Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Recomendação de Algoritmos em Fluxos de Dados com Mudança de Conceito

Jáder Martins Camboim de Sá

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Luís Paulo Faina Garcia

Brasília
2020

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Recomendação de Algoritmos em Fluxos de Dados com Mudança de Conceito

Jáder Martins Camboim de Sá

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Luís Paulo Faina Garcia (Orientador)
CIC/UnB

Prof. Dr. Geraldo Pereira Rocha Filho
CIC/UnB

Prof. Dr. Marcelo Antonio Marotta
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 16 de dezembro de 2020

Dedicatória

Dedico esse trabalho a minha família e amigos que sempre me forneceram o suporte necessário para trilhar esta graduação, em especial a meu pai que sempre me incentivou, motivou e cobrou no meu processo de educação e minha mãe pelo suporte e todos os cuidados para que eu pudesse me dedicar aos estudos.

Agradecimentos

Agradeço primeiramente ao meu orientador que sempre foi solicito e didático na concepção desse trabalho e também aos diversos educadores e organizações que disponibilizaram gratuitamente conteúdos de extrema qualidade que ajudaram por toda minha graduação e também na concepção desse trabalho.

Resumo

Muitas companhias vêm tirando proveito de mineração de dados para identificar informações valiosas em conjuntos de dados massivos gerados em alta frequência, o chamado *Big Data*. Técnicas de Aprendizado de Máquina podem ser aplicadas para descoberta de informação, visto que podem extrair padrões dos dados para induzir modelos que preverão eventos futuros. Entretanto, ambientes dinâmicos e progressivos comumente geram fluxos de dados não estacionários. Logo, modelos treinados nesse cenário costumam perecer com o tempo pela sazonalidade ou mudança de conceito. O retreinamento periódico pode ajudar, mas um espaço de hipóteses fixo pode não ser o mais apropriado ao fenômeno. Uma solução alternativa é usar meta-aprendizado para uma contínua seleção de algoritmos em ambientes que mudam com o tempo, escolhendo o viés que melhor condiz com os dados atuais. Nesse trabalho, apresentamos um *framework* aprimorado para seleção de algoritmos em fluxos de dados baseado no MetaStream. Nossa abordagem usa meta-aprendizado e aprendizado incremental para ativamente selecionar o melhor algoritmo para o presente conceito em ambientes que mudam com o tempo. Ao contrário de trabalhos prévios, nós usamos uma coleção diversificada de meta-atributos estado-da-arte e uma abordagem de aprendizado incremental para o nível meta baseada no algoritmo LightGBM. Os resultados mostram que essa nova estratégia pode aprimorar a acurácia de recomendação do melhor algoritmo em dados que mudam com o tempo.

Palavras-chave: meta-aprendizado, aprendizado de máquina, mudança de conceito, fluxos de dados

Abstract

In the last decades, many companies have had a growing interest in the "digital oil", also called Big Data. Data mining has been applied in these massive volumes of data to obtain valuable information for clients and industries worldwide. Machine Learning, a prominent technique for data mining, can be used to extract patterns from data and induce models to predict future events. Still, complex environments that are constantly evolving usually generate non-stationary data streams. Thus, these models may perish in this scenario due to concept drift. Retraining periodically can help, but the algorithm bias may no longer be appropriate. A response to this is to use meta-learning for regular algorithm selection in time-changing environments, choosing the hypothesis space that best suits the current data. In this work, we enhanced MetaStream, a framework for data stream algorithm selection, though a rich set of state-of-the-art meta-features, and an incremental learning approach in the meta-level based on LightGBM, combining this to actively select the best algorithm for the current concept in a time-changing environment. The results show that this new strategy can improve the recommendation accuracy of the best algorithm in time-changing data.

Keywords: metalearning, machine learning, concept shift, datastreams

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Sistemas digitais estão presentes no dia-a-dia das pessoas nas mais diversas formas. *Smartphones* com seus diversos aplicativos, *smart things* (TVs, relógios, etc.), sensores, computadores de bordo e muitos outros dispositivos exemplificam um cenário que, em função do progresso tecnológico, apresentará um crescimento ainda maior para os próximos anos [1, 2]. Tais sistemas, por estarem conectados à redes internas e externas, podem consumir e gerar um grande volume de dados. Esse fenômeno de geração e armazenamento de uma quantidade massiva de dados recebeu o nome "*Big Data*" [3, 4].

Essa grande quantidade de informação, mesmo sem qualquer filtragem ou supervisão, possibilitou tarefas de grande valor comercial, como recomendação de conteúdo e processamento de linguagem natural [5]. Desse modo, se tornou um dos bens mais valiosos do mundo moderno [6]. Companhias como Google®, Facebook®, Netflix® e Amazon® estão entre as empresas que competem por esse bem digital com o objetivo de extrair informações valiosas aos seus usuários e ao mercado como um todo [7, 8].

Fluxos de dados, ou *Data Streams*, são ambientes típicos em infraestruturas de *Big Data*, ao contrário de dados em lote, que tem inicio, fim e tamanho bem definidos, um fluxo de dados é uma fonte contínua de dados, exemplos são redes de sensores e monitoramento, registros de sistemas web, transações em comércio eletrônico, entre outros. Do ponto de vista de engenharia, pela natureza dos dados, há imposições limitantes quanto a leitura e manipulação, o que torna os processos de mineração para esses ambientes uma tarefa não trivial [9].

Outro aspecto limitado para esses sistemas são as técnicas de Mineração de Dados [9], dentre elas em especial o Aprendizado de Máquina (AM), campo que estuda métodos para a extração de padrões dos dados, para que esse conhecimento seja aplicado em tarefas futuras [10, 11]. Como todo método indutivo, algoritmos tradicionais de AM assumem duas premissas que devem ser satisfeitas para sua validade: (i) o futuro deve se comportar como o passado e (ii) eventos futuros devem ser independentes de eventos passado [12].

Entretanto, em um fluxo de dados, a distribuição que gera os dados usualmente está mudando com o tempo e dependências temporais podem ocorrer. Logo, essas premissas podem não se manter e o desempenho dos modelos induzidos podem perecer com o passar do tempo, causando uma má experiência aos usuários desses sistemas [9, 13].

Diversas técnicas foram desenvolvidas para lidar com os problemas ocorridos em ambientes com mudanças temporais: detecção de mudança de conceito [14, 15], retreino periódico de algoritmo [16, 17, 18] e novas abordagens de algoritmos para lidarem com esse contexto [19, 20, 21]. Por exemplo, alguns métodos de detecção de mudança de conceito [22] são baseados em alarmes que apontam quando há diferença maior que um dado limiar entre o desempenho do modelo atual e do melhor modelo. Outros métodos, como a indução periódica de modelos, podem não ser suficientes para resolver esses problemas, já que o espaço de hipótese fixado do algoritmo pode não mais ser apropriado ao problema [23].

Meta-Aprendizado (MtA) é uma técnica proeminente para resolver essas limitações por meio da detecção de mudança de conceito e recomendação de algoritmos para melhorar a predição [24, 25, 26]. Uma das abordagens para uso de MtA é a construção de meta-dados [27, 28] baseado em uma conjunto de características descritivas, nomeados meta-atributos, do problema sob análise [29, 30]. Esses meta-atributos são extraídos e combinados criando um meta-exemplo. Diferentes algoritmos de AM são aplicados ao problema, e seus desempenhos são utilizados para rotular os meta-exemplos. Esse procedimento é executado para o fluxo de dados em diferentes pontos no tempo, resultando em um conjunto de meta-dados, o qual pode ser usado para induzir um modelo capaz de predizer qual será o algoritmo mais apropriado para eventos futuros [31].

O MetaStream, é um *framework*[1] baseado em MtA, que mantém o treinamento contínuo de dois grupos de algoritmos por janelas deslizantes. Algoritmos a nível base, que vão efetivamente predizer a variável alvo do fluxo de dados e um algoritmo a nível meta, que irá recomendar qual melhor algoritmo para ser usado no momento atual do fluxo [32, 23].

Nesse trabalho, o MetaStream é aprimorado pela inclusão de meta-atributos mais modernos e informativos [29], e a inclusão de aprendizado incremental no nível meta, propondo o LightGBM [33] como meta-classificador. Diferente do trabalho original em [23], no qual eram usadas apenas medidas estatísticas simples como média, desvio padrão, curtose, etc. para descrever os dados, nesse trabalho estendemos para medidas estatísticas muito mais complexas, como esparsidade, Wilks' Lambda, e outras medidas de grupos extraídos de modelos induzidos (*Landmarking* e *Model Based*). O LightGBM como meta-classificador fornece diversas propriedades interessantes, apresenta desempenho preditivo superior em relação a algoritmos semelhantes aliado a um menor consumo de recursos [34],

[1]Modelo de aplicação em software.

lida nativamente com atributos faltantes, essencial para esse contexto, e por fim, permite o aprendizado incremental que reduz drasticamente o tempo de processamento e memória para induzir um novo modelo. Seguimos então para investigar se meta-atributos de ponta e aprendizado incremental provido pelo LightGBM são capazes de predizer de forma mais precisa que o método base e aprimorar o desempenho geral do sistema aprendiz.

1.1 Hipótese

O problema de mudança de conceito, isto é, uma alteração na função geradora dos dados, pode afetar a performance do sistema de aprendizado em vigência significativamente, dado que o seu espaço de hipóteses pré-definido pode não aproximar tão bem essa nova distribuição dos dados. Portanto, essa constatação inicia a motivação de pesquisa e nesta seção definiremos a hipótese, objetivos e a pergunta de pesquisa que esse trabalho buscará responder.

Em fluxos de dados, ambientes dinâmicos que geram dados continuamente, é comum a presença de mudança de conceito [35], logo, temos como hipótese inicial para esse trabalho que a recomendação continua de vieses distintos, apropriados as características atuais dos dados, resulta em uma sistema de aprendizado com um melhor poder preditivo em função de um único algoritmo definido *apriori*.

1.2 Objetivos

Embora no longo prazo esses algoritmos desempenhem em média de forma similar para o fluxo de dados, faz sentido realizar a seleção dinâmica se em períodos menores do tempo, um algoritmo domina o desempenho em relação aos outros. Pois, se um viés é dominante em todo momento observado, não haverá ganhos por essa seleção. Desta forma, esse trabalho tem como objetivo avaliar se a recomendação de algoritmos gera ganhos preditivos ao longo do tempo.

Traçado o objetivo desse trabalho, temos então as seguintes perguntas de pesquisa:

- **RQ1:** Meta-atributos mais modernos, "estado-da-arte" possibilitam uma recomendação do algoritmo mais adequado em um dado momento, levando a ganhos significativos de desempenho ao longo do tempo?
- **RQ2:** A nível meta, em que o conceito pode se manter ao longo do tempo, é possível obter poder preditivo equivalente para um algoritmo incremental, que tem menor consumo de recursos, em relação a um não-incremental?

1.3 Considerações

A partir dos estudos desse trabalho também foi concebido um artigo em conjunto com os professores Dr. André Rossi, Dr. Gustavo Batista e Dr. Luís Paulo Garcia (orientador desta monografia), que foi aceito para a conferência *"25th International Conference on Pattern Recognition (ICPR), 2020"*, classificada como QUALIS A2 (2016), com título *"Algorithm Recommendation for Data Streams"*, que está em anexo ao final do trabalho.

1.4 Estrutura do Trabalho

O restante desse texto esta estruturado da seguinte forma: o Capítulo 2 contém o que já tem sido feito para resolver o problema aqui proposto e as bases teóricas das soluções aqui empregas, o Capítulo 3 apresenta a metodologia, constituída do *framework* MetaStream, modificações aplicadas a ele e as configurações do experimento, o Capítulo 4 apresenta os resultados e analises dos experimentos conduzidos, e por fim, o Capítulo 5 conclui a monografia.

Capítulo 2

Aprendizado em Fluxos de Dados

A possibilidade de aprender e fazer previsões com dados data de ao menos dois séculos, como os trabalhos de epidemiologia de Snow ou a predição de movimentos planetários por Gauss no século 19 [36, 37, 38]. Porém, o uso de modelos extremamente complexos, com maior capacidade de inferência só se tornou possível devido a um fenômeno recente, o *Big Data* [3, 5]. Estendendo a abordagem tradicional de inferência, da regressão linear e análise de dados, o AM, que pode representar e modelar fenômenos muito mais complexos, tem chamado muita atenção com seu rápido progresso e resultados práticos desejáveis [39, 40]. Nesse capítulo iremos apresentar a teoria por trás desses dois fenômenos tecnológicos, o AM e o *Big Data*, representado neste trabalho por Fluxos de Dados.

2.1 Aprendizado de Máquina

No contexto de Inteligência Artificial, a Aprendizagem, ou o Aprendizado de Máquina (AM), é o estudo e a concepção de agentes que melhoram o seu desempenho nas tarefas futuras após fazer observações sobre o mundo [38, 41]. Um dos modelos mais adotados para descrever e especificar a tarefa de aprendizado por um computador é descrito formalmente em [10], este o apresenta da seguinte forma:

> **Definição 1** Um programa de computador (agente), *aprende* de uma experiência $\boldsymbol{E}$ a respeito de alguma classe de tarefa $\boldsymbol{T}$ e medida de avaliação $\boldsymbol{P}$, se o desempenho na tarefa $\boldsymbol{T}$, medido por $\boldsymbol{P}$, melhora com a experiência $\boldsymbol{E}$.

No modelo supervisionado[1], o qual lidaremos exclusivamente neste trabalho, a experiência $\boldsymbol{E}$ é um registro $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, em que x_i é um valor real computável ($\mathbb{R}_c$), e

[1] A literatura subdivide problemas de aprendizado em supervisionado, não supervisionado e por reforço [38, 11, 42], porém há outros paradigmas mais gerais, como a identificação de linguagem no limite [43]

seu respectivo rotulo y. A tarefa $\boldsymbol{T}$ consiste em atribuir um valor $\hat{y}$ para um $\boldsymbol{x}$ onde o y real ainda não é conhecido, mas assumindo que esse par ordenado $(\boldsymbol{x}, y)$ será gerado por um programa invariante e com regularidade de aprendizagem [41, 44].

Para isto, queremos induzir através dos pares de experiência $(\boldsymbol{x}, y)$ uma função computável f_c, isto é, uma máquina de Turing (MT) que sobre a entrada $\boldsymbol{w}$ para com exatamente $f_c(\boldsymbol{w})$ sobre sua fita, de tal forma que ao mensurarmos observações futuras de $\hat{y} = f_c(\boldsymbol{x})$ por $\boldsymbol{P}$, o valor de $\boldsymbol{P}(\hat{y}, y)$ tenda ao erro irredutível [41, 10, 11].

Codificaremos y em dois tipos de tarefas em $\boldsymbol{T}$, na primeira queremos computar uma cadeia de bits que representa um valor real computável a partir da entrada $\boldsymbol{x}$, ou seja, queremos uma f_c tal que $f_c : \mathbb{R}_c^n \longrightarrow \mathbb{R}_c$, essa tarefa denominamos regressão. Na segunda tarefa, queremos descobrir a função f que tenha como saída um único bit, de tal forma que $f : \mathbb{R}_c^n \longrightarrow \{0, 1\}$, essa tarefa denominamos classificação[2] [42, 41].

O processo indutivo descrito não é uma tarefa fácil, para um número finitos de exemplos existem infinitas funções (f_c) que interpolam perfeitamente todos esses pontos[46, 45], entretanto, para que se tenha um bom desempenho em tarefas futuras, tal função deve se aproximar do real programa gerador dos dados [41, 42, 47, 48], o que nos leva a pergunta: "qual dessas funções deve ser escolhida?"

Um princípio que pode ser adotado nessa escolha, é o chamado *"Navalha de Ockham"* [49], ele diz que para duas hipóteses de igual poder explicativo, é preferível escolher a mais simples. Para o caso da escolha entre polinômios, isto é, funções da forma

$$f(x) = a_0 x^n + a_2 x^{n-1} + ... + a_n,$$

essa é uma tarefa fácil, já que podemos compará-los em complexidade pelo polinômio de maior grau, porém estendendo essa tarefa para qualquer classe de funções, essa deixa de ser uma tarefa simples.

No início do desenvolvimento de técnicas de inferência indutiva, não havia uma definição formal do que seria uma hipótese "mais simples", em 1964 com o trabalho de Solomonoff da teoria de inferência indutiva universal [47], é apresentada uma definição formal para o que seria esse princípio: a MT de menor comprimento que computa tal observação, isso é, a f_c com menor complexidade de Kolmogorov. Esse sistema indutivo do ponto de vista teórico pode ser considerado perfeito [50], assumindo que o mundo é gerado por uma distribuição de probabilidade algorítmica, ele tem garantias de obter o menor erro teórico possível, também chamado de erro de Bayes.

Porém já se sabe que tal MT (programa) é indecidível, pois escolher um programa equivalente de menor comprimento é mapeável ao problema da parada [51, 52], logo, no

[2]Embora definido para duas classes o formalismo descrito aqui pode ser estendido para mais de duas classes com métodos multi classe [45].

caso geral, seria impossível descrever um método efetivo para proceder tal escolha [53]. Uma solução a esse problema[3] é definir *a priori* um espaço de hipóteses $\mathcal{H}$ (programas) de menor cardinalidade[4] que se aproxime do programa real, e.g. todos os programas que podem ser descritos em Python com no máximo 10^9 bits, assim, algoritmos efetivos poderiam escolher os programas de menor descrição em seu respectivo $\mathcal{H}$ [56].

Embora não exista uma máquina de Turing (MT) que resolva o caso geral, a teoria Provavelmente Aproximadamente Correto (PAC) de aprendizado nos dá garantias da existência de máquinas que encontrem uma função aproximada em tempo polinomial, isto é, para alguma amostra de tamanho n, existe uma MT que com tempo $O(n^k)$ para algum $k \in \mathbb{N}$, encontra uma MT c que mapeia $c : \mathcal{X} \to \mathcal{Y}$, de tal forma que essa MT se aproxime da real função geradora f_c com alta probabilidade. Para isso a classe de programas geradores que se pode assumir deve ser limitada, apenas dados gerados de forma independente por uma distribuição estacionaria nos garante a validade dos teoremas PAC.

Seja n um número que qualquer representação de $X \in \mathcal{X}$ tenha complexidade máximo $O(n)$ e representado por $size(c)$ o custo máximo da representação de uma f_c, ou seja, complexidade de Kolmogorov de $c \in \mathcal{C}$. Por exemplo, X pode ser uma sequência de bits representando R_c^m, de tal forma que essa sequência tenha $O(n)$. Adicionalmente, seja h_S a hipótese obtida pela MT $\mathcal{A}$ após receber uma amostra S.

Uma classe de conceitos $\mathcal{C}$ é dita PAC-aprendível se existe uma MT $\mathcal{A}$ e uma função polinomial $poly(.,.,.,.)$ de tal forma para que um dado $\epsilon > 0$ e um $\delta > 0$ para todas as distribuições $\mathcal{D}$ sobre $\mathcal{X}$ e para qualquer conceito alvo $c \in \mathcal{C}$, o seguinte é satisfeito para qualquer amostra de tamanho $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$:

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m}[R(h_S) \leq \epsilon] \geq 1 - \delta \tag{2.1}$$

Se $\mathcal{A}$ opera em $poly(1/\epsilon, 1/\delta, n, size(c))$, então $\mathcal{C}$ é dito ser eficientemente PAC-aprendível. Quando tal MT $\mathcal{A}$ existe, é chamado um algoritmo PAC-aprendível para $\mathcal{C}$.

Logo, dizemos que uma classe de conceitos $\mathcal{C}$ é PAC-aprendível, se a hipótese retornada pela MT aprendiz, após observar um número de pontos polinomial em relação a $1/\epsilon$ e $1/\delta$ é aproximadamente correto, isto é, tem um erro no máximo ϵ, com alta probabilidade, pelo menos $1 - \delta$. Ou seja, sendo $1 - \delta$ a confiança e $1 - \epsilon$ a acurácia, é definido que $\mathcal{A}$ obterá h com um custo polinomial em razão desses fatores.

Existe uma diversidade de algoritmos para seleção de hipóteses PAC-aprendível [41], neste trabalho apresentaremos a Árvore de Decisão [57], e algoritmos derivados como

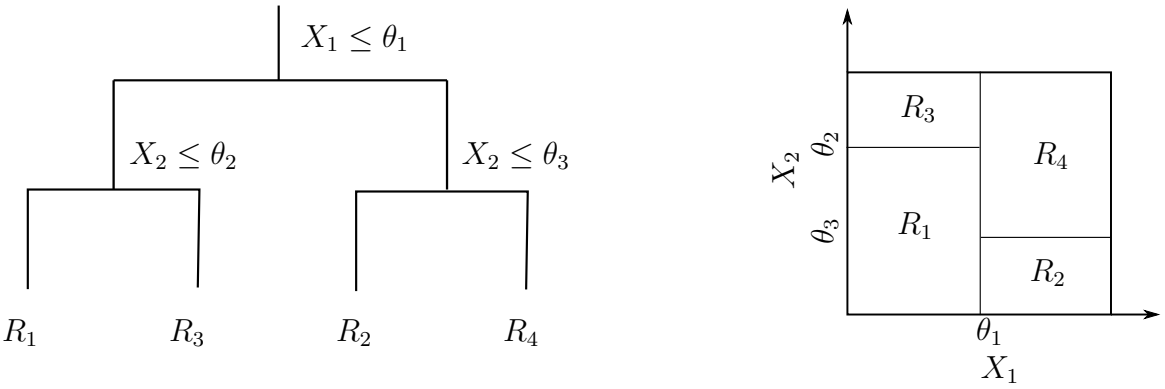
[3] As teorias Dimensão VC [12], Complexidade de Rademacher [54] e PAC [44] são casos restritos da inferência indutiva universal [50, 55]

[4] Também referido como Viés.

Floresta Aleatória [58, 59] e Máquinas de Reforço (*Boosting*) do Gradiente [60], também apresentaremos as Máquinas de Vetores de Suporte [12].

2.1.1 Árvores de Decisão

Árvores de Decisão são modelos que devido sua simplicidade tem fácil interpretabilidade e podem ser combinado com técnicas de amostragem e reforço em modelos muito mais poderosos. Embora exista um diversidade de algoritmos para sua concepção nos basearemos no modelo CART proposto em [57], a intuição por trás desse algoritmo é particionar o espaço de atributos $\mathcal{X}$ em regiões em R_i, de tal forma que cada R_i se aproxime do valor médio de $\mathcal{Y}$ para aquela região.



(a) Árvore de decisão com predicados booleanos sobre os atributos.

(b) Plano cartesiano particionado por uma árvore de decisão.

Figura 2.1: Exemplo de árvore de decisão

Na Figura 2.1b é exemplificado um particionamento dado pela árvore da Figura 2.1a, na raiz da árvore de decisão é aplicado um predicado booleano que irá dividir os dados em dois subconjuntos, a cada divisão nos nós esses subconjuntos são aplicados a novos predicados gerando subconjuntos menores.

Explicado o funcionamento das árvores, detalhamos agora como construí-las. Para isso, é feito uma busca exaustiva em todos os atributos j e seus valores t_m bi particionando os dados em $\theta = (j, t_m)$ como mostrado em 2.2.

$$\begin{aligned} Q_{\text{left}}(\theta) &= (x, y) | x_j <= t_m \\ Q_{\text{right}}(\theta) &= Q \setminus Q_{\text{left}}(\theta) \end{aligned} \tag{2.2}$$

De tal forma a minimizar L sobre a função de impureza H.

$$\arg\min_{\theta} \left[L(\theta) = \frac{n_{\text{left}}}{N_m} H(Q_{\text{left}}(\theta)) + \frac{n_{\text{right}}}{N_m} H(Q_{\text{right}}(\theta)) \right] \tag{2.3}$$

O cálculo do H é dado pela proporção de classes na região R_m, dado N_m observações, em razão do erro de classificação.

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{2.4}$$

Em [11] são apresentados alguns dos critérios que podem ser usados para computar a impureza da classificação, aqui aplicaremos o índice Gini.

$$H(X_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.5}$$

Esse particionamento é então aplicado recursivamente até que seja atingido algum critério de regularização, por exemplo, o limite mínimo para m ou definido um valor mínimo para H.

Embora árvores de decisão tenha diversas vantagens, elas tendem a sobreajustar aos dados, uma possível solução para isso é o uso de comitês.

2.1.2 Floresta Aleatória

Árvores de decisão podem "decorar" os dados de treino, não generalizando para casos desconhecidos, o chamado sobreajuste. O *Bagging*, uma técnica de reamostragem, tenta diminuir esse sobreajuste por meio da diminuição da variação entre os modelos, ou seja, modelos obtidos de forma independente tendam a uma mesma predição. Ele consiste em designar subamostras aleatórias (com repetição) do conjunto de dados e ao final as estimativas desses modelos são combinadas.

Porém ainda assim, dado a natureza determinística e a flexibilidade de uma árvore de decisão o problema do sobreajuste pode não ser resolvido, é necessário então aleatorizar o processo de geração da árvore. O método de subespaço aleatório (*random subspace method*), conceito chave para as Florestas Aleatórias (do inglês, Random Forest (RF)), provê isso limitando o espaço de busca por bipartições da árvore, a cada novo nó da árvore que está sendo construída, apenas um subconjunto aleatório dos atributos pode ser escolhido para a bipartição dos dados.

Essa limitação garante que as árvores de entropia máxima não possam ser sempre geradas, diminuindo a correlação entre as árvores e, portanto, melhorando a capacidade de generalização desse estimador. No Algoritmo 1 é apresentado o pseudo-código da Floresta Aleatória para a indução de um modelo.

Algorithm 1: Floresta Aleatória

Data: Conjunto de dados D com atributos p.

Input: B - Número de estimadores a serem construídos, m - Número de atributos nos subconjuntos.

Output: Classificador - Maioria dos votos de $\{T_b\}^B$.

1. Para $b = 1$ até B.
   (a) Obtenha um subconjunto aleatório com repetição (*Bootstrap*) de D.
   (b) Construa uma árvore T_b recursivamente da seguinte forma:
      i. Selecione m variáveis do conjunto p.
      ii. Selecione a melhor variável e bipartição em m.
      iii. Divida o nó em dois nós filhos.
2. Retorne o conjunto de árvores $\{T_b\}^B$.

2.1.3 Máquina de Reforço do Gradiente

O algoritmo de Floresta Aleatória nos dá uma ideia do poder que temos ao combinar algoritmos simples, também chamado *weak learner*, em algoritmos mais robustos. Uma outra alternativa a combinação de modelos é o Reforço, ou *Boosting*, esse consiste em criar estimadores a partir do resíduo de estimadores anteriores, assim esse novo estimador tem uma tarefa mais simples e logo, quando combinado com toda a sequência de estimadores, pode apresentar maior robustez no aprendizado. A esse algoritmo é dado o nome de *Gradient Boosting Machines* (GBM).

Demonstramos anteriormente como árvores CART são induzidas, sendo $R_j, j = 1, 2, ..J$ as regiões particionadas pela árvore, se $x \in R_j \implies f(x) = \gamma_j$, que no caso de classificação γ_j é o valor modal daquela região, para a regressão o valor médio da região. Logo podemos definir formalmente como

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j), \tag{2.6}$$

sendo $\Theta = \{R_j, \gamma_j\}_1^J$ parâmetros obtidos minimizando o risco empírico dado por

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} L(y_i, T(x; \Theta)). \tag{2.7}$$

Logo, as árvores de reforço são dadas por

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m), \tag{2.8}$$

induzidas sequencialmente resolvendo

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)), \tag{2.9}$$

logo, cada árvore deve aprender apenas o resíduo do erro da árvore anterior, isto é, a árvore m deve predizer $y_i - f_{m-1}(x_i)$. Porém o método descrito aqui embora seja intuitivo e correto, é ineficiente, sendo preferível sua aproximação que pode ser obtida em um único passo e não sequencialmente.

Definimos a função de custo por

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)), \tag{2.10}$$

portanto, queremos minimizar $L(f)$ em função de f, onde f é descrita na equação 2.8. Logo, sendo $\boldsymbol{f}$ as funções que aproximam $f(x_i)$ em cada ponto x_i de N, fazemos

$$\hat{\boldsymbol{f}} = \arg\min_{\boldsymbol{f}} L(\boldsymbol{f}). \tag{2.11}$$

Usamos então o método do gradiente para otimizar a equação 2.11, sendo ρ a taxa de aprendizado atualizamos

$$\boldsymbol{f}_m = \boldsymbol{f}_{m-1} - \rho_m \boldsymbol{g}_m, \tag{2.12}$$

onde os componentes do gradiente $\boldsymbol{g}_m$ são dados por

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}. \tag{2.13}$$

A implementação detalhada deste algoritmo está apresentada pelo pseudo-código do Algoritmo 2.

2.1.4 LightGBM

Em problemas de *Big Data* algoritmos de GBM se tornam muito ineficientes devido a sua construção ser realizada em função do número de instâncias e atributos [33], logo é necessário uma técnica de amostragem, isto é, operar com amostras menores, para redução da complexidade. O algoritmo LightGBM implementa a técnica de amostragem pelo método *Gradient-Based One-Side Sampling* (GOSS), pois, se observa que instâncias diferentes dos dados apresentam diferentes gradientes, esses que por sua vez atuam de diferentes formas para o ganho de informação. Aquelas de maior gradiente em valores absolutos, resultam em um maior ganho de informação.

Algorithm 2: Máquina de Reforço do Gradiente

Data: Conjunto de dados D com atributos p.
Input: M - Número de estimadores a serem construídos.
Output: Classificador - $f_M(x)$.

1. Construa $f_0(x) = arg\ min_\gamma \sum_{i=1}^{N}$
2. Para $m = 1$ até M.
   (a) Para $i = 1, 2, .., N$ calcule

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

   (b) obtenha a árvore de regressão para os r_{im} dada as regiões terminais $R_{jm}, j = 1, 2, .., J_m$.
   (c) Para $j = 1, 2, .., J_m$ calcule

$$\gamma_{jm} = \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

   (d) Atualize $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Retorne $\hat{f}(x) = f_M(x)$.

Esse algoritmo busca ao performar a amostragem obter apenas instâncias de maior gradiente, possibilitando uma redução significativa nos dados ao mesmo tempo em que mantêm parte significativa da informação. Uma outra vantagem dessa amostragem é que ela permite reduzir o ruído ao manter apenas dados com altos gradientes. Com isso, o algoritmo LightGBM performa até 20 vezes mais rápido que algoritmos de GBM tradicionais e consegue uma acurácia equivalente ou superior à esses [33, 34].

Uma segunda etapa que garante redução de recursos ao LightGBM é a utilização de histogramas. Diferente da configuração tradicional para a construção de árvores, em que é buscado um ponto ótimo de bipartição em dados contínuos, com complexidade O(atributos $\times$ instâncias), o LightGBM cria histogramas que discretizam os dados, tornando a operação de busca com complexidade O(atributos $\times$ baldes). Nos Algoritmos 3 e 4 são detalhados os procedimentos dessas etapas.

2.1.5 Máquina de Vetores de Suporte

Máquinas de Vetores de Suporte, do inglês *Support Vector Machines* (SVM), são construídas em cima de dois conceitos muito elegantes do aprendizado estatístico, classificadores

Algorithm 3: Algoritmo baseado em Histogramas

Data: I - dados de treino.
Input: d - profundidade máxima.
Output: H - histograma.

1. $nodeSet \leftarrow \{0\}$ - número de nós no nível atual.
2. $rowSet \leftarrow \{\{0, 1, 2, ...\}\}$ - índices dos dados nos nós das árvores.
3. Para $i = 1$ até d.
   (a) Para node em nodeSet.
      i. $usedRow \leftarrow rowSet[node]$
      ii. Para $k = 1$ até m.
         A. $H \leftarrow newHistogram()$
         B. Construa histograma
         C. Para j em usedRows
            - $bin \leftarrow I.f[k][j].bin$
            - $H[bin].y \leftarrow H[bin].y + I.y[j]$
            - $H[bin].n \leftarrow H[bin].n + I$
         D. Encontre a melhor separação no histograma H.
   (b) Atualize rowSet e nodeSet de acordo com os melhores pontos de separação.
4. Retorne H.

de máxima margem e projeções vetoriais. Classificadores de máxima margem obtém um hiperplano de solução única tal que a margem entre esse hiperplano e os pontos de duas classes, se linearmente separáveis, seja máximo, resultando em um hiperplano com garantias de boa generalização [12].

Porém, uma vasta gama de problemas não apresenta uma separação linear entre as classes, isto é, não podem ser separadas por um hiperplano, sendo então necessárias duas abordagens para generalizar esse algoritmo, projetar os dados em uma dimensão em que sejam linearmente separáveis, possivelmente uma dimensão infinita através do "truque de núcleo" (*kernel trick*), e também tornar as margens menos rígidas, para se adaptarem e tolerarem ruídos nos dados. Inicialmente definimos o hiperplano como

$$x^T\beta + \beta_0 = 0 \tag{2.14}$$

sendo β o vetor unitário, isto é $\|\beta\| = 1$ e a regra de classificação é dada por

$$G(x) = \text{sign}(x^T\beta + \beta_0). \tag{2.15}$$

Algorithm 4: *Gradient-Based One-Side Sampling*

Data: I - dados de treino.
Input: d - iterações, a - taxa de amostragem alta, b - taxa de amostragem baixa.
Input: $loss$ - função custo, L - *weak learner*.
Output: $models$ - Classificador.

1. $models \leftarrow \{\}$
2. $topN \leftarrow a \times len(I)$
3. $randN \leftarrow b \times len(I)$
4. Para $i = 1$ até d.
   (a) $preds \leftarrow models.predict(I)$
   (b) $g \leftarrow loss(I, preds)$
   (c) $w \leftarrow \{1, 1, 1, ...\}$
   (d) $sorted \leftarrow GetSortedIndices(abs(g))$
   (e) $topSet \leftarrow sorted[1 : topN]$
   (f) $randSet \leftarrow RandomPick(sorted[topN : len(I)], randN)$
   (g) $usedSet \leftarrow topSet + randSet$
   (h) $w[randSet] \times = fact$ Define os pesos como fact para dados com baixo gradiente
   (i) $newModel \leftarrow L(I[usedSet], -g[usedSet], w[usedSet])$
   (j) $models.append(newModel)$

Para obter o hiperplano de máxima margem otimizamos β restringindo o critério de otimização de tal forma que suavize para os pontos ξ_i, logo

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \quad \text{subject to } \xi_i \geq 0, y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall i, \tag{2.16}$$

onde C é um hiperparametro definido *a priori* que ajusta a rigidez das margens quanto a pontos erroneamente classificados. Essa operação tem por forma primal de Lagrange

$$L_p = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i. \tag{2.17}$$

Como anteriormente discutido, nem todos os problemas podem ser linearmente separáveis, mesmo suavizando as margens, o que limitaria e muito a aplicabilidade desse algoritmo, uma alternativa a isso é projetar os dados em uma dimensão em que as classes sejam separáveis pelo hiperplano, essa projeção é feita por uma função $h(x)$, for exemplo $h(x) = (x_1^2, x_2^2, x_1^2 + x_2^2)$ como mostrado na Figura 2.3.

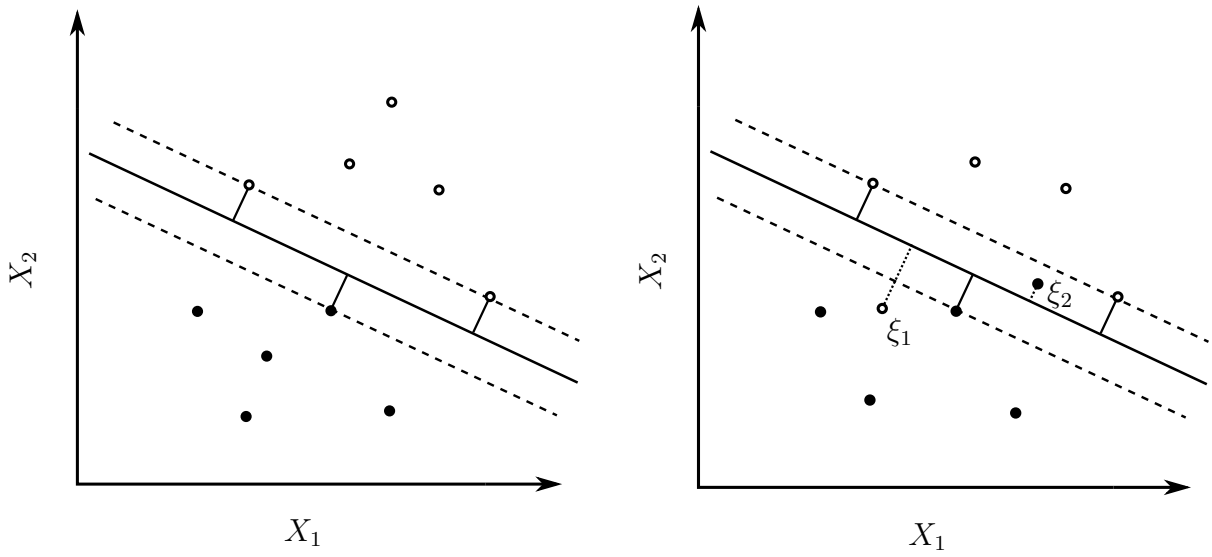


Figura 2.2: Máquina de vetores de suporte. A esquerda temos exemplos de dados que são linearmente separáveis logo é possível margens rígidas e a direita dados não-linearmente separáveis onde são necessárias margens permissivas. Os pontos ξ_i são aqueles que estão do lado errado do plano.

Desta forma, rescrevemos a função do hiperplano 2.14 como

$$h(x)^T\beta + \beta_0 = 0, \tag{2.18}$$

pela forma da equação 2.17 e 2.15 reescrevemos o classificador da seguinte forma

$$G(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0\right). \tag{2.19}$$

A literatura costuma apresentar três escolhas populares para núcleo, que são

$$\begin{aligned} \text{Polinomial:} &\quad (1 + \langle x, x_i\rangle) \\ \text{Base Radial:} &\quad \exp(-\gamma\|x - x_i\|^2) \\ \text{Rede Neural:} &\quad \tanh(k_1\langle x, x_i\rangle + k_2) \end{aligned} \tag{2.20}$$

Devido a elegância teórica desse algoritmo a sua generalização é dada em função da quantidade de vetores de suporte escolhidos, e embora essa projeção aumente a dimensionalidade dos dados, no caso do núcleo de base radial para uma dimensão infinita, ele ainda tem garantias de ter um baixo erro esperado fora dos dados conhecidos [12].

Vimos a viabilidade e descrevemos o projeto de alguns algoritmos, surge então a seguinte questão: "Para um dado problema qual desses algoritmos é o mais adequado?".

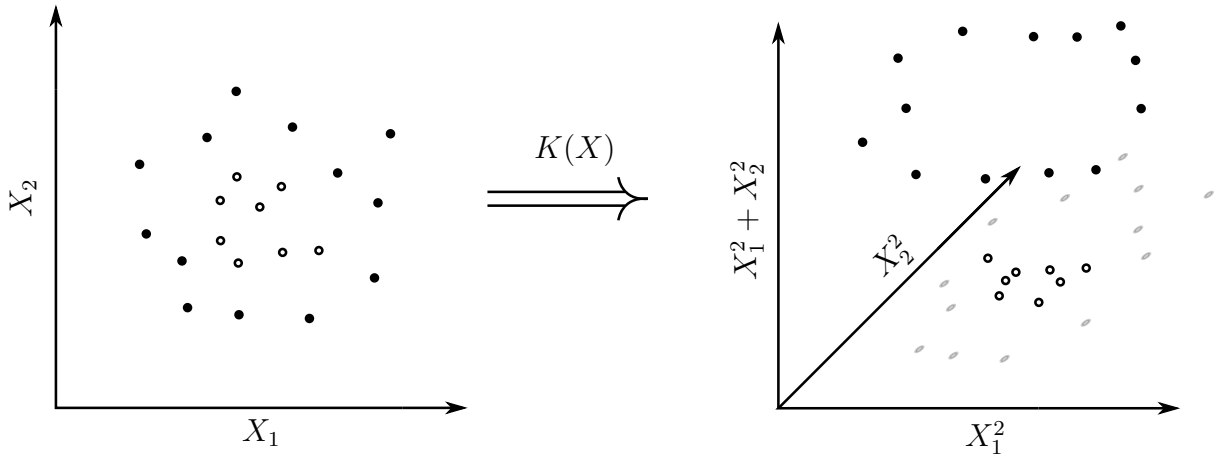


Figura 2.3: Operação de núcleo sobre os dados. A esquerda dados que não são linearmente separáveis se transformam em separáveis com a projeção demonstrada na figura a direita.

Como apresentado pelo teorema do *No Free Lunch* em [61, 62], não há uma distinção formal, *a priori*, entre esses algoritmos que aponte qual deve ser selecionado para um determinado problema, logo, métodos experimentais e estatísticos são aplicados nessa tarefa, como conjuntos de validação, para comparar e selecionar algoritmos. Porém esta abordagem apresenta algumas limitações dada a insuficiência estatística ou vieses causados pelo método empírico.

Neste trabalho buscamos automatizar e diminuir problemas causados pela seleção empírica de algoritmos, usando informações *a posteriori* do espaço de instâncias para essa escolha, essa abordagem também conhecida como meta-aprendizado.

2.2 Meta-Aprendizado

O problema de seleção de algoritmos foi inicialmente observado por J. R. Rice (1976) [63] com o principal objetivo de prever o melhor algoritmo para resolver um problema quando houver mais de um algoritmo disponível que encontre a solução. Os componentes desse modelo são: o espaço de instâncias do problema (P), que é composto por conjuntos de dados em Meta-Aprendizado (MtA); o espaço de instâncias de atributos (F), que são os meta-atributos usados para descrever os conjuntos de dados; o espaço de algoritmos (A), que contém um conjunto de algoritmos de AM que podem ser recomendados; e um espaço de medidas de avaliação (Y), responsável por recuperar os desempenhos dos algoritmos de AM que resolvem as instâncias dos problemas contidos em P. Usando os conjuntos descritos anteriormente, um sistema de MtA pode criar o mapeamento de um conjunto de dados p descrimináveis pelos meta-atributos f em um ou mais algoritmos α, de tal forma

que a recomendação de algoritmos por esse sistema tenha um desempenho medido por y tolerável, e.g., com máximo $y(\alpha(p))$.

K. A. Smith-Miles (2008) [64] aprimorou esse modelo abstrato propondo generalizações que podem também ser aplicadas ao problema do projeto de algoritmos. Nessa proposta, alguns componentes são adicionados: o conjunto de algoritmos de MtA; a generalização de regras empíricas ou ranqueamento de algoritmos; a verificação de resultados empíricos, que podem ser guiados por uma base teórica ao aprimoramento de algoritmos.

2.2.1 Espaço de Instâncias do Problema (P)

A definição do conjunto de instâncias de problema (P) é a etapa inicial do processo de concepção desse sistema, para o problema de MtA, o espaço de problemas consiste de conjuntos de dados com tarefas de mesma propriedade que o problema a ser resolvido, especialmente no caso de recomendação de algoritmos estamos lidando com tarefas de regressão ou classificação. Com uma vasta gama de conjuntos de dados, o ideal é utilizar de uma grande quantidade de conjuntos que sejam bem diversos entre si, para que populem a maior parte do espaço de problemas possível, com objetivo de induzir um meta-modelo confiável. Para reduzir o viés dessa escolha, conjuntos de dados de diversos repositórios, tais como o UCI[5] [65] e o OpenML[6] [66], podem ser usados.

Em problemas de fluxos de dados, a literatura costuma popular o espaço P com amostras, geralmente janelas deslizantes, obtidas do fluxo de dados momentos anteriores a inferência [24, 67], essa abordagem assume que os conceitos obtidos anteriormente são suficientes para descrever a complexidade dos novos conceitos que venham a surgir em momentos futuros [67].

2.2.2 Espaços de Instância de Atributos (F)

Como segunda etapa, é a definição do conjunto de meta-atributos (F) usados para descrever propriedades gerais dos conjuntos de dados por um vetor de características. Esses meta-atributos devem ser capazes de prover evidências sobre performance futura dos algoritmos em A [68, 69] e discriminar, com baixo custo computacional, a performance de um grupo de algoritmos. Formalmente em [29] essa caracterização é descrita da seguinte forma: Seja $D \in \mathcal{D}$ um conjunto de dados, $m : D \to \mathbb{R}^n$ uma medida de caracterização, $\sigma : \mathbb{R}^{n'} \to \mathbb{R}^n$ uma função de sumarização. Ambas m e σ tem hiperparâmetros associados, h_m e h_σ. Logo, um meta-atributo $f : D \to \mathbb{R}^n$ para um dado conjunto de dados D é

[5] `https://archive.ics.uci.edu/ml/index.php`
[6] `http://www.openml.org/`

$$f\Big(D\Big) = \sigma\Big(m(D, h_m), h_\sigma\Big). \tag{2.21}$$

Os principais meta-atributos usados na literatura de MtA podem ser divididos em cinco grupos: Gerais, Estatísticos, de Teoria da Informação, Baseados em Modelo e de Landmarkings. Os Gerais podem ser facilmente extraídos dos dados [70], com baixo custo computacional [69]. Os meta-atributos estatísticos capturam os indicadores principais sobre localização e distribuição dos dados, tais como medidas de média, desvio padrão, correlação e curtose. Meta-atributos de teoria da informação, usualmente medidas de entropia [71], capturam a quantidade de informação em um (sub)conjunto dos dados [64]. Já os baseados em modelo são propriedades extraídas de modelos de AM, geralmente Árvores de Decisão [72, 73], induzidos dos dados sob analise [70]. Os meta-atributos de Landmarking usam a performance de algoritmos de aprendizado simples e rápidos para caracterizar os conjuntos de dados [64].

Embora exista uma ampla variedade de meta-atributos para descrever conjuntos de dados, como apontado pela "maldição da dimensionalidade", conforme a adição de atributos cresce, o volume do espaço de instâncias cresce a uma razão exponencial disso [42], tornando os dados disponíveis esparsos e métodos de reconhecimento de padrões começam a perder precisão devido a baixa significância estatística das regiões [46]. Portanto, é necessário uma seleção cuidadosa dos meta-atributos a serem incorporados no modelo.

Em fluxos de dados, existem limitações especiais aos grupos de meta-atributos que podem ser utilizados. Como destacado anteriormente, o espaço P é obtido de janelas passadas do fluxo, logo, certos grupos de medidas apresentam baixa variabilidade entre as janelas (algumas medidas se mantém inalteradas), e portanto, baixo poder discriminativo, tornando esses grupos pouco interessantes para serem incorporados ao conjunto F [29]. Outro fator preponderante é dado pelo modo de operações de inferência, como os meta-atributos devem ser extraídos em tempo real para que sejam usados na predição da janela atual, grupos de alto custo computacional gerariam um atraso significativo no processo, e se caso o atraso for superior a carga de trabalho, ocorrerá o estouro do buffer da fila de processamento, inviabilizando o uso para sistemas em tempo real [9].

2.2.3 Espaço de Algoritmos (A)

O espaço de algoritmos A representa o conjunto de algoritmos candidatos a serem recomendados no processo de seleção de algoritmos. Idealmente, esses algoritmos devem também ser suficientemente distintos entre si, para que seja uma tarefa de menor complexidade, e que represente toda a região do espaço de algoritmos [31]. Os modelos induzidos pelo algoritmo podem ser avaliados por diferentes medidas, para tarefas de classificação, a

maioria dos estudos usando MtA usam acurácia. Entretanto, outros indicadores, como o F_β, AUC e coeficiente Kappa, também podem ser usados. Já no caso de regressão, MAE e RMSE são medidas comuns para essa avaliação.

Assim como a extração dos atributos em F pode causar o estouro da fila, o espaço de algoritmos tem como limitação o tempo de treino e inferência para sua escolha. Embora boa parte dos algoritmos possam operar em tempo irrisório quando acelerados por hardware multi-núcleo ou GPUs [74], essa não é uma realidade para muitos ambientes de fluxos de dados que tem uma configuração simples como sistemas embarcados [9]. Logo, algoritmos simples e eficientes são uma escolha mais viável nesse contexto [9].

2.2.4 Espaço de Medidas de Avaliação (Y)

Após a extração dos meta-atributos dos conjuntos de dados e a mensuração da performance do conjunto de algoritmos nesses conjuntos de dados, o próximo passo é rotular cada meta-exemplo nos meta-dados. Brazdil et al. (2009) [75] resume as quatro propriedades principais frequentemente usadas para rotular meta-exemplos em MtA: (i) o algoritmo que apresenta a melhor performance no conjunto de dados (uma tarefa de classificação); (ii) o ranqueamento dos algoritmos de acordo com suas performances no conjunto de dados (uma tarefa de ranqueamento), onde o algoritmo com melhor performance está no topo do ranque; (iii) o valor de performance obtido por cada algoritmo individualmente no conjunto de dados (uma tarefa de regressão) e; (iv) a descrição do modelo, que é geralmente baseado em agrupamentos ou regras de associação.

2.3 Fluxos de Dados

Dados estáticos e bem estruturados não se apresentam como uma boa alternativa a modelagem em ambientes com massiva interação livre de usuários, para esses ambientes uma das possíveis abordagens são Fluxos de Dados (FD) [76]. Embora FD sejam observados no tempo, similar a definição de uma série temporal eles diferem de séries temporais em alguns aspectos, fluxos de dados não tem uma frequência fixa entre as observações, sendo essa frequência definida geralmente pela ordem aleatória da geração dos dados que pode ou não ter uma natureza evolutiva [9]. Outro aspecto importante ao lidar com esse tipo de dado são as limitações de engenharia, um fluxo de dados é um conjunto infinito (não terminável) de dados e que não podem ser lidos livremente [76] e também, a taxa de processamento deve ser superior ao volume de chegada, caso contrário ocorrerá o estouro do buffer.

Essas restrições operacionais causam limitações significativa nos métodos e abordagens que podem ser aplicados na mineração de FD. Fluxos de dados podem ser gerados por dois

tipos de distribuições, estacionárias e não-estacionárias, para a primeira se tem medias e variâncias constantes para qualquer janela ao longo do tempo, já na segunda, esses dois parâmetros podem variar em função de fenômenos conhecidos ou desconhecidos por trás do processo gerador desses dados [77]. Tais padrões de variações costumam recorrer em diversas series e nomes foram atribuídos a cada comportamento, são eles, tendência, ciclos, sazonalidade e mudança de conceito [78, 79, 80].

Tendências e ciclos geralmente descrevem eventos econômicos de crescimento (ou decrescimento) e oscilações de subida e descida dado por flutuações de curto prazo, para simplicidade e por esses fenômenos terem uma natureza similar, eles são agregados em uma única componente e denominados por "ciclo-tendência" ou apenas "tendência" [78].

Com o passar dos anos é esperado um crescimento populacional e naturalmente a população empregada também irá crescer, a Figura 2.4 apresenta a série temporal do número de pessoas formalmente empregadas nos Estados Unidos, do ano 1947 a 1962, é observável o efeito da tendência no comportamento da série temporal já que podemos observar a tendência de crescimento como descrito acima.

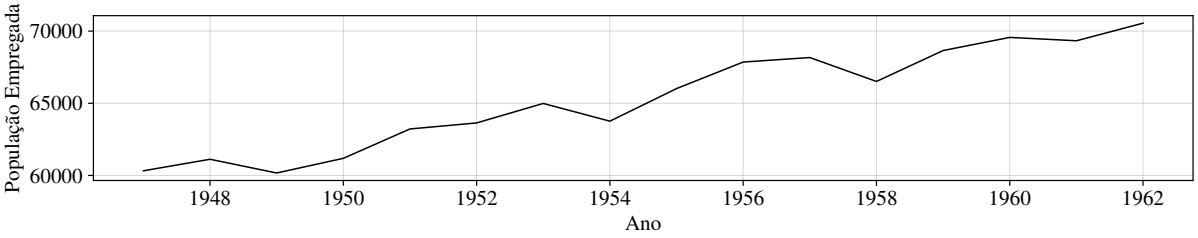


Figura 2.4: Exemplo de tendência.

Padrões sazonais ocorrem quando um fenômeno observado é afetado por fatores sazonais, por exemplo o período do ano ou a hora do dia, esse necessariamente de uma frequência fixa conhecida [78, 80]. Na Figura 2.5, é apresentado uma série de observações da temperatura média diária na cidade de Melbourne, Austrália ao longo de 10 anos, essa séria apresenta um padrão de sazonalidade, sendo a temperatura da terra afetada pelo ciclo solar e as estações do ano, é possível observar a periodicidade do aumento e diminuição da temperatura relacionado a isso.

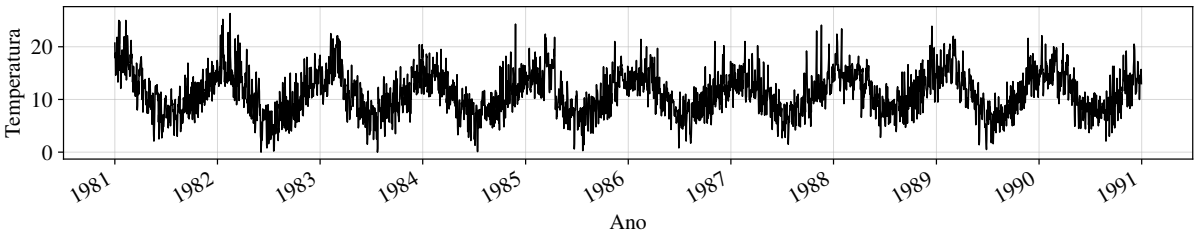


Figura 2.5: Exemplo de sazonalidade.

Já mudanças de conceito, diferente de tendência ou sazonalidade em que média e variância são alteradas por um movimento claro, na mudança de conceito há uma quebra abrupta entre essas medidas estatísticas ao longo do tempo. Na Figura 2.6, pode ser observado esse fenômeno, em que na altura do ponto 1000 a uma quebra da volatilidade observada anteriormente.

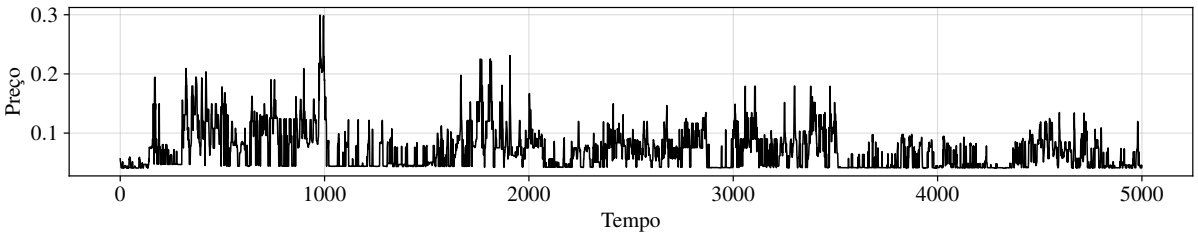


Figura 2.6: Exemplo de mudança de conceito.

A maioria das técnicas de AM assumem que os dados são independentes e identicamente distribuídos (*i.i.d.*), mas essas premissas geralmente não se sustentam em ambientes de FD, como apontado por A. Bifet and R. Gavaldà (2007) [16], tendo em vista que os dados chegam interminavelmente, é esperado uma evolução no processo gerador dos dados e, por isso, dependência temporal e mudanças de conceito podem ocorrer. Outro aspecto critico é a imposição de limites computacionais a esses ambientes, como memória, CPU, e largura de banda [81, 82].

2.4 Trabalhos Relacionados

Para resolver esses problemas, diversas abordagens foram desenvolvidas, dentre elas, as mais proeminentes são mecanismos de esquecimento, testes estatísticos e algoritmos adaptativos. Mecanismos de esquecimento fazem dados recentes tornarem-se mais relevantes e dados passados menos, um exemplo bastante adotado desse mecanismo são janelas deslizantes [83], janelas deslizantes é um método agnóstico de modelo que passa lotes de dados, a cada iteração, removendo amostras muito antigas e substituindo-as por amostras mais recentes. Já sistemas baseados em testes estatísticos agem monitorando uma métrica pré-definida, tal como performance preditiva do modelo (e.g. acurácia), e então "disparam" um alarme quando a qualidade dessas métricas está abaixo de algum limiar tolerável, demandando alguma ação para corrigir essa deterioração.

Exemplos dessa abordagem são o teste de Page-Hinkley e o *Statistical Process Control* [22]. Baseado nessas duas soluções, janelas deslizantes e testes estatísticos, o *ADaptive WINdowing* (ADWIN) [16] usa testes para definir o tamanho de cada janela, ajustando ao "tamanho do conceito". Entretanto, embora esses métodos sejam capazes de alarmar

uma redução na performance dos modelos e adaptar o tamanho da janela a isso, não é possível identificar os algoritmos ou modelos que se tornaram mais apropriados ao novo conjunto de dados que está sendo gerado.

Algoritmos de aprendizado adaptativo, como o VFDT [35], foram inicialmente introduzidos com FD em mente, desempenhando aprendizado online e baixo consumo de memória. Porém, não conseguiam se adaptar a mudança de conceito, um problema já conhecido na época. Mais tarde, aprimoramentos foram propostos, como o CVFDT (um aprimoramento direto sobre o VFDT) [84], o HAT [85] e o ARF [86] e outros [20, 87], que podem lidar com variação de conceito aplicando janelas deslizantes sobre o processo de treinamento. Entretanto, embora esses métodos esqueçam mais severamente dados passados quando ocorrer uma mudança de conceito, eles mantêm um espaço de hipótese fixo (Árvore de Decisão, para a maioria deles) para seleção em qualquer conceito que venha a surgir, logo, para conceitos distintos Árvores de Decisão podem não performar uma seleção ideal, gerando performances indesejáveis em algumas aplicações.

Uma outra abordagem presente na literatura, é o uso de um comitê de algoritmos (*Ensemble*), como os OzaBagging, SMOTE e CSB2 [88], em que o conjunto de algoritmos é continuamente treinado em janelas do fluxo de dados e suas previsão são combinadas em uma estimativa final.

Diferente das abordagens descritas, o MetaStream trata essas limitações da seguinte forma: (i) esse *framework* não precisa manter um sistema de alarme quanto a performance do algoritmo preditivo, pois dado um conjunto de algoritmos de desempenho razoável a nível base, é esperado que o método costume selecione o melhor ao longo do tempo, sem que nunca atinja, de forma prolongada, baixa performance; (ii) é flexível quanto ao viés indutivo, pois, diferente dos algoritmos anteriormente listados, que são limitados a Árvores de Decisão, a nível base qualquer algoritmo adequado a tarefa pode ser usado, como SVMs, Processos Gaussianos, modelos lineares, entre outros, já que pode almejar vieses algorítmicos como tarefa de aprendizado; (iii) comitês tem alto custo computacional para manter os modelos treinados continuamente e combinar suas predições, tornando inviável para certas aplicações [23].

Capítulo 3

Metodologia

Neste capítulo descrevemos o MetaStream, Figura 3.1, um *framework* para seleção periódica de algoritmos. Também apresentaremos os meta-atributos utilizados, justificativas, configurações do *framework*, incluindo os algoritmos a nível base analisador, o meta-classificador, LightGBM em configurações incremental e não-incremental e as respectivas avaliações nos experimentos.

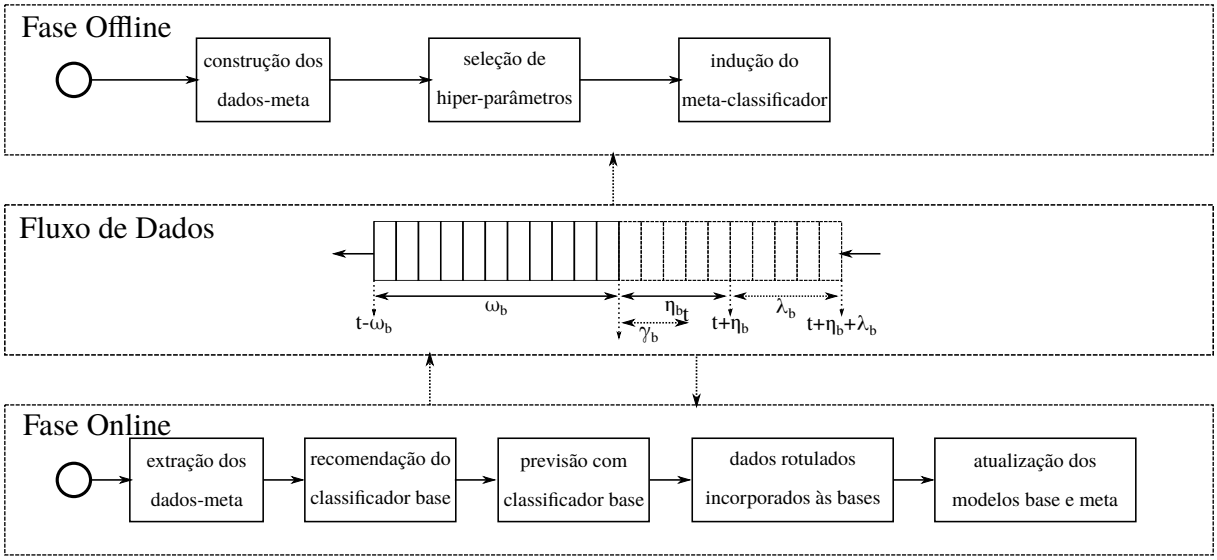


Figura 3.1: Diagrama de operação do MetaStream.

3.1 Metastream

Em ambientes de fluxos de dados, o espaço de instância de problemas é composto por apenas um problema p enquanto os meta-atributos F são extraídos dos dados de cada janela temporal para compor o meta-exemplo. É importante que o processo de extração dos meta-atributos tenha baixo custo computacional e alto grau de informação. Além

disso, a performance desses algoritmos A são obtidas de forma periódica para cada janela deslizante. Portanto, o algoritmo atual é substituído logo que o meta-modelo prediz que um algoritmo diferente é mais adequado para os exemplos da próxima janela deslizante. Essa é uma abordagem padrão na literatura de MtA em fluxos de dados [89, 67, 24].

A literatura apresenta alternativas a seleção de algoritmos (ou modelos) baseado em MtA para mudança de conceito. Um dos primeiros a apresentar é R. Klinkenberg (2005) [90], em que características obtidas do processo de aprendizado em si, no caso o tamanho das janelas deslizantes, foram usadas para induzir meta-modelos. Uma abordagem diferente foi investigada por J. Gama and P. Kosina (2014) [91] e R. Anderson et al. (2019) [24] reusando modelos previamente aprendidos mas usando o mesmo algoritmo para induzir esses modelos no fluxo de dados. Uma terceira alternativa é dada por J. N. van Rijn et al. (2018) [92], em que conjuntos de modelos são usados, tendo um alto custo computacional para induzir modelos para cada algoritmo e para manter os pesos atualizados.

Em seguida, é apresentado o MetaStream baseado no trabalho de A. L. D. Rossi et al. (2014) [23], com fases *offline* e *online*. A fase *offline* é projetada para escolha de hiper-parâmetros, validação e treinamento e geração dos dados em lote. A fase *online* age no ambiente dinâmico, recomendando um algoritmo para uma dada janela de dados. Ambas as fases são baseadas em sistemas de recomendação de MtA.

3.1.1 Fase *offline*

Essa fase se inicia aguardando o fluxo de dados gerar um lote de tamanho razoável para indução e seleção de modelos base, com esse lote em mãos, é realizada uma validação cruzada *k-fold* para estimar os hiper-parâmetros dos algoritmos base por meio de um Random Search [93], aqui assumimos que tais hiper-parâmetros serão os melhores para os eventos futuros. Após isso, em um configuração de janelas deslizantes, como mostrado na Figura 3.2, uma janela ω_{b1} é usada para induzir modelos com os algoritmos de nível base e também os meta-atributos x^m também são extraídos dessa janela.

Em seguida, os modelos induzidos são avaliados nos exemplos da janela η_{b1}, em que o algoritmo que produziu o modelo de melhor performance preditiva é definido como rótulo de y^m. Esse processo gera o primeiro meta-exemplo. O mesmo processo é aplicado continuamente por N passos, onde N é o número mínimo de instâncias para induzir um modelo consistente para gerar os meta-dados iniciais. Esses N meta-exemplos serão os primeiros dados que serão usados para induzir modelos na fase *online*.

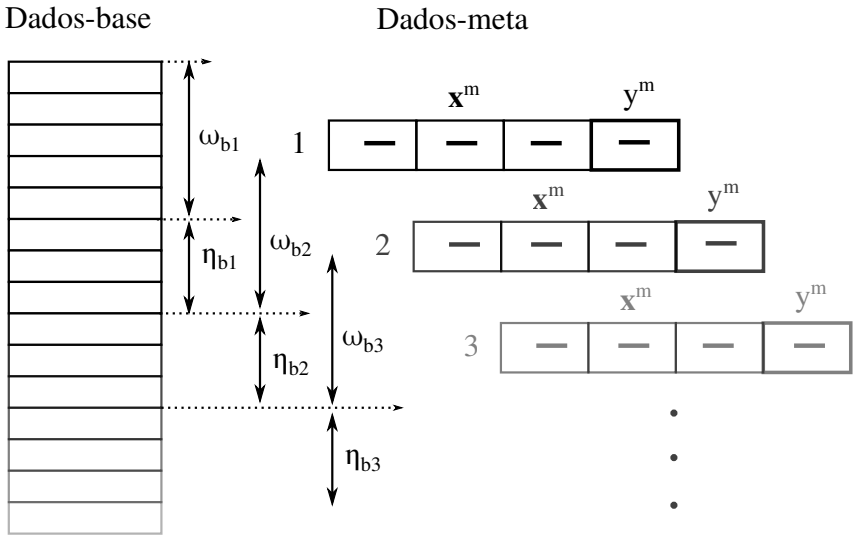


Figura 3.2: Extração de meta-atributos da janela ω_b e obtenção de rótulo da janela η_b.

3.1.2 Fase *online*

A fase *online* é iniciada quando o *framework* começa a receber um contínuo fluxo de dados. Inicialmente, ele recebe um vetor de atributos $\boldsymbol{x}_b = (x_1, ..., x_p)$, e após algum atraso, o rótulo $y_b \in \{0, 1, .., k\}$ dessa instância, onde k é o número de classes para uma tarefa de classificação ou $y_b \in \mathbb{R}$ para regressão.

A Figura 3.3 mostra um dado momento t na fase *online*. Ele tem uma janela de tamanho fixo ω_b que será usado para induzir um modelo, uma janela de tamanho fixo η_b em que o modelo induzido em ω_b será avaliado assim que o rótulo referente for descoberto, um tamanho γ_b que é o atraso em observar o rótulo para os exemplos na janela η_b e uma janela de tamanho variável λ_b de exemplos aguardando serem processados. Quando γ_b tem o mesmo tamanho que η_b, isto é, todos os rótulos de η_b foram obtidos, a janela ω_b é deslizada η_b instâncias para a direita e um novo modelo é induzido nessa janela.

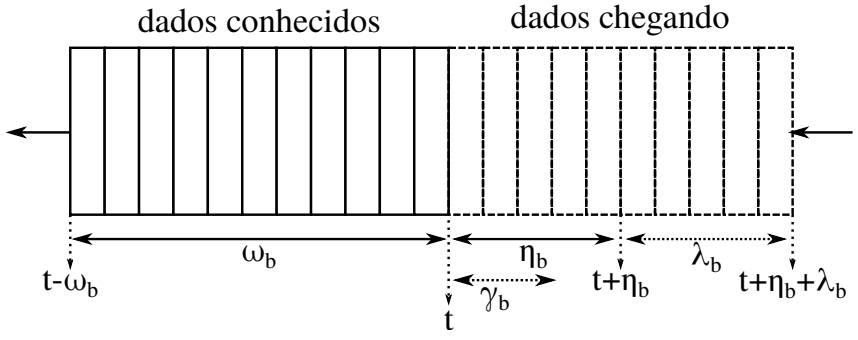


Figura 3.3: Discretização de janelas no nível base do fluxo de dados.

MtA entra em ação no segundo nível de processamento, nomeado nível meta. Como

mostrado na Figura 3.4, os meta-atributos são extraídos da janela ω_b e geram o meta-exemplo sem o rótulo que é mantido para atualização posterior.

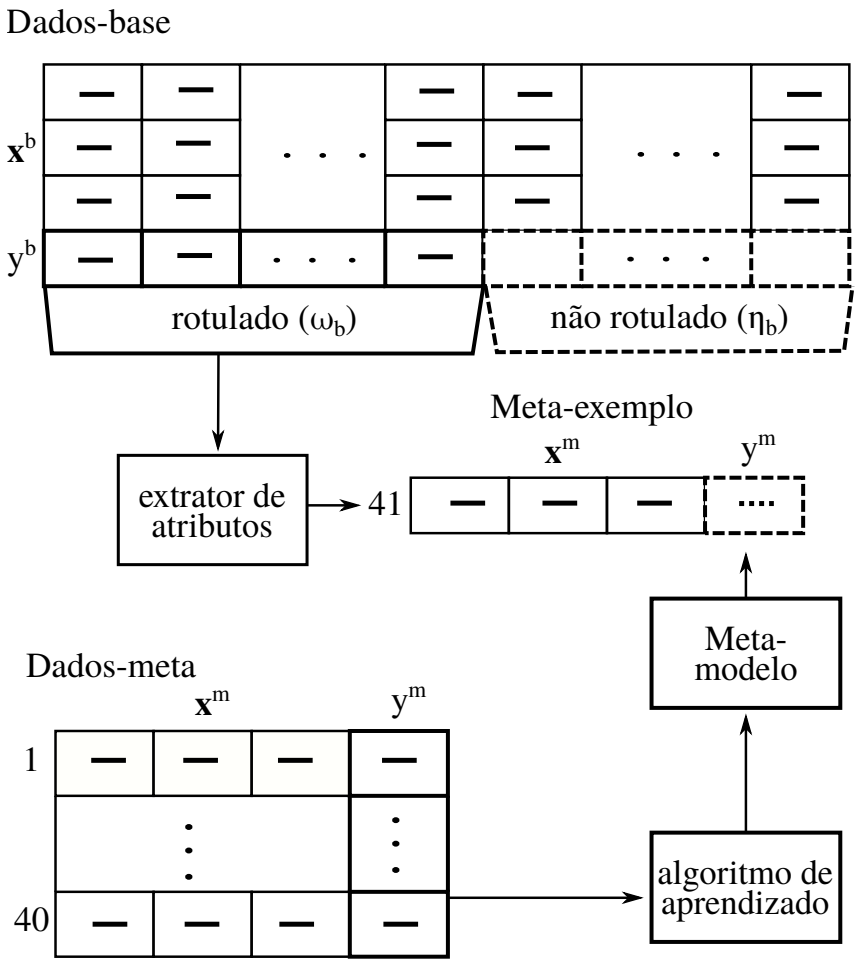


Figura 3.4: Extração de meta-atributos das janelas ω_b e η_b no nível meta.

O algoritmo de aprendizado (meta-aprendiz) usa meta-exemplos rotulados previamente na meta-base para induzir um meta-classificador, que é utilizado para recomendar um algoritmo que induzirá um modelo usando a janela ω_b que provavelmente será o mais preciso para prever os rótulos dos exemplos em η_b.

3.2 Parâmetros

Como descrito na Seção 2.2, nosso problema de seleção de algoritmos é dada pela escolha do espaço de hipóteses mais apropriado para a janela de dados mais recente. Portanto, sob a presença de mudança de conceito, é esperado que a metodologia aqui descrita consiga recomendar um classificador alternativo que se ajuste melhor aos dados em relação ao atual.

Muitas teorias foram desenvolvidas a respeito de espaços de hipóteses, exemplos são [12, 44], apontando direções para guiar essa escolha. Outro aspecto que deve ser considerado é o custo computacional, uso de memória e tempo de indução/inferência. Essas questões nos levaram a selecionar o Floresta Aleatória (RF) [59] e uma aproximação eficiente da Máquina de Vetores de Suporte com base radial (SVM-RBF) através do método Nyströem [94] como algoritmos de aprendizado no nível base.

Logo, nosso conjunto de dados em nível meta tem rótulos de acordo com a performance preditiva do modelos induzidos pelos RF e SVM para os exemplos na janela η_b. No mesmo instante, o meta-classificador busca qual desses dois classificadores é o mais apropriado para a próxima janela η_b.

Para o meta-classificador, isto é, o recomendador, selecionamos o LightGBM por três razões principais. Primeiro, o LightGBM apresenta o estado-da-arte em performance preditiva para diferentes problemas [33]. Segundo, o algoritmo lida automaticamente com atributos faltantes, o qual ocorrem comumente para os meta-atributos extraídos. Terceiro, ele apresenta uma capacidade de aprendizado incremental e baixo uso de recursos computacionais[1], encorajando essa escolha para aplicações *online*. Aprendizados incrementais e por lote apresentam uma distinção significativa. Enquanto o primeiro mantém as árvores e atualiza apenas os valores de seus nós, o segundo cria as árvores do zero. Não apenas o tempo de treinamento difere para essa variação, mas também o algoritmo incremental fixa *a priori* a importância dos atributos.

Para os meta-atributos, três grupos da Seção 2.2 foram analisados, denominados estatístico, baseado em modelo e *landmarking*. Os meta-atributos estatísticos capturam indicadores da localização e distribuição, as baseado em modelo extraem propriedades de árvores de decisão induzidas sobre os dados, enquanto as de *landmarking* usam a performance de classificadores simples e rápidos para caracterizar os conjuntos de dados [29]. No geral, eles tem alto poder discriminativo e um custo computacional médio. Para esses experimentos, usamos meta-atributos disponíveis no pacote *Python Meta-Feature Extractor* (pymfe) [95] disponível publicamente no GitHub[2].

Nós fixamos o tamanho das janelas para todos os experimentos, em ambos os níveis, base e meta, $\omega_{m;b} = 300$ e $\eta_{m;b} = 10$. O passo da janela de treinamento é o mesmo que η. Nós também performamos o ajuste de hiper-parâmetros nos algoritmos de nível base através de uma validação cruzada k-fold. Nós mantemos esses parâmetros fixos para toda indução futura que possa ocorrer. Embora essa decisão possa não levar a resultados ótimos, nós acreditamos que esses valores de hiper-parâmetros sejam melhor escolha do que não realizar nenhuma seleção, isto é, usar os valores padrões, tendo em vista que

[1] `https://lightgbm.readthedocs.io/en/latest/Experiments.html`
[2] `https://github.com/ealcobaca/pymfe`

realizar esse ajuste a cada janela teria alto custo computacional, tornando inviável para o desempenho *online*. Nós usamos acurácia, métrica Kappa e média geométrica para mensurar a performance preditiva dos modelos induzidos, tanto na fase *online* quanto na fase *offline*. Kappa e média geométrica são as medidas mais apropriadas que acurácia para avaliar uma distribuição de classes desbalanceada, que é frequente nos meta-dados.

3.2.1 Bases de Dados

Para os experimentos, nós selecionamos três conjuntos de dados populares para classificação na literatura de fluxos de dados que apresentam mudança de conceito [67, 89], *Electricity* [96], *Power Supply* [97] e *Covertype* [98], e mais três conjuntos de dados sintéticos gerados com o auxilio da biblioteca scikit-multiflow [99], *HyperPlane* [84], *Agrawal* [100] e *RandomRBF* [99].

- *Electricity:* O conjunto de dados foi coletado do mercado de eletricidade de Nova Gales do Sul, na Austrália. Os preços flutuam de acordo com a oferta e demanda, apresentando fatores sazonais. Ele contem 45.312 instâncias datadas de 7 de Maio de 1996 a 5 de Dezembro de 1998 (obtidas a cada 5 minutos). O atributo alvo é se a demanda foi maior ou menor comparada a média das 24 horas anteriores, e como atributos preditivos, o dia da semana, etiqueta de tempo, a demanda por eletricidade em Nova Gales do Sul, a demanda por eletricidade em Vitória e o agendamento de transferência de eletricidade entre esses estados.

- *Covertype:* Esse conjunto de dados contém variáveis cartográficas de células 30x30-metros obtidas do Serviço Florestal dos Estados Unidos, ela originalmente tinha 581.021 instâncias, mas nós reduzimos para apenas as 50.000 primeiras para reduzir o tempo de processamento. O atributo alvo é o tipo de cobertura florestal daquela célula (classes variando de 1 a 7), e como atributo preditivo, ele apresenta 53 propriedades da célula como elevação, inclinação, tipo do solo, etc.

- *PowerSupply:* O *Power Supply* contém os registros de 1995 a 1998, a tarefa é prever em que hora (classe) a fonte de energia em questão pertence (de 1 a 24), que está relacionado a alguns fatores sazonais e tendências, como atributo preditivo nós temos a produção de energia nas malhas de energia principal e auxiliares. Ele contém 29.928 instâncias.

- *HyperPlane:* Um hiperplano, em um espaço de dimensão d, é o conjunto de pontos x que satisfaz $\sum_{i=1}^{d} w_i x_i = w_0 = \sum_{i=1}^{d} w_i$, onde x_i é a i-ésima coordenada de x. Exemplos em que $\sum_{i=1}^{d} w_i x_i \leq w_0$ são classificados como negativos e caso contrário,

positivos, gerando então um problema de classificação binária. A mudança de conceito é simulada pela variação gradual e suave na orientação e posição do hiperplano pela mudança relativa dos pesos. Os pesos são atualizados para $w_i = w_i + d\sigma$, sendo σ a probabilidade da mudança ser revertida e d a mudança aplicada a cada novo exemplo. Adicionalmente é aplicada uma mudança abrupta entre duas configurações desse fluxo, ele é dado pela função $f(t) = 1/(1 + e^{-4(t-p)/w})$, no tempo t o fluxo terá probabilidade f de ocorrer a mudança abrupta, sendo p a posição da mudança e w o comprimento da transição. Ambos os fluxos tem 5 atributos preditivos com $p = 700$ e $w = 1000$, o primeiro fluxo é dado por $d = 0.1$ e $\sigma = 0.1$ o segundo é dado por $d = 0.1$ e $\sigma = 0.7$, adicionalmente é adicionado um ruído de 0.25 a três dos atributos preditivos para gerar 40.000 instâncias.

- *Agrawal:* Esse fluxo de dados contém 9 atributos preditivos, sendo 6 numéricos e 3 categóricos. Como tarefa uma das funções descrita em [100] define se um cliente deve ou não ter seu crédito aprovado. Como no fluxos anterior esse também apresenta uma mudança abrupta, com $p = 1000$ e $w = 1000$. O primeiro fluxo usa a primeira função de crédito e um ruído ocorre nos atributos preditivos com probabilidade 0.2, já o segundo, usa a terceira função e tem ruído com probabilidade 0.7, e classes desbalanceadas. Ao todo foram geradas 40.000 instâncias.

- *RandomRBF:* O Função de Base Radial Aleatória (Random Radial Basis Function) gera um conjunto de dados primeiro criando um centroide para cada classe, cada centro gerado aleatoriamente é atribuído um peso, um ponto central por atributo e um desvio padrão. Para gerar novas instâncias, um centro é escolhido aleatoriamente a partir de seus pesos. Os atributos são aleatoriamente gerados deslocados do centroide, onde o vetor descritivo foi dimensionado de modo que seu comprimento seja igual a um valor obtido aleatoriamente da distribuição gaussiana do centroide. Logo, o centroide, escolhido em questão, determina a classe dessa instância. Adicionalmente é aplicada uma mudança de conceito por meio do deslocamento dos centroides, onde a cada nova instância gerada, os centroides iniciais são deslocados por uma magnitude m, desta forma, o conjunto de dados foi gerado com a seguinte configuração. Número de classes 2, número de centroides 10, número de dimensões 5, magnitude do deslocamento $m = 0.1$. Como nos outros dataset, 40.000 instâncias foram geradas.

A analise dos experimentos tem uma fase *offline* e uma *online*. A *offline* analisa o *framework* no nível meta, isto é, o quão bem ele pode predizer o melhor algoritmo dado os meta-atributos descritivos. A *online* analisa, no nível base, os ganhos obtidos por selecionar o algoritmo recomendado em nível base, comparado ao método de referência,

nomeado Padrão, que representa o algoritmo base que teve majoritariamente uma melhor performance. Para isso, nós olhamos a soma acumulada dos ganhos de performance ao longo do tempo, que é dado pela diferença de performance entre o algoritmo recomendando e usando o Padrão.

Capítulo 4

Resultados

Similar ao Seção 3.1, os resultados são divididos em fases *offline* e *online*. Na avaliação *offline*, nós analisamos o meta-classificador a respeito do quão bem ele consegue discriminar as classes no nível meta. Na avaliação *online*, nós medimos os ganhos a respeito do desempenho preditivo no nível base quando aplicado a recomendação de algoritmos pelo meta-classificador. Os experimentos foram executados em um cluster com dois processadores Intel Xeon Silver 4114 de 2,20GHz com 256 Gb de RAM. O sistema operacional desta máquina é o Debian 9.

4.1 Analise *offline*

A Tabela 4.1 lista a distribuição de algoritmos (RF e SVM[1]) que apresentam o melhor desempenho para cada lote do fluxo de dados (*Electricity*, *CoverType*, *PowerSupply*, *HyperPlane*, *Agrawal* e *RandomRBF*). Na perspectiva de MtA, essa tabela apresenta a distribuição de classes dos meta-exemplos para os meta-dados *offline*.

Tabela 4.1: Distribuição de algoritmos nos meta-dados por problema de fluxo de dados.

Conjunto de Dados	SVM	RF
Electricity	75,2%	24,8%
CoverType	71,8%	28,2%
PowerSupply	53,5%	46,5%
HyperPlane	79,2%	20,8%
Agrawal	78,5%	21,5%
RandomRBF	53,5%	46,5%

Em termos gerais o algoritmo SVM apresentou melhor desempenho em relação ao RF para todos os conjuntos de fluxos de dados. Para os conjuntos *PowerSupply* e *RandomRBF*

[1]Uma aproximação do SVM como descrito na seção metodologia.

o desempenho foi quase balanceado entre eles. Já nos outros conjuntos a distribuição está bem desbalanceada para o SVM, logo, treinamos o meta-classificador para que lidasse com esse desbalanceamento, ajustando a função custo em razão da proporção das classes.

Nós então estimamos o desempenho preditivo do meta-classificador usando um validação cruzada para series temporais, como descrito em [78], para os dados iniciais. O modelo é inicialmente induzido na primeira janela de $\omega_m = 300$ exemplos de treino e prevê as classes dos próximos $\eta_m = 10$ exemplos de teste, então ambas as janelas deslizam 1 exemplo para a próxima avaliação. A Tabela 4.2 apresenta a média e o desvio padrão para essas métricas. Ganhos da meta-recomendação advém da recomendação não trivial dos algoritmos. Dado que o método Padrão representa um Kappa igual a zero, $Kappa = 0$, $Kappa > 0$ representa um ganho preditivo.

Tabela 4.2: Desempenho preditivo do meta-classificador na fase *offline*.

Fluxo	Kappa	M. Geométrica	Acurácia
Electricity	0.095±0.176	0.366±0.207	0.610±0.152
CoverType	0.018±0.215	0.329±0.232	0.517±0.147
PowerSupply	0.251±0.282	0.520±0.252	0.611±0.188
HyperPlane	0.118±0.326	0.120±0.325	0.841±0.080
Agrawal	0.007±0.138	0.046±0.164	0.780±0.078
RandomRBF	0.120±0.239	0.416±0.262	0.571±0.115

A Tabela 4.2 apresenta o desempenho preditivo da fase *offline* avaliadas por três medidas distintas na configuração de janela deslizante. A acurácia é estimada sobre $\eta_m = 10$, que é comparável a distribuição dos meta-dados da Tabela 4.1. Para as medidas Kappa e Média Geométrica, nós removemos os exemplos em η_m onde não tiveram diferença significativa entre os desempenhos dos algoritmos no nível base, isto é, nós computamos essas medidas para os exemplos base onde $|\text{Acc(RF)} - \text{Acc(SVM)}| > 0.1$. Essa abordagem foi aplicada com o objetivo de produzir uma mensuração mais apropriada do meta-classificador, pois onde não há uma diferença clara entre o desempenho dos modelos, qualquer um deles pode ser escolhido sem perdas.

Para todos os conjuntos, a medida Kappa é pelo menos levemente positiva, mostrando que o recomendador consegue selecionar o algoritmo correto quando há uma diferença significativa no desempenho preditivo entre eles, superando o método padrão. Acurácia é próxima da distribuição da classe majoritária, mas posteriormente iremos argumentar que grande parte dos exemplos não há diferença clara e alguns exemplos apresentam um ganho melhor de predição que outros, o que no nível base reflete em uma diferença significativa.

Para avaliar o quanto os meta-atributos contribuem ao sistema de recomendação, a Figura 4.1 mostra os 5 melhores meta-atributos, para cada conjunto, selecionado pelo algoritmo LightGBM. Nessa figura, o eixo x representa a importância dos atributos dos

meta-modelos dados pelo LightGBM, e o eixo y mostra esses meta-atributos. A grande maioria dos meta-atributos selecionados por nós para dividir os dados foram do grupo de landmarking, seguido por baseado em modelo e estatístico. Esses grupos tem o maior grau informativo e discriminativo de acordo com a literatura [29].

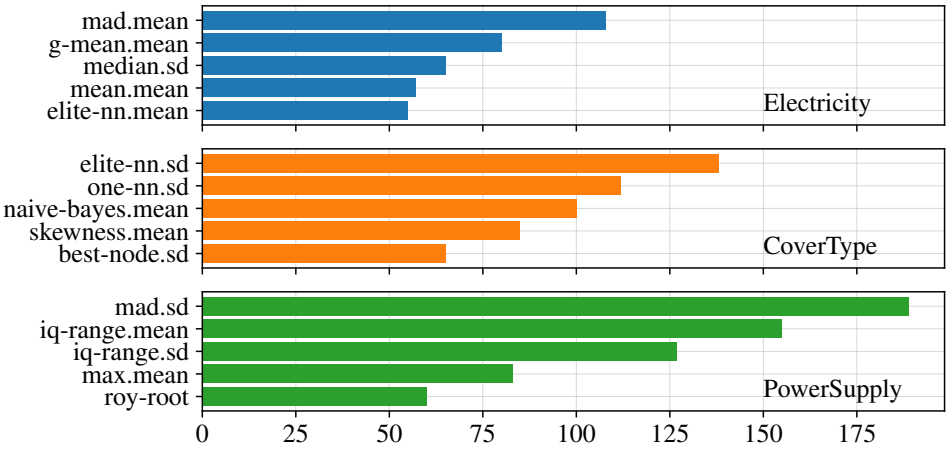


Figura 4.1: Importância dos atributos para o meta-algoritmo nos dados *offline*.

4.2 Analise *online*

Na fase *online*, nós avaliamos o meta-classificador (LightGBM) usando uma estratégia incremental e não incremental. Na avaliação passada, nós treinamos o meta-classificador usando o procedimento descrito na Seção 3.1.2. Quando novos meta-atributos estão disponíveis, o meta-classificador e retreinado do zero, e isto pode levar a um modelo completamente independente do modelo passado. Na estratégia *online*, o LightGBM atualiza o modelo anterior, adaptando as árvores de decisão para se ajustar aos meta-exemplos da janela mais recente, atualizando assim os pesos de cada nó, mas sem substitui-los. A Tabela 4.3 apresenta esses resultados.

Embora a Tabela 4.3 apresente valores para Kappa relativamente baixos, nós notamos que esse valor é positivo para quase todas as combinações de conjunto-estratégia. Esses resultados indicam que o meta-classificador consegue selecionar o algoritmo mais apropriado comparado com a classe majoritária. Esse resultado é mais evidente para o conjunto *PowerSupply*, que apresenta a melhor média geométrica e Kappa. Essa consistência refletirá posteriormente nos ganhos acumulados para esse conjunto. A estratégia incremental apresenta melhores desempenhos em relação a não incremental para os conjuntos *Electricity*, *HyperPlane* e *RandomRBF*, para os outros conjuntos o incremental desempenhou pior para essas métricas.

Tabela 4.3: Desempenho preditivo do meta-classificador na fase *online*.

Dataset	Estratégia	Kappa	M. Geométrica	Acurácia
Electricity	Non-Incremental	0.027	0.462	0.558
	Incremental	0.039	0.422	0.457
CoverType	Non-Incremental	0.114	0.461	0.694
	Incremental	-0.009	0.405	0.631
PowerSupply	Non-Incremental	0.092	0.502	0.607
	Incremental	0.074	0.541	0.539
HyperPlane	Non-Incremental	0.014	0.265	0.747
	Incremental	0.020	0.283	0.745
Agrawal	Non-Incremental	0.010	0.198	0.794
	Incremental	-0.041	0.327	0.700
RandomRBF	Non-Incremental	-0.031	0.484	0.484
	Incremental	0.004	0.486	0.501

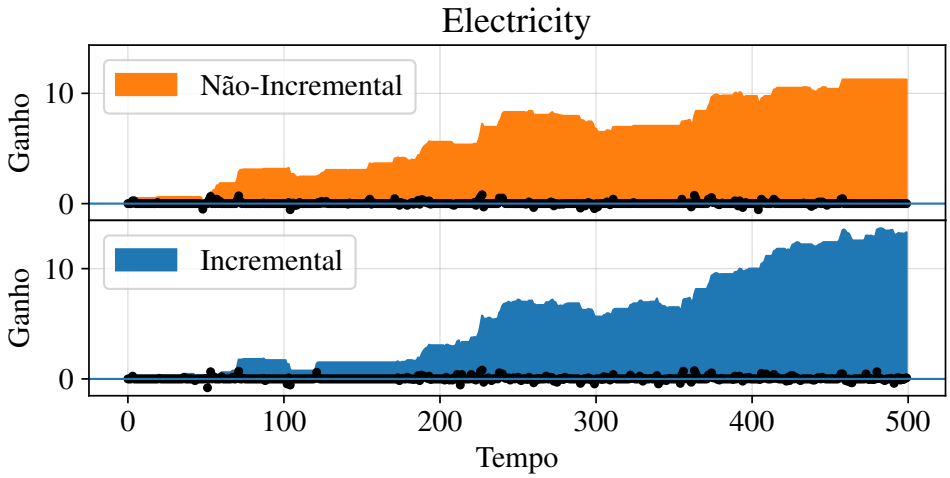


Figura 4.2: Ganho cumulativo de acurácia ao longo do tempo para o conjunto Electricity.

Figuras 4.2-4.4 mostram o ganho cumulativo, que é a diferença entre a acurácia do algoritmo recomendado e a acurácia do método Padrão. A região preenchida é a soma aculumada dessas diferenças de desempenho ao longo do tempo enquanto as cores laranja e azul representam as estratégias incrementais e não incrementais, respectivamente. Cada ponto preto representa o ganho de escore no tempo t.

Nessas figuras, nós observamos um ganho positivo para o *Electricity* e o *PowerSupply*. Já para o conjunto *CoverType*, para essas configurações, o MetaStream falhou em selecionar efetivamente o melhor algoritmo, representado pela perda desempenho, escore negativo. Em termos gerais há um empate entre as estratégias para os conjuntos de dados reais, nas próximas figuras são apresentados os conjuntos sintéticos.

O conjunto de dados sintético *HyperPlane* apresentou um ganho inconsistente para o método não incremental e um ganho ligeiramente positivo para o método incremental.

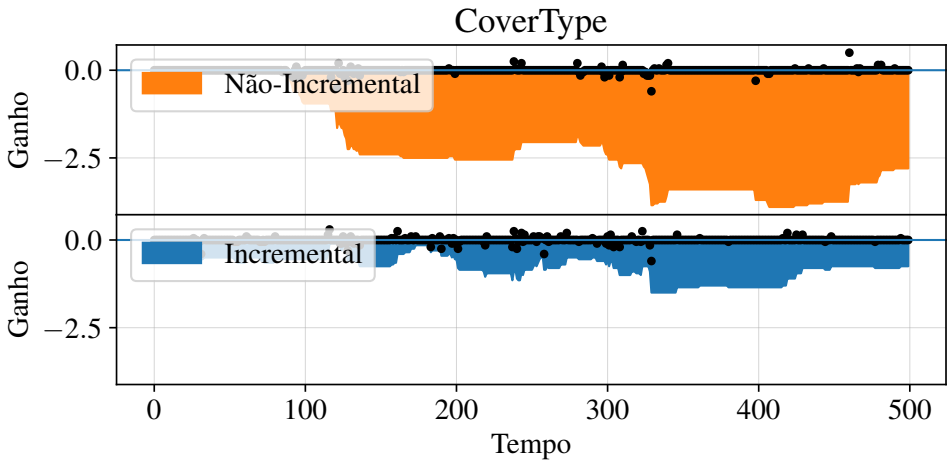


Figura 4.3: Ganho cumulativo de acurácia ao longo do tempo para o conjunto CoverType.

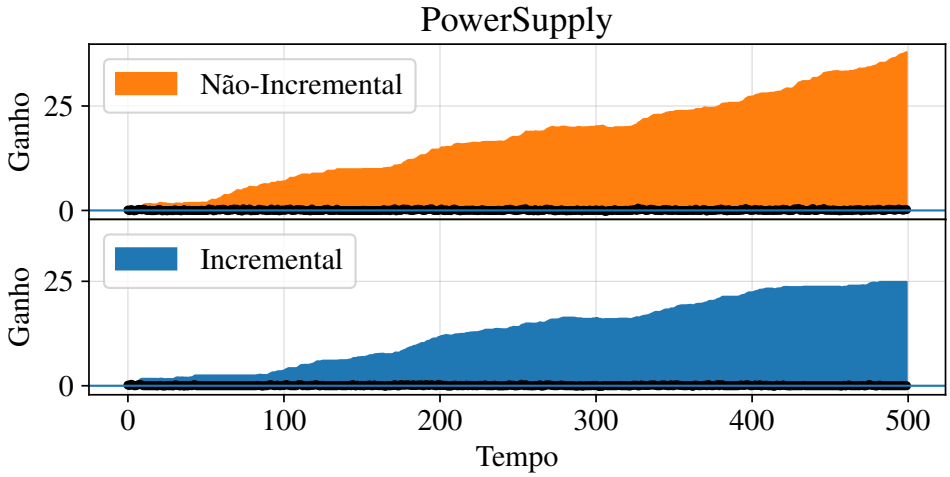


Figura 4.4: Ganho cumulativo de acurácia ao longo do tempo para o conjunto PowerSupply.

Com o Agrawal *HyperPlane* tivemos um resultado bem diferente, o *framework* conseguiu obter ganhos consideráveis para ambos os métodos, tendo em média ganho de mais de 0.1 de acurácia por recomendação.

Por fim, o conjunto *RandomRBF* apresentou ganhos positivos, mas mais significativamente para o método não incremental. Para os conjuntos sintéticos, podemos ver um ganho para todas as recomendações pelo algoritmo incremental, sendo no caso do *HyperPlane* ligeiramente negativo para o algoritmo não incremental.

Nas Figuras 4.8, 4.9 e 4.10 as acurácias para o Padrão são dispostas contra as acurácias do Recomendado em um histograma bi-dimensional por mapa de calor. Logo, cada quadrado no mapa de calor representa a acurácia do algoritmo recomendado pelo método Padrão (eixo x) e pelo meta-classificador (eixo y) para a mesma janela. A intensidade da

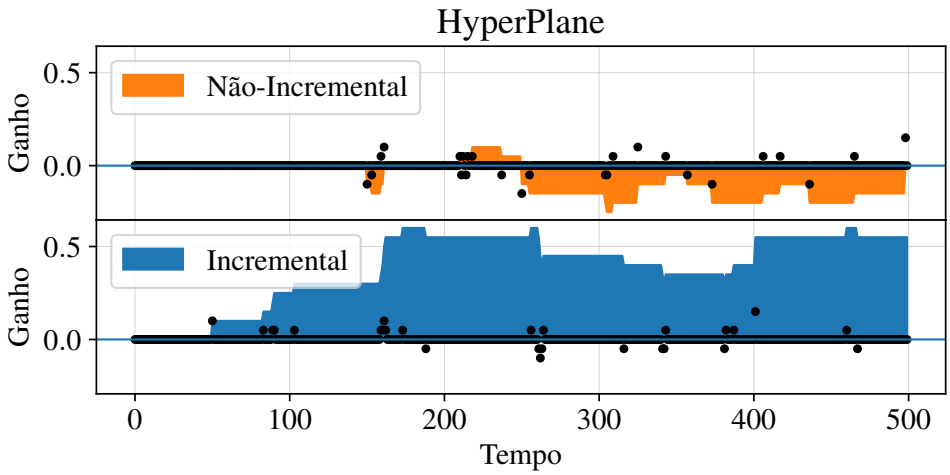


Figura 4.5: Ganho cumulativo de acurácia ao longo do tempo para o conjunto HyperPlane.

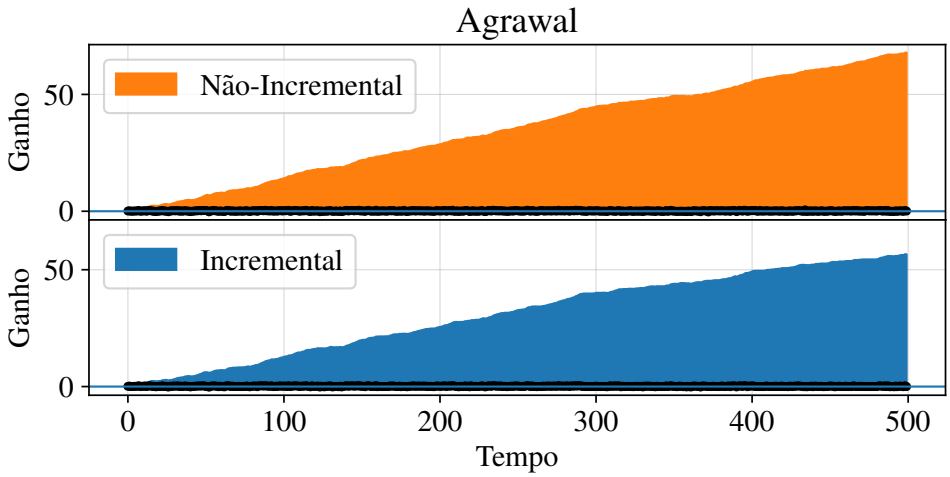


Figura 4.6: Ganho cumulativo de acurácia ao longo do tempo para o conjunto Agrawal.

cor, como mostrado na barra de cores, representa o numero de pontos naquela posição. Os pontos da diagonal, onde o algoritmo recomendado e o padrão tiveram a mesma acurácia, foram removidos e coloridos de branco, para enfatizar apenas os pontos de ganhos e perdas do *framework*. Se o classificador recomendado sempre for pior que o Padrão, isto é, recomendar a classe minoritária quando esta não é melhor que a majoritária, então todos os pontos ficaram abaixo da diagonal. Similarmente, esse pontos estarão acima se efetivamente selecionar o algoritmo da classe minoritária quando esse for melhor que a majoritária.

Na Figura 4.8, é visível na região superior direita do mapa de calor que o algoritmo incremental está recomendando a classe minoritária mais vezes que não-incremental, também o quadrado azul acima da diagonal branca tem mais pontos (100+ comparado a 30+ no não-incremental), isso reflete os ganhos mostrados na Figura 4.2.

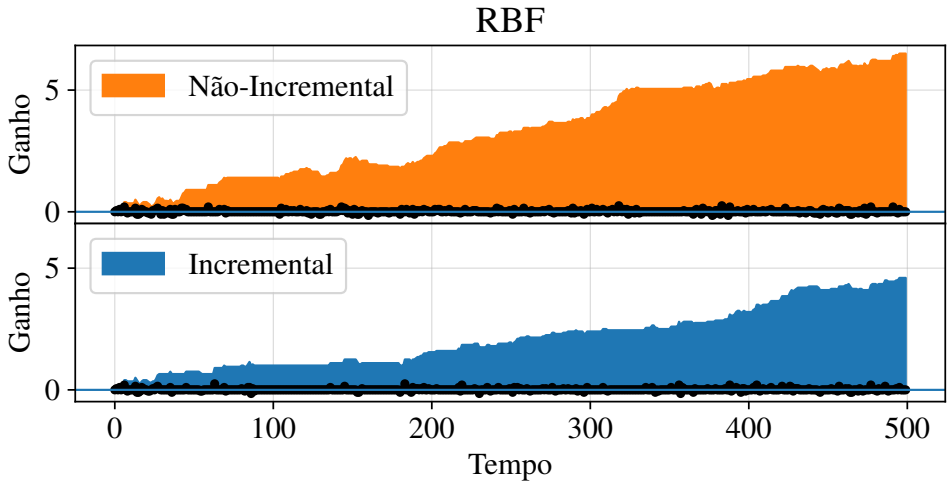


Figura 4.7: Ganho cumulativo de acurácia ao longo do tempo para o conjunto RandomRBF.

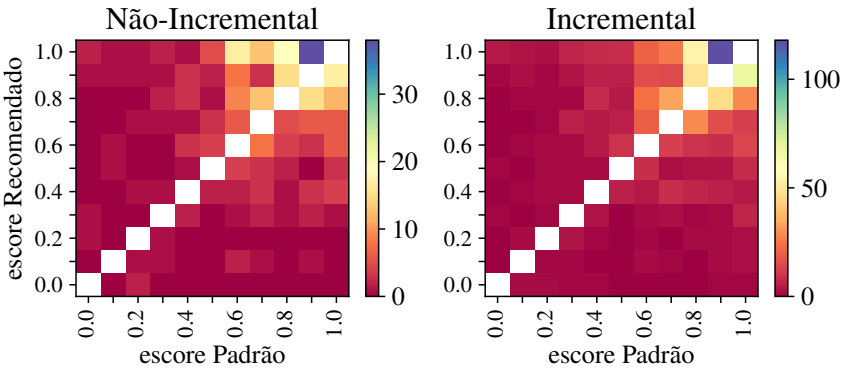


Figura 4.8: Comparação entre o método recomendado e o padrão para o Electricity.

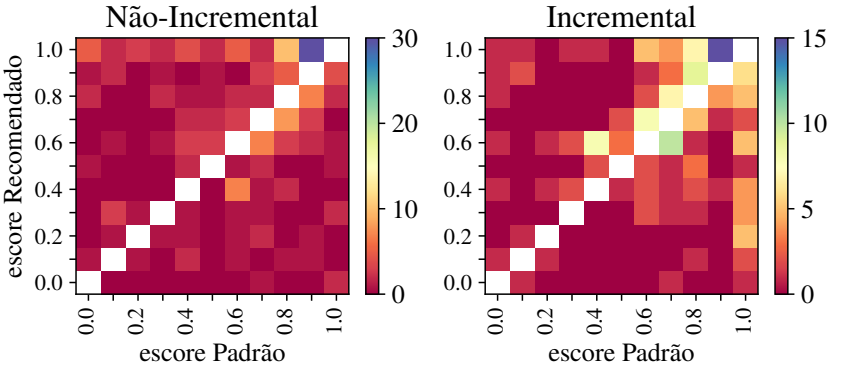


Figura 4.9: Comparação entre o método recomendado e o padrão para o CoverType.

Para o *Covertype* (Figura 4.9), nós observamos um comportamento diferente, há um quadrado azul acima da diagonal para o gráfico não-incremental com uma grande quanti-

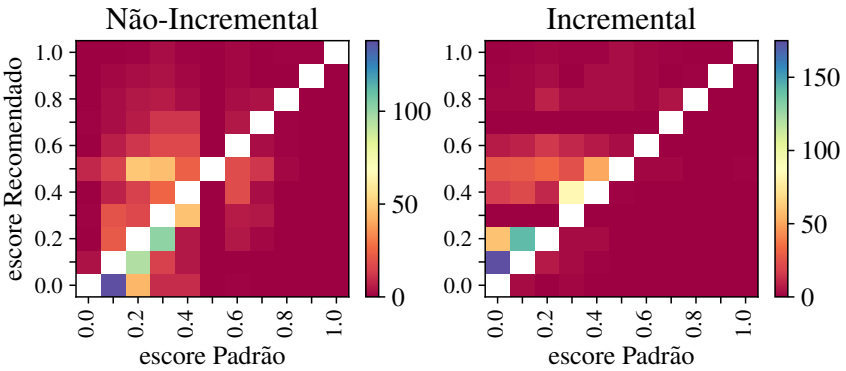


Figura 4.10: Comparação entre o método recomendado e o padrão para o PowerSupply.

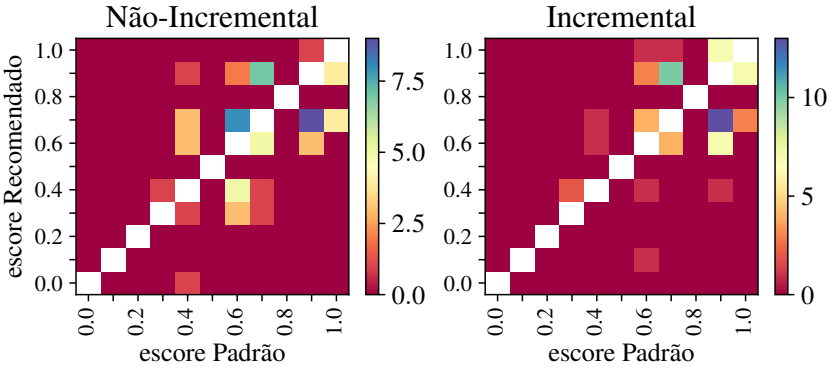


Figura 4.11: Comparação entre o método recomendado e o padrão para o HyperPlane.

dade de pontos e um quadrado verde claro abaixo da diagonal no incremental. Porém, a quantidade de pontos nessas regiões não é significativa em razão aos mil pontos do ganho acumulado e podemos ver que para esse conjunto de dados e a respectiva configuração do experimento o Metastream falha em recomendar o melhor algoritmo 4.3.

O *PowerSupply*, que tem 24 classes, é um dos mais difíceis de classificar, dado que todos os pontos tem acurácia abaixo de 0.8 como mostrado na Figura 4.10. Entretanto, no nível meta, o recomendador teve grande sucesso em discriminar os algoritmos, tendo em vista que é um meta-dado balanceado, isso se refletiu no maior ganho por recomendação entre os conjuntos de dados reais, confirmado pela Figura 4.4.

No histograma da Figura 4.11, podemos ver que ambos os métodos fizeram poucas recomendações para esse conjunto de dados, sendo a meta-base composta por quase 80% de exemplos da classe majoritária o meta-classificador não consegue aprender muito bem nessa configuração.

Para o conjunto *Agrawal* temos um gráfico bem assimétrico para cima da diagonal, tendo regiões com mais de 100 pontos, resultando assim em um ganho significativo ao longo do tempo.

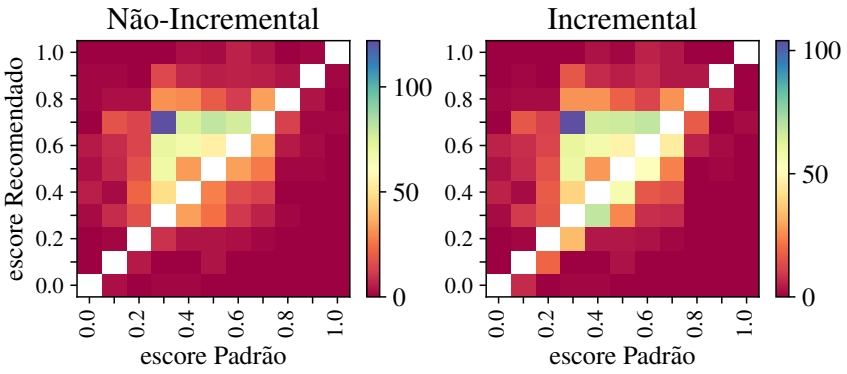


Figura 4.12: Comparação entre o método recomendado e o padrão para o Agrawal.

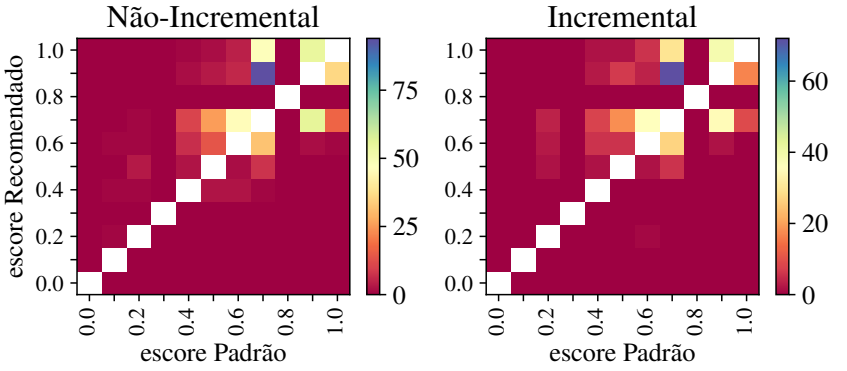


Figura 4.13: Comparação entre o método recomendado e o padrão para o RandomRBF.

Já o conjunto *RandomRBF* tiveram menos pontos acima da diagonal mas ainda sim os pontos de máximo se localizam acima da diagonal, tendo um ganho consistentemente positivo.

A importância dos atributos da estratégia incremental é dada pela Figura 4.1, dado que o método incremental mantém os mesmos atributos dos nós do treinamento nos dados *offline*, alterando apenas seus valores. Já no caso do método não-incremental uma nova árvore é construída do zero, podendo então variar a importância dos atributos. Como apresentado na Figura 4.14, há uma variação significativa dessa importância para cada janela de treinamento. Acreditamos que fixar a importância dos atributos acarreta em uma forma de regularização sobre o algoritmo. Entretanto, alternativamente, mudanças de conceito podem ocorrer a nível meta, sendo então possivelmente benéfico permitir que a escolha da árvore seja mais flexível.

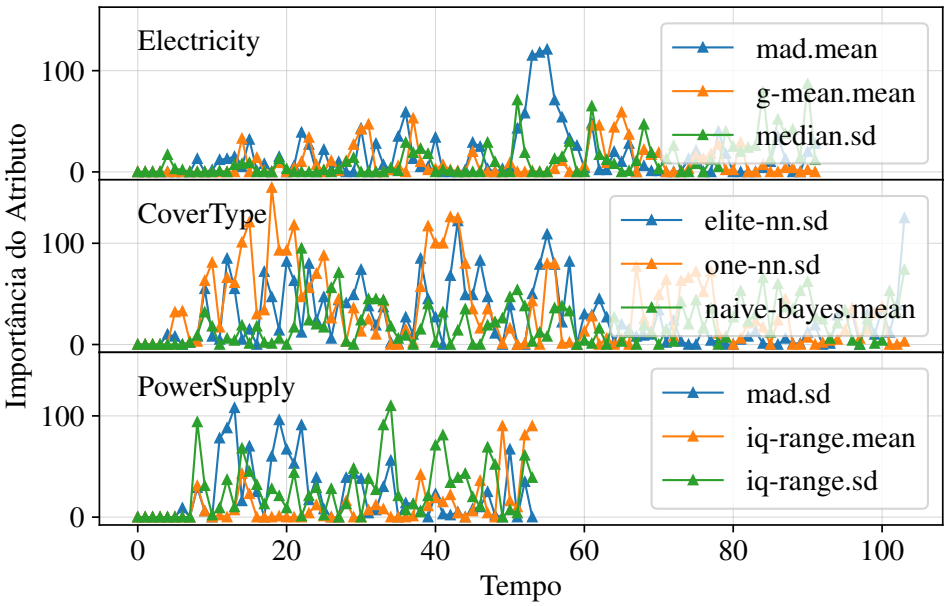


Figura 4.14: Variação na importância dos atributos ao longo do tempo para três meta-atributos.

Capítulo 5

Conclusão

Nesse trabalho, foi apresentado um método de MtA baseado no *framework* MetaStream. Este método aprimorou o MetaStream original pela adição de meta-atributos mais modernos e informativos e também pela inclusão do aprendizado incremental no nível MtA por meio do LightGBM como meta-classificador. Embora ambas as estratégias tenham desempenhado de maneira similar, a estratégia incremental apresenta maior robustez devido seu menor tempo para indução do modelo e menor uso de memória. Os resultados experimentais apresentaram que o meta-classificador pode consistentemente recomendar o melhor algorítimo para uma dada janela no fluxo de dados, conduzindo a um ganho acumulado de performance ao longo do tempo.

Como sugestões para trabalhos futuros, é possível estender o conjunto de algoritmos nível base para mais de duas classes, possibilitando a inclusão de mais vieses a serem selecionados para o nível base. Outra possibilidade é utilizar classificadores incrementais no nível base, melhorando ainda mais a performance desse framework. A extração de meta-atributos de forma incremental também é possível, incorporada ao MetaStream, permitiria a inclusão de algoritmo que no modo tradicional, em lote, seriam custoso demais para serem adicionados como já foi discutido. Alternativamente, embora tenham sido utilizados meta-atributos tradicionais, meta-atributos de séries temporais podem apresentar um grande poder discriminativo para esse tipo de problema, melhorando a performance meta do sistema. Por fim, uma abordagem recente é regredir modelos baseado em suas acurácias, como forma de recomendação, permitindo treinar de forma irrestrita classificadores a nível base sem as limitações de insuficiência estatística.

Algorithm Recommendation for Data Streams

Jáder M. C. de Sá
Department of Computer Science
University of Brasilia
Brasília, Brazil
jader.martins@ipea.gov.br

Andre L. D. Rossi
São Paulo State University (UNESP)
Itapeva, SP, Brazil
andre.rossi@unesp.br

Gustavo E. A. P. A. Batista
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
g.batista@unsw.edu.au

Luís P. F. Garcia
Department of Computer Science
University of Brasilia
Brasília, Brazil
luis.garcia@unb.br

***Abstract*—In the last decades, many companies have taken advantage of knowledge discovery to identify valuable information in massive volumes of data generated at high frequency. Machine learning techniques can be employed for knowledge discovery since they can extract patterns from data and induce models to predict future events. However, dynamic and evolving environments usually generate non-stationary data streams. Hence, models trained in these scenarios may perish over time due to seasonality or concept drift. Periodic retraining can help, but a fixed hypothesis space may no longer be appropriate. An alternative solution is to use meta-learning for regular algorithm selection in time-changing environments, choosing the bias that best suits the current data. In this paper, we present an enhanced framework for data stream algorithm selection based on MetaStream. Our approach uses meta-learning and incremental learning to actively select the best algorithm for the current concept in a time-changing environment. Different from previous work, we use a rich set of state-of-the-art meta-features, and an incremental learning approach in the meta-level based on LightGBM. The results show that this new strategy can improve the recommendation accuracy of the best algorithm in time-changing data.**

I. Introduction

Clicks in web pages, user interaction in streaming services and IoT devices typically generate data at a high frequency. In the era of Big Data [1], [2], many companies are taking advantage of massive databases to aggregate valuable information to their users, such as movie recommendation and content filtering in social media. Machine Learning (ML) is one of the research areas that stands out for knowledge discovery. ML extracts patterns from data through model induction, enabling the prediction of future events [3].

As every inductive method, traditional ML techniques have two fundamental assumptions: (i) the future must resemble the past and (ii) the future events must be independent of past events [4]. However, in continuous data flow (also known as a data stream), the underlying data distribution may change over time. Thus, these assumptions could not stand, and the accuracy of induced models could diminish as time passes, causing poor experiences to users of those systems [5], [6].

Most of the existing data stream approaches rely on a single model or an ensemble of models with the same base classifier. These approaches have a fundamental assumption that a unique hypothesis space is best for all the concepts. However, we have more predictive power by learning different concepts with different learning paradigms [7]. A central question is how to efficiently choose the most appropriate learning algorithm for a given data sample.

Meta-Learning (MtL) is a prominent technique to learn with changing distributions by detecting concept drifts and recommending algorithms to improve prediction [8], [9], [10]. MtL requires meta-data [11] in the form of a set of descriptive characteristics, named meta-features, of the problem at hand [12]. These meta-features are extracted and combined into a meta-example. The central idea of MtL is to apply different ML algorithms to the meta-data and have their performances recorded as labels of the meta-examples. This procedure is performed for a data stream at different time points. Finally, we use such labelled meta-dataset to induce a meta-classifier that predicts which is the most appropriate algorithm for future events. MtL avoids the use of expensive model selection processes, common in the ML literature [13].

In this paper, we enhance MetaStream [14], [7], an MtL framework based on periodic algorithm selection on data streams. Our contributions are two-fold: (i) We augment the meta-feature set with a broader set of state-of-the-art features [12], and, (ii) We include an incremental learner in the MtL level using LightGBM [15] as meta-classifier. Therefore, this investigation answers whether the use of state-of-the-art meta-features allied to incremental learning provided by LightGBM can predict more accurately than a baseline method and improve the general performance of the learning system.

This paper is organized as follows. Section II discusses the relevant literature, including MetaStream and the enhancements proposed in this paper for this framework. Section III presents the hyperparameters, meta-features and datasets used to evaluate the proposed methodology. Section IV shows the experimental measures and analyses the evaluation results. Finally, Section V concludes our work and present directions future research.

II. Related Work

This section presents the background information necessary to describe the proposed approach: Section II-A explains about data streams problems and how to deal with concept drifts and retraining. Section II-B presents the MtL framework, including the process of building meta-data and how to recommend

algorithms. Section II-C describes the MetaStream, a recommendation system based on MtL for data streams.

A. Data Streams

Most ML techniques assume data are independent and identically distributed (*i.i.d.*). However, this assumption usually does not hold in data stream environments [16]. Data in the form of a stream arrive sequentially and can present temporal dependency [17] and concept drifts. As data streams are potentially infinite, they impose restrictions on computational resources, such as memory and CPU. For instance, we cannot expect to store the entire stream in memory as in batch learning, and we need to process the events quickly to cope with fast-paced streams [18], [19]. These challenges have motivated an entire community to propose a diverse set of methods and strategies. The most prominent ones are forgetting mechanisms, statistical testing and adaptive algorithms.

Forgetting mechanisms make recent data more relevant than outdated data. A well-known example is sliding windows [20], a popular technique applicable to any ML technique. Statistical tests based systems act monitoring a predefined measure, such as models predictive performance, and alarms when this quality is below some tolerated threshold, demanding some actions. Examples of this approach are Page-Hinkley Test and Statistical Process Control [21]. Based on these two solutions, ADaptive WINdowing (ADWIN) [16] uses a statistical test to set each window size, adjusted to the "concept size". Although these methods can alarm a reduction in models' performance and adapt the sliding window, it is not possible to identify the algorithms or models that have become more suitable than the current one for the most recent data.

Another approach is the development of adaptive learning algorithms, such as VFDT [22]. VFDT was introduced as an online learning algorithm with low memory consumption, but not capable of adapting to concept drifts. Later, several enhancements addressed this issue, such as CVFDT (a direct improvement over VFDT) [23], HAT [24] and ARF [25], which can handle varying concepts by applying sliding windows in the training process. However, these methods have a fundamental assumption that a fixed hypothesis space (Decision Tree (DT) in the case of the mentioned approaches) is adequate for all concepts in the stream. In this paper, we can address this drawback by using change detectors based on MtL recommendation systems, targeting hypothesis spaces as learning tasks [7].

B. Meta-Learning

The algorithm selection problem was initially addressed by Rice (1976) [26] with the main goal to predict the best algorithm to solve a given problem when more than one algorithm is available. The components of this model are: the problem instances space (P), which is composed by datasets in MtL; the instance features space (F), which are the meta-features used to describe the datasets; the algorithms space (A), which contains the pool of ML algorithms for recommendation; and the evaluation measures space (M), responsible for assessing the performance of the ML algorithms in solving the problem instances contained in P. The MtL system implements an algorithm that maps a dataset $p \in P$, described by the meta-features $f \in F$, into one (or more) algorithm $\alpha \in A$ able to solve the problem with a good predictive performance according to $m \in M$, i.e., with maximum $m(\alpha(p))$.

Smith-Miles (2008) [27] improved this abstract model by proposing generalizations that can also be applied to the algorithm design problem. This proposal adds some components: the set of MtL algorithms; the generation of empirical rules or algorithm rankings; and, the examination of the empirical results, which may guide theoretical support to refine the algorithms.

One crucial component of the previous models is the definition of the set of meta-features (F) used to describe the general properties of datasets. These meta-features should provide evidence about the future performance of the algorithms in A [28], [29] and discriminate the performance of a group of algorithms with a low computational cost. We can divide the meta-features commonly used in the MtL literature into five groups: General, Statistical, Information-theoretic, Model-based and Landmarking.

General features are easily extracted from data [30], with low computational cost [29]. The statistical meta-features capture the main indicators of data localization and distribution, such as average, standard deviation, correlation, and kurtosis. Information-theoretic meta-features are based on information theory [31], usually entropy estimates [32], which capture the amount of information in (subsets of) a dataset [27]. Model-based meta-features are properties obtained from a model; usually, a DT [33], [34] induced from the data under analysis [30]. The Landmarking meta-features use the performance of simple and fast learning algorithms to characterize the datasets [27].

The definition of the set of problem instances (P) is another concern. The ideal scenario is to use a large number of diverse datasets to induce a reliable meta-model. To reduce the bias of this choice, datasets from several data repositories, such as UCI[1] [35] and OpenML[2] [36], can be used.

The algorithm space A represents a set of candidate algorithms to be recommended in the algorithm selection process. Ideally, these algorithms should also be sufficiently different from each other and represent all regions in the algorithm space [13]. Different measures can evaluate the models induced by the algorithms. For classification tasks, most of the MtL studies use accuracy. However, other indices, such as F_β, AUC and kappa coefficient, can also be used.

The next step is labelling each meta-example in the meta-data. Brazdil et al. (2009) [37] summarize the four main properties frequently used to label the meta-examples in MtL: (i) the algorithm that presented the best performance on the dataset (a classification task); (ii) the ranking of the algorithms according to their performance on the dataset (a ranking

[1] https://archive.ics.uci.edu/ml/index.php
[2] http://www.openml.org/

classification task); (iii) the performance value obtained by each evaluated algorithm on the dataset (a regression task) and; (iv) the model description, which is in general based on clustering and association rules.

In data stream environments, the problem instance space is composed of only one problem p. At the same time, the meta-features F are extracted from the data of each time window to form a meta-example. The meta-features extraction process should have a low computational cost and high information degree. Besides that, the performance of the algorithms A is assessed periodically, usually for each sliding window. Therefore, the meta-model replaces the current algorithm as soon as it predicts a different one is more suitable for the examples of the next sliding window [38], [39], [8].

In this paper, we present an MtL based method based on the MetaStream framework [7]. The basic idea is selecting the best algorithm for time-changing environments. MetaStream regularly induces a meta-classifier able to map the characteristics extracted from past and incoming data to the performance of classifiers on these data. Different from previous works, we use a set of state-of-the-art meta-features based on A. Rivolli et al. (2018) [12] and an incremental learning approach in the meta-level based on LightGBM [15].

C. Metastream

The literature presents alternatives to algorithm (or model) selection based on MtL for concept drift. One of the first to present it is Klinkenberg (2005) [40], where characteristics obtained from the learning process, such as sliding window sizes, were used to induce meta-models. Gama and Kosina (2014) [41] and Anderson et al. (2019) [8] investigated a different approach by reusing previously learned models but using the same algorithm to induce those models in the data stream. A third alternative is given by van Rijn et al. (2018) [42], where they use ensembles of models, having a high computational cost of inducing models for every algorithm and keep the weights updated.

Next, we present the MetaStream framework based on Rossi et al. (2014) [7], which has "offline" and "online" phases. The offline stage performs hyperparameter tuning, validation and training data generation with a small initial sample of data. The online phase acts in the dynamic environment recommending an algorithm for a given window of the data. Both stages are based on MtL recommendation systems.

1) Offline Phase: This phase starts after a given initial amount of training data has arrived. With this batch of data, Metastream induces the base-algorithms through k-fold cross-validation for hyperparameter tunning. With the same initial data, the algorithm continues by swiping a sliding window through the data, as shown in Figure 1. In this setting, Metastream induces base models using the base-level algorithms and extracts meta-features x^m for each window ω_{bi}.

Then, Metastream evaluates the induced models on the examples of the η_{bi} window. The best performing model becomes the label of meta-example (y^m). The same process repeats N steps further, where N is a specified number of

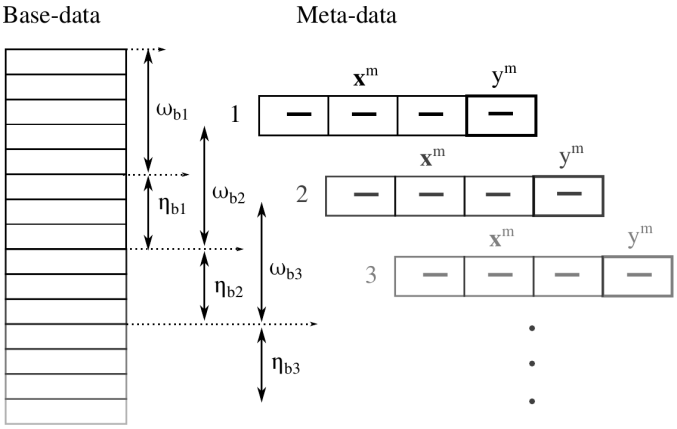


Fig. 1. Meta-feature extraction from ω_b and label obtaining from η_b windows.

instances to generate the initial meta-data, which Metastream uses to create the first meta-model.

2) Online Phase: In the online phase, the algorithm receives a continuous stream of data. At first, it gets a feature vector $\boldsymbol{x}_b = (x_1, ..., x_p)$, and with some delay, the target attribute $y_b \in \{1, 2, .., k\}$ for classification, where k is the number of classes.

Figure 2 shows the time t of the online phase. It has a window of fixed size ω_b that is used to induce the model and a window of fixed size η_b where the model induced on ω_b is evaluated. When Metastream processes all examples in η_b, it shifts the ω_b and η_b windows η_b instances to the right. Afterwards, Metastream induces a new model for this window.

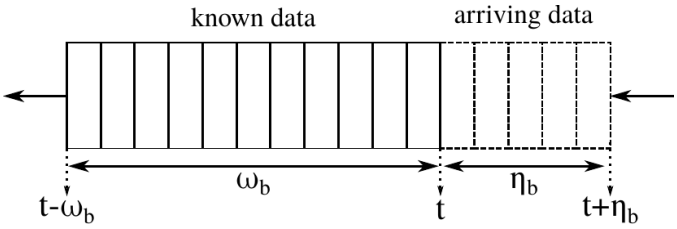


Fig. 2. Window discretization at base-level data stream.

MtL comes to play in the second level of processing, named meta-level. Figure 3 shows the meta-features extracted from ω_b windows. Metastream generates a meta-example without the target value, which is updated later.

The learning algorithm (meta-model) uses previously labelled meta-examples in the meta-base to induce a meta-classifier. The meta-classifier recommends an algorithm to induce a model using the ω_b window. Assuming the meta-model is accurate, it will likely suggest the fittest classifier to predict the labels of the η_b examples.

III. Methodology

As described in Section II-B, our problem of algorithm selection is given by choosing the most suitable hypothesis space according to a recent data window. Thus, in the presence

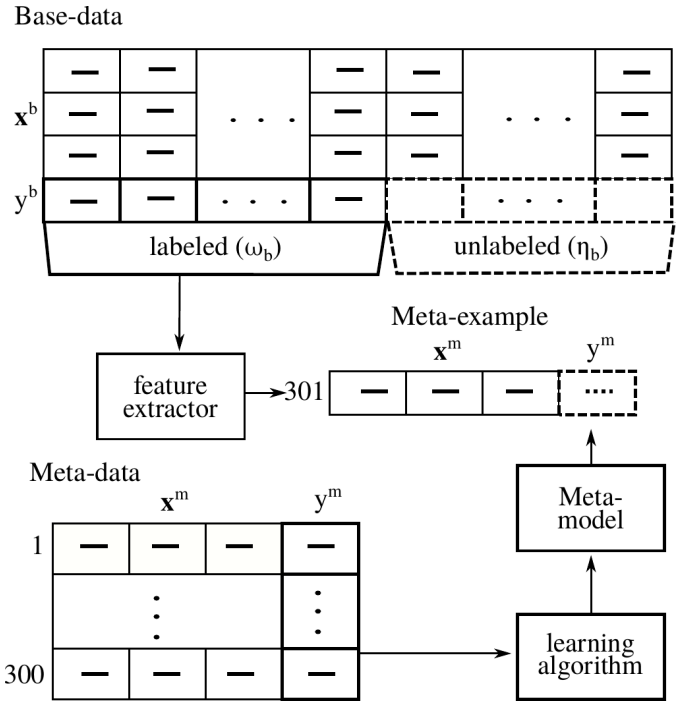


Fig. 3. Meta-feature extraction from ω_b and η_b windows at meta-level.

of concept drifts, we expect our proposal will recommend an alternative and more fit classifier to replace the current one.

Many theories have been developed to address the analysis of hypothesis spaces [4], [43], pointing directions to guide our choice. Another aspect to consider is the computational costs, memory and training/prediction time. This concern has led us to select Random Forest (RF) [44] and an efficient approximation of RBF-SVM through Nyströem method [45] as the base-level learning algorithms.

Therefore, our meta-dataset has labels according to the predictive performance of the models induced by RF and SVM for the examples in the η_b window. At the same time, the meta-classifier seeks to which one of these two classifiers is the most appropriate for the next η_b window.

As the meta-classifier, i.e., the recommender, we chose LightGBM for three main reasons. First, LightGBM presents state-of-the-art predictive performance for different problems [15]. Second, this algorithm automatically deals with missing features, which commonly occur for the extracted meta-features. Third, it has incremental learning capacity and low requirements for processing and memory[3], encouraging this choice for online applications. Incremental and batch learning has a significant distinction. While the first maintains trees and updates the values on the leaves, the second creates trees from scratch. Not only the training time for both differ, but also the incremental learning fixes the feature importance *apriori*.

For the meta-features, three groups from Section II-B were analyzed, namely statistical, model-based and landmarking. The statistical meta-features capture indicators of localization and distribution, the model-based extract properties of the DT models, while the landmarking use the performance of simple and fast learning algorithms to characterize the datasets [12]. In general, they have high discriminatory property and an average computational cost. For those experiments, we used the meta-features available in the Python Meta-Feature Extractor (pymfe) package [46] publicly available at GitHub[4].

We fixed the size of the windows across all experiments, in both base and meta-levels, $\omega_{m;b} = 300$ and $\eta_{m;b} = 10$. The training window step is the same as η. We perform a hyperparameter tuning in the base-level algorithms through k-fold cross-validation. We kept the hyperparameter values fixed for every future induction. Although this decision may not lead to optimal results, we believe that those hyperparameters values are a better choice than performing no selection at all (*i.e.*, using the default values). We use accuracy, Kappa score and Geometric Mean to assess the predictive performance of the induced models in both offline and online phases. , Kappa and Geometric Mean are more appropriate than accuracy to evaluate unbalanced class distributions, which is frequently the case of meta-data.

For the experiments, we selected three popular classification datasets in the data stream literature that present concept drift [39], [38], *Electricity* [47], *Power Supply*[48] and *Covertype* [49].

- *Electricity:* The Electricity dataset was collected from the Australian New South Wales Electricity Market. The prices fluctuate according to demand and supply, presenting seasonal factors. It contains 45,312 instances dated from 7 May 1996 to 5 December 1998 (set every 5 minutes). The target value is if the demand was higher or lower compared to the previous 24 hours and as predictive attributes the day of the week, the time stamp, the New South Wales electricity demand, the Victoria electricity demand, the scheduled electricity transfer between states.
- *Covertype:* This dataset contains the cartographic variables for 30 x 30-meter cells obtained from US Forest Service, it originally had 581,012 instances, but we reduced to 19,999 to minimize processing time. The target value is the cell's forest cover type (classes ranging from 1 to 7), and as predictive attributes, it has 53 properties of the cell including elevation, slope, soil type, etc.
- *PowerSupply:* Power Supply dataset contains records from 1995 to 1998, the task is to predict which hour (class) the current power supply belongs to (1 to 24), which is related to seasonal factors and trends, as predictive attributes we have the energy production in the main and auxiliary power grid. It has 29,928 instances, but we reduced to only 19,999.

The experimental analysis has an offline and online phase. The offline stages analyze the framework in the meta-level, that is, how well it can predict the best algorithm given the descriptive meta-features. The online phases analyze, in the base-level, the gains obtained by selecting the recommended

[3]https://lightgbm.readthedocs.io/en/latest/Experiments.html

[4]https://github.com/ealcobaca/pymfe

base algorithm, compared against a baseline method, namely Default, which represents the majority performance. For that, we look at the cumulative sum of performance gains over time, which is given by the difference in performance between using the recommended algorithm and using the Default one.

IV. Results

Similar to Section III, the results are divided into offline and online phases. In the offline evaluation, we analyze the meta-classifier to assess how well it discriminates classes at the meta-level. In the online assessment, we measure the gains regarding predictive performance at the base-level when employing the algorithms recommended by the meta-classifier.

A. Offline Analysis

Table I lists the distribution of algorithms (RF and SVM[5]) that presented the best performance for each batch of the data streams (*Electricity*, *CoverType* and *PowerSupply*). Under the perspective of MtL, this table shows the class distribution of the meta-examples for the offline meta-data.

TABLE I
ALGORITHM DISTRIBUTION IN META-DATA PER DATA STREAM PROBLEM.

	Electricity	CoverType	PowerSupply
SVM	0.752	0.718	0.535
RF	0.248	0.282	0.465

The algorithms RF and SVM presented distinct performances. For both *Electricity* and *CoverType* datasets, the class distribution is uneven, and, thus, the meta-classifier was trained to balance the loss for target proportion. On the other hand, the *PowerSupply* dataset is highly balanced.

We assessed the meta-classifier predictive performance using a time-series cross-validation [50] in the initial data. The model is induced using the first window of $\omega_m = 300$ train examples and predicts the label of the test window, composed of the $\eta_m = 10$ examples, then both windows slides by 1 example for the next evaluation. Table II presents the average and standard deviation for these metrics. Gains of meta-recommendation comes from non-trivially recommending the algorithms. Since the Default method represents a Kappa value equals to zero, $Kappa = 0$, $Kappa > 0$ means a prediction gain.

TABLE II
PREDICTIVE PERFORMANCE OF THE META-CLASSIFIER FOR THE OFFLINE PHASE.

	Kappa	G.Mean	Accuracy
Electricity	0.132±0.408	0.164±0.346	0.704±0.186
CoverType	0.322±0.517	0.330±0.470	0.752±0.177
PowerSupply	0.308±0.477	0.364±0.447	0.523±0.170

Table II presents the predictive performance in the offline phase assessed by three different measures using sliding window setting. The accuracy is estimated under $\eta_m = 10$, which is comparable to the meta-data distribution from Table I. For the Kappa and Geometric Mean measures, we removed the examples in η_m where there was no meaningful difference between the performance of the algorithms in the base level, i.e., we computed these measures for base examples where $|\text{Acc(RF)} - \text{Acc(SVM)}| > 0.1$. This approach was employed aiming to produce a more reliable estimate of the meta-classifier, i.e. when there is no clear distinction between models performance, any one of them could be chosen.

For all datasets, Kappa is at least slightly positive, showing that the recommender can select the correct algorithm when there is a patent difference between their predictive performance, outperforming the default method. Accuracy is close to dataset majority distribution, but later we argue that some examples present better gains to predict than others, as in base-level it has a significant difference.

To assess how the meta-features contribute to the recommendation problem, Figure 4 shows the 5 top-ranked meta-features, for each dataset, selected by LightGBM. In this figure, the x-axis represents the LightGBM feature importance in the meta-models, and the y-axis shows the top-ranked meta-features. The vast majority of meta-features selected to split nodes in the growing trees were from the landmarking group, followed by model-based and statistical meta-features. Those groups have the most informative and discriminative power meta-features according to the literature [12].

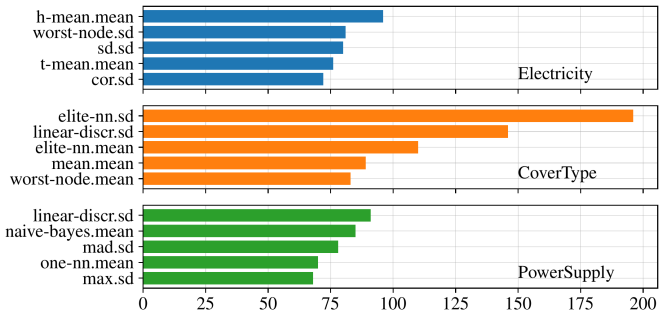


Fig. 4. Feature importance for the meta-algorithm in the offline data.

B. Online Analysis

In the online phase, we evaluate the meta-classifier (LightGBM) using a non-incremental and an incremental strategy. In the former evaluation, we train the meta-classifier using the procedure described in Section II-C2. When new meta-examples are available, the meta-classifier is retrained from scratch, and this procedure can lead to a model that is entirely independent of the previous models. In the incremental strategy, LightGBM updates the previous model, adapting the DTs to fit the meta-examples from the most recent window. Table III reports the results.

Although Table III presents relatively small Kappa values, we note that these values are positive for every combination of dataset-strategy. These results indicate that the meta-classifier can select the most appropriate algorithm compared to the majority class. This outcome is more evident for the

[5]An approximation for the SVM described in the methodology section.

TABLE III
PREDICTIVE PERFORMANCE OF THE META-CLASSIFIER FOR THE ONLINE PHASE.

Dataset	Strategy	Kappa	G.Mean	Accuracy
Electricity	Non-Incremental	0.013	0.326	0.670
	Incremental	0.023	0.510	0.645
CoverType	Non-Incremental	0.085	0.425	0.689
	Incremental	0.034	0.315	0.695
PowerSupply	Non-Incremental	0.067	0.556	0.565
	Incremental	0.006	0.466	0.524

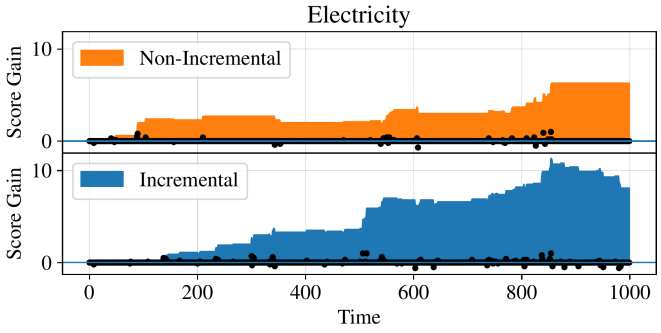


Fig. 5. Cumulative score gain over time for Electricity.

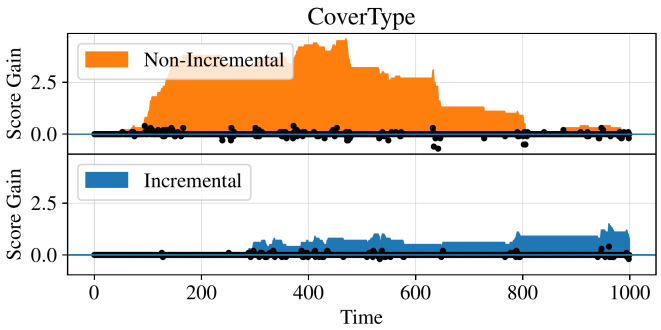


Fig. 6. Cumulative score gain over time for CoverType.

PowerSupply dataset, which presented the highest Geometric Mean and positive Kappa. This consistency reflects later in the cumulative gains of this dataset. The incremental strategy presented better performance than the non-incremental one for *PowerSupply* and *CoverType*, but for the *Electricity* it was worse for this metrics.

Figures 5-7 show the cumulative gain, which is the difference between the recommended algorithm accuracy and the Default method accuracy. The filled area is the cumulative sum of those score differences over time while the colours orange and blue represent incremental and non-incremental algorithms, respectively. Each black dot represents the score gain for time t.

In these figures, we observe a positive gain for the *Electricity* and *PowerSupply* datasets considering the recommendations of both strategies. However, the incremental strategy presents better performance for *Electricity* as opposed to *PowerSupply*, where the non-incremental is slightly better. In

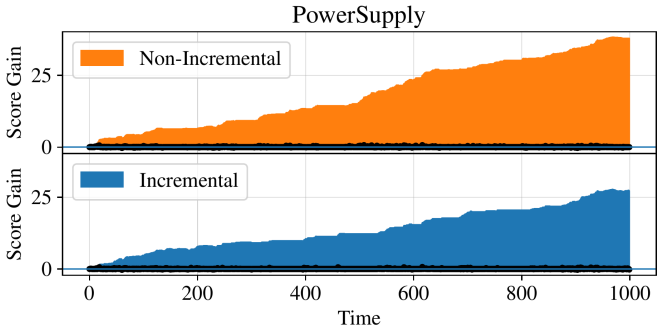


Fig. 7. Cumulative score gain over time for PowerSupply.

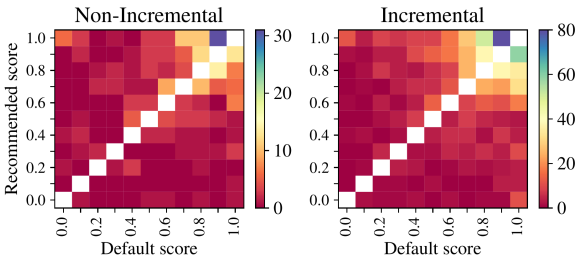


Fig. 8. Comparison between recommended and Default for Electricity.

the *CoverType* dataset we notice an interesting aspect. The non-incremental strategy presents a greater gain compared to the incremental one for lower points in time, but around $t = 450$, it starts to make incorrect recommendations. On the contrary, the incremental strategy is more constant, keeping gains slightly over time after $t-300$. This is partly determined by the difficulty of the dataset, mainly after $t - 400$, since the label that should be learned to infer the next window is not present in the training data. This is particularly difficult for the non-incremental strategy since the incremental one keeps the past learned target information.

In Figures 8, 9 and 10 the scores from Default are plotted against the scores from the recommended algorithms in a 2-dimensional histogram heatmap. Thus, each square in the heatmap represents the accuracy of the algorithm recommended by the Default method (x axis) and the meta-classifier (y axis) for the same window. The colour intensity as shown in the colour bar represents the number of points in that position. The points in the diagonal, where the recommended algorithm and the default algorithm had the same accuracy score, were removed and coloured as white to emphasize gains or losses of the framework. If the recommended classifier is always worse than the Default, i.e., recommending the minority class when it is not better than the majority, then all points will be below the white diagonal. Similarly, these points will be above if it is effectively selecting the algorithm from minority class when it is better than the majority.

In Figure 8, it is visible in the upper right region of the heatmap that the incremental algorithm is recommending the minority class more often than the incremental one, also the

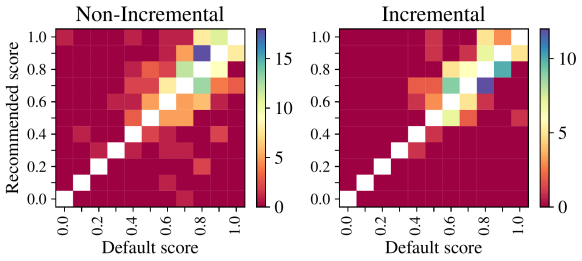


Fig. 9. Comparison between recommended and Default for CoverType.

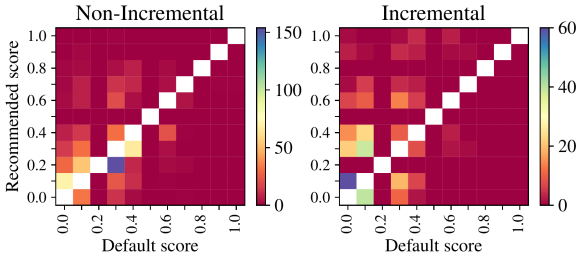


Fig. 10. Comparison between recommended and Default for PowerSupply.

blue square above the white diagonal has many more points (80 compared to 30 in the non-incremental), this reflects in gains as shown in Figure 5.

For the *Covertype* (Figure 9), we observe a different behaviour, there is a blue square above the diagonal for the non-incremental plot and a blue square below the diagonal in the incremental one. This points out that the incremental recommendation is performing worse than the default, as could be seen in Figure 6. The non-incremental also presents some light coloured squares below the diagonal, which resulted in reduced overall performance for later points in time.

The *PowerSupply* dataset, which has 24 classes, is the hardest one to classify, since most points are below 0.8 accuracy as shown in Figure 10. However, in the meta-level, was easier to discriminate algorithms as the meta-data is balanced, reflecting in a less symmetrical distribution for the diagonal (the upper region is more populated), it is also confirmed in Figure 7, with the highest cumulative score gain amongst the datasets.

The feature importance for the non-incremental strategy is given by Figure 4, as it keeps the same nodes obtained by training in the offline meta-data. The main difference between incremental and batch learning for this problem is that the feature importance is fixed in incremental learning. However, as shown in Figure 11, the importance varies for each trained window. It is believed that fixing feature importance acts as a regularization. Still, alternatively, concept drift could also occur in the meta-level, being this possibly beneficial to have a flexible tree choice.

V. Conclusion

In this paper, we presented a MtL method based on the MetaStream framework. We enhance MetaStream by extending the meta-features to modern and more informative ones, by including the incremental learning in the MtL level and by proposing LightGBM as meta-classifier. Although both strategies performed similarly, the incremental one had a significant lesser consumption of memory and processing time. The experimental results showed that the meta-classifier can consistently recommend the best algorithm for a given window in the data stream, leading to an increased gain of performance over time.

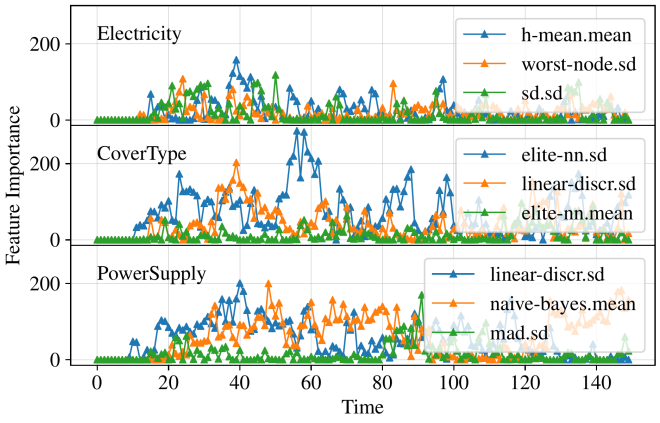


Fig. 11. Variation in feature importance over time for three meta-features.

As future work, we would like to rearrange the binary task to multi-class classification, enabling selecting between multiple hypothesis spaces in the base-level. Another idea is implementing a module for incremental meta-features, allowing to add more informative cutting edge meta-features. Also, as the original article proposed [7], time-series features could have significant discriminatory power. A recent approach is to perform recommendations based on regressing algorithms' accuracy which can be applied in this framework extending this study to more datasets.

Acknowledgements

The fourth author would like to thank FAPDF for the financial support (grant 40/2020). We also would like to thank E. Alcobaça and F. Siqueira, developers of the pymfe package, for implementing additional features in the package which help us in this study.

References

[1] B. Tarnoff. (2018) Big data for the people: it's time to take it back from our tech overlords. [Online]. Available: https://www.theguardian.com/technology/2018/mar/14/tech-big-data-capitalism-give-wealth-back-to-people

[2] L. Finger. (2014) 3 data products you need to know. [Online]. Available: https://www.forbes.com/sites/lutzfinger/2014/08/19/3-data-products-you-need-to-know

[3] T. M. Mitchell, *Machine Learning*, ser. McGraw Hill series in computer science. McGraw Hill, 1997.

[4] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[5] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[6] U. Johansson, C. Sönströd, H. Linusson, and H. Boström, "Regression trees for streaming data with local performance guarantees," in *IEEE International Conference on Big Data (Big Data)*, 2014, pp. 461–470.

[7] A. L. D. Rossi, A. C. P. L. F. de Carvalho, C. Soares, and B. F. de Souza, "MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data," *Neurocomputing*, vol. 127, pp. 52–64, 2014.

[8] R. Anderson, Y. S. Koh, G. Dobbie, and A. Bifet, "Recurring concept meta-learning for evolving data streams," *Expert Systems with Applications*, vol. 138, pp. 1–18, 2019.

[9] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Having a blast: Meta-learning and heterogeneous ensembles for data streams," in *IEEE International Conference on Data Mining (ICDM)*, 2016, pp. 1003–1008.

[10] M. N. Zarmehri and C. Soares, "Using metalearning for prediction of taxi trip duration using different granularity levels," in *14th International Symposium on Intelligent Data Analysis (IDA)*, 2015, pp. 205–216.

[11] J. Vanschoren, "Meta-learning: A survey," *eprint arXiv*, no. 1810.03548, pp. 1–29, 2018.

[12] A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, and A. C. P. L. F. de Carvalho, "Towards reproducible empirical research in meta-learning," *eprint arXiv*, no. 1808.10406, pp. 1–41, 2018.

[13] M. A. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles, "Instance spaces for machine learning classification," *Machine Learning*, vol. 107, no. 1, pp. 109–147, 2018.

[14] A. L. D. Rossi, A. C. P. L. F. Carvalho, and C. Soares, "Meta-learning for periodic algorithm selection in time-changing data," in *Brazilian Symposium on Neural Networks (SBRN)*, 2012, pp. 7–12.

[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *31st International Conference on Neural Information Processing System (NIPS)*, 2017, pp. 3149–3157.

[16] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *International Conference on Data Mining (SIAM)*, 2007, pp. 443–448.

[17] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Machine Learning*, vol. 98, no. 3, pp. 455–482, 2015.

[18] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[19] J. Gama, "A survey on learning from data streams: current and future trends," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 45–55, 2012.

[20] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 18–26, 2005.

[21] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[22] P. Domingos and G. Hulten, "Mining high-speed data streams," in *6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71–80.

[23] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 97–106.

[24] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *8th International Symposium on Intelligent Data Analysis (IDA)*, 2009, pp. 249–260.

[25] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[26] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[27] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–25, 2008.

[28] C. Soares, J. Petrak, and P. Brazdil, "Sampling-based relative landmarks: Systematically test-driving algorithms before choosing," in *10th Portuguese Conference on Artificial Intelligence (EPIA)*, 2001, pp. 88–95.

[29] M. Reif, "A comprehensive dataset for evaluating approaches of various meta-learning tasks," in *1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 2012, pp. 273–276.

[30] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel, "Automatic classifier selection for non-experts," *Pattern Analysis and Applications*, vol. 17, no. 1, pp. 83–96, 2014.

[31] C. Castiello, G. Castellano, and A. M. Fanelli, "Meta-data: Characterization of input features for meta-learning," in *2nd Modeling Decisions for Artificial Intelligence (MDAI)*, vol. 3558, 2005, pp. 457–468.

[32] S. Segrera, J. Pinho, and M. N. Moreno, "Information-theoretic measures for meta-learning," in *3rd Hybrid Artificial Intelligence Systems (HAIS)*, 2008, pp. 458–465.

[33] H. Bensusan, C. Giraud-Carrier, and C. Kennedy, "A higher-order approach to meta-learning," in *10th International Conference Inductive Logic Programming (ILP)*, 2000, pp. 1–10.

[34] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil, "Improved dataset characterisation for meta-learning," in *5th International Conference on Discovery Science (DS)*, vol. 2534, 2002, pp. 141–152.

[35] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[36] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[37] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning - Applications to Data Mining*, 1st ed., ser. Cognitive Technologies. Springer, 2009.

[38] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *11th International Symposium on Intelligent Data Analysis (IDA)*, 2012, pp. 313–323.

[39] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Algorithm selection on data streams," in *17th International Conference on Discovery Science (DS)*, 2014, pp. 325–336.

[40] R. Klinkenberg, "Meta-learning, model selection, and example selection in machine learning domains with concept drift," University of Dortmund, Tech. Rep., 2005.

[41] J. Gama and P. Kosina, "Recurrent concepts in data streams classification," *Knowledge and Information Systems*, vol. 40, no. 3, pp. 489–507, 2014.

[42] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "The online performance estimation framework: heterogeneous ensemble learning for data streams," *Machine Learning*, vol. 107, no. 1, pp. 149–176, 2018.

[43] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[44] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[45] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, 2001, pp. 682–688.

[46] E. Alcobaça, F. Siqueira, A. Rivolli, L. P. F. Garcia, J. T. Oliva, and A. C. P. L. F. de Carvalho, "MFE: Towards reproducible meta-feature extraction," *Journal of Machine Learning Research*, pp. 1–5, 2020.

[47] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *17th Brazilian Symposium on Artificial Intelligence*, 2004, pp. 286–295.

[48] X. Zhu, "Power supply: Stream data mining repository," 2010. [Online]. Available: http://www.cse.fau.edu/~xqzhu/stream.html

[49] J. A. Blackard, D. J. Dean, and C. W. Anderson, "Covertype data set, UCI machine learning repository," 1998. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/covertype

[50] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

Referências

[1] Ritchie, Hannah: *Technology adoption.* Our World in Data, 2017. https://ourworldindata.org/technology-adoption. 1

[2] Atzori, Luigi, Antonio Iera e Giacomo Morabito: *The internet of things: A survey.* Computer networks, 54(15):2787–2805, 2010. 1

[3] McAfee, Andrew, Erik Brynjolfsson, Thomas H Davenport, DJ Patil e Dominic Barton: *Big data: the management revolution.* Harvard business review, 90(10):60–68, 2012. 1, 5

[4] Dean, Jeffrey e Sanjay Ghemawat: *Mapreduce: simplified data processing on large clusters.* Communications of the ACM, 51(1):107–113, 2008. 1

[5] Halevy, Alon, Peter Norvig e Fernando Pereira: *The unreasonable effectiveness of data.* IEEE Intelligent Systems, 24(2):8–12, 2009. 1, 5

[6] Economist, The: *The world's most valuable resource is no longer oil, but data*, 2017. `https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data`, acesso em 2020-03-14. 1

[7] Tarnoff, Ben: *Big data for the people: it's time to take it back from our tech overlords*, 2018. `https://www.theguardian.com/technology/2018/mar/14/tech-big-data-capitalism-give-wealth-back-to-people`, acesso em 2020-03-14. 1

[8] Finger, Lutz: *3 data products you need to know*, 2014. `https://www.forbes.com/sites/lutzfinger/2014/08/19/3-data-products-you-need-to-know`, acesso em 2020-03-14. 1

[9] Gama, João e Mohamed Medhat Gaber: *Learning from data streams: processing techniques in sensor networks.* Springer, 2007. 1, 2, 18, 19

[10] Mitchell, Thomas M *et al.*: *Machine learning.* McGraw Hill series in computer science. McGraw-Hill, 1997. 1, 5, 6

[11] Friedman, Jerome, Trevor Hastie e Robert Tibshirani: *The elements of statistical learning.* Springer series in statistics New York, 2001. 1, 5, 6, 9

[12] Vapnik, Vladimir: *The nature of statistical learning theory.* Springer science & business media, 2013. 1, 7, 8, 13, 15, 27

[13] Johansson, Ulf, Cecilia Sönströd, Henrik Linusson e Henrik Boström: *Regression trees for streaming data with local performance guarantees*. Em *IEEE International Conference on Big Data (Big Data)*, páginas 461–470, 2014. 2

[14] Klinkenberg, Ralf e Thorsten Joachims: *Detecting concept drift with support vector machines*. Em *17th International Conference on Machine Learning (ICML)*, páginas 487–494, 2000. 2

[15] Zenisek, Jan, Florian Holzinger e Michael Affenzeller: *Machine learning based concept drift detection for predictive maintenance*. Computers & Industrial Engineering, 137:106031, 2019. 2

[16] Bifet, Albert e Ricard Gavaldà: *Learning from time-changing data with adaptive windowing*. Em *International Conference on Data Mining (SIAM)*, páginas 443–448, 2007. 2, 21

[17] Anderson, Robert, Yun Sing Koh, Gillian Dobbie e Albert Bifet: *Recurring concept meta-learning for evolving data streams*. Expert Systems with Applications, 138:112832, 2019. 2

[18] Ramanath, Rohan, Konstantin Salomatin, Jeffrey D Gee, Kirill Talanine, Onkar Dalal, Gungor Polatkan, Sara Smoot e Deepak Kumar: *Lambda learner: Fast incremental learning on data streams*. arXiv preprint arXiv:2010.05154, 2020. 2

[19] Zang, Wenyu, Peng Zhang, Chuan Zhou e Li Guo: *Comparative study between incremental and ensemble learning on data streams: Case study*. Journal Of Big Data, 1(5):1–16, 2014. 2

[20] Manapragada, Chaitanya, Geoffrey I Webb e Mahsa Salehi: *Extremely fast decision tree*. Em *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, páginas 1953–1962, 2018. 2, 22

[21] Cano, Alberto e Bartosz Krawczyk: *Kappa updated ensemble for drifting data stream mining*. Machine Learning, 109(1):175–218, 2020. 2

[22] Gama, João: *Knowledge discovery from data streams*. CRC Press, 2010. 2, 21

[23] Rossi, André L. D., André C. P. L. F. de Carvalho, Carlos Soares e Bruno Feres de Souza: *MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data*. Neurocomputing, 127:52–64, 2014. 2, 22, 24

[24] Anderson, Robert, Yun Sing Koh, Gillian Dobbie e Albert Bifet: *Recurring concept meta-learning for evolving data streams*. Expert Systems with Applications, 138:1–18, 2019. 2, 17, 24

[25] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer e Joaquin Vanschoren: *Having a blast: Meta-learning and heterogeneous ensembles for data streams*. Em *IEEE International Conference on Data Mining (ICDM)*, páginas 1003–1008, 2016. 2

[26] Zarmehri, Mohammad Nozari e Carlos Soares: *Using metalearning for prediction of taxi trip duration using different granularity levels.* Em *14th International Symposium on Intelligent Data Analysis (IDA)*, páginas 205–216, 2015. 2

[27] Vanschoren, Joaquin: *Meta-learning: A survey.* eprint arXiv, 1(1810.03548):1–29, 2018. 2

[28] Khan, Irfan, Xianchao Zhang, Mobashar Rehman e Rahman Ali: *A literature survey and empirical study of meta-learning for classifier selection.* IEEE Access, 8:10262–10281, 2020. 2

[29] Rivolli, Adriano, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren e André C. P. L. F. de Carvalho: *Towards reproducible empirical research in meta-learning.* eprint arXiv, 1(1808.10406):1–41, 2018. 2, 17, 18, 27, 33

[30] Lorena, Ana C, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto e Tin Kam Ho: *How complex is your classification problem? a survey on measuring classification complexity.* ACM Computing Surveys (CSUR), 52(5):1–34, 2019. 2

[31] Muñoz, Mario A, Laura Villanova, Davaatseren Baatar e Kate Smith-Miles: *Instance spaces for machine learning classification.* Machine Learning, 107(1):109–147, 2018. 2, 18

[32] Rossi, André L. D., Andre C. P. L. F. Carvalho e Carlos Soares: *Meta-learning for periodic algorithm selection in time-changing data.* Em *Brazilian Symposium on Neural Networks (SBRN)*, páginas 7–12, 2012. 2

[33] Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye e Tie Yan Liu: *LightGBM: A highly efficient gradient boosting decision tree.* Em *31st International Conference on Neural Information Processing System (NIPS)*, páginas 3149–3157, 2017. 2, 11, 12, 27

[34] *Experiments - comparison experiment.* `https://lightgbm.readthedocs.io/`. Accessed: 2020-03-31. 2, 12

[35] Domingos, Pedro e Geoff Hulten: *Mining high-speed data streams.* Em *6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 71–80, 2000. 3, 22

[36] Vinten-Johansen, Peter, Howard Brody, Nigel Paneth, Stephen Rachman, David Zuck, Michael Rip, Honorary Consultant Anaesthetist David Zuck *et al.*: *Cholera, chloroform, and the science of medicine: a life of John Snow.* Oxford University Press, 2003. 5

[37] Stigler, Stephen M: *The history of statistics: The measurement of uncertainty before 1900.* Harvard University Press, 1986. 5

[38] Russell, Stuart e Peter Norvig: *Artificial intelligence: a modern approach.* Elsevier, 2016. 5

[39] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski e Susan Zhang: *Dota 2 with large scale deep reinforcement learning*, 2019. 5

[40] Gibney, Elizabeth: *Google ai algorithm masters ancient game of go*. Nature News, 529(7587):445, 2016. 5

[41] Mohri, Mehryar, Afshin Rostamizadeh e Ameet Talwalkar: *Foundations of machine learning*. MIT press, 2018. 5, 6, 7

[42] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep learning*. MIT press, 2016. 5, 6, 18

[43] Gold, E Mark: *Language identification in the limit*. Information and control, 10(5):447–474, 1967. 5

[44] Valiant, Leslie G: *A theory of the learnable*. Communications of the ACM, 27(11):1134–1142, 1984. 6, 7, 27

[45] Abu-Mostafa, Yaser S, Malik Magdon-Ismail e Hsuan Tien Lin: *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012. 6

[46] Bishop, Christopher M: *Pattern recognition and machine learning*. springer, 2006. 6, 18

[47] Solomonoff, Ray J: *A formal theory of inductive inference. part i*. Information and control, 7(1):1–22, 1964. 6

[48] Kearns, Michael J e Umesh Vazirani: *An introduction to computational learning theory*. MIT press, 1994. 6

[49] Blumer, Anselm, Andrzej Ehrenfeucht, David Haussler e Manfred K Warmuth: *Occam's razor*. Information processing letters, 24(6):377–380, 1987. 6

[50] Li, Ming e Paul MB Vitányi: *Inductive reasoning and kolmogorov complexity*. Journal of Computer and System Sciences, 44(2):343–384, 1992. 6, 7

[51] Sipser, Michael: *Introduction to the Theory of Computation*. Cengage Learning, 2012. 6

[52] Solomonoff, Ray: *Complexity-based induction systems: comparisons and convergence theorems*. IEEE transactions on Information Theory, 24(4):422–432, 1978. 6

[53] Hutter, Marcus: *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2004. 7

[54] Bartlett, Peter L e Shahar Mendelson: *Rademacher and gaussian complexities: Risk bounds and structural results*. Journal of Machine Learning Research, 3(Nov):463–482, 2002. 7

[55] Blumer, Anselm, Andrzej Ehrenfeucht, David Haussler e Manfred K Warmuth: *Learnability and the vapnik-chervonenkis dimension*. Journal of the ACM (JACM), 36(4):929–965, 1989. 7

[56] Rathmanner, Samuel e Marcus Hutter: *A philosophical treatise of universal induction*. CoRR, abs/1105.5721, 2011. `http://arxiv.org/abs/1105.5721`. 7

[57] Breiman, Leo, Jerome Friedman, Charles J Stone e Richard A Olshen: *Classification and regression trees*. CRC press, 1984. 7, 8

[58] Ho, Tin Kam: *The random subspace method for constructing decision forests*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8):832–844, 1998. 8

[59] Breiman, Leo: *Random forests*. Machine Learning, 45(1):5–32, 2001. 8, 27

[60] Friedman, Jerome H: *Greedy function approximation: a gradient boosting machine*. Annals of statistics, páginas 1189–1232, 2001. 8

[61] Wolpert, David H e William G Macready: *No free lunch theorems for optimization*. IEEE transactions on evolutionary computation, 1(1):67–82, 1997. 16

[62] Wolpert, David H: *The lack of a priori distinctions between learning algorithms*. Neural computation, 8(7):1341–1390, 1996. 16

[63] Rice, John R.: *The algorithm selection problem*. Advances in Computers, 15:65–118, 1976. 16

[64] Smith-Miles, Kate A.: *Cross-disciplinary perspectives on meta-learning for algorithm selection*. ACM Computing Surveys, 41(1):1–25, 2008. 17, 18

[65] Dua, Dheeru e Casey Graff: *UCI machine learning repository*, 2017. `http://archive.ics.uci.edu/ml`. 17

[66] Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl e Luis Torgo: *OpenML: networked science in machine learning*. SIGKDD Explorations, 15(2):49–60, 2013. 17

[67] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer e Joaquin Vanschoren: *Algorithm selection on data streams*. Em *17th International Conference on Discovery Science (DS)*, páginas 325–336, 2014. 17, 24, 28

[68] Soares, Carlos, Johann Petrak e Pavel Brazdil: *Sampling-based relative landmarks: Systematically test-driving algorithms before choosing*. Em *10th Portuguese Conference on Artificial Intelligence (EPIA)*, páginas 88–95, 2001. 17

[69] Reif, Matthias: *A comprehensive dataset for evaluating approaches of various meta-learning tasks*. Em *1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, páginas 273–276, 2012. 17, 18

[70] Reif, Matthias, Faisal Shafait, Markus Goldstein, Thomas Breuel e Andreas Dengel: *Automatic classifier selection for non-experts*. Pattern Analysis and Applications, 17(1):83–96, 2014. 18

[71] Segrera, Saddys, Joel Pinho e María N. Moreno: *Information-theoretic measures for meta-learning*. Em *3rd Hybrid Artificial Intelligence Systems (HAIS)*, páginas 458–465, 2008. 18

[72] Bensusan, Hilan, Christophe Giraud-Carrier e Claire Kennedy: *A higher-order approach to meta-learning*. Em *10th International Conference Inductive Logic Programming (ILP)*, páginas 1–10, 2000. 18

[73] Peng, Yonghong, Peter A. Flach, Carlos Soares e Pavel Brazdil: *Improved dataset characterisation for meta-learning*. Em *5th International Conference on Discovery Science (DS)*, volume 2534, páginas 141–152, 2002. 18

[74] Coleman, Cody, Daniel Kang, Deepak Narayanan, Luigi Nardi, Tian Zhao, Jian Zhang, Peter Bailis, Kunle Olukotun, Chris Ré e Matei Zaharia: *Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark*. ACM SIGOPS Operating Systems Review, 53(1):14–25, 2019. 19

[75] Brazdil, Pavel, Christophe Giraud-Carrier, Carlos Soares e Ricardo Vilalta: *Metalearning - Applications to Data Mining*. Cognitive Technologies. Springer, 1ª edição, 2009. 19

[76] Babcock, Brian, Shivnath Babu, Mayur Datar, Rajeev Motwani e Jennifer Widom: *Models and issues in data stream systems*. Em *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, páginas 1–16, 2002. 19

[77] Read, Jesse: *Concept-drifting data streams are time series; the case for continuous adaptation*. arXiv preprint arXiv:1810.02266, 2018. 20

[78] Hyndman, Rob J. e George Athanasopoulos: *Forecasting: principles and practice*. OTexts, 2018. 20, 32

[79] Tsymbal, Alexey: *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin, 106(2):58, 2004. 20

[80] Brockwell, Peter J e Richard A Davis: *Introduction to time series and forecasting*. springer, 2016. 20

[81] Bifet, Albert, Geoff Holmes, Richard Kirkby e Bernhard Pfahringer: *MOA: Massive Online Analysis*. Journal of Machine Learning Research, 11:1601–1604, 2010. 21

[82] Gama, João: *A survey on learning from data streams: current and future trends*. Progress in Artificial Intelligence, 1(1):45–55, 2012. 21

[83] Gaber, Mohamed Medhat, Arkady Zaslavsky e Shonali Krishnaswamy: *Mining data streams: a review*. ACM SIGMOD Record, 34(2):18–26, 2005. 21

[84] Hulten, Geoff, Laurie Spencer e Pedro Domingos: *Mining time-changing data streams*. Em *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 97–106, 2001. 22, 28

[85] Bifet, Albert e Ricard Gavaldà: *Adaptive learning from evolving data streams*. Em *8th International Symposium on Intelligent Data Analysis (IDA)*, páginas 249–260, 2009. 22

[86] Gomes, Heitor M, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes e Talel Abdessalem: *Adaptive random forests for evolving data stream classification*. Machine Learning, 106(9-10):1469–1495, 2017. 22

[87] Bifet, Albert, Jiajin Zhang, Wei Fan, Cheng He, Jianfeng Zhang, Jianfeng Qian, Geoff Holmes e Bernhard Pfahringer: *Extremely fast decision tree mining for evolving data streams*. Em *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 1733–1742, 2017. 22

[88] Wang, Boyu e Joelle Pineau: *Online bagging and boosting for imbalanced data streams*. IEEE Transactions on Knowledge and Data Engineering, 28(12):3353–3366, 2016. 22

[89] Read, Jesse, Albert Bifet, Bernhard Pfahringer e Geoff Holmes: *Batch-incremental versus instance-incremental learning in dynamic and evolving data*. Em *11th International Symposium on Intelligent Data Analysis (IDA)*, páginas 313–323, 2012. 24, 28

[90] Klinkenberg, Ralf: *Meta-learning, model selection, and example selection in machine learning domains with concept drift*. Relatório Técnico, University of Dortmund, 2005. 24

[91] Gama, João e Petr Kosina: *Recurrent concepts in data streams classification*. Knowledge and Information Systems, 40(3):489–507, 2014. 24

[92] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer e Joaquin Vanschoren: *The online performance estimation framework: heterogeneous ensemble learning for data streams*. Machine Learning, 107(1):149–176, 2018. 24

[93] Bergstra, James e Yoshua Bengio: *Random search for hyper-parameter optimization*. The Journal of Machine Learning Research, 13(1):281–305, 2012. 24

[94] Williams, Christopher e Matthias Seeger: *Using the nyström method to speed up kernel machines*. Em *Advances in Neural Information Processing Systems*, páginas 682–688, 2001. 27

[95] Alcobaça, Edesio, Felipe Siqueira, Adriano Rivolli, Luís P. F Garcia, Jeferson T. Oliva e Andre C. P. L. F. de Carvalho: *MFE: Towards reproducible meta-feature extraction*. Journal of Machine Learning Research, páginas 1–5, 2020. 27

[96] Gama, João, Pedro Medas, Gladys Castillo e Pedro Rodrigues: *Learning with drift detection*. Em *17th Brazilian Symposium on Artificial Intelligence*, páginas 286–295, 2004. 28

[97] Zhu, X.: *Power supply: Stream data mining repository*, 2010. `http://www.cse.fau.edu/~xqzhu/stream.html`. 28

[98] Blackard, Jock A., Denis J. Dean e Charles W. Anderson: *Covertype data set, UCI machine learning repository*, 1998. `https://archive.ics.uci.edu/ml/datasets/covertype`. 28

[99] Montiel, Jacob, Jesse Read, Albert Bifet e Talel Abdessalem: *Scikit-multiflow: A multi-output streaming framework*. Journal of Machine Learning Research, 19(72):1–5, 2018. `http://jmlr.org/papers/v19/18-251.html`. 28

[100] Agrawal, Rakesh, Tomasz Imielinski e Arun Swami: *Database mining: A performance perspective*. IEEE transactions on knowledge and data engineering, 5(6):914–925, 1993. 28, 29