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

Meta aprendizado para detecção de mudança de conceito não supervisionada

Fernanda A. Melo

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Luís Paulo Faina Garcia

Brasília
2023

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

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

Meta aprendizado para detecção de mudança de conceito não supervisionada

Fernanda A. Melo

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

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

Prof. Dr. Geraldo Pereira Rocha Filho
UESB

Prof. Dr. Vinicius Ruela Pereira Borges
CIC/UnB

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

do Programa de Pós-graduação em Informática

Brasília, 30 de agosto de 2023

Dedicatória

Dedico este trabalho a todas as pessoas que tornaram possível a realização deste sonho. A Deus, meus pais, meu esposo, amigos, familiares, professores e à Universidade de Brasília. Seu amor, apoio e inspiração foram fundamentais para esta conquista.

Agradecimentos

Quero expressar meus agradecimentos a todas as pessoas que me apoiaram ao longo dessa jornada. Primeiramente, agradeço à Deus, cuja presença constante em minha vida me deu forças para enfrentar os desafios e seguir em frente com fé e determinação.

À minha família, em especial aos meus queridos pais, Suely e Edilson, dedico minha mais profunda gratidão. Ao longo dos anos, vocês abriram mão de seus próprios sonhos para assegurar que os meus fossem concretizados. Cada conquista alcançada nesta jornada é também uma vitória de vocês, pois sem o seu amor e dedicação, eu não estaria aqui hoje.

Ao meu amado esposo, André, expresso minha gratidão por ser o meu companheiro de vida. Seu apoio, paciência e compreensão foram essenciais para que eu pudesse enfrentar as exigências deste percurso acadêmico. Você esteve ao meu lado em cada desafio, enxugando minhas lágrimas nos momentos difíceis e compartilhando alegrias nas conquistas. Nosso amor foi a base sólida que me sustentou em meio a tempestades, e sou eternamente grata por ter você ao meu lado.

Por fim, gostaria de expressar minha gratidão à Universidade de Brasília e ao meu estimado orientador, Luis Paulo Faina Garcia. A instituição abriu as portas do conhecimento e proporcionou-me um ambiente enriquecedor para o crescimento acadêmico e pessoal. Agradeço ao Luis por sua orientação dedicada, incentivo constante e compreensão ao longo deste trabalho.

A todos que, de alguma forma, fizeram parte desta jornada, acreditem que seus gestos e palavras foram significativos e moldaram a pessoa que sou hoje. Sem vocês, nada disso seria possível. A gratidão que sinto é imensa e permanecerá em meu coração para sempre.

Resumo

O avanço tecnológico na geração e transmissão de dados impulsionou o surgimento de diversas aplicações de fluxos de dados. Esses ambientes altamente dinâmicos frequentemente enfrentam o desafio da mudança de conceito, em que as propriedades estatísticas das variáveis se alteram ao longo do tempo, resultando na perda de desempenho dos modelos de Aprendizado de Máquina. Neste estudo, é apresentada uma nova ferramenta para detecção de mudança de conceito em problemas de fluxo de dados com grande atraso na chegada da variável alvo, por meio do uso de meta aprendizado. Optou-se por empregar o meta aprendizado devido à sua capacidade de lidar adequadamente com ambientes de fluxo de dados e adaptabilidade em cenários com mudanças de conceito, dado que esta técnica é capaz de identificar padrões e tendências nos dados de forma dinâmica. No entanto, em oposição à abordagem tradicional de recomendação de algoritmos, utilizou-se um regressor no nível meta para prever o desempenho do modelo base em cada janela. Essa predição é então utilizada para gerar alertas de mudança de conceito antes da chegada da variável alvo. O treinamento do meta modelo foi realizado com base em diversos meta atributos não supervisionados descritos na literatura de meta aprendizado. Além disso, foram adicionadas medidas não supervisionadas de detecção de mudança de conceito como parte dos atributos, com o objetivo de aumentar a capacidade preditiva do modelo gerado. O algoritmo foi aplicado em bases de dados com mudança de conceito rotulada, e os alertas gerados foram comparados com métricas usuais da literatura de mudança de conceito. Ao final, foi realizado um teste de *Friedman* para comparação os resultados. O algoritmo proposto de meta aprendizado obteve uma precisão 15% superior à melhor *baseline*, com um p valor abaixo de 0.05 para os casos em que foram utilizados modelos base baseados em árvores.

Palavras-chave: mudança de conceito, meta aprendizado, monitoramento de modelos, aprendizado de máquina

Abstract

Advances in the data generation and transmission have enhanced the existence of many data flows applications. These highly dynamic environments often face the challenge of concept drift, where the statistical properties of variables change over time, leading to performance degradation in Machine Learning models. This study presents a novel tool for concept drift detection in data stream problems with significant delay in the arrival of the target variable, using meta learning. Meta learning was chosen due to its ability to handle data stream environments effectively and its adaptability in scenarios with concept drift, given that this technique can identify patterns and trends in data dynamically. However, contrary to the traditional approach of algorithm recommendation, a meta level regressor was employed to predict the performance of the base model in each time window. This prediction is then utilized to generate concept drift alerts before the arrival of the target variable. The meta model was trained using various unsupervised meta attributes from the meta learning literature. Additionally, unsupervised concept drift detection measures were incorporated as part of the attributes to enhance the predictive capability of the generated model. The algorithm was applied to labeled concept drift datasets, and the generated alerts were compared with standard concept drift literature metrics. Finally, a Friedman test was conducted to compare the results. The proposed meta learning algorithm achieved a 15% higher precision than the best baseline, with a p-value below 0.05 for cases where base models based on trees were employed.

Keywords: concept drift, meta learning, model monitoring, machine learning

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AI Artificial Intelligence.

ANN Artificial Neural Networks.

DM Data Mining.

DT Decision Tree.

IoT Internet of Things.

IQR Interquartile Range.

KNN K-Nearest Neighbors.

KS Kolmogorov Smirnov.

LR Logistic Regression.

MD3 Margin Density Drift Detection.

MFe Meta Features.

ML Machine Learning.

MPAI Multivariate Predictive Accuracy Index.

MSE Mean Squared Error.

MtL Meta Learning.

OmvPht Online Modified Version of the Page–Hinkley Test.

PCA Principal Component Analysis.

PSI Population Stability Index.

RF Random Forest.

SqsiIs Stream Quantification by Score Inspection with Instance Selection.

SVM Support Vector Machine.

UDetect Unsupervised Change Detection for Activity Recognition.

USP Universidade de São Paulo.

Capítulo 1

Introdução

O volume de dados gerado nos últimos anos é maior do que todo o resto da história da humanidade e a cada ano essa quantidade massiva de informações aumenta consideravelmente [1]. Os campos de estudo de mineração de dados, ou *data mining* (DM) [2, 3], e aprendizado de máquina, ou *machine learning* (ML) [4, 5, 6], surgiram como uma alternativa para transformar o conjunto de dados disponível nos data centers ao redor do mundo em informação útil e aplicada a diversos problemas do dia a dia.

Diferentes setores da sociedade se beneficiam atualmente dos avanços de ML. Existe uma infinidade de aplicações que impulsionam os mais diversos segmentos, como a indústria de varejo [7, 8], com modelos de logística de armazenamento e distribuição de mercadorias ou a área de entretenimento e marketing, com a recomendação de filmes [9], músicas [10] e demais produtos [11]. Além da otimização de processos no âmbito empresarial, esta linha de pesquisa acarreta em melhoria em áreas fundamentais da sociedade, como no ramo de saúde [12], onde modelos preditivos são capazes de fazer o diagnóstico precoce de doenças em exames de imagem [13] e laboratoriais [14], ou no segmento de segurança, com predição de fraudes [15, 16] ou identificação de crimes via reconhecimento de imagens [17].

O avanço tecnológico na geração, transmissão e armazenamento de dados potencializou a existência de muitas aplicações que consistem na geração massiva e contínua de informações, como o monitoramento de redes [18], aplicações no mercado financeiro [19] e os dados dos sensores coletados por sistemas de Internet das Coisas, ou *Internet of Things* (IoT) [20]. Nestes cenários, as informações são modeladas como fluxos de dados transientes, os *data streams*, e geralmente estão associados a ambientes altamente dinâmicos e não estacionários [21]. A aplicação de ML em fluxo de dados é um campo de estudos desafiador visto que novas dificuldades são incluídas pela natureza das aplicações de fluxo [22], como treinamento em ambientes distribuídos [23] ou limitações de tempo de processamento e memória disponível [24].

A dinamicidade do ambiente de fluxo de dados acarreta em outro problema à medida em que há alteração das propriedades estatísticas da variável alvo ao longo do tempo. Na prática, isso faz com que o modelo preditivo fique defasado e reduza seu desempenho. Este é um fenômeno denominado mudança de conceito, ou *concept drift* [25, 26], cuja detecção é um problema desafiador e seu estudo tem se tornado um tópico de pesquisa recente com número crescente de publicações [27, 28, 29, 30, 31, 32, 33, 34]; crescimento que também pode ser observado fora do meio acadêmico, onde as empresas dos mais diversos segmentos começam a destinar recursos para o monitoramento dos modelos implantados com o propósito de evitar prejuízos, financeiros ou não, decorrentes do equívoco na tomada de decisão [35].

O teorema sem almoço grátis, ou *No free lunch theorem* [36], afirma que conceitos de dados diferentes podem se adequar melhor a modelos com viéses diferentes, deste modo, a seleção de algoritmos surge como uma solução adequada para lidar com o problema da mudança de conceito. Esta linha é abordada pelo campo de estudos de meta aprendizado, ou *meta learning* (MtL) [37, 38, 39, 40, 41], uma técnica avançada que aprende o padrão das mudanças existentes nos dados de forma sistêmica para recomendar o modelo mais adequado para cada situação.

O MtL se resume em tomar um conjunto de bases de dados, utilizá-las no treinamento de diversos algoritmos de ML e, em seguida, utilizar um conjunto de métricas de desempenho para avaliar estes modelos. Após isso, diversas medidas de caracterização das bases de dados, denominadas meta atributos, ou *meta features* (MFe), são calculadas e utilizadas para compor a meta base, o novo conjunto de dados que sumariza o comportamento das bases originais e os associa com o desempenho obtido pelos modelos. Por fim, a meta base é utilizada no treinamento do meta modelo que é capaz de, então, recomendar o algoritmo mais adequado para cada problema. Em [42], os autores propõem o *metaStream*, uma adaptação do MtL para problemas de fluxo de dados, onde as MFe são extraídas de janelas temporais e o meta modelo é treinado com os meta dados de uma única base de dados, o fluxo em questão [42, 43, 44, 45].

1.1 Justificativa

Apesar dos múltiplos benefícios sociais da evolução de ML [46, 47] é preciso ser cauteloso quanto ao monitoramento dos modelos preditivos que tendem a perder eficácia com o tempo devido à mudança de conceito à medida em que a dinâmica do mundo se altera [48]. Quando se trata de problemas em fluxos de dados, a redução no desempenho dos modelos decorrente da mudança de conceito se torna ainda mais provável [49], por isso, é crucial

entender as consequências dessa alteração e implementar um método que possibilite, por exemplo, a detecção do momento correto de retreino ou substituição do modelo [50].

Atualmente, existem diversas técnicas de detecção com diferentes níveis de complexidade, desde medidas estatísticas de fácil implementação como o Índice de Estabilidade Populacional, ou *Population Stability Index* (PSI) [29] até medidas complexas que fazem uso de modelos de ML como a Detecção de Desvio de Densidade de Margem, ou *Margin Density Drift Detection* (MD3) [33]. O número crescente de publicações realizadas na área de detecção de mudança de conceito é um reflexo direto da relevância e atualidade do tema [51].

Alguns problemas enfrentam um novo desafio decorrente do atraso na chegada da variável alvo, situação que ocorre, por exemplo, nos modelos de detecção de fraude cuja variável alvo é indeterminada até o momento em que o dono legítimo do cartão percebe a compra indevida e abre uma reclamação com o banco emissor [52]. Com a ausência da variável alvo, o problema de medir o desempenho dos modelos comparando a predição com o valor esperado se torna muito mais desafiador e, no geral, esperar esse tempo para iniciar a avaliação de desempenho é inviável, dado o prejuízo decorrente das predições incorretas de um modelo preditivo. Apesar disso, a maior parte dos trabalhos existentes na literatura se concentram na solução de problemas supervisionados, onde o valor da variável alvo é descoberto simultaneamente ou com pouco atraso com relação aos atributos de entrada, conforme constatado por [27].

A seleção de algoritmos surge como estratégia promissora para a troca do modelo preditivo de nível base em casos de mudança de conceito para evitar a perda de desempenho do sistema. No entanto, ela é de fato adequada a problemas em que o desempenho dos algoritmos são similares em grandes períodos de tempo mas muito diferentes quando analisados em janelas temporais pequenas [42]. Considerando que essa característica não se aplica universalmente a todas as bases de dados, é possível que, em alguns casos, o uso do MtL não apresente um ganho significativo em relação às abordagens tradicionais de ML. Considerando que a aplicação de MtL implica em custos computacionais elevados, é necessário avaliar sua viabilidade e aplicabilidade nestes casos.

Conforme constatado em [53], diferentes pesquisadores apresentam definições distintas acerca do termo, de modo que as aplicações de MtL não se restringe à recomendação de modelos, apesar de ser utilizado majoritariamente para este fim. Em [53], por exemplo, os autores abordam o uso de MtL na construção de algoritmos de ML autoadaptáveis que evoluem dinamicamente por meio do acúmulo de meta conhecimento. Em [54] o autor relata outros usos como otimização de arquitetura em redes neurais artificiais, ou *Artificial Neural Networks* ANN, ou seleção de parâmetros em Máquinas de Vetores de Suporte, ou *Support Vector Machine* SVM [55].

Esta pesquisa apresenta uma proposta inovadora ao investigar o uso da arquitetura de MtL para realizar a detecção de mudança de conceito, em contraponto à abordagem tradicional de seleção de algoritmos. Essa é uma contribuição relevante, uma vez que, até o momento desta pesquisa, não encontramos outra pesquisa que tenha explorado o MtL para esse propósito específico, tornando este estudo pioneiro nesse campo. A motivação para esta abordagem é que, ao aplicar o MtL na seleção de algoritmos, é possível que o custo computacional não seja condizente com o ganho de desempenho sutil obtido em algumas bases. Por outro lado, a detecção precisa de mudança de conceito proporciona visibilidade sobre a defasagem do modelo, permitindo a aplicação de outras técnicas para a melhoria de desempenho e a consequente alocação de recursos para estratégias mais eficazes.

1.2 Objetivos

Neste estudo, busca-se validar a capacidade de um meta regressor em prever o desempenho de modelos de nível base com diferentes vieses e, com isso, desenvolver uma ferramenta de detecção de mudança de conceito baseada nessa predição de desempenho. Verificar se essa ferramenta é mais eficaz do que as medidas tradicionais de detecção de mudança de conceito encontradas na literatura constitui o principal objetivo da pesquisa.

A premissa é que a utilização do MtL como ferramenta para detecção de mudança de conceito resultará em resultados superiores às técnicas atualmente disponíveis. Dessa forma, espera-se que os alertas gerados forneçam uma forma confiável de aumentar a visibilidade do desempenho atual antes da ocorrência de mudanças significativas. permitindo ajustes adequados para aprimorar o desempenho com abordagens tradicionais de ML, como a criação de novos atributos no nível base, otimização de parâmetros, ajuste dos limiares de decisão, retreino do modelo, entre outros.

Com base nesse contexto, foram propostas algumas alterações na arquitetura original do MtL, incluindo a utilização de um único modelo no nível base a ser monitorado, em contraste com a abordagem tradicional que envolve múltiplos modelos. Além disso, foram considerados atrasos significativos na chegada da variável alvo, cenário que descreve situações mais realista, resultando no uso exclusivo de MFe não supervisionadas.

Um novo conjunto de MFe será proposto, incorporando métricas não supervisionadas de detecção de mudança de conceito. Acredita-se que esses atributos adicionais forneçam informações relevantes para a abordagem do problema em questão. Além disso, a saída de probabilidade do modelo base também foi utilizada como MFe como objetivo proporcionar informações relevantes para a predição de desempenho e aumentar o poder preditivo do meta regressor gerado.

Dito isto, o principal objetivo desta pesquisa é validar se o meta regressor é capaz de prever o desempenho dos modelos de nível base com diferentes vieses, gerando alertas de mudança de conceito confiáveis. Adicionalmente, busca-se determinar se esses alertas baseados no MtL proposto são superiores às medidas de detecção de mudança de conceito existentes na literatura. Para validar os resultados obtidos, serão realizadas comparações com outras métricas em bases de dados tradicionais de problemas de fluxo de dados e bases sintéticas que possuam mudança de conceito rotulada.

1.3 Hipóteses

Como hipótese principal, este trabalho procura validar se a aplicação do algoritmo de MtL em fluxo de dados é capaz de recomendar o desempenho dos algoritmos e gerar alertas confiáveis a partir da detecção da mudança de conceito da base. Também tenta-se validar as seguintes hipóteses secundárias:

1. O meta regressor é capaz de prever o desempenho do modelo base quando existe atraso grande na chegada da variável alvo.
2. O uso de MFe calculados a partir da predição do modelo base e das métricas de detecção de mudança de conceito melhoram o desempenho do meta modelo.
3. O MtL é adequado como nova métrica de detecção de mudança de conceito.

1.4 Organização do Documento

O restante deste documento será dividido da seguinte forma: os capítulos iniciais apresentam a revisão bibliográfica sobre os temas relevantes para a pesquisa. No capítulo 2 são abordados os conceitos de como fluxo de dados e mudança de conceito e no Capítulo 3 é apresentada a teoria de MtL e uma revisão bibliográfica dos estudos recentes e relevantes da área. O Capítulo 4 introduz a metodologia utilizada nesta pesquisa, detalhando e justificando as escolhas de projeto e os detalhes de implementação. O Capítulo 5 aborda os resultados obtidos e o Capítulo 6 apresenta as conclusões obtidas e detalha as limitações da pesquisa e possíveis trabalhos futuros.

Capítulo 2

Fluxo de dados

O avanço da tecnologia de geração, armazenamento e processamento de informações potencializou a existência de muitas aplicações que consistem na geração massiva e contínua de dados, alguns exemplos são o processamento de dados de redes sociais [56], de redes de sensores [57] ou de aplicações financeiras [19]. Nestes cenários as informações são modeladas como fluxos de dados transientes, em oposição às aplicações tradicionais de ML em que o volume de dados é reduzido e persistido em tabelas [58] com estratégias de aprendizado em lotes, ou *batches*, onde toda a base de treino está disponível para o algoritmo [59].

O fluxo de dados pode ser definido como o movimento contínuo de dados ou informações entre os elementos de um sistema, com o objetivo de fornecer, processar ou armazenar essas informações de maneira eficiente. Em [60], os autores definem fluxo de dados como um fluxo contínuo de informações que pode ser potencialmente infinito e que deve ser processado de forma incremental e online.

Dito isto, o ambiente altamente dinâmico de fluxo de dados introduz uma série de novos desafios para o aprendizado e tem se tornado um tópico de pesquisa recente [22], a alta taxa de geração de dados acarreta em diversas alterações como a forma de acesso aos dados, tempo e recursos disponíveis para o processamento. Em [23], os autores relatam que os fluxos de dados geralmente estão associados a ambientes distribuídos, que por si só adicionam novos desafios ao problema, como a redução de latência de comunicação entre os nós e o fato de a maior parte da literatura de ML estar atualmente focada em sistemas não distribuídos.

Como os dados são gerados continuamente, o tamanho do fluxo é potencialmente infinito para a maior parte dos casos, portanto, as limitações existentes de tempo de processamento e memória disponíveis para armazenamento acarretam no uso de uma porção pequena e mais recente do fluxo, descartando o restante dos dados, de modo que resultados aproximados são aceitáveis [22]. O tempo de processamento também é limitado

devido à natureza de resposta em tempo real das aplicações de fluxos de dados. A Tabela 2.1, detalha algumas destas diferenças entre o aprendizado em bases tradicionais e fluxo de dados.

Característica	Bases tradicionais	Fluxo de dados
Acesso aos dados	Aleatório	Sequencial
Tempo de processamento	Ilimitado	Restrito
Memória disponível	Ilimitada	Fixa
Resultado	Preciso	Aproximado

Tabela 2.1: Diferenças entre as bases de dados tradicionais e o fluxo de dados. Tabela adaptada de Gama et al. (2012)

Em relação ao acesso aos dados, as bases tradicionais permitem acesso aleatório aos registros, o que significa que é possível recuperar informações de forma direta e independente da ordem em que foram armazenadas. Por outro lado, no fluxo de dados os dados são analisados em janelas temporais, isso se deve não só à limitação de recursos temporais e computacionais mas também ao fato de que as informações recentes geralmente são mais relevantes do que os dados históricos [59]. Em [61], os autores descrevem dois tipos de janelas deslizantes usuais para problemas de fluxo de dados, a primeira se refere às janelas cujo tamanho é determinado por meio de intervalos de tempo de duração fixa, a segunda se refere às janelas cujo tamanho é determinado por um número fixo de instâncias do fluxo de dados. Devido à simplicidade de implementação e generalização para diferentes bases, este trabalho abordará o uso das janelas deslizantes baseadas no número de instâncias.

Quanto ao tempo de processamento, as bases tradicionais possuem um tempo ilimitado para execução de consultas e operações, ao passo que o fluxo de dados possui um tempo de processamento restrito, pois precisa lidar com informações em tempo real, em uma sequência contínua e dinâmica. A necessidade de resposta em tempo real somado ao alto volume de captação dos dados e ao custo computacional proveniente do processamento e uso de ML faz com que técnicas de amostragem de dados sejam necessárias em grande parte das aplicações de fluxo de dados. Desse modo, apenas algumas instâncias são selecionadas para análise e o volume total de informações é reduzido, bem como o custo computacional necessário.

Em [59] o autor constata que a amostragem em fluxo de dados enfrenta o desafio de encontrar amostras que sejam de fato representativas do fluxo original, visto que a maior parte das técnicas de amostragem exigem o conhecimento do tamanho total da população, que é desconhecido no problema em questão. Nesta pesquisa, as bases de dados utilizadas foram extraídas de fluxos de dados reais durante intervalos temporais fixados, possuindo

portanto tamanho limitado, de modo que as técnicas de amostragem não se mostraram necessárias para os experimentos realizados.

No que tange à disponibilidade de memória, os ambientes que abrigam as bases de dados tradicionais geralmente são projetados para acomodar um alto volume de informações, possibilitando o processamento eficiente de consultas e operações complexas. Em contrapartida, o fluxo de dados opera em um contexto onde a memória disponível é fixa e limitada, o que implica que apenas uma quantidade restrita de dados pode ser mantida durante o processamento contínuo. Esse cenário reforça a importância do uso de estratégias como janelas temporais e técnicas de amostragem, que permitem controlar a quantidade de dados retidos em memória em um determinado período.

O uso contínuo de modelos de ML pré treinados parte do pressuposto que as instâncias são geradas aleatoriamente de acordo com uma distribuição estacionária, isso não é verdade para a maioria dos problemas de fluxo de dados que, por natureza, possuem longas sequências de dados cuja distribuição se defasa com o tempo. Este problema, denominado mudança de conceito, consiste na alteração periódica do significado dos dados no problema abordado.

2.1 Mudança de conceito

Problemas do mundo real enfrentam o desafio de que nem sempre o conceito da variável alvo pode ser descrito integralmente pelas variáveis explicativas disponíveis para o modelo, conforme constatado por [26], a alteração destes atributos não acessíveis acarretam no fenômeno de mudança de conceito. Isto porque os fatores externos ao sistema modelado alteram a relação existente entre os atributos de entrada e saída do modelo, tornando-o inconsistente com o comportamento atual dos dados.

Em [21] os autores atestam o aumento do erro dos algoritmos quando há uma alteração na distribuição de probabilidade das instâncias. A mudança de conceito entre dois pontos distintos no tempo t_0 e t_1 pode ser definida formalmente como a alteração na função que mapeia os atributos de entrada na variável alvo, conforme a Equação 2.1, onde X é o conjunto de valores de atributos de entrada e y é o valor da variável alvo do problema.

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{2.1}$$

Em [24] os autores definem dois tipos de alteração nos dados, a primeira segue a definição da Equação 2.1, denominada mudança de conceito real, e se refere ao problema que esta pesquisa se propõe a resolver. O segundo tipo, denominado mudança de conceito virtual, que também pode ser definido como defasagem de dados ou *feature drift*, se refere à situação em que a função que mapeia os dados na variável alvo se mantém ainda que

haja alteração na distribuição dos atributos de entrada, não implicando, portanto, na perda de desempenho do modelo preditivo. A diferença entre as mudanças de conceito real e virtual pode ser observada na Figura 2.1.

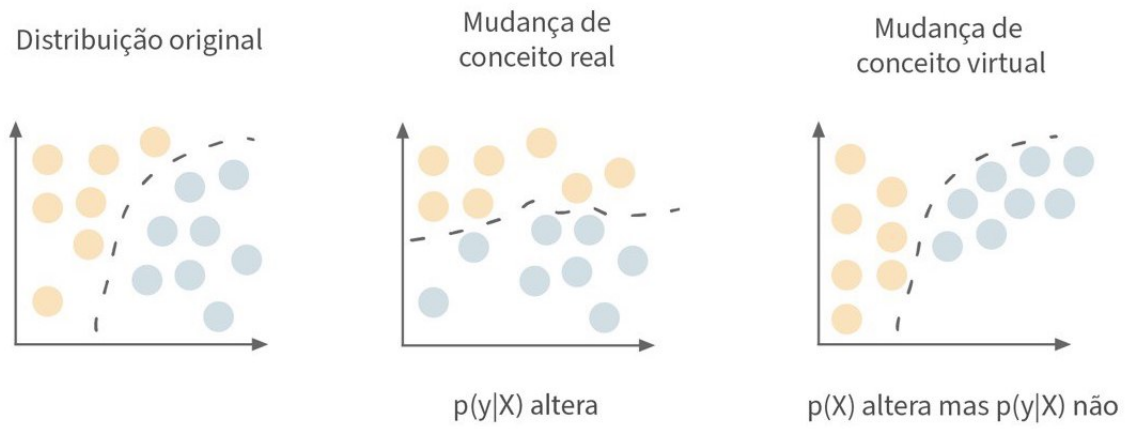


Figura 2.1: Ilustração de mudança de conceito real e virtual. Figura adaptada de Gama et al. (2014)

Na figura, a distribuição dos dados é representada pelos círculos e as cores distinguem duas classes. A linha tracejada ilustra a fronteira de decisão de um modelo de classificação fictício.

No primeiro gráfico, o modelo é capaz de separar as duas classes de dados corretamente. No segundo gráfico, uma mudança de conceito real é ilustrada, a alteração de posição e cor dos círculos representam a mudança nas características dos dados ao longo do tempo. Por consequência, a fronteira de decisão anterior não é capaz de separar corretamente as classes de dados, de modo que o modelo precisa ser adaptado ou reajustado para se adequar às novas características dos dados e manter um desempenho adequado.

Por fim, o terceiro gráfico demonstra a mudança de conceito virtual, na qual os círculos mudam de posição mas a fronteira de decisão permanece inalterada. Nesse cenário, mesmo com as mudanças na distribuição dos atributos de entrada, a fronteira de decisão original é capaz de separar corretamente as classes de dados, e o modelo mantém o desempenho preciso sem a necessidade de ajustes. O foco deste trabalho é abordar o desafiador problema da mudança de conceito real em modelos de ML.

Em [62], os autores definem seis tipos de mudança de conceito para modelos de classificação, ilustrados na Figura 2.2, onde os eixos x e y, juntamente com as cores, retratam dois conceitos fictícios ao longo do tempo. É possível observar como os conceitos evoluem e se alteram, possibilitando identificar padrões como mudanças abruptas, incrementais,

graduais, recorrentes, blips ou ruído, de acordo com a natureza dos dados e o problema específico.

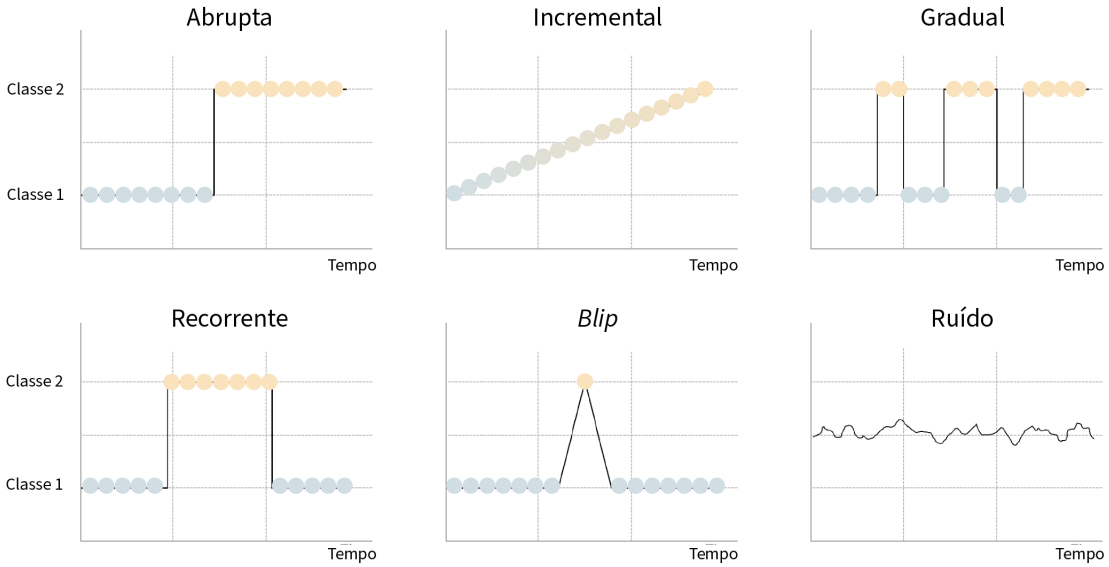


Figura 2.2: Tipos de mudança de conceito. Figura adaptada de Iwashita et al. (2019)

A mudança abrupta ocorre de forma súbita e irreversível, levando à extinção definitiva do conceito anterior. Um exemplo seria a troca de sensores em um sistema IoT, resultando em uma mudança abrupta na captura de informações. Por outro lado, a mudança incremental é mais lenta, ocorrendo em pequenos passos sequenciais e dificultando a detecção imediata, como o uso contínuo de um sensor defeituoso que gradualmente perde precisão.

A mudança gradual apresenta uma fase de transição, onde as instâncias são geradas por uma mistura do conceito atual e do próximo conceito, com a proporção variando gradativamente ao longo do tempo [63]. As mudanças de conceito recorrentes são aquelas em que um conceito pré-existente no passado pode reaparecer no futuro, como o caso das estações do ano. Já a mudança *blip* pode ser considerada um *outlier* na distribuição e pode ser ignorada, devido à sua natureza aleatória e à falta de informações significativas para o modelo [64]. Por fim, o ruído representa mudanças aleatórias nos exemplos de dados. De acordo com [62], um classificador eficiente deve aprender de forma incremental e se adaptar a esse tipo de mudança.

2.2 Detecção de mudança de conceito

Os algoritmos de ML aprendem com dados passados para fazer previsões sobre o futuro, portanto, a mudança de conceito é um grande obstáculo para problemas de fluxo de dados

devido à dificuldade em manter um modelo preciso em ambientes não estacionários. Uma abordagem comum para este problema é o uso de aprendizado incremental [22], por meio do treinamento contínuo de modelos de ML que permitam adaptações incrementais nos seus parâmetros internos.

Alguns destes sistemas consistem na adaptação contínua dos modelos em intervalos regulares, no entanto, existem abordagens mais interessantes que consistem no monitoramento da mudança de conceito com a finalidade de detectar o momento ideal de retreino ou ajuste do algoritmo. Em [59] o autor cita duas diferentes abordagens de monitoramento de fluxo de dados, a primeira envolve o monitoramento de indicadores de desempenho associado ao controle estatístico de processos e a segunda envolve a comparação das distribuições em pontos distintos do tempo, nesta pesquisa ambas abordagens serão utilizadas.

O número crescente de publicações realizadas na área de detecção de mudança de conceito é um reflexo direto da relevância e atualidade do tema, no entanto, a maior parte destes trabalhos se concentraram na solução de problemas supervisionados, onde o valor real da variável alvo é descoberto simultaneamente ou com pouco atraso com relação à chegada das variáveis de entrada, conforme constatado por [30]. Apesar disso, grande parte dos problemas do mundo real apresentam atraso na chegada da variável alvo, motivo pelo qual este trabalho se dedica ao estudo de detecção da mudança de conceito de forma não supervisionada. Dentre os diversos algoritmos não supervisionados propostos, esta seção apresenta algumas medidas clássicas da literatura, além de alguns dos algoritmos levantados pelo estudo de [30], que apresenta uma visão global dos principais métodos não supervisionados existentes na literatura.

O PSI, é uma técnica univariada comumente utilizada para detectar mudanças de conceito [29]. A sua ampla adoção pode ser atribuída à sua facilidade de implementação e, sobretudo, interpretação, uma vez que conta com limiares de referência bem estabelecidos, propostos por [65], conforme descrito na Tabela 2.2.

O PSI é responsável por medir a similaridade entre duas distribuições e é calculado individualmente para cada atributo da base. Essa abordagem torna a métrica propensa à falsos positivos decorrentes da detecção de mudança de conceito virtual dado que a alteração da distribuição dos atributos de entrada não implicam necessariamente em mudança de conceito real, apesar de ser um forte indício. Neste trabalho será descrita uma adaptação do PSI original para problemas de fluxo de dados.

O seu cálculo é feito a partir de uma base de dados de referência que garantidamente não possua mudança de conceito, como a base utilizada no treinamento do modelo. Inicialmente, cada atributo desta base é dividido em n grupos de mesmo tamanho, denominados quantis, conforme ilustrado na Figura 2.3 com coloração alaranjada. Quando um novo

Intervalo	Interpretação
$PSI \leq 0.1$	Sem mudanças significativas na distribuição
$0.1 < PSI < 0.25$	Alteração que requer investigação adicional
$PSI \geq 0.25$	Existem alterações significativas

Tabela 2.2: Interpretação do valor do PSI

conjunto de dados se acumula, cada atributo da janela é dividido em n grupos seguindo os intervalos dos dados de referência salvos na etapa anterior. A porcentagem dos dados em cada quantil é então calculada, conforme ilustrado em verde na Figura 2.3. Os quantis de referência e monitoramento são por fim utilizados no cálculo do PSI seguindo a equação 2.2.

$$PSI = \sum((\%monitoramento - \%referencia) * ln(\frac{\%monitoramento}{\%referencia})) \tag{2.2}$$

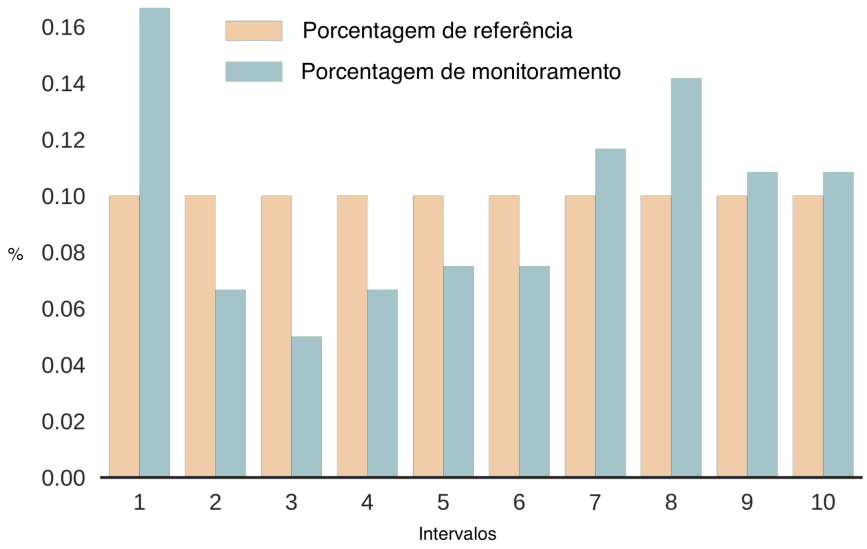


Figura 2.3: Comparação dos quantis de referência e monitoramento para cálculo do PSI

Em [28] os autores listam uma série de desvantagens para o uso do PSI, como o valor da métrica ser muito influenciado pela escolha do número de quantis, a falta de confiabilidade do método quando as frequências de uma categoria se aproximam de zero, o baixo impacto de valores discrepantes extremos e o fato de categorizar uma variável numérica implicar em não capturar mudanças na estabilidade adequadamente. Para resolver estes problemas, eles propõem uma nova métrica exclusiva para problemas de regressão batizada de Índice de Acurácia Preditiva Multivariada, ou *Multivariate Predictive Accuracy Index* (MPAI), descrita na equação 2.3.

$$MPAI = \frac{\sum_{j=1}^{N} r_j^T V r_j / N}{\sum_{i=1}^{n} x_i^T V x_i / n} \tag{2.3}$$

Onde r_j é o vetor de variáveis explicativas para a j-ésima observação dos dados de monitoramento com $1 \leq j \leq N$ e x_i é o vetor de variáveis explicativas para a i-ésima observação dos dados de referência com $1 \leq i \leq n$. E $V = MSE\left(X^T X\right)^{-1}$ é a matriz de variância-covariância dos coeficientes de regressão estimados $(\beta 1, \beta 2, ..., \beta p)$.

Segundo os autores, o MPAI é definido como a variância média do valor estimado para dados de monitoramento dividida pela variância média estimada nos dados de referência, portanto o resultado da métrica pode ser interpretado de forma direta, ao contrário do PSI, que é definido em uma escala sem interpretação óbvia. Um valor MPAI de 2, por exemplo, implica que a variância das predições de monitoramento é o dobro da variância da resposta média nos dados de referência. Os valores de MPAI podem ser interpretados conforme descrito na tabela 2.3.

Intervalo	Interpretação
$MPAI < 1.1$	Não existem mudanças significativas
$1.1 \leq MPAI < 1.5$	Alteração que requer investigação adicional
$MPAI \geq 1.5$	Existem alterações significativas

Tabela 2.3: Interpretação do valor do MPAI

Para além das limitações destacadas por [28], é importante ressaltar que o uso de técnicas univariadas como o PSI dificulta o monitoramento de modelos que possuam muitos atributos de entrada, de modo que esta métrica se mostra inadequada para ser utilizada de forma exclusiva no monitoramento de mudança de conceito. A fim de abordar algumas das questões mencionadas, uma alternativa é o *Domain Classifier*, uma técnica proposta por pesquisadores da *Dataiku*, plataforma líder mundial em AI [66], que lida com o problema de mudança de conceito de forma multivariada.

O algoritmo consiste na construção de um classificador binário que deve diferenciar amostras de treinamento, dados de referência sem mudança de conceito, e monitoramento, dados da última janela que podem ter mudança de conceito. Se a distribuição dos dados for muito similar, o classificador terá dificuldades para diferenciar as amostras e apresentará acurácia próxima de 0.5, à medida em que a defasagem de dados aumenta temos um aumento ou diminuição da acurácia. Ao final, um teste binomial é realizado e o p valor é utilizado como medida de defasagem.

Uma abordagem interessante é a utilização da predição do próprio modelo como critério de monitoramento. Essa estratégia permite verificar a consistência das previsões do

modelo em relação aos atributos de entrada, reduzindo a probabilidade de falso positivos durante o processo de detecção de mudança de conceito.

Neste aspecto, uma técnica interessante para problemas de classificação foi proposta por [31], denominada Detecção de Alteração não Supervisionada para Reconhecimento de Atividade, ou *Unsupervised Change Detection for Activity Recognition* (UDetect). Ela é baseada na premissa de que a alta variância das instâncias de determinada classe em relação às instâncias de treinamento dessa mesma classe é um indicativo de mudança de conceito.

O cálculo do UDetect tem início com o treinamento do classificador base empregando um conjunto de dados previamente rotulado. Esse modelo é o mesmo utilizado na predição da variável alvo, visando resolver um problema no nível base. À medida que novos dados são acumulados, eles são agrupados em conjuntos de tamanho fixo, contendo apenas instâncias da mesma classe, conforme classificado pelo modelo base treinado na etapa anterior. Para cada classe, a distância média até o centróide da classe é computada, seguindo a equação 2.4. Por fim, as distâncias de referência e de monitoramento são comparadas, com o propósito de determinar a ocorrência de mudança de conceito.

$$dist = \frac{1}{n} \sum_{1}^{n} (y_i - \widehat{y})^2 \tag{2.4}$$

Onde y_i é o valor da instância i e $\widehat{y}$ é o valor do centróide, calculado pela média dos valores de todas as instâncias da mesma classe.

Outra técnica interessante é a Quantificação de Fluxo por Inspeção de *Score* com Seleção de Instância, ou *Stream Quantification by Score Inspection with Instance Selection* (SqsiIs), é uma métrica proposta por [32] que consiste na detecção de mudança de conceito a partir da semelhança das distribuições de saída de referência e monitoramento do modelo base.

A saída dos classificadores consiste na probabilidade de a instância analisada pertencer a determinada classe, esta probabilidade é então convertida em uma predição categórica através da aplicação de um limiar. Portanto, quando se olha para o conjunto de predições em uma janela tem-se uma visão geral do grau de confiança do modelo sobre as suas predições, para um classificador binário. Uma distribuição com alta concentração de instâncias com probabilidade próxima da fronteira de decisão 0.5, por exemplo, indica que o modelo não foi capaz de distinguir com clareza as amostras de cada classe. O SqsiIs propõe que a alteração na distribuição de probabilidade das predições é um indício forte de mudança de conceito real por afetar o grau de confiança do modelo sobre a sua saída.

O cálculo do SqsiIs tem início com o treinamento do classificador base empregando um conjunto de dados previamente rotulado. Esse modelo é o mesmo utilizado na predição da variável alvo, visando resolver um problema no nível base. À medida em que novos

dados se acumulam em um janelas de tamanho fixo, o classificador base é utilizado para fazer a predição e um teste *Kolmogorov Smirnov* (KS) biamostral [67] é aplicado entre as predições de referência e monitoramento. Se a hipótese nula for rejeitada, a transformação linear da equação 2.5 é aplicada nos *scores* de referência para que eles tenham a mesma média e desvio padrão das predições de monitoramento, em seguida o teste KS é aplicado novamente. Se a hipótese nula for rejeitada novamente, o algoritmo assume que houve mudança de conceito nesta janelas de dados.

$$x_{i\ new} = \widehat{x}_{new} + (x_{i\ old} - \widehat{x}_{old})\,\frac{\sigma_{new}}{\sigma_{old}} \tag{2.5}$$

A Detecção de Desvio de Densidade de Margem, ou *Margin Density Drift Detection* (MD3) é outra técnica proposta por [33] para monitorar as mudanças na margem de um classificador, região na qual as predições são mais incertas. A métrica se baseia em utilizar a densidade de instâncias existentes na região de incerteza como uma medida de detecção de mudança de conceito, através da comparação dos dados de referência e monitoramento com um limiar pré definido.

O cálculo do MD3 se inicia com o treinamento do classificador base com SVM, em um conjunto de dados rotulado. Em seguida, as margens históricas mínima e máxima do treinamento do SVM são calculadas conforme a Figura 2.4, elas representam a fronteira de decisão dos dados entre as classes, portanto, as instâncias dentro dessa margem possuem alto grau de incerteza. À medida em que novos dados se acumulam em grupos de tamanho fixo, a quantidade de instâncias dentro da margem é calculada como uma medida da densidade da margem, por fim, a variação desta densidade é utilizada como medida de mudança de conceito de modo que um aumento corresponde a mudança gradual e uma diminuição corresponde a mudança abrupta. A possibilidade de detectar o tipo de mudança de conceito existente certamente é uma das grandes vantagens do MD3.

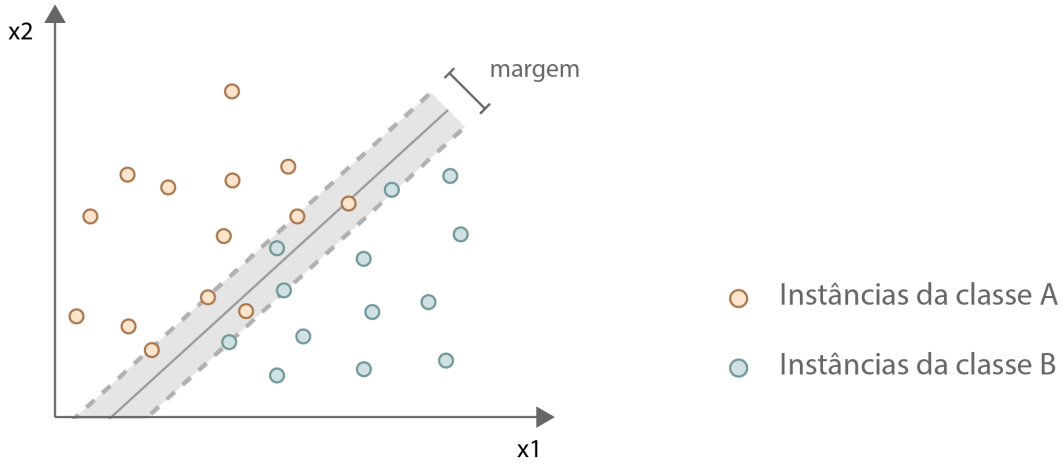


Figura 2.4: Ilustração da margem do algoritmo SVM utilizado no MD3

Conforme constatado por [30], a desvantagem de possibilitar somente o uso de SVM como classificador base fez com que os próprios autores desenvolvessem uma evolução [68] do MD3 dois anos depois. O novo algoritmo foi batizado como *MD3 Ensemble Generic Margin* e consiste em gerar uma margem genérica através do *ensemble* [69] de vários classificadores, a partir do treinamento de múltiplos modelos que decidem em conjunto a melhor predição. Neste caso, o custo computacional se torna um novo contra já que o treinamento de muitos modelos de ML escalaria os recursos necessários para a execução do algoritmo.

Por fim, destaca-se a Versão Online Modificada do Teste Page–Hinkley, ou *Online Modified Version of the Page–Hinkley Test* (OmvPht), [34], técnica utilizada na detecção de mudança de conceito no algoritmo proposto por [70]. Esta métrica consiste em assumir que as distribuições das predições das classes de um modelo de classificação podem ser modeladas como duas curvas gaussianas, conforme ilustrado na Figura 2.5, e que a sobreposição entre elas pode ser utilizada como medida de mudança de conceito.

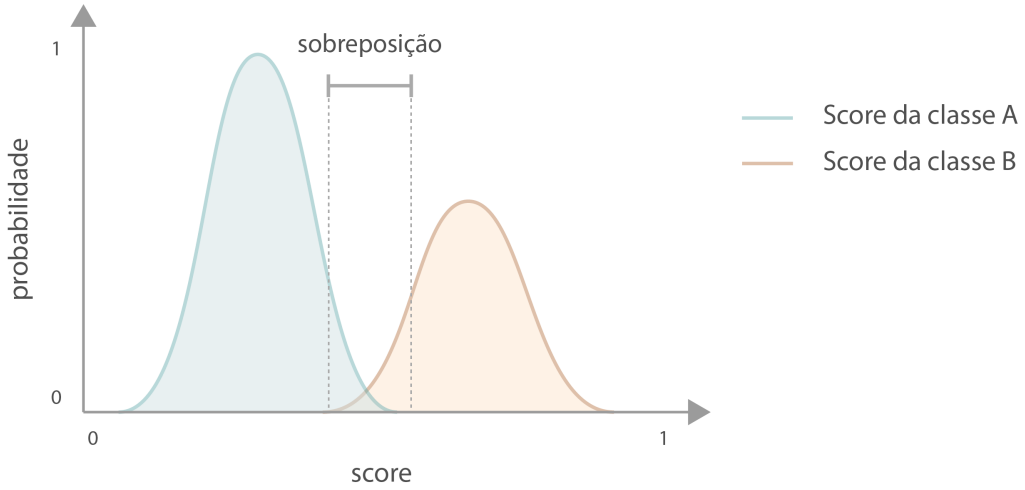


Figura 2.5: Ilustração da modelagem dos scores como duas gaussianas no OmvPht

Capítulo 3

Meta aprendizado para fluxo de dados

O MtL foi inicialmente proposto por Schmidhuber em sua tese de mestrado (1987) [71], se trata de uma técnica avançada que faz uso de ML para aprender o padrão das mudanças existentes nos dados de forma sistêmica para o auto-aperfeiçoamento. Conforme descrito por [39], o MtL se diferencia do aprendizado padrão no âmbito de adaptação, enquanto o ML básico é focado em encontrar padrões de tarefas de uso específico, o MtL é focado no acúmulo de experiência relacionado ao aprendizado e desempenho do modelo base.

Em [72] os autores o definem como a ciência de observar sistematicamente como diferentes abordagens de aprendizado de máquina são executadas em uma ampla gama de tarefas de aprendizado e, em seguida, aprender com essa experiência. O MtL já foi aplicado na literatura para diferentes contextos, como a construção de algoritmos de ML autoadaptáveis [53], seleção de hiperparâmetros ou otimização de arquitetura em redes neurais [54], a abordagem mais comum é o uso na seleção dinâmica e autônoma de algoritmos [73] [72] que será descrito em mais detalhes na próxima seção.

3.1 O problema da seleção de algoritmos

O teorema sem almoço grátis, ou *no free lunch theorem* [36], afirma que todos os algoritmos de otimização funcionam, em média, igualmente bem quando seu desempenho é calculado em todos os problemas possíveis, isto implica na inexistência um único algoritmo de otimização superior aos demais de modo que modelos com vieses diferentes podem se adequar melhor à conjuntos de dados distintos. Portanto, uma forma de otimizar o desempenho geral de um sistema ML é realizar a seleção dinâmica de algoritmos mais adequados a cada janela de dados.

O problema da seleção de algoritmos, descrito por Rice em [74], consiste na escolha de um algoritmo e seus parâmetros de modo que estes satisfaçam os objetivos do problema em questão da melhor forma possível. O autor divide a abordagem em três espaços: problema P, algoritmo A e métricas de desempenho Y, em [75] é adicionada a definição de espaço de atributo F que também será abordada nesta seção. A Figura 3.1 ilustra a relação entre os quatro espaços que pode ser definida da seguinte maneira: Para uma dada instância do problema $x \in P$, com atributos $f(x) \in F$, encontre o mapeamento de seleção $S(f(x))$ no espaço do algoritmo A tal que o algoritmo selecionado $\alpha \in A$ maximize a performance $y(\alpha(x)) \in Y$.

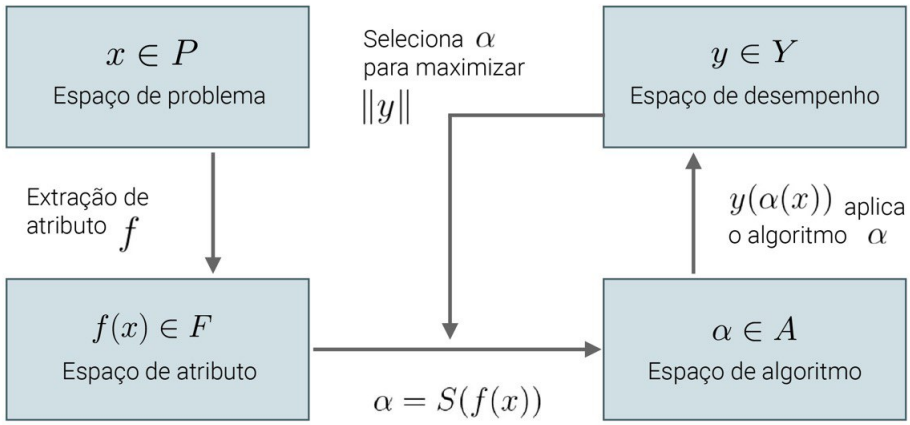


Figura 3.1: Problema de seleção de algoritmos. Figura adaptada de Salisu et al. (2017)

3.1.1 Espaço de problema P

Consiste no conjunto de bases de dados $x \in P$, utilizadas no estudo. Como esta pesquisa aborda o estudo de fluxos de dados, que são por natureza sequências de dados sem tamanho definido, apenas uma base de dados foi utilizada por experimento.

3.1.2 Espaço de algoritmo A

Consiste no conjunto de todos os algoritmos $\alpha \in A$, que podem ser utilizados para resolver determinado problema. Rice [74] também define que algoritmos similares que possuem diferentes configurações ou parâmetros são considerados distintos, de modo que o espaço A considera não só os diferentes tipos modelos de ML que podem ser utilizados para resolver o problema, mas também o conjunto de parâmetros possíveis de serem utilizados.

Idealmente, deveriam haver milhões de algoritmos α com o objetivo de encontrar aquele que mais se adeque à janela avaliada. Nesta pesquisa, no entanto, não será feita

a recomendação ou substituição do modelo de nível base, deseja-se somente prever o desempenho deste, de modo que a escolha dos algoritmos foi motivada pela validação da proposta em modelos com viéses diferentes e não necessariamente que apresentassem melhor desempenho no nível base.

3.1.3 Espaço de métricas de desempenho Y

Consiste no conjunto de métricas que podem ser utilizadas para medir o desempenho dos algoritmos α de determinado problema. Este critério é complexo e depende do objetivo do problema em questão, podem ser avaliados, por exemplo, o tempo de execução, uso de memória, acurácia ou precisão para modelos de classificação, r2 ou erro quadrático médio para modelos de regressão, dentre outros.

Nesta pesquisa, os experimentos se restringem à problemas de classificação, de modo que foram utilizadas medidas clássicas da literatura de ML como precisão e revocação, ou *recall*, além do *Kappa* de Cohen [76], medida clássica da literatura de fluxo de dados.

3.1.4 Espaço de atributo F

Consiste no conjunto de características que podem ser utilizados para descrever o problema em questão, como medidas estatísticas ou descritivas que representem a base e seus atributos. Em problemas de MtL, o espaço de atributo é nomeado de meta atributos, ou *meta features* (MFe), este será descrito com mais detalhes na seção 3.3.

3.2 Arquitetura de meta aprendizado

Nesta seção será apresentada a arquitetura utilizada comumente nos problemas de MtL, conforme a definição de [39] onde os autores dividem a arquitetura em duas macro etapas: a primeira se refere ao processo utilizado para aquisição de conhecimento da base de dados, o segundo se refere à etapa de predição.

3.2.1 Etapa de aquisição de conhecimento

O objetivo desta etapa é criar uma nova base de dados, denominada meta base, contendo informações referentes ao processo de aprendizado. Conforme ilustrado na Figura 3.2, o algoritmo inicia a partir da seleção de uma ou mais bases de dados $x \in P$ nas quais um conjunto de técnicas de pré processamento e ML são utilizadas para transformar as instâncias da base de dados em predições, aqui é gerado o espaço de algoritmos A. Em

seguida, um algoritmo específico $\alpha \in A$ é escolhido como estratégia principal para realização das predições a partir da otimização de alguma métrica de avaliação de desempenho $y \in Y$.

De modo paralelo, é realizada a extração de um conjunto de métricas de caracterização da base de dados analisada; idealmente, as MFe correspondem a uma série de atributos que contenham informações acerca do desempenho dos algoritmos A. Existem diversos tipos de MFe com níveis de complexidade e custo computacional distintos, esta parte será abordada com mais detalhes na Seção 3.3. Por fim, a métrica escolhida y é utilizada para avaliar o desempenho e, em conjunto com as MFe, são utilizadas na construção da meta base, a nova base responsável por acumular o conhecimento acerca do aprendizado dos algoritmos do espaço A.

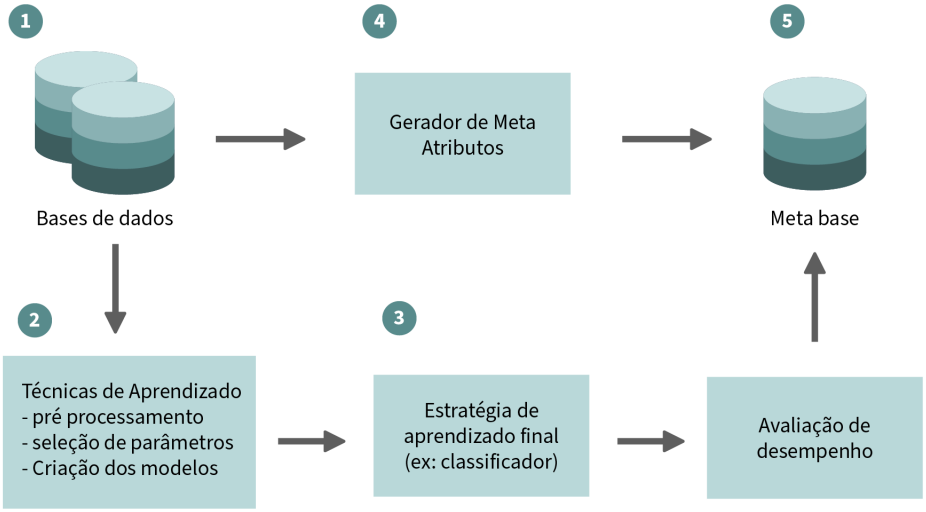


Figura 3.2: Etapa de aquisição de conhecimento. Figura adaptada de Vilalta et al. (2009)

3.2.2 Etapa de predição

A etapa de predição se inicia em sequência, o seu objetivo é predizer o algoritmo $\alpha \in A$ que melhor se adequa à janela analisada visando a otimização da métrica de avaliação de desempenho $y \in Y$ e, assim, melhorando o desempenho geral do sistema de ML proposto. A Figura 3.3 ilustra o seu funcionamento.

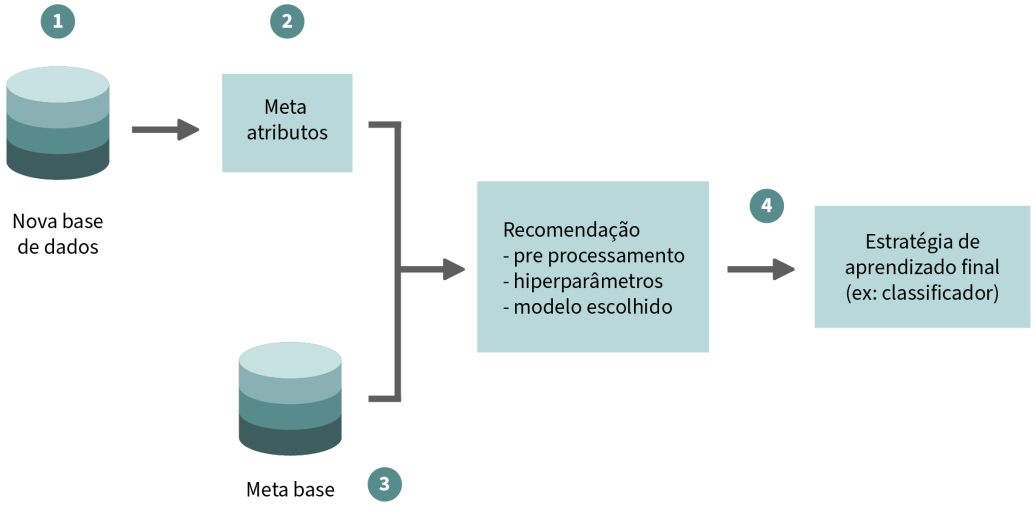


Figura 3.3: Etapa de predição. Figura adaptada de Vilalta et al. (2009)

Quando esta etapa se inicia, tem-se uma nova base de dados $x \in P$ à qual se deseja fazer predições utilizando o algoritmo do espaço A mais adequado; esta nova base é utilizada para criar novas MFe de modo semelhante à etapa de aquisição de conhecimento (item 4 da Figura 3.2). Em seguida, a meta base criada na etapa de aquisição de conhecimento (item 5 da Figura 3.2) é utilizada, em conjunto com as MFe recém calculadas, para gerar uma recomendação do algoritmo α ideal para a nova base. Por fim, esta recomendação gerada é utilizada como algoritmo preditivo na base atual, otimizando o seu desempenho por meio da seleção do modelo que contenha o viés mais adequado ao dado atual.

3.3 Meta atributos

Extrair informação útil para caracterizar os dados existentes em cada janela ainda é um desafio de MtL que impacta de forma direta na qualidade do meta modelo gerado. Em [77] os autores fazem um levantamento e padronização de medidas de caracterização de dados para bases de dados de classificação usados em MtL que serão descritas com mais detalhes nesta seção.

O grupo de MFe simples é composto por medidas facilmente extraídas dos dados e não requer recursos computacionais significativos [78], como a quantidade de atributos numéricos e categóricos da base ou o número de atributos e instâncias, estes dois últimos podem ser combinados para gerar medidas de dimensionalidade e esparsidade (*sparsity*).

Também são extraídas informações relativas à variável alvo, como a quantidade de classes e a proporção de cada classe na base, indicativo direto do desbalanceamento da variável alvo [77]. Por fim, são extraídas medidas referentes à qualidade dos dados, como

a quantidade de instâncias ou atributos nulos na base. Praticamente todas as MFe deste grupo podem ser calculadas para problemas de classificação ou regressão e não dependem da variável alvo sendo, portanto, MFe não supervisionadas.

As MFe estatísticas extraem informações sobre a distribuição das variáveis numéricas do modelo base. São calculados valores estatísticos básicos de cada variável, como média, mínimo, máximo e desvio padrão, além de outras métricas como média geométrica, Equação 3.1, e média harmônica, Equação 3.2. Praticamente todas as MFe deste grupo podem ser calculadas para problemas de classificação ou regressão e não dependem da variável alvo, sendo portanto, MFe não supervisionadas.

$$geometric\_mean = \left(\prod_{i=1}^{n} a_i \right)^{\frac{1}{n}} \tag{3.1}$$

$$harmonic\_mean = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \tag{3.2}$$

Este grupo também contempla o intervalo interquartil, ou *Interquartile Range* (IQR), medida de dispersão estatística para avaliar o grau de espalhamento dos dados que consiste na diferença entre o 1º e 3º quartil da distribuição, conforme ilustrado na Figura 3.4. Além do número de valores atípicos, ou *outliers*, que representa a quantidade de valores extremos que diferem da maioria das instâncias no conjunto de dados para um atributo, ele é calculado a partir do IQR seguindo a Equação 3.3.

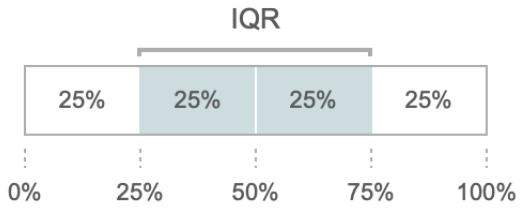


Figura 3.4: Cálculo do IQR

$$nr\_outliers = \sum f(x_i), f(x) = \begin{cases} 0, \ Q1 - 1.5 \ IQR < x < Q3 + 1.5 \ IQR \\ 1, \ otherwise \end{cases} \tag{3.3}$$

Por fim, tem-se as medidas de covariância e correlação que medem a dependência entre os atributos preditivos bem como o número de atributos altamente correlacionados, estas MFe indicam redundância nos dados se possuírem valores elevados [77]. A Equação 3.4 define o coeficiente de correlação de *Pearson* que será utilizado neste trabalho como medida de similaridade entre dois atributos contínuos.

$$r = \frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{\sqrt{\left[n\sum x^2 - \left(\sum x\right)^2\right]\left[n\sum y^2 - \left(\sum y\right)^2\right]}} \tag{3.4}$$

As MFe de clusterização são calculadas a partir dos metadados gerados por um algoritmo de clusterização nas variáveis explicativas do modelo base. Para isso, supõe-se que as instâncias estão distribuídas em grupos, denominados *clusters*. A compacidade, ou *compactness*, mede o quão compactos estão os grupos gerados pelo algoritmo a partir da soma da distância ao quadrado de cada ponto ao centróide seguindo a Equação 3.5.

$$compactness = \sum_{c}^{K} \sqrt[2]{\sum_{i}(x_i - \mu_c)^2} \tag{3.5}$$

Onde K é o número de grupos e μ_c é o centróide do grupo c.

Outras métricas intrínsecas dos algoritmos de clusterização são o número de iterações necessárias para executar o algoritmo até o critério de parada e a inércia que mede quão bem o conjunto de dados foi agrupado por meio do cálculo da soma dos erros quadrados entre o ponto e o seu respectivo centróide, conforme a Equação 3.6.

$$inertia = \sum_{i}(x_i - \mu)^2 \tag{3.6}$$

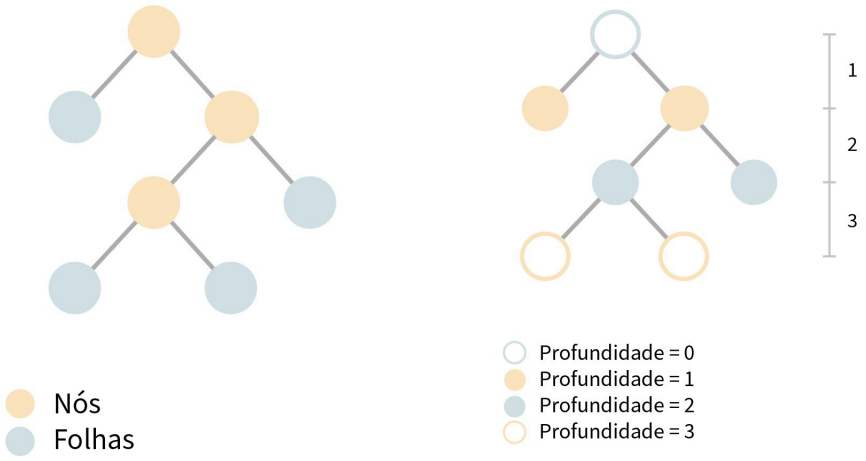
O grupo de MFe de teoria da informação se refere à medidas que capturam a quantidade de informação nos dados a partir de medidas de variabilidade e redundância dos atributos [77]. A maior parte deste grupo é restrita para problemas de classificação, como a entropia entre as classes, a informação mútua, medida da relação de cada atributo com a variável alvo ou a relação sinal ruído que mede a proporção dos dados que são irrelevantes para o problema avaliado. No levantamento de [77], a única MFe que não é exclusiva para problemas de classificação é entropia dos atributos preditivos, medida da incerteza média dos atributos preditivos, calculada segundo a Equação 3.7.

$$entropy = -\sum_{x} P(x)ln(P(x)) \tag{3.7}$$

As MFe baseadas em modelo são extraídas a partir do treinamento de modelos de ML, a maior parte se refere a propriedades de modelos baseados em árvore, como número de folhas, nós e profundidade, conceitos representados na Figura 3.5, além de medidas extraídas a partir destes valores, como a proporção de folhas por classe e a proporção de nós por atributos ou instâncias.

Aqui também é calculado o grau de desbalanceamento da árvore e a importância das variáveis, este último pode ser medido de diversas formas a depender do tipo de algoritmo de árvore utilizado [77]. Uma observação importante é o fato de o treinamento da árvore

estar condicionado à obtenção da variável alvo do problema em questão, portanto, todo este grupo se refere à um conjunto de MFe supervisionadas.



(a) Folhas e nós

(b) Profundidade

Figura 3.5: Conceitos dos algoritmos baseados em árvores

Também chamado de *Landmarking*, o grupo de MFe de referência faz uso do desempenho de algoritmos de aprendizado rápido como métricas de caracterização da base, em oposição ao grupo de MFe baseados em modelo, os metadados dos algoritmos não são utilizados, somente o seu desempenho. Duas MFe do levantamento de [77] originam do K-ésimo vizinho mais próximo, ou *K-Nearest Neighbors* (KNN), algoritmo simples que associa cada nova instância à classe majoritária dentre os seus K vizinhos mais próximos; ambas as MFe utilizam apenas um vizinho ($K = 1$) no cálculo, situação em que o algoritmo é nomeado 1NN, enquanto uma utiliza todos os atributos de entrada disponíveis no modelo, a outra utiliza somente os atributos mais informativos.

Outros algoritmos de cálculo rápido são utilizados como o *Naive Bayes*, que determina a probabilidade de uma nova instância pertencer à determinada classe com base no Teorema de Bayes; e a Análise discriminante, ou *Linear Discriminant*, encontrando a combinação linear dos atributos que maximize a separação entre as classes. De modo similar ao grupo de MFe baseadas em modelo, o treinamento destes algoritmos depende da obtenção da variável alvo do problema em questão, portanto, todo este grupo também se refere à um conjunto de MFe supervisionadas.

Por fim, há o grupo de MFe diversas, também denominado *Miscellaneous*, que se refere às demais MFe que foram citadas na literatura mas não pertencem a algum dos grupos pré existentes. Alguns exemplos são a porcentagem de valores únicos de um atributo, medidas referentes ao tempo transcorrido para o cálculo das demais MFe e a escassez, ou

(*sparsity*), valor que mede o quão discreto é um atributo, calculado a partir da contagem de elementos não nulos na variável conforme a Equação 3.8.

$$sparsity = 1 - \frac{count\_non\_zero}{total\_of\_elements} \tag{3.8}$$

Outra MFe é originada da Análise de Componentes Principais, ou *Principal Component Analysis* (PCA), técnica de redução de dimensionalidade que funciona a partir da criação de novas variáveis não correlacionadas, denominadas componentes principais. O algoritmo se baseia em encontrar hiperplanos que aproximam os dados da melhor forma possível por meio da aproximação de mínimos quadrados de modo que a variância seja maximizada. Esta MFe é definida como a proporção de componentes principais, com relação ao total de atributos da base de dados original, que explica 95% da variância dos dados.

3.4 Meta stream

A aplicação de MtL para fluxo de dados, também denominada *meta stream*, consiste na recomendação periódica de algoritmos em ambientes não estacionários proposta por [42, 43, 44, 45]. Os autores descrevem esta técnica como uma abordagem promissora nos problemas em que o desempenho dos algoritmos são similares em grandes períodos de tempo mas diferentes quando analisados em janelas temporais pequenas. Para tal, são feitas algumas alterações no processo original de MtL como o espaço de problemas P ser composto por apenas uma base de dados x, o fluxo de dados em questão, enquanto o desempenho $y \in Y$ dos algoritmos $\alpha \in A$ e as MFe $f(x) \in F$ são extraídos dos dados de cada janela temporal.

Em [42], a arquitetura da solução é divida em dois níveis de abstração, o primeiro, denominado nível base, consiste no treinamento de um conjunto de regressores $\alpha \in A$ que são avaliados com as instâncias seguintes do fluxo de dados. O segundo, denominado nível meta, consiste no treinamento de um novo algoritmo de ML, denominado meta modelo, cujo objetivo é encontrar a relação entre os dados no nível base e o desempenho dos regressores. Este meta modelo é, então, utilizado para predizer o algoritmo mais adequado a cada janela de dados. Esta arquitetura, descrita por [42], está detalhada nas próximas seções e será utilizada na presente pesquisa com algumas alterações, como o fato de ter se optado por trabalhar com problemas de classificação no nível base.

Em [45], os autores propõem um novo *framework* denominado *micro metaStream* baseado na hipótese de que o algoritmo mais adequado pode mudar repentinamente ao longo do tempo. A técnica se baseia na seleção de algoritmos para cada nova instância, em oposição ao *metaStream* que recomenda algoritmos periodicamente em janelas. Embora o

micro metaStream tenha apresentado um desempenho superior no nível meta, a melhoria no nível base foi sutil. Além disso, a predição por instância, e não janelas, aumenta o custo computacional do sistema, portanto optou-se por não utilizá-la neste trabalho.

Em [43], os autores apresentam uma abordagem detalhada para construir meta atributos visando a caracterização de fluxos de dados. O *metaStream* foi empregado para analisar a influência das MFe dependentes e independentes da morfologia dos dados. Essa investigação foi motivada pela hipótese de que, com informações adequadas sobre os dados, a seleção de algoritmos para fluxos de dados pode ser realizada de forma mais eficiente. Ao final, foram identificadas MFe relevantes a partir dos dados do nível base e suas relações, permitindo um processo de seleção de algoritmos mais refinado e ajustado às particularidades dos fluxos de dados em questão.

Em [44] os autores propõem um aprimoramento do trabalho de [42] a partir do treinamento incremental do meta modelo e do uso do conjunto de MFe listadas em [77], ao final, foi constatado que diversas destas MFe se tornaram atributos de alta importância para o meta modelo, sendo assim, optou-se por utilizá-las também nesta pesquisa em conjunto com outras MFe propostas. Por outro lado, o ganho resultante do treinamento incremental não foi extenso, de modo que esta alteração não foi mantida no escopo deste trabalho.

3.4.1 Nível base

No nível base, o fluxo de dados do problema em questão é tratado com uma janela deslizante de tamanho fixo, mecanismo de esquecimento que descarta instâncias muito antigas da base, conforme ilustrado na Figura 3.6. Os autores [42] consideram que existe atraso na chegada da variável alvo de tamanho η_b, além disso, para cada determinado momento t, a janela atual contém um conjunto de dados com variável alvo conhecida de tamanho ω_b. Estes dados, representados em azul na Figura 3.6, são utilizados para treinar os algoritmos do espaço A cujo objetivo é prever o valor da variável alvo para instâncias futuras do fluxo de dados na janela de avaliação de tamanho λ_b, representada em cinza na Figura 3.6.

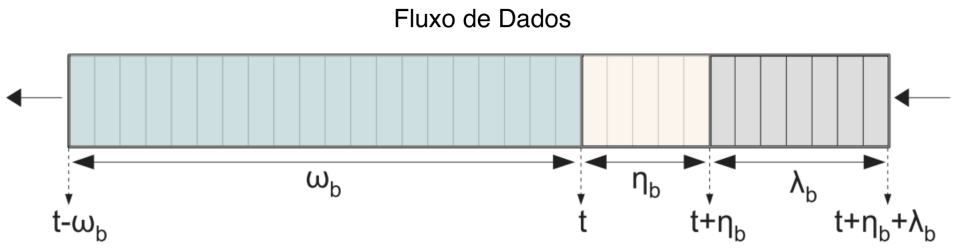


Figura 3.6: Fluxo de dados em uma janela no nível base. Figura adaptada de Rossi et al. (2014)

3.4.2 Nível meta

O objetivo aqui é criar um meta modelo capaz de relacionar adequadamente os dados no nível base e o desempenho dos algoritmos do espaço A, em [42] o nível meta é dividido em três etapas principais, conforme ilustrado na Figura 3.7. A primeira consiste na geração dos meta dados necessários, a segunda se refere ao treinamento do meta modelo e, por fim, a última consiste na utilização do meta modelo treinado para recomendar o algoritmo α ideal para a janela em questão.

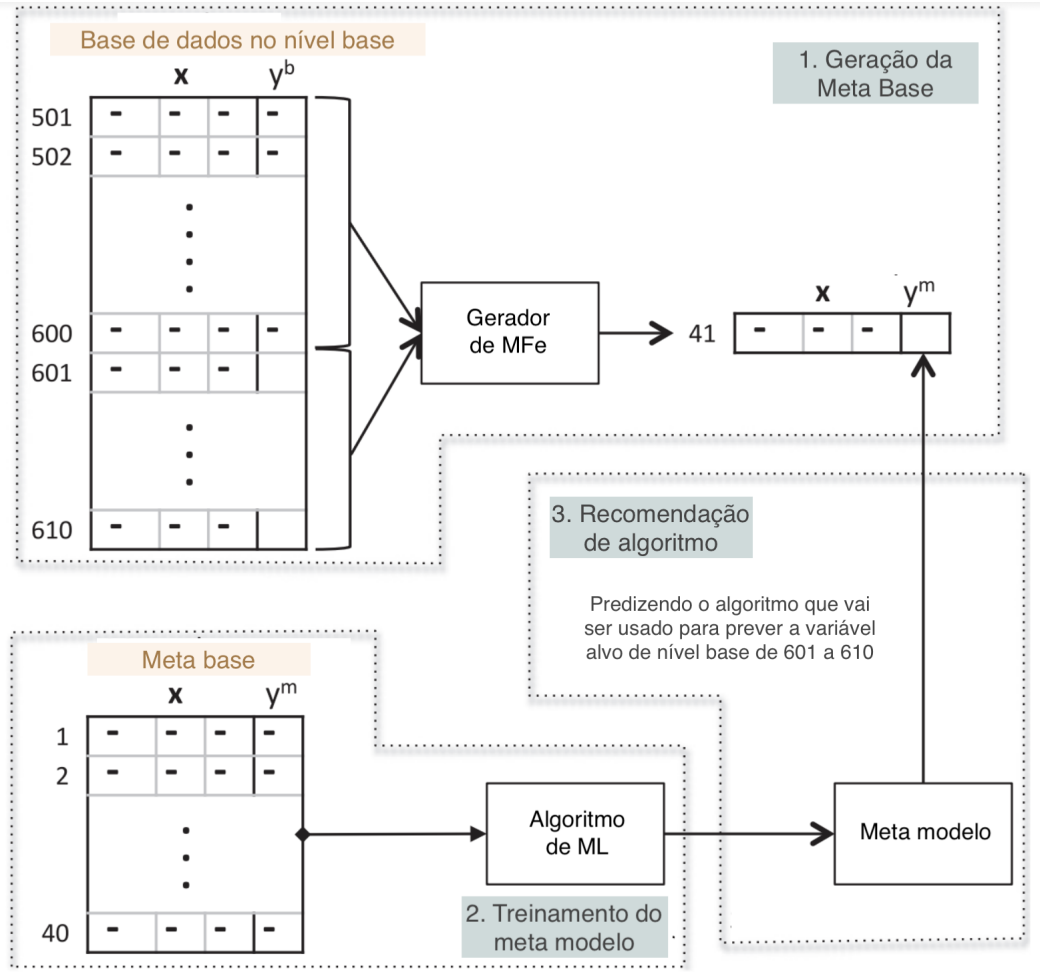


Figura 3.7: Exemplo de geração de uma meta instância e a predição do algoritmo ideal para a janela. Figura adaptada de Rossi et al. (2014)

A primeira etapa do nível meta consiste na construção de uma nova base que sumarize os meta dados de cada janela avaliada, aqui, um conjunto de MFe é extraído dos últimos dados conhecidos com (ω_b) ou sem (η_b) variável alvo. Cada meta instância é composta por estas MFe e a sua variável alvo é o identificador do algoritmo $\alpha \in A$ que apresentou melhor desempenho na janela conhecida ω_b.

Após a geração de uma quantidade razoável de meta instâncias, o meta modelo é treinado utilizando uma janela deslizante na meta base, de modo similar ao que é feito no nível base, para garantir que meta instâncias muito antigas não prejudiquem o aprendizado do classificador. Por fim, o meta modelo treinado na etapa anterior é utilizado para prever o algoritmo, ou combinação de algoritmos, de nível base $\alpha \in A$ que mais se adeque à janela atual.

3.5 Contribuições desta pesquisa

Em [42], os autores relatam que o *metaStream* apresenta uma proposta promissora para problemas em que o desempenho dos algoritmos são similares em grandes períodos de tempo mas muito diferentes quando analisados em janelas temporais pequenas, mas isso não é verdade para todos as bases de dados e, em algumas delas, é possível que o MtL esteja fornecendo um ganho sutil, ou mesmo negativo, com relação às abordagens tradicionais de ML em troca do custo computacional elevado desta técnica.

Com isto em mente, esta pesquisa se propõe a investigar o uso da arquitetura de MtL proposta por [42] para realizar a detecção de mudança de conceito, em oposição à abordagem tradicional de seleção de algoritmos. Espera-se que a robustez do MtL viabilize a obtenção de resultados superiores às técnicas atuais de detecção de mudança de conceito e que a geração de alertas confiáveis seja uma forma de aumentar a visibilidade do desempenho atual antes da chegada da variável alvo, permitindo a realização de ajustes para melhoria de desempenho com outras abordagens tradicionais de ML, como a criação de novos atributos no nível base, otimização de parâmetros, alteração dos limiares da fronteira de decisão, retreino do modelo, dentre outros.

Sabendo que o objetivo deste trabalho é fazer a detecção da mudança de conceito, outras alterações se mostram necessárias. A primeira é o fato de o espaço de Algoritmos A ser composto por um único modelo base α, o qual se quer monitorar, além disso, o meta modelo deve prever o desempenho do modelo base, o que implica na utilização de um regressor no nível meta, e não um classificador como no problema de seleção de algoritmos.

Por fim, a arquitetura de MtL descrita nesta seção assume que o atraso para a chegada da variável alvo (η_b) é pequeno, o que também não é verdade para muitos problemas de fluxo de dados. Na presente pesquisa, considera-se que exista um atraso grande na chegada da variável alvo e somente a janela η_b é utilizada no cálculo das MFe, portanto, somente as medidas não supervisionadas da Seção 3.3 serão utilizadas na geração da meta base. Por fim, algumas das métricas de detecção de mudança de conceito descritas na Seção 2.2 foram utilizadas em conjunto com as MFe citadas por [77], pois supõe-se que estes atributos adicionam informações relevantes para o problema abordado.

Capítulo 4

Metodologia

Neste capítulo será abordada a metodologia proposta para resolver o problema da detecção de mudança de conceito com auxílio de MtL. A proposta é validar se um meta regressor que utiliza métricas não supervisionadas de mudança de conceito como parte de seus meta atributos é capaz de prever o desempenho do modelo base e, então, utilizar as predições para realizar a geração de alertas confiáveis de mudança de conceito.

As bases de dados utilizadas se referem à fluxos de dados provenientes de ambientes altamente dinâmicos, sujeitos à mudança de conceito e que apresentam atraso grande na chegada da variável alvo. Além disso, supõe-se que existe uma base de dados rotulada com dados históricos utilizada no treinamento do modelo base. Por fim, todas as bases de dados utilizadas se referem a problemas de classificação com dados pouco ou nada desbalanceados.

O restante do capítulo está dividido da seguinte maneira: Na Seção 4.1 será apresentada uma visão geral da metodologia proposta, em seguida, na Seção 4.2 são detalhadas as escolhas e definições feitas para construção do problema. A Seção 4.3 detalha a arquitetura da solução e a Seção 4.4 esclarece os parâmetros da arquitetura utilizados no experimento, como as escolhas de projeto referentes ao meta modelo utilizado e como foi feita a geração dos alertas de mudança de conceito. Por fim, a Seção 4.5 descreve os métodos de avaliação utilizados para atestar os resultados obtidos e a Seção 4.6 detalha brevemente a implementação utilizada nos experimentos da pesquisa.

4.1 Visão geral

Esta seção oferece uma visão geral da metodologia proposta, seguindo a abordagem definida em [39] e descrita na Seção 3.2. A primeira etapa consiste na aquisição de conhecimento do meta modelo, representada na Figura 4.1. Em cada janela de dados, são extraídas características que resumem as informações daquela janela, constituindo os atri-

butos da meta base. Simultaneamente, quando o valor real da variável alvo associada à janela chega ao sistema, o desempenho do modelo de nível base é calculado. Esse valor de desempenho é usado como o meta rótulo da instância correspondente na meta base. À medida que várias instâncias da meta base são calculadas, é feito o treinamento do meta modelo, algoritmo que aprende como as variações nas características dos dados de entrada das janelas impactam o desempenho do modelo base.

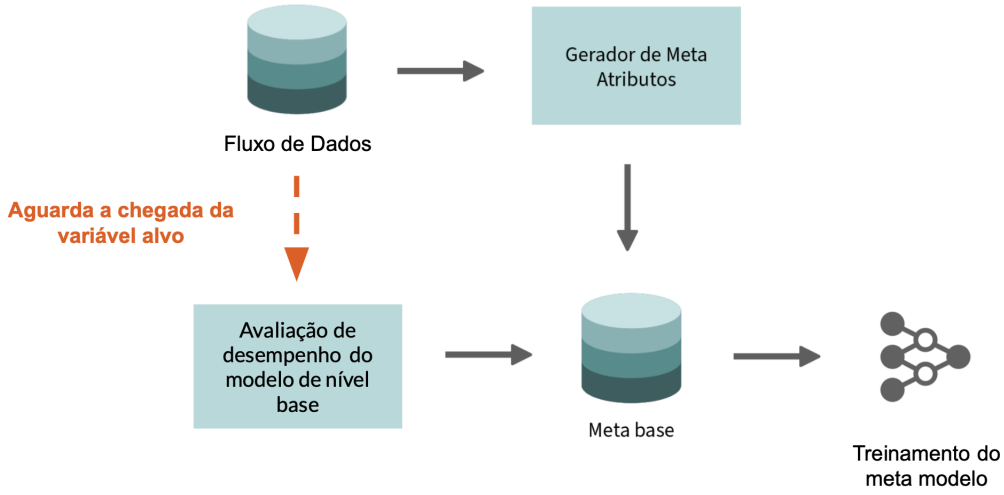


Figura 4.1: Visão geral da etapa de aquisição de conhecimento

Em seguida, se inicia a etapa de predição, conforme ilustrado no diagrama da Figura 4.2. Quando uma nova janela de dados, que não possui variável alvo definida, chega ao sistema, podemos extrair suas características e utilizar o meta modelo treinado na etapa anterior para descobrir o desempenho do modelo base naquela janela, mesmo na ausência da variável alvo. O desempenho previsto é então usado na geração de alertas confiáveis de mudança de conceito.

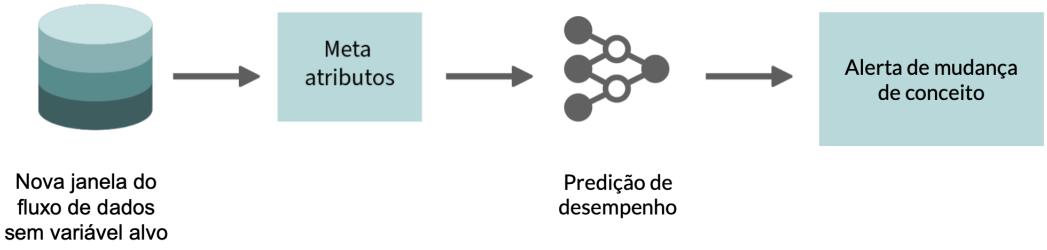


Figura 4.2: Visão geral da etapa de predição

4.2 Definições do problema

Nesta seção serão apresentadas as escolhas feitas na construção do problema seguindo a definição de [42]. É importante salientar que os problemas de MtL trabalham com os dados em dois níveis de abstração, o nível base corresponde ao fluxo tradicional de treinamento de modelos de ML com o objetivo de resolver um problema prático. Por outro lado, o nível meta faz um janelamento da base de dados original para construção da meta base, novo conjunto de dados em que cada instância caracteriza uma janela da base original de modo sumarizado. Este é então utilizado para treinar o meta modelo, que nesta pesquisa será responsável por prever o desempenho do modelo base.

4.2.1 Espaço de algoritmo

O espaço de algoritmo A consiste no conjunto de todos os algoritmos que podem ser utilizados para resolver o problema no nível base. Na abordagem original de seleção de algoritmos abordado em MtL, é ideal que muitos modelos sejam utilizados no nível base com o objetivo de encontrar aquele que mais se adéqua à janela avaliada. Nesta pesquisa, no entanto, não será feita a recomendação ou substituição do modelo de nível base, deseja-se somente prever o seu desempenho. Por este motivo, a escolha dos algoritmos se deve à validação da proposta em modelos com viéses diferentes e não necessariamente que apresentassem melhor desempenho no nível base.

Ao final, quatro algoritmos foram escolhidos para compor esta pesquisa: SVM [79], Floresta Aleatória, ou *Random Forest* (RF) [80], Regressão Logística, ou *Logistic Regression* (LR) [81], e o algoritmo CART para indução de Árvore de Decisão, que será denominado aqui como *Decision Tree* (DT) [82]. É importante notar que a LR é um algoritmo exclusivo para problemas de classificação binária, apesar disso, algumas bases de dados multiclasse foram utilizadas na pesquisa, para estes casos foi utilizada a estratégia um contra todos, ou *One-vs-the-rest*, que consiste em treinar um algoritmo para detectar cada classe do problema individualmente e combinar as classificações para realizar somente uma predição.

Para a construção dos modelos base, foram adotados os parâmetros padrão da biblioteca *Scikit-learn* [83] em todos os algoritmos, com exceção dos modelos RF e DT, nos quais o parâmetro '*max_depth*' foi especificamente configurado. Essa escolha foi motivada pelo fato de que, por padrão, a biblioteca não limita o crescimento das árvores, o que pode torná-las mais suscetíveis a *overfitting*.

4.2.2 Espaço de métricas de desempenho

A queda de desempenho do modelo base é um indicativo direto da existência de mudança de conceito, portanto, nesta pesquisa os meta rótulos são constituídos pelas métricas de desempenho do modelo base. Como as bases escolhidas para os experimentos se referem a problemas de classificação, foram utilizadas medidas clássicas da literatura de ML como a revocação, ou *recall*, e a precisão, ou *precision*, que também são aplicadas em problemas de fluxo de dados [84] apesar de serem métricas usuais da abordagem tradicional de ML.

Também foi utilizado o *F1 score*, medida estatística calculada a partir da revocação e precisão. Em [1] os autores constatam que as medidas de avaliação tradicionais não são suficientes no cenário de fluxo de dados, portanto, também foi incluído o Cohen *Kappa* [76], estatística que mede a concordância entre a predição do classificador e a variável alvo real.

4.2.3 Espaço de problema

O espaço de problema P é composto pelo conjunto de bases de dados empregadas no estudo. De acordo com a definição original de *metaStream* [42], cada meta base é gerada a partir de apenas um fluxo de dados. Portanto, cada experimento realizado neste trabalho utiliza uma única base de dados específica. Entretanto, para validar as hipóteses de pesquisa, foram realizados diversos experimentos com o propósito de ampliar a robustez das conclusões. Dessa forma, ao final do estudo, foram empregadas múltiplas bases de dados para enriquecer e fortalecer a análise.

Nesta seção, serão apresentadas as bases de dados utilizadas nos experimentos, divididos em dois tipos distintos. Os primeiros experimentos têm como objetivo avaliar a capacidade do meta modelo em realizar predições de desempenho adequadas. Nesse contexto, foram empregadas bases de dados reais sem mudança de conceito rotulada, buscando assimilar a aplicação real do algoritmo proposto. Em seguida, para avaliar a qualidade da predição de mudança de conceito proposta, foram utilizadas bases de dados sintéticas com mudança de conceito rotulada. Essa abordagem permitirá uma análise mais abrangente e precisa da eficácia da ferramenta em detectar mudanças de conceito ao longo do fluxo de dados.

Bases de dados reais

Em [85], os autores propõem a criação de um repositório público de dados para problemas reais de fluxo de dados, o repositório de *Data Stream* da Universidade de São Paulo (USP). Essa iniciativa visa mitigar as dificuldades encontradas por pesquisadores da área na comparação de algoritmos propostos na literatura devido à falta de conjuntos de dados

não estacionários do mundo real disponíveis publicamente. Algumas destas bases de dados foram utilizadas nos experimentos da pesquisa e serão detalhadas nesta Seção.

No escopo do presente trabalho, todas as bases reais utilizadas se referem à problemas de classificação, multiclasse ou binários, com pouco ou nenhum desbalanceamento na variável alvo. A Tabela 4.1 descreve os metadados das bases de dados.

Característica	Electricity	Airlines	Powersupply	Rialto
Nº de instâncias	45.312	539.383	29.928	82.250
Nº de atributos	6	6	2	27
Nº de classes	2	2	24	10
Nº de nulos	0	0	0	0
Nº de variáveis numéricas	5	2	2	27
Nº de variáveis categóricas	1	4	0	0

Tabela 4.1: Detalhes das bases de dados utilizadas na pesquisa

A base de dados *electricity* foi descrita por [86], ela contém dados coletados do mercado de eletricidade australiano de Nova Gales do Sul e é amplamente utilizada em problemas de fluxo de dados e estudos de mudança de conceito por se tratar de um ambiente altamente dinâmico, com alterações frequentes nas variáveis explicativas e na variável alvo.

Nesse mercado, os preços não são fixos e são afetados pela demanda e oferta. A variável alvo representa um aumento ou queda do preço com relação à uma média móvel das últimas 24 horas, sendo este um problema de classificação binário com dados pouco desbalanceados, cuja classe majoritária corresponde a aproximadamente 57% das instâncias. A base contém 45.312 instâncias datadas entre 7 de Maio de 1996 a 5 de Dezembro de 1998. Cada instância se refere a um período de 30 minutos deste intervalo; são nove atributos além da variável alvo, estes estão descritas na Tabela 4.2.

Nome da variável	Tipo	Quantidade de valores únicos
date	numérica	933
day	categórica	7
period	numérica	48
nwsprice	numérica	4089
nwsdemand	numérica	5266
vicprice	numérica	3798

Tabela 4.2: Atributos da base de eletricidade

A base de dados *airlines* tem por objetivo prever se um determinado voo sofrerá atraso a partir da informação da partida programada, ela foi inspirada no trabalho de

[87]. A base possui 539.383 instâncias e 6 atributos descritos com mais detalhes na tabela 4.3, além da variável alvo *"Delay"* marcador que indica se houve ou não atraso no voo em questão. Portanto, o problema se trata de uma classificação binária com dados pouco desbalanceados, cuja classe majoritária corresponde a aproximadamente 55% das instâncias.

Nome da variável	Tipo	Quantidade de valores únicos
Airline	categórica	18
Flight	numérica	6585
AirportFrom	categórica	293
AirportTo	categórica	293
DayOfWeek	categórica	7
Time	numérica	1131

Tabela 4.3: Variáveis explicativas da base *airlines*

A base de dados *powersupply* contém os dados do fornecimento de energia de uma empresa de eletricidade italiana que registra a energia de duas formas, a primeira se refere ao fornecimento de energia da rede principal e a segunda à energia transformada de outras redes. Estas duas informações numéricas são as variáveis explicativas da base e os detalhes associados podem ser visualizados na tabela 4.4.

Nome da variável	Tipo	Quantidade de valores únicos
attribute0	numérica	1966
attribute1	numérica	215

Tabela 4.4: Variáveis explicativas da base *powersupply*

Esta base contém registros das fontes de alimentação entre os anos 1995 e 1998, e o problema consiste em prever à qual período do dia, entre 1 e 24 horas, a fonte de alimentação atual pertence. Trata-se de um problema de classificação multiclasse com dados balanceados. Informações externas à base como estação do ano, clima, período do dia, dentre outros, influenciam fortemente a existência de mudança de conceito na base.

A base de dados *rialto* foi proposta por [88] e é composta por imagens de dez edifícios ao lado da ponte Rialto, em Veneza, que foram capturadas por uma *webcam* fixa durante vinte dias consecutivos entre Maio e Junho de 2016. As imagens foram codificadas como histograma RGB normalizado de 27 dimensões e são utilizadas como as variáveis explicativas da presente base tabular. Portanto, todas as variáveis explicativas são numéricas e nenhuma possui valores nulos em sua composição.

A variável alvo é categórica e pode assumir valores de 0 a 10 indicando o prédio relacionado à imagem original. Trata-se de um problema de classificação multiclasse balanceado com forte presença de mudança de conceito consequente das constantes alterações de iluminação e climática.

Bases de dados sintéticas

O pacote River [89], especializado em fluxo de dados foi utilizado para gerar bases de dados sintéticas onde a ocorrência de mudança de conceito é rotulada. Com isto, foi possível definir parâmetros que controlam a frequência, a magnitude e a natureza das mudanças de conceito que foram introduzidas.

Foram geradas bases de dados com diferentes configurações, como a presença ou ausência de ruído nos dados, o balanceamento das classes e diferentes tipos de mudança de conceito. Essa abordagem foi adotada para garantir que a hipótese fosse testada em diversos cenários, a fim de verificar se ela se mantém válida em diferentes contextos.

Os parágrafos que se seguem detalham os geradores do pacote *River* utilizados e a tabela 4.5 relaciona as bases geradas com as suas características. Para cada gerador de mudança de conceito abrupta, um conjunto de dados com mudança de conceito recorrente foi construído repetindo o conceito anterior.

O '*Sine*' é um gerador de fluxo de dados com mudança de conceito abrupta descrita em [21]. São geradas 4 variáveis numéricas, onde apenas 2 delas são relevantes para a tarefa de classificação e as outras 2 podem ser adicionadas opcionalmente como ruído. Uma função de classificação é escolhida para definir a maneira como os rótulos são gerados.

O '*SEA*' é um gerador de fluxo de dados com mudança de conceito abrupta descrita em [90]. Cada observação é composta por 3 variáveis, sendo que apenas as duas primeiras são relevantes para a tarefa de classificação. O alvo da classificação é binário e é considerado positivo quando a soma das características excede um determinado limite. Existem 4 limites diferentes para escolher. A mudança de conceito pode ser introduzida alterando o limite a qualquer momento durante o fluxo de dados.

O '*STAGGER*' é um gerador de fluxo de dados com mudança de conceito abrupta descrita em [91]. Os conceitos são funções booleanas com três características que descrevem objetos: tamanho (pequeno, médio e grande), forma (círculo, quadrado e triângulo) e cor (vermelho, azul e verde). Existem três funções que geram os conceitos e a mudança de conceito pode ser introduzida alterando a função de classificação.

- Verdadeiro se o tamanho for pequeno e a cor for vermelha.
- Verdadeiro se a cor for verde ou a forma for um círculo.
- Verdadeiro se o tamanho for médio ou grande.

O '*MIXED*' é um gerador de fluxo de dados com mudança de conceito abrupta sem ruído descrita em [21]. Ele possui quatro variáveis relevantes para o problema e os exemplos são rotulados dependendo da função de classificação escolhida. A mudança de conceito pode ser introduzida alterando a função de classificação.

O '*Agrawal*' é um gerador de fluxo de dados com mudança de conceito gradual descrita em [92]. O gerador produz um fluxo de dados contendo nove características, sendo seis numéricas e três categóricas. Existem 10 funções definidas para gerar rótulos de classe binários a partir das características, a mudança de conceito gradual é gerada a partir do incremento destas funções de classificação.

Função do River	Base Gerada	Tipo de mudança	Tem Ruído	Balanceada
Sine	abrupt_sine_balanced	Abrupta	Não	Sim
Sine	abrupt_sine_unbalanced	Abrupta	Não	Não
Sine	abrupt_sine_balanced_noise	Abrupta	Sim	Sim
Sine	abrupt_sine_unbalanced_noise	Abrupta	Sim	Não
Sine	abrupt_recurring_sine_balanced	Recorrente	Não	Sim
SEA	abrupt_sea	Abrupta	Não	-
SEA	abrupt_sea_noise	Abrupta	Sim	-
SEA	abrupt_recurring_sea	Recorrente	Não	-
STAGGER	abrupt_stagger_balanced	Abrupta	-	Sim
STAGGER	abrupt_stagger_unbalanced	Abrupta	-	Não
STAGGER	abrupt_recurring_stagger_balanced	Recorrente	-	Sim
Mixed	abrupt_mixed_balanced	Abrupta	-	Sim
Mixed	abrupt_mixed_unbalanced	Abrupta	-	Não
Mixed	abrupt_recurring_mixed_balanced	Recorrente	-	Sim
Agrawal	gradual_agrawal_balanced	Gradual	Não	Sim
Agrawal	gradual_agrawal_unbalanced	Gradual	Não	Não
Agrawal	gradual_agrawal_balanced_with_noise	Gradual	Sim	Sim
Agrawal	gradual_agrawal_unbalanced_with_noise	Gradual	Sim	Não

Tabela 4.5: Bases de dados sintéticas

4.2.4 Espaço de atributo

O espaço de atributo F consiste no conjunto de características que podem ser utilizados para descrever o problema em questão, como MFe estatísticas ou descritivas que representem a base e seus atributos. O problema abordado neste trabalho considera que há

atraso grande na chegada da variável alvo, portanto, apenas as MFe não supervisionadas descritas na Seção 3.3 foram utilizadas nos experimentos.

Além disso, algumas das métricas de detecção de mudança de conceito descritas na Seção 2.2 foram utilizadas em conjunto com as MFe citadas por [77], pois supõe-se que estes meta atributos adicionam informações relevantes para o problema abordado. Pelo mesmo motivo, também foi adicionada a predição do modelo base e o desempenho do mesmo na última janela com variável alvo conhecida no conjunto de MFe. As MFe foram separadas em categorias e, além do nome, foi detalhado o tipo da métrica entre:

- Univariado: Calculado individualmente para cada coluna, gerando uma MFe por atributo base.
- Bivariado: Calculado utilizando os atributos base combinados 2 a 2, gerando $\frac{n!}{2(n-2)!}$ MFe, onde n é o número de colunas da base.
- Multivariado: Utiliza múltiplos atributos no calculo da métrica, gerando uma única MFe para a janela avaliada.

Meta atributos tradicionais

As MFe simples se referem à informações de caracterização geral da base que não são informativas para o problema em questão, este grupo abrange MFe relativas ao número de atributos e classes da base, que são fixos para o caso em que o espaço de problemas P é composto por uma única base x, além de informações relacionadas à quantidade de instâncias, valor constante dado que optou-se por trabalhar com janela deslizante de tamanho fixo, portanto, nenhuma MFe deste grupo foi utilizada na pesquisa.

A maior parte das MFe estatísticas descritas na Seção 3.3 foram utilizadas nos experimentos e estão detalhadas na Tabela 4.6. Este grupo é responsável por extrair informações sobre a distribuição dos atributos de entrada do modelo base e constituem o grupo que gera a maior quantidade de variáveis. Isso ocorre porque, além de possuir mais métricas, a maioria delas é univariada, resultando em uma MFe específica para cada atributo base.

As MFe de teoria da informação são, em sua maioria, restritas a problemas de classificação cuja variável alvo é conhecida *a priori*. Como este não é o caso do problema abordado nesta pesquisa apenas uma medida deste grupo foi utilizada, a entropia, ou *entropy*, medida univariada que representa a incerteza média dos atributos. De modo similar, as MFe baseadas em modelo e MFe de referêcia, ou *Landmarking*, também se referem à medidas de caracterização supervisionadas, de modo que estas não foram adicionadas aos experimentos da pesquisa. Por fim, a Tabela 4.7 relaciona as medidas do grupo de MFe diversas, ou *Miscellaneous*, que foram utilizadas.

Nome	Descrição	Tipo
mean	Média	univariada
std	Desvio Padrão	univariada
min	Mínimo	univariada
max	Máximo	univariada
gmean	Média geométrica	univariada
hmean	Média harmônica	univariada
IQR	Faixa interquartil	univariada
nr_outliers	Número de outliers	univariada
corr	Correlação de pearson dos atributos	bivariada

Tabela 4.6: Meta atributos estatísticos

Nome	Descrição	Tipo
uniqueness_ratio	Porcentagem de valores únicos na coluna	univariada
sparsity	Mede o quão discreto é um atributo	univariada
prop_pca	Proporção de componentes principais que explicam 95% da variância da base de dados	multivariada

Tabela 4.7: Meta atributos diversos

As medidas do grupo MFe de clusterização foram calculadas por meio do algoritmo de agrupamento *kMeans*, algoritmo iterativo que particiona o conjunto de dados em grupos de tamanho predefinido k de modo que cada instância da base de dados esteja associado ao grupo que minimize a distância entre o ponto e o centróide do conjunto.

Além das MFe de clusterização descritas na Seção 3.3, algumas métricas internas do *kMeans* foram utilizadas, como o número de iterações necessárias para executar o algoritmo até o critério de parada. A inércia mede o quão bem o conjunto de dados foi agrupado por meio do cálculo da soma dos erros quadrados entre o ponto e o seu respectivo centróide.

O número de grupos utilizado é um parâmetro de inicialização do algoritmo e o seu valor ideal foi encontrado com auxílio do método do cotovelo, técnica que se resume em aplicar o *kMeans* com diferentes valores de k e calcular as suas respectivas inércias. De modo geral, o valor de inércia decresce à medida em que o valor de k aumenta, comportamento que é desejável até certo ponto. Após isso, o valor do erro é reduzido mas muitos grupos são gerados e a generalização do algoritmo não fica adequada, portanto, o ponto ótimo desta curva é o 'cotovelo' do gráfico.

Nome	Descrição	Tipo
compacness	Soma da distância ao quadrado de cada ponto ao centróide	multivariada
n_iter	Número de iterações necessárias para adequar os dados	multivariada
inertia	Soma dos erros quadrados	
n_clusters	Número ótimo de clusters	multivariada

Tabela 4.8: Meta atributos de clusterização

Meta atributos de mudança de conceito

Algumas métricas de mudança de conceito descritas na Seção 2.2 foram utilizadas como meta atributos como forma de investigar o seu impacto no meta modelo. A escolha das métricas que integraram esta pesquisa se deve à alguns fatores como simplicidade de implementação e custo computacional de execução, visto que o problema de fluxo de dados por si só implica em recursos computacionais limitados [59]. Além disso, foram consideradas medidas que possuíssem viéses diferentes para maximizar a contribuição deste grupo no desempenho do meta modelo.

A Tabela 4.9 detalha as MFe criados a partir de técnicas de detecção de mudança de conceito. Foi adicionada uma marcação indicativa da detecção de mudança de conceito feita por cada métrica, estas MFe possuem o sufixo *_drift_flag*. Além disso, os metadados utilizados por cada algoritmo, descritos na Seção 2.2, também são utilizados como meta atributos. Na coluna 'Tipo' da tabela, as métricas univariadas que possuem um asterisco (*) indicam algoritmos que utilizam somente a predição do modelo base, ou seja, apesar de se tratarem de métricas univariadas, somente uma MFe será gerada por janela.

A implementação do PSI para variáveis categóricas foi feita seguindo a fórmula original descrita na Equação 4.1, no entanto, ao invés de fazer a divisão da distribuição em percentis, como na abordagem original explicada na Seção 2.2, os dados foram agrupados a partir dos valores únicos discretos de cada atributo. Além disso, é gerada uma marcação (*psi_drift_flag*) para indicação de mudança de conceito caso o valor do PSI de qualquer atributo ultrapasse 0.2, limiar de segurança para o PSI definido por [65].

$$PSI = \sum((\%monitoramento - \%referencia) * ln(\frac{\%monitoramento}{\%referencia})) \tag{4.1}$$

No algoritmo implementado, o classificador utilizado foi o LightGBM [93]. Foi realizada uma otimização de hiperparâmetros para a profundidade máxima da DT, ou *max depth*, visto que a biblioteca utilizada [94] não apresenta limite máximo padrão para este

Nome	Descrição	Tipo
dc_drift_flag	Marcação de mudança de conceito do domain classifier	multivariada
dc_accuracy	Acurácia do DomainClassifier	multivariada
psi_drift_flag	Marcação de mudança de conceito do PSI	univariada
u_detect_drift_flag	Marcação de mudança de conceito do u-detect	multivariada
psi	Population Stability Index	univariada
centroid_distance	Distância dos grupos até o centróide (calculado no u-detect)	multivariada
sqsi_is_drift_flag	Marcação de mudança de conceito do sqsi-is	univariada*
score_ks	Kolmogorov smirnov dos scores (calculado no sqsi-is)	univariada*
md3_drift_flag	Marcação de mudança de conceito do md3	multivariada
svm_margin	Margem do SVM treinado no md3	multivariada
omv_pht_drift_flag	Marcação de mudança de conceito do omv-pht	univariada*
overlap_score	Overlap na distribuição dos scores (calculado no omv-pht)	univariada*

Tabela 4.9: Meta atributos de métricas de mudança de conceito

parâmetro, permitindo que a árvore cresça indefinidamente com os dados de treinamento e tornando o algoritmo propenso ao *overfitting*. A acurácia do classificador dá origem à meta variável *dc_accuracy* e os valores de acurácia abaixo de 0.2 ou acima de 0.8 são considerados *drift* e alteram o valor da MFe *dc_drift_flag*.

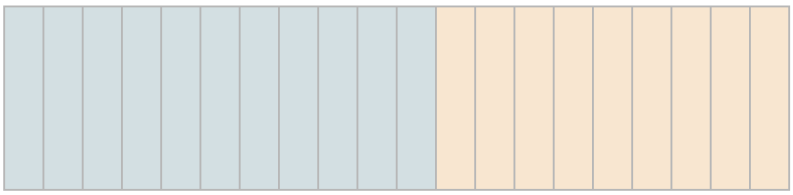
4.3 Arquitetura

Em [44], trabalho derivado de [42, 43], a arquitetura original de MtL foi dividida em dois estágios, o primeiro, denominado fase *offline*, consiste na otimização de hiperparâmetros, treino e validação dos algoritmos de nível base. Em seguida, é iniciada a fase *online*, que atua no fluxo de dados de forma dinâmica recomendando o melhor algoritmo para cada janela de dados. Apesar de esta divisão não ter sido feita de forma explícita no trabalho original de [42], ela facilita a compreensão do processo de *metaStream* e, portanto, será detalhada nesta seção.

4.3.1 Fase Offline

A fase *offline* se inicia quando um conjunto de dados históricos rotulados se acumula, este é então separado em duas partes ilustradas na Figura 4.3, a primeira fração de instâncias, representada em verde, é utilizada para efetuar o treinamento do modelo no nível base e, portanto, assume-se que estes dados não possuem mudança de conceito. Conforme detalhado na Seção 2.2, algumas métricas de detecção de mudança de conceito, que serão utilizadas nesta pesquisa como MFe, necessitam de um conjunto de dados de referência para efetuar a comparação com as janelas de avaliação, portanto, a primeira fração da base *offline* foi utilizada para este fim.

Em seguida as instâncias restantes, representadas em laranja na Figura 4.3, são utilizadas para validar o modelo no nível base, etapa com a qual se obtém o valor esperado de desempenho que será utilizado posteriormente como referência na geração do alerta de mudança de conceito. Por fim, estas últimas instâncias também são utilizadas no nível meta para construir a primeira meta base e treinar o primeiro meta modelo.



Nível Base	Treinamento do modelo base	Validação do modelo base
Nível meta	Usado como janela de referência para as métricas de mudança de conceito	Treinamento do meta modelo

Figura 4.3: Distribuição do conjunto de dados da fase *offline*

4.3.2 Fase Online

A fase *online* ocorre a partir de um fluxo de dados onde as informações chegam de forma contínua, conforme ilustrado na Figura 4.4. Na parte superior da imagem nota-se o fluxo de dados utilizado em nível base, onde as instâncias de cor verde representam os dados mais recentes cuja variável alvo ainda não é conhecida, supõe-se que exista atraso grande na sua chegada. A região laranja da imagem representa as instâncias rotuladas do fluxo de dados, estas são utilizadas para calcular o desempenho do modelo base e, então, rotular a meta base. A *baseline* dos experimentos é representada pelo desempenho real do modelo base na última janela temporal que possua variável alvo conhecida.

A parte inferior da imagem ilustra o funcionamento do algoritmo no nível meta, no qual as MFe são criadas de forma constante à medida em que novos dados chegam e

se acumulam em janelas de tamanho fixo η. Como somente MFe não supervisionadas foram incluídas, esta etapa, representada com a cor verde, não depende da chegada da variável alvo. A região alaranjada, por sua vez, representa as instâncias rotuladas da meta base. À medida em que a variável alvo chega no nível base, é possível fazer o cálculo do desempenho do modelo base, que constitui no meta rótulo, de modo que a metabase também pode ser rotulada. Por fim, quando um conjunto de dados rotulados na metabase é acumulado, o meta modelo é retreinado com as novas informações.

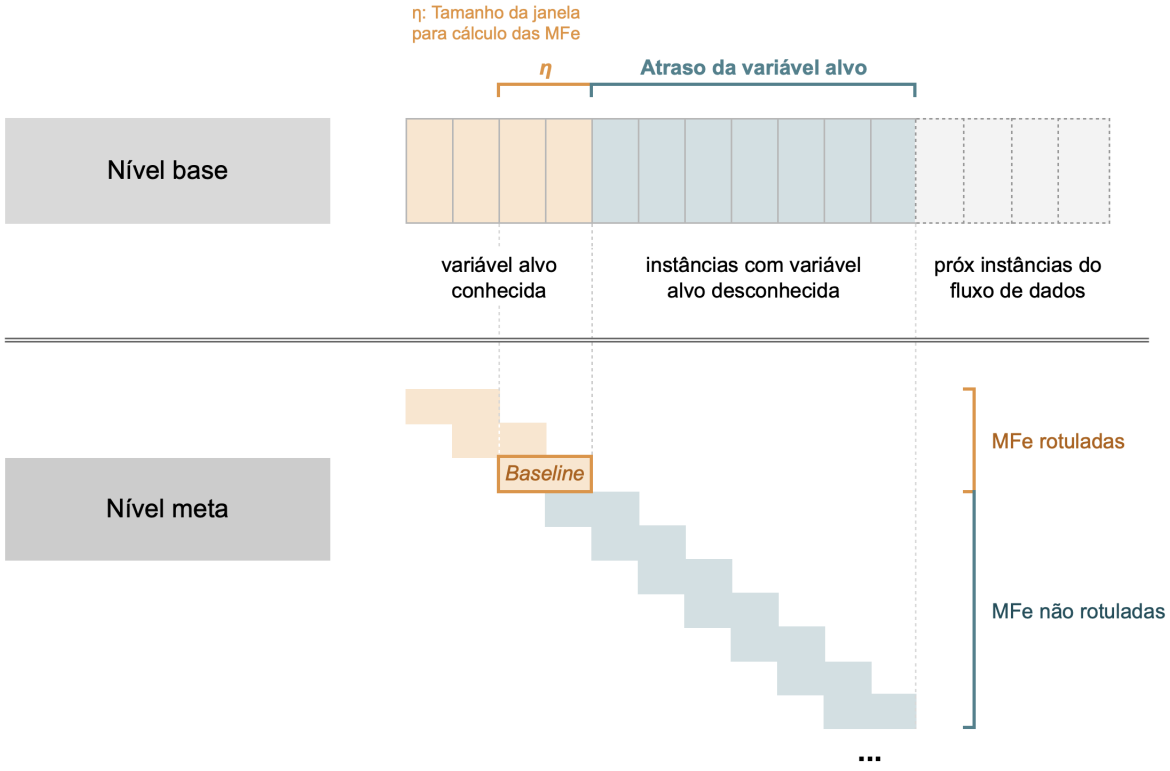


Figura 4.4: Fluxo da fase *online*

4.4 Parâmetros da arquitetura

4.4.1 Parâmetros do meta aprendizado

Durante a realização dos experimentos, os parâmetros referentes ao algoritmo de MtL foram configurados de forma diferente para cada base considerando o número total de instâncias da base. A Tabela 4.10 descreve os valores atribuídos a cada parâmetros.

Característica	Electricity, Powersupply and Rialto	Airlines	Bases sintéticas
Número de instâncias utilizadas no treinamento do modelo base	2.000	20.000	1.500
Número de instâncias utilizadas no treinamento do meta modelo	3.000	30.000	1.500
Tamanho da janela para o cálculo dos MFe	100	1.000	100
Tamanho do passo para início da próxima janela	30	300	30
Número de instâncias de atraso da variável alvo	500	2.000	300

Tabela 4.10: Parâmetros do meta aprendizado utilizados em cada base

4.4.2 Meta modelo

O meta regressor utilizado em todos os experimentos foi o *LightGBM* [95], a otimização de hiperparâmetros foi feita com Validação Cruzada de Séries Temporais, ou *Time Series Cross Validation*, [96], conforme ilustrado na Figura 4.5, em que o algoritmo é executado n vezes com conjuntos diferentes de treinamento e validação e cada divisão dos dados, ou *split*, é gerada de forma que o modelo só seja testado com dados futuros com relação ao treinamento.

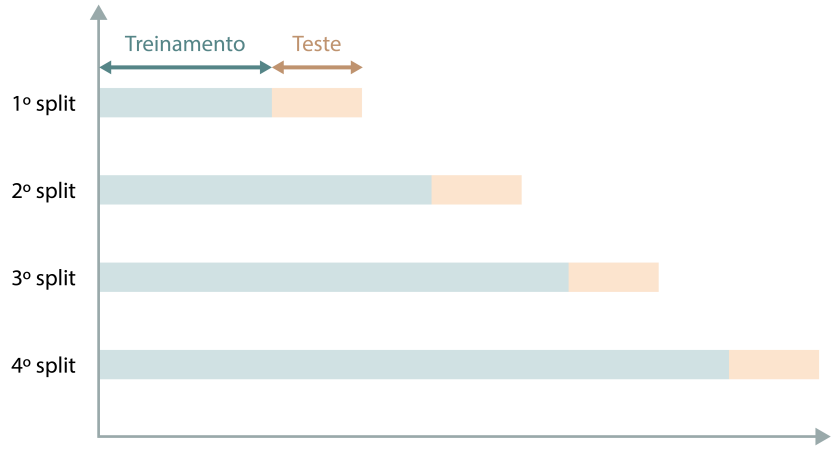


Figura 4.5: Time Series Cross Validation

O problema de MtL da forma que é construído implica na geração de meta bases com poucas amostras, visto que cada instância da meta base é gerada a partir de um janelamento da base original do problema, isso implica que o tamanho da meta base gerada vai ser uma fração do tamanho da base original, possuindo tamanho reduzido

de modo inversamente proporcional ao tamanho da janela. Além disso, várias MFe são criadas para cada atributo base, o que implica em um número crescente de atributos na meta base diretamente proporcional à quantidade de atributos da base original.

Estas características fazem com que a indução do meta modelo seja naturalmente propensa ao *overfitting* de modelos baseados em DT como o *LightGBM*, já que o número reduzido de instâncias dificulta a generalização do modelo e a quantidade elevada de variáveis explicativas aumenta a profundidade das árvores devido ao crescente número de divisões a serem feitas. Portanto, a seleção de hiperparâmetros foi feita considerando as recomendações da própria documentação do *LightGBM* [97] para tratamento de *overfitting*. Os parâmetros utilizados estão detalhados na tabela 4.11.

Nome do parâmetro	Descrição	Valor padrão	Intervalo testado
num_leaves	Número máximo de folhas	31	Entre 15 e 30
max_depth	Profundidade máxima	∞	Entre 3 e 12
max_bin	Número máximo de grupos	255	Entre 100 e 255
min_data_in_leaf	Mínimo de instâncias por folha	20	Entre 5 e 30
feature_fraction	Fração de atributos	1.0	Entre 0.2 e 1.0

Tabela 4.11: Hiperparâmetros utilizados na otimização do *LightGBM*

Conforme já mencionado, o elevado número de MFe gerado em problemas com muitos atributos no nível base faz com que o MtL seja altamente propenso ao *overfitting*, por isso o primeiro experimento desta pesquisa foi a validação da melhoria de desempenho do meta modelo caso fosse feita uma seleção de variáveis durante a fase *offline*. Para isso, o meta modelo foi treinado com um percentual de todas as MFe disponíveis para cada base, os atributos selecionados para cada execução foram aqueles que possuíam maior importância para o modelo após a etapa de otimização de hiperparâmetros. Ao final, foram testados valores entre 5% e 100% do total de MFe, em passos de 5%, e o percentual ótimo encontrado no conjunto de treinamento para cada base foi mantido nos experimentos subsequentes.

4.4.3 Alerta de mudança de conceito

Para determinar se uma mudança de conceito ocorreu, são extraídas duas variáveis da base de dados *offline*: a média e o desvio padrão do desempenho do modelo de nível base. Além disso, um parâmetro definido pelo usuário, denominado sensitividade, é levado em consideração. A sensitividade é uma escolha específica para cada problema e depende das suas particularidades.

O limiar de mudança de conceito é definido como a diferença entre a média (μ) e o desvio padrão (σ) multiplicado pela sensitividade (s), conforme descrito na Equação 4.2. Se o valor de desempenho predito pelo meta modelo estiver abaixo desse limiar, o algoritmo detecta a ocorrência de uma mudança de conceito [98]. Em outras palavras, a queda significativa do desempenho predito do modelo em relação ao esperado, provavelmente indica uma mudança no padrões dos dados.

$$thresh = \mu - \sigma * s \tag{4.2}$$

Durante a pesquisa, foram testados diferentes valores de sensitividade para avaliar a robustez e adaptabilidade do método de alerta de mudança de conceito. Essa abordagem permite aos pesquisadores ajustar o nível de sensitividade de acordo com as necessidades do problema em questão.

4.5 Avaliação

Nos experimentos iniciais da pesquisa, deseja-se validar a qualidade das predições de desempenho do meta modelo, de modo que foram utilizadas bases que não possuem mudança de conceito rotulada e a *baseline* dos experimentos é representada pelo desempenho real do modelo base na última janela temporal que possua variável alvo conhecida.

A avaliação inicial da qualidade das predições foi feita com uso do MSE, métrica usual para avaliação de problemas de regressão que informa o quão perto a linha de regressão está de determinado conjunto de pontos. Ele é definido pela Equação 4.3.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \tag{4.3}$$

Onde

- y_i: Índice i do *array* de valores observados
- $\widehat{y}_i$: Índice i do *array* de valores preditos

O MSE é uma métrica bastante difundida e fornece uma visão geral de como o regressor se compara com a *baseline* do problema, no entanto, não é possível visualizar a qualidade da regressão ao longo do fluxo de dados. Por isso, também foi implementado o gráfico do ganho acumulado, que consiste em dividir a meta base em janelas de tamanho fixo e calcular o MSE em cada uma para o meta regressor e para a *baseline*.

O ganho da abordagem proposta de MtL com relação à *baseline* é medido, em cada janela, conforme a Equação 4.4. Portanto, se o valor obtido de ganho for positivo significa que o erro da *baseline* foi maior naquela janela.

$$ganho = MSE_{baseline} - MSE_{MtL} \tag{4.4}$$

Quando o ganho é inserido em um gráfico acumulado, conforme ilustrado na Figura 4.6, temos visibilidade de em quais janelas a abordagem proposta foi superior, regiões crescentes da curva, ou inferior, regiões decrescentes da curva, com relação à *baseline*. Além disso, é possível interpretar que as regiões onde a área abaixo da curva de ganho acumulado é positiva representam os momentos em que o meta aprendizado se sobressaiu. Do mesmo modo, regiões da curva abaixo de zero indicam que a *baseline* seria uma proposta mais adequada ao problema em questão.

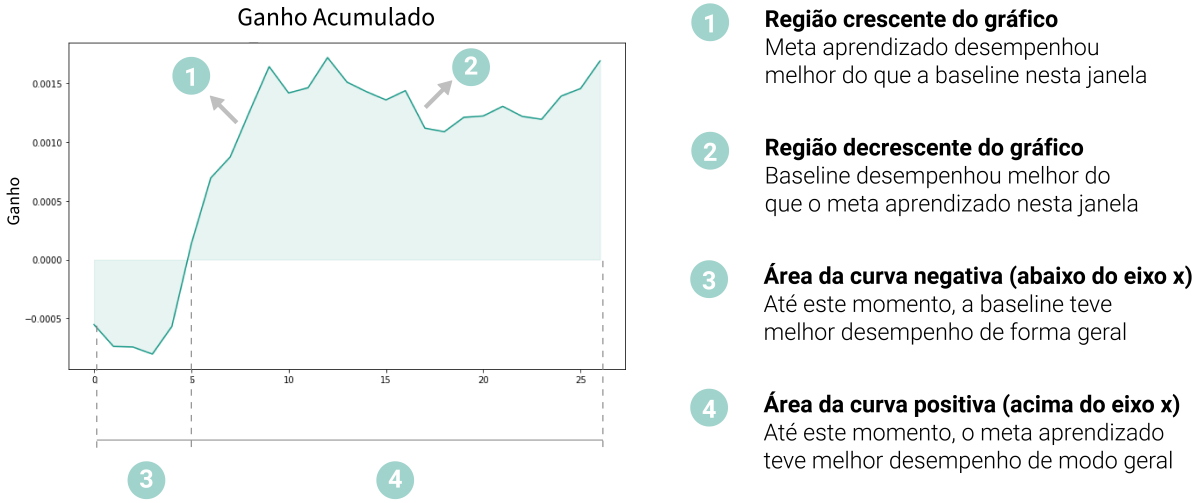


Figura 4.6: Ganho acumulado

O algoritmo de MtL envolve o retreino periódico do meta modelo e a geração contínua de dezenas de meta atributos, alguns dos quais possuem implementação complexa que pode envolver até o treinamento de novos modelos preditivos como os meta atributos de clusterização e algumas métricas de detecção de mudança de conceito, como o MD3 e o *Domain Classifier*. Isso faz com que o algoritmo seja extremamente custoso em termos de duração e recursos computacionais, por isso, identificar o tempo de execução se mostra uma análise crucial para validar se o ganho de desempenho compensa o aumento na latência em alguns contextos. Por este motivo, neste trabalho o tempo decorrido durante o processo foi medido como forma de análise dos resultados do meta modelo implementado.

A importância das variáveis para o meta regressor é calculada como forma de entender a contribuição individual dos meta atributos propostas nesta pesquisa já que, conforme mencionado anteriormente, a inclusão de métricas de detecção de mudança de conceito e

de meta atributos calculados a partir da predição do modelo base fazem parte do escopo que se quer validar neste trabalho.

No experimento final da pesquisa, a qualidade dos alertas de mudança de conceito gerados pelos algoritmos foi avaliada, comparando-os com as métricas de mudança de conceito pré-existentes na literatura, detalhadas na Seção 2.2. Essas métricas serviram como uma *baseline* para a avaliação dos algoritmos, permitindo determinar se os alertas gerados superam ou não as abordagens estabelecidas anteriormente.

Aqui, foram utilizadas as bases de dados rotuladas, definidas em 4.2.3, se tratando, portanto, de um problema de classificação supervisionado, exclusivamente para fins experimentais. Para a avaliação do desempenho destes classificadores, foram adotadas medidas de classificação comumente utilizadas, como f1-score, recall, precisão e gmean.

A análise estatística foi conduzida empregando o teste de *Friedman*, avaliação adequada para comparação de múltiplos classificadores em diferentes bases de dados, conforme [99]. O teste consiste em criar um *ranking* dos melhores classificadores para cada base de dados e, em seguida, calcular o *ranking* médio de todas as bases. Por fim, um teste estatístico de *Friedman* é realizado, visando avaliar se o p valor é menor do que 0.05, neste caso podemos rejeitar a hipótese nula de que pelo menos alguns classificadores apresenta desempenho significativamente diferente.

4.6 Implementação

Os experimentos desta pesquisa foram construídos utilizando a versão 3.8.5 da linguagem Python e o código fonte está disponível em repositório público do Github [100].

O cálculo das medidas de caracterização com implementação simples, como as MFe estatísticas e diversas, foram realizados com auxílio dos pacotes *scikit-learn* [94], *numpy* [101] e *scipy* [102], enquanto as medidas de clusterização foram calculadas com auxílio do *kMeans* do *scikit-learn* com implementação do método do cotovelo proveniente do pacote *KneeLocator* [103]. Todos os modelos de nível base utilizados são oriundos do *scikit-learn*, enquanto o meta modelo utiliza a implementação do *LightGBM* [95] com otimização de hiperparâmetros realizada por meio do pacote *Optuna* [104].

Capítulo 5

Resultados

Neste capítulo serão apresentados os resultados obtidos com a metodologia proposta. O capítulo está dividido da seguinte maneira: inicialmente, na Seção 5.1 são apresentados os resultados obtidos na etapa de seleção de MFe cujos parâmetros são utilizados ao longo dos demais experimentos. Em seguida, na Seção 5.2 é feita a comparação do MSE entre o algoritmo proposto, a abordagem tradicional de MtL e a *baseline*. A Seção 5.3 avalia a importância das MFe para o modelo com o intuito de entender a contribuição real das MFe propostas nesta pesquisa. Por fim, a Seção 5.4 analisa o tempo de execução do algoritmo em cada contexto e a Seção 5.5 detalha os resultados dos experimentos relacionados aos alertas de mudança de conceito.

5.1 Seleção de meta atributos

O elevado número de MFe gerado faz com que o aprendizado do meta modelo seja naturalmente propenso ao *overfitting*, de modo que o primeiro experimento desta pesquisa consistiu na validação da melhoria de desempenho do meta modelo caso fosse feita uma seleção de atributos.

Os resultados podem ser conferidos na Figura 5.1, onde o eixo x representa o número de MFe utilizadas no treinamento do meta modelo e o eixo y representa o MSE obtido nos dados de treinamento, calculado a partir da média de todos os modelos de nível base. Os sub gráficos ilustram as bases de dados avaliadas e as linhas representam os possíveis meta rótulos cuja descrição está representada na legenda. Por fim, o valor do erro foi normalizado para facilitar a visualização dos pontos de mínimo e máximo para todos os meta rótulos.

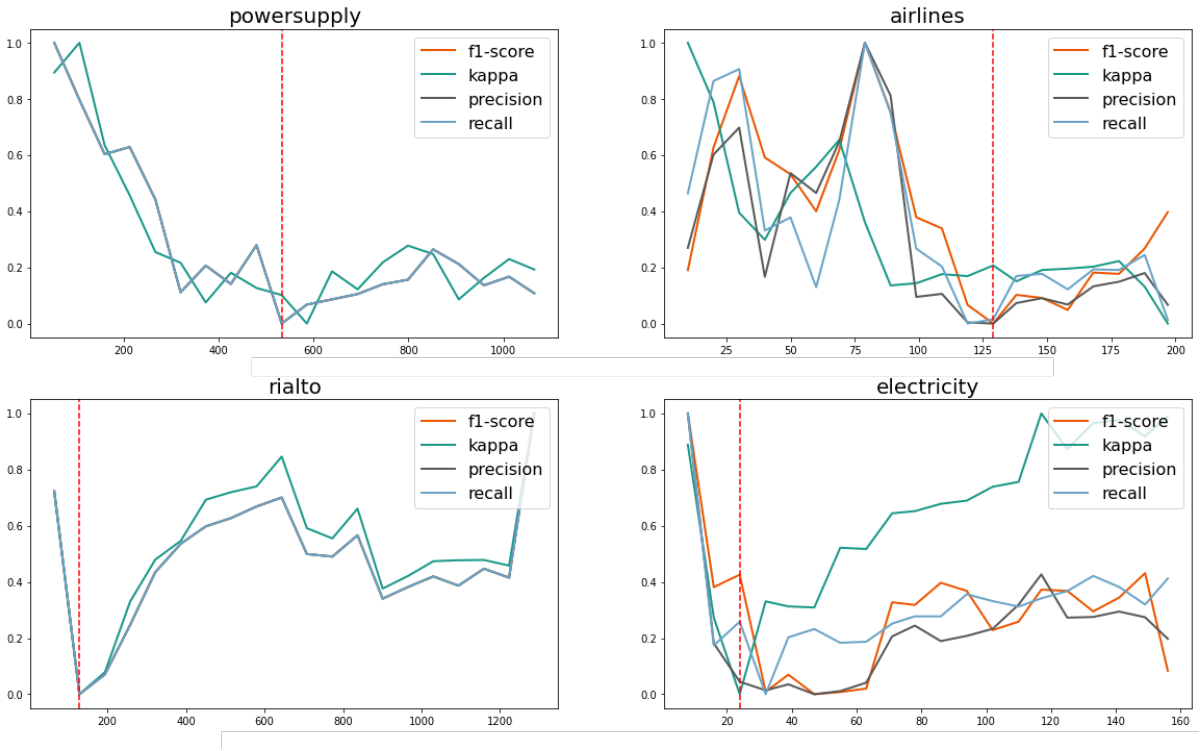


Figura 5.1: Resultado da seleção de meta atributos

É possível notar que o MSE segue uma tendência similar para todos os meta rótulos utilizados dentro da mesma base de dados, exceto pelo meta rótulo *Kappa* na base de eletricidade, no entanto, ainda assim o valor mínimo de MSE para este caso é similar aos demais meta rótulos da base. Dada esta convergência, o número de MFe associado ao menor MSE médio foi escolhido como parâmetro de seleção de atributos para os experimentos subsequentes, este valor está representado através da linha vermelha vertical e pode ser conferido na Tabela 5.1. É importante notar que o número máximo de MFe varia para cada base de dados visto que estas possuem quantidades distintas de atributos no nível base.

Base de dados	Número de MFe	Proporção de MFe
Powersupply	533	50%
Airlines	129	65%
Rialto	129	10%
Electricity	24	15%

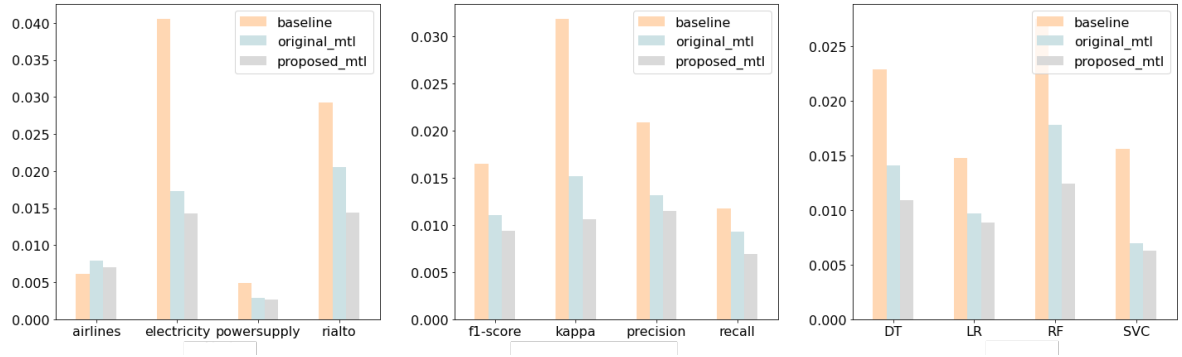
Tabela 5.1: Número ótimo de MFe para cada base de dados

5.2 Ganho do meta aprendizado

Para validar a hipótese de pesquisa de que um meta regressor é capaz de prever o desempenho do modelo base quando existe atraso grande na chegada da variável alvo, nesta seção será analisado o desempenho do MtL com relação à *baseline*. Além disso, também deseja-se validar a hipótese de que o uso de MFe calculados a partir da predição do modelo base e das métricas de detecção de mudança de conceito melhoram o desempenho do meta modelo. Por isso, foram treinados dois meta modelos em cada experimento: o primeiro será chamado de *MtL proposto* e utiliza todas as MFe descritas na Seção 4.2.4. O segundo foi treinado sem a inclusão das novas MFe propostas, como os atributos baseados na predição do modelo base e nas medidas de detecção de mudança de conceito, esta versão será chamada de *MtL original* ao longo desta Seção.

Inicialmente, calculou-se o MSE médio para todas as janelas considerando três modos de agrupar as execuções: por base de dados, ilustrado na Figura 5.2a, por meta rótulo, ilustrado na Figura 5.2b e por modelo no nível base, ilustrado na Figura 5.2c. É possível notar que o erro do algoritmo de MtL proposto é sempre menor do que o do original, além disso, ambos os algoritmos de MtL apresentam erros inferiores aos da *baseline* em todos os casos exceto na base de dados *Airlines*.

Para fechar o escopo desta análise, o restante desta seção abordará os resultados obtidos com as bases *Electricity* e *Airlines* nas quais o MtL apresentou o melhor e pior desempenho, respectivamente. Além disso, análises mais detalhadas serão feitas com o meta rótulo *Kappa*, visto que esta é uma métrica tradicional da literatura de fluxo de dados [1] e foi aquela em que o MtL apresentou melhor desempenho com relação à *baseline*.



(a) MSE médio por base de dado (b) MSE médio por meta rótulo (c) MSE médio por modelo base

Figura 5.2: MSE médio do algoritmo proposto de MtL, MtL original e baseline

5.2.1 Resultados da base *Airlines*

A Figura 5.3 representa o gráfico de ganho acumulado da abordagem de MtL proposta com relação à *baseline* com a base *Airlines*. O eixo x representa a janela utilizada para o cálculo do MSE e o eixo y representa o ganho acumulado da janela; cada *sub* gráfico se refere à um modelo base e as linhas representam os possíveis meta rótulos cuja descrição está detalhada na legenda.

É possível notar que a base de dados, bem como o modelo base, são fatores importantes para definir se a abordagem de MtL é adequada ao problema em questão, visto que bons resultados de ganho foram obtidos utilizando SVM e DT como modelos base, enquanto o RF e LR apresentaram resultados ruins, neste último, o MtL se mostrou inferior à *baseline* para todos os meta rótulos na maioria das janelas avaliadas.

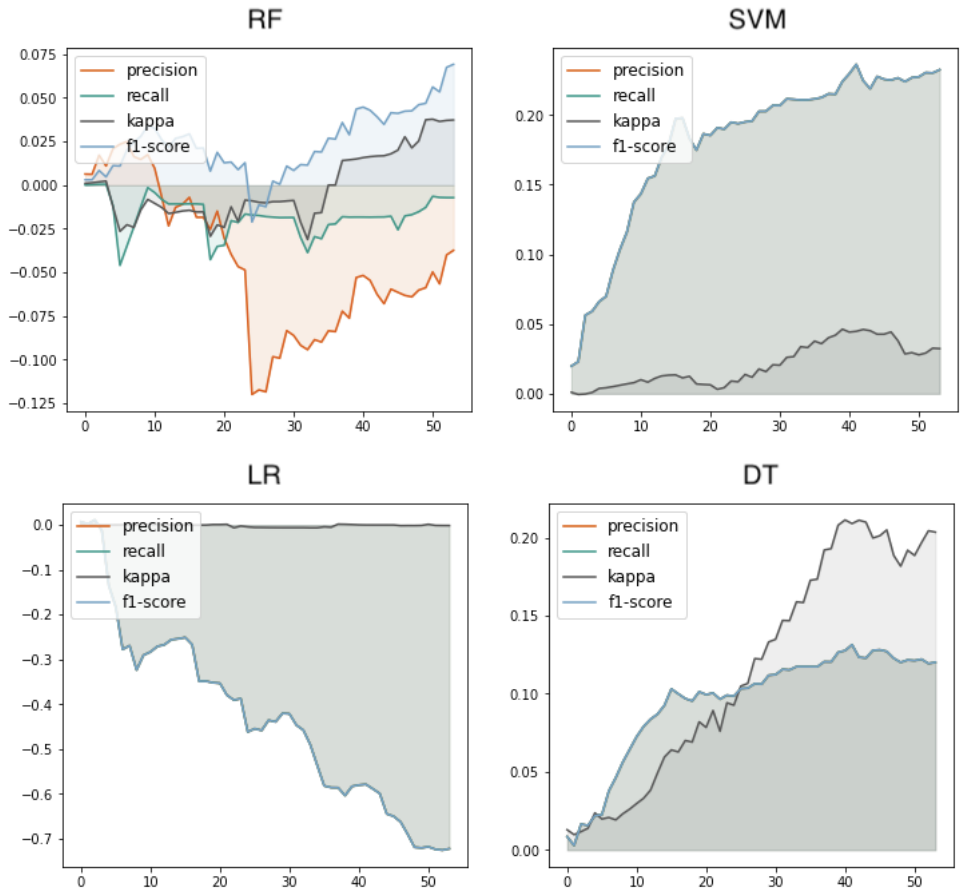


Figura 5.3: Ganho acumulado do algoritmo de MtL proposto na base de dados *Airlines*

A Figura 5.4 ilustra o gráfico de ganho acumulado da base *Airlines* com o meta rótulo *Kappa*. O eixo x representa a janela utilizada para o cálculo do MSE e o eixo y representa o ganho acumulado da janela; cada *sub* gráfico se refere à um modelo base. Nesta Figura,

a *baseline* está sendo comparada com o algoritmo proposto, com o MtL original e com um "regressor ideal" que apresenta MSE nulo em todas as janelas.

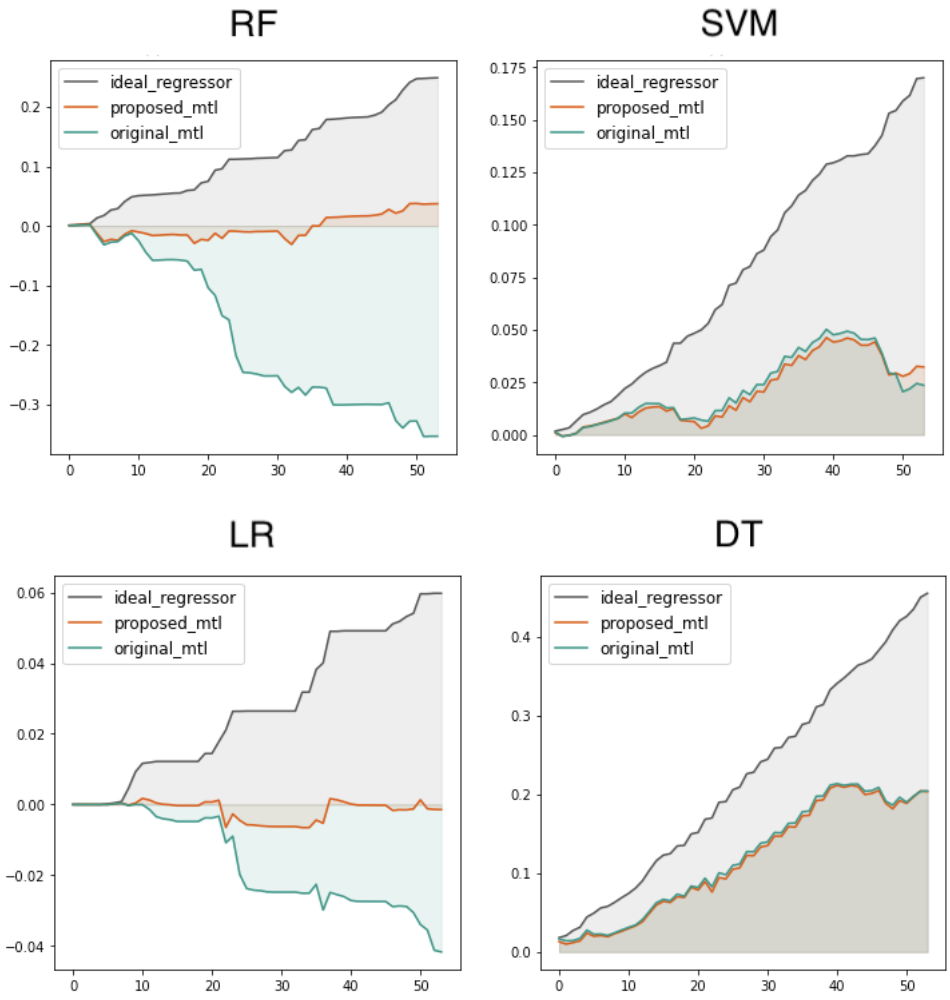


Figura 5.4: Ganho acumulado do algoritmo de MtL proposto na base de dados *Airlines* com meta rótulo *Kappa*

Nota-se que os desempenhos do MtL proposto e original são bem similares quando SVM e DT são utilizados como modelo base, indicando que as novas MFe propostas não apresentaram grandes ganhos para estes casos. Ainda assim, o desempenho de ambos os modelos de MtL foi superior ao da *baseline*.

Por outro lado, quando RF e LR são utilizados como modelo base, o MtL proposto apresentou desempenho bem superior devido à utilização das novas MFe. Ainda assim, ambos obtiveram resultados ruins com relação à *baseline* reiterando que o MtL não é indicado para estes casos.

5.2.2 Resultados da base *Electricity*

A Figura 5.5 ilustra o ganho acumulado da abordagem de MtL proposta com relação à *baseline* com a base *Electricity* para cada modelo base analisado. O eixo x representa a janela utilizada para o cálculo do MSE e o eixo y representa o ganho acumulado da janela. Cada *sub* gráfico se refere à um modelo base e as linhas representam os possíveis meta rótulos cuja descrição está detalhada na legenda.

Nesta base, o MtL obteve desempenho satisfatório para todos os modelos de nível base, indicando que a técnica não é inadequada para os modelos RF e LR e que o resultado ruim foi uma particularidade da base *Airlines*. Além disso, o gráfico de ganho acumulado é majoritariamente crescente, indicando que o MtL é mais adequado do que a *baseline* na maior parte das janelas avaliadas. Para esta base, todos os meta rótulos apresentaram ganho satisfatório com relação à *baseline* com destaque para a métrica *Kappa*.

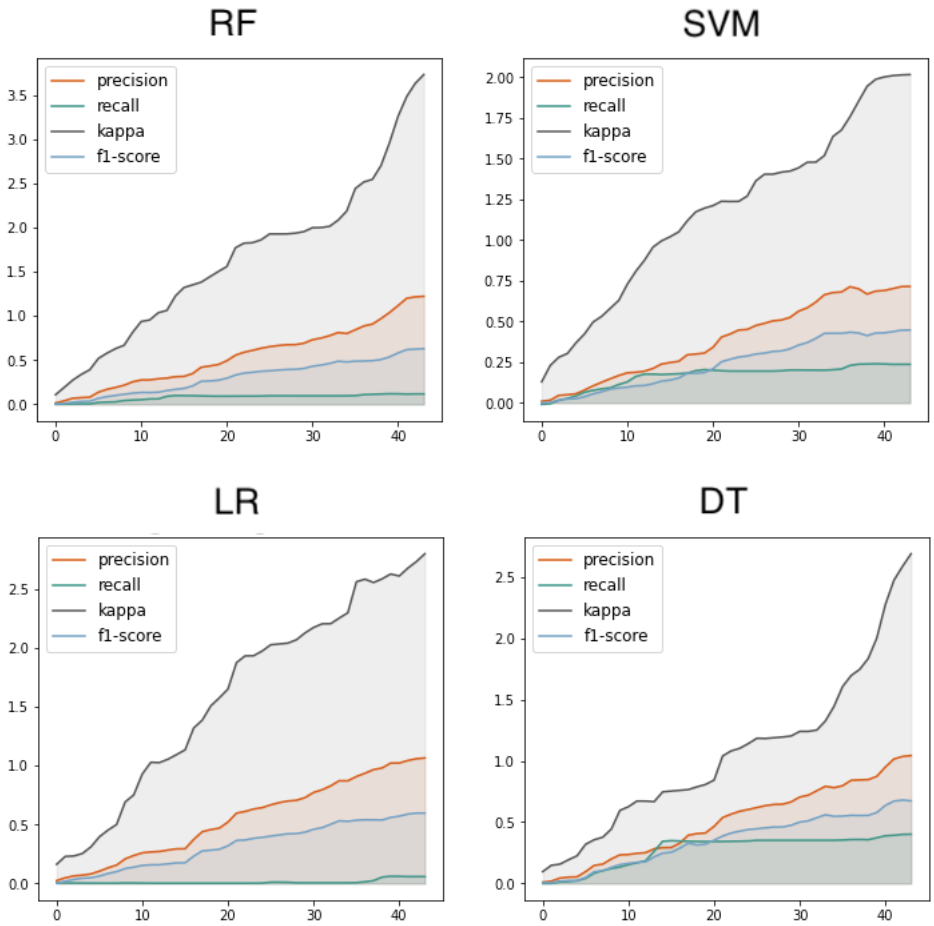


Figura 5.5: Ganho acumulado do algoritmo de MtL proposto na base de dados *Electricity*

A Figura 5.6 ilustra o gráfico de ganho acumulado da base *Electricity* com o meta

rótulo *Kappa*, aqui a *baseline* está sendo comparada com o algoritmo proposto, com o MtL original e com um "regressor ideal" que apresenta MSE nulo em todas as janelas. Neste caso, é possível notar que o desempenho do MtL proposto e do original foram excelentes atingindo, em média, 76% e 66% do desempenho do regressor ideal respectivamente. Desse modo, conclui-se que as novas MFe propostas foram responsáveis por um aumento médio de 10% no ganho para esta base.

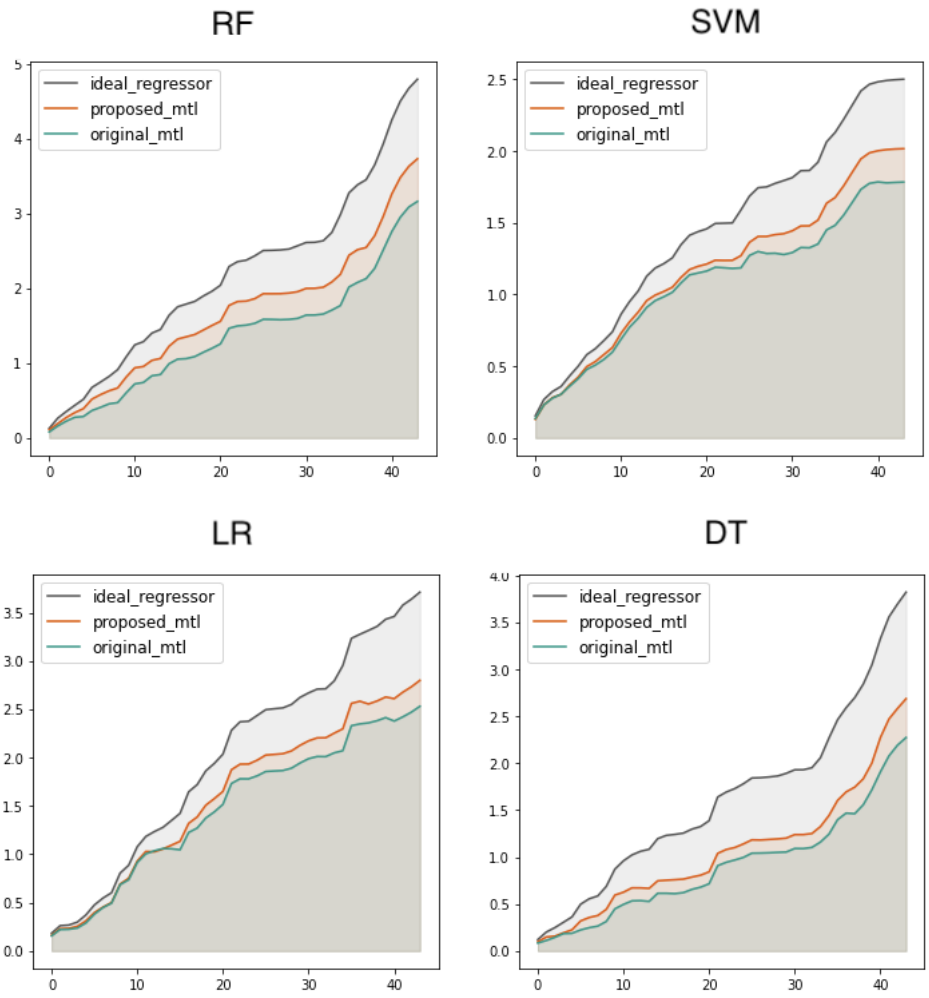


Figura 5.6: Ganho acumulado do MtL na base de dados *Electricity* com meta rótulo *Kappa*

5.3 Importância das variáveis

Esta Seção apresenta uma análise da importância das variáveis para entender a contribuição das novas MFe propostas e, assim, validar a hipótese de que o uso de MFe calculados a partir da predição do modelo base e das métricas de detecção de mudança de conceito

melhoram o desempenho do meta modelo. Para tal, o modelo base utilizado foi RF em que o MtL proposto apresentou, no geral, maior ganho se comparado ao MtL original.

As Figuras 5.7 e 5.8 ilustram a lista dos 15 atributos mais importantes para o meta modelo nas bases *Electricity* e *Airlines*, respectivamente. As barras azuis ilustram as MFe pré existentes na literatura de MtL e as alaranjadas representam as novas MFe propostas nesta pesquisa. Os círculos vermelhos indicam as MFe propostas construídas a partir das métricas de detecção de mudança de conceito.

Para a base *Electricity*, Figura 5.7, nota-se que mais da metade dos atributos mais importantes se referem às novas MFe, mostrando que as medidas de caracterização propostas são, de fato, capazes de aumentar o poder preditivo do meta modelo. Nesta execução, destacam-se as medidas de detecção de mudança de conceito representando 33% dos atributos mais importantes.

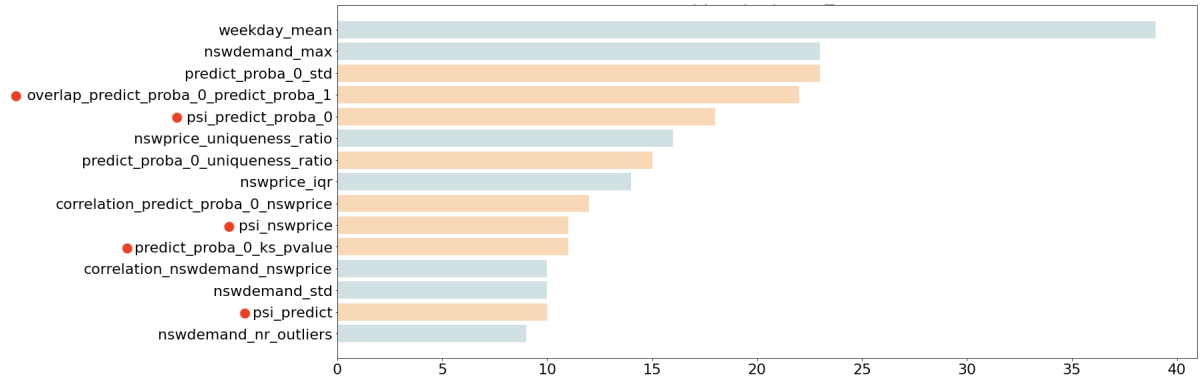


Figura 5.7: Importância das variáveis na base *Electricity* com meta rótulo *Kappa* e modelo base RF

Para a base *Airlines*, Figura 5.8, a contribuição das novas MFe foi ainda mais expressiva, representando cerca de 67% dos atributos mais importantes para o meta modelo. Esta análise reforça os resultados obtidos na Seção 5.2.1 onde foi constatado o ganho elevado do MtL proposto com relação ao original para a base *Airlines* e modelo RF.

De modo oposto ao observado na base *Electricity*, nesta execução apenas uma MFe se refere às medidas de detecção de mudança de conceito, de modo que a maior parte dos atributos importantes foram construídos com base na predição do modelo de nível base e no desempenho da última janela com variável alvo conhecida.

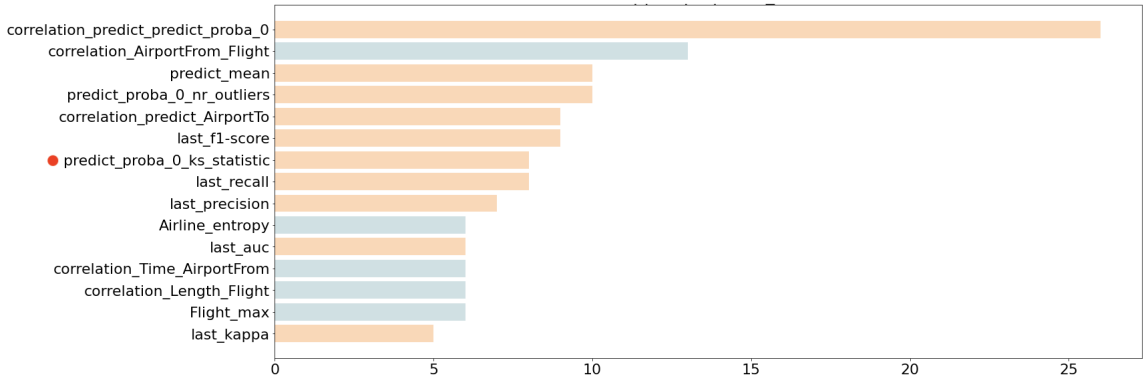


Figura 5.8: Importância das variáveis na base *Airlines* com meta rótulo *Kappa* e modelo base RF

5.4 Tempo de execução

O MtL por si só é uma técnica computacionalmente custosa e está sendo aplicado em problemas de fluxo de dados onde o tempo de processamento é restrito, portanto, além de avaliar o ganho produzido pelas MFe propostas é importante ponderar se o mesmo compensa o aumento na latência gerado pelo cálculo dos novos atributos.

A Figura 5.9 ilustra o tempo de execução médio para cada segmento de MFe, em laranja, e o somatório das importâncias por grupo de MFe, em azul. No eixo y observa-se o nome do grupo e no eixo x o percentual do tempo de execução total e o percentual do grupo na importância total. Esta importância foi calculada a partir da média de todos experimentos, incluindo as diferentes bases, modelos no nível base e meta rótulos. Os círculos vermelhos à esquerda do gráfico destacam os grupos referentes às novas MFe propostas nesta pesquisa

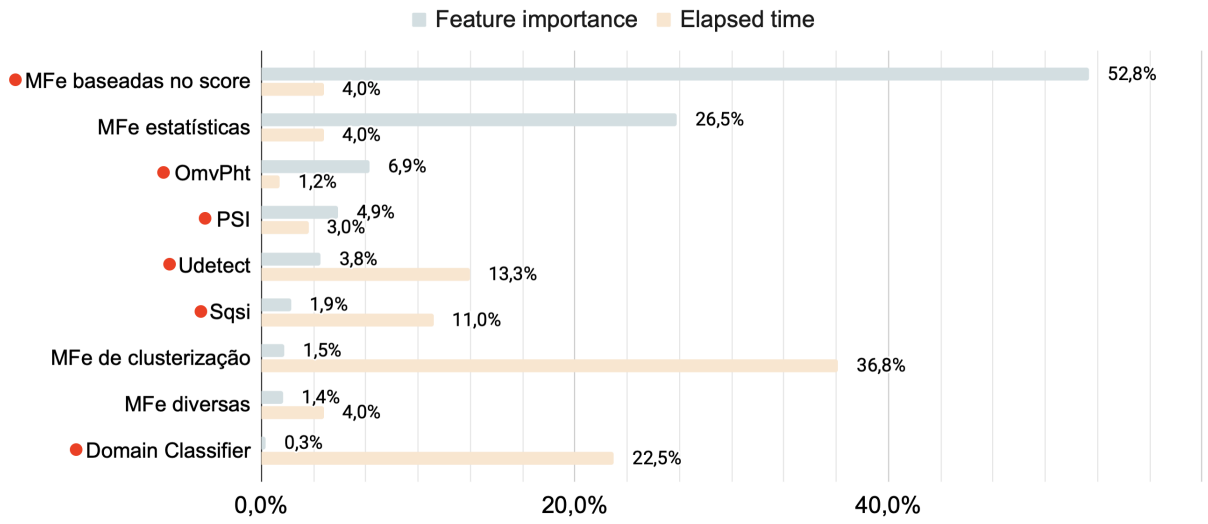


Figura 5.9: Proporção do tempo de execução e importância média de cada grupo de MFe por base de dados

É possível notar que a métrica de detecção de mudança de conceito *Domain Classifier* e o grupo de MFe de clusterização, pré existente na literatura, correspondem a mais da metade do tempo de processamento total. Além disso, as medidas de detecção de mudança de conceito PSI e OmvPht representam os grupos com menor tempo de execução.

Nota-se que as MFe de clusterização correspondem a somente 1.5% da importância total, apesar do longo tempo de execução, enquanto o *Domain Classifier* teve impacto praticamente nulo e levou 22.5% do tempo de execução, indicando que não é uma métrica adequada para uso no MtL.

Por outro lado, os demais grupos apresentaram resultados satisfatórios com destaque para as MFe estatísticas e métricas baseadas na predição do modelo base, representadas em azul no gráfico, que correspondem a aproximadamente 80% da importância e 12% do tempo de processamento total. Além dessas, as medidas de detecção de mudança de conceito OmvPht e PSI também apresentaram resultados interessantes com 7% e 5% de importância e somente 1.2% e 3% do tempo de execução respectivamente.

É importante notar que o volume total de MFe cada grupo influencia de forma direta a importância total do grupo, neste caso, dois grupos se destacam em importância e também em número de atributos utilizados. O primeiro grupo se refere às MFe estatísticas cujo volume aumenta de modo proporcional ao número de atributos da base, o segundo se refere às medidas de caracterização baseadas na predição do modelo base que crescem de modo proporcional à quantidade de classes do problema, visto que cada classe corresponde à uma distribuição que gera múltiplas outras MFe.

5.5 Alerta de mudança de conceito

Para validar a hipótese de que o MtL é adequado como nova métrica de detecção de mudança de conceito, o algoritmo de geração do alerta foi executado considerando todos os modelos de nível base, meta rótulos e bases de dados sintéticas com mudança de conceito rotulada. A média de desempenho resultante, obtida para cada combinação de modelo base, MFe e base de dados, é apresentada na Figura 5.10.

Na Figura, as linhas representam métricas de avaliação dos alertas e linhas representam os algoritmos utilizados para a geração do alerta de mudança de conceito. Entre esses algoritmos estão o MtL proposto, com diferentes valores de sensitividade, e as técnicas de detecção de mudança de conceito da literatura, que são utilizadas como *baselines*. Através deste mapa de calor, é possível visualizar rapidamente o desempenho relativo de cada algoritmo em relação às métricas de desempenho, com cores claras indicando um desempenho inferior.

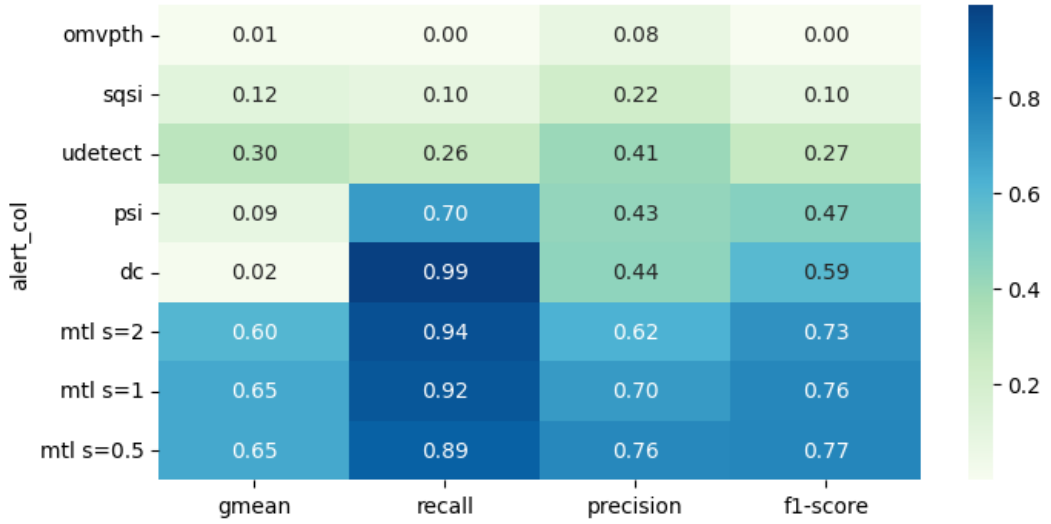


Figura 5.10: Resultado médio do alerta de mudança de conceito para cada métrica

É possível observar que a alteração do valor de sensitividade apresenta impacto direto no *recall* e precisão de forma inversa. Ao aumentar a sensitividade para detecção de mudanças, o algoritmo consegue identificar mais mudanças conceitos ao longo do tempo. No entanto, isto resulta em uma maior taxa de falsos positivos. Por outro lado, ao diminuir a sensitividade, o algoritmo torna-se mais seletivo, reduzindo os falsos positivos, mas correndo o risco de perder a detecção de mudanças relevantes, o que afeta o *recall*. Portanto, é necessário encontrar um equilíbrio entre a precisão e o *recall*, considerando a sensitividade adequada para o contexto específico do problema de mudança de conceito.

Em relação às métricas avaliadas, a única que demonstrou desempenho superior à abordagem de MtL foi o *recall* com o uso do *Domain Classifier*. No entanto, esse resultado

deve-se ao fato de esta métrica ser altamente sensível a mudanças, ativando o alerta de mudança de conceito com maior facilidade, o que resulta em um maior número de falsos positivos e uma quantidade menor de falsos negativos.

5.5.1 Teste de Friedman

O teste de *Friedman* foi executado para todas as combinações de modelos de nível base, meta rótulos e bases de dados sintéticas com mudança de conceito rotulada. O resultado para cada combinação está descrito na Tabela 5.2, valores similares de p-valor foram agrupados para facilitar a visualização da tabela.

Observa-se que todos os meta rótulos apresentaram p valor inferior a 0.05. Isso indica que os resultados da Figura 5.10 são válidos para quaisquer métricas de desempenho de nível base que se queira utilizar na detacção da mudança de conceito.

Além disso, podemos afirmar que o MtL demonstrou um desempenho superior às métricas de detecção de mudança de conceito apenas para os modelos de nível base RF e DT. Nos demais casos, a hipótese nula não foi rejeitada, o que implica que não podemos garantir que o MtL seja mais eficaz do que as métricas tradicionais de detecção de mudança de conceito. Deste modo, a escolha do modelo base pode ter um impacto relevante no desempenho do alerta de mudança de conceito

No que diz respeito às métricas de avaliação do alerta de mudança de conceito, verificou-se que apenas as métricas F1 e precisão apresentaram p valor inferior a 0,05.

5.5.2 Tipos de mudança de conceito

Por fim, analisa-se um caso específico em que o p valor é confiável para compreender a relação entre a qualidade do alerta de mudança de conceito e diferentes variáveis do problema, como o tipo de mudança de conceito, a presença de ruído, o balanceamento dos dados e a base de dados utilizada. Foram fixados os seguintes parâmetros: RF como modelo de nível base com meta rótulo *Kappa* e *F1-score* como métrica de avaliação do alerta de mudança de conceito.

A Figura 5.11 ilustra o mapa de calor com os resultados, onde as colunas representam as diferentes bases de dados utilizadas e linhas representam os algoritmos utilizados para a geração do alerta de mudança de conceito. Entre esses algoritmos estão o MtL proposto, com diferentes valores de sensitividade, e as técnicas de detecção de mudança de conceito da literatura, que são utilizadas como *baselines*. Através deste mapa de calor, é possível visualizar rapidamente o desempenho relativo de cada algoritmo em relação às diferentes bases de dados, com cores claras indicando um valor de F1 inferior. Além disso, no mapa de calor foram adicionados dendogramas, estruturas em forma de árvore que mostram

a hierarquia e a similaridade entre as linhas e colunas dos dados, ajudando a identificar agrupamentos ou padrões.

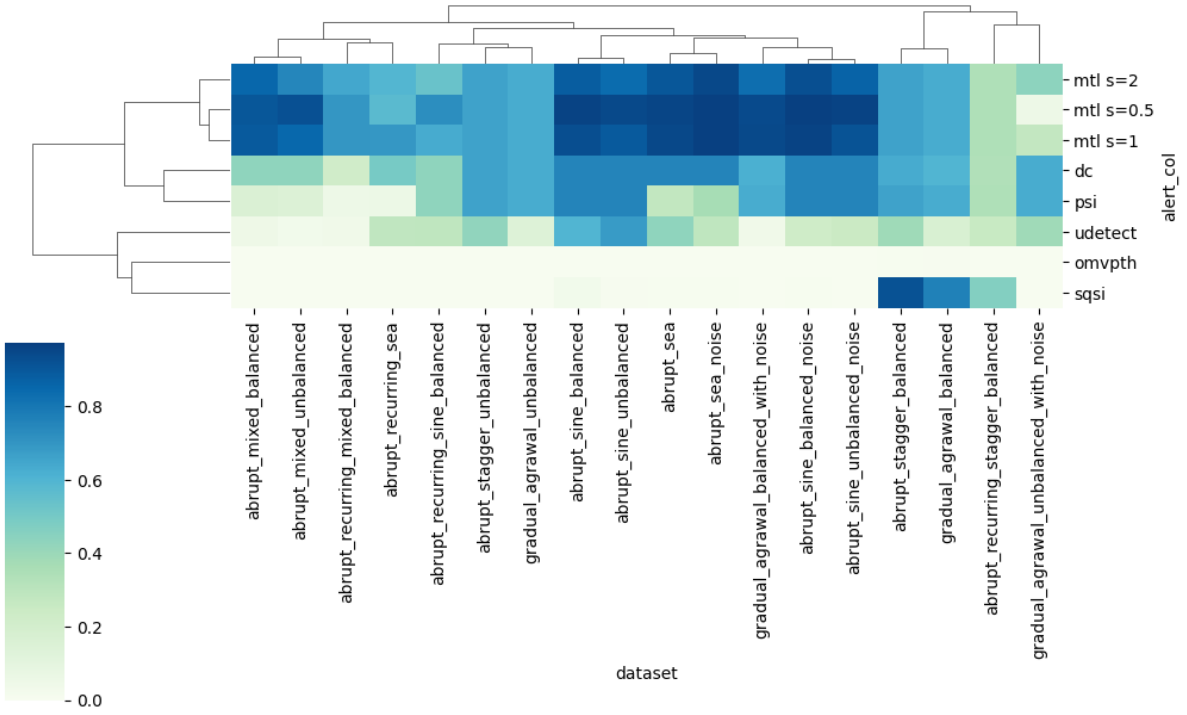


Figura 5.11: Resultado do alerta de mudança de conceito para o modelo base RF, meta rótulo Kappa e métrica de análise de mudança de conceito F1

Os resultados obtidos revelam que o desempenho do algoritmo é insatisfatório para todas as bases geradas pelo gerador *Stagger*, que é o único gerador com função de classificação *booleana*. Além disso, observou-se que a presença de ruído ou dados desbalanceados não parece interferir significativamente nos resultados, uma vez que foram observados valores altos e baixos de F1 para ambos os casos.

No entanto, as bases com mudança de conceito recorrente apresentaram resultados inferiores em relação às demais. Destaca-se que a combinação da mudança de conceito recorrente com o gerador *Stagger* resultou nas piores métricas de desempenho, indicando uma relação negativa entre essas duas características.

Através do dendrograma, é possível observar a relação entre as medidas de detecção de mudança de conceito. Podemos notar que os algoritmos propostos de MtL com diferentes níveis de sensitividade foram agrupados. Além disso, também foram agrupados o *Domain Classifier* e o PSI, ambas métricas com implementação voltadas para a detecção de mudança nas variáveis explicativas do modelo de nível base. Por fim, o SqsiIs e o OmvPht também foram agrupados, ambas são medidas baseadas na predição do modelo de nível base.

Métrica	P-valor	Modelo base	Meta rótulos	Rejeita a hipótese nula
f1-score	0.0147	RandomForest	precision, recall, f1-score	Sim
precision	0.0156	RandomForest	precision, recall, f1-score	Sim
precision	0.0242	DecisionTree	kappa	Sim
f1-score	0.0248	RandomForest	kappa	Sim
precision	0.0252	RandomForest	kappa	Sim
f1-score	0.0277	DecisionTree	kappa	Sim
f1-score	0.0280	DecisionTree	precision, recall, f1-score	Sim
precision	0.0339	LogisticRegression	precision, recall, f1-score	Sim
precision	0.0401	DecisionTree	precision, recall, f1-score	Sim
precision	0.0455	SVC	precision, recall, f1-score	Sim
precision	0.0668	LogisticRegression	kappa	Não
f1-score	0.0801	SVC	precision, recall, f1-score	Não
precision	0.0951	SVC	kappa	Não
recall	0.1345	LogisticRegression	kappa	Não
f1-score	0.1544	SVC	kappa	Não
f1-score	0.1776	LogisticRegression	precision, recall, f1-score	Não
recall	0.1854	SVC	kappa	Não
recall	0.2375	LogisticRegression	precision, recall, f1-score	Não
f1-score	0.2612	LogisticRegression	kappa	Não
recall	0.2971	DecisionTree	precision, recall, f1-score	Não
recall	0.3195	SVC	precision, recall, f1-score	Não
recall	0.3420	DecisionTree	kappa	Não
recall	0.5656	RandomForest	precision, recall, f1-score	Não
recall	0.5835	RandomForest	kappa	Não
gmean	0.5966	DecisionTree	kappa	Não
gmean	0.6506	SVC	precision, recall, f1-score	Não
gmean	0.7456	DecisionTree	precision, recall, f1-score	Não
gmean	0.8764	SVC	kappa	Não
gmean	0.8889	RandomForest	kappa	Não
gmean	0.9031	LogisticRegression	precision, recall, f1-score	Não
gmean	0.9120	RandomForest	precision, recall, f1-score	Não
gmean	0.9790	LogisticRegression	kappa	Não

Tabela 5.2: Resultados do teste de Friedman para cada combinação de modelo de nível base, MFe e métrica de avaliação do alerta de mudança de conceito.

Capítulo 6

Conclusão

Ao longo deste estudo, o objetivo principal foi validar a capacidade de um meta regressor em prever o desempenho de modelos de nível base com diferentes vieses, além de gerar alertas de mudança de conceito com base nessas previsões. A hipótese era que os alertas gerados com MtL teriam desempenho superior em relação às medidas de detecção de mudança de conceito existentes na literatura. Para isso, buscamos identificar as condições em que o MtL se mostraria mais eficaz e aplicável, propondo modificações na arquitetura original e explorando o uso de MFe não supervisionadas da literatura em conjunto com medidas de detecção de mudança de conceito.

Os experimentos conduzidos neste estudo destacam a capacidade do MtL em prever o desempenho de modelos de nível base, independentemente de seus viéses, e em bases de dados de fluxo contínuo com mudanças de conceito. A aplicação do algoritmo proposto demonstrou redução média de 12.8% no MSE em comparação com o MtL tradicional e redução de 38% em relação à *baseline*. Estes resultados indicam que o MtL proposto se destaca em relação aos métodos tradicionais em todos os experimentos, com exceção da base de dados *Airlines* quando os modelos de nível base utilizados são RF ou LR.

Ademais, o MtL proposto supera consistentemente o modelo tradicional na maioria dos casos, evidenciando que as novas MFe propostas têm o poder de aumentar a capacidade preditiva do meta modelo, mesmo que isso resulte em um aumento no tempo total de execução. A escolha de utilizar ou não alguns dos novos grupos de MFe depende das exigências de latência de cada problema. No entanto, algumas medidas de detecção de mudança de conceito se mostraram promissoras em termos de importância elevada e baixo tempo de processamento, como o PSI e o OmvPht. Por fim, a inclusão de MFe calculadas a partir da predição do modelo de nível base se revelou como uma contribuição significativa para a seleção dos meta atributos.

No que tange à geração dos alertas de mudança de conceito, o algoritmo proposto de MtL demonstrou desempenho notavelmente superior em comparação com as medidas de

detecção de mudança de conceito disponíveis na literatura. Esses resultados são especialmente adequados nos casos em que o modelo de nível base utilizado é RF ou DT, conforme confirmado pelo teste estatístico de *Friedman*, em que apresentaram um p valor abaixo de 0.05. Nestes casos, demonstra-se a existência de uma melhoria estatisticamente significativa proporcionada pelo MtL proposto em comparação com as medidas de detecção de mudança de conceito existentes.

Na análise de desempenho médio, considerando todas as bases de dados, a precisão do MtL foi 18% superior à melhor *baseline*, o que significa que o algoritmo proposto capaz de classificar corretamente os dados e evitar falsos positivos. Além disso, o *f1-score* do MtL foi 14% superior, o que evidencia um equilíbrio adequado entre precisão e *recall*. Esses resultados destacam a eficácia e o potencial do MtL proposto como uma abordagem promissora para a previsão de desempenho em ambientes que enfrentam mudanças de conceito. A capacidade do algoritmo de superar as medidas de detecção de mudança de conceito existentes é um indicativo claro de sua robustez e adaptabilidade.

6.0.1 Limitações

Embora os resultados obtidos tenham sido promissores em sua maioria, é importante destacar algumas limitações deste trabalho, como as predições de desempenho insatisfatórias para a base de dados *Airlines*. Esse resultado indica que o desempenho do algoritmo pode variar significativamente dependendo do conjunto de dados e das características específicas de cada cenário. Além disso, analisando os resultados dos alertas de mudança de conceito, observamos que o desempenho do algoritmo é insatisfatório para todas as bases de dados geradas pelo gerador *Stagger*, o qual possui uma função de classificação *booleana*. Essa limitação sugere que a abordagem proposta pode não ser tão adequada para cenários de classificação *booleana*, exigindo investigações adicionais para compreender melhor suas restrições e possíveis ajustes.

6.0.2 Trabalhos futuros

Uma área de pesquisa promissora para estudos futuros é a utilização dos alertas de mudança de conceito do MtL para realizar o retreino automático do modelo de nível base. Desse modo, seria possível ajustar dinamicamente o modelo base, considerando as mudanças de conceito detectadas pelo meta modelo, e, assim, melhorar continuamente o desempenho do sistema em tempo real. Além disso, seria interessante investigar o uso de outras técnicas de detecção de mudança de conceito como parte das MFe. A incorporação de medidas com viéses diferentes pode proporcionar uma visão mais abrangente e robusta

das mudanças nos fluxos de dados, resultando em uma melhoria adicional na previsão de desempenho do modelo de nível base.

Os resultados parciais desta pesquisa foram compilados em um artigo, intitulado '*Model performance prediction: a Meta-Learning approach for concept drift detection*', que foi aceito na conferência *18th International Conference on Hybrid Artificial Intelligence Systems* (HAIS 2023), a ser realizada em setembro em Salamanca, Espanha. Além disso, estamos atualmente em processo de redação de um novo artigo que abordará os resultados finais desta pesquisa. Pretendemos submetê-lo para publicação no periódico *Information Sciences*, revista que se dedica à publicação de trabalhos originais e inovadores na área de Ciência da Computação.

Referências

[1] Janardan e Shikha Mehta: *Concept drift in streaming data classification: Algorithms, platforms and issues.* Procedia Computer Science, 122:804–811, 2017. https://doi.org/10.1016/j.procs.2017.11.440. 1, 34, 52

[2] Adriaans, Pieter: *Data mining.* Pearson Education India, 1996. 1

[3] Wu, Xindong, Xingquan Zhu, Gong Qing Wu e Wei Ding: *Data mining with big data.* IEEE transactions on knowledge and data engineering, 26(1):97–107, 2013. 1

[4] Mitchell, T, B Buchanan, G DeJong, T Dietterich, P Rosenbloom e A Waibel: *Machine learning.* Annual Review of Computer Science, 4(1):417–433, 1990. https://doi.org/10.1146/annurev.cs.04.060190.002221. 1

[5] Chen, Zhiyuan e Bing Liu: *Lifelong Machine Learning.* Springer International Publishing, 2018. https://doi.org/10.1007/978-3-031-01581-6. 1

[6] Janiesch, Christian, Patrick Zschech e Kai Heinrich: *Machine learning and deep learning.* Electronic Markets, 31(3):685–695, abril 2021. https://doi.org/10.1007/s12525-021-00475-2. 1

[7] Mathur, Puneet: *Overview of machine learning in retail.* Em *Machine Learning Applications Using Python*, páginas 147–157. Apress, dezembro 2018. https://doi.org/10.1007/978-1-4842-3787-8_7. 1

[8] Weber, Felix e Reinhard Schütte: *A domain-oriented analysis of the impact of machine learning—the case of retailing.* Big Data and Cognitive Computing, 3(1):11, janeiro 2019. https://doi.org/10.3390/bdcc3010011. 1

[9] Sunilkumar, Chaurasia Neha: *A review of movie recommendation system: Limitations, survey and challenges.* ELCVIA Electronic Letters on Computer Vision and Image Analysis, 19(3):18, setembro 2020. https://doi.org/10.5565/rev/elcvia.1232. 1

[10] Song, Yading, Simon Dixon e Marcus Pearce: *A survey of music recommendation systems and future perspectives.* junho 2012. 1

[11] Aggarwal, Shilpi, Dipanjan Goswami, Madhurima Hooda, Amirta Chakravarty, Arpan Kar e Vasudha: *Recommendation systems for interactive multimedia entertainment.* Em *Data Visualization and Knowledge Engineering*, páginas 23–48. Springer International Publishing, agosto 2019. https://doi.org/10.1007/978-3-030-25797-2_2. 1

[12] Qayyum, Adnan, Junaid Qadir, Muhammad Bilal e Ala Al-Fuqaha: *Secure and robust machine learning for healthcare: A survey*. IEEE Reviews in Biomedical Engineering, 14:156–180, 2021. https://doi.org/10.1109/rbme.2020.3013489. 1

[13] Castiglioni, Isabella, Leonardo Rundo, Marina Codari, Giovanni Di Leo, Christian Salvatore, Matteo Interlenghi, Francesca Gallivanone, Andrea Cozzi, Natascha Claudia D'Amico e Francesco Sardanelli: *Ai applications to medical images: From machine learning to deep learning*. Physica Medica, 83:9–24, 2021, ISSN 1120-1797. https://www.sciencedirect.com/science/article/pii/S1120179721000946. 1

[14] De Bruyne, Sander, Marijn M Speeckaert, Wim Van Biesen e Joris R Delanghe: *Recent evolutions of machine learning applications in clinical laboratory medicine*. Critical Reviews in Clinical Laboratory Sciences, 58(2):131–152, 2021. 1

[15] Varmedja, Dejan, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic e Andras Anderla: *Credit card fraud detection - machine learning methods*. Em *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, páginas 1–5, 2019. 1

[16] Awoyemi, John O., Adebayo O. Adetunmbi e Samuel A. Oluwadare: *Credit card fraud detection using machine learning techniques: A comparative analysis*. Em *2017 International Conference on Computing Networking and Informatics (ICCNI)*, páginas 1–9, 2017. 1

[17] Shah, Neil, Nandish Bhagat e Manan Shah: *Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention*. Visual Computing for Industry, Biomedicine, and Art, 4(1), abril 2021. https://doi.org/10.1186/s42492-021-00075-z. 1

[18] Babu, Shivnath, Lakshminarayan Subramanian e Jennifer Widom: *A data stream management system for network traffic management*. Em *In Proceedings of Workshop on Network-Related Data Management (NRDM 2001)*, 2001. 1

[19] Chandramouli, Badrish, Mohamed Ali, Jonathan Goldstein, Beysim Sezgin e Balan Sethu Raman: *Data stream management systems for computational finance*. Computer, 43(12):45–52, 2010. 1, 6

[20] Khan, Muhammad Aamir, Aunsia Khan, Muhammad Nasir Khan e Sajid Anwar: *A novel learning method to classify data streams in the internet of things*. Em *2014 National Software Engineering Conference*, páginas 61–66, 2014. 1

[21] Gama, João, Pedro Medas, Gladys Castillo e Pedro Rodrigues: *Learning with drift detection*. Em *Advances in Artificial Intelligence – SBIA 2004*, páginas 286–295. Springer Berlin Heidelberg, 2004. https://doi.org/10.1007/978-3-540-28645-5_29. 1, 8, 37, 38

[22] Nguyen, Hai Long, Yew Kwong Woon e Wee Keong Ng: *A survey on data stream clustering and classification*. Knowledge and Information Systems, 45(3):535–569, dezembro 2014. https://doi.org/10.1007/s10115-014-0808-1. 1, 6, 11

[23] Verma, Dipti e Rakesh Nashine: *Data mining: Next generation challenges and futuredirections*. International Journal of Modeling and Optimization, 2(5):603, 2012. 1, 6

[24] Gama, João, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy e Abdelhamid Bouchachia: *A survey on concept drift adaptation*. ACM Computing Surveys, 46(4):1–37, abril 2014. https://doi.org/10.1145/2523813. 1, 8

[25] Žliobaitė, Indrė, Mykola Pechenizkiy e João Gama: *An overview of concept drift applications*. Em *Studies in Big Data*, páginas 91–114. Springer International Publishing, dezembro 2015. https://doi.org/10.1007/978-3-319-26989-4_4. 2

[26] Tsymbal, Alexey: *The problem of concept drift: Definitions and related work*. maio 2004. 2, 8

[27] Gemaque, Rosana Noronha, Albert Franca Josua Costa, Rafael Giusti e Eulanda Miranda dos Santos: *An overview of unsupervised drift detection methods*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10, 2020. 2, 3

[28] Taplin, Ross e Clive Hunt: *The population accuracy index: A new measure of population stability for model monitoring*. Risks, 7(2):53, maio 2019. https://doi.org/10.3390/risks7020053. 2, 12, 13

[29] Pisanie, J. du, J. S. Allison e I. J. H. Visagie: *A proposed simulation technique for population stability testing in credit risk scorecards*, 2022. https://arxiv.org/abs/2206.11344. 2, 3, 11

[30] Gemaque, Rosana Noronha, Albert França Josuá Costa, Rafael Giusti e Eulanda Miranda Santos: *An overview of unsupervised drift detection methods*. WIREs Data Mining and Knowledge Discovery, 10(6), julho 2020. https://doi.org/10.1002/widm.1381. 2, 11, 16

[31] Bashir, Sulaimon Adebayo, Andrei Petrovski e Daniel Doolan: *A framework for unsupervised change detection in activity recognition*. International Journal of Pervasive Computing and Communications, 13(2):157–175, junho 2017. https://doi.org/10.1108/ijpcc-03-2017-0027. 2, 14

[32] Maletzke, André G., Denis M. dos Reis e Gustavo E. A. P. A. Batista: *Combining instance selection and self-training to improve data stream quantification*. Journal of the Brazilian Computer Society, 24(1), outubro 2018. https://doi.org/10.1186/s13173-018-0076-0. 2, 14

[33] Sethi, Tegjyot Singh e Mehmed Kantardzic: *Don't pay for validation: Detecting drifts from unlabeled data using margin density*. Procedia Computer Science, 53:103–112, 2015. https://doi.org/10.1016/j.procs.2015.07.284. 2, 3, 15

[34] Mouss, Hayet, M.Djamel Mouss, Kinza Mouss e Sefouhi Linda: *Test of page-hinckley, an approach for fault detection in an agro-alimentary production system.* páginas 815 – 818 Vol.2, agosto 2004, ISBN 0-7803-8873-9. 2, 16

[35] Muthusamy, Vinod, Aleksander Slominski e Vatche Ishakian: *Towards enterprise-ready AI deployments minimizing the risk of consuming AI models in business applications.* Em *2018 First International Conference on Artificial Intelligence for Industries (AI4I)*. IEEE, setembro 2018. https://doi.org/10.1109/ai4i.2018.8665685. 2

[36] Wolpert, D.H. e W.G. Macready: *No free lunch theorems for optimization.* IEEE Transactions on Evolutionary Computation, 1(1):67–82, 1997. 2, 18

[37] Vanschoren, Joaquin, Carlos Soares, Pavel Brazdil e Lars Kotthoff: *Meta-Learning and Algorithm Selection.* agosto 2014. 2

[38] Brazdil, Pavel e Christophe Giraud-Carrier: *Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue.* Machine Learning, 107, dezembro 2017. 2

[39] Vilalta, Ricardo, Christophe Giraud-Carrier e Pavel Brazdil: *Meta-learning - concepts and techniques.* Em *Data Mining and Knowledge Discovery Handbook*, páginas 717–731. Springer US, 2009. https://doi.org/10.1007/978-0-387-09823-4_36. 2, 18, 20, 31

[40] Brazdil, P.: *Metalearning: Applications to Automated Machine Learning and Data Mining.* Cognitive Technologies. Springer International Publishing, 2022, ISBN 9783030670245. https://books.google.com.br/books?id=Ob9gEAAAQBAJ. 2

[41] Smith-Miles, Kate A.: *Cross-disciplinary perspectives on meta-learning for algorithm selection.* ACM Comput. Surv., 41(1), jan 2009, ISSN 0360-0300. https://doi.org/10.1145/1456650.1456656. 2

[42] Rossi, André Luis Debiaso, André Carlos Ponce de Leon Ferreira 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, março 2014. https://doi.org/10.1016/j.neucom.2013.05.048. 2, 3, 26, 27, 28, 30, 33, 34, 42

[43] Rossi, André Luis Debiaso, Bruno Feres de Souza, Carlos Soares e André Carlos Ponce de Leon Ferreira de Carvalho: *A guidance of data stream characterization for meta-learning.* Intelligent Data Analysis, 21(4):1015–1035, agosto 2017. https://doi.org/10.3233/ida-160083. 2, 26, 27, 42

[44] Sá, Jáder M. C. de, Andre L. D. Rossi, Gustavo E. A. P. A. Batista e Luís P. F. Garcia: *Algorithm recommendation for data streams.* Em *2020 25th International Conference on Pattern Recognition (ICPR)*, páginas 6073–6080, 2021. 2, 26, 27, 42

[45] Rossi, André Luis Debiaso, Carlos Soares, Bruno Feres de Souza e André Carlos Ponce de Leon Ferreira de Carvalho: *Micro-MetaStream: Algorithm selection for time-changing data*. Information Sciences, 565:262–277, julho 2021. https://doi.org/10.1016/j.ins.2021.02.075. 2, 26

[46] Lotfian, Maryam, Jens Ingensand e Maria Antonia Brovelli: *The partnership of citizen science and machine learning: Benefits, risks, and future challenges for engagement, data collection, and data quality*. Sustainability, 13(14):8087, julho 2021. https://doi.org/10.3390/su13148087. 2

[47] Pi, Yulu: *Machine learning in governments: Benefits, challenges and future directions*. JeDEM - eJournal of eDemocracy and Open Government, 13(1):203–219, agosto 2021. https://doi.org/10.29379/jedem.v13i1.625. 2

[48] Zliobaite, Indre: *Learning under concept drift: an overview*. outubro 2010. 2

[49] Agrahari, Supriya e Anil Kumar Singh: *Concept drift detection in data stream mining : A literature review*. Journal of King Saud University - Computer and Information Sciences, dezembro 2021. https://doi.org/10.1016/j.jksuci.2021.11.006. 2

[50] Sobhani, Parinaz e Hamid Beigy: *New drift detection method for data streams*. Em *Adaptive and Intelligent Systems*, páginas 88–97. Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-23857-4_12. 3

[51] *Dimensions*. https://app.dimensions.ai/discover/publication. 3

[52] Li, Junxuan, Yung wen Liu, Yuting Jia, Yifei Ren e Jay Nanduri: *Predictive modeling with delayed information: a case study in e-commerce transaction fraud control*, 2018. https://arxiv.org/abs/1811.06109. 3

[53] Vilalta, Ricardo e Youssef Drissi. *Artificial Intelligence Review*, 18(2):77–95, 2002. https://doi.org/10.1023/a:1019956318069. 3, 18

[54] Peng, Huimin: *A comprehensive overview and survey of recent advances in meta-learning*, 2020. https://arxiv.org/abs/2004.11149. 3, 18

[55] Ali, Shawkat e Kate A. Smith-Miles: *A meta-learning approach to automatic kernel selection for support vector machines*. Neurocomputing, 70(1):173–186, 2006, ISSN 0925-2312. https://www.sciencedirect.com/science/article/pii/S0925231206001056, Neural Networks. 3

[56] Barddal, Jean Paul, Heitor Murilo Gomes e Fabrício Enembreck: *SNCStream*. Em *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, abril 2015. https://doi.org/10.1145/2695664.2695674. 6

[57] Gaber, Mohamed Medhat: *Data stream processing in sensor networks*. Em *Learning from Data Streams*, páginas 41–48. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-73679-4_4. 6

[58] 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 - PODS '02*. ACM Press, 2002. https://doi.org/10.1145/543613.543615. 6

[59] Gama, João: *A survey on learning from data streams: current and future trends*. Progress in Artificial Intelligence, 1(1):45–55, janeiro 2012. https://doi.org/10.1007/s13748-011-0002-6. 6, 7, 11, 41

[60] Gaber, Mohamed, Arkady Zaslavsky e Shonali Krishnaswamy: *Mining data streams: A review*. SIGMOD Record, 34:18–26, junho 2005. 6

[61] Babcock, Brian, Mayur Datar e Rajeev Motwani: *Sampling from a moving window over streaming data*. Em *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, página 633–634, USA, 2002. Society for Industrial and Applied Mathematics, ISBN 089871513X. 7

[62] Kadam, Shweta Vinayak: *A Survey on Classification of Concept Drift with Stream Data*. https://hal.archives-ouvertes.fr/hal-02062610, working paper or preprint, março 2019. 9, 10

[63] *Data stream generation with concept drift*. https://edouardfouche.com/Data-Stream-Generation-with-Concept-Drift/, 2017. 10

[64] Krawczyk, Bartosz e Alberto Cano: *Online ensemble learning with abstaining classifiers for drifting and noisy data streams*. Applied Soft Computing, 68:677–692, julho 2018. 10

[65] Siddiqi, Naeem: *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. John Wiley Sons, 2005. 11, 41

[66] *Dataiku*. https://www.dataiku.com, 2013. 13

[67] Hodges, Joseph L.: *The significance probability of the smirnov two-sample test*. Arkiv för Matematik, 3:469–486, 1958. 15

[68] Sethi, Tegjyot Singh e Mehmed Kantardzic: *On the reliable detection of concept drift from streaming unlabeled data*. Expert Systems with Applications, 82:77–99, outubro 2017. https://doi.org/10.1016/j.eswa.2017.04.008. 16

[69] Dietterich, Thomas G.: *Ensemble methods in machine learning*. Em *Multiple Classifier Systems*, páginas 1–15. Springer Berlin Heidelberg, 2000. https://doi.org/10.1007/3-540-45014-9_1. 16

[70] Lughofer, Edwin, Eva Weigl, Wolfgang Heidl, Christian Eitzinger e Thomas Radauer: *Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances*. Information Sciences, 355-356:127–151, agosto 2016. https://doi.org/10.1016/j.ins.2016.03.034. 16

[71] Schmidhuber, Jurgen: *Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook*. Diploma thesis, Technische Universitat Munchen, Germany, 1987. http://www.idsia.ch/~juergen/diploma.html. 18

[72] Vanschoren, Joaquin: *Meta-learning: A survey*. arXiv preprint arXiv:1810.03548, 2018. 18

[73] Souto, Marcilio C. P. de, Ricardo B. C. Prudencio, Rodrigo G. F. Soares, Daniel S. A. de Araujo, Ivan G. Costa, Teresa B. Ludermir e Alexander Schliep: *Ranking and selecting clustering algorithms using a meta-learning approach*. Em *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, páginas 3729–3735, 2008. 18

[74] Rice, John R.: *The algorithm selection problem**this work was partially supported by the national science foundation through grant gp-32940x. this chapter was presented as the george e. forsythe memorial lecture at the computer science conference, february 19, 1975, washington, d. c.* Volume 15 de *Advances in Computers*, páginas 65–118. Elsevier, 1976. https://www.sciencedirect.com/science/article/pii/S0065245808605203. 19

[75] Salisu, Mamman, Salisu Abdulrahman, Alhassan Adamu, Yazid Ado e Akilu Rilwan: *An overview of the algorithm selection problem*. International Journal of Computer (IJC), janeiro 2017. 19

[76] Cohen, Jacob: *A coefficient of agreement for nominal scales*. Educational and Psychological Measurement, 20(1):37–46, abril 1960. https://doi.org/10.1177/001316446002000104. 20, 34

[77] Rivolli, Adriano, Luís P.F. Garcia, Carlos Soares, Joaquin Vanschoren e André C.P.L.F. de Carvalho: *Meta-features for meta-learning*. Knowledge-Based Systems, 240:108101, março 2022. https://doi.org/10.1016/j.knosys.2021.108101. 22, 23, 24, 25, 27, 30, 39

[78] Reif, Matthias: *A comprehensive dataset for evaluating approaches of various meta-learning tasks*. janeiro 2012. 22

[79] Hearst, M.A., S.T. Dumais, E. Osuna, J. Platt e B. Scholkopf: *Support vector machines*. IEEE Intelligent Systems and their Applications, 13(4):18–28, 1998. 33

[80] Breiman, Leo. *Machine Learning*, 45(1):5–32, 2001. https://doi.org/10.1023/a:1010933404324. 33

[81] LaValley, Michael P.: *Logistic regression*. Circulation, 117(18):2395–2399, maio 2008. https://doi.org/10.1161/circulationaha.106.682658. 33

[82] Ville, Barry de: *Decision trees*. Wiley Interdisciplinary Reviews: Computational Statistics, 5(6):448–455, outubro 2013. https://doi.org/10.1002/wics.1278. 33

[83] Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt e Gaël Varoquaux: *API design for machine learning software: experiences from the scikit-learn project*. Em *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, páginas 108–122, 2013. 33

[84] Liu, Hongyan, Yuan Lin e Jiawei Han: *Methods for mining frequent items in data streams: an overview*. Knowledge and Information Systems, 26(1):1–30, novembro 2009. https://doi.org/10.1007/s10115-009-0267-2. 34

[85] Souza, V. M. A., D. M. Reis, A. G. Maletzke e G. E. A. P. A. Batista: *Challenges in benchmarking stream learning algorithms with real-world data*. Data Mining and Knowledge Discovery, 34:1805–1858, 2020. 34

[86] Harries, Michael, U Nsw cse tr e New South Wales: *Splice-2 comparative evaluation: Electricity pricing*. Relatório Técnico, 1999. 35

[87] Ikonomovska, Elena, João Gama, Raquel Sebastião e Dejan Gjorgjevik: *Regression trees from data streams with drift detection*. Em *Discovery Science*, páginas 121–135. Springer Berlin Heidelberg, 2009. https://doi.org/10.1007/978-3-642-04747-3_12. 36

[88] Losing, Viktor, Barbara Hammer e Heiko Wersing: *KNN classifier with self adjusting memory for heterogeneous concept drift*. Em *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, dezembro 2016. https://doi.org/10.1109/icdm.2016.0040. 36

[89] Montiel, Jacob, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem *et al.*: *River: machine learning for streaming data in python*. 2021. 37

[90] Street, W. Nick e YongSeog Kim: *A streaming ensemble algorithm (SEA) for large-scale classification*. Em *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, agosto 2001. https://doi.org/10.1145/502512.502568. 37

[91] Schlimmer, Jeffrey C. e Richard H. Granger: *Incremental learning from noisy data*. Machine Learning, 1(3):317–354, setembro 1986. https://doi.org/10.1007/bf00116895. 37

[92] Agrawal, R., T. Imielinski e A. Swami: *Database mining: a performance perspective*. IEEE Transactions on Knowledge and Data Engineering, 5(6):914–925, 1993. 38

[93] 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, página 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc., ISBN 9781510860964. 41

[94] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011. 41, 49

[95] *Lightgbm repository*. `https://github.com/microsoft/LightGBM`, 2017. 45, 49

[96] Arlot, Sylvain e Alain Celisse: *A survey of cross-validation procedures for model selection*. arXiv, 2009. `https://arxiv.org/abs/0907.4728`. 45

[97] *Lightgbm documentation page*. `https://lightgbm.readthedocs.io/en/v3.3.2/`, 2017. 46

[98] Labs, Cloudera Fast Forward: *Inferring concept drift without labeled data*. `https://concept-drift.fastforwardlabs.com/`, 2021. 47

[99] Demšar, Janez: *Statistical comparisons of classifiers over multiple data sets*. J. Mach. Learn. Res., 7:1–30, dec 2006, ISSN 1532-4435. 49

[100] Melo, F.A.: *Drift meta learning*. `https://github.com/feamelo/drift-meta-learning`, 2022. 49

[101] Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke e Travis E. Oliphant: *Array programming with NumPy*. Nature, 585(7825):357–362, setembro 2020. `https://doi.org/10.1038/s41586-020-2649-2`. 49

[102] Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt e SciPy 1.0 Contributors: *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17:261–272, 2020. 49

[103] Satopaa, Ville, Jeannie Albrecht, David Irwin e Barath Raghavan: *Finding a "kneedle" in a haystack: Detecting knee points in system behavior*. Em *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE, junho 2011. `https://doi.org/10.1109/icdcsw.2011.20`. 49

[104] Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta e Masanori Koyama: *Optuna: A next-generation hyperparameter optimization framework*, 2019. `https://arxiv.org/abs/1907.10902`. 49