Dissertação de Mestrado

Otimização de hiperparâmetros do XGBoost utilizando meta-aprendizagem

Tiago Lima Marinho
tlm@ic.ufal.br

Orientador:

Prof. Dr. Bruno Pimentel

Maceió, Dezembro de 2021

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

M338o Marinho, Tiago Lima.
Otimização de hiperparâmetros do XGBoost utilizando meta-aprendizagem / Tiago Lima Marinho. – 2021.
71 f. : il.

Orientador: Bruno Pimentel.
Dissertação (mestrado em Informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió.

Bibliografia: f. 56-59.
Apêndices: f. 60-71.

1. Meta-aprendizagem. 2. Aprendizagem de máquina. 3. Custo. 4. Ciência de dados. 5. XGBoost. I. Título.

CDU: 004.81:159.953.5

Tiago Lima Marinho

Otimização de hiperparâmetros do XGBoost utilizando meta-aprendizagem

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Informática de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Bruno Pimentel

Maceió, Dezembro de 2021

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PPGI
Instituto de Computação/UFAL
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Folha de Aprovação

TIAGO LIMA MARINHO

OTIMIZAÇÃO DE HIPERPAR METROS DO XGBOOST UTILIZANDO META-APRENDIZAGEM

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 16 de dezembro de 2021.

Banca Examinadora:

Prof. Dr. BRUNO ALMEIDA PIMENTEL
UFAL – Instituto de Computação
Orientador

Prof. Dr. EVANDRO DE BARROS COSTA
UFAL – Instituto de Computação
Examinador Interno

Prof. Dr. ROBERTA VILHENA VIEIRA LOPES
UFAL – Instituto de Computação
Examinador Externo

Prof. Dr. DIEGO CARVALHO DO NASCIMENTO
UDA - Universidad de Atacama
Examinador Externo

Agradecimentos

Gostaria de agradecer ao Prof. Dr. Bruno Pimentel, meu orientador, por ser sempre atencioso e prestativo.

À minha família e em especial aos meus pais pelo apoio, incentivo e compreensão ao longo do curso.

Aos meus primos Dimitri e Tarsis, que foram uma peça fundamental no meu crescimento profissional e pelo incentivo.

Aos meus amigos que me ajudaram ao longo do curso, principalmente nos primeiros períodos, onde foram os momentos mais apertados.

À Débora, por sua compreensão, companheirismo, incentivo e apoio sempre.

À todos os companheiros e amigos que conheci durante a faculdade e àqueles que não foram citados aqui mas que também participaram deste processo.

Aos meus amigos fora da faculdade, pelo incentivo em todos os momentos.

À Deus.

Resumo

Com a evolução computacional, houve um crescimento no número de algoritmos de aprendizagem de máquina e em paralelo, foram se tornando cada vez mais robustos. Este crescimento nos algoritmos de aprendizagem de máquina, ocasionou uma complexidade maior na configuração dos algoritmos, com o intuito de aumentar a precisão. Com isso, a escolha dos hiperparâmetros mais adequados para um determinado conjunto de dados, pode ser uma tarefa custosa tanto em questão de tempo, quanto em questão de dinheiro. Sendo assim, é necessário que hajam maneiras mais rápidas e práticas para achar hiperparâmetros que vão configurar cada algoritmo individualmente. Este trabalho visa utilizar da Meta-Aprendizagem como uma solução viável para a recomendação de hiperparâmetros para o recente algoritmo de aprendizagem de máquina XGBoost, a fim de que haja uma redução de custos computacionais, visando também a redução de custo para as empresas. Este trabalho utilizou de 198 conjuntos de dados, seguindo a ideia de validação cruzada *leave-one-out* para os experimentos, fazendo assim, com que cada um dos conjuntos de dados fossem testados em comparação a todos os outros disponíveis. Além disso, foram utilizados 3 conjuntos de Meta-Características disponíveis na literatura: *general*, *statistical* e *info-theory* para a fase de caracterização de cada um dos conjuntos de dados fazendo assim, com que houvesse uma comparação de similaridade entre os conjuntos de dados para que dessa forma, pudesse ser aplicado a Meta-Aprendizagem e ter por fim, a recomendação de cada um dos hiperparâmetros. Os resultados obtidos foram promissores, fazendo com que em alguns casos, 86.36% dos testes tivessem resultados positivos, ou seja, a acurácia do XGBoost utilizando a Meta-Aprendizagem, tivesse um resultado melhor do que os hiperparâmetros padrões utilizados pelo XGBoost em 86.36% dos casos. Outro ponto que é importante concluir em torno dos resultados, é que a Meta-Aprendizagem visa utilizar a similaridade entre os conjuntos de dados para a recomendação dos hiperparâmetros; com isso, a similaridade dos conjuntos de dados tendiam a dar hiperparâmetros mais efetivos.

Palavras-chave: Meta-aprendizagem, aprendizagem de máquina, custo, ciência de dados, XGBoost.

Abstract

With computional evolution, there was a growth in the number of machine learning algorithms and in parallel, they became more complex and robust. With the growth of this complexity, it was increasingly necessary to focus on the configuration of algorithms, in accordance with their hyperparameters, with the aim of increasing the precision in the result of each one of them: which is not a trivial task. Thus, choosing the most suitable hyperparameters for a given data set can be a costly task both in terms of time and money. Thus, it is necessary that there are faster and more practical ways to find hyperparameters that will set up each algorithm individually. This work aims to use Meta-Learning as a viable solution for the recommendation of hyperparameters for the recent XGBoost machine learning algorithm, in order to reduce computional costs, also aiming at reducing costs for companies. This work used 198 datasets, following the idea of leave-one-out cross-validation for the experiments, thus making each of the datasets tested in the experiment against all others. In addition, were used 3 sets of Meta-Features available in the literature: general, statistical and info-theory for the characterization phase of each one of the datasets to compare the similarity among them. The experimental results attested the success of the application of the heuristics using Meta-Learning for their recommendation, Thus, initially a characterization of the data sets was made using three sets of Meta-Features, so that there was a way to compare the similarity between them and thus apply Meta-Learning to recommend the hyperparameters between the data sets used in the experiments. The results obtained were promising, making that in some cases, 86.36% of the tests had positive results, that is, the accuracy of XGBoost using Meta-Learning, had a better result than the standard hyperparemeters used by XGBoost in 86.36% of cases.

***Keywords**: Meta-learning, XGBoost, recommendation systems*

Lista de Figuras

Conteúdo

1

Introdução

O presente capítulo apresentará uma introdução deste trabalho; primeiramente, mostrando uma motivação para o trabalho, após isso, o objetivo do trabalho e os avanços que ele pode trazer, com o intuito de demonstrar a utilização e a importância da meta-aprendizagem nos dias atuais e o quão importante ela será para o futuro. Por fim, será mostrado a estrutura do documento.

1.1 Motivação

Desde o surgimento dos primeiros computadores juntamente com a *Internet*, houve um grande crescimento de usuários e atualmente há cerca de 4.8 bilhões de usuários conectados a *Internet* [11]. Com isso, há uma grande quantidade de dados que os sistemas têm que coletar e assim, usar com alguma finalidade; como usos industriais, por exemplo, visto que as empresas estão cada vez mais focando em automatizar tarefas e analisar dados com alguma finalidade. Além disso, tornou-se comum no dia a dia dos usuários da *internet*, enfrentarem situações como sugestão de pesquisa ao utilizar a plataforma *Google* ou *Youtube* e serem surpreendidos por recomendações de alguma outra palavra ou vídeos e *sites* que tenham relação com aquilo que pesquisaram. Contudo, é normal também os usuários não refletirem sobre como funciona o mecanismo por trás dos sistemas de recomendação e quão robusto vem se tornando por causa da quantidade de usuários hoje na *internet* como um todo, juntamente com a quantidade de dados que vem sendo gerado dia após dia.

Um ponto que é necessário entender antes de prosseguir é que a Aprendizagem de Máquina vem crescendo todos os dias e está diretamente relacionada com o aumento de informação e além disso, com o aumento tecnológico. Um exemplo clássico de se citar e um dos pontos mais fortes da utilização da Aprendizagem de Máquina hoje, é o fato de pessoas procurarem produtos em lojas físicas e terem algumas dificuldades como: o vendedor achar produtos para aquele

cliente sem ter informação alguma sobre ele ou não ter uma boa heurística de recomendação e o segundo ponto seria que mesmo tendo boas informações sobre o cliente, achar produtos relevantes para ele em meio a tantos outros pode ser uma tarefa altamente custosa e para um ser humano, o número de itens não precisa ser tão grande para que seja uma tarefa difícil.

Por conta do problema citado anteriormente juntamente com a evolução da tecnologia e o número de usuários crescendo na *internet*, foram surgindo mais lojas que fazem seus serviços na própria *internet* e com isso, foi possível utilizar a própria Aprendizagem de Máquina para fazer sugestões ao usuário utilizando compras anteriores, filtros e até mesmo elementos do site que ele tenha acessado. Além disso, um ponto é que um vendedor pode utilizar a questão da experiência. Um vendedor que já tenha visto um cliente várias vezes, já pode saber de forma direta, itens que aquele cliente geralmente gosta de escolher e o mesmo pode ser feito utilizando a meta-aprendizagem; que nada mais é do que utilizar a ideia de experiência no âmbito da computação. Geralmente a meta-aprendizagem é utilizada em problemas de classificação e de regressão e consequentemente, na aprendizagem supervisionada, onde a tarefa mapeia uma entrada para uma saída.

Sendo assim, é necessário entender a importância e a evolução da Aprendizagem de Máquina, que partiu do estudo de reconhecimento de padrões e da teoria de aprendizado computacional na área de inteligência artificial [20]. Com o avanço dos estudos relacionados a Aprendizagem de Máquina, foram surgindo cada vez mais algoritmos e consequentemente, as empresas foram se adaptando e atualizando seus sitemas utilizando algoritmos mais robustos (como as empresas de varejo, por exemplo, para sistemas de recomendação). Com isso, um crescimento ainda maior foi em relação a quantidade de dados, como dito anteriormente, fazendo com que as empresas tivessem que lidar com um número de dados cada vez maior, fazendo com que em paralelo, houvesse um problema de custo computacional relacionado a cada algoritmo quando aplicado a grandes quantidades de dados. Além disso, um dos pontos principais na aplicação de um algoritmo sobre um conjunto de dados, é "configurar"um algoritmo através dos seus hiperparâmetros (visto que com um bom conjunto de hiperparâmetros, o algoritmo pode alcançar um desempenho melhor), e fazer uma busca nos hiperparâmetros para uma boa configuração do algoritmo, se tornou uma tarefa ainda mais desafiadora a depender da grande quandidade de dados.

Com o passar do tempo e o aperfeiçoamento da Aprendizagem de Máquina, foram surgindo novas formas de predizer, explicar relações ou associações, o valor da variável alvo de uma instância, que podem ser divididas em duas grandes cateogiras: regressão e classificação. A análise por regressão surgiu como um método antigo, cujo o método de estimação mais simples é o de mínimos quadrados, proposto por Legendre e Gauss [3, 18] e basicamente analisa um conjunto de processos estatísticos para estimar relações entre uma variável dependente e uma ou mais variáveis independentes, ou seja, a variável dependente depende de como as variáveis independentes são manipuladas. Por fim, há também a análise por classificação que visa identificar a qual conjunto de categorias pertence uma nova observação, ou seja, literalmente classificar uma

nova observação [2].

Dada a ampla quantidade de algoritmos presentes em Aprendizagem de Máquina, há também algoritmos que são focados em análise de classificação, como: *K-Nearest Neighbor*, *Random Forest*, *Neural Networks*, dentre outros [40]. O mesmo acontece para análise de regressão, por exemplo: *Support Vector Machines*, *Decision Trees*, *Linear Regression*, dentre outros [31]. Portanto, recentemente (se comparado aos outros algoritmos) surgiu um projeto aberto chamado *XGBoost* que fornece um framework para algumas linguagens e dentre elas, o python [48]; que é uma linguagem utilizada entre os cientistas de dados [5]. Sendo assim, o *XGBoost* foi um dos algoritmos de Aprendizagem de Máquina mais utilizados na plataforma nos últimos tempos por ser um algoritmo totalmente escalável e utiliza uma estrutura de *Gradient Boosting*, fazendo com que funcione bem tanto para problemas de regressão quanto para problemas de classificação [49]. Basicamente, o XGBoost utiliza uma implementação do *gradient boosting*, que nada mais é do que uma técnica para resolver problemas de regressão e classificação, sendo assim, o pacote do XGBoost inclui um modelo de resolução linear e um algoritmo de aprendizagem em árvore [8].

Como dito anteriormente, atualmente há um grande desafio em achar bons hiperparâmetros para os algoritmos e esse problema ocorre principalmente com o *XGBoost*. Primeiramente é possível citar dois pontos positivos do *XGBoost*: o primeiro devido ao fato de ser um algoritmo bem escalável, já que utiliza um esquema de *Tree Boosting* que será explicado mais a frente. O segundo ponto é por ser um algoritmo flexível, já que por causa de sua ampla quantidade de hiperparâmetros, ele pode ser encaixado em uma gama de conjuntos de dados e problemas de classificação ou regressão. Por ser um algoritmo totalmente escalável e flexível, gera um problema de configuração que faz com que ele tenha uma grande quantidade de hiperparâmetros, já que é possível configurar hiperparâmetros gerais (relacionados a qual *booster* estará sendo utilizado), hiperparâmetros a depender de qual *booster* escolhido, entre outras possibilidades. Além disso, para cada variável dada no hiperparâmetro, há um infinito número de combinações a fim de aumentar a acurácia do algoritmo,o que pode deixar custosa a tarefa de encontrar o hiperparâmetro ideal. [47].

Sendo assim, com o surgimento de um subcampo da Aprendizagem de Máquina chamado Meta-Aprendizagem, foi possível aplicar algoritmos de aprendizagem em Meta-Dados, que nada mais é do que dados sobre outros dados ou informações sobre outros dados, fazendo com que os hiperparâmetros ou algoritmos de Aprendizagem de Máquina não precisassem ser calculados ou testados novamente; sendo uma forma do modelo de Aprendizagem de Máquina aprender a aprender [35]. Com isso, através da Meta-Aprendizagem, é possível gerar uma economia tanto de tempo quanto de complexidade (no quesito de achar novos hiperparâmetros) para empresas ou entidades que precisam utilizar a Aprendizagem de Máquina em grandes conjuntos de dados e principalmente utilizando algoritmos com uma ampla quantidade de hiperparâmetros ou até mesmo, quando precisam checar qual algoritmo utilizar em um certo modelo; a Meta-Aprendizagem é capaz de otimizar ambos os casos.

É possível utilizar a Meta-Aprendizagem como forma de experiência para o modelo; resumidamente é possível utilizar as chamadas Meta-Características para verificar se existem conjuntos de dados com características semelhantes e assim, utilizar cálculos ou resultados que já foram previamente concluídos. Esta solução pode cobrir tanto o problema de encontrar hiperparâmetros quanto para encontrar algoritmos para um dado modelo. O foco deste trabalho é encontrar hiperparâmetros mas é interessante também citar o problema de encontrar algoritmos de Aprendizagem de Máquina visto que esse é um dos outros grandes problemas clássicos da literatura, por hoje em dia existir uma ampla quantidade de algoritmos de Aprendizagem de Máquina que resolvem o mesmo problema e para cada um deles, uma gama de hiperparâmetros e combinações dos mesmos.

Por fim, tendo em vista o foco na busca por melhores hiperparâmetros, hoje em dia a literatura foca em uma forma de busca por hiperparâmetros e existem algumas formas; uma delas é o *GridSearch* que visa a busca exaustiva dos hiperparâmetros. Como citado anteriormente, isso pode ter um grande custo de tempo e dinheiro para as empresas. Sendo assim, a Meta-Aprendizagem visa reduzir esses custos, fazendo com que haja uma chance menor de ter que calcular uma ampla quantidade de hiperparâmetros novamente. Além deste último ponto positivo no que condiz a utilização da Meta-Aprendizagem, um outro ponto importante citar é a motivação da utilização do *XGBoost* junto a Meta-Aprendizagem. Além disso, a quantidade de materiais, principalmente no que condiz ao tema deste trabalho (Meta-Aprendizagem e recomendação de hiperparâmetros) para o XGBoost é menor em relação aos algoritmos mas antigos. Com isso, foi notado a importância e relevância de focar no *XGBoost* para fazer o estudo da Meta-Aprendizagem juntamente com a recomendação de hiperparâmetros. Uma outra justificativa que pode ser ressaltada, é explicar o porquê de utilizar a recomendação de hiperparâmetros. Um ponto é que na literatura há bastante materiais sobre a recomendação utilizando a Meta-Aprendizagem, portanto, utilizando como exemplo um dos livros bases para o assunto [6] são mostrados exemplos que visam a recomendação dos próprios algoritmos de Aprendizagem de Máquina. Um dos pontos mais influenciadores da escolha de recomendar os hiperparâmetros ao invés de algoritmos, foi de focar no algoritmo *XGBoost* e assim, a melhor opção seria focar em recomendar os próprios hiperparâmetros.

1.2 Objetivo

Este trabalho tem por finalidade utilizar a Meta-Aprendizagem no algoritmo XGBoost para verificar se o seu uso é efetivo na recomendação de hiperparâmetros para novos conjuntos de dados; fazendo com que não seja necessário calcular os hiperparâmetros completamente, ou seja, verificar se os hiperparâmetros recomendados usando a Meta-Aprendizagem são tão bons ou melhores quando comparados com os hiperparâmetros padrões do *XGBoost*.

Com isso, propõe-se como objetivos específicos:

- Estabelecer conjuntos de dados para os experimentos, fazendo com que haja um número significativo de conjuntos de dados visando alcançar uma quantidade de resultados significativos;
- Realizar um estudo em torno da Meta-Aprendizagem, utilizando técnicas da literatura para a construção do sistema para a recomendação dos hiperparâmetros;
- Conhecer e comparar diferentes formas de utilização da Meta-Aprendizagem, como o uso do sistema de recomendação de algoritmos como base para o sistema de recomendação dos hiperparâmetros, por exemplo;
- Realizar um estudo de métricas de disimilaridade e similaridade, checar abordagens e técnicas utilizadas;
- Realizar um estudo acerca de otimização de hiperparâmetros de algoritmos de aprendizagem de máquina;
- Avaliar os resultados de cada hiperparâmetro afim de possibilitar um sistema para a recomendação dos mesmos;
- Estabelecer formas de recomendação dos hiperparâmetros baseados nos experimentos.

1.3 Estrutura do Documento

Nesta seção será mostrada a organização desta dissertação. Após este capítulo de introdução, o trabalho foi dividido em cinco capítulos.

O Capítulo 2 apresenta a revisão de literatura, mostrando os principais tópicos em torno da aprendizagem de máquina, dando uma demonstração e descrição sobre o XGBoost, algoritmo utilizado neste presente trabalho; além disso, detalhando sobre os hiperparâmetros utilizados no algoritmo para os experimentos. Será mostrado também um breve comentário sobre métricas de similaridade e dissimilaridade, principalmente sobre a distância euclidiana, que foi a escolhida para calcular a dissimilaridade entre os conjuntos de dados. São apresentados tópicos atuais e explicações relacionadas do estado da arte com o presente trabalho.

O Capítulo 3 mostra de forma mais aprofundada sobre a meta-aprendizagem, dando uma visão geral sobre e mostrando principalmente, o esquema utilizado neste trabalho. Além disso, também é descrito sobre as Meta-Características, os conjuntos e quais foram utilizadas neste trabalho.

No Capítulo 4, é detalhado o desenvolvimento do experimento dando uma introdução do que foi feito, o foco e após isso, uma descrição sobre os 198 conjuntos de dados utilizados. É descrito também como se deu a busca por hiperparâmetros ideias para que eles servissem

de referência para cada um dos conjuntos de dados, metodologia, execução do XGBoost, extração das Meta-Características, aplicação da técnica de dissimilaridade e comparação com os *baselines*.

Já no capítulo 5, serão mostrados os resultados que foram obtidos com o decorrer do experimento. Neste capítulo serão apresentados os resultados para os 4 experimentos feitos utilizando os conjuntos de Meta-Características citados mais adiante e os 2 *baselines* que serviram de referência para a comparação.

Por fim, no capítulo 6 será dada uma conclusão deste trabalho, mostrando um pouco mais sobre resultados e contribuições, mostrando limitações do trabalho e trabalhos futuros.

2

Revisão da Literatura

Este capítulo tem por objetivo apresentar uma revisão da literatura à respeito da Meta-Aprendizagem em conjunto com a aprendizagem de máquina e métricas de dissimilaridade. Primeiramente será apresentado um conceito de Meta-Aprendizagem, sua importância de acordo com a evolução da computação e os pontos positivos que ela pode trazer em relação à aprendizagem de máquina. Em seguida, será introduzido os princípios básicos de aprendizagem de máquina e será mostrado um estudo mais afundo sobre o XGBoost, que é o algoritmo foco deste trabalho. Por fim, serão mostradas as técnicas de dissimilaridades para a comparação das Meta-Características, que serão explicadas mais detalhadamente e aplicadas nos experimentos.

2.1 Introdução

Uma relação que pode ser feita com a Meta-Aprendizagem, é o uso indireto da programação dinâmica. Basicamente, a programação dinâmica funciona de forma que um algoritmo não precise calcular novamente alguns valores que já foram calculados em outros momentos e isso pode ser abstraído para outros campos de outras formas. Um exemplo de uso que pode ser relacionado e principalmente por conta do crescimento computacional, seria o fato de que usuários, estudiosos e empresas, utilizam bastantes conjuntos de dados públicos (principalmente vindo da plataforma *Kaggle*) e isso faz com que pessoas diferentes possam utilizar os mesmos conjuntos de dados várias vezes e devido a grande quantidade de conjuntos de dados existentes, também pode acontecer de que haja alguns conjuntos de dados com características semelhantes; sendo assim, os conjuntos de dados podem ser caracterizados e detalhados por Meta-Características fazendo com que algoritmos de aprendizagem de máquina já calibrados, não precisem ser calibrados novamentes do zero (já que usuários podem utilizar resultados que já foram calculados em outros conjuntos de dados semelhantes ou até naquele mesmo em que ele trabalha).

Por conta da enorme quantidades de dados que existe hoje e principalmente pelo fato de continuar crescendo cada vez mais, surge a necessidade de utilizar este tipo de abordagem para

que não seja necessário calcular novamente hiperparâmetros para a calibragem dos algoritmos, principalmente quando o algoritmo requer um custo computacional maior ou quando o conjunto de dados é ligeiramente grande (o que torna algo ainda pior juntando estes dois problemas, custo computacional e o tamanho do conjunto de dados), o que é o caso do XGBoost aplicado à grandes conjuntos de dados, por exemplo.

Sendo assim, é necessário entender alguns pontos antes do aprofundamento da Meta-Aprendizagem. O primeiro ponto é que a Meta-Aprendizagem é utilizado em conjunto com a aprendizagem de máquina, os dois são relacionados. Com isso, é necessário entender a história da aprendizagem de máquina, como ela evoluiu nos últimos anos e sua importância, já que o algoritmo utilizado para a Meta-Aprendizagem aqui, vem da aprendizagem de máquina, o XGBoost. Além disso, para este trabalho, é necessário o estudo de algumas métricas de dissimilaridade, já que é necessário utilizá-las na Meta-Aprendizagem, para saber os conjuntos de dados que são semelhantes entre si.

2.2 Aprendizagem de Máquina

Pode-se dizer que aprendizagem de máquina (ou *machine learning*) é um método de analisar dados que de forma automática, constrói modelos e assim, faz aprender por seus erros e também previsões sobre seus dados [32]. De 1959 aos dias atuais, a computação foi evoluindo, o investimento em pesquisas e principalmente na área de inteligência artificial foi crescendo e a aprendizagem de máquina, foi ganhando seu espaço. Uma maneira de definir a aprendizagem de máquina, seria: "campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados“. [37].

Basicamente, o aprendizado de máquina é dividido em 3 categorias: aprendizado supervisionado, não supervisionado e por reforço. Cada um tem suas vantagens e desvantagens, portanto, o aprendizado supervisionado é o foco neste trabalho já que o XGBoost está sendo utilizado e ele é um algoritmo supervisionado (tanto para classificação quanto para regressão).

No aprendizado supervisionado, são utilizados conjuntos de dados para treinar e nos próprios dados, contém a resposta desejada. Resumidamente, constrói um modelo matemático de um conjunto de dados contendo as entradas e as saídas [34]. Exemplos de técnicas para problemas utilizando a categoria supervisionada são: árvores de decisão, k-vizinhos mais próximos e o próprio XGBoost. Na aprendizagem supervisionada, cada amostra consiste em um par onde o primeiro elemento é uma entrada (que geralmente é um vetor) e uma saída desejada. Na figura 2.1, é possível visualizar uma amostra do conjunto de dados Iris, uns dos conjuntos de dados mais comuns referente ao estudo de aprendizagem de máquina. É possível notar que as primeiras 4 colunas, são referentes ao vetor de entrada (*Features*) e a última coluna é referente à saída (*Labels*); com isso, os algoritmos supervisionados podem utilizar esses vetores de entradas e os resultados da saída, para o treinamento e aperfeiçoamento do algoritmo.

Features				Labels
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica

Figura 2.1: Tabela do dataset Iris [10]

De maneira geral, é necessário seguir alguns passos padrões para resolver problemas com a Aprendizagem Supervisionada, a imagem 2.2 mostra passo a passo de como funciona a estrutura básica da Aprendizagem Supervisionada.

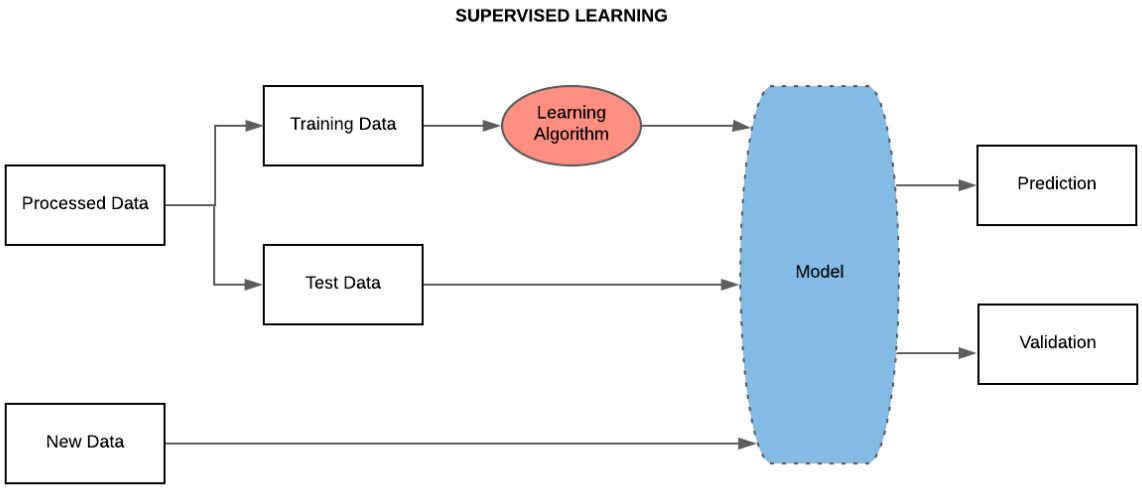


Figura 2.2: Exemplo do funcionamento básico da Aprendizagem Supervisionada

Utilizando a imagem 2.2 como exemplo, é possível visualizar que uma das primeiras partes da Aprendizagem Supervisionada, é o processamento dos dados, logo após, é necessário separar os dados em dados de treino e dados de teste, fazendo assim, com que seja possível construir um modelo e assim, executar a validação e a predição. Na literatura essa separação é denominada de *holdout method*.

Para este presente trabalho, foi necessário utilizar a Aprendizagem Supervisionada para o

XGBoost então foram utilizados os seguintes passos:

- Determinar o tipo dos dados para o treino, no caso do XGBoost, é necessário que todos os dados sejam dados reais, visto que o XGBoost não aceita dados em formato de strings, devem ser transformados em dados reais. Uma solução para isso, é separar em duas colunas diferentes, por exemplo: há uma coluna no conjunto de dados chamada "gênero"e os valores são "Masculino"e "Feminino"; uma solução para isso é separar em duas colunas, onde o valor de cada uma é "1"para caso o valor "masculino"esteja naquele vetor e "0"caso contrário;
- Fazer um tratamento dos dados, tirando todos os dados que podem atrapalhar no treinamento do algoritmo (removendo *outliers*). Os dados precisam ser representativos;
- Determinar a estrutura da função de aprendizagem correspondente ao algoritmo utilizado, no caso deste trabalho, o XGBoost;
- Executar o algoritmo escolhido no conjunto de dados de treino e no conjunto de dados de teste para avaliar a acurácia;
- Utilizar uma validação para os resultados, um exemplo é o *k fold cross-validation*, que foi utilizado neste trabalho e visa avaliar a capacidade de generalização dos modelos;
- Checar a acurácia da meta-aprendizagem em cada um dos conjuntos de dados.

Um outro algoritmo que importante ressaltar também, é o *k Nearest Neighbors (kNN)*, já que neste trabalho, foi utilizado em conjunto com as métricas de dissimilaridade para a recomendação utilizando a Meta-Aprendizagem (detalhado no capítulo 4). O kNN é um dos mais importantes algoritmos para classificar e funciona calculando a distância do conjunto de dados (no caso deste trabalho) a cada um dos exemplos na base de dados; ou seja, que está na base do meta-aprendiz. Após a comparação, verifica a qual classe pertence os K mais similares. No caso deste trabalho para fins de experimentos, o K utilizado foi dinâmico, ou seja, não foi utilizado um K estático para todo o trabalho, e sim foram comparados os resultados utilzando o K de 1 a 10.

Com isso, ao contrário do supervisionado, o não supervisionado parte do princípio de não ter por onde "começar“ no algoritmo e ele começa ingênuo, ou seja, vai aprendendo aos poucos. Um exemplo de uso do aprendizado não supervisionado, seriam os sistemas de recomendação; que é um dos focos deste trabalho. Nos sistemas de recomendação, geralmente o sistema não tem nenhuma referência do usuário quando ele começa a utilizar o sistema, um exemplo disso é no sistema de gerenciamento de filmes *Netflix*. No sistema da *Netflix*, quando o usuário se cadastra, não se tem referência alguma dos filmes que possam interessar àquele usuário, então o que acontece é: na medida que o usuário vai assistindo filmes, o sistema "aprende“ e analisa

quais filmes seriam interessantes para o usuário e isso faz com que o aprendizado não seja supervisionado.

Por fim, há a aprendizagem por reforço, que basicamente se ajusta baseado em *feedbacks*, sejam eles premiações ou punições. Algo importante de falar sobre o aprendizado por reforço é que ele utiliza técnicas de programação dinâmica [42] (uma técnica que auxilia na redução da complexidade assintótica de algoritmos; basicamente reutiliza respostas anteriores para não precisar calcular novamente).

2.2.1 Aprendizagem supervisionada

Como citado anteriormente, o algoritmo utilizado no trabalho foi o XGBoost, devido ao fato de ser um algoritmo relativamente novo e ter ganho cada vez mais espaço nas plataformas de Ciência de Dados. O *XGBoost* é focado na aprendizagem supervisionada, portanto, é necessário aprofundar um pouco mais o entendimento acerca da aprendizagem supervisionada. Basicamente o exemplo de ser um supervisor na aprendizagem é ser um elemento que detém o conhecimento do domínio e assim, fornece exemplos na forma de pares de entrada e saída [14] e o algoritmo vai nada mais do que aprender a relação entre a entrada e a saída de forma consistente.

Como explicado anteriormente, a aprendizagem supervisionada precisa aprender para dar alguma saída. A forma de aprendizagem pode ser dada utilizando como exemplo um conjunto de *N* conjuntos de dados onde pode-se dizer que:

$$Y = T(x_i^1, x_i^2, ..., x_i^n)$$

Utilizando a equação anterior de exemplo a variável resposta Y (contendo a i-ésima informação), que será igual a função T, mapeia as variáves observarsionais x, onde i vai de 1 até n (que é o número de observações). Além disso, é importante ressaltar a diferença entre "n"e "N". Como já mostrado anteriormente, o "n“ minúsculo é o número de observações e o “N“ maiúsculo, é o número de conjunto de dados, que no caso deste trabalho, são 198. Com isso, faz com que seja um dos passos principais da aprendizagem supervisionada, também conhecido como treinamento.

Com isso, induzir um classificador através de um conjunto de dados é também visto como um problema de busca, onde é necessário encontrar a hipótese (entre todas que o algoritmo de aprendizagem de máquina é capaz de gerar), com a melhor capacidade. Além disso, um dos termos que são importantes falar quando relacionado a aprendizagem supervisionada, é o viés. Normalmente, várias hipóteses conseguem modelar bem e assim, é necessário uma espécie de viés para o processo de busca da hipótese. O termo busca um critério de preferência de uma hipótese em relação a outra, portanto que ambas sejam consistentes com os exemplos [28].

Aprofundando um pouco mais sobre o viés de aprendizagem, o mesmo quando é incosistente ou os conjuntos de dados com pouca representatividade ou poucos dados, podem afetar o

classificador fazendo com que sua acurácia na predição diminua. Há dois problemas comuns na literatura quando relacionado ao processo de aprendizagem: *overfitting* e o *underfitting* [29]. O primeiro, *overfitting*, como o próprio nome diz, é uma espécie de superajuste e ele ocorre quando o modelo se ajustou muito bem aos dados que estão sendo utilizados de treino; parece ser algo bom em primeira vista mas este tipo de problema faz com que ele não generalize bem para novos dados (ou também nos conjuntos de teste). Através da imagem 2.3 é possível visualizar que em alguns momentos o algoritmo tem um desempenho muito melhor do que a média. Basicamente faz com que o modelo decore o conjunto de dados que foi utilizado como treino e não realmente aprendeu a diferença entre os dados para utilizar o aprendizado em novos testes. O segundo, *underfitting*, ao contrário do *overfitting*, não se adapta bem nem com os dados que foram treinados, fazendo com que tenha um resultado ruim em ambos os casos, também utilizando a figura 2.3 como examplo, é possível visualizar quando o algoritmo teve um desempenho bem abaixo da média. Isso geralmente acontece quando os dados não têm informações suficiente.

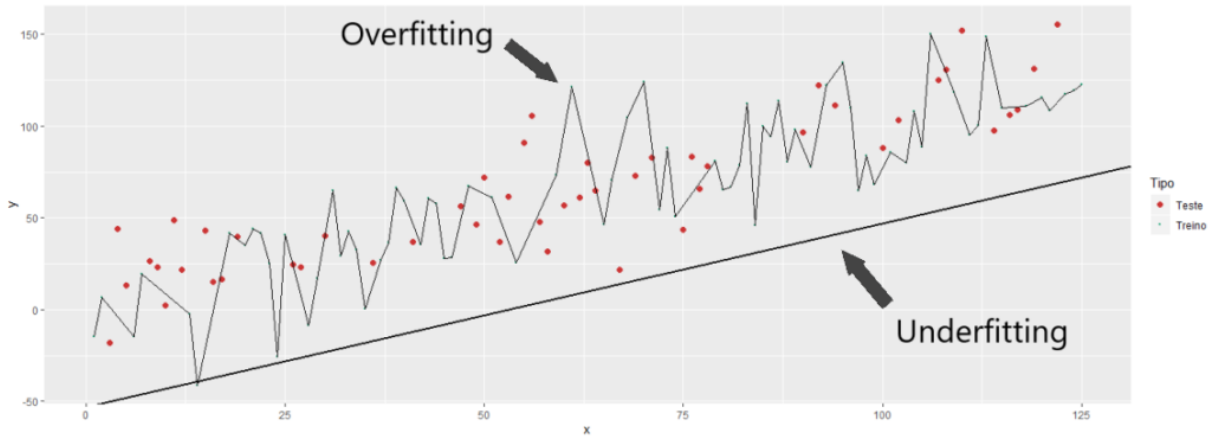


Figura 2.3: Exemplo de *overfitting* e *underfitting* [41]

2.2.2 XGBoost

Além da base da aprendizagem de máquina, para este trabalho é necessário também entender como funciona o XGBoost incluindo suas vantagens e desvantagens em relação aos outros algoritmos de aprendizagem de máquina, principalmente, supervisionados.

Para um melhor entendimento: o XGBoost é uma biblioteca *open-source* que implementa algoritmos de aprendizagem de máquina que funcionam sobre um framework de *gradient boosting* [45]. Basicamente o *gradient boosting* é uma técnica de aprendizagem de máquina utilizada para problemas de regressão e classificação, na qual produzem um modelo de prediçao na forma de árvores de decisão, surgindo a partir da observação de Leo Breiman, na qual acreditava que o *boosting*, pode ser interpretado como um algoritmo de otimização em uma função de custo adequada [7]. Sendo assim, o XGBoost provê uma otimização em árvore que funciona de forma paralela fazendo com que resolva vários problemas de ciência de dados de um jeito preciso e rápido, dando assim, flexibilidade, eficiência e portabilidade [45]; fazendo com que seja um dos motivos de ser um dos algoritmos mais queridos nas competições que acontecem na plataforma *Kaggle*, por exemplo.

Para entender um pouco mais a fundo a história do XGBoost, ele surgiu a partir de um projeto de pesquisa feito por Tianqi Chen [8], utilizando árvores de decisão com o *boosted gradient*, dispõe de algumas vantagens quando comparado a outros algoritmos de aprendizagem de máquina. Como citado anteriormente, uma das principais vantagens é o processamento paralelo (quando citado que utiliza uma otimização em árvore paralela).

Além disso há também a regularização que ajuda o algoritmo a reduzir a chance de ter um *overfitting*. Basicamente reduzindo o erro já que quando o modelo tem uma baixa acurácia, pode causar o *overfitting*, já que o modelo pode tentar capturar os ruídos do conjunto de dados de treino [44].

Como terceira vantagem, há uma grande flexbilidade em ajustar hiperparâmetros tais como funções objetivo (que seriam os ajustes das funções de perda, por exemplo). Antes de rodar o XGBoost, é necessário setar 3 tipos de parâmetros: *general parameters*, *booster parameters* e *task parameters* [47].

O primeiro parâmetro *general*, define o XGBoost de forma geral e é importante entender alguns hiperparâmetros utilizados neste trabalho e o foco será a explicação de hiperparâmetros escolhidos. Uma das principais funções é definir qual tipo de *boosting* será utilizado: árvore ou linear. Para este trabalho, foi utilizado o modelo em árvore, já que um dos diferenciais do XGBoost é ter este tipo de modelo. Para um breve entendimento sobre o modelo; a priori é importante entender como funciona um modelo baseado em conjunto de árvores de decisão. Um bom exemplo citado na documentação do próprio XGBoost [46], é o mostrado na figura 2.4:

Basicamente funciona de forma que os membros das famílias são definidos em diferentes folhas da árvore onde cada folha possui uma pontuação. É importante ressaltar que o nome

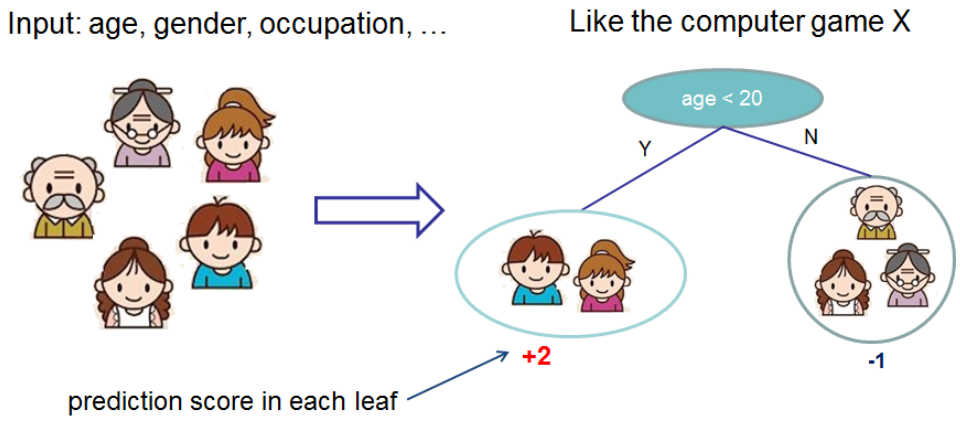


Figura 2.4: Exemplo de árvore de decisão [46]

na literatura desse tipo de árvore de decisão utilizado no caso do XGBoost é *classification and regression trees (CART)*, que foi inicialmente introduzido por Breiman [39], que pode ter diferenças quando utilizado em classificação ou regressão, como no jeito de dividir a árvore, por exemplo [39]; é importante entender este breve conceito já que o XGBoost consegue ser utilizado tanto para regressão quanto para classifação e é importante também, entender os conceitos básicos sobre árvore de decisão e *CART* para entender por fim, como funciona o *Tree Boosting* utilizado pelo XGBoost. Para este tipo de árvore, a cada folha é atribuído um valor real que pode dar mais informações e interpretações para com a classificação [46].

É importante um maior detalhamento em relação ao CART, que seria a base do funcionamento do algoritmo. Basicamente consiste em um conjunto de árvores de classificação e regressão. Além disso, o CART pode ser representado como uma estrutura de dados chamada de árvore binária. Onde cada nó da árvore pode ter até dois filhos. Com isso, a criação de um modelo que visa utilizar este tipo de árvore, é importante selecionar variáveis e dividir os pontos da árvore até que uma estrutura adequada seja construída, ou seja, que detalhe bem o modelo dos dados. Além disso, neste trabalho foi focado na utilização dos algoritmos para a classificação, sendo assim, quando o CART é utilizado como uma árvore de classificação, é necessário calcular um coeficiente GINI, que é uma medida de desigualdade e no CART, é utilizada para definir as divisões da árvore. Um detalhe que é possível citar aqui, é que também há a opção de utilizar como uma árvore de regressão, como citado anteriormente; quando isto acontece, a divisão da árvore é baseada no cálculo de variância mínima. [46]

Sendo assim, após entender a utilização do *CART*, é importante ressaltar que uma única árvore pode não ser suficiente para interpretar e detalhar o modelo; então o XGBoost foca em um modelo de conjuntos, que soma a predição de múltiplas árvores, como mostrado na imagem 2.5. [46]:

Matematicamente, é possível escrever esse modelo de árvore da seguinte forma:

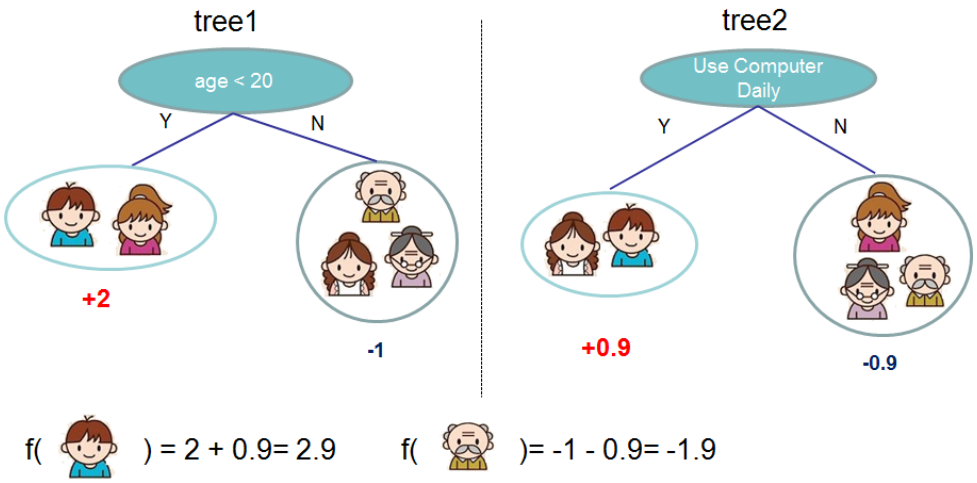


Figura 2.5: Exemplo do conjunto de árvores [46]

$$y = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

Onde K é o número de árvores, f é uma função no espaço F e o espaço F é o conjunto de todos os *CARTs* possíveis. Após entender a base do modelo utilizando um conjunto de *CARTs*, é possível prosseguir para o entedimento do *Tree Boosting*, que é a expressão final utilizada pelo XGBoost. Sendo assim, é necessário utilizar de uma função objetivo, otimizar, para que a aprendizagem supervisionada funcione uilitzando as árvores, fazendo assim, com que o *Tree boosting* funcione no XGBoost.

Como um dos hiperparâmetros gerais utilizados para este trabalho, foi o modelo baseado no *Tree Booster*, é necessário passar algumas configurações que também são passadas por hiperparâmetros no XGBoost. Por padrão há dezenas de hiperparâmetros a serem configurados mas neste trabalho só serão explicados os hiperparâmetros que foram utilizados no experimento.

Após um melhor entendimento em relação ao coeficiente GINI e o anteriormente citado, *Gradient Boost* é possível detalhar melhor o XGBoost. Uma das vantagens é que ele pode ser paralelizado, quando utilizado em uma *CPU Multi-Threaded*, e essa configuração é possível utilizando seus próprios hiperparâmetros [47]. Essa paralelização faz com que seja possível melhorar o *Gradient Boosting* controlando melhor a complexidade do modelo.

Alguns dos hiperparâmetros mais utilizados que pertencem aos parâmetros do *Tree Boosting*, é o *max depth*, *min child weight* e *gamma*. Estes 3 hiperparâmetros foram utilizados já que eles podem diretamente controlar a complexidade do modelo e podem ajudar a controlar um possível *overfitting*, que pode acontecer quando há uma alta acurácia no conjunto de dados de treino e uma baixa acurácia no conjunto de dados de teste [50]. O segundo jeito é configurando os hiperparâmetros *subsample* e *colsample bytree*, eles fazem com que seja adicionado uma espécie de aleatoriedade para fazer com que o treino do algoritmo seja mais resistente a ruidos [50]. Além disso, há alguns hiperparâmetros focados em aumentar a performance e a

velocidade do treino como o *tree method*, *hist* e *gpu hist* e também lidar com conjunto de dados desbalanceados, mas não foram utilizados visto que a importância aqui era apenas mostrar a acurácia do XGBoost de forma geral, utilizando a Meta-Aprendizagem. Sendo assim, é importante entender brevemente como funciona cada hiperparâmetro utilizado para o *Tree Booster*, de acordo com a documentação do próprio XGBoost. Nas próximas subseções, serão explicados brevemente os hiperparâmetros utilizados neste trabalho.

Max depth

O primeiro hiperparâmetro e um dos mais utilizados é o *max depth*, que informa ao algoritmo a profundidade máxima de cada árvore. Basicamente quanto maior o valor, mais complexo se torna o modelo e está mais propenso a sofrer um *overfitting*, como citado anteriormente. Da mesma forma que quanto menor o valor, pode causar um *underfitting*. Por padrão, o XGBoost define o valor da profundidade da árvore em 6 e é aceitado um intervalo de 0 a infinito, onde o 0 pode ser setado apenas quando outros hiperparâmetros em específico, estão sendo utilizados.

Min child weight

O segundo hiperparâmetro utilizado é o *min child weight*, ele informa a soma mínima do peso da instância hessiana que é necessária em um filho na árvore. Basicamente ele pode dizer para parar de dividir a árvore uma vez que o tamanho da amostra em um vértice fica abaixo do limite. Basicamente, se no momento da partição da árvore resulta em uma folha com a soma das instâncias dos pesos menor que o valor passado para o parâmetro *min child weight*, o processo de construção vai evitar um particionamento posterior. De forma um pouco diferente, na regressão ele corresponde ao mínimo número de instâncias que precistam estar em cada vértice.

Gamma

O terceiro hiperparâmetro é o *gamma* e ele é utilizado em conjunto com o *min child weight* e o *max depth*, os 3 regularizam informações da árvore, como dito anteriormente. Em paralelo aos outros dois hiperparâmetros citados anteriormente, o *gamma* trabalha por regularizar o *overfitting* e também é conhecido por *min split loss*, servindo assim para representar o quanto de perda tem que ser reduzido quando for considerado uma partição na árvore. É um dos hiperparâmetros que mais dependem da configuração de outros hiperparâmetros, então basicamente ele não consegue tanto impacto quando modificado sozinho e recomendam utilizar o seu valor mais alto quando á uma alta profundidade na ávore do XGBoost.

Subsample

O penúltimo hiperparâmetro configurado foi o *Subsample*, que basicamente funciona como uma razão das instâncias de treinamento. De acordo com a documentação, foi definido que se ajustado o valor 0.5, o XGBoost aleatoriza a amostra em metade dos dados antes de crescer as árvores do modelo. Isso poderia evitar um *overfitting* e essa atividade de *subsampling* ocorrerá uma vez durante toda iteração de *boosting* do algoritmo.

Colsample by tree

Por fim, o último hiperparâmetro utilizado foi o *colsample by tree*. Como dito anteriormente na explicação do *Boosting Tree* utilizado pelo XGBoost, o mesmo constrói multriplas árvores para fazer as predições. Basicamente o *colsample by tree* define a porcentagem de colunas que serão utilizados para construir cada árvore e de acordo com a documentação, isso ocorre uma vez para toda a árvore construída.

2.3 Métrica de dissimilaridade

Para que haja uma comparação entre os conjuntos de dados, é necessário que haja um cálculo de similaridade ou dissimilaridade entre eles para que, baseado em um conjunto de dados, seja encontrado algum outro conjunto de dados semelhante à esse. Portanto, o conceito de similaridade e dissimilaridade precisa ser algo mensurável e quantitativo [23]. Além do mais, é importante também falar sobre similaridade, porque também é uma alternativa além da dissimilaridade. A diferença é que para a dissimilaridade, quanto menor o valor entre os dois conjuntos de dados, mais similar eles são; a similaridade funciona de forma contrária, quanto maior a distância, mais similar.

Antes de prosseguir, é importante também ressaltar que na literatura, há várias outras métricas ou funções de distância, como: Minkowsky, Quadrática, Correlação, Chi-Quadrado. Qualquer função de distância que seja possível mapear os valores das meta-características, podem ser aplicadas neste tipo de trabalho. Algumas famosas que também entregam bons resultados na literatura são a similaridade por cosseno e Pearson. A distância euclidiana simples foi escolhida para este trabalho por também apresentar bons resultados e ter uma aplicação mais simples e intuitiva do que as outras funções. Mas é importante mostrar também que outras métricas de similaridade e dissimilaridade podem ser aplicadas. Além disso, foi possível visualizar melhor o mapeamento das Meta-Características e seus respectivos valores na distância euclidiana, tendo assim, um melhor controle ao executar os experimentos.

As métricas utilizadas para o cálculo de dissimilaridade e similaridade, são cálculos de distâncias conhecidos na matemática; sendo assim, são necessárias informações numéricas sobre as características de cada item, um exemplo podem ser os vetores utilizados no conjunto de dados deste trabalho, onde o conjunto de Meta-Características (que será detalhado mais a frente)

formarão um ponto em um espaço de N dimensões, onde o número de dimensões é dado pelo número de Meta-Características.

Sendo assim, uma função distância nada mais é do que uma fórmula matemática usada para métricas de distância [30]. No presente trabalho, foi utilizado a distância euclidiana para calcular a distância entre os conjuntos de dados.

2.3.1 Distância euclidiana

No caso da distância euclidiana, seria um cálculo mais simples por ser apenas a distância em linha reta entre o item A e o item B, do ponto de vista da Geometria, a linha reta que liga os dois pontos em um espaço vetorial. Além disso, a distância euclidiana também pode ser usada para n-dimensões. Formalmente, a dissimilaridade entre dois itens é dado pela distância no espaço vetorial entre o ponto "a", onde o ponto $a = (a_1, a_2, \ldots, a_n)$, e o ponto "b", $b = (b_1, b_2, \ldots, b_n)$, como mostra a figura 2.6.

$$D(a,b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$

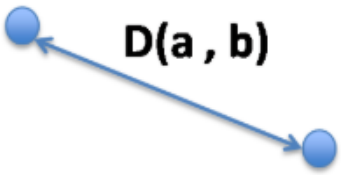


Figura 2.6: Cálculo da distância euclidiana [16]

A vantagem de utilizar a distância euclidiana se dá quando os vetores estão normalizados. Portanto, como a fórmula condiz com a distância, é necessário transformar em uma relação de dissimilaridade, então a equação final fica na forma:

$$1 - \sqrt{(\sum_{i=1}^{n} (bi - ai)^2)}$$

3

Meta-Aprendizagem

Nesta seção será mostrada de forma mais aprofundada sobre a Meta-Aprendizagem, que é um dos focos principais deste trabalho. Primeiramente, será introduzida uma visão geral sobre Meta-Aprendizagem e logo após, o modelo de Meta-Aprendizagem focado para este trabalho. Além disso, será explicado como foi utilizada a Meta-Aprendizagem juntamente com os dados usados neste trabalho e como foi utilizada a "experiência"que a Meta-Aprendizagem fornece.

3.1 Visão geral

Nesta seção, será dada uma visão geral em torno da Meta-Aprendizagem e aprofundará na importância de utilizar a "experiência", assim como é utilizada nos seres humanos, só que no ramo da computação. Além disso, será explicado como a experiência que a Meta-Aprendizagem provê, foi utilizada na utilização de novos hiperparâmetros.

Basicamente, a Meta-Aprendizagem provê aos algoritmos de aprendizagem de máquina, uma forma de ganhar experiência após alguns episódios de aprendizagem [43], ou seja, após a execução do algoritmo sobre algumas bases de dados; fazendo assim, com que a Meta-Aprendizagem obtenha experiência sobre um determinado tipo de conjunto de dados, por exemplo. Sendo assim, esse ganho de experiência pode ser interpretado por selecionar modelos que já foram previamente otimizados ou "tunados", de acordo com a literatura, e utilizá-los em novos problemas. Um ponto importante antes do aprofundamento da Meta-Aprendizagem, é entender o conceito de: meta. A palavra "meta"significa resumidamente: um nível acima de abstração. Um termo famoso é o uso dos "meta-dados", que nada mais é do que dados sobre outros dados. Agora com o entendimento do uso do "meta", a "Meta-Aprendizagem"nada mais é do que aprender sobre o aprendizado, ou seja, a experiência. Além disso, um dos objetivos da Meta-Aprendizagem, é entender como o processo de aprendizagem se torna flexível de acordo com o objetivo de cada tarefa. Os algoritmos de aprendizagem de máquina assim como seus hi-

perparâmetros, funcionam se adaptando a algum ambiente, sendo assim, a Meta-Aprendizagem pode funcionar diretamente em cima disso.

De maneira geral, o meta-aprendizado é relacionado aos próprios processos de aprendizagem. Um dos principais pontos da Meta-Aprendizagem, foi utilizado por John Biggs, que defendia o ponto de defender um estado que assumisse o controle da própria aprendizagem [4]. No conceito da própria psicologia e no primeiro significado da palavra Meta-Aprendizagem, é possível definir como se fosse uma conscientização de uma pessoa em relação a compreensão da aprendizagem de acordom com a experiência de cada pessoa. No contexto da computação, com o passar do tempo, tornou-se um subcampo da área de aprendizagem de máquina onde foi aplicado esse meta-conhecimento em cima de meta-dados, como citado anteriormente, sendo assim, tendo seu principal objetivo o: entender o processo de aprendizagem automática e o quão flexível poderia ser na hora de resolver alguns problemas de aprendizagem. Neste problema é possível visualizar um exemplo deste problema; já que essa flexibilidade, por exemplo tem que ser analisada sucintamente.

Utilizando da breve explicação anterior sobre a experiência e flexbilidade da Meta-Aprendizagem, é possível visualizar essas duas características neste trabalho. A experiência condiz com os conjuntos de dados que ficam no meta-aprendiz (que será explicado mais a frente), utilizadas como experiência. Ou seja, sempre que entrar um conjunto de dados novo e que seja necessário procurar novos hiperparâmetros para aquele conjunto de dados novo, é necessário apenas checar se já existe algum conjunto de dados com características semelhantes. A flexbilidade condiz com o fator de que: o que pode ser utilizado dos conjuntos de dados tratados anteriormente. Ou seja, os hiperparâmetros. Essa flexbilidade é importante já que cada algoritmo ou conjunto de dados, funcionam de forma diferente com cada um dos hiperparâmetros.

De acordo com a literatura, uma das propostas para a definição de Meta-Aprendizagem é a seguinte [19]:

- O sistema deve incluir um subistema de aprendizagem e além disso, a própria Meta-Aprendizagem estuda como aumentar a eficiência através da experiência, visando entender como o aprendizado pode ser flexível de acordo com o domínio, no caso deste trabalho, os conjuntos de dados.

- A experiência é ganha através da exploração dos meta-conhecimentos extraídos, sejam em processos de aprendizados anteriores ou algum episódio que tenha acontecido em algum conjunto de dados, como é o caso deste trabalho, onde são encontradas as melhores soluções para cada um a fim de experimentos. Além disso, o conhecimento pode ser extraído em diferentes domínios.

- O viés de aprendizagem é escolhido dinamicamente. Basicamente o viés de aprendizagem pode ser individualizado por cada algoritmo, como o treino dos dados e utilizado em conjunto com a Meta-Aprendizagem. Nos experimentos o XGBoost foi utilizado

como classificador, então o próprio algoritmo dispõe de um sistema de aprendizagem, separando em conjuntos de treinos.

- A Meta-Aprendizagem monitora o processo automatico de aprendizagem, de acordo com o contexto de problemas de aprendizagem. Em projetos utilizando a Meta-Aprendizagem, o ideal é que sempre que um novo modelo entrar no meta-aprendiz, o sistema saiba utilizar essa nova experiência, de uma maneira eficiente.

Sendo assim, é importante salientar alguns conceitos importantes, antes de prosseguir para a explicação do uso da Meta-Aprendizagem, juntamente com o XGBoost. O primeiro deles, seria o conceito de meta-algoritmos, que utilizando da explicação anterior sobre o significado de "meta", seria uma abreviação para a Meta-Aprendizagem de um algoritmo de aprendizagem de máquina; e com isso, é possível informar também sobre as bases de meta-conhecimento e dentro delas, há algoritmos de aprendizagem de máquina. Para este trabalho, um outro conceito importante seria os meta-classificadores, algoritmos de Meta-Aprendizagem responsáveis por problemas de modelagem preditiva. Após um algoritmo de Meta-Aprendizagem treinado, o resultado vem no modelo de Meta-Aprendizagem. Por fim, há também o conceito de Meta-Características, que descreve o elemento do modelo para comparar suas semelhanças. Este trabalho foca na recomendação de hiperparâmetros para o algoritmo XGBoost, portanto, é possível fazer uma analogia com o problema de recomendação de algoritmos de aprendizagem de máquina. No livro *Metalearning - Applications to Data Mining, Brazdil* [6], é possível utilizar um exemplo de como funciona um sistema utilizando a Meta-Aprendizagem para a recomendação de algoritmos de aprendizagem de máquina, como mostra na figura 3.1

Fazendo uma breve explicação da figura 3.1, é importante focar em alguns pontos e fazer a analogia com este trabalho para um melhor entendimento e além disso, notar que a imagem é representada por um grafo direcionado, onde o resultado sempre vai para (e), o que dá como resultado o melhor algoritmo para o conjunto de dados de entrada.

O primeiro ponto é a entrada, onde é introduzido um conjunto de dados (a) e que é feito da mesma forma neste trabalho. O segundo ponto é (b), onde é retirado do conjunto de dados de entrada, as Meta-Características que vão caracterizar os conjuntos de dados (a parte das Meta-Características serão detalhadas na próxima subseção). O terceiro fator importante, é a base do meta-conhecimento em (c), que dispõe de toda a "experiência"utilizada para o sistema em si. Basicamente dispõe dos algoritmos de aprendizagem de máquina (viés inicial), só que no caso deste trabalho, seriam os hiperparâmetros e o algoritmo fixado seria o XGBoost sempre. Além disso contem informações dos conjuntos de dados anteriores junto com suas respectivas Meta-Características, para que seja possível comparar a similaridade com as Meta-Características do conjunto de dados de entrada, juntamente com o resultados de cada conjunto de dados. Sendo assim, é possível prosseguir para a próxima fase, a fase de busca da base de Meta-Conhecimento e da junção entre as Meta-Características e a base de Meta-Conhecimento.

Com a busca na base de conhecimento (*meta-learner*), e encontrado o conjunto de dados

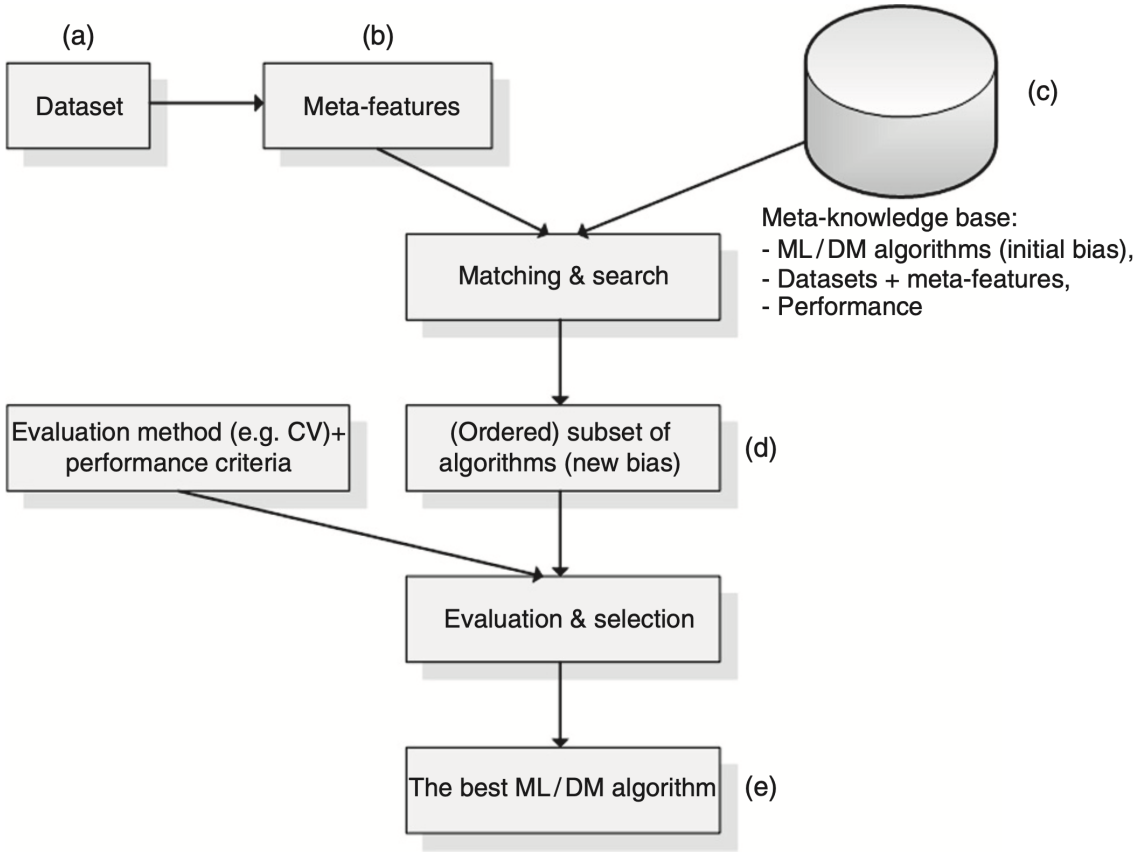


Figura 3.1: Exemplo de um sistema para a seleção de algoritmos de aprendizagem de máquina, reduzindo o espaço e procurando o algoritmo mais adequado [6]

com as Meta-Características mais semelhantes (após a seleção dos algoritmos), é apenas utilizado o hiperparâmetro do conjunto de dados encontrado, no conjunto de dados de entrada. Com isso, há um passo importante para saber se o desempenho da recomendação foi bom o suficiente, que é utilizando métodos de validação. Neste trabalho, por exemplo, foi utilizado o *K-Fold Cross Validation* sempre que testado um novo hiperparâmetro em algum conjunto de dados, é utilizado este método para avaliar a acurácia do hiperparâmetro com o XGBoost. Sendo assim, com esses passos já é possível utilizar da Meta-Aprendizagem para a recomendação de hiperparâmetros, mais a frente será explicado mais detalhadamente sobre as Meta-Características e como os experimentos aconteceram, onde cada passo será explicado mais detalhadamente.

Por fim, após o levantamento de alguns trabalhos [12] [38] [36] [22] foi visto que era sempre seguido um padrão de acordo com a literatura ao criar um projeto envolvendo a Meta-Aprendizagem. O modelo citado em 3.1 serve como base para estes trabalhos, principalmente por ser um modelo flexível no sentido de que pode servir para vários tipos de recomendações como a recomendação de algoritmos ou até mesmo hiperparâmetros, como é o caso deste presente trabalho. Sendo assim, foi analisado também em relação aos outros trabalhos, quais Meta-Características eram geralmente utilizadas para a comparação dos conjuntos de dados e como funciona o processo de extração das mesmas juntamente com a comparação entre elas.

Sendo assim, é importante ressaltar que este trabalho utilizou como base este modelo descrito na figura 3.1 e que também é utilizado em outros trabalhos na literatura.

3.2 Caracterização dos conjuntos de dados (Meta-Características)

Basicamente, a caracterização dos conjuntos de dados consiste em extrair atributos que detalhem a base de dados seguindo algum critério. Com isso, tem como objetivo fornecer informações sobre dados. A caracterização dos dados é uma das fases mais importantes no processo da aplicação da Meta-Aprendizagem, já que ele é necessário para identificar base de dados com caraterísticas semelhantes e dependendo do sistema que está sendo utilizado a Meta-Aprendizagem, como a recomendação de algoritmos (diferente deste trabalho que trata da recomendação de hiperparâmetros), saber o comportamento de algum algoritmo sob um determinado conjunto de dados em um determinado ambiente. Um exemplo citado por Aha [1] é que alguns algoritmos podem não ter um comportamento esperado na presença de alguns atributos.

De acordo com a literatura, uma das áreas de pesquisa no que condiz com a caracterização dos conjuntos de dados, é a caracterização direta. [21]. Basicamente foi um dos primeiros estudos em larga escala para relacionar a caracterização entre conjuntos de dados, algoritmos e seus desempenhos, sendo feito através do projeto STATLOG [27]. Basicamente procurava entender a acurácia dos algoritmos em alguns domínios, já que eles eram bons em uns e não tinha o resultado esperado em outros e as Meta-Características utilizadas foram a *general*, *statistical* e *info-theory* assim como neste presente trabalho. Para o primeiro (*general*) incluem medidas gerais dos conjuntos de dados, como número de atributos, número de classes etc. A *statistical* indica a distribuição dos dados e a *info-theory* pode informar a relação entre os atributos e as classes. No capítulo de experimentos será detalhado como foi utilizado cada Meta-Característica e já que foi utilizado uma biblioteca para a extração das Meta-Características, haviam pelo menos 20 variáveis para cada tipo de Meta-Característica para uma melhor descrição dos conjuntos de dados.

Sendo assim, um ponto importante durante as fases de entrada do sistema, é a extração das Meta-Características dos conjuntos de dados para a caracterização e comparação da similaridade, como dito anteriormente. É importante ressaltar que há centenas de formas de caracterizar os conjuntos de dados, sejá por número de insâncias, número de atributos etc. Hoje em dia há bibliotecas que lidam com isso de forma mais organizada, como é o caso da *pymfe* (biblioteca utilizada para a extração), que é uma biblioteca para a linguagem *Python*, que funciona para extrair justamente as Meta-Características. Sendo assim, as Meta-Características foram divididas em alguns grupos, mas neste trabalho, como citado anteriormente, foram utilizadas 3[33]:

- *General*: dá informações gerais relacionadas ao conjunto de dados, como o número de instâncias, classes, atributos, número de atributos categoricos entre várias outras informa-

ções. A utilização deste conjunto de Meta-Características é importante visto que todos os conjuntos de dados dispõe dessas informações gerais e isso faz com que seja um quesito mais genérico do que os outros conjuntos de Meta-Características. [24]

- *Statistical*: dá informações estatísticas padrões, descrevendo propriedades numéricas e distribuição dos dados (normalização, por exemplo). Este conjunto de Meta-Característica provê informações mais específicas dos conjuntos de dados. Um exemplo de informação que este conjunto provê, é a covariância absoluta dos atributos, que mede a covariância entre cada par de atributos numéricos. Além disso, também provê o número de funções discriminantes, que tem o intuito de minimizar má classificações e também dispõe de valores específicos como: média geométrica, média harmônica, curtose dos atributos etc.[26]

- *Information-theoretic*: por fim, o último conjunto descreve a relação dos atributos e os relacionamentos com as classes. Este conjunto de Meta-Característica dispõe de informações como a entropia dos atributos, para medir a aleatoridade dos atributos no conjunto de dados, concentração das classes, entropia das classes, que descreve o quanto de informação é necessária para descrever uma classe específica do conjunto de dados e entre outras variáveis. [25]

Além das 3 citadas acima, há em torno de 9 grupos a mais, cada grupo tem um conjunto de dezenas de variáveis, onde cada variável detalha uma parte do conjunto de dados. Portanto, com esses 3 conjuntos já é possível caracterizar bem os conjuntos e também é possível utilizar grupos em conjuntos. Como o *General* e o *Statistical* ao mesmo tempo, por exemplo. Com isso, no capítulo de experimentos, haverá um detalhamento maior sobre o uso das Meta-Características utilizando esta biblioteca.

3.3 Medidas de validação

Um dos pontos importantes no que condiz na aplicação da Meta-Aprendizagem, é procurar uma métrica de validação para checar o quão bom foi tanto a aplicação dos hiperparâmetros quando a recomendação feita, sendo assim, é necessário decidir algumas métricas de desempenho. Uma das métricas citadas na literatura para a validação é a validação cruzada, que nada mais é do que uma técnica para avaliar a generalização de um conjuntos de dados. Basicamente se mantém uma porção do conjuntos de dados fazendo um treino com os dados restantes [17].

Basicamente a validação cruzada particiona o conjunto de dados em subconjuntos e utiliza para estimar hiperparâmetros, no caso deste trabalho. Usando este trabalho como exemplo, durante o particionamento explicado mais adiante no capítulo 4, durante a validação cruzada, eram testados os hiperparâmetros passados para o modelo naquele momento.

Além disso, há algumas formas de utilizar a validação cruzada, são elas: *k-fold*, *holdout* e *leave-one-out*. Portanto, para qualquer uma das formas citadas, o resultado final do modelo é feito através da seguinte fórmula:

$$Ac_f = \frac{1}{v}\sum_{i=1}^{v} \varepsilon_{yi,\hat{yi}} = \frac{1}{v}\sum_{i=1}^{v} (yi - \hat{yi})$$

Onde v é a quantidade de dados para validação e as variáveis no somatório mostra o restante dado pela diferença entre o valor da saída e o valor da predição. Com o resultado é possível verificar a capacidade do modelo de generalizar.

Neste presente trabalho, foi utilizado o método *k-fold* já que tinha uma grande compatibilidade com a bilioteca utilizada para o *Grid Search* (método utilizado para a buscas de hiperparâmetros e para fazer os testes do experimento final. Basicamente o *K-Fold* funciona computando a média dos valores através das repetições. [9].

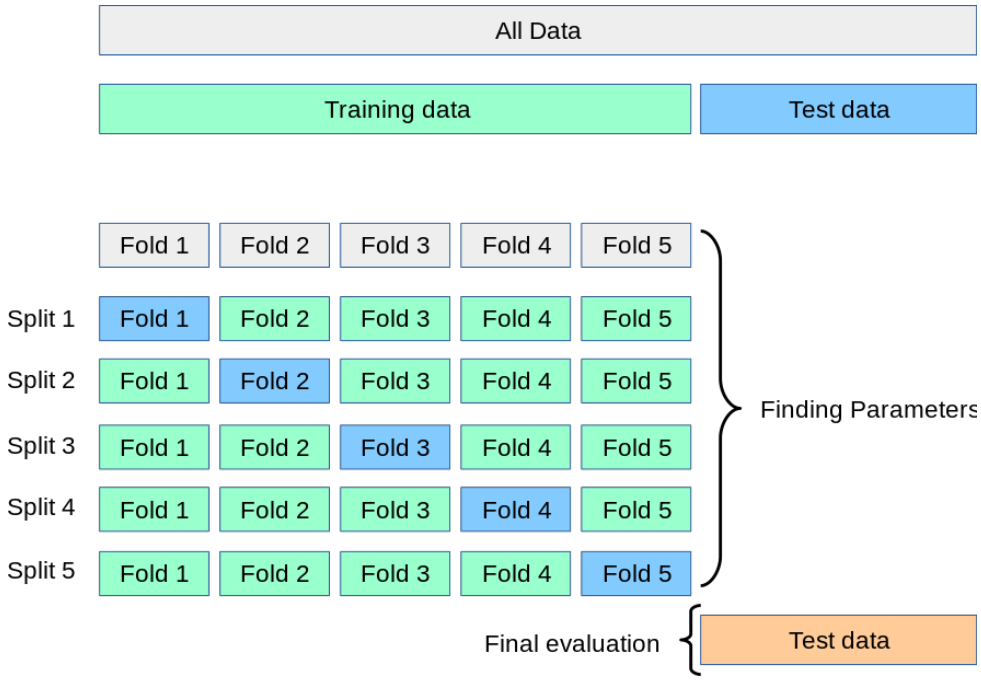


Figura 3.2: Exemplo do funcionamento do K-Fold [9]

Utilizando a imagem 3.2 como exemplo para uma melhor visualização, o método *K-Fold* consiste em dividir o conjunto de dados (*All data*) em K subconjuntos do mesmo tamanho e daí em diante um subconjunto é utilizado para testes (*Test data*) e a partir os k-1 *folds* restantes, são utilizados para estimar os hiperparâmetros. Com isso, o teste é refeito K vezes de forma cíclica. Por fim, ao final das K iterações é calculado a acurácia através dos erros encontrados, como mostra o cálculo anteriormente citado.

3.4 Formas de sugestão

De acordo com a literatura e mais específicamente por Kalousis [15] há 3 formas de sugestão. A primeira é fornecer o melhor algoritmo ou hiperparâmetro, sendo aquele que produz a melhor acurácia ou o modelo mais apropriado para uma dada tarefa. Um ponto negativo sobre esta forma de sugestão é que o algoritmo sugerido pode não funcionar de maneira esperada em outras situações. Apesar do ponto positivo no que condiz a facilidade de sugerir apenas um algoritmo ou hiperparâmetro, o mesmo pode não ter resultados esperados quando usado em situações diferentes. Além disso, pode haver algum viés durante o experimento ou no uso da Meta-Aprendizagem para a recomendação, fazendo com que os parâmetros de comparação de semelhança não sejam aplicados corretamente ou de forma ótima para aquele algoritmo ou hiperparâmetro sugerido. Um exemplo para esse problema é utilizar Meta-Características que não descrevam bem os conjuntos de dados fazendo com que o meta-aprendiz sugira um algoritmo que pode ser bom para uma situação específica mas pode não ser bom para o conjunto de dados que precisa da recomendação.

A segunda forma é indicar os algoritmos ou hiperparâmetros mais indicados para um determinado problema. Basicamente é possível utilizar alguns parâmetros para determinar o quão bom é o desempenho de algum algoritmo ou hiperparâmetro. Sendo assim, uma alternativa é fornecer ao usuário que está utilizando o sistema de recomendação, não só o algoritmo ou hiperparâmetro com o melhor desempenho mas sim outros que tiveram bons desempenhos também. No caso deste trabalho, foi utilizado o algoritmo de aprendizagem de máquina KNN para determinar os hiperparâmetros que tiveram os melhores resultados.

Por fim, o terceiro método é exibir os algoritmos em ordem de preferência em relação aos conjuntos de dados que estão no meta-aprendiz. Um ponto positivo para este método é utilizar vários critéiros de ordenação dos resultados. No caso deste trabalho, será detalhado no capítulo de experimentos como foi feito o sistema de recomendação, mas, para ordenar a sugestão dos hiperparâmetros, foi utilizado a acurácia dos hiperparâmetros calculadas previamente e além disso, foram utilizadas diferentes formas de sugestão, como: hiperparâmetros diretos, onde foi utilizado o hiperparâmetro encontrado diretamente, média de hiperparâmetro, entre outros.

3.5 Construção da sugestão

Um dos pontos principais no quesito da recomendação utilizando a Meta-Aprendizagem é fazer a relação entre os conjuntos de dados do meta-aprendiz, seus resultados e o conjunto de dados que está "precisando"da recomendação.

Como citado anteriormente, por conta da flexbilidade da Meta-Aprendizagem, é possível utilizar algumas heurísticas para a construção da sugestão. É possível visualizar a construção da sugestão também como um problema de aprendizagem de máquina já que é possível, por exemplo, utilizar algum algoritmo de aprendizagem de máquina para uma ordenação dos hiper-

parâmetros que serão recomendados. Sendo assim, é possível utilizar um KNN, por exemplo, de forma que seja possível pegar os conjuntos de dados mais próximos com seus respectivos hiperparâmetros. Como citado anteriormente, as formas de construção e e recomendação da sugestão tem pontos positivos e também negativos. Um ponto negativo na sugestão dos hiperparâmetros diretamente, é que ele pode de fato, funcionar para um conjunto de dados específico mas pode ter uma acurácia reduzida quando utilizado em algum outro.

Por conta da flexbilidade citada anteriormente, é possível utilizar os hiperparâmetros que são a priori recomendados e utilizados de alguma forma, como fazer a média ou moda dos hiperparâmetros, por exemplo, além de utilizar os hiperparâmetros diretamente, é possível manipulá-los de alguma forma para que seja possível gerar outras recomendações a fim de obter também bons resultados.

Supondo uma situação onde seja necessário utilizar um *GridSearch* para a busca de hiperparâmetros em um grande conjunto de dados e isso seja uma atividade que demore alguns dias ou até semanas, utilizar a manipulação de uma primeira sugestão para criar novas sugestões que também dêem bons resultados, também é proveitoso para quem está usando o sistema de Meta-Aprendizagem.

4

Experimentos

Nesta seção será explicado de forma mais aprofundada a relação entre a meta-aprendizagem, o *XGBoost* e todo o desenvolver dos experimentos. Primeiramente será dada uma breve introdução sobre como foi dividido o experimento, as etapas iniciais e uma breve explicação sobre as escolhas feitas no experimento. Em seguida, serão mostrados alguns pontos do experimentos como a parte do tratemento dos dados e a busca pelos hiperparâmetros utilizados no experimento; após isso, a metodologia para descrever cada uma das partes do experimento de forma mais aprfundada, inclusive, a construção do meta-aprendiz e por fim, uma explicação sobre as métricas de dissimilaridade e a comparação com os *baselines*, que seria a forma de comparar os resultados do experimento.

4.1 Introdução

O experimento foi dividido em algumas etapas e em diferentes conjuntos. Como citado anteriormente, 3 conjuntos de Meta-Características foram utilizados: *general*, *statistical* e *information-theoretic*. Após as etapas iniciais que serão detalhadas mais adiante, os experimentos foram divididos em etapas conclusivas diferentes, utilizando pares de Meta-Características e uma final, utilizando as 3 Meta-Características de uma vez só. Por se tratar da proposta da dissertação, posteriormente serão feitos experimentos utilizando as 3 Meta-Características separadamente, a fim de analisar o desempenho de cada uma individualmente. Isso foi feito com o intuito de checar a similaridade e influência do uso das Meta-Características sendo usadas em diferentes conjuntos. Por exemplo: o primeiro teste feito foi utilizando a *Meta-Característica general* juntamente com a *Meta-Característica statistical*; depois a *Meta-Característica statistical* juntamente com a *Meta-Característica information-theoretic* e assim por diante.

Os códigos foram feitos na linguagem Python, utilizando o Jupyter Notebook (uma aplicação web de código aberto para criar e compartilhar documentos), fazendo com que facilitasse

os códigos e a compreensão, tendo em vista que ambos já são ferramentas com vários recursos para este tipo de problema. Com isso, foram utilizados os 198 conjuntos de dados e para cada um, foi possível calcular um resultado.

4.2 Conjuntos de dados utilizados

Os conjuntos de dados utilizados foram obtidos através da plataforma *Kaggle*, *openML* e outros sites abertos que disponibilizam conjuntos de dados relacionados a diversas áreas, mas para o caso deste trabalho, sempre focando em classificação. Um ponto importante que é necessário focar aqui, é que como dito anteriormente, o próprio algoritmo do XGBoost não consegue lidar bem com dados categóricos, então para cada um dos conjuntos de dados, na etapa de tratamento, era necessário transformar as categorias em diferentes colunas com valores numéricos, o que serão explicados mais a frente. Além disso, há alguns tipos conhecidos de conjuntos de dados como: Conjuntos de dados numéricos, categóricos, multivariáveis etc. Para este presente trabalhos, as colunas dos conjuntos de dados foram transformadas para numéricas, para que o XGBoost lidasse de forma melhor.

4.3 *Grid Seach* para a busca de hiperparâmetros ideais

Uma outra problemática citada neste trabalho, é que para a construção de um *baseline* de comparação, para cada um dos conjuntos de dados, é necessário encontrar hiperparâmetros que são melhores do que os padrões utilizados pelo XGBoost. Para isso, foi utilizado o *Grid Search*, que basicamente funciona como uma busca exaustiva por hiperparâmetros [13]. A biblioteca utilizado foi a *scikit-learn*, que dispõe de uma função *GridSearchCV* e funciona considerando todas as combinações dos hiperparãmetros passados por parâmetro.

Como explicado anteriormente, os hiperparâmetros utilizados neste trabalho foram: *max depth*, *min child weight*, *gamma*, *subsample* e *colsample by tree*. Sendo assim, para cada um deles, era passado um intervalo de onde começaria e onde terminaria o valor e o *Grid search*, faria todas as combinações possíveis de hiperparâmetro para encontrar a melhor acurácia.

Um outro fator importante na utilização do *GridSearchCV* é que ele já dispõe da utilização do *cross-validation*. Fazendo assim, com que haja uma confiança maior na acurácia do algoritmo.

4.4 Metodologia

Nesta seção será demonstrada a metodologia utilizada, desde o tratamento dos dados até os resultados obtidos pelo código para os experimentos. Será explicado de forma mais detalhada, primeiramente, sobre o tratamento dos dados e sua imporância para o *XGboost*, após

isso, sobre a própria execução do *XGBoost* no experimento, em seguida, será explicado como funcionou o esquema de extração das meta-características; por ser uma das partes mais importantes do experimento, visto que é o que visa caracterizar cada um dos 198 conjuntos de dados e por fim, serão mostrados o funcionamento da distância euclidiana como forma de métrica de similaridade e a comparação com os *baselines* no experimento. É importante ressaltar que a metodologia referente ao experimento foi baseada nos seguintes passos:

- Tratar os 198 conjuntos de dados utilizados no trabalho, visto que é necessário que as variáveis categóricas sejam tratadas em todos os conjuntos de dados;
- Extração das Meta-Características em cada um dos conjuntos de dados;
- *GridSearch* nos hiperparâmetros para encontrar combinações de hiperparâmetros melhores do que o padrão do XGBoost;
- Explorar técnicas, abordagens e métricas de similaridade utilizadas na comparação de Meta-Características;
- Testar cada hiperparâmetro encontrado utilizando os conjuntos de dados mais próximos de acordo com a similairdade das Meta-Características;
- Avaliar os resultados de cada hiperparâmetro utilizado em cada um dos 198 conjuntos de dados;
- Comparar os resultados de cada recomendação dentre os 10 conjuntos de dados mais próximos para cada um testado no experimento, verificando se os resultados foram melhores do que os hiperparâmetros padrão do XGBoost utilizando também, os *baselines*.

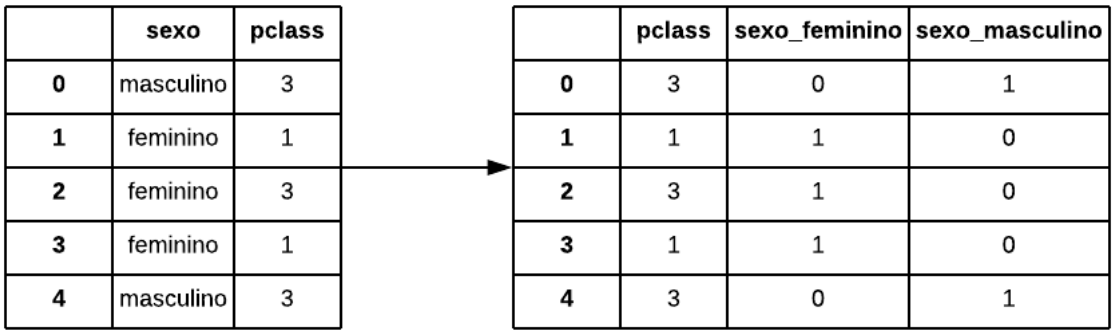
Seguindo assim, um maior detalhamento sobre as etapas mais importantes do experimento.

4.4.1 Tratamento dos dados

Para fins didáticos e para uma melhor visualização dos dados e dos resultados, foi utilizado o *Jupyter Notebook*; um *software* de código aberto que visa prover uma melhor interface de integração com o usuário e uma variedade de funcionalidades que auxiliam nos trabalhos de cientistas de dados.

Através do *Jupyter Notebook*, foi possível utilizar código Python para o tratamento dos dados, através da biblioteca *Pandas*. Através desta biblioteca, é possível transformar todos os conjuntos de dados em *Dataframes*, um tipo de estrutura de dados que facilita na hora do tratamento e limpeza dos dados, além de facilitar também, a execução de algoritmos de aprendizagem de máquina. Sendo assim, como cada conjunto de dados tinha uma particularidade e seria necessário transformar todas as colunas categóricas em dados numéricos, foi criado um algoritmo transformando todos os conjuntos em *Dataframes* e fazendo a conversão nas colunas.

Além disso, um ponto que é importante detalhar, é sobre como ocorreu o tratamento dos dados para transformar os valores das colunas em números reais, quando eram categóricos. Na figura 4.1 é possível visualizar um resumo de como houve este tratamento.



	sexo	pclass
0	masculino	3
1	feminino	1
2	feminino	3
3	feminino	1
4	masculino	3

	pclass	sexo_feminino	sexo_masculino
0	3	0	1
1	1	1	0
2	3	1	0
3	1	1	0
4	3	0	1

Figura 4.1: Exemplo de tratamento de variáveis categóricas para variáveis reais

Basicamente, a coluna “sexo“ possui dois valores: masculino e feminino. Com isso, ela pode ser separada em duas colunas, onde terá o valor 1 na coluna “feminino“ e o valor 0 para “masculino“, caso o valor da instância seja feminino e o mesmo para o caso contrário.

4.4.2 Execução do XGBoost juntamente com a aplicação do *GridSearch*

Após a limpeza e tratamento dos dados, os hiperparâmetros ideais para cada algoritmo eram encontrados utilizando o *GridSearchCV*. Nesta etapa, o *GridSearchCV* recebia como parâmetro o *estimator*, que no caso deste trabalho, é o Classificador do XGBoost, já que o foco são problemas de classificação. O segundo parâmetro é a "grade de hiperparâmetros", que seriam os intervalos para cada um dos hiperparâmetros escolhidos dando assim, um total de 648 combinações possíveis de hiperparâmetros. É importante ressaltar que a intenção era não colocar um valor que fosse muito acima ou muito abaixo dos valores padrões do XGBoost, a fim de não causar um *overfitting* ou um *underfitting*, mas que fossem valores que fizessem a diferença em relação ao padrão. Com isso, foram escolhidos os seguintes valores para cada um:

- *Max depth*: foi utilizado o intervalo de 3 a 8. O valor padrão do XGBoost é utilizar o *Max depth* como 6.
- *Min child weight*: foi utilizado um intervalo de 1 a 3. O valor padrão que o XGBoost utiliza é 1.
- *Gamma*: os valores testados para este hiperparâmetro foram: 0, 0.2, 0.5 e 1. O padrão é 0.
- *Subsample*: os valores utilizados foram: 0.2, 0.5 e 1. Para este hiperparâmetro, o padrão é 1.

- *Colsample bytree*: por fim, para este hiperparâmetro foram utilizados também os valores: 0.2, 0.5 e 1. O padrão para este hiperparâmetro também é 1.

Sendo assim, após a definição dos valores de cada hiperparâmetro o *GridSearchCV* tem o trabalho de combinar todos eles a fim de achar o melhor conjunto para O XGBoost em cada conjunto de dados. Além disso, é possível passar como terceiro parâmetro do *GridSearchCV* o número de *folds*, para manipular o particionamento da validação cruzada, por padrão, foi utilizado 5.

Além da execução do XGBoost com os hiperparâmetros encontrados através do *Grid Search*, também era executado em cima do conjunto de dados, o XGBoost utilizando os hiperparâmetros padrões. Com essas informações, seria possível construir um objeto que seria utilizado em comparação mais a frente, com as seguintes informações, por exemplo:

Informações do conjunto de dados	
Nome	acute-inflammations_1556
Resultado com os hiperparâmetros padrões	95.83%
Resultado com os hiperparâmetros otimizados	100%
Hiperparâmetros	
max_depth	3
min_child_weight	1
gamma	0
subsample	1
colsample_bytree	0.2

Tabela 4.1: Tabela com as informações contidas no objeto

Utilizando a tabela 4.1 como exemplo, para o experimento, foi criado um objeto que desse algumas informações de cada conjunto de dados a fim de usar cada um de forma quadrática, ou seja, cada um seria comparado com todos os outros que estivessem no meta-aprendiz.

Primeiramente, há algumas informações gerais. A primeira informação é o nome, já que seria necessário para saber qual conjunto de dados estava sendo utilizado. A segunda informação é a acurácia com os hiperparâmetros padrões do XGBoost, ou seja, assim que é criada uma instância do algoritmo do XGBoost, utilizando os hiperparâmetros carregados por padrão, é guardada a acurácia no objeto a fim de comparação futuramente. Por fim a terceira informação geral concede o resultado utilizando os hiperparâmetros encontrados através da técnica de *Grid Search*; que também será utilizado para comparação mais adiante. Por fim, como informação específica, é dado os hiperparâmetros que foram utilizados para a otimização do algoritmo naquele conjunto de dados escolhido, isso será utilizado também no experimento para testar esses hiperparâmetros em outros conjuntos de dados, quando esse conjunto de dados for um dos mais próximos a outros.

4.4.3 Extração e comparação das Meta-Características

Da mesma forma que a execução de XGBoost, que foi feita em cada um dos conjuntos de dados, houve também a extração das Meta-Características para cada um. Como citado anteriormente, foram usados 3 conjuntos: *General*, *Statistical* e *Info-theory*. Sendo assim, para os experimentos, foram separados em conjuntos de pares e no final, todos os grupos foram utilizados juntos (posteriormente, serão analisados individualmente também), ou seja: *General* com *Statistical*, *General* com *Info-theory*, *Statistical* com *Info-theory* e por fim, os 3 utilizados em conjunto. Isso foi feito com a intenção de ter mais variáveis de Meta-Características trabalhando em conjuntos e foi feito um teste para cada conjunto para analisar o quão seria melhor a utilização de cada um deles em conjunto com outros. Mas com a construção do sistema, é possível também colocar qualquer outro conjunto de *Meta-Característica* disponível na biblioteca utilizada.

Sendo assim, de forma análoga a execução do XGBoost, também foi criado um objeto JSON para cada um dos conjuntos de dados, que dá informação sobre o nome do conjunto de dados utilizados e as variáveis das Meta-Características utiliadas naquele objeto. Na tabela 4.2 mostra um exemplo com uma parte do objeto criado para o experimento relacionado à Meta-Características.

Informações do conjunto de dados	
Nome	acute-inflammations_1556
Meta-Características	
attr_conc.mean	0.10743062316399
attr_conc.sd	0.12663304982868845
attr_ent.mean	1.1122714794209874
attr_ent.sd	0.440167083879265
attr_to_inst	0.05
cat_to_num	0.0
nr_attr	6.0
nr_inst	120.0

Tabela 4.2: Tabela com as informações das Meta-Características

É importante ressaltar que esta tabela mostra apenas uma parte das variáveis contidas no objeto, cada um deles contém cerca de 10 a 30 variáveis de que descrevem os grupos de Meta-Características escolhidos. Sendo assim, com essas informações é possível utilizar de comparação para a similaridade entre os conjuntos de dados.

Para os experimentos, todas as variáveis que descrevem as Meta-Características foram utilizadas como se fossem vetores, fazendo assim, com que fosse possível aplicar a métrica de distância euclidiana. Basicamente para cada conjunto de dados, há uma comparação com todos os outros conjuntos de dados do meta-aprendiz, fazendo com que cada valor da variável, seja aplicado à distância euclidiana e para cada um dos conjuntos de dados, é obtido um objeto com os 10 conjuntos de dados mais próximos. Foram escolhidos os 10 primeiros com o intuito de se

ter uma comparação utilizando os "N"primeiros mais próximos, já que um dos experimentos, alguns dos *baselines* de comparação, foi utilizando a média e moda dos hiperparâmetros dos "N"conjuntos de dados mais próximos.

4.4.4 Aplicação da distância euclidiana, execução dos hiperparâmetros e extração dos resultados

Como citado anteriormente, após a execução dos passos do XGBoost juntamente com a extração dos hiperparâmetros, foi aplicado para cada um dos conjunto de dados, a distância euclidiana entre cada um dos conjuntos. Sendo assim, foi criado um objeto que divide os resultados em três partes. A primeira parte como demonstra a tabela 4.2, dá as seguintes informações: qual conjunto de dados foi escolhido atualmente, o resultado com os hiperparâmetros padrões do XGBoost, após isso, o resultado para os primeiros 10 primeiros (os 10 primeiros mais próximos de acordo com a distância euclidiana) conjuntos de dados e por fim, um resumo dos resultados para a comparação dos *baselines* utilizados no final do experimento.

Na tabela 4.2 é possível visualizar que a ordem dos conjuntos de dados, estão crescente de acordo com a distância, sendo assim, os mais próximos vêm primeiro. Com isso, teoricamente os conjuntos de dados nas primeiras posições, tem uma similaridade maior com o conjunto de dados escolhido no exemplo *acute-inflammations_1556*. Sendo assim, teoricamente os hiperparâmetros dele (por ser mais próximo) funcionariam com uma acurácia melhor do que os mais distantes. O objeto também informa a acurácia utilizando os hiperparâmetros de cada um dos 10 conjuntos de dados. É possível notar com este exemplo, por exemplo, que o primeiro tem uma semelhança em torno de 46.1404 como conjunto de dados escolhido *acute-inflammations_1556* e utilizando os hiperparâmetros dele (*fri c3 100 5 916*) foi possível obter um aumento na acurácia de 95.83 para 97.5, fazendo assim, com que já tivesse um resultado melhor do que o resultado padrão.

Com a primeira parte dos resultados que eram formado no objeto, é possível construir a segunda parte dos resultados. Basicamente a segunda parte é formada por uma iteração dos 10 conjuntos de dados mais próximos. Sendo assim, é possível pegar a média e moda dos hiperparâmetros. Outro ponto que é importante ressaltar, foi o citado no começo do trabalho sobre os "bons conjuntos de dados", que seriam os que só utilizando os hiperparâmetros deles, já tinham um resultado melhor do que o padrão. Sendo assim, além da média e moda de todos os conjuntos de dados, também foi feito a média e moda apenas utilizando os "bons conjuntos de dados". Na tabela 4.3 é possível visualizar a segunda parte dos resultados.

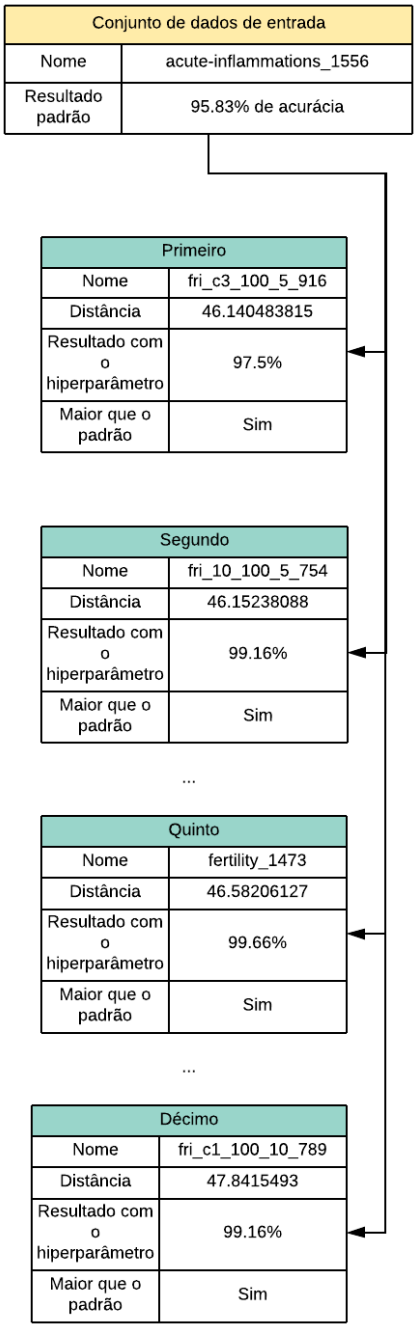


Figura 4.2: Tabela de exemplo com a primeira parte dos resultados (utilizando os conjuntos de *Meta-Características general* e *statistical*)

A ordem feita na tabela 4.3 é a mesma ordem feita na tabela 4.2, em ordem crescente em relação a distância euclidiana dos conjuntos de dados mostrados. É possível notar que inicialmente há 4 valores: média, moda, média dos bons conjuntos e moda dos bons conjuntos. Quando o N é 1, há apenas o conjunto de dados mais próximo e os hiperparâmetros dele foram utilizados diretamente. Quando há 2, é feito a média dos hiperparâmetros (pegando os valores dos hiperparâmetros, somando e dividindo por 2) e a moda dos hiperparâmetros, o mesmo é feito utilizando apenas os 2 primeiros conjuntos de dados considerado bons. E o mesmo é feito sucessivamente até chegar no décimo conjunto de dados. É importante ressaltar que alguns conjuntos de dados no momento do experimento, não tinham 10 conjuntos de dados considerados bons, já que alguns conjuntos de dados poderiam recomendar hiperparâmetros que poderiam fornecer uma acurácia pior do que a dada por padrão. Sendo assim, o cálculo da média e moda dos conjuntos bons, eram feitos apenas até o número de conjuntos bons, considerando um valor máximo de até 10 conjuntos de dados.

Resultados de acordo com o N	
1	
Média	97.5%
Moda	97.5%
Média com bons conjuntos	97.5%
Moda com bons conjuntos	97.5%
2	
Média	96.6%
Moda	97.5%
Média com bons conjuntos	96.6%
Moda com bons conjuntos	97.5%
3	
Média	92.5%
Moda	97.5%
Média com bons conjuntos	92.5%
Moda com bons conjuntos	97.5%
4	
Média	100%
Moda	95.8%
Média com bons conjuntos	100%
Moda com bons conjuntos	95.8%
5	
Média	100%
Moda	97.8%
Média com bons conjuntos	100%
Moda com bons conjuntos	97.8%
6	
Média	99.1%
Moda	95.8%
Média com bons conjuntos	99.1%
Moda com bons conjuntos	95.8%
7	
Média	99.1%
Moda	99.1%
Média com bons conjuntos	99.1%
Moda com bons conjuntos	99.1%
8	
Média	100%
Moda	99.1%
Média com bons conjuntos	100%
Moda com bons conjuntos	99.1%
9	
Média	100%
Moda	99.1%
Média com bons conjuntos	100%
Moda com bons conjuntos	99.1%
10	
Média	100%
Moda	99.1%
Média com bons conjuntos	100%
Moda com bons conjuntos	99.1%

Tabela 4.3: Tabela de exemplo com a segunda parte dos resultados (utilizando os conjuntos de *Meta-Características general* e *statistical*)

Por fim, após a construção das duas primeiras partes dos resultados, é possível construir a terceira e última parte, que é o resumo dos resultados anteriores junto com algumas observações. Um resumo demonstrando o melhor resultado utilizando a média, outro utilizando moda, o mesmo para bons conjuntos de dados e o melhor resultado utilizando os hiperparâmetros diretamente. Neste momento é possível notar que há 5 formas de recomendação de hiperparâmetros: a média e moda dos hiperparâmetros, média e moda dos hiperparâmetros utilizando apenas bons conjuntos de dados e utilizando o hiperparâmetro dos conjuntos de dados diretamente. É possível visualizar um exemplo do resultado 4.4.

Resumo dos resultados		
	Valor	N
Melhor resultado com a média	100%	3
Melhor resultado com a moda	99.1%	6
Melhor resultado com a média utilizando bons conjuntos	100%	3
Melhor resultado com a moda utilizando bons conjuntos	99.1%	9
Melhor resultado utilizando os hiperparâmetros diretamente	99.1%	-

Tabela 4.4: Tabela de exemplo com a terceira parte dos resultados (utilizando os conjuntos de *Meta-Características general* e *statistical*)

Para um breve entendimento sobre a tabela 4.4, ela dispõe apenas dos melhores resultados para cada (moda e média, tanto para todos os conjuntos quanto para os considerados bons) e o hiperparâmetro diretamente. A separação e resumo dos resultados em 3 partes teve como importância a flexibilidade para a comparação com os *baselines*. Tendo em vista que os resultados para cada um dos 198 conjuntos de dados, seriam comparados igualmente. Na próxima sub-seção, será explicada de forma mais aprofundada, como as 3 partes dos resultados foram utilizadas para o resultado geral do experimento.

4.4.5 Comparação com os *baselines*

Primeiramente é importante entender sobre o que é necessário comparar para a validação dos resultados. O foco deste trabalho é mostrar que é possível recomendar hiperparâmetros que tenham valores maiores que utilizando os hiperparâmetros padrões do XGBoost. Sendo assim, a comparação gira em torno dos próprios hiperparâmetros padrões. Outro caso de comparação, é utilizando a média e moda de todos os conjuntos de dados. Basicamente para comprovar que apenas os melhores (ou 10 mais próximos) conjuntos de dados poderiam influenciar na acurácia do algoritmo. Sendo assim, os resultados foram separados em etapas. Além disso, é importante ressaltar que houveram os *baselines* de comparação com cada uma das etapas, ou seja, os resultados de cada etapa foram comparados com os resultados padrões do XGBoost e

além disso, houve também um estudo sobre o uso da média e moda de todos os hiperparâmetros contidos no meta-aprendiz, também como forma de *baseline*, para checar se eles teriam alguma influência na acurácia e não só a utilização dos conjuntos de dados mais próximos. Com isso, as etapas citadas anteriormente foram divididas da seguinte forma:

- *General*
- *Statistical*
- *Info-Theory*
- *General e Statistical*
- *General e Info-Theory*
- *Info-Theory e Statistical*
- *General, Statistical e Info-Theory*
- Média dos hiperparâmetros de todos os conjuntos de dados
- Moda dos hiperparâmetros de todos os conjuntos de dados

Basicamente, as 7 primeiras etapas foram feitas com o intuito de checar a influência de diferentes grupos Meta-Características utilizados em conjunto. As 2 últimas etapas, foram feitas com o intuito de checar o viés de utilizar a média e moda dos hiperparâmetros, como maneira de checar se utilizando a média de todos os hiperparâmetros disponíveis no meta-aprendiz, teria algum resultado efetivo. Além disso, é importante ressaltar que para cada uma dessas 9 etapas, tiveram a mesma base de resultados:

- Mostrar o número de conjuntos de dados que tiveram a acurácia melhorada utilizando a média dos hiperparâmetros dos conjuntos de dados mais próximos
- Utilizando a moda dos hiperparâmetros dos conjuntos de dados mais próximos
- A média dos hiperparâmetros dos conjuntos de dados utilizando os conjuntos de dados "bons"(que com o hiperparâmetro utilizado individualmente, já aumentaria a acurácia do conjunto de dados que estava sendo testado)
- A moda dos hiperparâmetros dos conjuntos de dados bons

5

Resultados e Discussões

Este capítulo tem por finalidade mostrar os resultados obtidos em cada uma das etapas do experimento. Haverá uma breve descrição sobre cada etapa, mostrando a diferença entre elas e os resultados em cada uma. Além disso, será mostrado uma comparação entre cada uma das etapas mostrando as que tiveram os melhores resultados e principalmente, o custo computacional juntamente com o tempo economizado em utilizar a meta-aprendizagem ao invés de fazer uma busca custosa por bons hiperparâmetros. A motivação na escolha dos conjuntos de *meta-features* (*general, statistical* e *info-theory*) foi a popularidade de cada uma, já que ela tem os detalhes mais gerais de cada conjunto de dados; já que o intuito do trabalho é apenas mostrar que é possível melhorar utilizando a meta-aprendizagem e não mostrar qual é o melhor hiperparâmetro ou conjunto de *meta-feature* para cada uma. Além disso, é importante ressaltar que para os experimentos, foram utilizados exatos 198 conjuntos de dados. Por fim, no final do capítulo serão descritas ameaças a validade dos experimentos, tendo em vista que é importante uma análise em torno dos resultados para mostrar o que pode comprometer o resultado dos mesmos ao longo do tempo.

5.1 Introdução

Primeiramente é importante entender e se ter uma breve explicação de como foram separados os experimentos e o que foi importante ressaltar dentre os resultados para que se tivesse uma melhor visualização dos dados. Para cada um dos experimentos, os resultados foram separados em: informações gerais e informações específicas. Para as informações gerais, teve-se os seguintes campos:

- Conjuntos otimizados no total: contendo informações em torno de quantos conjuntos de dados foram otimizados dentre os 198 citados anteriormente que compõem o meta-aprendiz como um todo.

- Conjuntos com resultado igual ao padrão: durante o experimento teve-se a preocupação de checar quantos conjuntos não seriam melhorados mas que tivessem resultados idênticos ao resultado do XGBoost por padrão. Os que não pertencem a esses dois conjuntos (otimizados e com resultados iguais ao padrão), tiveram a acurácia piorada em relação ao padrão do XGBoost.

- Conjuntos melhorados utilizando a média: dentre esses, estão os conjuntos de dados que foram otimizados utilizando apenas a média dos hiperparâmetros entre os 10 mais próximos de cada um testado no experimento. Não podendo ser exatamente os "10 mais próximos", mas podendo ser os 2 mais próximos, 3 mais próximos, indo até 10.

- Conjuntos melhorados utilizando a moda: o mesmo esquema do anterior só que utilizando a moda dos hiperparâmetros.

- Conjuntos melhorados utilizando a média dos bons conjuntos: como dito anteriormente, neste trabalho a definição de bons conjuntos de dados são aqueles que utilizando os hiperparâmetros deles diretamente no conjunto de dados testado, já melhoram a acurácia. Com isso, dentre os 10 conjuntos de dados mais próximos, foi pego a média dos hiperparâmetros apenas dos bons conjuntos.

- Conjuntos melhorados utilizando a moda dos bons conjuntos: mesmo esquema do campo citado anteriormente mas utilizando a moda dos hiperparâmetros.

- Conjuntos melhorados utilizando os hiperparâmetros diretamente: por fim, o último campo condiz com os "bons conjuntos", que seriam os que utilizando o hiperparâmetro diretamente, já melhorava a acurácia.

Com isso, para as informações específicas houve um estudo em torno do quanto a acurácia melhoraria utilizando a meta-aprendizagem. Sendo assim, foram divididos nos seguintes campos:

- Média de acurácia melhorada: para este campo, foi apenas pego a média de melhora na acurácia dentre todos os conjuntos testados no experimento (que seriam os próprios 198 no meta-aprendiz).

- Melhora máxima alcançada: para aquele conjunto de *meta-features* sendo testado, foi pego a maior diferença entre a acurácia utilizando a meta-aprendizagem e a acurácia padrão do XGBoost.

- Melhora mínima alcançada: o mesmo que o passo anterior só que pegando a diferença mínima.

GENERAL		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	165	83.33%
Iguais ao padrão	22 (187)	11.11% (94.44%)
Utilizando a média	151	76.26%
Utilizando a moda	138	69.69%
Utilizando a média dos bons conjuntos	160	80.80%
Utilizando a moda dos bons conjuntos	150	75.75%
Utilizando hiperparâmetro diretamente	160	80.80%
Informações Específicas		
Media de acurácia melhorada		3.60%
Melhora máxima alcançada		21.60%
Melhora mínima alcançada		1.42%

Tabela 5.1: Tabela com o resultado utilizando o conjunto de meta-features general

5.2 Utilizando os conjuntos de *meta-features general*

Para o primeiro resultado utilizando os hiperparâmetros individualmente, foi utilizado o conjunto de Meta-Características referentes ao *General*. Para este primeiro experimento, já foi possível visualizar resultados significativos, na qual houve um alcance 83.3% de conjuntos de dados otimizados dentre os 198.

De acordo com a tabela 5.1 é possível visualizar que utilizando a recomendação com a média dos bons conjuntos de dados e os hiperparâmetros diretamente, já foi obtido resultados significativos, conseguindo otimizar 80.80% dos conjuntos de dados. Além disso, houveram 22 conjuntos de dados que tiveram resultados iguais ao padrão do XGBoost, com isso, 94.44% dos conjuntos de dados tiveram a acurácia tão boa quanto a do XGBoost sem o ajuste de hiperparâmetros.

Outro fator que é importante ressaltar, houve uma melhora máxima da acurácia em 21.60% além disso, a melhora mínima alcançada foi 1,42% e a média ficou em 3.60%. É importante ressaltar novamente que essa média poderia ter sido um pouco maior caso houvesse uma procura mais profunda de hiperparâmetros que melhorasse cada conjunto de dados individualmente, mas como esse foi não o foco deste presente trabalho, foram achados apenas hiperparâmetros que já dessem uma melhora significativa em cada um dos conjuntos de dados.

5.3 Utilizando os conjuntos de *meta-features statistical*

Para o segundo resultado utilizando os hiperparâmetros individualmente, foi utilizado o conjunto de Meta-Características referentes ao *Statistical*. Para este segundo experimento, houveram resultados gerais melhores em relação ao anterior, como é o caso do número de conjuntos de dados otimizados no total mas em contrapartida, os resultados individiais utilizando média, média dos bons entre outros, tiveram resultados um pouco piores.

De acordo com a tabela 5.2 é possível notar que utilizando a recomendação com a média

STATISTICAL		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	166	83.33%
Iguais ao padrão	20 (187)	10.10% (93.43%)
Utilizando a média	150	75.75%
Utilizando a moda	139	70.20%
Utilizando a média dos bons conjuntos	156	78.78%
Utilizando a moda dos bons conjuntos	150	75.75%
Utilizando hiperparâmetro diretamente	156	78.78%
Informações Específicas		
Media de acurácia melhorada		3.61%
Melhora máxima alcançada		20.79%
Melhora mínima alcançada		0.08%

Tabela 5.2: Tabela com o resultado utilizando o conjunto de meta-features statistical

dos bons conjuntos e os hiperparâmetros diratemente, foram obtidos resultados significativos, portanto, menores que o experimento anterior e de forma geral, os resultados para acurácia melhorada mínima, foi menor também.

Em contrapartida, por mais que os resultados tenho sidos inferiores ao experimento anterior, é importante utilizar da Meta-Característica *Statistical* já que ela pode dispor de informações mais detalhadas sobre os conjuntos de dados que os outros conjuntos de Meta-Características podem não obter. E além disso, ainda assim, 93.43% dos conjuntos de dados tiveram resultados tão bom quanto os padrões do XGBoost.

5.4 Utilizando os conjuntos de *meta-features info-theory*

Por fim, para o terceiro e último experimento utilizando o conjunto de Meta-Características individuais, foi utilizado a Meta-Característica *Info-Theory* que em comparação com os dois anteriores, obteve o melhor resultado.

Um ponto interessante para este experimento, é que em comparação ao primeiro, ele também teve 160 conjuntos de dados melhorados no melhor caso; portanto, o número de conjuntos de dados otimizados no total, foi de 170, 5 a mais em relação ao primeiro. Além disso, as informações específicas para a acurácia melhorada, tiveram resultados semelhantes aos anteriores; com uma média de acurácia melhorada em 3.61% e máxima de 24%, que também houve uma melhora em relação aos anteriores.

INFO-THEORY		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	170	85.85%
Iguais ao padrão	15 (185)	07.57% (93.42%)
Utilizando a média	151	76.26%
Utilizando a moda	141	71.21%
Utilizando a média dos bons conjuntos	160	80.80%
Utilizando a moda dos bons conjuntos	153	77.27%
Utilizando hiperparâmetro diretamente	160	80.80%
Informações Específicas		
Media de acurácia melhorada		3.61%
Melhora máxima alcançada		24.00%
Melhora mínima alcançada		0.002%

Tabela 5.3: Tabela com o resultado utilizando o conjunto de meta-features info-theory

5.5 Utilizando os conjuntos de *meta-features general* e *statistical*

Para o primeiro resultado, foi utilizado o conjunto de *meta-features general* e *statistical*. Foram utilizadas cerca de 37 variáveis para a descrição dos dois conjuntos e para o uso no cálculo da similaridade. Sendo assim, para este primeiro experimento, houveram resultados interessantes, já que houve um alcance de 83.83% de conjuntos de dados otimizados dentre os 198. Outro fator que é importante notar, é o fato de que 183 conjuntos de dados (92.41% dentre os 198) não tiveram sua acurácia piorada em relação ao padrão e 166 tiveram melhora na acurácia com o uso da meta-aprendizagem. Na tabela 5.4 é possível visualizar os resultados para o primeiro experimento.

GENERAL E STATISTICAL		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	166	83.83%
Iguais ao padrão	17 (183)	8.58% (92.41%)
Utilizando a média	155	78.28%
Utilizando a moda	143	72.22%
Utilizando a média dos bons conjuntos	159	80.30%
Utilizando a moda dos bons conjuntos	153	77.27%
Utilizando hiperparâmetro diretamente	159	80.30%
Informações Específicas		
Media de acurácia melhorada		3.67%
Melhora máxima alcançada		20.79%
Melhora mínima alcançada		0.08%

Tabela 5.4: Tabela com o resultado utilizando o conjunto de meta-features general e statistical

De acordo com a tabela 5.4 é possível notar que utilizando a média dos bons conjuntos e os hiperparâmetros diretamente, foram as que tiveram os melhores resultados. Cerca de 80.30% dos conjuntos de dados dentre os 198, foram melhorados. Ainda assim, os outros testes tiveram resultados próximos.

Um ponto interessante para este primeiro experimento, é que alguns conjuntos de dados tiveram conjuntos próximos (com uma distância pequena), como é o caso do *acute inflammations* que entre os 10 conjuntos mais próximos, a distância ficou entre 46.14 e 47.84 e o resultado, portanto, foi interessante: tendo uma melhora de 95.83% para 100% de acurácia. Por outro lado, houveram também alguns conjuntos de dados mas com resultados interessantes; esperava-se que quando a distância fosse muito grande, não houvesse melhora alguma na acurácia mas alguns conjuntos de dados como o *allbp 40707*, o conjunto de dados mais próximo tinha 3378.14 e o último (dentre os 10) 4764.28 e ainda assim, houve uma melhora de 97.61% para 97.74% o que não é uma melhora significativa na acurácia se comparado ao exemplo anterior *acute inflammations* mas ainda assim é uma melhora.

Um segundo ponto que é possível abrir uma discussão sobre, é o fato de que houve uma melhora na acurácia de 20.79% em algum conjunto de dados utilizando a meta-aprendizagem e que em média, a acurácia foi melhorada em 3.67% utilizando a meta-prendizagem.

5.6 Utilizando os conjuntos de *meta-features general* e *info-theory*

Para o segundo experimento, foram utilizados os conjuntos de *meta-features general* e *info-theory*. Foram utilizadas cerca de 24 variáveis para a descrição dos dois conjuntos e para o uso no cálculo da dissimilaridade. Para este segundo experimento, houveram 5 conjuntos melhorados a mais em relação ao anterior e um fator interessante em relação ao anterior, é que a distância de alguns conjuntos de dados diminuíram. Como é o caso do conjunto de dados *acute inflammations* citado anteriormente; que de acordo com os conjuntos do experimento anterior, tinha em torno de 46 a 47.84 de distância e para estes dois conjuntos (*general* e *info-theory*) e utilizando os conjuntos de *meta-features* de agora, desceram para 18 a 26. Fazendo com que eles tivessem uma dissimilaridade maior. Na tabela 5.5 é possível visualizar melhor os resultados para o primeiro experimento.

GENERAL E INFO-THEORY		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	171	86.36%
Iguais ao padrão	14 (185)	7.07% (93.43%)
Utilizando a média	148	74.74%
Utilizando a moda	144	72.72%
Utilizando a média dos bons conjuntos	163	82.32%
Utilizando a moda dos bons conjuntos	154	77.77%
Utilizando hiperparâmetro diretamente	163	82.32%
Informações Específicas		
Media de acurácia melhorada		3.49%
Melhora máxima alcançada		19.19%
Melhora mínima alcançada		1.42%

Tabela 5.5: Tabela com o resultado utilizando o conjunto de meta-features general e info-theory

Com a tabela 5.5 é possível notar que utilizando a média dos bons conjuntos e os hiperparâmetros diretamente, foram as que tiveram os melhores resultados, assim como o experimento anterior, portanto, com valores ainda maiores. O interessante para este conjunto é que por mais que utilizando a média dos hiperparâmetros apenas com bons conjuntos e dos hiperparâmetros diretamente foram melhores que os do experimento anterior, as outras etapas foram piores ou obtiveram apenas uma leve melhora (apenas um conjunto de dados a mais). Além disso, houve uma melhora no que condiz com o número de conjuntos de dados otimizados em relação ao

experimento anterior; o número aumentou para 171 e cobriu 86.36% dos conjuntos de dados do meta-aprendiz. Além disso, houve também uma melhora em relação aos melhores resultados de cada experimento, neste caso, utilizando a média dos bons conjuntos e utilizando os hiperparâmetros diretamente, aumentaram o número de conjuntos de dados otimizados para 163. Em contrapartida, os resultados dos outros testes (utilizando média, moda e moda dos bons conjuntos) não foram tão bons.

Um outro ponto que é possível notar, é que a média de acurácia melhorada em todo o experimento, foi próximo ao do experimento anterior, mas os resultados pareceram mais consistentes em relação a melhora máxima alcançada e a melhora mínima alcançada, já que a melhora mínima aumentou bastante em relação ao experimento anterior.

5.7 Utilizando os conjuntos de *meta-features info-theory* e *statistical*

Para o terceiro experimento, foi utilizado o conjunto de *meta-features statistical* e *info-theory*. Foram utilizadas cerca de 56 variáveis para a descrição dos dois conjuntos e para o uso no cálculo da dissimilaridade. Para este terceiro experimento, houve um resultado inferiror aos anteriores. Isso pode ter acontecido devido ao fato dos dois conjuntos de *meta-features* utilizados não pegarem aspectos bons de cada conjunto de dados fazendo com que a dissimilaridade tivesse resultados bem diferentes dos conjuntos anteriores. É importante ressaltar que em comparação aos experimentos anteriores, este teve uma piora em relação a moda dos hiperparâmetros e alguns outros tiverem resultados similares. Através da tabela 5.6 é possível visualizar os resultados deste experimento. Outro ponto que é possível discutor é que houve um grande aumento nas variáveis que caracterizam os conjuntos de dados, então isso pode ter dado uma espécie de *overfitting* que pode ter atrapalhado um pouco o cálculo da dissimilaridade, mas ainda assim é interessante fazer o experimento com esses conjuntos como forma de comparação com os outros.

Com a tabela 5.6 é possível visualizar que assim como os experimentos anteriores, as etapas utilizando a média dos bons conjuntos e os hiperparâmetros diretamente tiveram um resultado melhor do que as outras etapas. Um ponto que é possível destacar neste experimento em relação aos outros, é que a acurácia máxima alcançada foi de 22.40%, fazendo com que a acurácia fosse melhorada em relação aos experimentos anteriores. Outro ponto é que a média da acurácia continuou similar aos experimentos anteriores.

STATISTICAL E INFO-THEORY		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	166	83.83%
Iguais ao padrão	17 (183)	8.58% (92.41%)
Utilizando a média	149	75.25%
Utilizando a moda	140	70.70%
Utilizando a média dos bons conjuntos	158	79.79%
Utilizando a moda dos bons conjuntos	153	77.27%
Utilizando hiperparâmetro diretamente	158	79.79%
Informações Específicas		
Media de acurácia melhorada		3.60%
Melhora máxima alcançada		22.40%
Melhora mínima alcançada		0.017%

Tabela 5.6: Tabela com o resultado utilizando o conjunto de meta-features info-theory e statistical

5.8 Utilizando os conjuntos de *meta-features general*, *statistical* e *info-theory*

Por fim, para o quarto resultado dos experimentos, foram utilizados os 3 conjuntos de *meta-features* simultâneamente (*general, statistical e info-theory*). Foram utilizadas cerca de 70 variáveis para descrever os 3 conjuntos e para o cálculo da dissimilaridade. Com isso, houve um resultado mediano no que condiz aos experimentos anteriores. Não obteve os melhores resultados e nem os piores (em relação aos anteriores), mas ainda assim, obteve bons resultados conseguindo otimizar pelo menos 83.33% dos conjuntos de dados no pior caso. Além disso, 16 conjuntos de dados não obtiveram piora, fazendo assim, com que 91.41% dos conjuntos de dados tivessem resultados iguais ou melhores que os padrões do XGBoost. Através da tabela 5.7 é possível visualizar os resultados para cada uma das etapas.

GENERAL, STATISTICAL E INFO-THEORY		
Informações Gerais		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Conjuntos otimizados no total	165	83.33%
Iguais ao padrão	16 (181)	8.08% (91.41%)
Utilizando a média	150	75.75%
Utilizando a moda	143	72.22%
Utilizando a média dos bons conjuntos	158	79.79%
Utilizando a moda dos bons conjuntos	151	76.26%
Utilizando hiperparâmetro diretamente	158	79.79%
Informações Específicas		
Media de acurácia melhorada		2.79%
Melhora máxima alcançada		20.0%
Melhora mínima alcançada		0.08%

Tabela 5.7: Tabela com o resultado utilizando o conjunto de meta-features general, info-theory e statistical

É possível notar que não houve nenhuma etapa que chegasse a cobrir 80% dos conjuntos de dados mas os resultados foram aproximados aos anteriores, o que ainda assim pode ser

um resultado significativo, já que estaria otimizando 158 conjuntos de dados dentre os 198 existentes no meta-aprendiz. Além disso, é importante também notar a diferença entre o número de conjuntos otimizados e o melhor resultado dentre os testes de utilizar a média, moda, média dos bons conjuntos, moda dos bons conjuntos e hiperparâmetro diretamente. Basicamente, pode haver uma união entre os conjuntos otimizados entre os testes, o campo que mostra o "número de conjuntos de dados otimizados", mostra o valor real de todos os conjuntos de dados que foram otimizados dentre os 5 testes do experimento.

Outro ponto que serve de comparação com os outros experimentos, é o fato da média de acurácia melhorada que foi um pouco menor em relação aos experimentos anteriores, a acurácia máxima e mínima alcançada foram similares aos experimentos anteriores. Este último experimento teve o intuito de mostrar o quão bom se sairia a otimização na acurácia utilizando muitas variáveis que caracterizassem os conjuntos de dados, já que agora estaria utilizando mais de 70 variáveis para a caracterização dos conjuntos de dados e nos experimentos anteriores, utilizava bem menos.

5.9 Utilizando a média e moda de todos os hiperparâmetros

O intuito deste 2 *baselines* é mostrar se há um viés em utilizar a média dos hiperparâmetros e mostrar se utilizando as médias de todos os hiperparâmetros, realmente teria algum resultado significativo. Sendo assim, da mesma forma que a etapa de calcular a média e moda dos hiperparâmetros foi feita nos experimentos anteriores, dessa vez foi feita com todos os conjuntos de dados do meta-aprendiz (ao invés de apenas os 10 mais próximos). A tabela 5.8 mostra os resultados desta parte dos experimentos.

BASELINES		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Moda de todos os conjuntos	79	39.89%
Média de todos os conjuntos	91	45.95%

Tabela 5.8: Tabela com os resultados dos baselines

Como dito anteriormente, o intuito destes *baselines* é a comparação com os resultados anteriores. É possível notar que com os experimentos anteriores, houve uma melhora de pelo menos 70% e alguns chegaram até mais de 80% dos conjuntos de dados com a acurácia melhorada utilizando a meta-aprendizagem. Já utilizando a moda ou média de todos os conjuntos, foram tidos como resultados apenas 39.89% e 45.95% dos conjuntos de dados, o que seria quase metade dos conjuntos de dados dos experimentos anteriores. Pode-se assim visualizar que a meta-aprendizagem tem uma influência em escolher os conjuntos de dados "certos" para a otimização dos hiperparâmetros.

5.10 Comparativo geral

Para uma melhor visualização e comparativo dentre os experimentos e os experimentos para os *baseline* a tablela 5.9 demonstra uma melhor visualização para todos os resultados, mostrando assim os resultados dos *baselines*, o pior e o melhor resultado utilizado no experimento com a meta-aprendizagem.

RESUMO DOS RESULTADOS		
Baselines		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Moda de todos os conjuntos	79	39.89%
Média de todos os conjuntos	91	45.95%
Experimentos utilizando a meta-aprendizagem		
	Conjuntos de dados melhorados	% de acordo com os 198 conjuntos
Pior resultado	165	83.33%
Melhor resultado	171	86.36%

Tabela 5.9: Tabela com o resumo dos resultados

Primeiramente é importante ressaltar que foram utilizados apenas o pior e o melhor resultado a fim de comparação com o *baseline*, então todos os outros resultados estariam dentro deste intervalo e é irrelevante citar agora. Com isso, é importante notar que o pior resultado dentre os experimentos com a meta-aprendizagem, já otimizou 165 conjuntos de dados dentre os 198 disponíveis no meta-aprendiz; o que já é um resultado significante se comparado com o melhor resultado dos *baselines*, que otimizou 91 conjuntos de dados. Sendo assim, é possível visualizar que o melhor resultado utilizando a meta-aprendizagem, melhora 80 conjuntos de dados a mais do que o melhor resultado do *baseline* conseguindo cobrir mais de 86% dos conjuntos de dados disponíveis no meta-aprendiz, fazendo assim com que o uso da meta-aprendizagem para a recomendação de hiperparâmetros, tivesse uma boa relevância.

Outro ponto que é possível destacar, é o que foi citado na explicação de cada experimento. O melhor resultado, por exemplo, foi utilizando os conjuntos de *meta-features general e info-theory*, tendo 171 dos conjuntos de dados otimizados. Além disso, 14 conjuntos de dados tiveram resultados iguais aos padrão do XGBoost fazendo com que 185 dos 198 conjuntos de dados, tivessem um resultado pelo menos igual ao XGBoost, ou seja, 93.43% dos conjuntos de dados, tiveram resultados maior ou igual ao padrão do XGboost.

Para uma melhor visualização e comparação dos baselines, é possível visualizar a figura 5.1 e 5.2. É possível notar uma diferença entre os resultados do baseline, que foram utilizados como forma de comparação e validação dos resultados, em comparação com os resultados utilizando a meta-aprendizagem (começando da Meta-Característica *General* até o uso das 3 Meta-Características em conjunto).

Groups	N	Min	Q_1	Median	Q_3	Max	Mean	SD
Baseline	2	79	82	85	88	91	85	8.4853
General	6	138	150.25	155.5	160	165	154	9.7365
Statistical	6	139	150	153	156	166	152.8333	8.9536
Info-Theory	6	141	151.5	156.5	160	171	156	10.1587
General e Statistical	6	143	153.5	157	159	166	155.8333	7.705
General e Info-Theory	6	144	149.5	158.5	163	171	157.1667	10.2648
Statistical e Info-Theory	6	140	150	155.5	158	166	154	8.9219
General, Statistical e Info-Theory	6	143	150.25	154.5	158	165	154.1667	7.7309

Figura 5.1: Resumo dos resultados

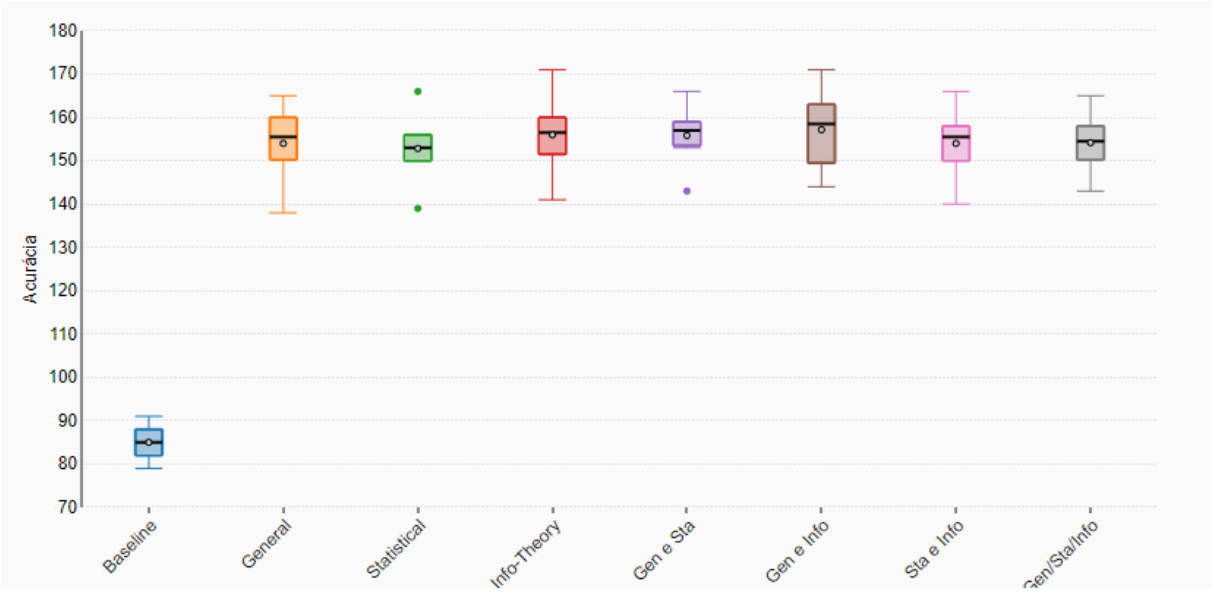


Figura 5.2: Gráfico com o resumo dos resultados

5.11 Ameaças a validade dos resultados

Um dos pontos mais importantes em torno de todo o experimento, é a utilização dos conjuntos de dados, já que são necessários tanto para a utilização do algoritmo XGBoost, tanto a utilização da Meta-Aprendizagem, quanto para a otimização dos hiperparâmetros. Com isso, é importante ressaltar que a maioria dos conjuntos de dados podem ser divididos em algumas categorias básicas como: Dados Numéricos, Categóricos, Séries Temporais e Texto.

O primeiro ponto que já foi citado anteriormente mas que é importante relatar aqui novamente, é que para a utilização do XGBoost, foi necessário a conversão dos dados categóricos em dados numéricos (explicada anteriormente), e resumidamente, as colunas com dados categóricos, foram divididas em colunas com dados numéricos, fazendo com que não influenciasse no conjunto de dados. Portanto, os conjuntos de dados numéricos foram os utilizados neste trabalho, mas os conjuntos de dados categóricos podem ser convertidos nos numéricos sem problemas.

O segundo ponto, é em relação aos dados de séries temporais; de maneira geral, este tipo de conjunto de dados é comumente utilizado em uma observação sequencial feita através do tempo, como a altura do oceano ao longo de um dia por exemplo. Este tipo de conjunto também não foi utilizado neste trabalho.

Por fim, conjuntos de dados utilizando apenas textos ou imagens, também não foram testados. O foco do trabalho foi apenas utilizar conjuntos de dados focados em dados numéricos para a classificação. Uma das propostas para trabalhos futuros é a expansão da recomendação de hiperparâmetros utilizando o XGboost, visando outros tipos de conjuntos de dados e assim, fazer um teste maior dos limites do algoritmo.

Uma outra ameaça vista é que o *XGBoost* é um algoritmo de código aberto e por ser um algoritmo flexível no sentido de poder ser possível a utilização em vários problemas de aprendizagem de máquina, podem ocorrer mudanças nos próprios hiperparâmetros que podem afetar um pouco os estudos e experimentos feitos neste trabalho; os hiperparâmetros utilizados foram os mais sólidos possíveis, utilizando literalmente a essência direta do *XGBoost*, que é algo que é mais difícil de ser mudado ao longo do tempo. Ou seja, o hiperparâmetro *max_depth*, por exemplo, não é um hiperparâmetro que será facilmente mudado já que ele lida diretamente com a ideia inicial do algoritmo.

6

Conclusão

Neste capítulo serão apresentados uma breve descrição dos resultados e suas contribuições, obtidas no desenvolver do trabalho. Além disso, serão comentadas limitações do trabalho e sugestões para trabalhos futuros.

6.1 Resultados e Contribuições

Através deste trabalho foi possível analisar o uso da meta-aprendizagem para a recomendação de hiperparâmetros e além disso, checar o quão bom foram as recomendações para cada etapa: moda, média, moda dos conjuntos de dados mais próximos, média dos conjuntos de dados mais próximos e utilizando o hiperparâmetro diretamente. É importante ressaltar que é possível utilizar outras métricas de similaridade e dissimilaridade, como a similaridade por cosseno ou utilizando distância de *manhattan* e além disso, é possível também achar hiperparâmetros individuais ainda melhores do que os encontrados neste trabalho fazendo assim com o *GridSearch* ou alguma outra técnica de otimização de hiperparâmetros. Contudo, o foco principal do trabalho não é achar o melhor conjunto de hiperparâmetros (o que pode ser uma tarefa custosa) mas sim utilizar conjuntos de hiperparâmetros que já deem um resultado melhor do que o padrão do XGBoost e mostrar que com eles, já são possíveis boas recomendações.

Um segundo ponto que é importante focar, é que utilizando um computador com uma capacidade de processamento normal, com: processamento intel i5, 16 gigabytes de memória RAM, foi uma tarefa custosa de alguns dias de experimentos dentro de um conjunto pequeno de hiperparâmetros e alguns conjuntos de dados pequenos. Para cada um dos conjuntos de dados, foram executados 648 combinações de hiperparâmetros para o GridSearch e além disso, ainda foi feito uma validação, então para executar em todos os 198 conjuntos de dados para cada uma das Meta-Características foram gastos em torno de 12 a 18 horas. Em contrapartida, para a execução da Meta-Aprendizagem, era necessário ao invés de executar 648 combinações

de hiperparâmetros, executar apenas uma métrica de similaridade que envolvia em torno de 20 a 60 variáveis para a aplicação da distância euclidiana, e assim, o teste dos hiperparâmetros; sendo feito em questão de alguns minutos a depender do tamanho do conjunto de dados; com isso, foram concluídos em cerca de 2 a 4 horas. Quando este problema abrange empresas de grande porte, os conjuntos de dados podem ser grandes o suficiente para a busca por bons hiperparâmetros demorarem meses e isso pode ser custoso para a empresa, tanto de tempo quanto financeiramente. Com isso, o intuito deste trabalho é mostrar também que utilizando a meta-aprendizagem, apenas com métricas de dissimilaridade que compara cerca de 50 a 70 variáveis utilizando (podendo aumentar até a mais variáveis, o que não aumentaria muito o custo e nem a complexidade) distância euclidiana, em alguns segundos é possível obter hiperparâmetros que podem aumentar de fato, a acurácia do algoritmo em cima de algum conjunto de dados.

Com isso, de acordo com os resultados mostrados nos experimentos em comparação com o *baseline*, utilizar a meta-aprendizagem para a recomendação dos algoritmos provê um aumento na acurácia dos conjuntos de dados e isso é feito em questão de alguns segundos, diferente da busca profunda padrão por hiperparâmetros que podem durar horas ou até dias. Além disso, é possível expandir as recomendações, usando os 15 conjuntos de dados mais próximos por exemplo ou várias outras métricas de similaridade e dissimilaridade. Além do mais, neste presente trabalho é importante também considerar a facilidade no uso do XGBoost, já que atualmente provém de bibliotecas para a linguagem *Python*, utilizada no trabalho da mesma forma que a extração das *meta-features* também podem ser feitas através do uso de bibliotecas. A distância euclidiana também é algo de fácil entendimento e podem ser feitas em poucas linhas de código, juntamente com a execução do XGBoost e a extração das *meta-features*, fazendo com que se reduza a complexidade na hora de executar todos os passos.

Por fim, de acordo com os resultados apresentados no capítulo 5, os objetivos apresentados puderam ser atingidos, sendo possível a partir da utilização da meta-aprendizagem, utilizando o XGBoost, extração das *meta-features* e métricas de dissimilaridade, ter um resultado significativo para a recomendação de hiperparâmetros. Atualmente, os conjuntos de dados vêm crescendo dia após dia e vem surgindo novos conjuntos de dados, já que o campo da aprendizagem de máquina cresce cada vez mais, por isso, toda forma de evitar complexidade e tempo é válida.

6.2 Limitações do Trabalho

Alguns pontos tiveram uma complexidade maior de se lidar. O primeiro ponto seria o fato de encontrar vários conjuntos de dados bons. Apesar de hoje em dia existirem inúmeros conjuntos de dados, nem todos são válidos e no caso da utilização do XGBoost, não é possível utilizar em conjuntos de dados com variáveis categóricas, portanto, é necessário o tratamento de cada um dos 198 conjuntos de dados utilizados no trabalho. Além disso, foi utilizado o *Jupyter Notebook*

para uma melhor visualização e execução das etapas, portanto, processar todos os conjuntos de dados e executar os exeperimentos em cada um, acaba por ser uma tarefa um pouco mais custosa já que o *Jupyter Notebook* é um pouco mais lento do que *softwares* focados para isso.

Outra limitação do trabalho seria relacionado ao fato de achar conjuntos de dados similares ao trabalhado atualmente. Em um mundo real, pode não ser tão fácil achar conjuntos de dados similares já que eles estão espalhados em várias plataformas diferentes, apesar de que algumas hoje em dia, junta os mais variados. A ideia é se criar um sistema justamente para o uso da meta-aprendizagem, o que será sugerido mais adiante.

6.3 Trabalhos Futuros

Uma das principais propostas deste trabalho é apresentar as recomendações utilizando a meta-aprendizagem e mostrar o quão efetiva e mais rápida é em comparação a busca exaustiva tradicional. Por conta da quantidade de algoritmos de aprendizagem que vem sendo criados e os enormes conjuntos de dados que vem sendo alimentados cada vez mais, é possível utilizar este mesmo experimento com diferentes algoritmos, métricas de similaridade e conjuntos de *meta-features*.

Além disso, é sugerido um sistema focado na meta-aprendizagem. Onde o usuário pode apenas colocar um conjunto de dados lá e automatizadamente, o conjunto de dados pode ser tratado (este passo pode acontecer previamente também) e encontrado conjuntos de dados similares de acordo com *meta-features* escolhidas pelo usuário ou o próprio sistema pode fazer um conjunto de testes que façam isso de forma automatizada também. Além do mais, o sistema pode ser alimentado por vários usuários assim como acontece com a plataforma *kaggle*. Sendo assim, os 198 conjuntos de dados foram extraídos de bases da internet como a própria plataforma *Kaggle*. Uma direção que estudos futuros podem tomar, é extrair base de dados cada vez mais distantes.

Bibliografia

[1] David W Aha e Dennis F Kibler. "Noise-Tolerant Instance-Based Learning Algorithms." Em: *IJCAI*. Citeseer. 1989, pp. 794–799.

[2] E. Alpaydin. *Introduction to Machine Learning, third edition*. Adaptive Computation and Machine Learning series. MIT Press, 2014. ISBN: 9780262325752. URL: https://books.google.com.br/books?id=7f5bBAAAQBAJ.

[3] Joshua D Angrist e Jörn-Steffen Pischke. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2008.

[4] Biggs J. B. *The role of meta-learning in study process.* British Journal of Educational Psychology, 1985.

[5] *Best Machine Learning languages*. https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7. Accessed: 2020-12-10.

[6] Pavel Brazdil. *Metalearning: Applications to Data Mining*. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN: 3642092314.

[7] Leo Breiman. *Arcing the edge*. Rel. téc. Technical Report 486, Statistics Department, University of California at ..., 1997.

[8] Tianqi Chen e Carlos Guestrin. "Xgboost: A scalable tree boosting system". Em: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[9] *CrossValidation*. https://scikit-learn.org/stable/modules/cross_validation.html. Accessed: 2021-01-31.

[10] *Dataset Iris*. https://archive.ics.uci.edu/ml/datasets/iris. Accessed: 2021-11-23.

[11] *Digital Around the World*. https://datareportal.com/global-digital-overview. Accessed: 2021-12-03.

[12] Arthur Guilherme Santos Fernandes et al. “Meta aprendizagem de extraçao de caracterısticas aplicada ao diagnóstico de glaucoma”. Em: *Anais do XIX Simpósio Brasileiro de Computação Aplicada à Saúde*. SBC. 2019, pp. 342–347.

[13] *Grid Search*. https://scikit-learn.org/stable/modules/grid_search.html. Accessed: 2021-04-19.

[14] Simon Haykin e N Network. “A comprehensive foundation”. Em: *Neural networks* 2.2004 (2004), p. 41.

[15] Alexandros Kalousis. “Algorithm selection via meta-learning”. Tese de doutoramento. University of Geneva, 2002.

[16] Kei Kubo, Peter Nelson, Erik Ruggles, James Sheridan, and Sam Tucker. *k Nearest Neighbors*. http://cs.carleton.edu/cs_comps/0910/netflixprize/final_results/knn/index.html. Online; Acessado 22 de Janeiro de 2020.

[17] Ron Kohavi et al. “A study of cross-validation and bootstrap for accuracy estimation and model selection”. Em: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145.

[18] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[19] Christiane Lemke, Marcin Budka e Bogdan Gabrys. “Metalearning: a survey of trends and technologies”. Em: *Artificial intelligence review* 44.1 (2015), pp. 117–130.

[20] *Machine Learning*. https://www.britannica.com/technology/machine-learning. Accessed: 2020-12-10.

[21] Oded Maimon e Lior Rokach. “Data mining and knowledge discovery handbook”. Em: (2005).

[22] Rafael Gomes Mantovani et al. “Meta-learning Recommendation of Default Hyper-parameter Values for SVMs in Classification Tasks.” Em: *MetaSel@ PKDD/ECML*. 2015, pp. 80–92.

[23] Marlesson Santana. *Sistemas de Recomendação: Desvendando uma parte da mágica!* http://medium.com/data-hackers/deep-learning-para-sistemas-de-recomendacao-por-similaridade-d788c126d808/. Online; Acessado 21 de Janeiro de 2020. 2019.

[24] *Meta-Feature General*. https://rdrr.io/cran/mfe/man/general.html. Accessed: 2021-08-15.

[25] *Meta-Feature Info-Theory*. https://rdrr.io/cran/mfe/man/infotheo.html. Accessed: 2021-08-15.

[26] *Meta-Feature Statistical*. https://rdrr.io/cran/mfe/man/statistical.html. Accessed: 2021-08-15.

[27] Donald Michie, David J Spiegelhalter e Charles C Taylor. "Machine learning, neural and statistical classification". Em: (1994).

[28] Tom M Mitchell. "Does machine learning really work?" Em: *AI magazine* 18.3 (1997), pp. 11–11.

[29] Maria Carolina Monard e José Augusto Baranauskas. "Conceitos sobre aprendizado de máquina". Em: *Sistemas inteligentes-Fundamentos e aplicações* 1.1 (2003), p. 32.

[30] Natasha Sharma. *Importance of Distance Metrics in Machine Learning Modelling*. https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d. Online; Acessado 21 de Janeiro de 2020. 2019.

[31] *Popular regression algorithms*. https://www.jigsawacademy.com/popular-regression-algorithms-ml/. Accessed: 2020-12-10.

[32] Foster Provost e Ron Kohavi. "On applied research in machine learning". Em: *MACHINE LEARNING-BOSTON*- 30 (1998), pp. 127–132.

[33] *Python Meta-Feature Extractor*. https://pypi.org/project/pymfe/. Accessed: 2021-01-27.

[34] Stuart J Russell e Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[35] Jürgen Schmidhuber. "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook". Tese de doutoramento. Technische Universität München, 1987.

[36] Vânia Patrıcia Pereira Serra. "Meta-aprendizagem no problema de seleção de algoritmo de Análise Classificatória". Em: (2017).

[37] Phil Simon. *Too big to ignore: the business case for big data*. Vol. 72. John Wiley & Sons, 2013.

[38] Bruno Feres de Souza. "Meta-aprendizagem aplicada à classificação de dados de expressão gênica". Tese de doutoramento. Universidade de São Paulo, 2010.

[39] Dan Steinberg e Phillip Colla. "CART: classification and regression trees". Em: *The top ten algorithms in data mining* 9 (2009), p. 179.

[40] *Top 10 Machine Learning Algorithms*. https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies. Accessed: 2020-12-10.

[41] *Undefitting and Overfitting*. https://didatica.tech/underfitting-e-overfitting/. Accessed: 2021-11-23.

[42] Martijn Van Otterlo e Marco Wiering. "Reinforcement learning and markov decision processes". Em: *Reinforcement Learning*. Springer, 2012, pp. 3–42.

[43] Joaquin Vanschoren. "Meta-learning: A survey". Em: *arXiv preprint arXiv:1810.03548* (2018).

[44] *XG Boost enhancement*. https://medium.com/@ODSC/xgboost-enhancement-over-gradient-boosting-machines-73abafa49b14. Accessed: 2021-01-10.

[45] *XGBoost about*. https://xgboost.ai/about. Accessed: 2021-01-10.

[46] *XGBoost Model*. https://xgboost.readthedocs.io/en/latest/tutorials/model.html. Accessed: 2020-12-10.

[47] *XGBoost parameter documentation*. https://xgboost.readthedocs.io/en/latest/parameter.html. Accessed: 2020-12-10.

[48] *XGBoost Project*. https://pypi.org/project/xgboost/. Accessed: 2020-12-10.

[49] *XGBoost top kaggle algorithm*. https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html. Accessed: 2020-12-10.

[50] *XGBoost Tuning*. https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html. Accessed: 2020-12-10.

Apêndice

FUNÇÕES PRINCIPAIS

```
import pandas as pd
import numpy as np
import math
import xgboost as xgb
import json

import matplotlib.pyplot as plt
from os import listdir
from os.path import isfile, join

from scipy import spatial
from pymfe.mfe import MFE

from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

json_result = {}
json_result['datasets'] = []
json_mfe_result = {}
json_mfe_result['datasets'] = []
onlyfiles = [f for f in listdir('RealDatasets') if isfile(join('RealDatasets', f))]
mfe_groups = ['general', 'statistical', 'info-theory']

def checkIfExistsInJson(data_name):
    file = open('result.json')
    d = json.load(file)
    if('datasets' in d):
            for p in d['datasets']:
                if(p['name'] == data_name):
                    print("Já tem!")
                    return True
    return False

def build_json_object(dataset_name, default_params_result, best_params_result, params):
    json_result['datasets'].append({
        'name': dataset_name,
        'default_params_result': default_params_result,
        'best_params_result': best_params_result,
        'params': {
            'max_depth': params['max_depth'],
```

```
            'min_child_weight': params['min_child_weight'],
            'gamma': params['gamma'],
            'subsample': params['subsample'],
            'colsample_bytree': params['colsample_bytree']
        }
    })

def build_json_mfe_object(dataset_name, mfe):
    json_mfe_result['datasets'].append({
        'name': dataset_name,
        'mfe': mfe
    })

def mfe_extract(dataframe):
    y = dataframe['target'].tolist()
    X = dataframe.drop('target', axis=1).values

    mfe = MFE(mfe_groups)
    mfe.fit(X, y)
    ft = mfe.extract()

    result = {}
    for i in range(0, len(ft[0])):
        if(str(ft[1][i]) != 'nan'):
            result[ft[0][i]] = np.float64(ft[1][i])

    return result

    def xgboost(dataframe):
    estimator = xgb.XGBClassifier()

    X = dataframe[dataframe.columns[:-1]]
    y = dataframe['target']

    k_folds = 5
    for i in df['target'].value_counts():
        if(i < k_folds):
            k_folds = i

    if(k_folds == 1):
        return(0.0, 0.0, 0.0)
    # Parâmetros reduzidos
     parameters = {
         'max_depth': range(3, 8, 1), # Maximum depth = more overfit
         'min_child_weight': range(1, 3, 1),
         'gamma': [0, 0.2, 0,5, 1],
         'subsample': [0.2, 0.5, 1.0],
         'colsample_bytree': [0.2, 0.5, 1.0]
     }

    default_parameters = {
        'max_depth': [6],
        'min_child_weight': [1],
        'gamma': [0],
        'subsample': [1],
        'colsample_bytree': [1]
    }

```

```
100     grid_search = GridSearchCV(
101         estimator=estimator,
102         param_grid=default_parameters,
103         cv=k_folds
104     )
105     grid_search.fit(X, y)
106 
107     default_value = grid_search.best_score_*100
108 
109      grid_search = GridSearchCV(
110          estimator=estimator,
111          param_grid=parameters,
112          cv=k_folds
113      )
114     grid_search.fit(X, y)
115 
116     value = grid_search.best_score_*100
117 
118     # Default params accuracy / best params accuracy / params
119     return (default_value, value, grid_search.best_params_)
```

FUNÇÃO DE MÉTRICA DE SIMILARIDADE EUCLIDIANA

```
def euclidean(v1, v2):
    return sum((p-q)**2 for p, q in zip(v1, v2)) ** .5

def calculates_metafeature_similarity(listA, listB):
    arrayAuxA = []
    arrayAuxB = []

    for i in listA['mfe']:
        for j in listB['mfe']:
            if(i == j):
                arrayAuxA.append(float(listA['mfe'][i]))
                arrayAuxB.append(float(listB['mfe'][j]))

    return euclidean(arrayAuxA, arrayAuxB)
```

EXECUÇÃO DOS EXPERIMENTOS

```
for i in range(0, len(onlyfiles)):
    print('Dataset: ', i, ' - ', onlyfiles[i])
    path = ('RealDatasets/' + onlyfiles[i])
    df = pd.read_csv(path)
    df.columns = [*df.columns[:-1], 'target']

    result = xgboost(df)
    mfe = mfe_extract(df)

    if(checkIfExistsInJson(onlyfiles[i]) == False and result[0] != 0):
        build_json_object(onlyfiles[i], result[0], result[1], result[2])
        build_json_mfe_object(onlyfiles[i], mfe)

    build_json_mfe_object(onlyfiles[i], mfe)

with open('result.json', 'w') as outfile:
    json.dump(json_result, outfile)
with open('mfe_result.json', 'w') as outfile:
    json.dump(json_mfe_result, outfile)

import copy
file = open('result.json')
file_mfe = open('mfe_result.json')
data = json.load(file)
data_mfe = json.load(file_mfe)
estimator = xgb.XGBClassifier()
optimized_datasets = 0

def testHyperparams(dataframe, data):
    params = data['params']

    X = dataframe[dataframe.columns[:-1]]
    y = dataframe['target']

    for i in params:
        if(type(params[i]) != type([])):
```

```
            params[i] = [(params[i])]

    grid_search = GridSearchCV(
        estimator = estimator,
        param_grid = params
    )
    grid_search.fit(X, y)
    answer = grid_search.best_score_*100

    return answer

def getDatasetInfo(name):
    d = ''
    for p in data['datasets']:
        if(p['name'] == name):
            d = p

    return d

def getDatasetMfeInfo(name):
    d = ''
    for p in data_mfe['datasets']:
        if(p['name'] == name):
            d = p

    return d

def getHyperparamsMean(hyperparams):
    params_mean = copy.deepcopy(hyperparams[0]['params'])

    for i in range(1, len(hyperparams)):
        params = hyperparams[i]['params']
        for j in params:
            params_mean[j][0] += params[j][0]
    for i in params_mean:
        params_mean[i][0] /= len(hyperparams)
        if(i == 'max_depth'):
            params_mean[i][0] = int(round(params_mean[i][0]))

    return {"params": params_mean}

def getMode(a):
    return max(set(a), key = a.count)

def getHyperparamsMode(hyperparams):
    params_mode = {
        "max_depth": [],
        "min_child_weight": [],
        "gamma": [],
        "subsample": [],
        "colsample_bytree": []
    }

    for i in hyperparams:
        for j in i['params']:
            params_mode[j].append(i['params'][j][0])

    for i in params_mode:
```

```
95          params_mode[i] = [getMode(params_mode[i])]
96
97      return {"params": params_mode}
98
99
100 def testMeanAndModeAllHyperparams(df, hyperparams):
101     result = []
102     hyperparams_mean = []
103     hyperparams_mode = []
104
105     for i in range(0, len(hyperparams)):
106         atual = testHyperparams(df, getHyperparamsMean(hyperparams[0:i+1].copy()))
107         hyperparams_mean.append(atual)
108         atual = testHyperparams(df, getHyperparamsMode(hyperparams[0:i+1].copy()))
109         hyperparams_mode.append(atual)
110
111     return(hyperparams_mean, hyperparams_mode)
112
113 def testMeanAndModeGoodHyperparams(df, good_hyperparams):
114     result = []
115     hyperparams_mean = []
116     hyperparams_mode = []
117
118     for i in range(0, len(good_hyperparams)):
119         atual = testHyperparams(df, getHyperparamsMean(good_hyperparams[0:i+1].copy()))
120         hyperparams_mean.append(atual)
121         atual = testHyperparams(df, getHyperparamsMode(good_hyperparams[0:i+1].copy()))
122         hyperparams_mode.append(atual)
123
124     return(hyperparams_mean, hyperparams_mode)
125
126 def getBestNHyperparams(name, n):
127     answer = []
128     ordem = []
129     d1 = getDatasetInfo(name)
130     d1_mfe = getDatasetMfeInfo(name)
131
132     results = []
133     good_datasets = []
134
135     for i in range(0, len(onlyfiles)):
136         d2 = getDatasetInfo(onlyfiles[i])
137         d2_mfe = getDatasetMfeInfo(onlyfiles[i])
138
139         if(onlyfiles[i] == name):
140             continue
141
142         if(d2):
143             ordem.append((calculates_metafeature_similarity(d1_mfe, d2_mfe), onlyfiles[i]))
144
145     ordem.sort()
146
147     qt = 0
148
149     hyperparams = []
150     good_hyperparams = []
151
152     path = ('RealDatasets/' + name)
```

```
153     df = pd.read_csv(path)
154     df.columns = [*df.columns[:-1], 'target']
155 
156     j = 0
157     for i in ordem[:n]:
158         d2 = getDatasetInfo(i[1])
159         atual = testHyperparams(df, d2)
160         hyperparams.append(d2)
161 
162         maior = False
163         if(float(atual) > float(d1['default_params_result'])):
164             maior = True
165             qt += 1
166 
167         if(maior == True):
168             good_hyperparams.append(d2)
169 
170         d = {"name": i[1],
171             "distance": i[0],
172             "result_with_hiperparameter": atual,
173             "greater_than_default": maior
174             }
175 
176         results.append(d)
177         j += 1
178 
179     result_all_hyperparams = testMeanAndModeAllHyperparams(df, hyperparams)
180     result_good_hyperparams = testMeanAndModeGoodHyperparams(df, good_hyperparams)
181 
182     results_according_k = []
183 
184     for i in range(0, len(result_all_hyperparams[0])):
185         results_according_k.append({
186             "k": i,
187             "mean": result_all_hyperparams[0][i],
188             "mode": result_all_hyperparams[1][i],
189             "mean_good_datasets": result_good_hyperparams[0][i] if i < len(result_good_hyperparams[0
190             "mode_good_datasets": result_good_hyperparams[1][i] if i < len(result_good_hyperparams[1
191         })
192 
193     if(qt > 0):
194         global optimized_datasets
195         optimized_datasets += 1
196 
197 
198     best_mean_result = -1
199     best_mean_k_number = -1
200     best_mode_result = -1
201     best_mode_k_number = -1
202     best_mean_good_datasets = -1
203     best_mean_good_datasets_k_number = -1
204     best_mode_good_datasets = -1
205     best_mode_good_datasets_k_number = -1
206     best_result_using_hyperparameter_directly = -1
207 
208     for i in results:
209         best_result_using_hyperparameter_directly = max(best_result_using_hyperparameter_directly, i
210 
```

```
    for i in results_according_k:
        if(i['mean'] > best_mean_result):
            best_mean_result = i['mean']
            best_mean_k_number = i['k']

        if(i['mode'] > best_mode_result):
            best_mode_result = i['mode']
            best_mode_k_number = i['k']

        if(i['mean_good_datasets'] > best_mean_good_datasets):
            best_mean_good_datasets = i['mean_good_datasets']
            best_mean_good_datasets_k_number = i['k']

        if(i['mean_good_datasets'] > best_mode_good_datasets):
            best_mode_good_datasets = i['mode_good_datasets']
            best_mode_good_datasets_k_number = i['k']



    return {
        "results_each_hyperparameter": results,
        "results_according_k": results_according_k,
        "best_mean": best_mean_result,
        "best_mean_k_number": best_mean_k_number,
        "best_mode": best_mode_result,
        "best_mode_k_number": best_mode_k_number,
        "best_mean_good_datasets": best_mean_good_datasets,
        "best_mean_good_datasets_k_number": best_mean_good_datasets_k_number,
        "best_mode_good_datasets": best_mode_good_datasets,
        "best_mode_good_datasets_k_number": best_mode_good_datasets_k_number,
        "best_result_using_hyperparameter_directly": best_result_using_hyperparameter_directly

    }
```

ANÁLISE DOS RESULTADOS

```
improved_datasets_mean = 0
improved_datasets_mode = 0
improved_datasets_directly = 0

improved_datasets_mean_good = 0
improved_datasets_mode_good = 0

mean_count_according_k = []
mode_count_according_k = []
mean_good_count_according_k = []
mode_good_count_according_k = []

for i in range(0, len(d)):
    if(d[i]['best_mean'] > d[i]['default_result']):
        improved_datasets_mean += 1
    if(d[i]['best_mode'] > d[i]['default_result']):
        improved_datasets_mode += 1
    if(d[i]['best_result_using_hyperparameter_directly'] > d[i]['default_result']):
        improved_datasets_directly += 1

```

```
    if(d[i]['best_mean_good_datasets'] > d[i]['default_result']):
        improved_datasets_mean_good += 1
    if(d[i]['best_mode_good_datasets'] > d[i]['default_result']):
        improved_datasets_mode_good += 1

    for j in d[i]['results_according_k']:
        if(j['mean'] > d[i]['default_result']):
            mean_count_according_k.append(j['k'])
        if(j['mode'] > d[i]['default_result']):
            mode_count_according_k.append(j['k'])
        if(j['mean_good_datasets'] > d[i]['default_result']):
            mean_good_count_according_k.append(j['k'])
        if(j['mode_good_datasets'] > d[i]['default_result']):
            mode_good_count_according_k.append(j['k'])



print('Improved Datasets with Mean: ', improved_datasets_mean)
print('Improved Datasets with Mode: ', improved_datasets_mode)
print('Improved Datasets with mean using good datasets: ', improved_datasets_mean_good)
print('Improved Datasets with mode using good datasets: ', improved_datasets_mode_good)
print('Improved Datasets Directly: ', improved_datasets_directly)
```

DEADLINE 1 - Calcular a média dos hiperparâmetros

```
file = open('result.json')
d = json.load(file)

hyperparam_deadline = {'max_depth': 0, 'min_child_weight': 0, 'gamma': 0, 'subsample': 0, 'colsampl


print(d['datasets'][0]['params'])
for i in d['datasets']:
    hyperparam_deadline['max_depth'] += i['params']['max_depth']
    hyperparam_deadline['min_child_weight'] += i['params']['min_child_weight']
    hyperparam_deadline['gamma'] += i['params']['gamma']
    hyperparam_deadline['subsample'] += i['params']['subsample']
    hyperparam_deadline['colsample_bytree'] += i['params']['colsample_bytree']

hyperparam_deadline['max_depth'] = int(round(hyperparam_deadline['max_depth']/len(d['datasets'])))
hyperparam_deadline['min_child_weight'] /= len(d['datasets'])
hyperparam_deadline['gamma'] /= len(d['datasets'])
hyperparam_deadline['subsample'] /= len(d['datasets'])
hyperparam_deadline['colsample_bytree'] /= len(d['datasets'])

hyperparam_deadline['max_depth'] = [hyperparam_deadline['max_depth']]
hyperparam_deadline['min_child_weight'] = [hyperparam_deadline['min_child_weight']]
hyperparam_deadline['gamma'] = [hyperparam_deadline['gamma']]
hyperparam_deadline['subsample'] = [hyperparam_deadline['subsample']]
hyperparam_deadline['colsample_bytree'] = [hyperparam_deadline['colsample_bytree']]

optimized_datasets = 0


for i in range(0, len(onlyfiles)):
```

```
    if(i == 48):
        continue
    print(i, ' - ', onlyfiles[i])
    path = ('RealDatasets/' + onlyfiles[i])
    df = pd.read_csv(path)
    df.columns = [*df.columns[:-1], 'target']

    X = df[df.columns[:-1]]
    y = df['target']

    k_folds = 5
    for j in df['target'].value_counts():
        if(j < k_folds):
            k_folds = j
            break

    answer = 0
    if(k_folds != 1):
        grid_search = GridSearchCV(
            estimator = estimator,
            param_grid = hyperparam_deadline,
            cv = k_folds
        )
        grid_search.fit(X, y)
        answer = grid_search.best_score_*100

        answer_default = xgboost(df)
    else:
        continue

    if(answer_default[0] > answer):
        optimized_datasets+=1
    print(answer_default[0], ' - ', answer)

print('Optimized Datasets: ', optimized_datasets)
```

DEADLINE 2 - Moda de todos os hiperparâmetros

```
file = open('result_1.json')
d = json.load(file)

hyperparam_deadline = {'max_depth': [], 'min_child_weight': [], 'gamma': [], 'subsample': [], 'cols


print(d['datasets'][0]['params'])
for i in d['datasets']:
#       print(i['params'])
    hyperparam_deadline['max_depth'].append(i['params']['max_depth'])
    hyperparam_deadline['min_child_weight'].append(i['params']['min_child_weight'])
    hyperparam_deadline['gamma'].append(i['params']['gamma'])
    hyperparam_deadline['subsample'].append(i['params']['subsample'])
    hyperparam_deadline['colsample_bytree'].append(i['params']['colsample_bytree'])


hyperparam_deadline['max_depth'] = [max(set(hyperparam_deadline['max_depth']), key=hyperparam_deadli
hyperparam_deadline['min_child_weight'] = [max(set(hyperparam_deadline['min_child_weight']), key=hyp
hyperparam_deadline['gamma'] = [max(set(hyperparam_deadline['gamma']), key=hyperparam_deadline['gamm
hyperparam_deadline['subsample'] = [max(set(hyperparam_deadline['subsample']), key=hyperparam_deadli
hyperparam_deadline['colsample_bytree'] = [max(set(hyperparam_deadline['colsample_bytree']), key=hyp

optimized_datasets = 0

for i in range(0, len(onlyfiles)):
    if(i == 48):
        continue
    print(i, ' - ', onlyfiles[i])
    path = ('RealDatasets/' + onlyfiles[i])
    df = pd.read_csv(path)
    df.columns = [*df.columns[:-1], 'target']

    X = df[df.columns[:-1]]
    y = df['target']

    k_folds = 5
    for j in df['target'].value_counts():
        if(j < k_folds):
            k_folds = j
            break

    answer = 0
    if(k_folds != 1):
        grid_search = GridSearchCV(
            estimator = estimator,
            param_grid = hyperparam_deadline,
            cv = k_folds
        )
        grid_search.fit(X, y)
        answer = grid_search.best_score_*100

        answer_default = xgboost(df)
    else:
        continue

```

```
56      if(answer_default[0] > answer):
57          optimized_datasets+=1
58      print(answer_default[0], ' - ', answer)
59
60  print('Optimized Datasets: ', optimized_datasets)
```