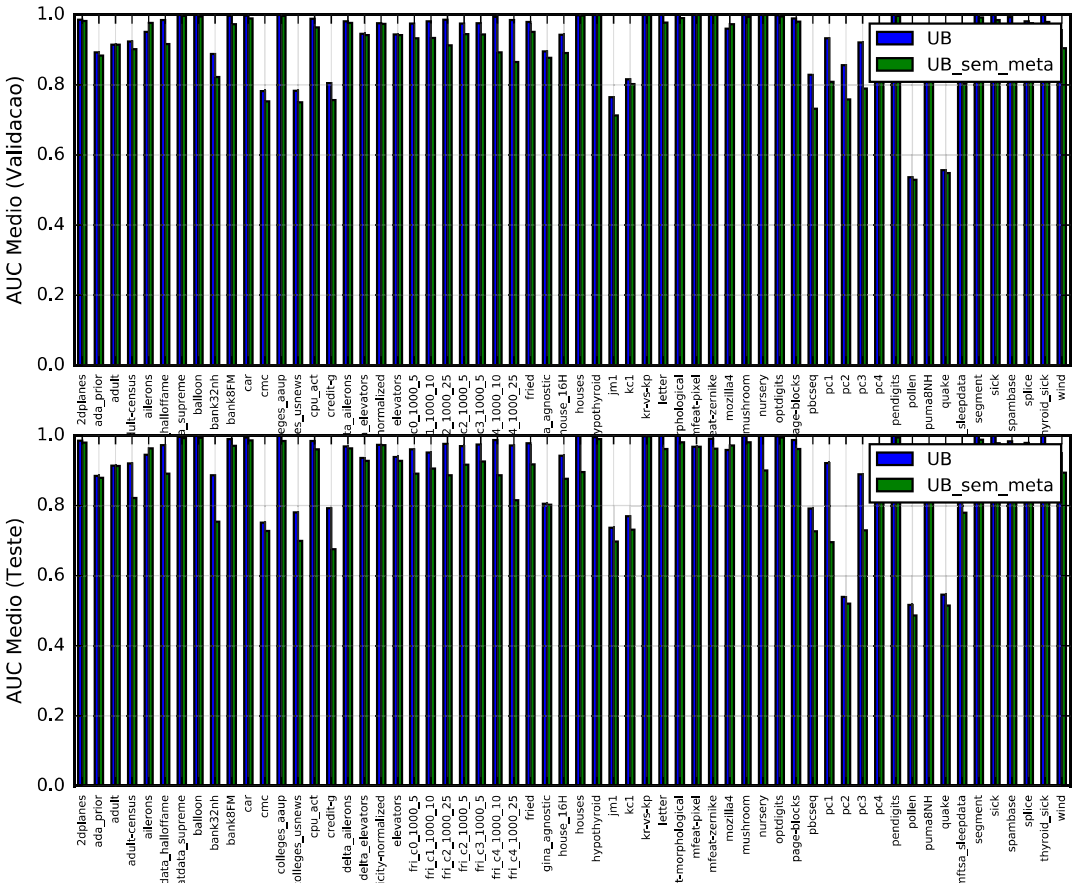

Figura 7.5 – Comparação de Desempenho (em AUC) do Uberband comparado com Uberband sem Meta-Aprendizado.

7.2.4 Avaliação da Estratégia de Otimização

Analogamente à seção anterior, para avaliar separadamente estratégia de otimização, foram executados experimentos com e sem este componente.

Para tanto, foi desenvolvida uma versão do algoritmo Hyperband, doravante chamada meta-hyperband, (Algoritmo B.1) que inclui a estratégia de Meta-Aprendizado. Neste caso, o MA é utilizado na amostragem dos subconjuntos de operadores, e para cada subconjunto, os hiper-parâmetros são amostrados aleatoriamente. A otimização é feita seguindo o procedimento padrão do Hyperband. A Figura 7.6 apresenta uma visão geral desta estratégia quando comparada ao Uberband, o pseudo-código desde algoritmo é apresentado no Apêndice B.

A Figura 7.7 apresenta os resultados dessa avaliação, a partir da distribuição do desempenho medido em 5 execuções de cada estratégia e a Figura 7.8 apresenta a comparação do desempenho, enquanto a Tabela 7.5 apresenta a média e o desvio padrão dessas execuções . De acordo com esses resultados, o Uberband utilizando a estratégia de otimi-

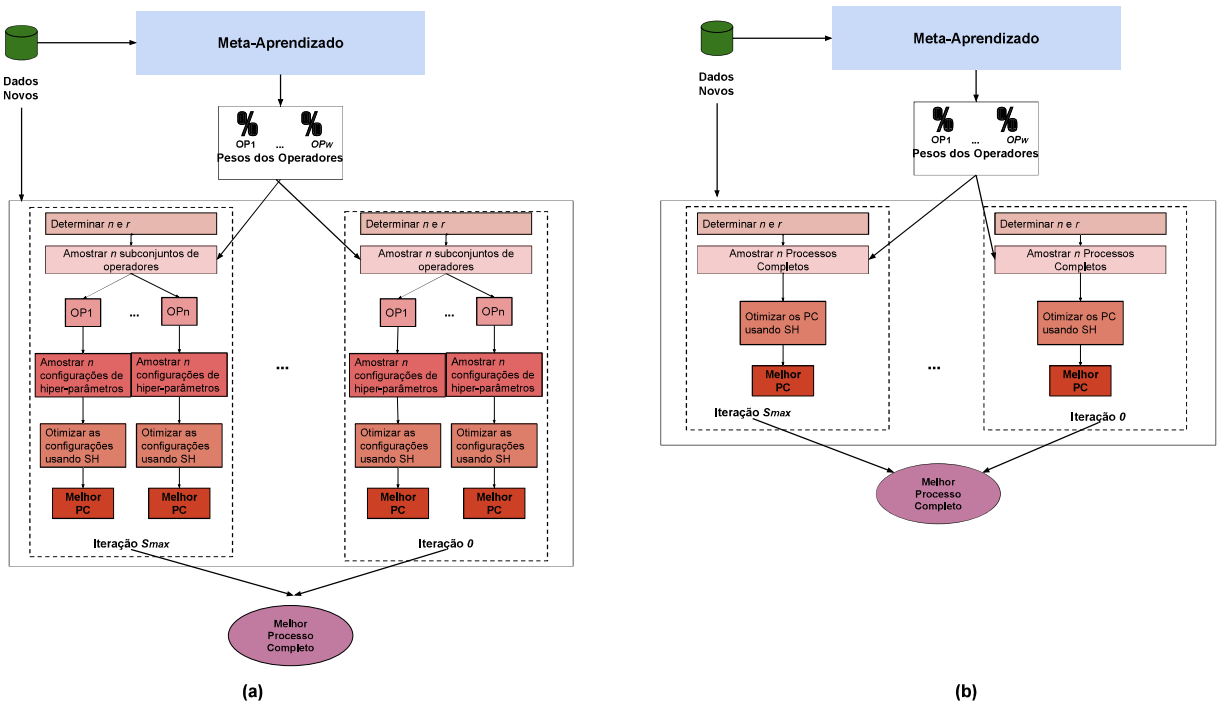

Figura 7.6 – Visão Geral do Uberband (a) e Hyperband com Meta-Aprendizado (b)

zação que explora tanto o conjunto de operadores quanto o conjunto de hiper-parâmetros é superior a estratégia de otimização que favorece mais a exploração do conjunto de operadores, pois esta última realiza apenas um nível de otimização. Neste caso, mesmo usando Meta-Aprendizado, a quantidade de configurações de hiper-parâmetros otimizado pelo Hyperband fica limitado a uma por subconjunto de operadores amostrado.

Tabela 7.5 – Média e desvio-padrão do desempenho (em AUC) do Uberband comparado com Hyperband acrescido de Meta-Aprendizado para amostragem do subconjunto de operadores

	Validação		Teste	
	Uberband	Meta-Hyperband	Uberband	Meta-Hyperband
Média	**0.936**	0.899	**0.921**	0.873
Desvio Padrão	**0.098**	0.112	**0.115**	0.126

A avaliação estatística também aplicou o teste de Wilcoxon, com Uberband sendo o algoritmo 1 e Hyperband com Meta-Aprendizado sendo o algoritmo 2. Dessa forma, para 60 conjuntos de dados, obteve-se $z = -6.64$ para o conjunto de validação e $z = -6.65$, para o conjunto de testes. Assim como na avaliação anterior, esse resultado rejeita a hipótese nula de igualdade e corrobora a significância do componente de Otimização para o Uberband.

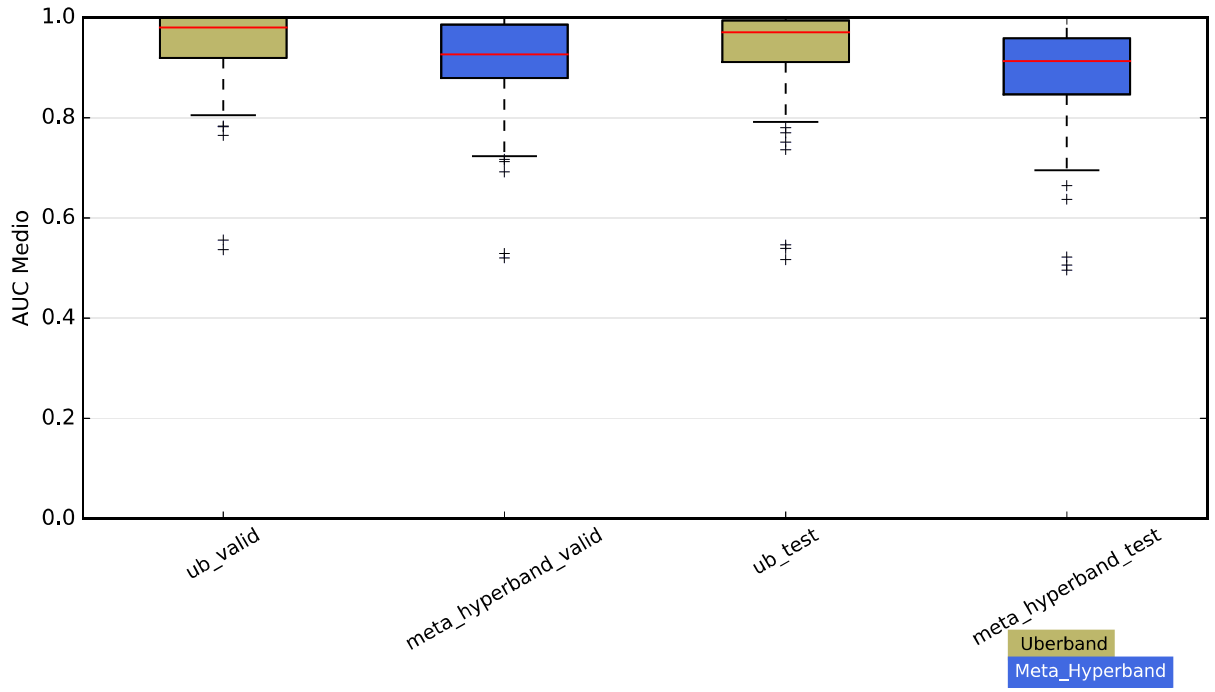

Figura 7.7 – Distribuição de Desempenho (em AUC) do Uberband comparado com Hyperband acrescido de Meta-Aprendizado para amostragem de subconjuntos de operadores.

7.2.5 Avaliação do Uberband Comparado ao Hyperband

Por fim, Uberband foi comparado ao Hyperband para avaliar o ganho de desempenho de ambos os componentes. A Figura 7.9 apresenta uma visão geral do Uberband quando comparado ao Hyperband, cujo pseudo-código é apresentado em Algoritmo B.1.

Na Figura 7.10 são apresentados os resultados da distribuição de desempenho das duas abordagens, na Figura 7.11 a comparação do desempenho e na Tabela 7.6 a média e desvio-padrão desse desempenho. Nestes resultados, Uberband, que combina meta-aprendizado com a estratégia de otimização que explora tanto o conjunto de operadores quanto o conjunto de hiper-parâmetros, se mostrou superior ao Hyperband para todos os conjuntos de dados.

Tabela 7.6 – Média e desvio-padrão do desempenho (em AUC) do Uberband comparado com Hyperband padrão

	Validação		Teste	
	Uberband	Hyperband	Uberband	Hyperband
Média	**0.936**	0.900	**0.921**	0.866
Desvio Padrão	**0.098**	0.112	**0.115**	0.127

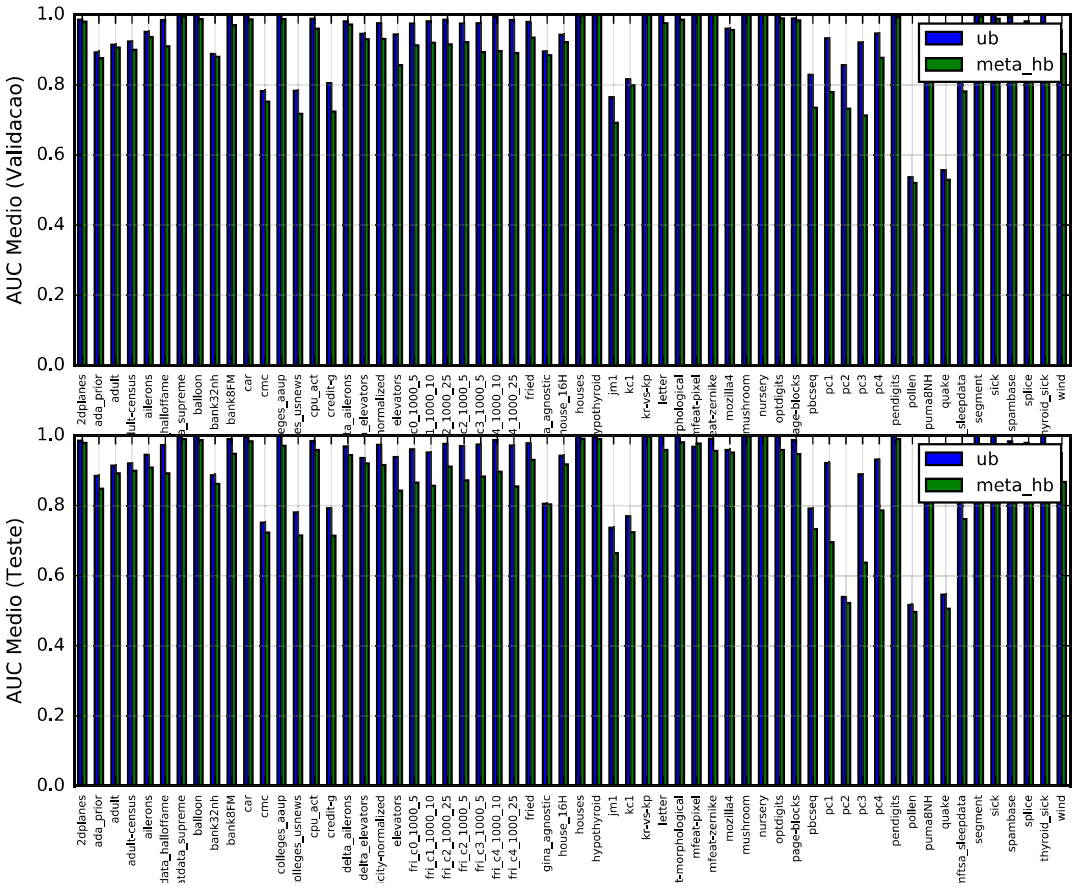

Figura 7.8 – Comparação de Desempenho (em AUC) do Uberband comparado com Hyperband acrescido de Meta-Aprendizado para amostragem de subconjuntos de operadores.

Na avaliação estatística com teste de Wilcoxon, obteve-se $z = -6.72$ tanto para o conjunto de validação e quanto para o conjunto de testes, já que em ambos os casos Uberband foi obteve desempenho superior em todos os conjuntos de dados avaliados, comprovando efetivamente a superioridade do Uberband quando comparado a versão padrão de Hyperband com foco apenas na otimização de hiper-parâmetros.

Avaliando os Subconjuntos de Operadores Selecionados por Uberband e Hyperband

Adicionalmente, na Tabela 7.7 podemos observar os melhores subconjuntos de operadores selecionados pelo Uberband e pelo Hyperband, os processos completos (que incluem os hiper-parâmetros) podem ser consultados no Apêndice C.

Uma vez que ambos os algoritmos possuem funções estocásticas e cada um foi executado 5 vezes para cada conjunto de dados, o melhor processo completo selecionado sofreu variações entre as execuções. Assim, na Tabela 7.7 são apresentados os subcon-

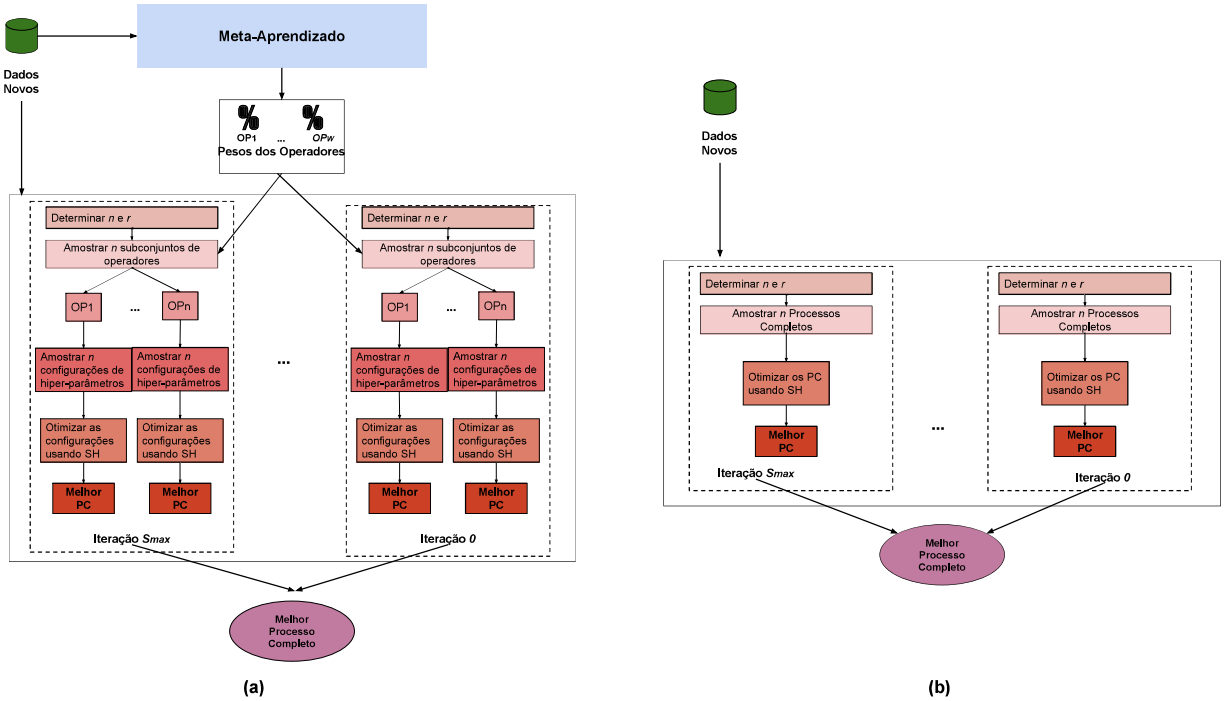

Figura 7.9 – Visão Geral do Uberband (a) e Hyperband (b)

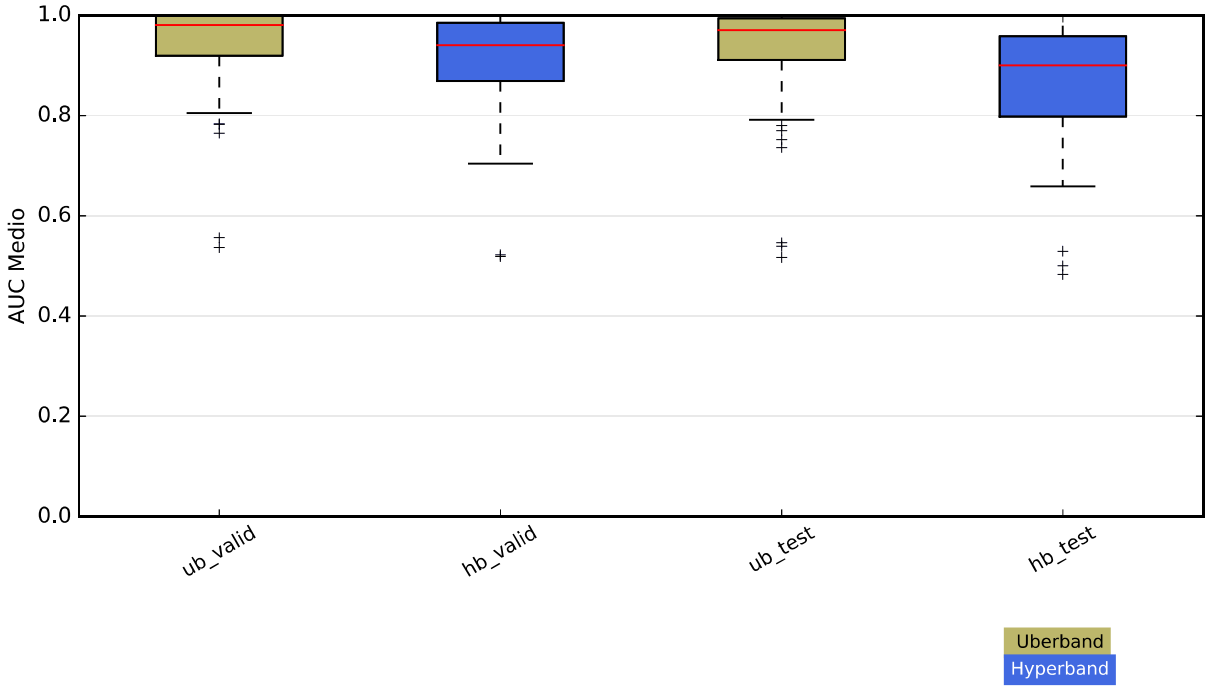

Figura 7.10 – Distribuição de Desempenho (em AUC) do Uberband comparado com Hyperband.

juntos de operadores que correspondem aos processos completos que obtiveram o maior AUC para cada conjunto de dados.

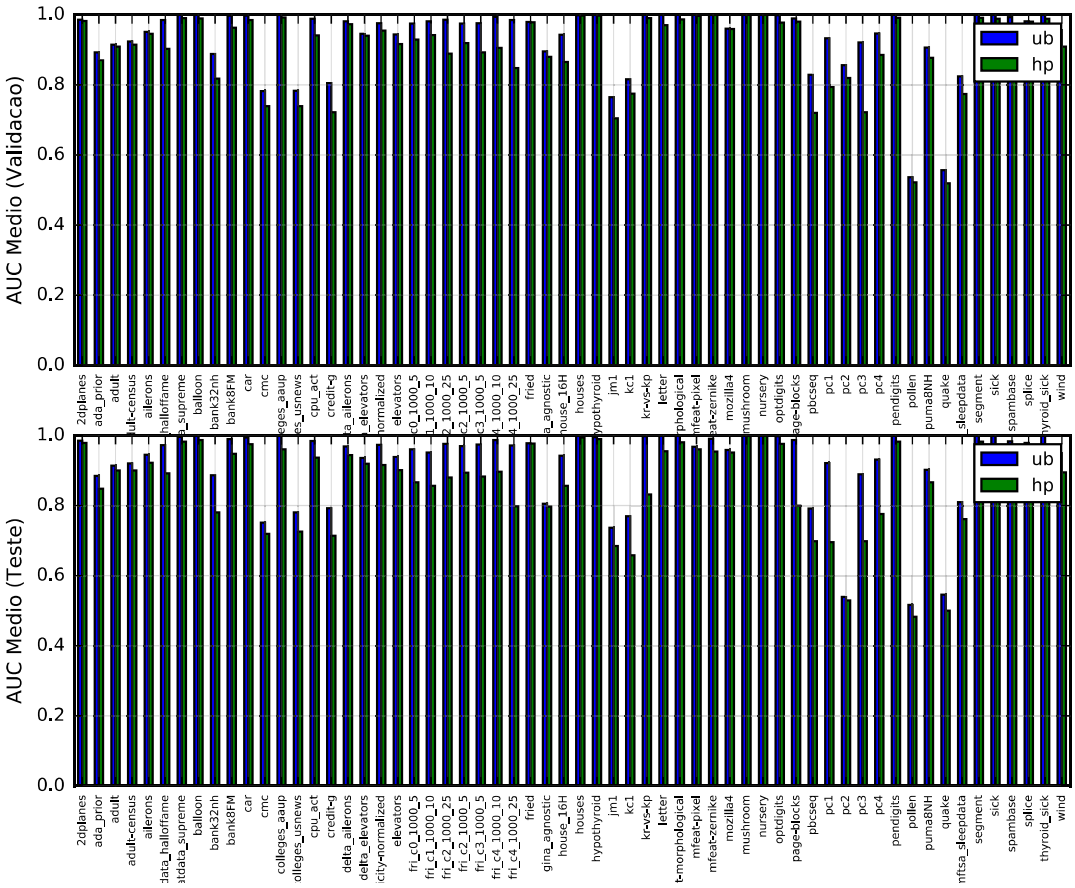

Figura 7.11 – Comparação de Desempenho (em AUC) do Uberband comparado com Hyperband.

De acordo com esses resultados, podemos observar que cerca de 90% dos melhores processos completos selecionados pelo Uberband não contém operadores de pré-processamento (Seleção de Características (SC)), este é um fator interessante, uma vez que todos os processos completos amostrados são experimentados no conjunto de treinamento (mantendo a proporção de instâncias de acordo com a alocação adaptativa), processos completos que não contém pré-preprocessamento tendem a ser mais rápidos do que seus correspondentes com esses operadores. Dado que os subconjuntos de operadores são amostrados de acordo com os pesos estabelecidos pelo Meta-Aprendizado, podemos ratificar a importância deste componente.

Em relação ao Hyperband, uma vez que os tanto os subconjuntos de operadores quanto os hiper-parâmetros são amostrados aleatoriamente, notamos uma maior diversidade nos processos completos com melhor desempenho.

Tabela 7.7 – Comparação entre os melhores subconjuntos de operadores selecionados por Uberband e Hyperband, incluindo as etapas de Seleção de Características (SC) e Classificação (CL)

Dados	Uberband	Hyperband
2dplanes	SC: Nenhum, CL: 'J48'	SC: 'OneRAttributeEval', CL: 'BayesNet'
ada_prior	SC: Nenhum, CL: 'NaiveBayes'	SC: 'RELIEF Eval', CL: 'REPTree'
adult	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: Nenhum, CL: 'RandomSubSpace'
adult-census	SC: Nenhum, CL: 'LMT'	SC: Nenhum, CL: 'RandomSubSpace'
ailerons	SC: Nenhum, CL: 'J48'	SC: 'Symmetrical Uncert', CL: 'SVM'
analcatdata-halloffame	SC: Nenhum, CL: 'SimpleLogistic'	SC: 'OneRAttributeEval', CL: 'NaiveBayes'
analcatdata-supreme	SC: Nenhum, CL: 'LogitBoost'	SC: 'Info Gain Eval', CL: 'JRip'
balloon	SC: Nenhum, CL: 'RandomSubSpace'	SC: 'Symmetrical Uncert', CL: 'ZeroR'
bank32nh	SC: Nenhum, CL: 'SimpleLogistic'	SC: 'CFSSubsetEval', CL: 'SVM'
bank8FM	SC: Nenhum, CL: 'SimpleLogistic'	SC: 'Gain Ration Eval', CL: 'Stacking'
car	SC: Nenhum, CL: 'RandomCommitee'	SC: 'RELIEF Eval', CL: 'Stacking'
cmc	SC: Nenhum, CL: 'AdaBoost M1'	SC: Nenhum, CL: 'RandomForest'
colleges_aaup	SC: Nenhum, CL: 'SimpleLogistic'	SC: 'Info Gain Eval', CL: 'RandomCommitee'
colleges_usnews	SC: Nenhum, CL: 'SimpleLogistic'	SC: Nenhum, CL: 'SimpleLogistic'
cpu_act	SC: Nenhum, CL: 'RandomForest'	SC: 'RELIEF Eval', CL: 'RandomSubSpace'
credit-g	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: 'RELIEF Eval', CL: 'REPTree'
delta_ailerons	SC: Nenhum, CL: 'RandomCommitee'	SC: 'Info Gain Eval', CL: 'MultilayerPerceptron'
delta_elevators	SC: Nenhum, CL: 'LMT'	SC: 'Wrapper Sub Eval', CL: 'MultilayerPerceptron'
electricity-normalized	SC: Nenhum, CL: 'RandomForest'	SC: 'Info Gain Eval', CL: 'RandomForest'
elevators	SC: Nenhum, CL: 'SVM'	SC: Nenhum, CL: 'RandomSubSpace'
fri_c0_1000_5	SC: Nenhum, CL: 'NaiveBayes'	SC: Nenhum, CL: 'RandomSubSpace'
fri_c1_1000_10	SC: Nenhum, CL: 'RandomCommitee'	SC: 'Symmetrical Uncert', CL: 'SVM'
fri_c2_1000_25	SC: Nenhum, CL: 'RandomForest'	SC: Nenhum, CL: 'RandomCommitee'
fri_c2_1000_5	SC: Nenhum, CL: 'RandomCommitee'	SC: 'Symmetrical Uncert', CL: 'MultilayerPerceptron'
fri_c3_1000_5	SC: Nenhum, CL: 'RandomCommitee'	SC: 'OneRAttributeEval', CL: 'Stacking'
fri_c4_1000_10	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: 'Info Gain Eval', CL: 'J48'
fri_c4_1000_25	SC: Nenhum, CL: 'RandomForest'	SC: Nenhum, CL: 'NaiveBayes'
fried	SC: Nenhum, CL: 'RandomForest'	SC: Nenhum, CL: 'RandomSubSpace'
gina_agnostic	SC: 'Symmetrical Uncert', CL: 'RandomSubSpace'	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'
house_16H	SC: Nenhum, CL: 'RandomSubSpace'	SC: 'OneRAttributeEval', CL: 'OneR'
houses	SC: Nenhum, CL: 'LogitBoost'	SC: 'OneRAttributeEval', CL: 'RandomSubSpace'
hypothyroid	SC: Nenhum, CL: 'LogitBoost'	SC: 'RELIEF Eval', CL: 'ZeroR'
jm1	SC: Nenhum, CL: 'RandomForest'	SC: Nenhum, CL: 'RandomForest'
kc1	SC: Nenhum, CL: 'RandomCommitee'	SC: 'Classifier Attribute Eval', CL: 'AdaBoost M1'

kr-vs-kp	SC: 'Gain Ration Eval', CL: 'Stacking'	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'
letter	SC: Nenhum, CL: 'RandomCommitee'	SC: Nenhum, CL: 'RandomCommitee'
mfeat-morphological	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: 'Wrapper Sub Eval', CL: 'Stacking'
mfeat-pixel	SC: 'Classifier Attribute Eval', CL: 'RandomTree'	SC: Nenhum, CL: 'RandomForest'
mfeat-zernike	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: Nenhum, CL: 'RandomSubSpace'
mozilla4	SC: Nenhum, CL: 'DecisionTable'	SC: 'OneRAttributeEval', CL: 'AdaBoost M1'
mushroom	SC: 'CFSSubsetEval', CL: 'OneRR'	SC: Nenhum, CL: 'IBk'
nursery	SC: 'CFSSubsetEval', CL: 'SGD'	SC: 'Gain Ration Eval', CL: 'Stacking'
optdigits	SC: Nenhum, CL: 'SimpleLogistic'	SC: 'Symmetrical Uncert', CL: 'Stacking'
page-blocks	SC: Nenhum, CL: 'RandomCommitee'	SC: 'Classifier Attribute Eval', CL: 'OneR'
pbcseq	SC: Nenhum, CL: 'LogitBoost'	SC: Nenhum, CL: 'SimpleLogistic'
pc1	SC: 'CFSSubsetEval', CL: 'SimpleLogistic'	SC: Nenhum, CL: 'AdaBoost M1'
pc2	SC: Nenhum, CL: 'BayesNet'	SC: 'OneRAttributeEval', CL: 'RandomTree'
pc3	SC: Nenhum, CL: 'RandomCommitee'	SC: 'Gain Ration Eval', CL: 'Stacking'
pc4	SC: Nenhum, CL: 'RandomSubSpace'	SC: 'CFSSubsetEval', CL: 'Stacking'
pendigits	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: Nenhum, CL: 'RandomCommitee'
pollen	SC: Nenhum, CL: 'REPTree'	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'
puma8NH	SC: Nenhum, CL: 'RandomForest'	SC: 'CFSSubsetEval', CL: 'Logistic'
quake	SC: Nenhum, CL: 'Logistic'	SC: 'Gain Ration Eval', CL: 'IBk'
rmftsa_sleepdata	SC: Nenhum, CL: 'RandomSubSpace'	SC: 'Gain Ration Eval', CL: 'RandomForest'
segment	SC: Nenhum, CL: 'SimpleLogistic'	SC: Nenhum, CL: 'NaiveBayes'
sick	SC: Nenhum, CL: 'LogitBoost'	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'
spambase	SC: Nenhum, CL: 'RandomForest'	SC: 'Symmetrical Uncert', CL: 'RandomTree'
splice	SC: Nenhum, CL: 'LMT'	SC: Nenhum, CL: 'JRip'
thyroid_sick	SC: Nenhum, CL: 'RandomCommitee'	SC: 'RELIEF Eval', CL: 'Vote'
wind	SC: Nenhum, CL: 'MultilayerPerceptron'	SC: Nenhum, CL: 'RandomSubSpace'

7.3 Análise Comparativa com Estado da Arte em Seleção de Processos Completos

Uma vez comprovada experimentalmente a qualidade dos componentes individuais do Uberband, a solução proposta é comparada a outras soluções de Seleção de Processo Completo, de maneira a avaliar se o principal objetivo proposto de otimizar o processo de avaliação de processos completos obteve êxito frente aos trabalhos já existentes.

7.3.1 Métodos de Comparação

Como métodos de comparação foram utilizados Auto-Weka [Thornton et al., 2013] e Auto-Sklearn [Feurer et al., 2015a].

Auto-Weka [Thornton et al., 2013] foi a primeira solução de AutoAM proposta para resolver o problema CASH (Combinação de Seleção de Algoritmos e Otimização de Hiper-Parâmetros), utilizando Otimização Bayesiana. Para estes experimentos foi utilizada a versão 2.0, que utiliza o otimizador *Sequencial Model-based Algorithm Configuration* (SMAC). O espaço de busca do Auto-Weka consiste em 1 método de seleção de características (combinando 3 métodos de busca) e diversas abordagens de classificação implementadas no WEKA, o que incluiu cerca de 30 algoritmos de classificação, além das configurações de hiper-parâmetros para cada classificador.

Auto-Sklearn, proposto por Feurer *et al.* [Feurer et al., 2015a] é um sistema de AutoAM baseado na biblioteca scikit-learn [Pedregosa et al., 2011]. Também utilizando o algoritmo de otimização Bayesiana SMAC, essa solução inclui uma estratégia de Meta-Aprendizado, que considera o desempenho passado em conjuntos de dados semelhantes para recomendar um subconjunto de processos completos para inicializar a fase de otimização. Além disso, esta solução gera comitês (*ensembles*) dos processos completos considerados pela Otimização Bayesiana. A versão atual do Auto-Sklearn inclui 15 algoritmos de classificação, 14 métodos de redução de dimensionalidade (incluindo seleção e extração de características) e 4 métodos de transformação de dados.

Como apresentado na Seção 3.3, métodos baseados em Otimização Bayesiana são considerados estado- da-arte para Configuração de Algoritmos, tendo sido apenas ultrapassados pelos métodos baseado em Bandidos Multi-Armados recentemente, com a proposição do Hyperband. Da mesma forma, conforme apresentado na Revisão Sistemática da Literatura, esses métodos têm se destacado pelas suas características de extensibilidade, já que não dependem de conhecimento externo, e por avaliarem diretamente os processos candidatos. Por outro lado, essa avaliação é custosa computacionalmente, pois cada processo completo candidato é treinado em todo o conjunto de treinamento (usando Validação Cruzada de 10-folds). Embora as soluções permitam que seja especificado um tempo máximo de treinamento (onde interrupções são realizadas em configurações que excedam um limiar de tempo), essa é uma decisão de implementação da solução, e não conceitual, de melhoria de custo computacional.

Assim, devido ao destaque dos métodos de Otimização Bayesiana e a disponibilidade das estratégias, essas foram as soluções escolhidas para comparação com o Uberband.

7.3.2 Método de Avaliação

Tendo como base o objetivo de otimizar o custo computacional da Seleção de Processo Completo, mantendo a qualidade da predição, a solução proposta neste trabalho, Uberband, utiliza alocação adaptativa de instâncias de treinamento na avaliação de configurações de processo completos, além de Meta-Aprendizado para predição do desempenho dos subconjuntos de operadores disponíveis. Com isso, esta avaliação tem como objetivo comparar o Uberband a soluções proeminentes na literatura para SPC de maneira a verificar se o objetivo foi atingido.

Nesta avaliação, as soluções são comparadas, levantando-se inicialmente as seguintes hipóteses principais:

- Hipótese Nula, H_0: Não há diferença significativa entre o tempo de execução de todas as soluções avaliadas.
- Hipótese Alternativa, H_1: Há diferença significativa entre o tempo de execução das soluções.

Caso a hipótese nula seja refutada em favor de H_1, é aplicado um pós-teste para comparação dos algoritmos par a par. Ao final desta avaliação, deseja-se saber se o tempo médio de execução do algoritmo Uberband, que utiliza a alocação de recursos adaptativa, é superior ao tempo médio de execução dos algoritmos que avaliam sempre no conjunto de treinamento completo.

Adicionalmente, levando-se em consideração que as soluções avaliadas reportam um desempenho satisfatório para as tarefas nas quais foram aplicadas, deseja-se avaliar se o processo completo recomendado pelo Uberband possui um desempenho no mínimo equivalente a essas soluções. Para isso, as seguintes hipóteses foram levantadas:

- Hipótese Nula, H_0: Não há diferença significativa entre o desempenho preditivo de todas as soluções avaliadas.
- Hipótese Alternativa, H_1: Há diferença significativa entre o desempenho preditivo das soluções.

Assim como na avaliação de tempo, uma vez que a hipótese nula seja refutada, a hipótese H_1 é avaliada com base em um pós-teste. Ao final desta avaliação, deseja-se saber se o desempenho preditivo do algoritmo do Uberband é igual ou superior ao desempenho preditivo das demais soluções.

Por se tratar de uma comparação de múltiplos algoritmos todos contra todos, foi utilizado o teste Friedman com pós-teste de Nemenyi, conforme procedimento descrito por Demŵsar [Demšar, 2006].

O desempenho de dois algoritmo é considerado significativamente diferente se a média dos *ranks* correspondentes diferem em pelo menos a Diferença Crítica.

7.3.3 Preparação Experimental

Na avaliação do Auto-Weka e Auto-Sklearn, os conjuntos de dados foram amostrados em treino (80%) e teste (20%), sendo o conjunto de testes aplicado ao melhor processo completo recomendado por cada solução. Para a avaliação do Uberband, o conjunto de treinamento foi adicionalmente dividido em treino (80%) e validação (20%). Para o Auto-Sklearn, seguindo a recomendação do artigo original [Feurer et al., 2015a], devido a limitações do scikit-learn, que é restrito a dados numéricos, os atributos categóricos de cada conjunto de dados foram transformados utilizando o procedimento de *One Hot Encoding*.

Uma vez que para o Auto-Weka e Auto-Sklearn deve-se definir o tempo máximo de processamento e o limite de memória a ser utilizada, seguindo as configurações experimentais dos artigos de referência, foi determinado para o experimento um limite de 3GB de mémoria RAM e um tempo máximo de 1 hora para a avaliação de cada algoritmo em cada conjunto de dados, e tempo máximo de 4 minutos para avaliação de cada configuração de processo completo.

Em relação aos hiper-parâmetros, Auto-Weka e Auto-Sklearn foram executados com as configurações padrões, enquanto Uberband foi parametrizado com $\eta = 3$, e R como sendo o tamanho do conjunto de treinamento.

Os experimentos utilizam Auto-Weka 2.0 e Auto-Sklearn 0.3.0. Para o Uberband, a versão subjacente do Weka foi Weka 3.9.

7.3.4 Comparando Uberband com Auto-Weka e Auto-Sklearn

Cada estratégia foi executada 5 vezes em todos os conjuntos de dados e o desempenho médio (medido em AUC) e tempo de execução médio (medido em segundos) são apresentados na Figura 7.12, na Tabela 7.8 são apresentados a média e o desvio padrão dessas execuções.

Tabela 7.8 – Média e desvio-padrão do desempenho (em AUC) e Tempo de Execução (em seg) do Uberband comparado com Auto-Sklearn e Auto-Weka

	Desempenho (em AUC)			Tempo de Execução (em segundos)		
	Uberband	Auto-Sklearn	Auto-Weka	Uberband	Auto-Sklearn	Auto-Weka
Média	0.9247	0.9289	0.8918	2035	3602	3271
Desvio	0.1130	0.1205	0.1381	324	10	464

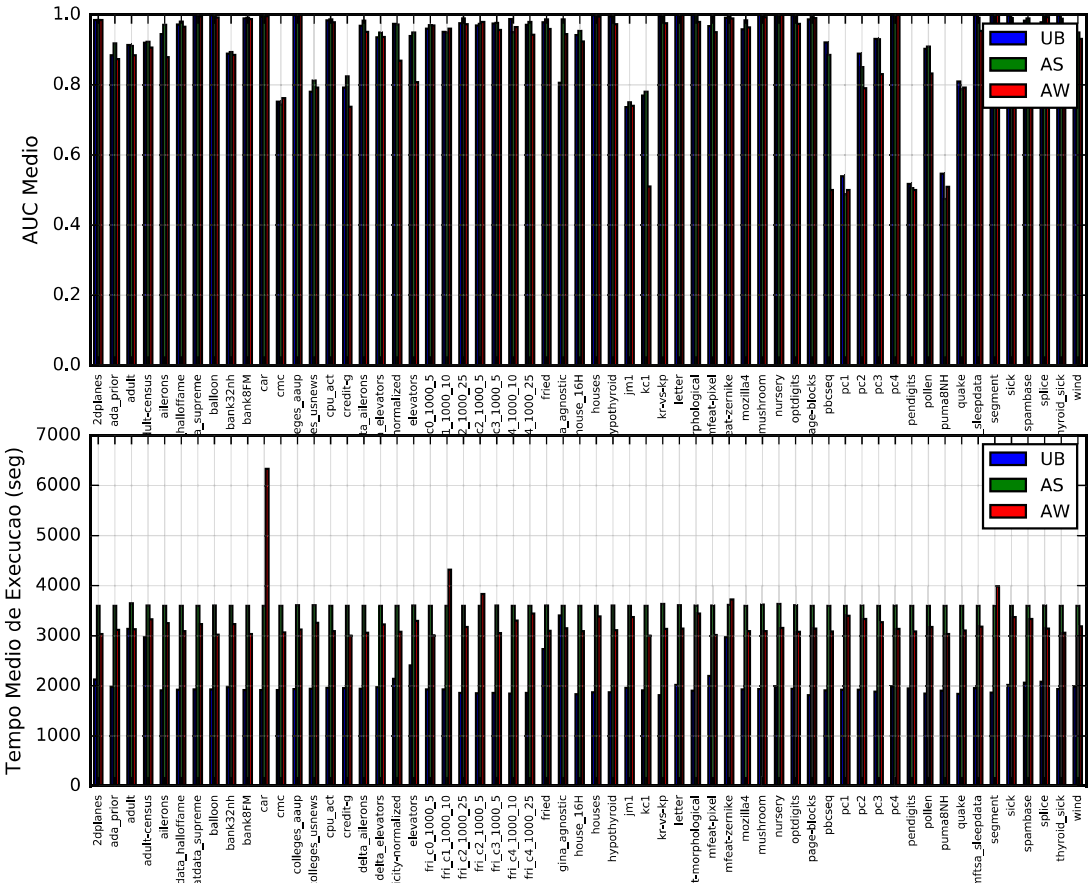

Figura 7.12 – Comparação de Desempenho (em AUC) e Tempo de Execução do Uberband com Auto-Sklearn e Auto-Weka.

A partir desses resultados podemos observar que o tempo médio de execução do Uberband é significativamente inferior aos demais para todos os conjuntos de dados, apenas se aproximando do tempo limite máximo para conjuntos de dados maiores como *adult*, *adult-census* e *gina-agnostic*. De maneira geral, a execução de configurações no Uberband, com uma sub-amostra do conjunto de dados, geralmente leva poucos segundos, dependendo dos hiper-parâmetros selecionados, conforme a amostra de dados aumenta este tempo também aumenta, mas sem ultrapassar o limite de 1 hora, já que o tempo máximo por configuração foi definido como 4 minutos e configurações que ultrapassam esse tempo são interrompidas. A Figura 7.13 apresenta a distribuição do tempo médio de execução por cada solução.

O teste estatístico de Friedman com pós-teste de Nemenyi foi aplicado para comprovar que estes resultados são significativos. Considerando o número de conjuntos de dados avaliados como $N = 60$, e $\alpha = 0.05$, os *ranks* médios foram os seguintes (com base no teste de Friedman): $R_{UB} = 1.033$, $R_{AW} = 2.050$ e $R_{AS} = 2.9167$, o que faz com a hipótese nula de igualdade entre todos eles seja rejeitada.

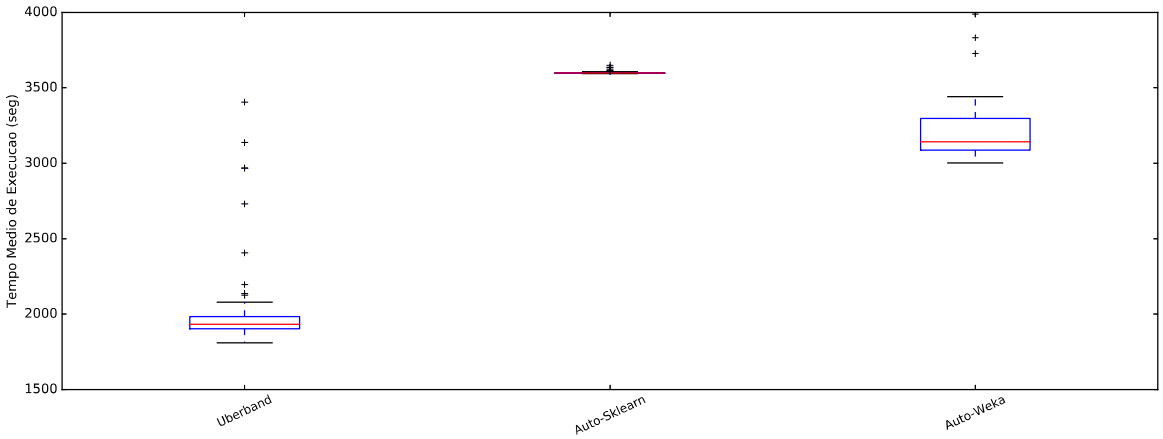

Figura 7.13 – Avaliação de Tempo de Execução (em Segundos) do Uberband comparado com Auto-Sklearn e Auto-Weka.

Assim, aplicando-se o pós-teste de Nemenyi, a diferença crítica (Equação 7.5) foi de $CD = 0.4278$. O que significa que deve haver pelo menos esta diferença entre os valores de *rank* médio.

A Figura 7.14 apresenta o resultado nesta análise, a ausência de uma linha horizontal indica que há diferença significativa entre os todos os pares de algoritmos.

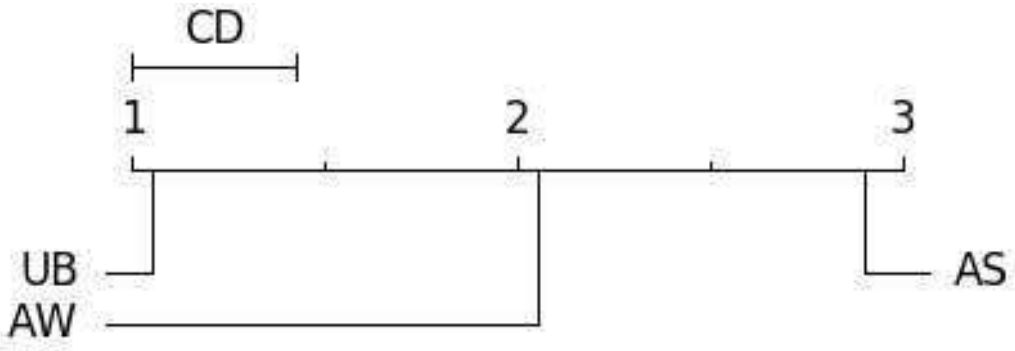

Figura 7.14 – Análise de Significância Estatística entre Uberband, Auto-Weka e Auto-Sklearn em Termos de Tempo de Execução

Como o *rank* foi baseado em tempo, menores valores são considerados melhores. Assim, pode-se comprovar que Uberband obteve o melhor desempenho em termos de tempo de execução, sendo estatisticamente superior aos demais algoritmos. Além disso, embora Auto-Weka e Auto-Sklearn tenham levado um tempo de execução médio similar (próximo ao limite estabelecido), os resultados da análise estatística indicam que o tempo do Auto-Weka é significativamente melhor que do Auto-Sklearn.

Uma vez comprovado que Uberband consegue melhorar a SPC em relação ao tempo de execução, deve-se comprovar se esta melhoria não afeta a qualidade do processo recomendado. De acordo coma Tabela 7.8 pode-se observar que o desempenho médio (em AUC) obtido pelo Uberband superior ao do Auto-Weka e, ligeiramente inferior ao do

Auto-Sklearn, um indicativo de que, além de ter melhorado o tempo de execução, ainda foi mantida a qualidade em termos de desempenho. A Figura 7.15 apresenta a distribuição do desempenho por cada solução.

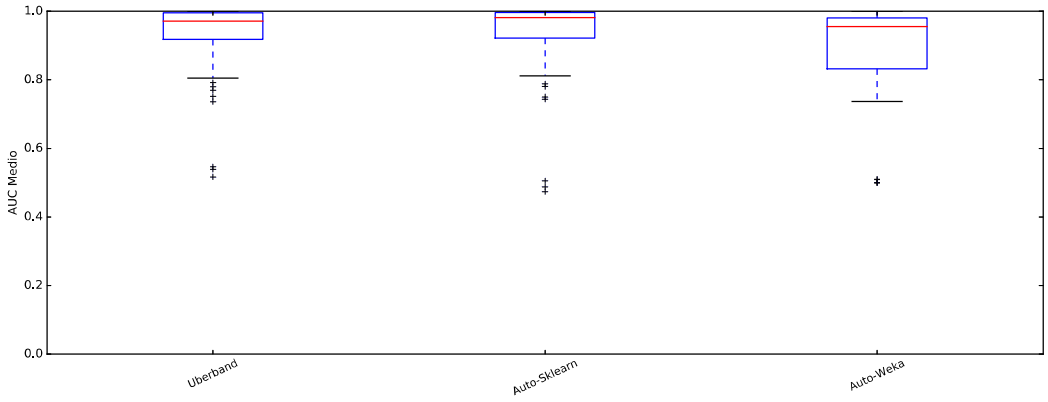

Figura 7.15 – Avaliação de Desempenho (em AUC) do Uberband comparado com Auto-Sklearn e Auto-Weka.

Para corroborar esta afirmação, foi aplicado novamente o teste de Friedman com pós-teste de Nemenyi. Os *ranks* médios obtidos pelo teste de Friedman foram: : R_{AW} = 1.4008, R_{UB} = 2.1167 e R_{AS} = 2.4750, o que faz com a hipótese nula de igualdade entre todos eles seja rejeitada.

Por ambos os testes utilizarem o mesmo número de conjuntos de dados, a diferença crítica obtida no teste de Nemenyi foi a mesma anterior, de CD = 0.4278. A Figura 7.16 apresenta o resultado nesta análise, a linha horizontal indica que não há diferença significativa entre os algoritmos.

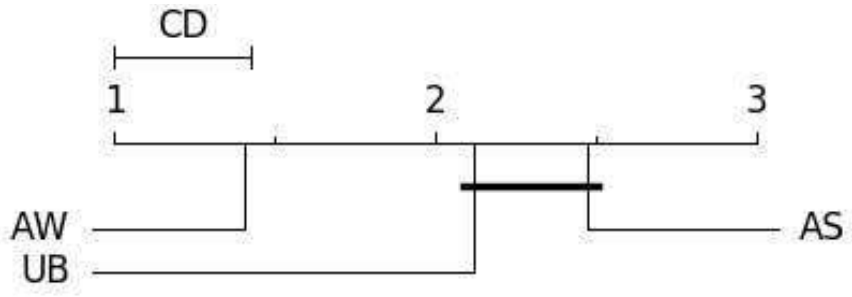

Figura 7.16 – Análise de Significância Estatística entre Uberband, Auto-Weka e Auto-Sklearn em Termos de Desempenho (AUC)

Como o *rank* foi baseado em AUC, maiores valores são considerados melhores. Assim, apesar o Auto-Sklearn ter obtido um valor médio de *rank* superior, os resultados indicam que não há diferença estatística entre esta solução e o Uberband. Os resultados também indicam que a diferença obtida entre Uberband e Auto-Weka é significante.

7.4 Considerações Finais do Capítulo

Neste Capítulo foi apresentada a avaliação experimental do Uberband. Inicialmente foram avaliados separadamente os componentes da solução, Meta-Aprendizado e Otimização, onde o algoritmo proposto foi comparado a versões sem o componente sob avaliação. Os resultados desta análise mostraram que ambos os componentes são fundamentais para a qualidade da recomendação proposta, tendo resultados estatisticamente superiores as versões sem eles.

Além disso, resultados comparativos com duas das abordagens mais relevantes da área de SPC, Auto-Sklearn e Auto-Weka, demonstraram que o objetivo principal desta Tese, de propor uma SPC eficiente e efetiva a partir da otimização do processo de avaliação de modelos candidatos, obteve êxito ao demonstrar tempo de execução significativamente inferior as abordagens comparadas, mas mantendo a qualidade da recomendação já oferecida.

8. CONSIDERAÇÕES FINAIS

Seleção de Processos Completos (SPC) objetiva auxiliar usuários, especialmente os não-especialistas, na tarefa onerosa de escolher o processo completo, que inclui métodos de pré- processamento, algoritmos de aprendizado de máquina e suas configurações de hiper-parâmetros, mais adequadas a um determinado problema. Embora diversas soluções já existam para esta tarefa, tais soluções são limitadas do ponto de vista da avaliação do processo completo: i) algumas soluções não realizam experimentação do processo e se baseiam em estimativas de desempenho, o que pode levar a recomendações não-precisas, ii) as demais soluções avaliam os processos completos repetidas vezes sobre o conjunto de treinamento completo até encontrar a melhor opção, estas soluções geralmente obtém resultados mais precisos, porém, se tornam computacionalmente custosas em termo de tempo, à medida em que os conjuntos de dados aumentam de tamanho, o que é uma tendência natural desta área em expansão.

Assim, o foco da pesquisa desenvolvida durante este doutorado foi propor e investigar um novo algoritmo para Seleção de Processos Completos , denominado Uberband. Este algoritmo faz uso de uma estratégia de meta-aprendizado para predição do desempenho de subconjuntos de operadores e alocação adaptativa de instâncias do conjunto de dados de treinamento durante seu processo de otimização, utilizando um método baseado em bandidos multi-armados, o que auxilia os usuários a partir da seleção de processos completos com bom desempenho e em um tempo relativamente menor do que as soluções atuais.

8.1 Contribuições

A principal contribuição desta tese é o algoritmo Uberband. Esse algoritmo, apresentado no Capítulo 5, faz uso de Meta- Aprendizado para predizer o desempenho dos subconjuntos de operadores disponíveis no espaço de busca e com isso apoiar a amostragem dos melhores subconjuntos por parte do otimizador, baseado em bandidos multi-armados. Este otimizador, por sua vez, utiliza a alocação adaptativa de instâncias do conjunto de dados de treinamento para minimizar os custos da avaliação de processos completos candidatos.

Além disso, como contribuições específicas, inicialmente foi realizada uma Revisão Sistemática da Literatura sobre Seleção de Processos Completos, apresentada no Capítulo 4, onde os trabalhos relacionados foram detalhados, organizados em uma taxonomia, discutidos em termos de seus componentes principais e analisados em termos de suas vantagens e desvantagens, levando em consideração os principais desafios da área.

A análise detalhada das soluções existentes permitiu também um amplo entendimento sobre a estrutura de uma solução de SPC básica. Com isso, foi possível projetar uma Arquitetura Geral que categoriza senão todas, mas boa parte das soluções existentes, tanto do ponto de vista técnico como de contribuições científicas. Esta arquitetura, denominada Arquitetura Genérica de Componentes de uma Soluções de Seleção de Processos Completos, foi apresentada no Capítulo 3. Além da possibilidade de caracterizar os trabalhos existentes, esta arquitetura tem potencial de auxiliar o desenvolvimento de trabalhos futuros na área.

Embora seja um componente da solução Uberband, a estratégia de Meta- Aprendizado para predição de desempenho dos subconjuntos de operadores pode ser valorada separadamente, uma vez que, como comprovado Capítulo 6, ela é capaz de recomendar *rankings* de subconjuntos de operadores para um novo problema com um bom desempenho.

Por fim, conforme visto na revisão das soluções existentes, não é comum que seja feita uma avaliação detalhada de desempenho sobre as soluções propostas. Em contrapartida, esta tese apresentou uma avaliação empírica detalhada dos componentes da solução proposta, onde foram desenvolvidas soluções alternativas do algoritmo principal, a fim de verificar a importância de cada componente individualmente. Além disso, foi apresentado um experimento em um grande volume de dados comparando a solução com o estado-da arte até então. Ambas as avaliações ainda contam com testes de significância estatística, para corroboração dos resultados. Esses experimentos são apresentados no Capítulo 7.

8.1.1 Publicações

Como publicações direta e indiretamente geradas como resultado desta pesquisa tem-se os trabalhos:

- das Dôres, Silvia N., et al. "A meta-learning framework for algorithm recommendation in software fault prediction."Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, 2016.
- das Dôres, Silvia N., et al. "Effect of Metalearning on Feature Selection Employment."Proceedings of Workshop AutoML@ECML/PAKDD. 2017.
- das Dôres, Silvia Nunes et al. “Bandit-Based Automated Machine Learning.” 2018 7th Brazilian Conference on Intelligent Systems (BRACIS) (2018): 121-126.
- (em produção) das Dôres, Silvia Nunes et al. "Automatic Workflow Selection: a Survey". ACM Computing Surveys (CSUR)

8.2 Limitações

Embora os resultados reportados nesta tese demonstrem que os principais objetivos da pesquisa foram cumpridos, pode-se destacar algumas limitações na proposta e nos experimentos conduzidos.

Nos experimentos reportados no Capítulo 7, no que diz respeito a avaliação dos componentes individuais do Uberband, especialmente na avaliação do componente de Meta-Aprendizado foi possível observar pontos onde a estratégia de "Uberband Sem Meta Aprendizado"conseguia obter melhores resultados parciais. Isso se deve ao fato de Meta-Aprendizado ter sido utilizado apenas para auxiliar na seleção do subconjunto de operadores, enquanto a seleção de hiper-parâmetros permanece aleatória. Neste caso, como os hiper-parâmetros podem mudar os valores de desempenho, em alguns pontos da otimização. Uma possível solução para isto seria utilizar uma estratégia diferente de aleatória para amostragem de hiper-parâmetros, o que será discutido na próxima seção.

Na configuração experimental, o espaço de busca avaliado pelo Uberband foi limitado a uma única fase de pré-processamento, a seleção de características. Embora seja possível ampliar este espaço para incluir novas etapas do processo de Descoberta de Conhecimento, sem grande interferência humana, isto ainda incorre na execução de diversos experimentos em termos de Meta-Aprendizado, para predizer o desempenho dos novos subconjuntos de operadores.

Por fim, a solução proposta está atualmente limitada a problemas de classificação, e incluir uma tarefa diferente não só incorre na alteração do espaço de busca, mas na estratégia de Meta-Aprendizado, uma vez que os meta-atributos, medidas de desempenho e algoritmos base são diferentes para outras tarefas, como regressão e agrupamento.

8.3 Trabalhos Futuros

Conforme apontado nas limitações, um grande desafio ainda em aberto no Uberband é utilizar uma estratégia diferente da amostragem aleatória para selecionar os hiper-parâmetros, nas configurações de processo completo. A partir dos experimentos realizados, pode-se observar que a utilização de meta-dados de problemas anteriores contribui para aumentar a qualidade da seleção do subconjuntos de operadores. Porém, uma vez que utilizar Meta-Aprendizado também para recomendar a configuração dos processos completos acarretaria uma explosão combinatorial na meta-base, esta não parece ser uma solução ideal otimizar a configuração. De outro modo, pode-se usar o conhecimento obtido durante o próprio processo de otimização como forma de melhorar essa escolha progressivamente.

Em termos experimentais, outro trabalho futuro seria ampliar o espaço de busca para incluir novas etapas do processo de Descoberta de Conhecimento, bem como novas medidas de desempenho passiveis de serem avaliadas (por exemplo, área sobre a curva precisão-revocação, acurácia balanceada) . Neste caso, o esforço seria a inclusão na meta-base de conhecimento, uma vez que é trivial alterar a medida avaliada pelo otimizador. Avaliando-se essas possibilidades de trabalhos futuros em termos experimentais, pode-se verificar que o maior impacto se dá na estratégia de Meta-Aprendizado, onde alterações no espaço de busca e forma de avaliação incorrem em uma nova gama de experimentos de serem realizados. Dessa forma, a utilização do repositório de Aprendizado de Máquina OpenML [Vanschoren et al., 2014] pode ser interessante, tanto para busca por resultados experimentais de execuções já realizadas sobre os conjuntos de dados e algoritmos que estão sendo utilizados no Uberband, quanto para armazenar os resultados de novos (e atuais) experimentos de nível base, de maneira a facilitar a reprodutibilidade da solução proposta e contribuir com a comunidade na disponibilização de um grande volume de resultados experimentais, que podem ser utilizados para os mesmos fins ou para novas aplicações.

REFERÊNCIAS BIBLIOGRÁFICAS

Althoff, K.-D. (2001). Case-based reasoning. In: *Handbook of Software Engineering and Knowledge Engineering: Fundamentals*, vol. 1, pp. 38. World Scientific, 1 ed..

Bansal, B. e Sahoo, A. (2015b). Full model selection using bat algorithm. In: *Proceedings of the 1st International Conference on Cognitive Computing and Information Processing*, pp. 4, Bangalore, IN. IEEE.

Bansal, B. e Sahoo, A. (Mar, 2015a). Bat algorithm for full model selection in classification: A multi-objective approach. *International Journal of Innovations & Advancement in Computer Science*, vol. 4, pp. 376–384.

Bellman, R. (Jul, 1966). Dynamic programming. *Science*, vol. 153, pp. 34–37.

Bensusan, H. (1998). God doesn't always shave with occam's razor - learning when and how to prune. In: *Proceedings of the 10th European Conference on Machine Learning*, pp. 6, Chemnitz, DE. Springer-Verlag.

Bensusan, H. e Giraud-Carrier, C. (2000). Casa batló is in passeig de gràcia or how landmark performances can describe tasks. In: *Proceedings of the 1st Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pp. 19, Barcelona, ES. Springer.

Bergstra, J. e Bengio, Y. (Fev, 2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, vol. 13, pp. 281–305.

Bergstra, J. S., Bardenet, R., Bengio, Y. e Kégl, B. (2011). Algorithms for hyper-parameter optimization. In: *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, pp. 10, Granada, ES. Curran Associates Inc.

Bernstein, A. e Dänzer, M. (2007). The next system: Towards true dynamic adaptations of semantic web service compositions. In: *Proceedings of the 4th European Semantic Web Conference*, pp. 10, Innsbruck, AT. Springer-Verlag.

Bernstein, A., Provost, F. e Hill, S. (Abr, 2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 503–518.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K. e Wiswedel, B. (Jan, 2009). Knime-the konstanz information miner: version 2.0 and beyond. *ACM Explorations Newsletter*, vol. 11, pp. 26–31.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K. et al. (Ago, 2016). Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, vol. 237, pp. 41–58.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, DE.

Boussaïd, I., Lepagnot, J. e Siarry, P. (Jul, 2013). A survey on optimization metaheuristics. *Information Sciences*, vol. 237, pp. 82–117.

Brazdil, P., Gama, J. e Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In: *Proceedings of the 7th European Conference on Machine Learning*, pp. 20, Catania, IT. Springer-Verlag.

Brazdil, P. e Giraud-Carrier, C. (Jan, 2018). Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue. *Machine Learning*, vol. 107, pp. 1–14.

Brazdil, P., Giraud-Carrier, C., Soares, C. e Vilalta, R. (2008). *Metalearning: Applications to Data Mining*. Springer Publishing Company, Berlin, DE.

Breiman, L. (Fev, 1996). Bagging predictors. *Machine Learning*, vol. 24, pp. 123–140.

Breiman, L. (Jan, 2001). Random forests. *Machine Learning*, vol. 45, pp. 5–32.

Brochu, E., Cora, V. M. e De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Recuperado de https://arxiv.org/abs/1012.2599. Janeiro 2019.

Charest, M. e Delisle, S. (2006). Ontology-guided intelligent data mining assistance: Combining declarative and procedural knowledge. In: *Proceedings of the 3rd Artificial Intelligence and Soft Computing*, pp. 6, Palma de Mallorca, ES. IASTED/ACTA Press.

Charest, M., Delisle, S., Cervantes, O. e Shen, Y. (2006). Intelligent data mining assistance via cbr and ontologies. In: *Proceedings of the 17th International Workshop on Database and Expert Systems Applications*, pp. 5, Krakow, PL. IEEE.

Charest, M., Delisle, S., Cervantes, O. e Shen, Y. (Fev, 2008). Bridging the gap between data mining and decision support: A case-based reasoning and ontology approach. *Intelligent Data Analysis*, vol. 12, pp. 211–236.

Chong, C. S., Zhang, T., Lee, K. K., Hung, G. G., Lee, B.-S. et al. (2013). Collaborative analytics with genetic programming for workflow recommendation. In: *Proceedings of the 15th International Conference on Systems, Man, and Cybernetics*, pp. 13, Washington, USA. IEEE.

Dahl, G. E., Sainath, T. N. e Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In: *Proceedings of the 29th International Conference on Acoustics, Speech and Signal Processing*, pp. 5, Vancouver, CA. IEEE.

De la Rosa, T., Olaya, A. G. e Borrajo, D. (2007). Using cases utility for heuristic planning improvement. In: *Proceedings of the 7th International Conference on Case-Based Reasoning*, pp. 12, Belfast, IE. Springer-Verlag.

de Sá, A. G., Pinto, W. J. G., Oliveira, L. O. V. e Pappa, G. L. (2017). Recipe: a grammar-based framework for automatically evolving classification pipelines. In: *Proceedings of the 20th European Conference on Genetic Programming*, pp. 17, Amsterdam, NL. Springer-Verlag.

Deb, K., Pratap, A., Agarwal, S. e Meyarivan, T. (Fev, 2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197.

Demšar, J. (Jan, 2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, vol. 7, pp. 1–30.

Diamantini, C., Potena, D. e Storti, E. (2009). Ontology-driven kdd process composition. In: *Proceedings of the 8th International Symposium on Intelligent Data Analysis*, pp. 11, Lyon, FR. Springer-Verlag.

Elsken, T., Metzen, J. H. e Hutter, F. (2018). Neural architecture search: A survey. Recuperado de https://arxiv.org/abs/1808.05377. Janeiro 2019.

Escalante, H. J., Montes, M. e Sucar, L. E. (Fev, 2009). Particle swarm model selection. *Journal of Machine Learning Research*, vol. 10, pp. 405–440.

Faceli, K., Lorena, A. C., Gama, J. e Carvalho, A. (2011). *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. LTC, Rio de Janeiro, BR.

Fayyad, U., Piatetsky-Shapiro, G. e Smyth, P. (Mar, 1996a). From data mining to knowledge discovery in databases. *AI magazine*, vol. 17, pp. 37–54.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. et al. (1996b). Knowledge discovery and data mining: towards a unifying framework. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 7, Portland, USA. AAAI Press.

Fernández, S., De La Rosa, T., Fernández, F., Suárez, R., Ortiz, J., Borrajo, D. e Manzano, D. (Fev, 2009). Using automated planning for improving data mining processes. *The Knowledge Engineering Review*, vol. 28, pp. 157–173.

Fernández, S., Súarez, R., De La Rosa, T., Ortiz, J., Fernández, F., Borrajo, D. e Manzano, D. (2010). Improving the execution of kdd workflows generated by ai planners. In: *Proceedings of the 4th Planning to Learn Workshop*, pp. 6, Lisboa, PT. IOS Press.

Ferri, C., Hernández-Orallo, J. e Modroiu, R. (Jan, 2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, vol. 30, pp. 27–38.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. e Hutter, F. (2015a). Efficient and robust automated machine learning. In: *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pp. 8, Quebec, CA. MIT Press.

Feurer, M., Springenberg, J. T. e Hutter, F. (2015b). Initializing bayesian hyperparameter optimization via meta-learning. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 8, Austin, USA. AAAI Press.

Gama, J. e Brazdil, P. (1995). Characterization of classification algorithms. In: *Proceedings of the 7th Portuguese Conference on Artificial Intelligence*, pp. 11, Funchal, PT. Springer.

Gendreau, M. e Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. Springer US, New York, USA.

Goldberg, D. E. e Holland, J. H. (Fev, 1988). Genetic algorithms and machine learning. *Machine Learning*, vol. 3, pp. 95–99.

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., Macia, N., Ray, B., Saeed, M., Statnikov, A. et al. (2015). Design of the 2015 chalearn automl challenge. In: *Proceedings of the 12th International Joint Conference on Neural Networks*, pp. 8, Killarney, IE. IEEE.

Han, J., Kamber, M. e Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, USA.

Hilario, M., Kalousis, A., Nguyen, P. e Woznica, A. (2009). A data mining ontology for algorithm selection and meta-mining. In: *Proceedings of the 2nd Workshop on Third Generation Data Mining*, pp. 11, Bled, SI. Springer-Verlag.

Hilario, M., Nguyen, P., Do, H., Woznica, A. e Kalousis, A. (2011). Ontology-based meta-mining of knowledge discovery workflows. In: *Meta-Learning in Computational Intelligence*, vol. 358, pp. 44. Springer-Verlag, 1 ed..

Hoffmann, J. e Nebel, B. (Jan, 2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, Cambridge, USA.

Hutter, F., Hoos, H. H. e Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In: *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, pp. 17, Rome, IT. Springer.

Hutter, F., Hoos, H. H., Leyton-Brown, K. e Stützle, T. (Set, 2009). Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306.

Hutter, F., Kotthoff, L. e Vanschoren, J., editores (2019). *Automatic Machine Learning: Methods, Systems, Challenges*. Springer International Publishing, New York, USA.

Jamieson, K. e Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 8, Cadiz, ES. JMLR.org.

Jones, D. R., Schonlau, M. e Welch, W. J. (Abr, 1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, vol. 13, pp. 455–492.

Kadioglu, S., Malitsky, Y., Sellmann, M. e Tierney, K. (2010). Isac-instance-specific algorithm configuration. In: *Proceedings of the 19th European Conference on Artificial Intelligence*, pp. 6, Amsterdam, NL. IOS Press.

Kalousis, A. (2002). *Algorithm Selection via Meta-Learning*. Tese de Doutorado, Université de Genève, Kallipefki, GR.

Kandasamy, K., Dasarathy, G., Schneider, J. e Poczos, B. (2017). Multi-fidelity bayesian optimisation with continuous approximations. In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 10, Sydney, AU. JMLR.org.

Kennedy, J. (2011). Particle swarm optimization. In: *Encyclopedia of Machine Learning*, vol. 1, pp. 13. Springer-Verlag, 1 ed..

Kietz, J., Serban, F., Bernstein, A. e Fischer, S. (2009). Towards cooperative planning of data mining workflows. In: *Proceedings of the 20th European Conference on Machine Learning*, pp. 12, Bled, SI. Springer-Verlag.

Kietz, J.-U., Serban, F., Bernstein, A. e Brazdil, P. (2010a). eproplan: A tool to model automatic generation of data mining workflows. In: *Proceedings of the 4th Planning to Learn Workshop*, pp. 6, Lisboa, PT. IOS Press.

Kietz, J.-U., Serban, F., Bernstein, A. e Fischer, S. (2010b). Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In: *Proceedings of the 21st European Conference on Machine Learning*, pp. 12, Barcelona, ES. Springer-Verlag.

Kietz, J.-U., Serban, F., Bernstein, A. e Fischer, S. (2012). Designing kdd-workflows via htn-planning. In: *Proceedings of the 20th European Conference on Artificial Intelligence*, pp. 2, Montpellier, FR. IOS Press.

Kietz, J.-U., Serban, F., Fischer, S. e Bernstein, A. (2014). "semantics inside!" but let's not tell the data miners: Intelligent support for data mining. In: *Proceedings of the 11th European Semantic Web Conference*, pp. 16, Anissaras, GR. Springer-Verlag.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Relatório Técnico, Keele University.

Kitchenham, B. e Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Relatório Técnico, Keele University and Durham University Joint Report.

Kohavi, R. e John, G. H. (1995). Automatic parameter selection by minimizing estimated error. In: *Proceedings of the 12th International Conference on Machine Learning*, pp. 9, Tahoe City, USA. Morgan Kaufmann Publishers Inc.

Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

Komer, B., Bergstra, J. e Eliasmith, C. (2014). Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In: *Proceedings of the 1st Workshop on Automatic Machine Learning*, pp. 7, Beijing, CN. JMLR.org.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F. e Leyton-Brown, K. (Jan, 2016). Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, vol. 17, pp. 1–5.

Koza, J. R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT press, Cambridge, USA.

Krizhevsky, A., Sutskever, I. e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, pp. 9, Lake Tahoe, USA. Curran Associates Inc.

Lattimore, T. e Szepesvári, C. (2018). Bandit algorithms. Recuperado de https://tor-lattimore.com/downloads/book/book.pdf. Janeiro 2019.

Lavrac, N. (2008). Planning to learn with a knowledge discovery ontology. In: *Proceedings of the 2nd Planning to Learn Workshop*, pp. 6, Helsinki, FI. ACM.

LeCun, Y., Bengio, Y. e Hinton, G. (Mai, 2015). Deep learning. *Nature*, vol. 521, pp. 436–444.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. e Talwalkar, A. (Jan, 2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, vol. 18, pp. 6765–6816.

Lindauer, M., Hoos, H. H., Hutter, F. e Schaub, T. (Mai, 2015). Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, vol. 53, pp. 745–778.

Lindner, G. e Studer, R. (1999). Ast: Support for algorithm selection with a cbr approach. In: *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 6, Prague, CZ. Springer-Verlag.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J. e Murphy, K. (2018). Progressive neural architecture search. In: *Proceedings of the 15th European Conference on Computer Vision*, pp. 17, Munich, DE. Springer-Verlag.

Ma, J. (2012). *Parameter Tuning Using Gaussian Processes*. Tese de Doutorado, University of Waikato, Hamilton, NZ.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D. e Wilkins, D. (1998). Pddl-the planning domain definition language. Relatório Técnico, Yale Center for Computational Vision and Control.

McGuinness, D. L., Van Harmelen, F. et al. (Fev, 2004). Owl web ontology language overview. *W3C Recommendation*, vol. 10, pp. 1–22.

Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J. e Neumann, F. (Fev, 2013). A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, vol. 69, pp. 151–182.

Michie, D., Spiegelhalter, D. J., Taylor, C. C. e Campbell, J. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Upper Saddle River, USA.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, USA.

Montgomery, D. C. (2017). *Design and Analysis of Experiments*. John Wiley & Sons, New Jersey, USA.

Murata, T. (Abr, 1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, vol. 77, pp. 541–580.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, USA.

Nau, D. S., Smith, S. J., Erol, K. et al. (1998). Control strategies in htn planning: Theory versus practice. In: *Proceedings of the 15th AAAI Conference on Artificial Intelligence*, pp. 7, Madison, USA. The MIT Press.

Nguyen, P., Hilario, M. e Kalousis, A. (Set, 2014). Using meta-mining to support data mining workflow planning and optimization. *Journal of Artificial Intelligence Research*, vol. 51, pp. 605–644.

Nguyen, P., Kalousis, A. e Hilario, M. (2011). A meta-mining infrastructure to support kd workflow optimization. In: *Proceedings of the 22nd European Conference on Machine Learning*, pp. 10, Atenas, GR. Springer-Verlag.

Nguyen, P., Kalousis, A. e Hilario, M. (2012a). Experimental evaluation of the e-lico meta-miner. In: *Proceedings of the 5th Planning to Learn Workshop*, pp. 2, Montpellier, FR. CEUR-WS.org.

Nguyen, P., Wang, J., Hilario, M. e Kalousis, A. (2012b). Learning heterogeneous similarity measures for hybrid-recommendations in meta-mining. In: *Proceedings of the 12th International Conference on Data Mining*, pp. 6, Brussels, BE. IEEE.

Olson, R. S. e Moore, J. H. (2016). Tpot: A tree-based pipeline optimization tool for automating machine learning. In: *Proceedings of the 3rd Workshop on Automatic Machine Learning*, pp. 9, New York, USA. JMLR.org.

Parmezan, A. R. S., Lee, H. D. e Wu, F. C. (Jun, 2017). Metalearning for choosing feature selection algorithms in data mining: Proposal of a new framework. *Expert Systems with Applications*, vol. 75, pp. 1–24.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. e Duchesnay, E. (Fev, 2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.

Peng, Y., Flach, P. A., Soares, C. e Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In: *Proceedings of the 5th International Conference on Discovery Science*, pp. 12, Lubeck, DE. Springer.

Pérez-Castro, N., Acosta-Mesa, H. G., Mezura-Montes, E. e Escalante, H. J. (2016). Multi-objective full model selection in temporal databases: Optimizing time and performance. In: *Proceedings of the 4th IEEE International Autumn Meeting on Power, Electronics and Computing*, pp. 6, Ixtapa, MX. IEEE.

Pfahinger, B., Hilan, B. e Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In: *Proceedings of the 17th International Conference on Machine Learning*, pp. 8, San Francisco, USA. Morgan Kaufmann Publishers Inc.

Pinto da Costa, J. e Soares, C. (Abr, 2005). A weighted rank measure of correlation. *Australian & New Zealand Journal of Statistics*, vol. 47, pp. 515–529.

Podpečan, V., Zemenova, M. e Lavrač, N. (Jan, 2011). Orange4ws environment for service-oriented data mining. *The Computer Journal*, vol. 55, pp. 82–98.

Post, M. J., van der Putten, P. e van Rijn, J. N. (2016). Does feature selection improve classification? a large scale experiment in openml. In: *Proceedings of the 15th International Symposium on Intelligent Data Analysis*, pp. 12, Stockholm, SE. Springer-Verlag.

RapidMiner (2018). Rapidminer studio. Recuperado de https://rapidminer.com/. Janeiro 2019.

Rice, J. R. (Jan, 1976). The algorithm selection problem. *Advances in computers*, vol. 15, pp. 65–118.

Roberts, M., Howe, A. e Flom, L. (2007). Learned models of performance for many planners. In: *Proceedings of the 1st Workshop AI Planning and Learning*, pp. 4, Providence, USA. AAAI.org.

Rokach, L. (Jan-Fev, 2010). Ensemble-based classifiers. *Artificial Intelligence Review*, vol. 33, pp. 1–39.

Rossi, A. (2013). *Meta-Aprendizado Aplicado à Fluxo Contínuo de Dados*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação, ICMC-USP, São Carlos, BR.

Sagi, O. e Rokach, L. (Abr, 2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, pp. 1–18.

Salvador, M. M., Budka, M. e Gabrys, B. (2016a). Adapting multicomponent predictive systems using hybrid adaptation strategies with auto-weka in process industry. In: *Proceedings of the 3rd Workshop on Automatic Machine Learning*, pp. 9, New York, USA. JMLR.org.

Salvador, M. M., Budka, M. e Gabrys, B. (2016b). Automatic composition and optimisation of multicomponent predictive systems. Recuperado de https://arxiv.org/abs/1612.08789. Janeiro 2019.

Serban, F. (2010). Auto-experimentation of kdd workflows based on ontological planning. In: *Proceedings of the 9th International Semantic Web Conference*, pp. 8, Shanghai, CN. Springer-Verlag.

Serban, F., Vanschoren, J., Kietz, J.-U. e Bernstein, A. (Jul, 2013). A survey of intelligent assistants for data analysis. *ACM Computing Surveys*, vol. 45, pp. 1–31.

Silverthorn, B. e Miikkulainen, R. (2010). Latent class models for algorithm portfolio methods. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 6, Atlanta, USA. AAAI Press.

Simon, H. A. (1981). *The Sciences of the Artificial*. MIT Press, Cambridge, USA.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. e Katz, Y. (Fev, 2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, pp. 51–53.

Smith-Miles, K. A. (Jan, 2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, vol. 41, pp. 6.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M. e Adams, R. (2015). Scalable bayesian optimization using deep neural networks. In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 9, Lille, FR. JMLR.org.

Soares, C. (2004). *Learning Rankings of Learning Algorithms: Recomendation of Algorithms with Meta-Learning*. Tese de Doutorado, Universidade do Porto, Porto, PT.

Soares, C. e Brazdil, P. B. (2000). Zoomed ranking: Selection of classification algorithms based on relevant performance information. In: *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 9, Lyon, FR. Springer.

Sohn, S. Y. (Nov, 1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 1137–1144.

Souza, B. F. d. (2010). *Meta-Aprendizagem Aplicada a Classificação de Dados de Expressão Gênica*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação, ICMC-USP, São Carlos, BR.

Sparks, E. R., Talwalkar, A., Haas, D., Franklin, M. J., Jordan, M. I. e Kraska, T. (2015). Automating model search for large scale machine learning. In: *Proceedings of the 6th ACM Symposium on Cloud Computing*, pp. 12, Kohala, USA. ACM.

Streeter, M., Golovin, D. e Smith, S. F. (2007). Combining multiple heuristics online. In: *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pp. 7, Vancouver, CN. AAAI Press.

Sun, Q. (2014). *Meta-Learning and the Full Model Selection Problem*. Tese de Doutorado, University of Waikato, Waikato, NZ.

Sun, Q., Pfahringer, B. e Mayo, M. (2013). Towards a framework for designing full model selection and optimization systems. In: *Proceedings of the 11th International Workshop on Multiple Classifier Systems*, pp. 12, Nanjing, CN. Springer.

Tan, P.-N., Steinbach, M. e Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA.

Thompson, W. R. (Mar-Abr, 1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, vol. 25, pp. 285–294.

Thornton, C., Hutter, F., Hoos, H. H. e Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, pp. 9, Chicago, USA. ACM.

Utgoff, P. E. (1984). *Shift of Bias for Inductive Concept Learning*. Tese de Doutorado, Rutgers University, New Brunswick, USA.

Vanschoren, J., Van Rijn, J. N., Bischl, B. e Torgo, L. (Fev, 2014). Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, vol. 15, pp. 49–60.

Vilalta, R. e Drissi, Y. (Fev, 2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, vol. 18, pp. 77–95.

Vilalta, R., Giraud-Carrier, C. e Brazdil, P. (2010). *Meta-Learning - Concepts and Techniques*, cap. 1, pp. 1–17. Springer US, Boston, USA.

Wilcoxon, F. (Jun, 1945). Individual comparisons by ranking methods. *Biometrics bulletin*, vol. 1, pp. 80–83.

Witten, I. H. e Frank, E. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, pp. 10, London, UK. ACM.

Yang, Q. e Wu, X. (Abr, 2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, vol. 5, pp. 597–604.

Yang, X.-S. (2010). *A new metaheuristic bat-inspired algorithm*, cap. 6, pp. 65–74. Springer-Verlag, Berlin, DE.

Záková, M., Kremen, P., Zelezný, F. e Lavrac, N. (2008). Using ontological reasoning and planning for data mining workflow composition. In: *Proceedings of the 1st Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*, pp. 8, Antwerp, BE. Springer-Verlag.

Záková, M., Kremen, P., Zelezny, F. e Lavrac, N. (Fev, 2011). Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering*, vol. 8, pp. 253–264.

Záková, M., Podpecan, V., Zeleznỳ, F. e Lavrac, N. (2009). Advancing data mining workflow construction: A framework and cases using the orange toolkit. In: *Proceedings of the 2nd Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*, pp. 15, Bled, SI. CEUR-WS.org.

Zoph, B. e Le, Q. V. (2016). Neural architecture search with reinforcement learning. Recuperado de https://arxiv.org/abs/1611.01578. Janeiro 2019.

APÊNDICE A – PSEUDO-CÓDIGO DA VERSÃO DO UBERBAND SEM META-APRENDIZADO

R, η
$n_{max} = \max\{9, R/1000\}$
$s_{max} = \lfloor \log_{\eta}(n_{max}) \rfloor$
$B = (s_{max} + 1)R$
for $s \in \{s_{max}, ..., 0\}$ **do**
  $n = \left\lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \right\rceil$, $r = R\eta^{-s}$
  $O = obter\_operadores(n)$
  for $j \in \{1, ..., O\}$ **do**
    $H = obter\_hiper\_parametros(n, j)$
    for $i \in \{0, ..., s\}$ **do**
      $n_i = \lceil n\eta^{-i} \rceil$
      $r_i = r\eta^i$
      $L = \{avaliacao\_de\_desempenho(h, r_i) : h \in H\}$
      $H = top\_k(H, L, \left\lfloor \frac{n_i}{\eta} \right\rfloor)$
    end for
  end for
end for
return melhor processo completo

Algoritmo A.1 – Algoritmo Uberband sem Meta-Aprendizado

APÊNDICE B – PSEUDO-CÓDIGO DA VERSÃO DO HYPERBAND COM META-APRENDIZADO

entrada : novo conjunto de dados D^{P+1}, modelos de predição de desempenho $\varphi = \{\varphi_1, ..., \varphi_W\}$ para cada modelo completo $j = 1, ..., W$, R, η
Calcular os meta-atributos para o novo conjunto de dados $\mathbf{m}_F^{P+1}$
Aplicar os $j = 1, ..., W$ modelos φ_j para predizer o desempenho de cada modelo completo no novo conjunto de dados $\mathbf{w}_W^{P+1}$
$pesos \leftarrow metodo\_roleta(\mathbf{w}^{P+1})$
$n_{max} = \max\{9, R/1000\}$
$s_{max} = \lfloor \log_\eta(n_{max}) \rfloor$ {Início da Otimização}
$B = (s_{max} + 1)R$
for $s \in \{s_{max}, ..., 0\}$ **do**
    $n = \left\lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \right\rceil$, $r = R\eta^{-s}$
    $T = obter\_configuracoes(n, pesos)$
    for i $\in \{0, ..., s$ **do**
        $n_i = \lceil n\eta^{-i} \rceil$
        $r_i = r\eta^i$
        $L = \{avaliacao\_de\_desempenho(t, r_i) : t \in T\}$
        $T = top\_k(T, L, \left\lfloor \frac{n_i}{\eta} \right\rfloor)$
    end for
end for
return Melhor Processo Completo

Algoritmo B.1 – Algoritmo Hyperband com Meta-Aprendizado

APÊNDICE C – PROCESSOS COMPLETOS SELECIONADOS POR UBERBAND E HYPERBAND

Tabela C.1 – Comparação entre os melhores processos completos selecionados por Uberband e Hyperband, incluindo hiper-parâmetros

	Uberband		Hyperband	
Dados	Operadores	Hiper-Parâmetros	Operadores	Hiper-Parâmetros
2dplanes	SC: Nenhum, CL: 'J48'	('useMDL': False, 'collapseTree': True, 'binarySplits': False, 'minNumObj': 46, 'confidenceFactor': 0.0235, 'subtreeRaising': False, 'unpruned': True, 'useLaplace': True)	SC: 'OneRAttributeEval', CL: 'BayesNet'	('search': 'bayes.net.search.local.K2', 'useAT': True),(use_training': False, 'seed': 1, 'minimum_bucket': 5)
ada_prior	SC: Nenhum, CL: 'NaiveBayes'	('useKernelEstimator': False, 'useSupervisedDiscretization': True)	SC: 'RELIEFEval', CL: 'REPTree'	('minVarProp': 0.0463, 'minNum': 53, 'pruning': True, 'maxDepth': 11),(sampleSize': 4, 'sigma': 9, 'numNeighbours': 6, 'weightByDistance': True)
adult	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': True, 'decay': True, 'hiddenLayers': 'i', 'normalizeNumClasses': True, 'seed': 1, 'learningRate': 0.1720, 'momentum': 0.4987, 'nominalToBinaryFilter': False)	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.6331460134144352, 'numIterations': 23, 'seed': 1)
adult-census	SC: Nenhum, CL: 'LMT'	('binary': False, 'crossValidated': True, 'min_inst': 52, 'probabilities': False, 'weighting': 0, 'residuals': True)	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.7576, 'numIterations': 61, 'seed': 1)

ailerons	SC: Nenhum, CL: 'J48'	('useMDL': True, 'collapseTree': True, 'binarySplits': False, 'minNumObj': 50, 'confidenceFactor': 0.4707, 'subtreeRaising': False, 'unpruned': True, 'useLaplace': False)	SC: 'Symmetrical Uncert', CL: 'SVM'	('buildCalibrationModels': True, 'C': 0.6278, 'filterType': 1, 'kernel': 'NormalizedPolyKernel'),('missing_merge': True)
analcatdata_halloffame	SC: Nenhum, CL: 'SimpleLogistic'	('stop': True, 'aic': True, 'weight': 0)	SC: 'OneRAttributeEval', CL: 'NaiveBayes'	('useKernelEstimator': False, 'useSupervisedDiscretization': False),(use_training': True, 'seed': 1, 'minimum_bucket': 8)
analcatdata_supreme	SC: Nenhum, CL: 'LogitBoost'	('per_setsize': 94, 'numIterations': 58, 'seed': 1, 'resampling': True, 'threshold': 4)	SC: 'Info Gain Eval', CL: 'JRip'	('checkerror': True, 'optimizations': 5, 'pruning': True),(missingMerge': True, 'binarizeNumericAttributes': False)
balloon	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.9178, 'numIterations': 4, 'seed': 1)	SC: 'Symmetrical Uncert', CL: 'ZeroR'	('debug': False),(missing_merge': True)
bank32nh	SC: Nenhum, CL: 'SimpleLogistic'	('stop': False, 'aic': True, 'weight': 0)	SC: 'CFSSubsetEval', CL: 'SVM'	('buildCalibrationModels': False, 'C': 0.9876, 'filterType': 1, 'kernel': 'RBFKernel'),('missingSeparate': False, 'search': 'BestFirst', 'locallyPredictive': True)
bank8FM	SC: Nenhum, CL: 'SimpleLogistic'	('stop': True, 'aic': True, 'weight': 0)	SC: 'Gain Ration Eval', CL: 'Stacking'	('folds': 3, 'seed': 1),(missing_merge': True)
car	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 15, 'seed': 1)	SC: 'RELIEF Eval', CL: 'Stacking'	('folds': 3, 'seed': 1),(sampleSize': 7, 'sigma': 7, 'numNeighbours': 3, 'weightByDistance': False)
cmc	SC: Nenhum, CL: 'AdaBoostM1'	('numIterations': 75, 'seed': 1, 'useResampling': False, 'weightThreshold': 93)	SC: Nenhum, CL: 'RandomForest'	('per_setsize': 10, 'numIterations': 5, 'seed': 1, 'out_of_bag': False)

colleges_aaup	SC: Nenhum, CL: 'SimpleLogistic'	('stop': False, 'aic': True, 'weight': 0)	SC: 'Info Gain Eval', CL: 'RandomCommitee'	('numIterations': 51, 'seed': 1),(missingMerge': True, 'binarizeNumericAttributes': True)
colleges_usnews	SC: Nenhum, CL: 'SimpleLogistic'	('stop': True, 'aic': False, 'weight': 0)	SC: Nenhum, CL: 'SimpleLogistic'	('stop': False, 'aic': True, 'weight': 0)
cpu_act	SC: Nenhum, CL: 'RandomForest'	('depth': 13, 'numIterations': 135, 'numattr': 4)	SC: 'RELIEF Eval', CL: 'RandomSubSpace'	('per_setsize': 0.4013, 'numIterations': 32, 'seed': 1),(sampleSize': 0, 'sigma': 9, 'numNeighbours': 2, 'weightByDistance': True)
credit-g	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 'i', 'normalizeNumClasses': True, 'seed': 1, 'learningRate': 0.1189, 'momentum': 0.6711, 'nominalToBinaryFilter': True)	SC: 'RELIEF Eval', CL: 'REPTree'	('minVarProp': 0.09722004643528755, 'minNum': 18, 'pruning': True, 'maxDepth': 14),(sampleSize': 9, 'sigma': 10, 'numNeighbours': 10, 'weightByDistance': False)
delta_ailerons	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 57, 'seed': 1)	SC: 'Info Gain Eval', CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 't', 'normalizeNumClasses': True, 'seed': 1, 'learningRate': 0.7145, 'momentum': 0.6740, 'nominalToBinaryFilter': False),(missingMerge': True, 'binarizeNumericAttributes': True)

delta_elevators	SC: Nenhum, CL: 'LMT'	('binary': True, 'crossValidated': False, 'min_inst': 14, 'probabilities': True, 'weighting': 0, 'residuals': False)	SC: 'Wrapper Sub Eval', CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 'o', 'normalizeNumClasses': False, 'seed': 1, 'learningRate': 0.6445, 'momentum': 0.4071, 'nominalToBinaryFilter': False),(threshold': 0.0229, 'search': 'GreedyStepwise', 'seed': 1, 'ev_measure': 'auc')
electricity-normalized	SC: Nenhum, CL: 'RandomForest'	('depth': 18, 'numIterations': 168, 'numattr': 7)	SC: 'Info Gain Eval', CL: 'RandomForest'	('depth': 9, 'numIterations': 97, 'numattr': 19),(missingMerge': True, 'binarizeNumericAttributes': False)
elevators	SC: Nenhum, CL: 'SVM'	('buildCalibrationModels': True, 'C': 0.5600, 'filterType': 1, 'kernel': 'PolyKernel')	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.7331, 'numIterations': 51, 'seed': 1)
fri_c0_1000_5	SC: Nenhum, CL: 'NaiveBayes'	('useKernelEstimator': False, 'useSupervisedDiscretization': False)	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.3585, 'numIterations': 46, 'seed': 1)
fri_c1_1000_10	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 25, 'seed': 1)	SC: 'Symmetrical Uncert', CL: 'SVM'	('buildCalibrationModels': False, 'C': 0.8337, 'filterType': 2, 'kernel': 'RBFKernel'),('missing_merge': False)
fri_c2_1000_25	SC: Nenhum, CL: 'RandomForest'	('depth': 17, 'numIterations': 172, 'numattr': 14)	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 47, 'seed': 1)
fri_c2_1000_5	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 18, 'seed': 1)	SC: 'Symmetrical Uncert', CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 'i', 'normalizeNumClasses': True, 'seed': 1, 'learningRate': 0.1524, 'momentum': 0.5672, 'nominalToBinaryFilter': False),(missing_merge': False)

fri_c3_1000_5	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 55, 'seed': 1)	SC: 'OneRAttributeEval', CL: 'Stacking'	('folds': 3, 'seed': 1),(use_training': True, 'seed': 1, 'minimum_bucket': 7)
fri_c4_1000_10	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 't', 'normalizeNumClasses': True, 'seed': 1, 'learningRate': 0.6635, 'momentum': 0.7700, 'nominalToBinaryFilter': False)	SC: 'Info Gain Eval', CL: 'J48'	('useMDL': False, 'collapseTree': True, 'binarySplits': True, 'minNumObj': 46, 'confidenceFactor': 0.9608, 'subtreeRaising': False, 'unpruned': True, 'useLaplace': False),(missingMerge': True, 'binarizeNumericAttributes': False)
fri_c4_1000_25	SC: Nenhum, CL: 'RandomForest'	('depth': 8, 'numIterations': 228, 'numattr': 22)	SC: Nenhum, CL: 'NaiveBayes'	('useKernelEstimator': True, 'useSupervisedDiscretization': False)
fried	SC: Nenhum, CL: 'RandomForest'	('depth': 18, 'numIterations': 191, 'numattr': 2)	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.3829, 'numIterations': 59, 'seed': 1)
gina_agnostic	SC: 'Symmetrical Uncert', CL: 'RandomSubSpace'	('per_setsize': 0.6471, 'numIterations': 48, 'seed': 1),(missing_merge': True)	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'	('per_setsize': 0.3380, 'numIterations': 44, 'seed': 1),(missing_merge': True)
house_16H	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.7569, 'numIterations': 32, 'seed': 1)	SC: 'OneRAttributeEval', CL: 'OneRR'	('bucket': 29),(use_training': False, 'seed': 1, 'minimum_bucket': 5)
houses	SC: Nenhum, CL: 'LogitBoost'	('per_setsize': 71, 'numIterations': 64, 'seed': 1, 'resampling': True, 'threshold': 3)	SC: 'OneRAttributeEval', CL: 'RandomSubSpace'	('per_setsize': 0.7461, 'numIterations': 45, 'seed': 1),(use_training': True, 'seed': 1, 'minimum_bucket': 9)
hypothyroid	SC: Nenhum, CL: 'LogitBoost'	('per_setsize': 81, 'numIterations': 44, 'seed': 1, 'resampling': False, 'threshold': 6)	SC: 'RELIEFEval', CL: 'ZeroR'	('debug': True),(sampleSize': 9, 'sigma': 6, 'numNeighbours': 2, 'weightByDistance': False)
jm1	SC: Nenhum, CL: 'RandomForest'	('depth': 16, 'numIterations': 81, 'numattr': 13)	SC: Nenhum, CL: 'RandomForest'	('depth': 5, 'numIterations': 84, 'numattr': 8)

kc1	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 53, 'seed': 1)	SC: 'ClassifierAttribute Eval', CL: 'AdaBoostM1'	('numIterations': 123, 'seed': 1, 'useResampling': True, 'weightThreshold': 86),(outputDetailedInfo': True)
kr-vs-kp	SC: 'Gain Ration Eval', CL: 'Stacking'	('folds': 3, 'seed': 1),(missing_merge': True)	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'	('per_setsize': 0.4124, 'numIterations': 49, 'seed': 1),(missing_merge': True)
letter	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 59, 'seed': 1)	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 48, 'seed': 1)
mfeat-morphological	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': True, 'decay': False, 'hiddenLayers': 'o', 'normalizeNumClasses': True, 'seed': 1, 'learningRate': 0.7516, 'momentum': 0.2353, 'nominalToBinaryFilter': True)	SC: 'Wrapper Sub Eval', CL: 'Stacking'	('folds': 3, 'seed': 1),(threshold': 0.0617, 'search': 'GreedyStepwise', 'seed': 1, 'ev_measure': 'auc')
mfeat-pixel	SC: 'ClassifierAttribute Eval', CL: 'RandomTree'	('unclass': True, 'folds': 0, 'depth': 5, 'minInst': 39, 'numAttr': 0),(outputDetailedInfo': False)	SC: Nenhum, CL: 'RandomForest'	('weighting': 1, 'neighbours': 90)
mfeat-zernike	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 't', 'normalizeNumClasses': False, 'seed': 1, 'learningRate': 0.3087, 'momentum': 0.7056, 'nominalToBinaryFilter': True)	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.4487, 'numIterations': 43, 'seed': 1)
mozilla4	SC: Nenhum, CL: 'DecisionTable'	('evaluation': 'auc', 'search': 'GreedyStepwise', 'useNN': False, 'cv': 1)	SC: 'OneRAttributeEval', CL: 'AdaBoost M1'	('numIterations': 90, 'seed': 1, 'useResampling': True, 'weightThreshold': 89),(use_training': False, 'seed': 1, 'minimum_bucket': 7)
mushroom	SC: 'CFSSubsetEval', CL: 'OneRR'	('bucket': 15),(missingSeparate': True, 'search': 'GreedyStepwise', 'locallyPredictive': False)	SC: Nenhum, CL: 'IBk'	('K': 59, 'crossValidated': False, 'distanceWeighting': 'No distance weighting', 'meanSquared': False)

nursery	SC: 'CFSSubsetEval', CL: 'SGD'	('normalize': False, 'lambda': 2.399, 'learningRate': 0.0575, 'missing': False),(missingSeparate': True, 'search': 'BestFirst', 'locallyPredictive': False)	SC: 'Gain Ration Eval', CL: 'Stacking'	('folds': 3, 'seed': 1),(missing_merge': False)
optdigits	SC: Nenhum, CL: 'SimpleLogistic'	('stop': False, 'aic': False, 'weight': 0)	SC: 'Symmetrical Uncert', CL: 'Stacking'	('folds': 3, 'seed': 1),(missing_merge': False)
page-blocks	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 46, 'seed': 1)	SC: 'Classifier Attribute Eval', CL: 'OneRR'	('bucket': 4),(outputDetailedInfo': False)
pbcseq	SC: Nenhum, CL: 'LogitBoost'	('per_setsize': 89, 'numIterations': 51, 'seed': 1, 'resampling': False, 'threshold': 2)	SC: Nenhum, CL: 'SimpleLogistic'	('stop': True, 'aic': False, 'weight': 0)
pc1	SC: 'CFSSubsetEval', CL: 'SimpleLogistic'	('stop': False, 'aic': False, 'weight': 1),(missingSeparate': False, 'search': 'BestFirst', 'locallyPredictive': True)	SC: Nenhum, CL: 'AdaBoostM1'	('numIterations': 36, 'seed': 1, 'useResampling': False, 'weightThreshold': 95)
pc2	SC: Nenhum, CL: 'BayesNet'	('search': 'weka.classifiers.bayes.net.search.local.K2', 'useAT': False)	SC: 'OneRAttributeEval', CL: 'RandomTree'	('unclass': False, 'KDHillClimber', 'depth': 0, 'minInst': 30, 'numAttr': 0),(use_training': False, 'seed': 1, 'minimum_bucket': 7)
pc3	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 56, 'seed': 1)	SC: 'Gain Ration Eval', CL: 'Stacking'	('folds': 3, 'seed': 1),(missing_merge': False)
pc4	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.8325, 'numIterations': 61, 'seed': 1)	SC: 'CFSSubsetEval', CL: 'Stacking'	('folds': 3, 'seed': 1),(missingSeparate': False, 'search': 'BestFirst', 'locallyPredictive': False)

pendigits	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': False, 'decay': False, 'hiddenLayers': 'i', 'normalizeNumClasses': False, 'seed': 1, 'learningRate': 0.2774, 'momentum': 0.1904, 'nominalToBinaryFilter': True)	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 24, 'seed': 1)
pollen	SC: Nenhum, CL: 'REPTree'	('minVarProp': 0.0749, 'minNum': 3, 'pruning': False, 'maxDepth': 10)	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'	('per_setsize': 0.2176, 'numIterations': 42, 'seed': 1),(missing_merge': True)
puma8NH	SC: Nenhum, CL: 'RandomForest'	('depth': 15, 'numIterations': 167, 'numattr': 13)	SC: 'CFSSubsetEval', CL: 'Logistic'	('ridge': 3.8478e-06),(missingSeparate': True, 'search': 'BestFirst', 'locallyPredictive': False)
quake	SC: Nenhum, CL: 'Logistic'	('ridge': 5.4129e-07)	SC: 'Gain Ration Eval', CL: 'IBk'	('K': 53, 'crossValidated': True, 'distanceWeighting': 'Weight by 1-distance', 'meanSquared': False),(missing_merge': False)
rmftsa_sleepdata	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.9520, 'numIterations': 62, 'seed': 1)	SC: 'Gain Ration Eval', CL: 'RandomForest'	('per_setsize': 38, 'numIterations': 96, 'seed': 1, 'out_of_bag': False),(missing_merge': False)
segment	SC: Nenhum, CL: 'SimpleLogistic'	('stop': True, 'aic': False, 'weight': 0)	SC: Nenhum, CL: 'NaiveBayes'	('useKernelEstimator': False, 'useSupervisedDiscretization': True)
sick	SC: Nenhum, CL: 'LogitBoost'	('per_setsize': 92, 'numIterations': 58, 'seed': 1, 'resampling': False, 'threshold': 3)	SC: 'Gain Ration Eval', CL: 'RandomSubSpace'	('per_setsize': 0.3939, 'numIterations': 35, 'seed': 1),(missing_merge': True)
spambase	SC: Nenhum, CL: 'RandomForest'	('depth': 14, 'numIterations': 113, 'numattr': 8)	SC: 'Symmetrical Uncert', CL: 'RandomTree'	('unclass': False, 'folds': 0, 'depth': 0, 'minInst': 20, 'numAttr': 32),(missing_merge': True)

splice	SC: Nenhum, CL: 'LMT'	('binary': False, 'crossValidated': False, 'min_inst': 47, 'probabilities': True, 'weighting': 0.6398, 'residuals': False)	SC: Nenhum, CL: 'JRip'	('checkerror': True, 'optimizations': 2, 'pruning': False)
thyroid_sick	SC: Nenhum, CL: 'RandomCommitee'	('numIterations': 59, 'seed': 1)	SC: 'RELIEFEval', CL: 'Vote'	('seed': 1, 'rule': 'PROD'),('sampleSize': 7, 'sigma': 6, 'numNeighbours': 9, 'weightByDistance': False)
wind	SC: Nenhum, CL: 'MultilayerPerceptron'	('reset': False, 'decay': True, 'hiddenLayers': 'i', 'normalizeNumClasses': False, 'seed': 1, 'learningRate': 0.9563, 'momentum': 0.2434, 'nominalToBinaryFilter': False)	SC: Nenhum, CL: 'RandomSubSpace'	('per_setsize': 0.9095, 'numIterations': 8, 'seed': 1)

ANEXO A – Conjuntos de Dados

Tabela A.1 – Principais características dos conjuntos de dados utilizados na Fase de Meta-Aprendizado, incluindo nome do conjunto de dados (dados), número de instâncias (#inst.), número de atributos (#atrib.), dimensionalidade (dimen.), porcentagem de atributos categóricos (%Cat.), porcentagem de atributos numéricos (%Num.), porcentagem de valores ausentes (%aus.) e porcentagem de instâncias da classe majoritária (%Maj.).

dados	#inst.	#atrib.	dimen.	%Cat.	%Num.	%Aus.	%Maj
2dplanes	40768	11	0.000	0.00	100.00	0.00	50.09
ada prior	4562	15	0.003	57.14	42.86	0.13	75.19
adult census	32561	16	0.000	53.33	46.67	0.82	75.92
adult	48842	15	0.000	85.71	14.29	0.88	76.07
ailerons	13750	41	0.003	0.00	100.00	0.00	57.61
analcatdata apnea1	475	4	0.008	66.67	33.33	0.00	87.16
analcatdata apnea2	475	4	0.008	66.67	33.33	0.00	86.53
analcatdata apnea3	450	4	0.009	66.67	33.33	0.00	87.78
analcatdata authorship	841	71	0.084	0.00	100.00	0.00	62.31
analcatdata birthday	365	4	0.011	66.67	33.33	2.05	85.48
analcatdata boxing1	120	4	0.033	100.00	0.00	0.00	65.00
analcatdata boxing2	132	4	0.030	100.00	0.00	0.00	53.79
analcatdata broadway-mult	285	8	0.028	57.14	42.86	1.18	58.60
analcatdata creditscore	100	7	0.070	50.00	50.00	0.00	73.00

analcatdata dmft	797	5	0.006	100.00	0.00	0.00	80.55
analcatdata draft	366	6	0.016	40.00	60.00	0.05	91.26
analcatdata gsssexsur-vey	159	10	0.063	55.56	44.44	0.38	77.99
analcatdata halloffame	1340	18	0.013	11.76	88.24	0.08	90.67
analcatdata lawsuit	264	5	0.019	25.00	75.00	0.00	92.80
analcatdata marketing	364	33	0.091	100.00	0.00	0.67	68.41
analcatdata michiganacc	108	5	0.046	50.00	50.00	0.00	55.56
analcatdata neavote	100	4	0.040	66.67	33.33	0.00	93.00
analcatdata reviewer	379	9	0.024	100.00	0.00	40.11	56.99
analcatdata seropositive	132	4	0.030	33.33	66.67	0.00	65.15
analcatdata supreme	4052	8	0.002	0.00	100.00	0.00	76.04
analcatdata vineyard	468	4	0.009	33.33	66.67	0.00	55.56
analcatdata wildcat	163	6	0.037	40.00	60.00	0.00	71.17
anneal	898	39	0.043	84.21	15.79	63.32	76.17
AP Breast Kidney	604	10937	18.108	0.00	100.00	0.00	56.95
AP Breast Omentum	421	10937	25.979	0.00	100.00	0.00	81.71
AP Breast Prostate	413	10937	26.482	0.00	100.00	0.00	83.29
AP Colon Kidney	546	10937	20.031	0.00	100.00	0.00	52.38
AP Colon Omentum	363	10937	30.129	0.00	100.00	0.00	78.79

AP Colon Ovary	484	10937	22.597	0.00	100.00	0.00	59.09
AP Colon Prostate	355	10937	30.808	0.00	100.00	0.00	80.56
AP Endometrium Breast	405	10937	27.005	0.00	100.00	0.00	84.94
AP Endometrium Lung	187	10937	58.487	0.00	100.00	0.00	67.38
AP Endometrium Prostate	130	10937	84.131	0.00	100.00	0.00	53.08
AP Lung Uterus	250	10937	43.748	0.00	100.00	0.00	50.40
AP Omentum Kidney	337	10937	32.454	0.00	100.00	0.00	77.15
AP Omentum Lung	203	10937	53.877	0.00	100.00	0.00	62.07
AP Omentum Ovary	275	10937	39.771	0.00	100.00	0.00	72.00
AP Omentum Uterus	201	10937	54.413	0.00	100.00	0.00	61.69
AP Ovary Uterus	322	10937	33.966	0.00	100.00	0.00	61.49
AP Prostate Kidney	329	10937	33.243	0.00	100.00	0.00	79.03
AP Prostate Uterus	193	10937	56.668	0.00	100.00	0.00	64.25
AP Uterus Kidney	384	10937	28.482	0.00	100.00	0.00	67.71
ar1	121	30	0.248	0.00	100.00	0.00	92.56
ar4	107	30	0.280	0.00	100.00	0.00	81.31
ar6	101	30	0.297	0.00	100.00	0.00	85.15
arsenic female bladder	559	5	0.009	25.00	75.00	0.00	85.69
arsenic female lung	559	5	0.009	25.00	75.00	0.00	96.60
arsenic male bladder	559	5	0.009	25.00	75.00	0.00	95.71

arsenic male lung	559	5	0.009	25.00	75.00	0.00	97.67
audiology	226	70	0.310	100.00	0.00	2.00	74.78
autoHorse	205	26	0.127	32.00	68.00	1.07	59.51
autoMpg	398	8	0.020	42.86	57.14	0.19	52.51
autoPrice	159	16	0.101	0.00	100.00	0.00	66.04
autos	205	26	0.127	40.00	60.00	1.11	67.32
auto price	159	16	0.101	6.67	93.33	0.00	66.04
backache	180	33	0.183	81.25	18.75	0.00	86.11
badges2	294	12	0.041	27.27	72.73	0.00	71.43
balance scale	625	5	0.008	0.00	100.00	0.00	53.92
balloon	2001	3	0.001	0.00	100.00	0.00	75.91
bank32nh	8192	33	0.004	0.00	100.00	0.00	68.96
bank8FM	8192	9	0.001	0.00	100.00	0.00	59.63
biomed	209	9	0.043	12.50	87.50	0.80	64.11
bodyfat	252	15	0.060	0.00	100.00	0.00	50.79
boston	506	14	0.028	15.38	84.62	0.00	58.70
boston corrected	506	21	0.042	15.00	85.00	0.00	55.93
breast cancer	286	10	0.035	100.00	0.00	0.31	70.28
breast w	699	10	0.014	0.00	100.00	0.23	65.52
breastTumor	286	10	0.035	88.89	11.11	0.31	58.04
bridges	107	13	0.121	75.00	25.00	5.10	58.88
cal housing	20640	9	0.000	0.00	100.00	0.00	59.38
car	1728	7	0.004	100.00	0.00	0.00	70.02
cars	406	9	0.022	25.00	75.00	0.38	62.56
chatfield 4	235	13	0.055	0.00	100.00	0.00	60.43
cholesterol	303	14	0.046	53.85	46.15	0.14	54.79
chscase census2	400	8	0.020	0.00	100.00	0.00	50.75
chscase census3	400	8	0.020	0.00	100.00	0.00	52.00
chscase census5	400	8	0.020	0.00	100.00	0.00	51.75
chscase census6	400	7	0.018	0.00	100.00	0.00	58.75

chscase funds	185	4	0.022	33.33	66.67	0.00	52.97
chscase geyser1	222	3	0.014	0.00	100.00	0.00	60.36
chscase vine2	468	3	0.006	0.00	100.00	0.00	54.70
chscase whale	228	10	0.044	0.00	100.00	0.88	51.32
cleveland	303	14	0.046	53.85	46.15	0.14	54.13
cmc	1473	10	0.007	77.78	22.22	0.00	57.30
colic	368	23	0.063	68.18	31.82	22.77	63.04
colleges aaup	1161	17	0.015	18.75	81.25	1.30	70.03
colleges usnews	1302	35	0.027	5.88	94.12	17.18	52.84
collins	500	24	0.048	13.04	86.96	0.00	84.00
cpu	209	8	0.038	14.29	85.71	0.00	74.64
cpu act	8192	22	0.003	0.00	100.00	0.00	69.76
cpu small	8192	13	0.002	0.00	100.00	0.00	69.76
credit a	690	16	0.023	60.00	40.00	0.61	55.51
credit g	1000	21	0.021	65.00	35.00	0.00	70.00
cylinder bands	540	40	0.074	53.85	46.15	4.63	57.78
datatrieve	130	9	0.069	0.00	100.00	0.00	91.54
delta ailerons	7129	6	0.001	0.00	100.00	0.00	53.06
delta elevators	9517	7	0.001	0.00	100.00	0.00	50.28
diabetes	768	9	0.012	0.00	100.00	0.00	65.10
diggle table a2	310	9	0.029	12.50	87.50	0.00	53.23
disclosure x bias	662	4	0.006	0.00	100.00	0.00	52.11
disclosure x noise	662	4	0.006	0.00	100.00	0.00	50.30
disclosure x tampered	662	4	0.006	0.00	100.00	0.00	50.60
disclosure z	662	4	0.006	0.00	100.00	0.00	52.57
echoMonths	130	10	0.077	33.33	66.67	7.46	50.77
ecoli	336	8	0.024	0.00	100.00	0.00	57.44

electricity normalized	45312	9	0.000	12.50	87.50	0.00	57.55
elevators	16599	19	0.001	0.00	100.00	0.00	69.09
flags	194	30	0.155	93.10	6.90	0.00	64.43
fried	40768	11	0.000	0.00	100.00	0.00	50.11
fri c0 1000 10	1000	11	0.011	0.00	100.00	0.00	50.90
fri c0 1000 25	1000	26	0.026	0.00	100.00	0.00	50.30
fri c0 1000 5	1000	6	0.006	0.00	100.00	0.00	50.30
fri c0 1000 50	1000	51	0.051	0.00	100.00	0.00	51.00
fri c0 100 10	100	11	0.110	0.00	100.00	0.00	55.00
fri c0 100 25	100	26	0.260	0.00	100.00	0.00	50.00
fri c0 100 5	100	6	0.060	0.00	100.00	0.00	54.00
fri c0 100 50	100	51	0.510	0.00	100.00	0.00	51.00
fri c0 250 10	250	11	0.044	0.00	100.00	0.00	50.00
fri c0 250 25	250	26	0.104	0.00	100.00	0.00	50.40
fri c0 250 5	250	6	0.024	0.00	100.00	0.00	50.00
fri c0 250 50	250	51	0.204	0.00	100.00	0.00	53.20
fri c0 500 10	500	11	0.022	0.00	100.00	0.00	51.80
fri c0 500 25	500	26	0.052	0.00	100.00	0.00	51.00
fri c0 500 5	500	6	0.012	0.00	100.00	0.00	50.20
fri c0 500 50	500	51	0.102	0.00	100.00	0.00	51.20
fri c1 1000 10	1000	11	0.011	0.00	100.00	0.00	56.40
fri c1 1000 25	1000	26	0.026	0.00	100.00	0.00	54.60
fri c1 1000 5	1000	6	0.006	0.00	100.00	0.00	54.30
fri c1 1000 50	1000	51	0.051	0.00	100.00	0.00	54.70
fri c1 100 10	100	11	0.110	0.00	100.00	0.00	53.00
fri c1 100 25	100	26	0.260	0.00	100.00	0.00	53.00
fri c1 100 5	100	6	0.060	0.00	100.00	0.00	55.00
fri c1 100 50	100	51	0.510	0.00	100.00	0.00	56.00
fri c1 250 10	250	11	0.044	0.00	100.00	0.00	56.00
fri c1 250 25	250	26	0.104	0.00	100.00	0.00	57.20
fri c1 250 5	250	6	0.024	0.00	100.00	0.00	52.40
fri c1 250 50	250	51	0.204	0.00	100.00	0.00	54.80
fri c1 500 10	500	11	0.022	0.00	100.00	0.00	54.80
fri c1 500 25	500	26	0.052	0.00	100.00	0.00	53.40
fri c1 500 5	500	6	0.012	0.00	100.00	0.00	53.40
fri c1 500 50	500	51	0.102	0.00	100.00	0.00	52.40
fri c2 1000 10	1000	11	0.011	0.00	100.00	0.00	58.00
fri c2 1000 25	1000	26	0.026	0.00	100.00	0.00	56.30

fri c2 1000 5	1000	6	0.006	0.00	100.00	0.00	58.40
fri c2 1000 50	1000	51	0.051	0.00	100.00	0.00	58.20
fri c2 100 10	100	11	0.110	0.00	100.00	0.00	55.00
fri c2 100 25	100	26	0.260	0.00	100.00	0.00	57.00
fri c2 100 5	100	6	0.060	0.00	100.00	0.00	60.00
fri c2 100 50	100	51	0.510	0.00	100.00	0.00	58.00
fri c2 250 10	250	11	0.044	0.00	100.00	0.00	63.60
fri c2 250 25	250	26	0.104	0.00	100.00	0.00	55.60
fri c2 250 5	250	6	0.024	0.00	100.00	0.00	56.00
fri c2 250 50	250	51	0.204	0.00	100.00	0.00	54.80
fri c2 500 10	500	11	0.022	0.00	100.00	0.00	57.20
fri c2 500 25	500	26	0.052	0.00	100.00	0.00	60.80
fri c2 500 5	500	6	0.012	0.00	100.00	0.00	59.60
fri c2 500 50	500	51	0.102	0.00	100.00	0.00	59.00
fri c3 1000 10	1000	11	0.011	0.00	100.00	0.00	56.00
fri c3 1000 25	1000	26	0.026	0.00	100.00	0.00	55.70
fri c3 1000 5	1000	6	0.006	0.00	100.00	0.00	56.30
fri c3 1000 50	1000	51	0.051	0.00	100.00	0.00	55.50
fri c3 100 10	100	11	0.110	0.00	100.00	0.00	60.00
fri c3 100 25	100	26	0.260	0.00	100.00	0.00	55.00
fri c3 100 5	100	6	0.060	0.00	100.00	0.00	56.00
fri c3 100 50	100	51	0.510	0.00	100.00	0.00	62.00
fri c3 250 10	250	11	0.044	0.00	100.00	0.00	54.00
fri c3 250 25	250	26	0.104	0.00	100.00	0.00	55.60
fri c3 250 5	250	6	0.024	0.00	100.00	0.00	56.40
fri c3 250 50	250	51	0.204	0.00	100.00	0.00	56.80
fri c3 500 10	500	11	0.022	0.00	100.00	0.00	54.40
fri c3 500 25	500	26	0.052	0.00	100.00	0.00	56.00
fri c3 500 5	500	6	0.012	0.00	100.00	0.00	52.60
fri c3 500 50	500	51	0.102	0.00	100.00	0.00	56.40
fri c4 1000 10	1000	11	0.011	0.00	100.00	0.00	56.00
fri c4 1000 100	1000	101	0.101	0.00	100.00	0.00	56.40
fri c4 1000 25	1000	26	0.026	0.00	100.00	0.00	54.70
fri c4 1000 50	1000	51	0.051	0.00	100.00	0.00	56.00
fri c4 100 10	100	11	0.110	0.00	100.00	0.00	53.00
fri c4 100 100	100	101	1.010	0.00	100.00	0.00	53.00
fri c4 100 25	100	26	0.260	0.00	100.00	0.00	54.00
fri c4 100 50	100	51	0.510	0.00	100.00	0.00	56.00

fri c4 250 10	250	11	0.044	0.00	100.00	0.00	53.20
fri c4 250 100	250	101	0.404	0.00	100.00	0.00	56.00
fri c4 250 25	250	26	0.104	0.00	100.00	0.00	54.40
fri c4 250 50	250	51	0.204	0.00	100.00	0.00	54.00
fri c4 500 10	500	11	0.022	0.00	100.00	0.00	55.20
fri c4 500 100	500	101	0.202	0.00	100.00	0.00	56.60
fri c4 500 25	500	26	0.052	0.00	100.00	0.00	56.80
fri c4 500 50	500	51	0.102	0.00	100.00	0.00	52.80
fruitfly	125	5	0.040	50.00	50.00	0.00	60.80
gina agnostic	3468	971	0.280	0.00	100.00	0.00	50.84
gina prior	3468	785	0.226	0.00	100.00	0.00	50.84
glass	214	10	0.047	0.00	100.00	0.00	64.49
haberman	306	4	0.013	33.33	66.67	0.00	73.53
hayes roth	132	5	0.038	0.00	100.00	0.00	61.36
heart c	303	14	0.046	53.85	46.15	0.17	54.46
heart statlog	270	14	0.052	0.00	100.00	0.00	55.56
hepatitis	155	20	0.129	68.42	31.58	5.39	79.35
houses	20640	9	0.000	0.00	100.00	0.00	56.81
house 16H	22784	17	0.001	0.00	100.00	0.00	70.40
house 8L	22784	9	0.000	0.00	100.00	0.00	70.40
housing	506	14	0.028	7.69	92.31	0.00	58.70
humandevel	130	4	0.031	33.33	66.67	0.00	50.00
hungarian	294	14	0.048	53.85	46.15	19.00	63.95
hypothyroid	3772	30	0.008	75.86	24.14	5.36	92.29
ionosphere	351	35	0.100	0.00	100.00	0.00	64.10
iris	150	5	0.033	0.00	100.00	0.00	66.67
irish	500	6	0.012	60.00	40.00	1.07	65.00
jEdit 4.0 4.2	274	9	0.033	0.00	100.00	0.00	51.09
jEdit 4.2 4.3	369	9	0.024	0.00	100.00	0.00	55.28
jm1	10885	22	0.002	0.00	100.00	0.01	80.65
kc1 binary	145	95	0.655	0.00	100.00	0.00	58.62
kc1 top5	145	95	0.655	0.00	100.00	0.00	94.48
kc1	2109	22	0.010	0.00	100.00	0.00	84.54
kc3	458	40	0.087	0.00	100.00	0.00	90.61
kdd el nino small	782	9	0.012	25.00	75.00	6.62	64.96
kdd internet usage	10108	69	0.007	100.00	0.00	0.39	73.14

kdd ipums la 97 small	7019	61	0.009	45.00	55.00	10.23	63.04
kdd ipums la 98 small	7485	56	0.007	70.91	29.09	7.74	89.43
kdd synthetic control	600	62	0.103	1.64	98.36	0.00	83.33
kin8nm	8192	9	0.001	0.00	100.00	0.00	50.88
kr vs kp	3196	37	0.012	100.00	0.00	0.00	52.22
letter	20000	17	0.001	0.00	100.00	0.00	95.94
lowbwt	189	10	0.053	77.78	22.22	0.00	52.38
lymph	148	19	0.128	83.33	16.67	0.00	54.73
machine cpu	209	7	0.033	0.00	100.00	0.00	73.21
MagicTelescope	19020	12	0.001	0.00	100.00	0.00	64.84
mammography	11183	7	0.001	0.00	100.00	0.00	97.68
mc1	9466	39	0.004	0.00	100.00	0.00	99.28
mc2	161	40	0.248	0.00	100.00	0.00	67.70
meta	528	22	0.042	9.52	90.48	4.34	89.77
mfeat factors	2000	217	0.109	0.00	100.00	0.00	90.00
mfeat karhunen	2000	65	0.033	0.00	100.00	0.00	90.00
mfeat morphological	2000	7	0.004	0.00	100.00	0.00	90.00
mfeat pixel	2000	241	0.121	100.00	0.00	0.00	90.00
mfeat zernike	2000	48	0.024	0.00	100.00	0.00	90.00
molecular biology promoters	106	59	0.557	100.00	0.00	0.00	67.92
monks problems 1	554	7	0.013	100.00	0.00	0.00	50.36
monks problems 2	601	7	0.012	100.00	0.00	0.00	50.08
monks problems 3	556	7	0.013	100.00	0.00	0.00	51.08
mozilla4	15545	6	0.000	0.00	100.00	0.00	67.14
mu284	284	11	0.039	0.00	100.00	0.00	50.00
mushroom	8124	23	0.003	100.00	0.00	1.33	51.80
musk	6598	170	0.026	1.18	98.82	0.00	84.59
newton hema	140	4	0.029	33.33	66.67	0.00	50.00

no2	500	8	0.016	0.00	100.00	0.00	50.20
nursery	12960	9	0.001	100.00	0.00	0.00	66.67
optdigits	5620	65	0.012	0.00	100.00	0.00	89.82
OVA Breast	1545	10937	7.079	0.00	100.00	0.00	77.73
OVA Endometrium	1545	10937	7.079	0.00	100.00	0.00	96.05
OVA Ovary	1545	10937	7.079	0.00	100.00	0.00	87.18
OVA Uterus	1545	10937	7.079	0.00	100.00	0.00	91.97
page blocks	5473	11	0.002	0.00	100.00	0.00	89.77
pbc	418	19	0.045	44.44	55.56	15.60	55.02
pbcseq	1945	19	0.010	33.33	66.67	3.07	50.03
pc1	1109	22	0.020	0.00	100.00	0.00	93.06
pc1 req	320	9	0.028	12.50	87.50	0.00	66.56
pc2	5589	37	0.007	0.00	100.00	0.00	99.59
pc3	1563	38	0.024	0.00	100.00	0.00	89.76
pc4	1458	38	0.026	0.00	100.00	0.00	87.79
pendigits	10992	17	0.002	0.00	100.00	0.00	89.59
pharynx	195	12	0.062	90.91	9.09	0.09	62.05
plasma retinol	315	14	0.044	23.08	76.92	0.00	57.78
pm10	500	8	0.016	0.00	100.00	0.00	50.80
pol	15000	49	0.003	0.00	100.00	0.00	66.39
pollen	3848	6	0.002	0.00	100.00	0.00	50.00
primary tumor	339	18	0.053	100.00	0.00	3.69	75.22
prnn fglass	214	10	0.047	0.00	100.00	0.00	64.49
prnn synth	250	3	0.012	0.00	100.00	0.00	50.00
puma32H	8192	33	0.004	0.00	100.00	0.00	50.39
puma8NH	8192	9	0.001	0.00	100.00	0.00	50.22
pwLinear	200	11	0.055	0.00	100.00	0.00	51.50
quake	2178	4	0.002	0.00	100.00	0.00	55.51
rabe 266	120	3	0.025	0.00	100.00	0.00	52.50
rmftsa ctoarrivals	264	3	0.011	50.00	50.00	0.00	61.74
rmftsa ladata	508	11	0.022	0.00	100.00	0.00	56.30
rmftsa sleepdata	1024	3	0.003	50.00	50.00	0.00	50.29
scene	2407	300	0.125	1.67	98.33	0.00	82.09
schizo	340	15	0.044	14.29	85.71	16.35	52.06

segment	2310	20	0.009	0.00	100.00	0.00	85.71
sensory	576	12	0.021	100.00	0.00	0.00	58.51
servo	167	5	0.030	100.00	0.00	0.00	77.25
sick	3772	30	0.008	75.86	24.14	5.36	93.88
sleuth case2002	147	7	0.048	66.67	33.33	0.00	53.06
socmob	1156	6	0.005	80.00	20.00	0.00	77.85
solar flare 1	323	13	0.040	100.00	0.00	0.00	97.83
sonar	208	61	0.293	0.00	100.00	0.00	53.37
space ga	3107	7	0.002	0.00	100.00	0.00	50.40
spambase	4601	58	0.013	0.00	100.00	0.00	60.60
SPECT	267	23	0.086	100.00	0.00	0.00	58.80
SPECTF	349	45	0.129	0.00	100.00	0.00	72.78
spectrometer	531	103	0.194	1.96	98.04	0.00	89.64
splice	3190	62	0.019	100.00	0.00	0.00	51.88
stock	950	10	0.011	0.00	100.00	0.00	51.37
strikes	625	7	0.011	0.00	100.00	0.00	50.40
sylva agnostic	14395	217	0.015	0.00	100.00	0.00	93.85
sylva prior	14395	109	0.008	0.00	100.00	0.00	93.85
tae	151	6	0.040	40.00	60.00	0.00	65.56
tecator	240	125	0.521	0.00	100.00	0.00	57.50
thyroid sick	3772	30	0.008	75.86	24.14	5.36	93.88
tic tac toe	958	10	0.010	100.00	0.00	0.00	65.34
transplant	131	4	0.031	0.00	100.00	0.00	63.36
triazines	186	61	0.328	0.00	100.00	0.00	58.60
vehicle	846	19	0.022	0.00	100.00	0.00	74.23
vehicle sensIT	98528	101	0.001	0.00	100.00	0.00	50.00
veteran	137	8	0.058	57.14	42.86	0.00	68.61
vinnie	380	3	0.008	0.00	100.00	0.00	51.32
visualizing environmental	111	4	0.036	0.00	100.00	0.00	52.25
visualizing galaxy	323	5	0.015	0.00	100.00	0.00	54.18
visualizing livestock	130	3	0.023	100.00	0.00	0.00	80.77
vote	435	17	0.039	100.00	0.00	5.30	61.38

water treatment	527	39	0.074	44.74	55.26	2.72	84.82
waveform 5000	5000	41	0.008	0.00	100.00	0.00	66.16
wind	6574	15	0.002	0.00	100.00	0.00	53.26
wisconsin	194	33	0.170	0.00	100.00	0.00	53.61
yeast ml8	2417	117	0.048	11.21	88.79	0.00	98.59
zoo	101	18	0.178	94.12	5.88	0.00	59.41

ANEXO B – Hiper-Parâmetros Configurados no Espaço de Busca Experimental do Uberband

Tabela B.1 – Métodos de Pré-Processamento e Algoritmos de Aprendizado do espaço de busca do Uberband e seus respectivos hiper-parâmetros

Algoritmo	Parâmetro	Valores	Default
AdaBoost M1	Q	{True, False}	False
	P	[50,100]	100
	I	[2,128]	10
	S	{1}	1
BayesNet	D	{True, False}	False
	Q	{K2, HillClimber, LAGDHillClimber, SimulatedAnnealing, TabuSearch, TAN}	K2
Decision Stump	D	{True, False}	False
Decision Table	E	{acc, rmse, mae, auc}	acc
	I	{True, False}	False
	S	{BestFirst,GreedyStepwise}	BestFirst
	X	{1, 2, 3, 4}	1
Hoeffding Tree	L	[0, 2]	0
	S	[0, 1]	1
	E	[0.0001, 0.01]	0.001
	H	[0, 0.1]	0
	M	[0, 0.1]	0
	G	[0, 100]	10
J48	O	{True, False}	False
	U	{True, False}	False
	B	{True, False}	False
	J	{True, False}	False
	A	{True, False}	False
	S	{True, False}	False
	M	[1,64]	2
	C	[0,1]	0.25
	N	[1,5]	2
Jrip	E	{True, False}	False

	P	{True, False}	False
	O	[1,5]	2
	E	{True, False}	False
	K	[1,64]	1
IBk	X	{True, False}	False
	F	{True, False}	False
	I	{True, False}	False
	B	{True, False}	False
	R	{True, False}	False
	C	{True, False}	False
LMT	P	{True, False}	False
	M	[1,64]	15
	W	{0,1}	0
	A	{True, False}	False
	Q	{True, False}	False
	P	[10, 100]	10
LogitBoost	I	[2,64]	10
	S	{1}	1
	Z	[1,10]	1
Logistic	R	[1e-12, 10]	1.00E-07
	L	[0.1, 1]	0.3
	M	[0.1, 1]	0.2
	B	{True, False}	False
MultilayerPerceptron	H	{a, i, o, t}	a
	C	{True, False}	False
	R	{True, False}	False
	D	{True, False}	False
	S	{1}	1
NaiveBayes	K	{True, False}	False
	D	{True, False}	False
OneR	B	[1, 32]	6
	N	[2, 5]	3
PART	M	[1, 64]	2
	R	{True, False}	False
	B	{True, False}	False
	I	[2, 256]	10
RandomForest	K	{0, 1}	0
	depth	{0, 1}	0
	M	[1, 64]	1
RandomTree			

	K	[2, 32]	2
	depth	[2, 20]	2
	N	[2, 5]	3
	U	{True, False}	False
	I	[2,64]	10
RandomSubSpace	P	[0.01, 1.0]	0.5
	S	{1}	1
RandomCommitte	I	[2,64]	10
	S	{1}	1
	M	[1, 64]	2
REPTree	V	[1e-5, 1e-1]	1.00E-03
	L	[2, 20]	2
	P	{True, False}	False
Stacking	X	{10}	10
	S	{1}	1
	S	{True, False}	False
SimpleLogistic	W	{0, 1}	0
	A	{True, False}	False
	F	{0, 1, 2}	0
	L	[0.00001, 0.1]	0.01
SGD	R	[1e-12, 10]	1.00E-04
	N	{True, False}	False
	M	{True, False}	False
	C	[0.5, 1.5]	1
SVM	N	{0, 1, 2}	0
	M	{True, False}	False
	K	{NormalizedPolyKernel, PolyKernel, Puk, RBFKernel}	NormalizedPolyKernel
Vote	R	{AVG,PROD,MAJ,MIN,MAX}	AVG
	S	{1}	1
	I	[1, 10]	1
VotedPerceptron	M	[5000, 50000]	10000
	E	[0.2, 5]	1
ZeroR	D	{True, False}	False
	M	{True, False}	False
CFSSubsetEval	L	{True, False}	False
	search	{GreedyStepwise, BestFirst}	BestFirst

ClassifierAttributeEval	D	{True, False}	False
	search	{Ranker}	Ranker
GainRatioEval	M	{True, False}	False
	search	{Ranker}	Ranker
InfoGainEval	M	{True, False}	False
	B	{True, False}	False
	search	{Ranker}	Ranker
OneR Attribute Eval	D	{True, False}	False
	S	{1}	1
	B	[1, 10]	1
	search	{Ranker}	Ranker
RELIEF Eval	W	{True, False}	False
	M	[-1, 10]	1
	K	[1, 10]	1
	A	[1, 10]	1
	search	{Ranker}	Ranker
SymmetricalUncert	M	{True, False}	False
	search	{Ranker}	Ranker
WrapperSub Eval	E	{acc, rmse, mae, f-meas, auc, auprc}	acc
	R	{1}	1
	T	[0.01, 01]	0.001
	search	{GreedyStepwise, BestFirst}	BestFirst
BestFirst	N	{0, 1, 2}	0
	D	[2, 10]	2
Greedy Stepwise	C	{True, False}	False
	B	{True, False}	False
	R	{True, False}	False
Ranker	N	[10, 1000]	10

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br