UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO – BCC

FERNANDA MARIA DE SOUZA

ADAPTAÇÃO DA BIBLIOTECA AXGBOOST PARA REGRESSÃO EM FLUXOS DE DADOS COM DETECÇÃO E RESPOSTA À MUDANÇA DE CONCEITOS REALIZADA POR MÚLTIPLOS DETECTORES

JOINVILLE
2022

FERNANDA MARIA DE SOUZA

ADAPTAÇÃO DA BIBLIOTECA AXGBOOST PARA REGRESSÃO EM FLUXOS DE DADOS COM DETECÇÃO E RESPOSTA À MUDANÇA DE CONCEITOS REALIZADA POR MÚLTIPLOS DETECTORES

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Fabiano Baldo

JOINVILLE
2022

Para gerar a ficha catalográfica de teses e
dissertações acessar o link:
https://www.udesc.br/bu/manuais/ficha

,
Adaptação da Biblioteca AXGBoost para Regressão em Fluxos de Dados com Detecção e Resposta à Mudança de Conceitos Realizada por Múltiplos Detectores / Fernanda Maria de Souza. - Joinville, 2022.
63 p. : il. ; 30 cm.

Orientador: Fabiano Baldo.
.
Dissertação - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Bacharelado em Ciência Da Computação, Joinville, 2022.

1. Regressão. 2. Mudança de conceito. 3. Fluxos de dados. 4. AXGB. I. Baldo, Fabiano. II. , . III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Bacharelado em Ciência Da Computação. IV. Título.

FERNANDA MARIA DE SOUZA

ADAPTAÇÃO DA BIBLIOTECA AXGBOOST PARA REGRESSÃO EM FLUXOS DE DADOS COM DETECÇÃO E RESPOSTA À MUDANÇA DE CONCEITOS REALIZADA POR MÚLTIPLOS DETECTORES

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Fabiano Baldo

BANCA EXAMINADORA:

Fabiano Baldo - Doutor
UDESC

Membros:

Rafael Stubs Parpinelli - Doutor
UDESC

Fernando José Braz - Doutor
IFC

Joinville, 11 de Julho de 2022

RESUMO

Com cada vez mais dados sendo gerados de maneira exponencial, a ciência de dados transformou-se em uma área essencial para compreensão e identificação de padrões de informação. Esses dados muitas vezes podem vir de forma contínua e sem limitações, sendo conhecidos como fluxo de dados. Atualmente, a regressão de fluxo de dados é uma área que apresenta pouco estudos e ainda carece de algoritmos que consigam analisar adequadamente fluxos com presença de mudança de conceito. Portanto, tendo em vista que os fluxos de dados estão suscetíveis a apresentarem mudanças de conceito, este trabalho propõe o AFXGBReg, um algoritmo de regressão adaptativo rápido baseado em XGBoost, que utiliza detectores ativos de mudança de conceito. Para isso, o AFXGBReg usa uma estratégia para treinamento de modelo de forma alternada, combinado os algoritmos de detecção de mudanças de conceito: ADWIN, KSWIN e DDM. Para os testes, duas variantes do AFXGBReg, o AFXGBReg-D e o AFXGBReg-Dr, foram comparados com os algoritmos de regressão *benchmarks* da literatura. Como resultado, ambas as variações do AFXGBReg apresentaram MSE semelhante ao ARFReg, sendo eles os algoritmos com menor MSE dos testes. Entretanto, apesar dos modelos AFXGBReg e ARFReg apresentarem números de MSE similares, analisando seus tempos de execução, o algoritmo AFXGBReg-Dr e AFXGBReg-D obtiveram tempos de execução 33 vezes mais rápido que o ARFReg, o que significa que são capazes de manter o mesmo nível de MSE com um tempo de execução consideravelmente menor. Ainda, os modelos gerados pelo AFXGBReg-Dr e AFXGBReg-D conseguem se recuperar mais rapidamente das mudanças de conceito em comparação aos outros algoritmos analisados.

Palavras-chave: Regressão, Mudança de conceito, Fluxos de dados, AXGB.

ABSTRACT

With more and more data being generated exponentially, data science has become an essential area for understanding and identifying information patterns. This data can often come in a continuous and unconstrained form, and is known as a data stream. Currently, data stream regression is an area that is poorly studied and still lacks algorithms that can adequately analyze streams with the presence of concept change. Therefore, given that data streams are susceptible to having concept drift, this paper proposes AFXGBReg, an XGBoost-based fast adaptive regression algorithm that uses active concept change detectors. To do this, AFXGBReg uses a strategy for alternating model training, combining the ADWIN, KSWIN and DDM algorithms for concept change detection. For testing, two variants of AFXGBReg, AFXGBReg-D and AFXGBReg-Dr, were compared with regression algorithms *benchmarks* from the literature. As a result, both variations of AFXGBReg showed similar MSE to ARFReg, and they were the algorithms with the lowest MSE of the tests. However, although the AFXGBReg and ARFReg models presented similar MSE numbers, analyzing their execution times, the AFXGBReg-Dr and AFXGBReg-D algorithm obtained execution times 33 times faster than ARFReg, which means that they are able to maintain the same MSE level with a considerably shorter execution time. Also, the models generated by AFXGBReg-Dr and AFXGBReg-D are able to recover faster from concept changes compared to the other analyzed algorithms.

Keywords: Regression, Concept Drift, Data Streams, AXGB.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADWIN	Adaptive Windowing
AM	Aprendizado de Máquina
ARF	Adaptive Random Forest
AFXGB	Adaptive Fast eXtreme Gradient Boosting
AXGB	Adaptive eXtreme Gradient Boosting
DDM	Drift Detection Method
HYP	Hyperplane Generator
IOT	Internet Of Things
KNN	K-Nearest Neighbors Algorithm
KSWIN	Kolmogorov-Smirnov Windowing
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
PHT	Page Hincley
PWPAE	Performance Weighted Probability Averaging Ensemble
SEA	Streaming Ensemble Algorithm
SRP	Streaming Random Patches
XGB	eXtreme Gradient Boosting

SUMÁRIO

1 INTRODUÇÃO

Atualmente, com a evolução das tecnologias de armazenamento em nuvem, a melhoria da infraestrutura de comunicações e da popularização dos dispositivos móveis, dados são produzidos continuamente em escala crescente. A partir da observação dessa abundância de dados é que surgiu o conceito de *Big Data*, cuja ciência de dados tenta extrair potenciais benefícios de sua análise para a sociedade (LARSON; CHANG, 2016). O termo *Big Data* visa descrever o crescimento exponencial de dados com base em cinco princípios: i) o volume de dados, ii) a variabilidade dos tipos de dados, iii) a velocidade que os dados são gerados, capturados e processados, iv) a sua veracidade, de forma a obter dados verídicos, de acordo com a realidade e v) o valor, os custos envolvidos nessa operação e o valor agregado de todo esse trabalho (LANEY, 2001).

Considerando os potenciais benefícios da extração de padrões e informações relevantes a partir de grandes volumes de dados, dentro da ciência de dados outro termo de forte consolidação é o conceito de mineração de dados. Nele, a partir de algoritmos de aprendizado de máquina, é possível explorar um conjunto de dados com o objetivo de estabelecer relações difíceis de visualizar manualmente.

No contexto do aprendizado de máquina, a grande maioria das pesquisas se concentra em construir modelos estáticos, que são treinados uma vez sobre um conjunto de dados de treinamento e depois são aplicados a análise de novos dados (GAMAGE; PREMARATNE, 2017). Contudo, com a crescente geração de dados por sensores, dispositivos IoT (Internet of Things), tráfego de rede de computadores, conversas telefônicas e transações financeiras, a demanda por processamento de fluxos de dados não estacionários, que mudam continuamente com o tempo, vêm crescendo consideravelmente. Os fluxos de dados podem ser entendidos como sequências de dados não estacionários gerados de forma constante e por tempo indeterminado (YU; LU; ZHANG, 2021). Essas características fazem com que os algoritmos de mineração não possam armazenar os dados e que eles precisem lidar com as mudanças de conceitos inerentes à característica não estacionária dos fluxos (KRAWCZYK et al., 2017a).

Com isso, a mineração de fluxos de dados lida com desafios sobre as seguintes dimensões: i) rapidez: os dados são processados em tempo limitado para não gerar fila por atraso no processamento; ii) eficiência de memória: os dados processados anteriormente precisam ser descartados, pois pode não haver espaço para armazená-los; iii) não iterativo: os dados são processados uma única vez, pois não podem ser reutilizados; e iv) adaptativo: o modelo de aprendizagem deve ser ajustado às mudanças de conceito (LU et al., 2018).

No contexto de aprendizado de máquina existem duas abordagens principais utilizadas pelos algoritmos para aprender a partir dos dados de treinamento. A primeira é o aprendizado não-supervisionado, onde não há um conjunto anotado de treinamento e, portanto, a própria estrutura dos dados é utilizada para realizar o aprendizado. A segunda é o aprendizado supervisionado, cujo objetivo é projetar um modelo usando exemplos anotados de treinamento para prever

corretamente os rótulos para novos dados não rotulados (MARSLAND, 2011). Para ambas as abordagens, quando os rótulos a serem previstos são discretos, o problema é resolvido como uma tarefa de classificação. Entretanto, quando os rótulos a serem previstos são contínuos, então, ela é resolvida como uma tarefa de regressão.

Assim como na classificação, no contexto da regressão em fluxo de dados o modelo preditivo precisa se adaptar às mudanças de conceito. Portanto, para dados que mudam dinamicamente, os métodos tradicionais de aprendizagem supervisionada são inadequados e podem causar perda no desempenho do modelo. Para tentar manter a melhor precisão possível, o modelo treinado precisa se adaptar usando novos dados de entrada. Portanto, os modelos preditivos precisam ser treinados de forma incremental, seja por atualização contínua ou por retreinamento usando lotes recentes de dados (GAMA et al., 2014b).

Embora existam algumas propostas de abordagens para regressão de fluxos de dados, quando comparadas às iniciativas de classificação de fluxos de dados, ela atrai menos atenção da comunidade científica, pois é considerada um problema mais complexo e, portanto, requer mais esforço para solução (YU; LU; ZHANG, 2021).

No aprendizado de máquina, a mudança de conceito significa que as propriedades estatísticas da variável de destino, que o modelo está tentando prever, mudam ao longo do tempo de maneira imprevista (LIAO; WANG, 2018). Portanto, isso resulta em problemas nas predições, pois o modelo se torna menos preciso com o tempo. As formas de detecção de mudança de conceito para aprendizado de máquina adaptativo podem ser classificadas em duas categorias principais, passivas e ativas. Nas formas passivas, os modelos de aprendizagem são atualizados sem a detecção prévia da mudança de conceito (DITZLER et al., 2015), portanto, tendem a serem mais lentas na detecção da mudança. Nas ativas, as abordagens são focadas em detectar primeiro o desvio/mudança de conceito e, posteriormente, acelerar o processo de atualização do modelo para que a alteração seja adsorvida pelo modelo o quanto antes, tornando o processo de recuperação mais rápido do que na passiva (ELWELL; POLIKAR, 2011).

Para lidar com a mudança de conceito em dados não estacionários, as abordagens passivas são capazes de se adaptar com atualizações contínuas, mas exigem maior esforços computacionais por serem mais lentas (MEHMOOD et al., 2021). Entretanto, as ativas tentem a serem mais rápidas, porém, introduzem ao processo de aprendizado mais uma tarefa a ser executa, que é a execução do algoritmo de detecção. Neste trabalho, será explorada a utilização de detecção ativa de mudança de conceito a fim de explorar seus potenciais benefícios e limitações em relação a detecção passiva na regressão de fluxos de dados.

Dentre as abordagens para a classificação e regressão de fluxos de dados, os conjuntos de classificadores se destacam com os mais robustos e precisos. Analisando especificamente as abordagens de conjuntos de classificadores, os algoritmos de *boosting* representam uma abordagem promissora, pois são capazes de produzir um modelo geral preciso a partir da combinação de vários modelos moderadamente imprecisos (SCHAPIRE, 2003). A ideia básica desses algoritmos é treinar iterativamente novos modelos e combinar suas soluções para obter

um resultado com melhor predição (MAYR et al., 2014). Dois de seus algoritmos mais famosos são o *AdaBoost* e o *Gradient Boosting*, que trabalham adicionando sequencialmente modelos que corrigem os erros residuais de predição dos modelos anteriores, geralmente constituídos de árvores de decisão.

Porém, a construção de modelos baseados na abordagem *boosting* pode ser lenta, especialmente para grandes conjuntos. Portanto, para mitigar esse problema foi proposto o algoritmo *eXtreme Gradient Boosting* (XGBoost), o qual cria árvores individuais usando vários processadores e os dados são organizados para minimizar o tempo de pesquisa, o que diminui o tempo de treinamento e melhora o desempenho (RAMRAJ et al., 2016). O algoritmo XGBoost é eficaz para uma ampla gama de problemas de modelagem preditiva de classificação e regressão (CHEN et al., 2015), porém, sua aplicação em fluxo de dados ainda foi pouco explorada. Os principais trabalhos são dedicados a classificação binária de fluxos de dados, sendo eles o *Adaptative Extreme Gradient Boosting* (AXGB) e o *Adaptative Fast Extreme Gradient Boosting* (AFXGB). O AXGB usa uma estratégia incremental para construir um conjunto de classificadores XGBoost, onde cada novo classificador é criado a partir de uma amostra de dados obtida da janela deslizante adaptativa de dados recebidos pelo fluxo (MONTIEL et al., 2020). Posteriormente, Bonassa (2021) propôs o AFXGB que é uma adaptação do algoritmo proposto por Montiel et al. (2020) que, ao invés de usar um conjunto de classificadores XGBoost, usa uma estratégia de substituição de classificador único XGBoost. Essa estratégia tornou o processo de treinamento mais rápido, sem diminuir a precisão dos modelos gerados pelo algoritmo.

Portanto, sabendo que em fluxo de dados as informações chegam de forma contínua e estão suscetíveis a apresentarem mudança de conceito, e tendo em vista que o algoritmo XGBoost demonstrou bons resultados na classificação, mas ainda não foi aplicada na regressão de fluxos, este trabalho apresenta a seguinte pergunta de pesquisa: Como a aplicação de diferentes técnicas de detecção de mudança de conceito pode diminuir o tempo de adaptação dos modelos às mudanças em regressão de fluxos de dados utilizando o algoritmo XGBoost?

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo portar o algoritmo AFXGB para regressão de fluxo de dados, e introduzir nele técnicas de detecção ativa de mudança de conceito para acelerar o processo de adaptação dos modelos e, por consequência, melhorar seu desempenho tanto na precisão dos resultados quanto no tempo de processamento do fluxo de dados.

1.2 OBJETIVOS ESPECÍFICOS

Os seguintes objetivos específicos foram definidos baseados no objetivo geral:

- Adaptar o algoritmo AFXGB proposto por Bonassa (2021) para regressão;

- Adaptar o algoritmo AFXGB proposto por Bonassa (2021) para suportar a execução de técnicas ativas de detecção de mudança de conceito.
- Implementar as adaptações projetadas usando a implementação do AFXGB realizada por Bonassa (2021).

1.3 METODOLOGIA

O procedimento para realizar este trabalho se inicia com a revisão e estudo dos conceitos abordados pela temática da pesquisa, sendo eles: algoritmos de regressão, aprendizado supervisionado, fluxo de dados, mudança de conceito, algoritmos de *boosting*, detectores ativos de mudança de conceito e métricas de regressão para validação. Além disso, os trabalhos relacionados encontrados na literatura também serão revisados.

Após o levantamento bibliográfico, é feito o projeto do que será necessário alterar para portar a implementação do AFXGB Bonassa (2021) para regressão em fluxo de dados. O algoritmo implementado deve ser capaz de suportar fluxo de dados e lidar com a mudança de conceito. Além disso, os testes para verificação de mudança de conceito serão implementados por meio de testes de detecção ativa. Após o projeto, o algoritmo será implementado utilizando a linguagem de programação Python.

Após terminada a implementação, o algoritmo será testado e validado em bases de dados reais e sintéticas, utilizando métricas como MSE e R^2. Os resultados da utilização da detecção ativa serão comparados com algoritmos de *benchmark* encontrados na literatura para verificar se houve alguma vantagem na sua utilização.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido da seguinte maneira. O Capítulo 2 traz uma revisão dos principais conceitos envolvidos no entendimento do trabalho, bem como uma revisão dos trabalhos relacionados. O Capítulo 3 apresenta a proposta de algoritmo baseado na união de detectores ativos de mudança de conceito na regressão de fluxos de dados. O Capítulo 4 apresenta análise dos resultados. Por fim, o Capítulo 5 apresenta as considerações finais sobre o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais conceitos necessários para o entendimento deste trabalho. Serão abordados os conceitos de Aprendizado de Máquina e suas técnicas, os desafios existentes envolvendo fluxos de dados, bem como uma análise dos trabalhos relacionados.

2.1 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina (AM) é um campo da ciência da computação, derivado da Inteligência Artificial (IA), que introduz nos sistemas formas para automaticamente aprenderem conceitos e conhecimentos sem serem explicitamente programados para isso (DARGAN et al., 2019).

Ainda, segundo Provost e Kohavi (1998), o AM explora o estudo e construção de algoritmos que podem aprender e fazer previsões sobre dados. Com isso, esses algoritmos superam instruções de programas estritamente estáticas, realizando previsões ou decisões por meio da construção de um modelo baseado em conjuntos de dados, que possuem características (atributos) comuns.

Em grande maioria dos modelos de aprendizado de máquina, parte dos dados são usados para treinar o modelo, enquanto outra parte para testar o modelo. Neste cenário, *overfitting* (super-especialização) e *underfitting* (sub-especialização) são algumas das maiores causas de má generalização dos algoritmos de aprendizado de máquina. Koehrsen (2018) define esses dois conceitos: "o *overfitting* se caracteriza pelo momento onde o modelo se adaptou muito bem aos dados com os quais está sendo treinado, porém para novos dados de teste que chegam isso não acontece da mesma forma. Inversamente, a sub-especialização *underfitting* é definida quando o modelo não se adapta bem sequer aos dados com os quais foi treinado".

Ademais, algoritmos de aprendizado de máquina podem ser divididos por duas características principais, de acordo com a categorização feita por Verbraeken et al. (2020),:

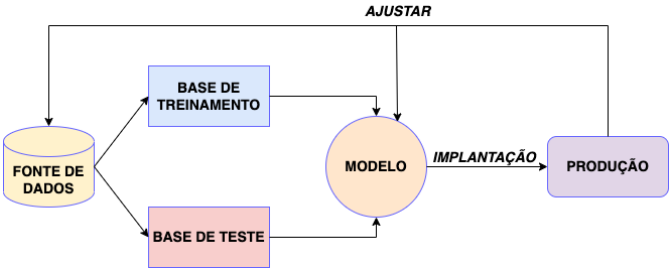
- *Feedback* - o tipo de experiência que é dada ao algoritmo durante o aprendizado.
- Objetivo - o resultado final desejado do algoritmo.

2.1.1 *Feedback*

Nesse tipo de abordagem, no treinamento do algoritmo ele requer um *feedback* para que possa melhorar gradativamente a qualidade do modelo. Existem diferentes tipos de *feedback*:

- **Aprendizado Supervisionado**: nesse método, o algoritmo é treinado usando exemplos rotulados, como uma entrada onde a saída desejada é conhecida. Portanto, o objetivo principal desse aprendizado é fazer com que a função que mapeia os dados de entrada possa ser aplicada a novos dados de entrada visando prever a saída. A Figura 1 apresenta o fluxograma do processo de aprendizado supervisionado.

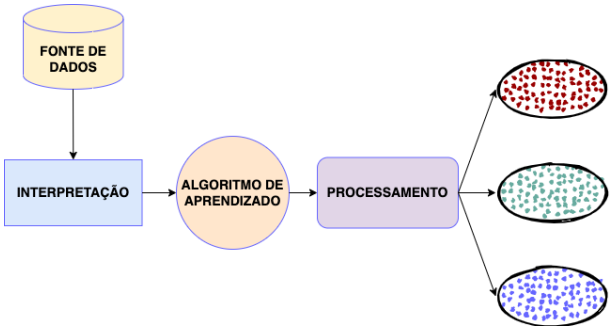
Figura 1 – Fluxo do Aprendizado Supervisionado



Fonte: O autor.

- **Aprendizado Não-Supervisionado**: as instâncias do conjunto de entrada não possuem classificação, ou seja, usa dados de treinamento que consistem em objetos de entrada sem valores de saída. Um ponto importante a ser destacado é que esse tipo de técnica de aprendizado não gera modelos. Como ilustrado na Figura 2, quando novos dados são introduzidos, ele usa todos as instâncias do conjunto de entrada para reconhecer a classe dos dados.

Figura 2 – Fluxo do Aprendizado Não-Supervisionado



Fonte: O autor.

- **Aprendizado Semi-Supervisionado**: é uma combinação dos dois métodos anteriores, usando dados rotulados e não rotulados para treinamento e construção de modelos.

2.1.2 Objetivo

De acordo com Kwon et al. (2019), os algoritmos de AM podem ser usados para uma ampla variedade de propósitos, porém normalmente são utilizados para dois objetivos principais: a classificação e a regressão.

- **Classificação**: a classificação de dados é o processo de formação do modelo ou classificador para prever os rótulos categóricos, que são os rótulos responsáveis por tornar as classes de dados distintas (HAN; PEI; KAMBER, 2011). A classificação é um processo de duas etapas. Durante a primeira etapa, o modelo é criado aplicando-se o algoritmo de classificação no conjunto de dados de treinamento. Durante a segunda etapa o modelo extraído deve ser validado contra um conjunto de dados teste predefinido, visando avaliar o desempenho e precisão do modelo treinado (NIKAM, 2015).

- **Regressão**: seu objetivo é estimar valores numéricos ao invés de classes. Modelos de regressão fazem uso de atributos ou recursos de dados de entrada e seus valores de saída numéricos contínuos correspondentes, visando aprender relações e associações específicas entre as entradas e suas saídas correspondentes (GOLLAPUDI, 2016).

Portanto, enquanto a classificação é a tarefa de prever um rótulo de classe discreta, a regressão é a tarefa de prever uma quantidade contínua. (MAULUD; ABDULAZEEZ, 2020).

2.1.3 Conjuntos de Classificadores

O aprendizado por conjuntos de classificadores ganhou uma popularidade significativa na comunidade de mineração de dados ao longo dos últimos anos (KRAWCZYK; PFAHRINGER; WOŹNIAK, 2018). De acordo com Krawczyk et al. (2017a), um *ensemble*, também chamado de conjunto de classificadores, é a união de componentes individuais cujas previsões são combinadas para prever novas instâncias de entrada. Eles têm se mostrado uma maneira eficiente de melhorar a acurácia preditiva, bem como decompor um problema de aprendizado complexo e difícil em subproblemas mais fáceis (DONG et al., 2020).

Para Ho (2000), os principais objetivos no projeto de um conjunto de classificadores são:

- A otimização da cobertura deve focar na geração de um conjunto de classificadores mutuamente complementares, que podem ser combinados para alcançar a acurácia ideal usando uma função de combinação de decisão fixa.

- A otimização de decisões deve concentrar-se em projetar e treinar uma função de combinação de decisão apropriada, responsável pela decisão final do conjunto, que deve explorar os pontos fortes dos classificadores.

Um conjunto ideal inclui classificadores individuais mutuamente complementares que são caracterizados por alta diversidade e acurácia (KROGH; VEDELSBY et al., 1995). Ainda, segundo Zenobi e Cunningham (2001), é geralmente aceito que não apenas a precisão, mas também a diversidade dos classificadores, são ingredientes chave para aumentar a precisão do conjunto. Com o objetivo de combinar os resultados desses classificadores surgiram técnicas de destaque, tais como:

- ***Bagging*:** Breiman (1996) introduziu o conceito de *Bootstrapped Aggregating* para construir conjuntos de classificadores. Galar et al. (2011) definiu que o método de *Bagging* consiste em treinar diferentes classificadores com réplicas *bootstrap* do conjunto de dados de treinamento original. Assim, quando uma instância desconhecida é apresentada a cada classificador, a maioria define qual a classe mais prevista por eles.

- ***Boosting*:** O método de Boosting foi proposto por Schapire (1990), que provou que um aprendiz fraco (que é ligeiramente melhor do que uma adivinhação aleatória) pode ser

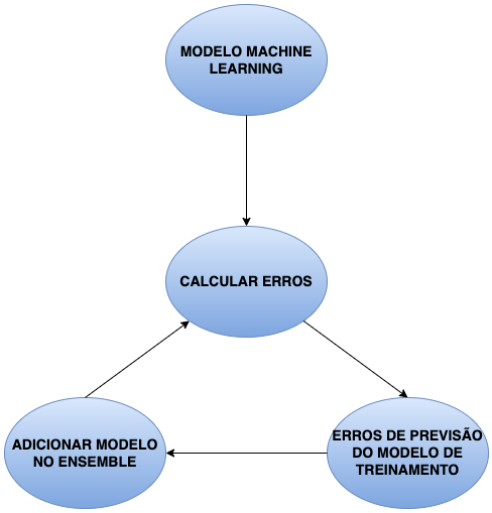
transformado em um aprendiz forte. Para isso, novos classificadores são influenciados pela saída de classificadores anteriores, de forma que a mudança nos dados de treinamento para os novos modelos é baseada nas instâncias mal classificadas anteriormente. Com isso, o *Boosting* tenta melhorar o desempenho de cada novo modelo de forma iterativa com base em modelos anteriores.

- ***Gradient Boosting*:** Este método, assim como o algoritmo de *Boosting*, treina novos modelos baseados nos anteriores com o objetivo de aumentar a acurácia do classificador. A principal diferença é como os algoritmos identificam as deficiências dos aprendizes fracos. Para o *Gradient Boosting*, existe uma função de perda a ser otimizada, que indica quão bons são os coeficientes do modelo em ajustar os dados subjacentes.

2.1.4 XGBoost

O XGBoost é um algoritmo de aprendizado de máquina escalável desenvolvido sobre a abordagem *Boosting* proposto por Chen e Guestrin (2016). Ele é o resultado de uma otimização do *Gradient Boosting* que explora os benefícios da paralelização na criação dos modelos do conjunto e possui uma biblioteca de código aberto projetado para ser altamente eficiente e flexível.

Figura 3 – Fluxo do XGBoost.

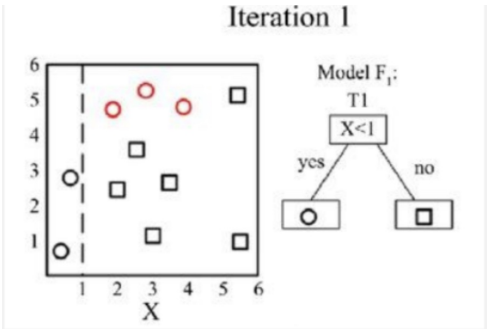


Fonte: O autor.

Assim com no *Gradient Boosting*, no XGBoost a cada iteração o resíduo do preditor anterior é usado para corrigir o próximo preditor de maneira eficiente pela otimização da função de perda, conforme Figura 3. Dessa forma, um peso maior é dado às instâncias que o XGBoost tem maior dificuldade de predizer visando estabelecer a função objetiva com melhor desempenho para o modelo. A flexibilidade do *XGBoost* se dá pela utilização de hiper-parâmetros passíveis de aperfeiçoamento, onde é possível ajustar o *XGBoost* para utilização em diferentes problemas.

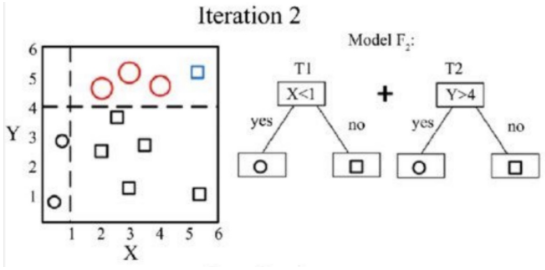
As Figuras 4, 5 e 6 demonstram o funcionamento com o processo iterativo de aprendizado da mesma forma que o XGB utiliza. O objetivo é combinar vários aprendizes fracos para produzir

Figura 4 – Funcionamento do XGB - parte 1.



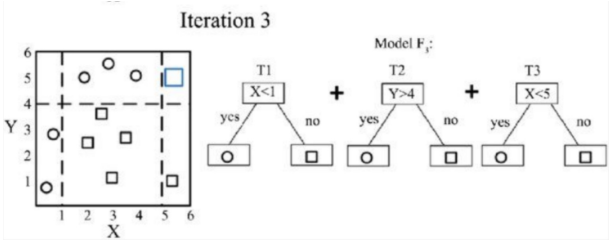
Fonte: Fafalios, Charonyktakis e Tsamardinos (2020).

Figura 5 – Funcionamento do XGB - parte 2.



Fonte: Fafalios, Charonyktakis e Tsamardinos (2020).

Figura 6 – Funcionamento do XGB - parte 3.



Fonte: Fafalios, Charonyktakis e Tsamardinos (2020).

um aprendiz poderoso. Os aprendizes são construídos sequencialmente e os resultados finais são somados tentando reduzir o viés. A partir da primeira iteração o modelo é testado e os resíduos considerados para aprendizados futuros, de modo a construir novos modelos de forma paralelizada com árvores de decisão.

2.2 REGRESSÃO DE FLUXO DE DADOS

A aplicação do aprendizado de máquina em fluxos de dados tem crescido em importância nos últimos anos devido à grande quantidade de dados em tempo real gerados por redes, telefones celulares e a grande variedade de sensores atualmente disponíveis (DONG et al., 2020). Fluxo de dados pode ser definido como uma sequência de exemplos, contextualmente chamados de eventos, que ocorrem no tempo em uma taxa que não permite o seu armazenamento permanente em memória (REIS, 2017).

O processo de mineração de fluxo de dados difere do processo de mineração de dados tradicional devido ao ambiente e às características dos dados serem diferentes (HAN; PEI; KAMBER, 2011). Bifet (2009) descreve as principais restrições para os algoritmos que processam fluxos de dados:

- Não é possível armazenar todos os dados do fluxo: somente uma pequena parte pode ser processada e armazenada;
- A velocidade de chegada dos dados do fluxo exige que cada elemento seja processado em tempo real para em seguida ser descartado;
- A distribuição dos dados pode mudar com o tempo, logo, os dados que foram processados anteriormente e fazem parte do modelo podem se tornar irrelevantes e até mesmo prejudicar os conceitos atuais.

Junto com a classificação, a tarefa de regressão é um dos tópicos mais comuns na área de aprendizado de máquina. No entanto, a regressão baseada em fluxos de dados é ainda pouco explorada no meio acadêmico, embora sua pesquisa seja bastante significativa para a era do aprendizado de máquina Liao e Wang (2018). Muitos métodos que utilizam conjuntos de classificadores podem ser encontrados na literatura para resolver tarefas de classificação em fluxos de dados, entretanto, para a tarefa de regressão são poucos. Dentre eles, os que merecem destaque são: **KNN** (ALTMAN, 1992), **ARFReg** (GOMES et al., 2018), **AMRules** (DUARTE; GAMA; BIFET, 2016), **eFIMT-DD** (IKONOMOVSKA; GAMA; DŽEROSKI, 2015), **Análise de regressão linear multidimensional de fluxos de dados de séries temporais** (CHEN et al., 2002), e **Rival Learner** (LIAO; WANG, 2018).

Outra característica importante do fluxo de dados é a mudança de conceito. Os modelos de aprendizagem devem prever mudanças à medida que os dados evoluem, por isso é necessário adaptar o modelo aos dados mais recentes.

2.2.1 Mudança de conceito

Um dos principais pressupostos dos tradicionais métodos de mineração de dados é que cada conjunto de dados é gerado a partir de uma única função estática, ou seja, a função usada para gerar as dados de treinamento é a mesma usada para testes (HOENS; POLIKAR; CHAWLA, 2012). No modelo de fluxo de dados isso não necessariamente é verdade, visto que a função pode variar durante o tempo de geração do fluxo. Essa (potencial) variação é conhecida como mudança de conceito.

Logo, a mudança de conceito ocorre quando as distribuições de dados mudam ao longo do tempo de maneira imprevisível, tornando os resultados consequentemente imprecisos (PINAGÉ et al., 2017). Widmer (1994) define a mudança de conceito da seguinte forma: “Em muitos domínios do mundo real, o contexto em que alguns conceitos de interesse dependem pode mudar,

resultando em mudanças mais ou menos abruptas e radicais na definição do conceito-alvo. A mudança no conceito alvo é conhecido como mudança de conceito".

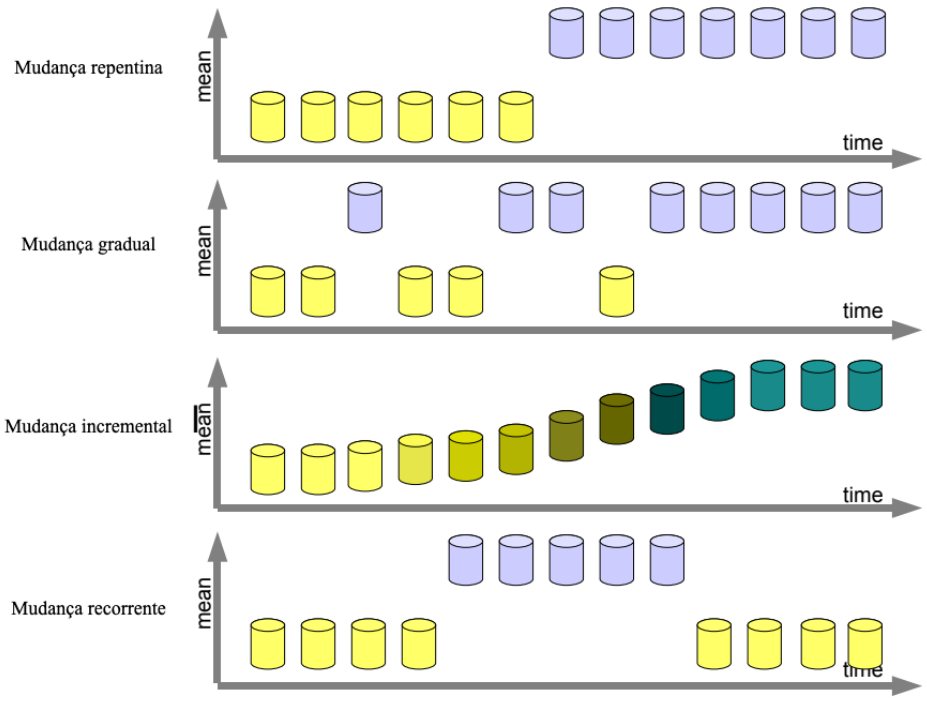
Um exemplo da ocorrência de mudança de conceito em situações reais é referente aos padrões de preferências de compras de clientes que muitas vezes podem alterar com o tempo, dependendo do dia da semana, avaliação de alternativas, inflação e outros fatores (NUNES, 2017). Outro domínio real são os ambientes de *IoT (Internet of Things)* não estacionários, onde a análise de fluxo de dados muitas vezes enfrenta desafios de mudança de conceito em que as distribuições de dados mudam ao longo do tempo.

Segundo Lu et al. (2018), a mudança de conceito pode ser caracterizada em quatro tipos:

- **Repentina**: onde um determinado conceito é instantaneamente alterado, sendo substituído por outro;
- **Gradual**: ocorrem alternâncias entre os conceitos, com um novo conceito substituindo gradualmente o antigo ao longo do tempo;
- **Incremental**: as mudanças ocorrem de forma mais lenta, com o conceito sendo inserido de forma incremental;
- **Recorrente**: é aquela quando um conceito que existia anteriormente e foi extinto reaparece depois de algum tempo.

A Figura 7 resume os tipos de mudança existentes.

Figura 7 – Tipos de mudança de conceito



Fonte: Adaptado de Žliobaitė (2010).

Desse modo, visando lidar com os diversos tipos de mudanças de conceito em fluxo de dados, de acordo com Bose et al. (2013), as soluções de mudança de conceito devem se concentrar em duas direções: como detectar desvios (mudanças); e como adaptar o modelo preditivo aos desvios.

Segundo Krawczyk et al. (2017a), a detecção de mudanças de conceito pode ser feita de duas maneiras:

- **Ativa**: As abordagens ativas são projetadas para detectar mudanças de conceito usando diferentes tipos de detectores. Se existir desvio de conceito, o modelo é atualizado (MAHDI et al., 2020).
- **Passiva**: As abordagens passivas atualizam continuamente o modelo sempre que novos dados se tornam disponíveis, independentemente se a mudança está ocorrendo ou não (MAHDI et al., 2020).

Neste trabalho, serão explorados diferentes métodos de detecção ativa de mudança de conceito, cuja abordagem e funcionamento são descritos na subseção 2.2.2.

2.2.2 Métodos de detecção ativa de mudança de conceito

A detecção refere-se às técnicas e mecanismos que caracterizam e quantificam a mudança de conceito por meio da identificação de pontos de mudança ou intervalos de mudança de tempo (BASSEVILLE; NIKIFOROV et al., 1993). A Figura 8 aborda uma estrutura geral de como funcionam os *frameworks* de detecção de mudança de conceito.

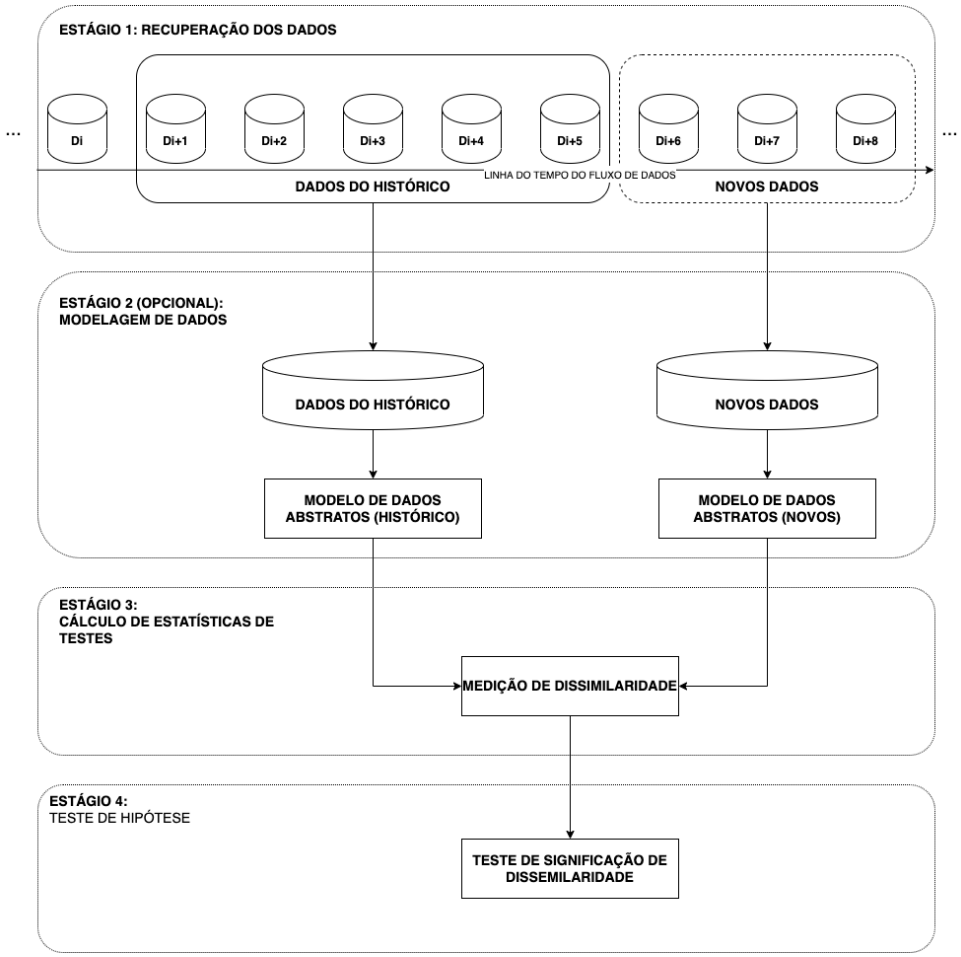
No Estágio 1 o objetivo é recuperar blocos de dados do fluxo de dados, enquanto no Estágio 2 é abstraído dos dados recuperados os principais recursos que mais impactam um sistema se eles sofrem com mudança de conceito. Adiante, no Estágio 3 é calculada a medida de dissimilaridade ou estimativa de distância. Dessa forma, é quantificada a gravidade da mudança de conceito, formando estatísticas de teste para o teste de hipótese. A Fase 4 (Teste de Hipótese) usa um teste de hipótese específico para avaliar a significância estatística da mudança observada no Estágio 3, ou o valor-p.

As estatísticas de teste adquiridas no Estágio 3 não fazem sentido para detecção de mudança de conceito sem a execução do Estágio 4 porque elas não podem determinar o intervalo de confiança para a mudança de conceito, ou seja, quão provável a mudança é causada por uma mudança de conceito e não por um ruído ou uma amostra aleatória (LU et al., 2018).

2.2.2.1 ADWIN

O detector Adaptive Sliding Window (ADWIN) monitora uma sequência de entradas de valor real usando janelas deslizantes (BARDDAL, 2019). Ele usa duas janelas: (i) uma janela de referência e (ii) uma janela de teste. A janela de referência ajuda a calcular a taxa de erro e outros parâmetros quando não há alteração e o valor de saída é tomado como referência. A

Figura 8 – Diagrama geral da detecção de mudança de conceito



Fonte: Adaptado de Lu et al. (2018).

janela de teste representa o novo lote de dados e o algoritmo monitora sua taxa de erro na detecção da mudança de conceito (MANICKASWAMY; BHUVANESWARI, 2020). O ADWIN identifica a ocorrência de mudança de conceito por uma grande diferença entre as médias das duas sub-janelas. Uma vez que um ponto de mudança é detectado, todas as amostras de dados antigas antes desse ponto de tempo de mudança são descartadas (YANG; MANIAS; SHAMI, 2021).

Além disso, como parâmetros o ADWIN não exige que os usuários definam antecipadamente o tamanho das janelas comparadas, apenas é configurado o valor de confiança $\delta \in (0, 1)$ para ajustar o valor de sensibilidade da detecção de mudança de conceito (GRULICH et al., 2018).

Sejam n_0 e n_1 os tamanhos de W_0 e W_1 e n o tamanho de W (janela deslizante), então $n = n_0 + n_1$. Ainda, μ_{w_0} e μ_{w_1} sejam os valores médios em W_0 e W_1, e μ_{w_0} e μ_{w_1} seus valores esperados. Para obter uma performance rigorosa, é definido no algoritmo:

$$m = \frac{1}{\frac{1}{n_0} + \frac{1}{n_1}}; \tag{1}$$

$$\delta' = \frac{\delta}{n}; \tag{2}$$

$$\varepsilon_{cut} = \sqrt{\frac{1}{2m}.\ln\frac{4}{\delta'}} \tag{3}$$

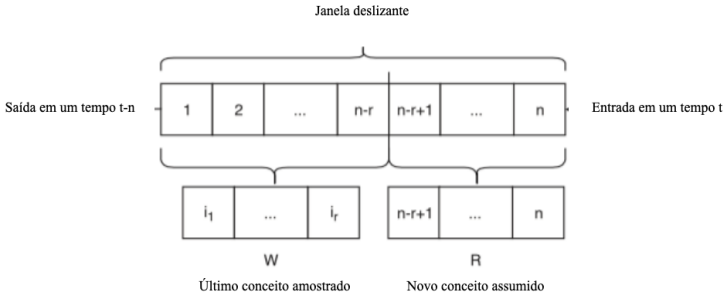
O teste estatístico para diferentes distribuições em W_0 e W_1 simplesmente verifica se a média observada em ambas as sub janelas difere por mais do que o limite de corte ε_{cut} (BIFET; GAVALDA, 2007).

O método de detecção ativa ADWIN pode detectar efetivamente desvios graduais, pois a janela deslizante pode ser estendida para comportar um grande número de instâncias do histórico para identificar mudanças de longo prazo (YANG; MANIAS; SHAMI, 2021). Além disso, o ADWIN proporciona uma rigorosa garantia do seu desempenho, sob a forma de limites nas taxas de falsos positivos e falsos negativos (GONÇALVES-JR et al., 2014).

2.2.2.2 KSWIN

KSWIN é um método recente para detecção de mudança de conceito que é baseado no teste estatístico de Kolmogorov-Smirnov (KS) (LOPES; REID; HOBSON, 2007). O teste de KS é um teste não paramétrico e, portanto, não precisa de nenhuma suposição da distribuição de dados subjacente. Semelhante ao ADWIN, o KSWIN também implementa as duas funções: adicionar os novos dados unidimensionais na janela deslizante local; e a segunda função é a que informa se o desvio foi detectado ou não. A primeira janela tem um tamanho fixo e, portanto, antes de adicionar qualquer novo dado nessa janela, os dados antigos são removidos e os novos são adicionados na fila da janela deslizante (TOGBE et al., 2021). A Figura 9 exibe esse processo.

Figura 9 – Detecção de mudança de conceito pelo método KSWIN.



Fonte: Adaptado de Togbe et al. (2021).

Para o KSWIN, a janela deslizante local L é dividida em duas partes. A primeira chamada de R contém o parâmetro predefinido r com os dados mais recentes que chegaram:

$$R = \{x_i \in L\}_{i=n-r+1}^{n} \tag{4}$$

A segunda janela W seleciona uniformemente os dados mais antigos em L. A distribuição uniforme usada é:

$$U(x|1,n-r) = \frac{1}{n-r} \tag{5}$$

Portanto,

$$W = \{x_i \in L | i < n-r+1, p(x) = U(x_i|1,n-r)\} \tag{6}$$

A mudança de conceito é detectada pelo KSWIN quando a $dist(R,W) > \sqrt{-\frac{\ln(\alpha)}{r}})$, onde α é a probabilidade para a estatística do teste KS.

2.2.2.3 DDM

O *DDM (Drift Detection Method)* monitora a taxa de erro, assumindo que ela diminuirá enquanto o número de instâncias aumentar e a distribuição de dados for estacionária no processo de aprendizagem incremental (YAN, 2020). Ele usa o conceito de distribuição binomial, e quando a distribuição de dados é estável o erro do modelo diminui com o tempo e vice-versa (ABBASZADEH; AMIRI; KHANTEYMOORI, 2015). O método possui dois modos: o nível de aviso, no qual os dados de entrada são armazenados na memória temporária, e o nível de mudança, em que o modelo do algoritmo funciona reaprendendo a partir dos dados na memória temporária (KRAWCZYK et al., 2017b).

O DDM foi o primeiro algoritmo a definir um nível de alerta e um nível de mudança para detecção de mudança de conceito. Quando uma nova instância de dados fica disponível para avaliação, o DDM detecta se a taxa de erro geral dentro da janela de tempo aumentou significativamente. Se o nível de confiança da taxa de erro observada atingir o nível de aviso, o DDM começa a construir um novo aprendiz e continua usando o antigo aprendiz para previsões. Se a mudança atingiu o nível de mudança de conceito, o antigo aprendiz será substituído pelo novo aprendiz para outras tarefas de previsão (LU et al., 2018).

O DDM funciona melhor na detecção de mudanças em fluxos com mudança repentina, ao contrário dos fluxos em que gradualmente os conceitos podem passar sem acionar o nível de mudança (GAMA et al., 2014a).

2.2.2.4 Page Hincley

O detector *Page-Hinckley* (PHT) é uma abordagem acumulativa sem memória para monitorar o desempenho dos aprendizes ao longo do tempo. O PHT é uma adaptação da detecção de mudança abrupta na média de um sinal gaussiano, e funciona acumulando a diferença entre os valores observados e sua média até o momento atual (BARDDAL, 2019). Sempre que a média extrapola um limite definido como parâmetro um desvio é sinalizado.

O teste de PHT considera uma variável acumulativa m_T, definida como a diferença acumulada entre os valores observados e suas médias até o momento atual:

$$m_{t+1} = \sum_{1}^{t}(x_t - x_t + \alpha) \tag{7}$$

onde, $x = \frac{1}{t}\sum_{l=1}^{t} x_l$ e α corresponde a magnitude das mudanças que são permitidas.
O menor valor dessa variável é também computado pela seguinte fórmula:

$$M_T = min(m_t, t = 1...T) \tag{8}$$

Então, o teste monitora a diferença entre M_T e m_T:

$$PHT = m_T - M_T \tag{9}$$

Quando a diferença é maior que o valor limite definido por δ, é disparado um alarme de mudança de conceito na distribuição.

2.2.3 Métodos de Validação e Medidas de Avaliação

Diversas técnicas foram criadas e adaptadas ao longo dos anos para medir a eficiência de algoritmos de classificação e regressão. Por isso, um conceito amplamente aceito é escolher a estratégia de avaliação do desempenho baseado no objetivo da aplicação.

Os modelos de previsão atribuem rótulos a uma determinada instância que quando comparado com o rótulo real pode ser:

- ***True Positives (TP)***: Refere-se à quantidade de instâncias verdadeiras previstas corretamente;
- ***False Negatives (FN)***: Refere-se à quantidade de instâncias falsas previstas incorretamente;
- ***True Negatives (TN)***: Refere-se à quantidade de instâncias falsas previstas corretamente;
- ***False Positives (FP)***: Refere-se à quantidade de instâncias verdadeiras previstas incorretamente.

Com base nisso, é possível utilizar as seguintes relações para se analisar um classificador:

- ***True Positive Rate (TPR)***: é a quantidade de classificações verdadeiras feitas corretamente dividido pelo mesmo valor somado a quantidade de classificações falsas incorretas.

$$TPR = \frac{TP}{TP + FN} \tag{10}$$

- ***False Positive Rate (FPR)***: é a quantidade de classificações verdadeiras previstas incorretamente sobre o mesmo valor somado com a quantidade de verdadeiros negativos.

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

- ***Matthews Correlation Coefficient (MCC)***: é a única taxa de classificação binária que consegue produzir uma pontuação alta se o preditor foi capaz de obter bons resultados nas quatro categorias da matriz de confusão (CHICCO; JURMAN, 2020).

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{12}$$

Krawczyk et al. (2017a) destaca os meios mais populares de se avaliar a capacidade preditiva de um algoritmo de classificação:

- **Acurácia**: é a proporção das previsões corretas feitas pelo algoritmo;

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

- **Sensibilidade**: diz respeito a precisão de uma determinada classe;

$$Sensibilidade = \frac{TP}{TP + FN} \tag{14}$$

- **Média G**: similar a sensibilidade mas voltado a fluxos de dados desbalanceados, que consiste em uma ou mais classes predominantes no decorrer do fluxo;

- **Estatística Kappa**: considera a acurácia e a probabilidade de um classificador aleatório fazer uma predição correta;

- **Estatística Kappa Generalizadas**: similar a Estatística Kappa, mas voltada a fluxos de dados desbalanceados.

Para a regressão, a predição resulta em um valor numérico contínuo, envolvendo assim um grau de incerteza quanto a exatidão do valor predito. Com isso, ao invés de focar no valor exato, o objetivo é minimizar a distância entre o valor predito (y') e o valor real (y). Algumas métricas para avaliar a regressão são:

- **Erro Médio Absoluto ou *Mean Absolute Error* (MAE)**: refere-se à magnitude da diferença entre a previsão de uma observação (y') e o valor real dessa observação (y):

$$\frac{1}{n} \sum_{i-1}^{n} |y - y'| \tag{15}$$

- **Erro Médio Quadrado ou *Mean Squared Error* (MSE)**: é comumente usado para verificar a acurácia de modelos e dá um maior peso aos maiores erros.

$$\frac{1}{n}\sum_{i-1}^{n}|y-y'|^2 \tag{16}$$

- R^2: R-quadrado é uma medida estatística de quão próximos os dados estão da linha de regressão ajustada. Também é conhecido como coeficiente de determinação, ou coeficiente de determinação múltipla para regressão múltipla. R-quadrado está sempre entre 0 e 100 %, sendo que 0% indica que o modelo não explica nenhuma variabilidade dos dados de resposta em torno de sua média e 100% indica que o modelo explica toda a variabilidade dos dados de resposta em torno de sua média.

- **Erro Médio Absoluto Percentual ou *Mean Absolute Percentage Error* (MAPE)**: é uma medida de quão preciso é um algoritmo de regressão. Ele mede essa precisão como uma porcentagem e é a média de todos os erros absolutos percentuais, fornecendo uma indicação do tamanho médio do erro.

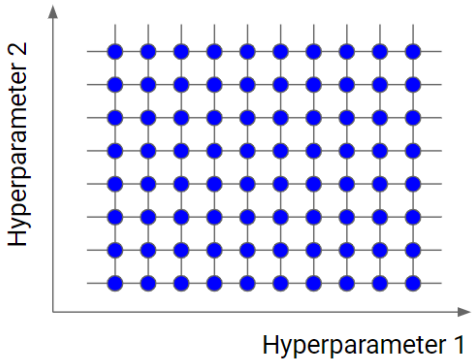
2.2.4 Ajuste automatizado de Parâmetros usando *Grid Search*

Ao se utilizar um algoritmo de aprendizado de máquina é necessário definir os melhores valores para seus hiperparâmetros a fim de obter os modelos de predição mais acurados. Portanto, no aprendizado de máquina, a otimização de hiperparâmetros é o problema de escolher um conjunto de hiperparâmetros em busca de melhores soluções para um algoritmo de aprendizado. Entretanto, definir a combinação dos melhores valores para todos os hiperparâmetros de um algoritmo de classificação ou regressão não é uma tarefa trivial, pois envolve o teste de uma infinidade de combinações de valores. Quando feito manualmente e de forma empírica, esse processo pode resultar em valores enviesados. Portanto, para minimizar esse problema podem ser utilizados algoritmos de busca que automatizam essa tarefa. Dentre eles, pode-se destacar o *Grid Search*.

A maneira tradicional de realizar a otimização de hiperparâmetros é a pesquisa em grade ou *Grid Search*, que é uma busca exaustiva que é realizada nos valores de parâmetros específicos de um estimador que simplesmente faz uma pesquisa completa sobre um determinado subconjunto do espaço de hiperparâmetros do algoritmo de treinamento (LIASHCHYNSKYI; LIASHCHYNSKYI, 2019). A Figura 10 ilustra o espaço de pesquisa do *Grid Search*. O *Grid Search* recebe como entrada o algoritmo a ser testado e um conjunto de valores para cada um de seus hiperparâmentros, e faz uma busca completa sobre as possíveis combinações. Em outras palavras, o algoritmo faz uma busca força bruta completa que normalmente leva um tempo considerável de execução.

O algoritmo de *Grid Search* sofre com a dimensionalidade, ou seja, com a quantidade de hiperparâmetros a serem configurados, mas geralmente é adaptado para execução paralela,

Figura 10 – Exemplo do espaço de pesquisa do Grid Search



Fonte: (LIASHCHYNSKYI; LIASHCHYNSKYI, 2019).

porque as configurações de hiperparâmetros que ele avalia são tipicamente independentes umas das outras (BERGSTRA; BENGIO, 2012).

2.3 TRABALHOS RELACIONADOS

Os trabalhos relacionados foram selecionados com base em temáticas envolvendo o *AXGBoost*, regressão de fluxo de dados e também em estratégias para lidar com a mudança de conceito. Esta seção apresenta o resumo da revisão da literatura que foi realizada.

A metodologia utilizada para encontrar artigos pertinentes ao tema se baseou na pesquisa nas principais bibliotecas digitais, como ACM, IEEE e *Google Scholar*. As palavras-chave utilizadas foram "*regression data stream*", "*concept drift*"e "*drift detectors*".

Chen e Guestrin (2016) propôs o algoritmo XGBoost, baseado no aprendizado de máquina escalável para *Boosting* de árvore. Segundo os autores, o XGBoost executa até dez vezes mais rápido que as soluções populares existentes em uma única máquina e escala para bilhões de exemplos em distribuição ou configurações de memória limitada. As principais contribuições do artigo foram: desenvolvimento de uma árvore de ponta a ponta altamente escalável com sistema de *Boosting*; introdução de um novo algoritmo com reconhecimento de esparsidade para aprendizado de árvores paralelas; e por fim, uma estrutura de bloco com reconhecimento de *cache* eficaz para aprendizado em árvore fora do núcleo.

Montiel et al. (2020) propuseram o algoritmo chamado AXGB *(Adaptive eXtreme Gradient Boosting)*, como uma adaptação do XGBoost, para a classificação de fluxo de dados com mudança de conceito. A ideia central do AXGB é a criação/atualização incremental do conjunto de classificadores, ou seja, aprendizes fracos são treinados em mini-lotes de dados e então adicionados ao conjunto. O tamanho máximo do conjunto é fixo, mas quando ele é atingido os classificadores mais antigos são descartados para dar lugar aos novos modelos para garantir o bom desempenho do conjunto. A atualização do conjunto pode ser feita de duas formas: *Push*: com modelos novos sendo adicionados na mesma medida que os antigos são removidos; e *Replacement*, os modelos mais antigos são substituídos pelos novos modelos. Segundo os autores, a partir dos testes realizados, o método de *Replacement* apresentou o melhor desempenho em termos de acurácia, tempo de treinamento e uso de memória.

Bonassa (2021) propôs uma adaptação ao AXGB (MONTIEL et al., 2020) com intuito de diminuir o seu tempo de treinamento, eliminando da proposta original o conjunto de classificadores. Para resolver esse problema, foi utilizada uma característica presente no XGBoost que permite salvar o modelo que está sendo treinado. Dessa forma, sempre que a janela deslizante de entrada de dados está cheia, o modelo treinado salvo é carregado e atualizado com os dados do novo lote de treinamento, e salvo posteriormente. Utilizando essa estratégia, o modelo se torna incremental sem a necessidade de utilizar um conjunto de modelos e também evitando a super-especialização. Além disso, Bonassa (2021) também adaptou o AXGB para o aprendizado semi-supervisionado. Essa adaptação foi feita pela implementação de um *wrapper* para pseudo-rotulagem dos dados não classificados utilizando o algoritmo KNN. Em relação aos testes, a acurácia permaneceu a mesma em comparação ao AXGB original proposto por Montiel et al. (2020), porém com tempo de execução até 10 vezes menor que o original. Isso ocorreu devido à

retirada do conjunto de classificadores utilizados na proposta original do AXGB.

Liao e Wang (2018) elaborou um trabalho envolvendo a regressão em fluxo de dados com adaptação a mudança de conceito por meio da criação de um novo algoritmo, denominado *Rival Learner*. O algoritmo utiliza um conjunto de classificadores e possui dois submodelos, um baseado em amostras históricas, chamado de modelo global, e o outro baseado em amostras de janelas deslizantes atuais, chamado de modelo local. Esses dois modelos competem entre si para ver quem tem o melhor desempenho e o vencedor assume como modelo global. Nos testes realizados pelos autores, quando ocorre um desvio de conceito, o algoritmo proposto mantém o modelo efetivamente atualizado sem necessidade da detecção ativa, mesmo com a necessidade de inicializar o submodelo com frequência, o desempenho do modelo também se mantém eficaz.

Yang, Manias e Shami (2021) descreveu um *framework* de conjuntos de classificadores para adaptação à mudança de conceito em fluxo de dados por meio da união de detectores ativos chamado *Performance Weighted Probability Averaging Ensemble* (PWPAE). O *framework* proposto é baseado no aprendizado de um conjunto de modelos que usa as combinações de dois métodos de detecção, ADWIN e DDM, e dois métodos de adaptação à deriva, *Adaptive Random Forest* (ARF) e *Streaming Random Patches* (SRP), para construir aprendizes básicos. Os aprendizes são ponderados de acordo com seu desempenho em tempo real e integrados para construir um modelo robusto. Os resultados apresentados pelos autores mostram que o algoritmo alcançou altos índices de acurácia comparado com os trabalhos de última geração.

Boulegane et al. (2020) introduziu uma nova abordagem para previsão de séries temporais em fluxo de dados usando o método de seleção dinâmica do conjunto. A seleção de conjunto visa escolher um subconjunto de um conjunto de modelos considerados os melhores preditores para uma instância. No trabalho foi formulado uma tarefa de Regressão *Multi target* onde um meta-aprendiz é responsável por simultaneamente prever o erro de cada modelo base no conjunto, dada uma instância de teste. Ao contrário dos métodos existentes, que não lidam com as dependências entre os modelos base, o método explora as informações de dependência em uma fase inicial ao prever os erros.

Gomes et al. (2018) propuseram uma adaptação ao *Adaptive Random Forest* (ARF) para a tarefa de regressão em fluxo de dados, chamado ARF-Reg. O algoritmo funciona de modo a possuir uma votação responsável por realizar a média das previsões individuais para obter a previsão final, juntamente com o detector de mudança de conceito ativa ADWIN. A diversidade também é uma característica importante induzida na floresta, treinando as árvores em diferentes sub-conjuntos de dados e limitando as decisões de divisão a um subconjunto de recursos selecionado aleatoriamente dos recursos de entrada originais. Por sua vez, o aprendiz base é uma árvore de regressão, com o algoritmo de FIMT-DD. Finalmente, a dinâmica de atualização no ARF-Reg depende tanto interna quanto externamente dos detectores de mudança de conceito para cada árvore e pelo crescimento de árvores no *background* quando um aviso é detectado. Como resultado, o ARF-Reg obteve taxas de erro consideravelmente pequenas, principalmente em cenários no mundo real, mostrando assim a sua eficácia.

O artigo escrito por Altman (1992) realizou a criação da regressão baseada em vizinhos (KNN ou K-Nearest Neighbors), que pode ser usada nos casos em que os rótulos de dados são variáveis contínuas em vez de discretas. O rótulo atribuído a um ponto de consulta é calculado com base na média dos rótulos de seus vizinhos mais próximos.O algoritmo KNN é um tipo de método supervisionado e não paramétrico. Em sua versão mais básica, o uso do KNN requer três elementos essenciais: um conjunto de objetos rotulados, uma métrica de distância ou similaridade para calcular a distância entre objetos e o valor de k (número de vizinhos mais próximos) (LI; QIU; LIU, 2017). O algoritmo utilizado nesse trabalho para comparação com os modelos propostos foi retirado por meio da biblioteca scikit-learn, o KNeighborsRegressor. Segundo a própria documentação, o princípio por trás dos métodos do vizinho mais próximo é encontrar um número predefinido de amostras de treinamento mais próximas em distância do novo ponto e prever o rótulo a partir delas. O número de amostras pode ser uma constante definida pelo usuário (aprendizado de k-vizinho mais próximo). A distância pode, em geral, ser qualquer medida métrica: a distância euclidiana padrão é a escolha mais comum. Os métodos baseados em vizinhos são conhecidos como métodos de aprendizado de máquina não generalizantes, pois eles simplesmente "lembram" todos os seus dados de treinamento (PEDREGOSA et al., 2011).

O artigo Domingos e Hulten (2000) propôs um algoritmo eficiente para mineração de árvores de decisão a partir de fluxos de dados em constante mudança, denominado de VFDT (Very Fast Decision Tree). Esse algoritmo foi nominado o primeiro utilizando a estrutura Hoeffding Tree. A partir do mesmo, outros algoritmos foram criados e adaptados, inclusive portados para regressão, que é o caso do Hoeffding Tree Regressor (HTR). O HTR é uma adaptação do algoritmo de árvore incremental de mesmo nome para classificação. Da mesma forma que sua contraparte de classificação, o HTR usa o limite de Hoeffding para controlar suas decisões divididas. Diferentemente do algoritmo de classificação, o HTR baseia-se no cálculo da redução da variância no espaço alvo para decidir entre os candidatos divididos. Quanto menor a variância em seus nós folha, mais homogêneas são as partições. Em seus nós folha, o HTR ajusta os modelos lineares perceptron ou usa a média da amostra como preditor (MONTIEL et al., 2018). Além do HTR, foi criado também um algoritmo portando o mesmo para fluxos de dados lidando com mudança de conceito, que é o caso do Hoeffding Adaptative Tree Regressor (HTRA). O HTR utiliza o detector de mudança de conceito ADWIN para detectar mudanças de conceito e PERCEPTRON para realizar previsões.

O artigo de Junior et al. (2014) e a dissertação de mestrado com um dos autores Santos (2015) apresentam estudos de comparação para detectores de mudança de conceito. Os estudos avaliaram oito detectores de desvio de conceito diferentes, entre eles DDM, PHT, STEPD e ADWIN, realizando testes usando conjuntos de dados sintéticos que sofrem mudanças de conceito abruptas e graduais. Os resultados evidenciaram algumas características próprias de cada detector, sendo as principais: (i) ADWIN: foi o método mais rápido. Em relação ao tempo de execução e ao uso de memória, ADWIN foi, na média, o mais rápido e também o mais econômico no uso da memória. O ADWIN também proporciona uma rigorosa garantia do seu

desempenho, sob a forma de limites nas taxas de falsos positivos e falsos negativos. (ii) DDM: apresentou as taxas mais baixas de falsos positivos juntamente com o ADWIN. O método DDM apresentou a melhor precisão média para detecção de mudanças de conceito, com destaque para conjuntos de dados que sofrem de mudanças de conceito graduais.

E por fim, também foi citado essa mesma monografia de Souza (2022), que desenvolveu uma adaptação da biblioteca AXGB para portar o algoritmo de classificação para regressão de fluxo de dados, utilizando uma técnica formada por detectores ativos de mudança de conceito: ADWIN, DDM e KSWIN.

2.3.1 Considerações sobre os Trabalhos Relacionados

A Tabela 1 apresenta uma comparação entre os trabalhos relacionados revisados na seção 2.3. Para melhor compreensão sobre os critérios utilizados na comparação, a seguir é apresentada uma breve descrição dos critérios utilizados:

1. **Tipo de aprendizagem**: se o algoritmo visa realizar classificação ou regressão;
2. **Fluxo**: se o algoritmo suporta classificação ou regressão de fluxo de dados;
3. **Mudança de conceito**: se o algoritmo trata o problema de mudança de conceito em fluxos;
4. **XGBoost**: se o algoritmo utiliza o XGBoost como base para a classificação ou regressão;

Tabela 1 – Resumo dos trabalhos relacionados

Trabalho	Tipo de aprendizagem	Fluxo de dados	Mudança de conceito	XGBoost
Chen e Guestrin (2016)	Classificação	Não	Não	Sim
Montiel et al. (2020)	Classificação	Sim	Substituição do modelo + ADWIN	Sim
Bonassa (2021)	Classificação	Sim	Substituição do modelo + ADWIN	Sim
Liao e Wang (2018)	Regressão	Sim	Adaptação passiva a mudança de conceito	Não
Yang, Manias e Shami (2021)	Classificação	Sim	União de detectores ativos de mudança de conceito	Não
Boulegane et al. (2020)	Regressão	Sim	Não	Não
Gomes et al. (2018)	Regressão	Sim	ADWIN	Não
Altman (1992)	Regressão	Sim	Não	Não
Domingos e Hulten (2000)	Regressão	Não	Não	Não
Santos (2015)	Regressão/Classificação	Sim	Estudo comparativo (DDM, ADWIN)	Não
Souza (2022)	Regressão	Sim	DDM, ADWIN e KSWIN	Sim

A Tabela 1 apresenta um resumo dos trabalhos relacionados. É possível notar uma escassez de propostas que envolvam a regressão de fluxos de dados utilizando a biblioteca XGBoost

Chen et al. (2016) propôs o XGBoost para classificação e regressão de conjuntos de dados estacionários. Com a adaptação do XGBoost para fluxo de dados feita no trabalho de Montiel et al. (2020), ele pode também ser utilizado em conjuntos de dados não estacionários, pois ele apresenta a possibilidade de lidar com a mudança de conceito, por meio da utilização do detector ativo ADWIN.

Bonassa (2021), por sua vez, propôs uma adaptação ao trabalho de Montiel et al. (2020) para o aprendizado semi-supervisionado, com a classificação rápida realizada por um único classificador, ao invés de utilizar um conjunto como proposto por Montiel et al. (2020). A proposta de Bonassa (2021) melhorou consideravelmente a velocidade de treinamento do modelo sem diminuir a acurácia do algoritmo, também evitando a super-especialização do modelo.

Liao e Wang (2018) é um dos trabalhos mais recentes envolvendo a regressão em fluxo de dados. O algoritmo desenvolvido utiliza uma abordagem incremental para adaptação a mudança de conceito de forma passiva.

Na temática de detecção ativa da mudança de conceito, o trabalho de Yang, Manias e Shami (2021) utiliza diferentes detectores, e tira proveito de suas particularidades e complementariedades, de forma que a depender do tipo de mudança de conceito, alguns detectores específicos podem ser mais propícios para detectar a mudança e atualizar o modelo. Ademais, o artigo Junior et al. (2014) e a dissertação de mestrado com um dos autores Santos (2015) apresentam uma discussão completa dos pontos positivos e negativos da utilização dos principais detectores ativos de mudança de conceito.

Os trabalhos de Gomes et al. (2018) e Boulegane et al. (2020) são os principais trabalhos sobre regressão em fluxos de dados, porém os algoritmos utilizados por cada um deles não propôs uma alternativa eficaz para adaptação e detecção de mudança de conceito. Por essa razão, ainda é uma área com poucos trabalhos e mais pesquisas sobre essa temática precisam ser realizadas.

2.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Nesse capítulo foram revisados os principais conceitos necessários para o entendimento deste trabalho, sendo eles: aprendizado de máquina, classificação e regressão de dados, classificação e regressão de fluxo de dados, mudança de conceito, detectores ativos para mudança de conceito, XGBoost e métricas de avaliação.

O aprendizado de máquina é uma área que visa aprender automaticamente conceitos e reconhecer padrões sem necessariamente ser programado especificamente para uma dessas tarefas. Dentro dessa área, esses algoritmos possuem duas características principais, que são o *feedback* e objetivo do algoritmo. Os diferentes tipos de *feedback* são o aprendizado supervisionado, aprendizado não-supervisionado e o semi-supervisionado.

O objetivo dos algoritmos pode ser dividido em Classificação e Regressão. Enquanto a primeira é definida para previsão de rótulos categóricos, a segunda foca na previsão de valores numéricos. Por sua vez, dentro da classificação, existe a possibilidade da utilização de conjuntos de classificadores, conhecidos como *ensembles*. Um conjunto de classificadores inclui classificadores individuais mutuamente complementares, e essa abordagem vem ganhando destaque nos últimos anos dentro da área de aprendizado de máquina.

Com o surgimento de fluxos de dados, o XGBoost foi adaptado para suportá-los, porém originalmente proposta para utilizar um conjunto de classificadores, essa abordagem foi substituída por uma outra cujo algoritmo treina um único classificador. Contudo, o tema da mudança de conceito ainda não foi abordado de forma extensa na literatura sobre XGBoost.

Por ser uma característica de dados não estacionários, a detecção da mudança de conceito também foi estudada nos fluxos de dados. A mudança de conceito pode acontecer de forma repentina, gradual, incremental ou recorrente. Suas formas de detecção se dividem em ativas e passivas. Como o trabalho em tela tem por objetivo explorar a detecção ativa, os detectores ADWIN, KSWIN, DDM e PHT foram abordados e estudados.

Com as fundamentações abordadas foi possível ampliar o conhecimento a respeito das técnicas que serão utilizadas na contribuição deste trabalho, bem como a motivação para sua utilização.

3 DESENVOLVIMENTO

Este capítulo apresenta a proposta de adaptação do algoritmo AFXGB (BONASSA, 2021) para realizar a tarefa de regressão de fluxos de dados. É importante destacar que o AFXGB foi originalmente proposto para realizar apenas a classificação binária de fluxo de dados. Além disso, também é proposta uma adaptação visando prover ao algoritmo uma abordagem ativa de detecção de mudança de conceito. Para isso, a união da aplicação de diferentes detectores ativos de mudança de conceito será explorada.

A Seção 3.1 esboça as adaptações projetadas sobre o algoritmo AFXGB original, tanto para realização de regressão como para detecção ativa de mudança de conceito. A Seção 3.3 exibe os parâmetros necessários na execução do algoritmo. A Seção 3.2 detalha o algoritmo do AFXGBReg proposto e as adaptações nele realizadas. Por fim, a Seção 3.4 explica o processo de implementação.

3.1 PROPOSTA

Como observado na revisão de trabalhos relacionado apresentada na Seção 2.3, a tarefa de regressão, apesar de ser um dos tópicos mais comuns na área de aprendizado de máquina, não é um tema amplamente abordado e estudado na análise de fluxos de dados. Portanto, este trabalho visa adaptar o algoritmo AFXGB para suportar regressão de fluxo de dados com detecção ativa de mudança de conceito.

O algoritmo AXGB, proposto por Montiel et al. (2020), utilizava um conjunto de classificadores, entretanto isso prejudicava consideravelmente o tempo de execução do algoritmo. Por isso, será utilizado como base do trabalho o algoritmo proposto por Bonassa (2021), que é uma adaptação ao AXGB chamada AFXGB que obteve menor tempo de treinamento mantendo a mesma acurácia obtida pelo AXGB original. Na proposta de Bonassa (2021), apenas um modelo XGBoost é treinado e atualizado de forma incremental. Entretanto, após esse classificador ultrapassar um limite de dados treinados, um outro classificador é treinado visando substituí-lo, evitando assim a super-especialização (*overfitting*) do primeiro.

Embora a estratégia incremental de atualização do modelo AXGB proposta por Bonassa (2021) consiga lidar com a detecção passiva de mudança de conceito, pois o modelo é atualizado com base em dados mais recentes, a atualização do modelo para mudanças abruptas de conceito não é adequadamente realizada devido a sua demora na percepção da mudança. Por isso, este trabalho propõe a inclusão de técnicas ativas de detecção de mudança de conceito ao algoritmo AFXGB, além da sua adaptação para suportar a regressão.

Como forma de suportar a detecção ativa de mudança de conceito, este trabalho propõe a utilização das seguintes técnicas: ADWIN, KSWIN e DDM. Esses detectores têm o objetivo de identificar as mudanças de conceito e acionar o mecanismo de atualização do modelo. Esse mecanismo consiste em diminuir o tamanho da janela de dados vinda do fluxo utilizada para atualizar o modelo, fazendo com que a atualização do modelo seja realizada com maior

frequência, resultando em uma adaptação à mudança de conceito mais rápida.

3.2 ALGORITMO

O pseudocódigo do algoritmo proposto pode ser visualizado no Algoritmo 1. As modificações implementadas no algoritmo AFXGB de Bonassa (2021) incluem a adição de uma etapa de detecção ativa de mudança de conceito, realizada pelo conjunto de detectores composto por: ADWIN, DDM e KSWIN, a troca da função para realização de regressão e a implementação de uma *flag* que possibilita a execução do algoritmo com um mecanismo de *reset* do tamanho da janela quando acontece a substituição do modelo corrente pelo modelo em treinamento.

O algoritmo recebe como entrada instâncias do fluxo compostas por x e y, sendo x o conjunto dos valores dos atributos e y o valor da variável objetivo da regressão. No algoritmo, quando a janela deslizante de dados vindos do fluxo está cheia (Linha 3), o algoritmo verifica se existe algum modelo de regressão XGBoost já treinado (Linha 4). Caso não exista, um regressor XGBoost é treinado (Linha 18). Caso já exista, outras condições são testadas com a proposta de verificar o tempo de vida do regressor (Linhas 5 a 11) e, na sequência, o regressor é carregado, treinado, atualizado e salvo (Linhas 13 a 16).

A adaptação para portar o XGBoost para regressão é realizada com a substituição da função objetivo do algoritmo para "reg:squarederror". Dessa forma, a métrica de erro avaliada no modelo passa a ser a Raiz Quadrada do Erro-Médio (RMSE). Além disso, a função de predição também foi adaptada para retornar valores do tipo ponto flutuante ao invés de valores discretos (rótulos) como no caso da classificação. Após o modelo ser atualizado, caso a *flag* de detecção ativa esteja ligada (Linha 20), cada um dos detectores de mudança de conceito é executado para identificar se houve uma mudança (Linhas 23 a 28), sendo que se um dos detectores detectar uma mudança os outros não precisam ser executados (Linha 28).

Os detectores foram incluídos de forma a se tornarem complementares uns aos outros. Ou seja, caso um detector não identifique a mudança de conceito, outro pode conseguir devido à complementariedade entre eles. Entretanto, para não aumentar demasiadamente e desnecessariamente o tempo de processamento da detecção de mudança, caso um dos detectores detecte a mudança os outros não precisaram ser executado (Linhas 24 a 28). Para esta implementação, foram escolhidos os detectores ADWIN, KSWIN e DDM. Para escolha final dos detectores, alguns pontos chave foram considerados:

- **Implementação por bibliotecas influentes na comunidade**: Os três detectores escolhidos para o trabalho são implementações disponibilizadas pela biblioteca pública scikit-multiflow do Python.
- **Melhor adaptabilidade a diferentes tipos de mudança de conceito**: Cada detector possui sua particularidade de adaptação e detecção aplicada a diferentes casos de mudança de conceito, portanto, são complementares.

Algoritmo 1: Adaptação ao AXGB com a união de detectores ativos

Entrada: $(x, y) \in$ Fluxo de dados

Data: w = quantidade máxima de exemplos no *buffer*, M = Modelo de regressores XGBoost, W = janela, SW = janela deslizante dos dados rotulados, *life_time* = Tempo de vida de cada regressor, *training_time* = tempo de treinamento de cada regressor, D = *array* de detectores, count = quantas vezes a janela foi executada

```
1  Realiza a predição de (x,y) usando o modelo XGBoost treinado;
2  Adicione (x,y) na janela (W)
3  se |W| > w então
4      se M <> Null então
5          se training_time >= (life_time - count) então
6              # Verifica se esta dentro do tempo de treinamento do
                 próximo regressor.
7              nextM = Treina o próximo regressor XGBoost;
8          se count >= life_time então
9              # Reinicia o contador e começa a utilizar o novo
                 regressor.
10             count = 0;
11             M = nextM;
12             se active_reset == 'S' então
                   tamanho_janela = 0;
13         M = Carrega o modelo anterior;
14         M' = Treina regressor XGBoost com W' usando M e adiciona a M;
15         M = M';
16         Salva o novo modelo M treinado;
17     senão
18         M = Treina o regressor XGBoost com W';
19         Salva o novo modelo M treinado;
20     se detect_drift == True então
21         # Calcula o erro absoluto.
22         error = abs(self.predict(X) - y);
23         para cada detector_ativo de D faça
24             se detectar_mudanca(detector_ativo, error) então
25                 # Se algum detector ativo de mudança de conceito
                     detectar alguma mudança com base no erro absoluto, o
                     tamanho da janela é zerado.
26                 tamanho_janela = 0;
27                 # Na primeira detecção de mudança de conceito por algum
                     detector o laço é quebrado, passando para próximas
                     instâncias
28                 break;
29     W = W - W';
30     count = count + 1; # conta o número de vezes que a janela reiniciou
31 retorna M;
```

- **Utilização de hiperparâmetros configuráveis**: Possibilidade de configurar e aplicar algoritmos exaustivos para testagem de melhores resultados.
- **Estudos comparativos**: Foram pesquisados artigos de comparação dos detectores ativos de mudança de conceito para descobrir em quais melhores casos os mesmos eram aplicados.

Além da escolha de quais detectores iriam integrar o algoritmo, também foi necessária uma análise considerando sua ordem de execução, bem como valores de hiperparâmetros para cada detector. Analisando as características dos detectores escolhidos, foi identificado por meio de experimentos e análises empíricas que a sensibilidade para detecção de mudanças de conceito do detector KSWIN obteve elevado número de detecções do tipo falso-positivo. Ou seja, o KSWIN detectou mudanças que não estavam acontecendo de fato. Portanto, considerando o aspecto relacionado à quantidade de mudanças de conceito detectadas de forma equivocada pelo detector KSWIN, ADWIN e DDM, os detectores foram escalonados da seguinte forma:

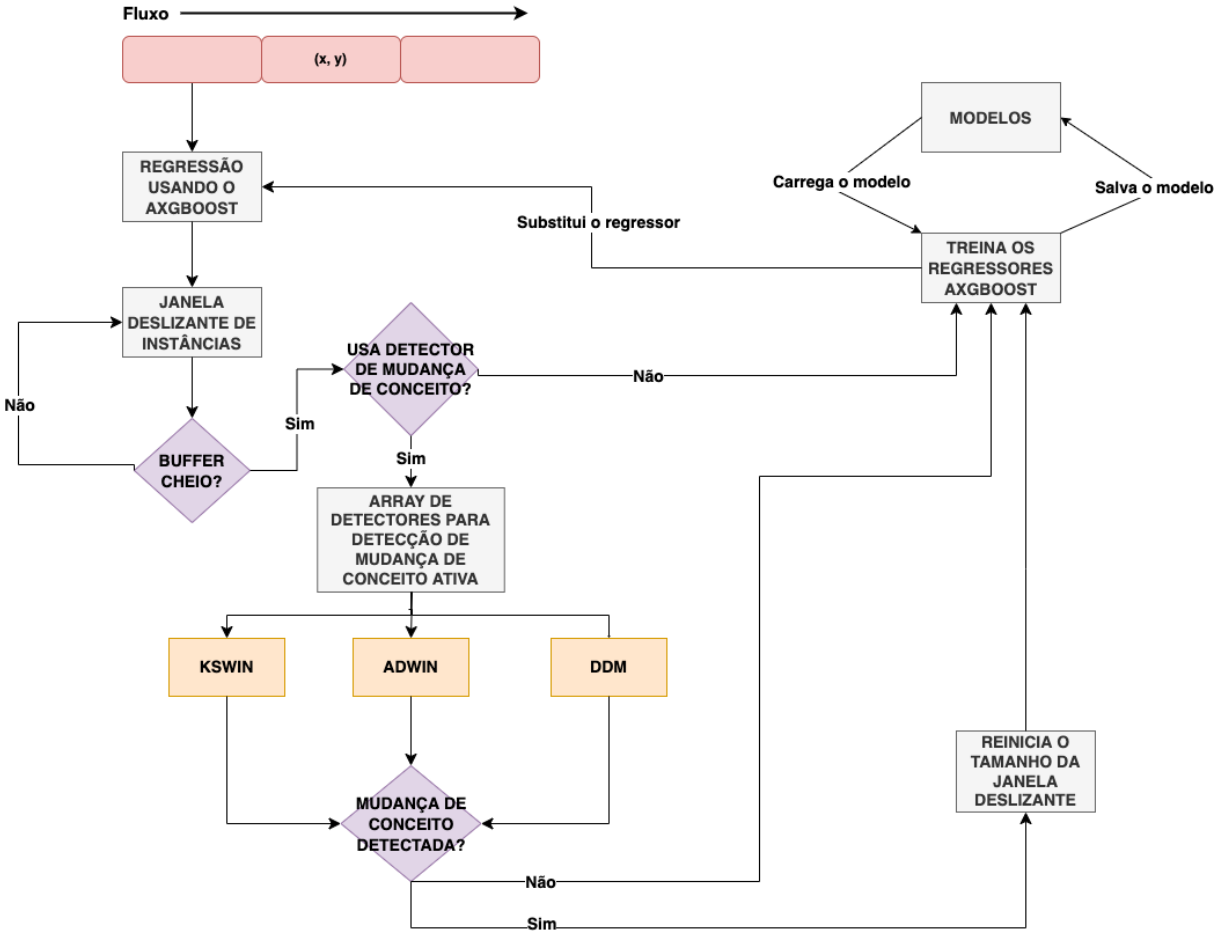
1. **ADWIN**: Foi escolhido como primeira opção de detecção devido a sua assertividade para mudanças de conceito recorrentes e graduais, enquanto mantém uma bom pertentual de detecção para mudanças abruptas;
2. **DDM**: Performance eficaz e neutra para mudanças de conceito abruptas;
3. **KSWIN**: Foi posto como última opção, pois caso uma mudança de conceito não seja detectada por nenhum dos outros detectores citados acima, o KSWIN possui alto percentual quantitativo de detecções para mudanças de conceito, principalmente abruptas, podendo considerar falsos positivos, ou seja, a identificação de mudanças de conceito em instantes onde a mesma não ocorre.

Ainda, além do escalonamento utilizado com os detectores, outra estratégia foi utilizada para trazer melhores resultados de maneira assertiva para as detecções: a *tunagem*/otimização dos hiperparâmetros de cada detector. O algoritmo utilizado para otimização foi o *Grid Search*, disponibilizado em Python pela biblioteca scikit-learn, o qual caracteriza-se por uma técnica de busca que calcula os valores ótimos dos hiperparâmetros.

De maneira semelhante ao Algoritmo 1, a partir da Figura 11 é exposto o fluxograma do algoritmo proposto. No diagrama pode-se destacar a implementação do *array* de detectores, onde a partir de uma variável *booleana* é verificado se a estratégia deve ser utilizada. Caso identificada a variável com valor *True* definida pelo usuário, a partir de qualquer mudança de conceito detectada por qualquer detector, é disparado o mecanismo para reinicialização da janela deslizante. Entretanto, caso não for detectado nenhuma mudança de conceito, o fluxo continua para treinar os regressores AXGB.

As adaptações propostas ao algoritmo neste capítulo e, principalmente, o método da união de detectores, têm como objetivo adaptar os modelos mais rapidamente a qualquer mudança de

Figura 11 – Fluxo de dados do projeto



Fonte: O autor.

conceito detectada, consequentemente visando aumentar a acurácia para classificação e diminuir o erro para regressão, sem aumentar consideravelmente o tempo de predição dos dados do fluxo e de treinamento do modelo.

É importante ressaltar a grande quantidade de parâmetros existentes no XGBoost. Portanto, é imprescindível alinhar todos os parâmetros da melhor forma possível para assim obter o resultado desejado.

3.3 PARÂMETROS

O XGBoost conta com diversos hiperparâmetros que precisam ser configurados para que ele possa obter o melhor desempenho possível para cada cenário de análise onde seja utilizado. O AFXGB de Bonassa (2021) utiliza alguns deles e inclui alguns próprios, sendo os dois últimos listados abaixo parâmetros adicionados neste trabalho e focados na utilização dos detectores ativos de mudança de conceito:

- **Taxa de aprendizado (eta)**: Um modelo XGBoost envolve a criação de árvores sequencialmente. As novas árvores são criadas para reduzir o erro das anteriores. Quanto mais

próximo de 0 menor o aprendizado, e inversamente quanto mais próximo de 1 maior ele é.

- **Profundidade máxima (*max_depth*)**: A profundidade máxima que uma árvore pode atingir. Quanto maior esse valor mais o modelo cresce em complexidade e consumo de memória.
- **Tempo de vida (*life_time*)**: Tempo de vida de cada classificador alternado. Esse tempo é baseado no número de vezes que a janela deslizante foi reiniciada.
- **Tempo de treinamento (*training_time*)**: Tempo de treinamento de cada classificador alternado. Esse tempo é baseado no número de vezes que a janela deslizante foi reiniciada.
- **Tamanho máximo da janela (*max_window_size*)**: Tamanho máximo do *buffer* que armazena os dados do fluxo. O algoritmo só atualiza os modelos quando o tamanho máximo é atingido.
- **Reset de janela na troca de regressores (*active_reset*)**: Booleano usado para identificar se a estratégia de *reset* do tamanho da janela na troca de regressores será utilizada ou não;
- **Mudança de conceito (*detect_drift*)**: Esse booleano determina se a mudança de conceito com o *array* de detectores vai ser aplicada durante a aprendizagem.

3.4 IMPLEMENTAÇÃO

As adaptações propostas ao AXGB de Bonassa (2021) foram implementadas a partir do código original obtido do próprio autor. O AXGB foi desenvolvido em linguagem de programação *Python*. O *Python* é uma linguagem de programação de alto nível e orientada a objetos, de tipagem dinâmica e forte. Sua utilização é amplamente difundida nas áreas de *Big Data* e Aprendizado de Máquina, possuindo diversas bibliotecas de código aberto disponibilizadas por sua comunidade, bem como o próprio AXGBoost (KADIYALA; KUMAR, 2018).

Uma das bibliotecas *Python* utilizadas é a scikit-multiflow. Ela contém vários pacotes com algoritmos e ferramentas de aprendizado de máquina para fluxo de dados, dentre elas, é possível destacar:

- **Mecanismos para geração de fluxos de dados com diferentes aspectos** como: *SEA Generator*, *Agrawal Stream Generator*, *Regression Generator*, *Random Tree Generator*, *ConceptDriftStream*.
- **Monitor com gráficos e métricas para comparação de diferentes algoritmos**.
- **Diferentes algoritmos de regressão** como: *Adaptive Random Forest Regressor*, *k-Nearest Neighbors regressor*, *Hoeffding Adaptive Tree regressor*, *Hoeffding Tree regressor*.
- **Detectores de mudança de conceito ativa** como: ADWIN, DDM e KSWIN.

4 ANÁLISE DOS RESULTADOS

Este capítulo apresenta os resultados dos testes feitos com o algoritmo completo. Esta etapa é fundamental para qualquer trabalho de pesquisa, pois é responsável por validar a proposta bem como seus pontos fortes e fracos. Com isso, a Seção 4.1 esboça a metodologia utilizada para a avaliação e a Seção 4.2 detalha os resultados obtidos.

4.1 METODOLOGIA DE AVALIAÇÃO

Para realizar a avaliação do algoritmo proposto, seis implementações foram consideradas. Esses seis algoritmos, dois são variações da abordagem proposta neste trabalho e quatro são algoritmos *benchmarks* encontrados na literatura. Segue abaixo uma breve descrição de cada um deles:

- AXGBReg_Dr: variação do algoritmo proposto neste trabalho, que utiliza classificadores alternados, união de detectores ativos de mudança de conceito e *reset* de janela deslizante sempre que uma mudança de conceito é detectada e quando o regressor corrente é substituído pelo que estava sendo treinado.
- AXGBReg_D: variação do algoritmo proposto neste trabalho, que utiliza classificadores alternados, união de detectores ativos de mudança de conceito e *reset* de janela **APENAS** nas detecções de mudança de conceito.
- HTR (Hoeffding Tree regressor): uma adaptação para regressão do algoritmo de árvore incremental de mesmo nome para classificação. O HTR usa a desigualdade de Hoeffding para controlar suas decisões de divisão de nós.
- HTRA (Hoeffding Adaptive Tree regressor): similar a implementação anterior, entretanto, ele incorpora os detectores de mudança de conceito ADWIN e PERCEPTRON (rede neural de camada única) para fazer previsões.
- KNN (k-Nearest Neighbors regressor): método de regressão não paramétrica onde as previsões são obtidas agregando os valores das amostras armazenadas de n_vizinhos mais próximos em relação a uma amostra de consulta.
- ARFReg (Adaptive Random Forest Regressor): adaptação para regressão do algoritmo existente para classificação, utilizando o detector de mudança de conceito ADWIN.

Os aspectos avaliados nesta experimentação incluem o MSE, o tempo de execução e a capacidade de adaptação do regressor a diferentes tipos de mudança de conceito. O método de avaliação selecionado é o *Prequential evaluation*, onde os dados são analisados de forma sequencial quando chegam ao regressor, antes de serem utilizados para atualizar os modelos.

As bases de dados utilizadas para os testes foram retiradas da biblioteca *scikit-multiflow* disponibilizadas por Montiel et al. (2018), sendo bases sintéticas geradas por meio dos métodos de geração *Hyperplane Generator* e *ConceptDriftStream*. A Tabela 2 apresenta as bases selecionadas para a avaliação.

Tabela 2 – Bases de dados

Nome	Instâncias	Qtd. Mudanças Conceito	Tipo da Base	Mudança
CDS_ABRUP	500000	10	Sintética	Abrupto
HYP	500000	1	Sintética	Incremental
CDS_GRAD	500000	3	Sintética	Gradual

Fonte: O autor.

No cenário utilizado para testes, apenas três bases de dados sintéticas foram escolhidas para simulação. Essa característica se deve a falta de bases de dados reais com dados contínuos disponíveis com números consideráveis de registros. Para a experimentação, foram utilizados 500.000 registros, porém não foram encontradas bases com número de registros similares. Entretanto, como consequência entende-se que o ranking criado para ordenação dos modelos poderia ter sido melhor explorado caso houvesse um número maior de bases com diferentes características.

Foram feitas 5 execuções para cada algoritmo e conjunto de dados para obter a média do Erro Quadrado Médio (MSE), tempo de treinamento, tempo de teste e tempo total. Logo, foram geradas 90 execuções e mantidos os respectivos arquivos de log, utilizando para os experimentos uma máquina com o seguinte ambiente: Intel® Core™ i7-11700 (8 cores, 16MB cache, até 4.9GHz); com 32GB (2 de 16GB) de memória (DDR4, UDIMM, memória non-ECC); com armazenamento 512 GB de M.2 Class 40 SSD; executando Ubuntu 20.04.4 LTS (CLI Server); Linux 5.4.0-113-generic.

4.1.1 Parâmetros

Antes de iniciar os testes é necessário definir os valores para os parâmetros que serão utilizados pelos algoritmos a fim de extrair o melhor desempenho deles. Os valores dos parâmetros foram definidos a partir dos trabalhos de Montiel et al. (2020) e Bonassa (2021), trabalhos dos quais o trabalho em tela utilizou como referências. Posteriormente, eles foram calibrados por meio de testes empíricos.

O primeiro parâmetro a ser configurado é o número de modelos gerados pelo XGBoost. O XGBoost é um método iterativo onde cada modelo gerado tem como objetivo melhorar o erro de predição dos modelos anteriores. Entretanto, se por um lado quanto mais modelos o regressor tiver menor será seu MSE, mais tempo de treinamento e de predição ele terá. Por isso, é necessário encontrar um equilíbrio entre a quantidade de modelos e o tempo necessário para treiná-los. Com base no trabalho de Bonassa (2021) e em testes empíricos realizados, constatou-se que o número adequado de modelos seria 30.

Outro fator importante é a taxa de aprendizado do modelo, que varia entre 0 e 1: valores próximos a 1 tendem a ocasionar um ganho de MSE do modelo com o passar do treinamento, pois o algoritmo faz correções muito grandes que podem prejudicar o aprendizado, porém diminui o tempo de processamento. Em contraste, valores próximos a 0 tendem a diminuir o MSE, mas o tempo de processamento aumenta consideravelmente. Com os testes feitos também por Bonassa (2021), o valor escolhido para o parâmetro taxa de aprendizado foi 0,3.

Para o tamanho da janela, ou seja, o *buffer* que armazena os dados a serem utilizados no treinamento, com base nos testes empíricos realizados, o número selecionado foi 1000 instâncias.

A profundidade máxima de uma árvore (*max_depth*) foi definida como 6 devido a uma pequena melhora na predição em comparação com os demais valores próximos.

O tempo de vida e de treinamento do modelo foram definidos como, respectivamente, 25 e 15. A partir de análises empíricas e trabalhos estudados, os valores foram definidos visando não comprometer a memória e velocidade do algoritmo.

Portanto, os valores selecionados para os parâmetros utilizados pelos algoritmos foram:

- **Classificadores**: 30
- **Taxa de aprendizado**: 0,3
- **Tamanho da janela**: 1000
- **Tempo de vida**: 25
- **Tempo de treinamento**: 15

Para definição dos parâmetros dos outros algoritmos de regressão utilizados para comparação na análise de resultados, os mesmos foram definidos de acordo com o padrão sugerido pelas bibliotecas de implementação.

Conforme abordado na Seção 2.2.4, foi utilizado a busca força bruta utilizando o algoritmo de Grid Search para otimização de hiperparâmetros dos detectores ativos de mudança de conceito. Considerando o tempo de execução do Grid Search, os parâmetros de cada detectores foram predefinidos como espaço de pesquisa para estimar o resultado de cada combinação. Foram considerados alguns valores, e, após a execução, o resultado apontou com base nos menores MSE e tempo de execução, a melhor combinação de valores para serem utilizados como hiperparâmetro para os detectores ativos de mudança de conceito (ADWIN, KSWIN e DDM):

- ***adwin_delta***: [1, 0.002, 0.003, 0.00001, 0.0001, 0.0000001] -> 0.0000001,
- ***kswin_alpha***: [1, 0.005, 0.003, 0.00001, 0.0001, 0.0000001] -> 0.0000001,
- ***kswin_window_size***: [100, 500] -> 100,
- ***kswin_stat_size***: [30, 100] -> 30,

- ***ddm_min_num_instances***: [30, 50] -> 30,
- ***ddm_warning_level***: [1, 2] -> 2,
- ***ddm_out_control_level***: [3, 5] -> 3

4.2 RESULTADOS

Esta seção apresenta os resultados de tempo de processamento e MSE alcançados pelos dois algoritmos adaptados propostos neste trabalho, em relação aos outros quatro outros algoritmos utilizados para comparação. Por fim, também são feitas análises sob a recuperação e comportamento dos modelos na presença das respectivas mudanças de conceito para cada base.

4.2.1 Análise em relação ao MSE

Apresenta-se na Tabela 3 o MSE médio das 5 execuções dos algoritmos para cada conjunto de dados e o desvio padrão entre as 5 execuções para os seis algoritmos no estudo sobre as 3 bases de dados ser explicitado. O melhor MSE é destacado e o ranking correspondente do algoritmo para esse conjunto de dados fica ao lado do MSE. O ranking do algoritmo é definido pelo teste de Friedman, ordenando o MSE de todos eles para cada conjunto de dados do menor para o maior, o que significa que a menor classificação é a melhor. Além disso, o ranking foi elaborado com base na média do ranking de todas as execuções realizadas, para cada um dos seis algoritmos considerados no estudo sobre as 3 bases de dados testadas.

Tabela 3 – MSE médio e ranking para os seis algoritmos considerados no estudo sobre as 3 bases de dados.

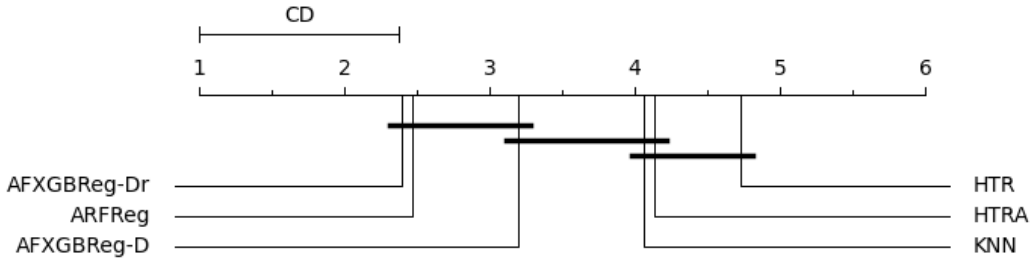
Dataset	AFXGBReg-Dr	AFXGBReg-D	HTR	HTRA	KNN	ARFReg
CDS_A	**3459.59** $^{(1)}$ $\pm 57,44$	3833.69 $^{(2)}$ $\pm 101,21$	9806.61 $^{(6)}$ $\pm 3,62$	7971.14 $^{(5)}$ $\pm 45,91$	7190.04 $^{(4)}$ $\pm 0,11$	5064.42 $^{(3)}$ $\pm 74,34$
HYP	0.14 $^{(5.2)}$ $\pm 0,008$	0.15 $^{(5.6)}$ $\pm 0,005$	0.109 $^{(2.2)}$ $\pm 0,009$	0.105 $^{(2.4)}$ $\pm 0,009$	0.13 $^{(4.2)}$ $\pm 0,002$	**0.10** $^{(1.4)}$ $\pm 0,002$
CDS_G	**4037.43** $^{(1)}$ $\pm 66,26$	4399.73 $^{(2)}$ $\pm 80,79$	11587.09 $^{(6)}$ $\pm 2,88$	9268.18 $^{(5)}$ $\pm 199,34$	7663.68 $^{(4)}$ $\pm 2,98$	5273.96 $^{(3)}$ $\pm 94,26$
Avg. MSE	2499.06	2744.52	7131.27	5746.48	4951.28	3446.16
Avg. rank	2.4	3.2	4.733	4.133	4.067	2.467

Fonte: O autor.

Percebe-se que o MSE médio de todos os algoritmos varia muito. Para confirmar a diferença de desempenho entre os algoritmos, aplicamos um teste *post-hoc* de todos os pares para comparações múltiplas (DEMŠAR, 2006). O teste *post-hoc* aplicado foi o teste Nemenyi usando o ranking médio dos algoritmos. Este teste determina uma Diferença Crítica (CD), o que significa que dois algoritmos cuja diferença de ranking seja maior que a diferença crítica são considerados significativamente diferentes. O Nemenyi resultou em um CD de 1,38 e as comparações de pares são mostradas na Figura 12.

É perceptível que AFXGBReg-Dr e ARFReg possuem desempenhos semelhantes, seguidos por AFXGBReg-D. Como a diferença do ranking médio entre esses três algoritmos é menor que o valor de CD, esse teste confirma que a diferença na capacidade de regressão não é

Figura 12 – Teste de Nemenyi aplicado aos rankings dos modelos.



Fonte: O autor.

significativa. Isso também significa que os algoritmos AFXGBReg-Dr, ARFReg e AFXGBReg-D são superiores aos demais em termos de minimização do MSE.

Para base de dados incremental (HYP), houve um comprometimento do MSE em relação aos algoritmos AFXGBReg, pela característica dos detectores presentes se tornarem sensíveis a qualquer mudança com uma força muito maior do que se deve agir, perdendo para estratégias mais neutras sem muitas reinicializações do modelo.

4.2.2 Análise em relação ao tempo de execução

Após a análise do MSE, foram comparados os tempos de execução dos algoritmos analisados, principalmente do AFXGBReg-Dr, AFXGBReg-D e ARFReg, que foram os que apresentaram melhor desempenho de MSE. A Tabela 4 apresenta o tempo médio de treinamento, o tempo médio de teste e o tempo total médio em minutos de todas as simulações em todos os conjuntos de dados.

Tabela 4 – Tempo médio em minutos das execuções para os algoritmos considerados no trabalho.

	Tempo (minutos)		
Model	Avg. training	Avg. testing	Avg. Total
AFXGBReg-Dr	$3.53 \pm 0,02$	$1.88 \pm 0,01$	$5.41 \pm 0,04$
AFXGBReg-D	$3.53 \pm 0,01$	$1.87 \pm 0,01$	$5.40 \pm 0,02$
HTR	$1.52 \pm 0,003$	$0.24 \pm 0,002$	$1.75 \pm 0,001$
HTRA	$5.89 \pm 0,05$	$0.23 \pm 0,003$	$6.12 \pm 0,07$
KNN	$0.22 \pm 0,003$	$3.79 \pm 0,01$	$4.01 \pm 0,006$
ARFReg	$179.30 \pm 0,75$	$3.27 \pm 0,02$	$182.57 \pm 0,76$

Fonte: O autor.

É evidente que ambos HTR e KNN possuem tempos totais menores. Isso é esperado devido à simplicidade dos algoritmos, uma vez que não foram desenvolvidos para fluxos de dados. Por outro lado, eles também tiveram MSE maior do que os outros algoritmos. Em relação ao grupo com MSE menor, os algoritmos propostos neste trabalho (AFXGBReg-Dr e AFXGBReg-D) apresentaram tempos semelhantes, como esperado, pois ambos são muito semelhantes. Entretanto, o ARFReg apresentou um tempo de treinamento excepcionalmente

maior quando comparado aos propostos no trabalho em tela. O tempo total médio de execução do algoritmo ARFReg foi 33 vezes maior do que os modelos AFXGBReg-Dr e AFXGBReg-D.

Como pode ser observado na Tabela 4, os dois algoritmos desenvolvidos (AFXGBReg-Dr e AFXGBReg-D) obtiveram um tempo de treinamento consideravelmente superior ao HTR e KNN. Essa questão pode ser explicada porque é na etapa de treinamento que os detectores de mudança de conceito são executados, como também acontece da mesma forma um determinado aumento de tempo de treinamento para os outros algoritmos que utilizam algum detector de mudança de conceito embutido. Com relação ao HTR e KNN, que não realizam essa detecção, seus tempos de treinamento são menores. Em relação ao tempo de predição, o AFXGBReg-Dr tem tempo semelhante ao AFXGBReg-D, justamente porque eles compartilham a mesma estratégia de atualização dos modelos, que é realizada por meio da alternância entre modelos.

Vale ressaltar que o tempo de execução do AFXGBReg pode ser reduzido ainda mais usando um tempo de vida menor, diminuindo assim a quantidade de modelos contidos dentro do XGBoost a partir da substituição mais frequente dos modelos. Porém, é necessário equilibrar a quantidade de modelos do XGBoost com a frequência de substituição dos modelos para não gerar um possível aumento do MSE.

4.2.3 Análise em relação a mudança de conceito

As próximas subseções apresentam os resultados da avaliação dos algoritmos em relação a recuperação dos modelos a partir da presença de mudança de conceito no fluxo de dados. São analisados três cenários, um com mudança abrupta (Seção 4.2.3.1), um com mudança gradual (Seção 4.2.3.2) e, por fim, um com mudança incremental (Seção 4.2.3.3).

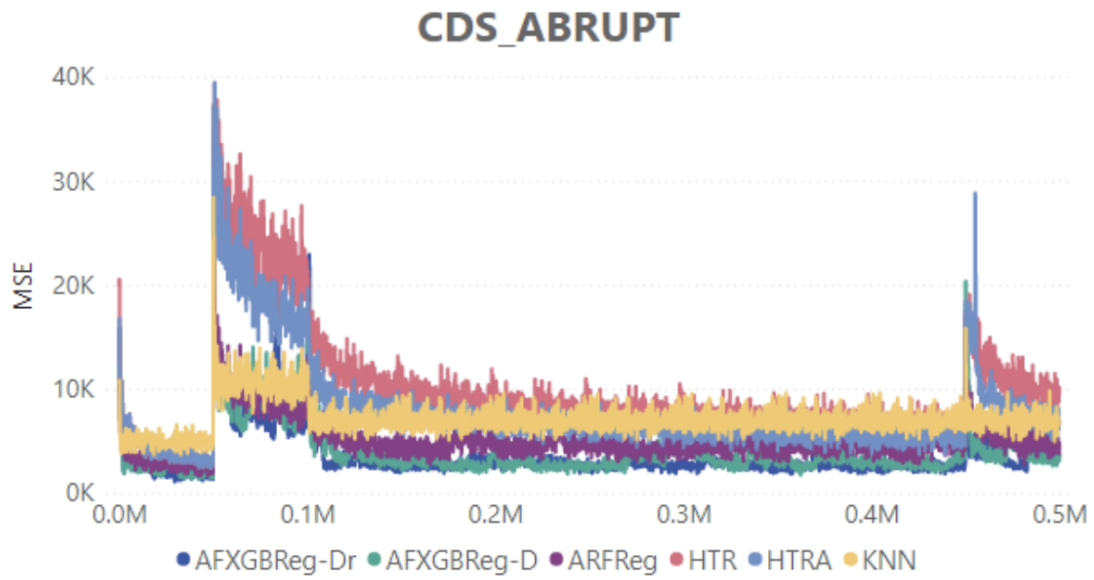
4.2.3.1 Mudança Abrupta - CDS_A

Para avaliar a capacidade dos modelos para se recuperarem de mudanças de conceito, foi analisado o erro de cada modelo ao longo do tempo após cada mudança. Foi selecionada uma execução de CDS_A para mostrar o comportamento do modelo para mudança de conceito abrupta.

Na Figura 13 é possível observar os modelos em relação ao MSE para a simulação do conjunto de dados CDS_A, onde ambos os modelos AFXGBReg possuem um pico de MSE menor que os demais. Isso pode ser percebido com mais detalhes nas Figuras 14 e 15, onde são destacados os desvios dos pontos 50k e 450k. Vê-se que o HTR e o HTRA têm picos de MSE mais altos, e também que o HTR e o KNN não se recuperam completamente da mudança de conceito. Isso acontece porque o HTR e o KNN não possuem detecção ativa de mudança de conceito, o que faz com que eles mantenham conceitos antigos.

Na mudança de conceito que ocorreu no instante 50k, representada na Figura 14, pode-se ver que a recuperação dos modelos AFXGBReg-Dr e AFXGBReg-D são mais rápidas do que outros modelos, mostrando a eficácia do conjunto de detectores ativos de mudança de conceito

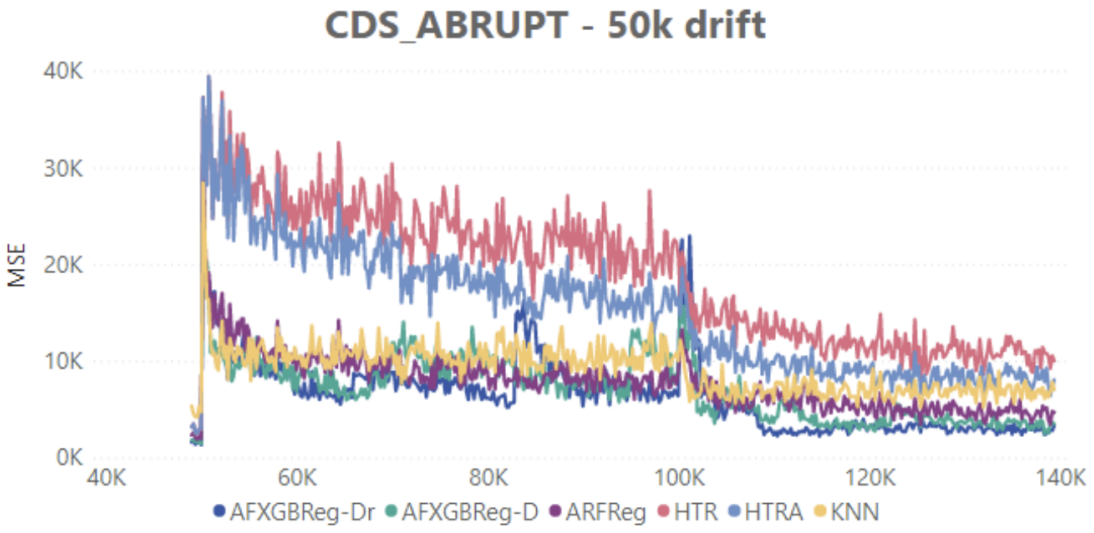
Figura 13 – Valores de MSE sob o tempo utilizando a base de dados CDS_A.



Fonte: O autor.

com a redefinição do modelo. Isso pode ser observado na Tabela 5, onde o MSE de todos os modelos é exibido para cinco pontos de dados relacionados a mudança de conceito no instante 50k com a base de dados CDS_A.

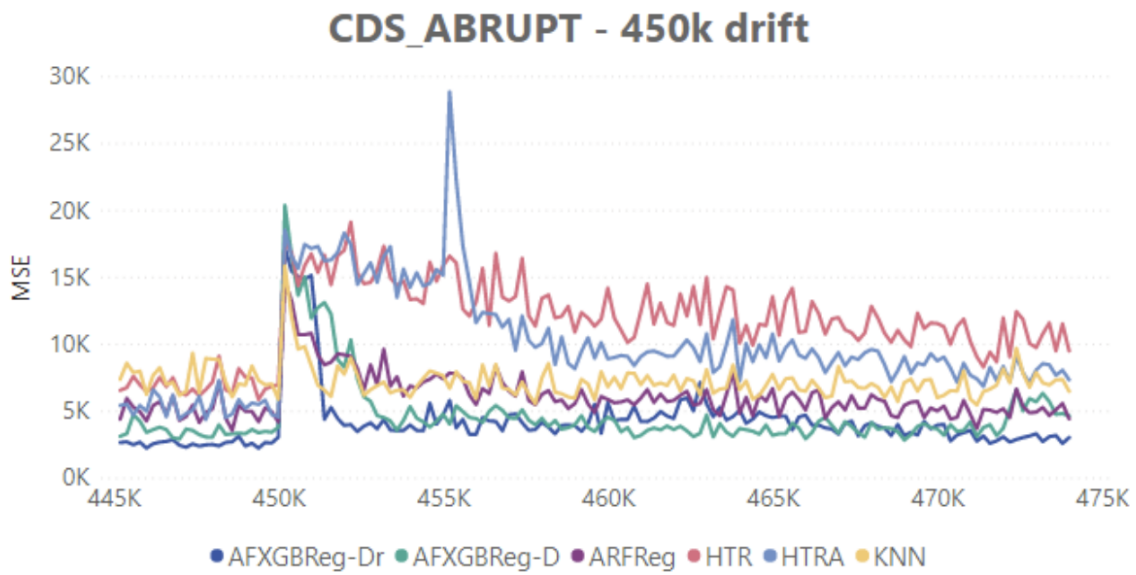
Figura 14 – Ampliação do gráfico de MSE no instante 50k, onde ocorreram mudanças de conceito na execução.



Fonte: O autor.

Na Tabela 5 é possível observar que no instante 50k a mudança de conceito ainda não aconteceu, então todos os modelos estão com seu menor MSE. Nos próximos instantes após

Figura 15 – Ampliação do gráfico de MSE no instante 450k, onde ocorreram mudanças de conceito na execução.



Fonte: O autor.

Tabela 5 – MSE dos modelos em instantes antes e após a mudança de conceito que ocorreu no instante 50k para a primeira execução da base de dados CDS_A.

# data	AFXGBReg-Dr	AFXGBReg-D	ARFReg	HTR	HTRA	KNN
50k	**1326.38**	1432.10	1963.47	2918.87	2991.71	4750.09
50.2k	**26358.49**	30496.94	35018.90	37292.58	37220.32	28408.24
50.4k	18703.27	**16580.87**	21904.89	32722.66	32722.39	22855.88
55.6k	**8520.81**	8726.02	11613.83	28335.11	24888.05	11196.76
82.4k	**5646.00**	9823.47	7950.37	22663.54	15872.01	11912.78

Fonte: O autor.

a mudança de conceito (50.2k) todos os modelos sofrem um grande aumento no MSE, com destaque nos modelos ARFReg, HTR e HTRA. Já no instante 50.4k os modelos começam a se ajustar ao novo conceito, diminuindo o MSE de forma a serem capazes de chegar a previsões estáveis para o novo conceito no ponto 55.6k. Nesse cenário, foram necessários cerca de 5.6k instâncias de dados para obter a recuperação completa do desvio de conceito. O último dado destacado na tabela é o menor MSE obtido neste conceito, alcançado pelo AFXGBReg-Dr no instante 82.4k. Pode-se ver que nenhum dos modelos conseguiu atingir o MSE tão baixo quanto anteriormente, mas em comparação com as variantes AFXGBReg-(DR e D) e ARFReg, os outros modelos tiveram um erro muito maior.

Como é possível observar na Tabela 5, a partir do instante 50k, os modelos AFXGBReg recuperam seu MSE devido ao mecanismo que diminui o tamanho da janela deslizante de treinamento dos dados de entrada. Essa recuperação rápida não acontece da mesma forma com os outros modelos, demonstrando a demora na adaptabilidade com a inserção de mudanças de conceito abruptas.

Ainda, vale destacar que aumentos acentuados que acontecem fora de instantes com mudança de conceito abrupta, mesmo após adaptações, é uma característica que está relacionada a memória do regressores utilizados. Como eles demoram mais tempo para se atualizar, ainda existem partes do modelo que não foram adequadamente atualizadas e quando são acessadas causam erro na predição.

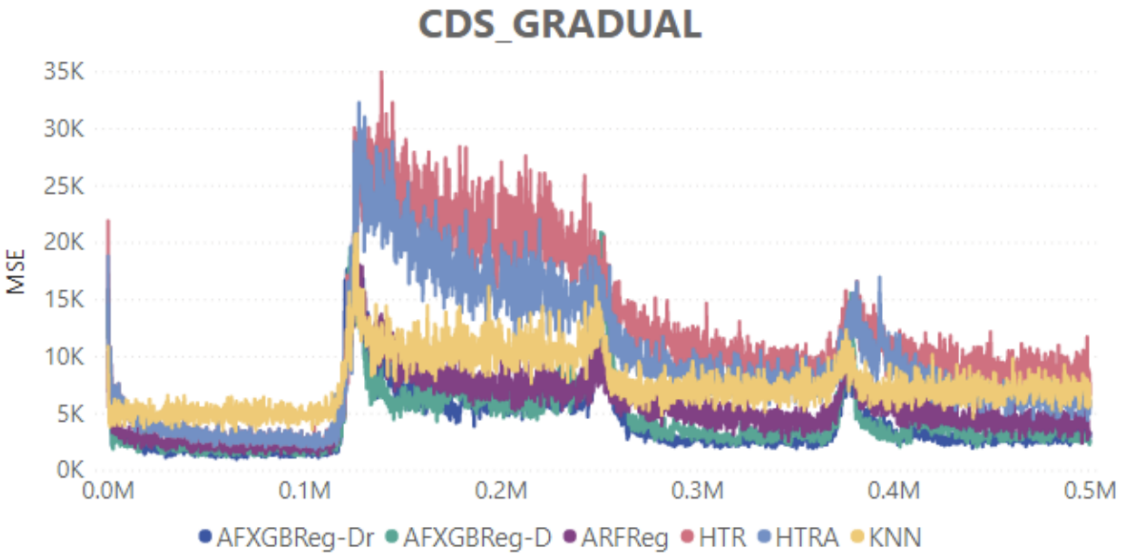
4.2.3.2 Mudança Gradual - CDS_G

Visando analisar a execução dos modelos para recuperação a diferentes tipos de mudança de conceito, após o estudo sobre a base de dados abrupta, foi realizada a análise da recuperação de mudança de conceito na base de dados gradual CDS_G.

Comparando a Figura 16 com a execução utilizando mudança de conceita abrupta, nota-se a mesma característica em relação aos picos de MSE. Ou seja, os modelos AFXGBReg-(Dr e D) apresentam picos menores que os demais. Essa mesma característica pode ser percebida com mais detalhes nas Figuras 17 e 18, onde são destacadas as mudanças nos instantes 120k e 370k.

Na Tabela 6 verifica-se o MSE dos modelos antes e após a mudança de conceito gradual inserida no instante 125k. Percebe-se a mesma característica presente para a base de dados abrupta, onde os modelos AFXBReg(Dr e D) possuem adaptação mais rápida a mudanças de conceito e conseguem se recuperar diminuindo o tamanho da janela deslizante dos dados de entrada.

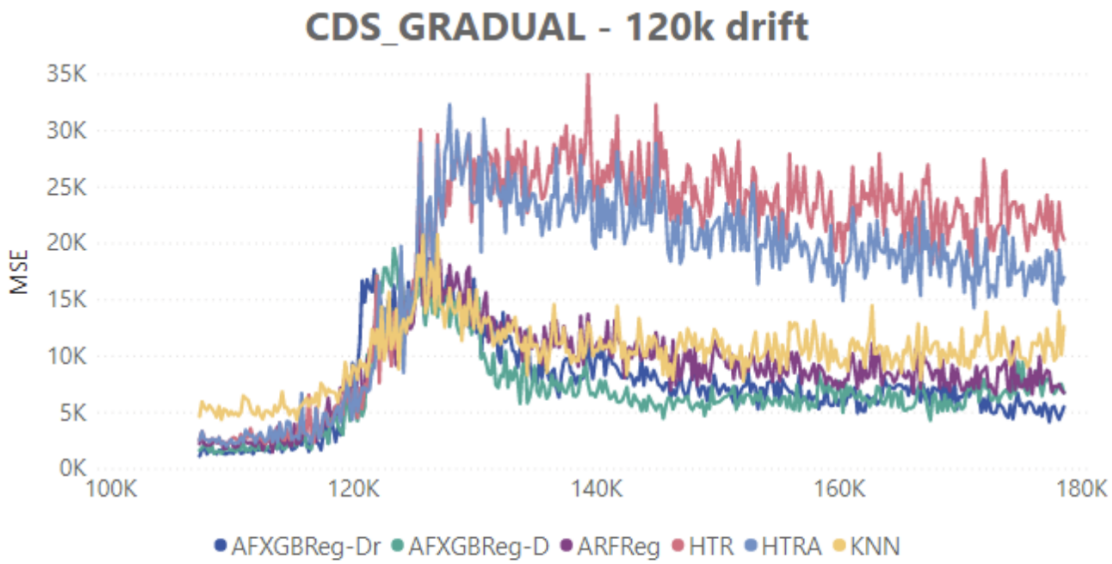
Figura 16 – MSE sob o tempo para os modelos utilizando a base de dados CDS_G.



Fonte: O autor.

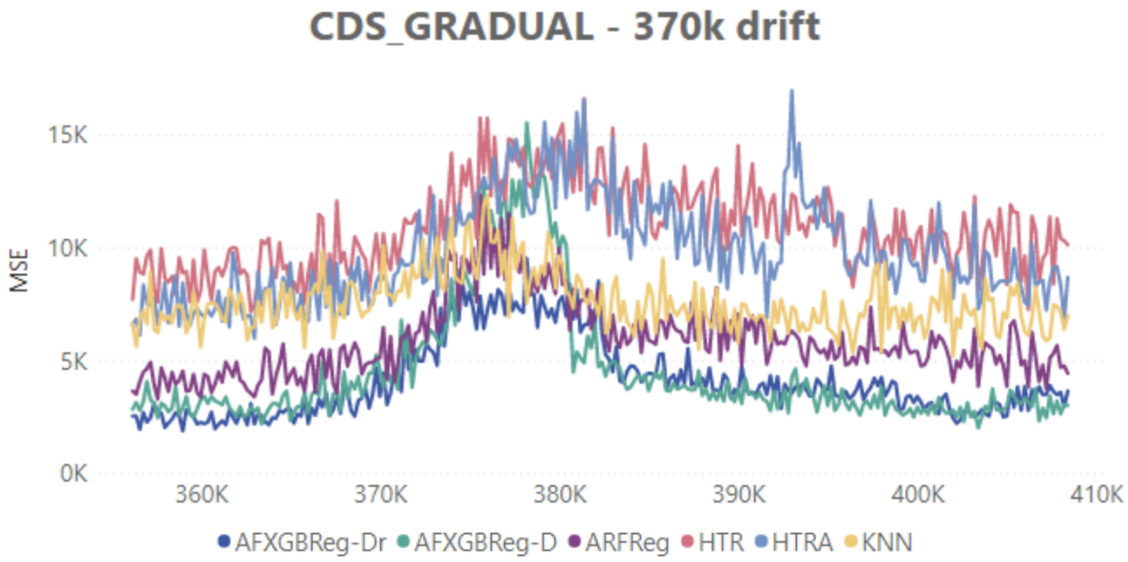
Um ponto interessante a ser observado se deve a característica dos regressores AFXBReg possuírem condições sensíveis para detecção de qualquer tipo de mudança de conceito. Conforme observado anteriormente, para uma mudança abrupta, a utilização do conjunto de detectores

Figura 17 – Ampliação do gráfico de MSE no instante 120k, onde ocorreram mudanças de conceito na execução.



Fonte: O autor.

Figura 18 – Ampliação do gráfico de MSE no instante 370k, onde ocorreram mudanças de conceito na execução.



Fonte: O autor.

é extremamente favorável para identificação em menor tempo e recuperação da mudança de conceito. Porém, para mudanças de conceito que acontecem em ocorrências consideráveis de forma gradual, o mecanismo de reinicializar o tamanho da janela com mudanças de conceito detectadas pode comprometer o MSE do modelo, pois desse modo, se perde o que se sabia antes devido a reinicialização do tamanho da janela para zero.

Tabela 6 – MSE dos modelos em instantes antes e após a mudança de conceito que ocorreu no instante 125k para a primeira execução da base de dados CDS_G.

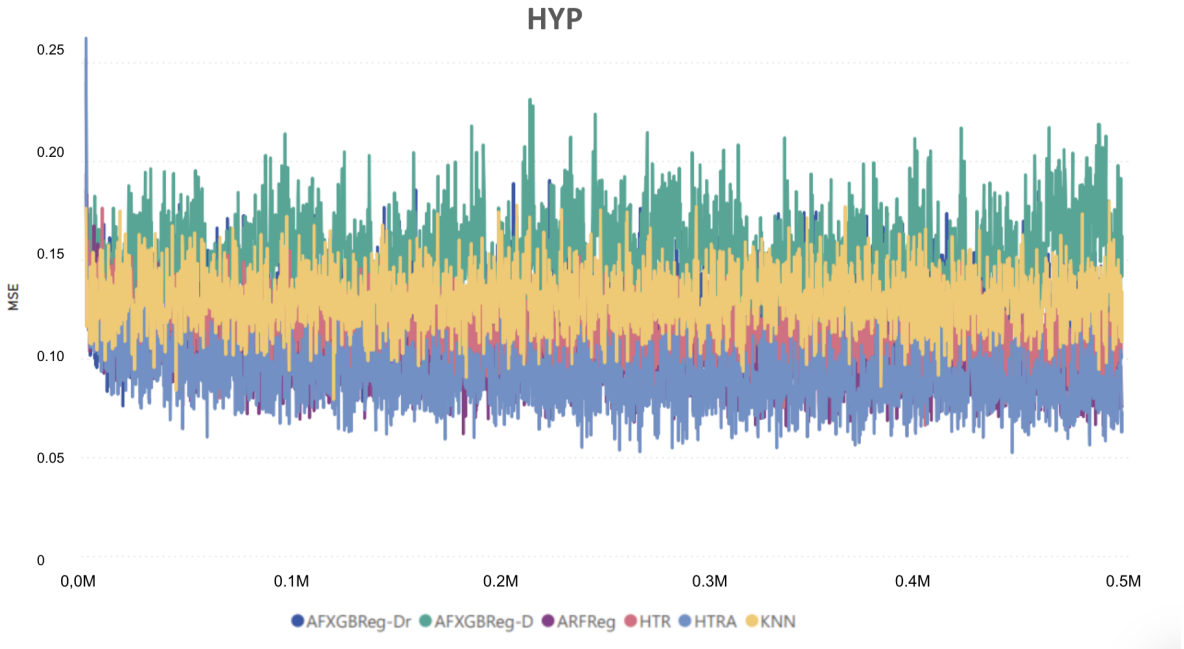
# data	AFXGBReg-Dr	AFXGBReg-D	ARFReg	HTR	HTRA	KNN
123k	2217.12	**2004.65**	2755.54	3653.11	3640.09	5198.65
125k	2420.51	**2141.22**	2915.45	3819.55	3803.91	5319.89
136k	3272.02	**2818.43**	3858.67	5627.93	5537.77	5975.94
268k	4902.39	**4304.12**	5834.54	13060.14	11002.90	8115.02
400k	4357.34	**3971.20**	5648.85	12214.63	10281.53	7824.19

Fonte: O autor.

4.2.3.3 Mudança Incremental - HYP

A última base de dados analisada nos experimentos foi gerada pelo gerador *Hyperplane-fast* (HYP), no qual o Hiperplano com D dimensões muda de posição e orientação. Outra característica dessa base de dados é que ela possui uma mudança de conceito incremental.

Figura 19 – MSE sob o tempo para os modelos utilizando a base de dados HYP.



Fonte: O autor.

Na Figura 19 tem-se o gráfico do MSE sob o tempo para os modelos utilizando a base de dados HYP com mudança de conceito incremental. Neste fluxo é possível notar que os modelos possuíram MSE semelhante. Com a própria base de dados testada, durante a execução não esperava-se nenhum grande pico devido a própria característica do tipo de mudança de conceito da base, a incremental. Pelas mudanças acontecerem de forma lenta, com o conceito sendo inserido incrementalmente, observou-se que todos os modelos conseguiram se adaptar eficientemente a essa mudança.

Alguns pontos para se destacar no gráfico de MSE são as etapas de deterioração dos

regressores AFXGBReg. O algoritmo apresentou vários pontos consideráveis de alta, enquanto os outros modelos como HTR, HTRA, ARFReg e KNN mantiveram a média de MSE. Como conclusão, destaca-se que para esse tipo de mudança de conceito, a detecção ativa de mudança de conceito e a aceleração da atualização do modelo não se demonstrou uma boa estratégia.

Na Tabela 7 mostra o valor de MSE dos modelos em alguns instantes selecionados. Verifica-se que para os instantes 123k, 125k e 400k o melhor MSE era do modelo ARFReg, seguido pelo algoritmo HTRA nos instantes 136k e 268k. Ainda, é possível perceber que tanto para os instantes detalhados quanto para média geral de MSE os modelos AFXGBReg-(Dr e D) apresentaram os dois piores resultados.

Com esse resultado é possível inferir que o algoritmo proposto (AFXGBReg-Dr e AFXGBReg-D) ainda não apresenta uma estratégia eficaz para detectar mudanças de conceito incrementais, pois o detector se torna sensível a qualquer mudança com uma força muito maior do que se deveria agir, perdendo assim para estratégias tradicionais.

Tabela 7 – MSE dos modelos em alguns instantes para a primeira execução da base de dados HYP.

# data	**AFXGBReg-Dr**	**AFXGBReg-D**	**ARFReg**	**HTR**	**HTRA**	**KNN**
123k	0.122147	0.146556	**0.093453**	0.122485	0.104741	0.122600
125k	0.121041	0.141404	**0.076350**	0.109174	0.101041	0.140000
136k	0.111426	0.175730	0.097279	0.142007	**0.090835**	0.098800
268k	0.131601	0.173749	0.074807	0.120047	**0.065680**	0.121600
400k	0.131628	0.160864	**0.063120**	0.105674	0.083095	0.118600

Fonte: O autor.

4.3 CONSIDERAÇÕES SOBRE OS RESULTADOS

Com base nos resultados obtidos, foi possível observar que ambos modelos AFXGBReg-(Dr e D) e ARFReg apresentaram comportamentos bastante parecidos referentes ao MSE, comparado aos outros algoritmos simulados (HTR, HTRA e KNN). Apesar de não apresentarem significância entre si (AFXGBReg e ARFReg) verificando o MSE, em relação aos testes feitos para comparar o tempo de execução dos algoritmos, foi possível notar uma grande diferença entre a versão AFXGBReg-(Dr e D) e ARFReg, sendo a primeira 33x mais rápida do que a segunda. Tal diferença se deve a característica do AFXGBReg-(Dr e D) utilizar apenas um modelo XGBoost, que é treinado e atualizado de forma incremental. Após o regressor ultrapassar um limite de dados treinados, outro regressor é treinado visando a substituição. Nesta etapa de testes os algoritmos foram executados individualmente, eliminando a possibilidade de um algoritmo influenciar na execução do outro.

Referente a análise sobre as mudanças de conceito, neste trabalho foram estudados os tipos de mudança Abrupta, Gradual e Incremental. Os resultados obtidos nos testes demonstraram que a recuperação dos modelos AFXGBReg-(Dr e D) a mudanças de conceito abruptas e graduais foram consideravelmente mais rápidas que o restante dos algoritmos. Essa recuperação

ágil se deve ao mecanismo que diminui o tamanho da janela deslizante de treinamento dos dados de entrada. Consequentemente, por acumularem com o passar do tempo conceitos já considerados inválidos, os outros modelos também atingiram picos de MSE maiores que os modelos AFXGBReg-(Dr e D).

Com relação a mudança de conceito incremental, os testes feitos resultaram na conclusão de que para mudanças de conceito que acontecem incrementalmente o mecanismo de reinicialização do tamanho da janela com mudanças de conceito pode comprometer o MSE, pois se perde o que sabia antes devido a reinicialização do tamanho da janela. Essa limitação é mais visível verificando a experimentação para base de dados incremental. Em determinados instantes o detector se torna sensível a qualquer mudança com uma força muito maior do que se deve agir, perdendo para estratégias mais neutras sem muitas reinicializações do modelo.

Analisando especificamente o comportamento das variantes AFXGBReg-Dr e AFXGBReg-D, ambos são algoritmos similares porém com o primeiro utilizando um mecanismo de reinicialização de janela deslizante no momento da troca do regressor corrente pelo novo regressor que estava em treinamento. Analisando os resultados de ambos, os números de MSE do primeiro foram levemente menores do que do segundo, inferindo que o primeiro seja melhor que o segundo, mas para se chegar a essa conclusão são necessárias mais análises.

Portanto, os modelos AFXGBReg-Dr e AFXGBReg-D apresentaram resultados satisfatórios, pois apesar de apresentarem MSE similar estatisticamente ao ARFReg, têm tempo de execução consideravelmente menores. Observou-se melhores resultados para bases com mudança de conceito abrupta, porém ainda sim mantendo níveis similares de tempo de execução, MSE e adaptabilidade a mudanças de conceito em comparação aos outros modelos para as outras duas bases testadas: gradual e incremental.

5 CONCLUSÃO

Com o grande crescimento da geração de fluxo de dados em tempo real, extrair informações e predições úteis tornou-se uma parte importante em diversas áreas. Porém, por se tratarem de dados não estacionários, uma série de dificuldades surgem durante o processo de classificação ou regressão em fluxo de dados, principalmente com relação a rapidez do algoritmo, sua eficiência no uso de memória e adaptação a diferentes tipos de mudança de conceito.

Logo, esse trabalho tem como objetivo propor uma adaptação ao algoritmo de classificação binária AFXGB proposto por Bonassa (2021) para torná-lo hapto a realizar a regressão em fluxos de dados, com suporte para se adaptar rapidamente a mudança de conceito por meio da união de detectores ativos.

O AFXGB utiliza uma estratégia de substituição do modelos XGBoost que o torna uma ferramenta robusta na análise de fluxos de dados, entretanto, até o momento seu potencial não tinha sido explicado na regressão de fluxos de dados. Portanto, este trabalho explorou o potencial do AFXGB no contexto da regressão de fluxo com presença de diferentes tipos de mudança de conceito. A mudança de conceito foi tratada por meio da aplicação de detectores ativos no intuito de acelerar a detecção e assim atualizar o modelo de forma mais rápida.

Por ser um algoritmo altamente dependente dos parâmetros utilizados, diferentes comportamentos foram observados, porém, com base em trabalhos recentes foram escolhidos valores que proporcionaram o seu melhor desempenho. Para definir os hiperparâmetros dos detectores ativos de mudança de conceito, foi utilizado o algoritmo de busca por força bruta *Grid Search*. Durante os testes, foi possível detectar a eficiência dos algoritmo proposto para diferentes tipos de mudança de conceito: abrupto, gradual e incremental. A união de detectores apresentou menor MSE para detecção de mudanças de conceito abruptas devido à sua característica de detecção rápida e consequente adaptação com os três mecanismos utilizados: ADWIN, DDM e KSWIN. Consequentemente, por se mostrar sensível a qualquer mudança de conceito e reagir com a reinicialização do tamanho da janela deslizante, conceitos considerados antigos não foram considerados nessa nova janela, permitindo a melhor readaptação do modelo para novos conceitos.

Além disso, os detectores ativos de mudança de conceito foram escolhidos com base na utilização complementar de cada um, de forma a tornar cada mecanismo mais eficiente em determinado tipo de mudança de conceito, mas com seus hiperparâmetros configurados de maneira a não aumentar demasiadamente o tempo de execução e utilização de memória do modelo. Portanto, os pontos-chave para escolha dos detectores ativos foram a melhor adaptabilidade a diferentes tipos de mudança de conceito e a utilização de hiperparâmetros configuráveis.

Além da escolha dos detectores, também foi necessário análises considerando sua ordem de escalonamento e execução. Em testes realizados com todos os detectores avaliados, foi possível observar que o KSWIN apresentou uma sensibilidade para detecção de mudanças de conceito elevada, ocasionando muitas ocorrências de detecções falso-positivas. Portanto, por esse

motivo, os detectores foram escalonados de forma que o ADWIN fosse escolhido em primeiro lugar, por sua assertividade para mudanças de conceito recorrentes e graduais, o DDM como segunda opção, por seu desempenho e eficácia para mudanças de conceito abruptas e, por fim, o KSWIN como última opção caso uma mudança de conceito não seja detectada por nenhum outro detector acima.

Pela análise dos resultados dos experimentos realizados foi possível observar que o algoritmo AFXGBReg proposto (em ambas as formas AFXGBReg-Dr e AFXGBReg-D) foi eficaz na detecção e recuperação rápida de mudanças de conceito abruptas e graduais em comparação aos outros algoritmos analisados. Ainda, ele apresentou resultados competitivos sobre as métricas avaliadas, tendo valores de MSE estaticamente similares aos do ARFReg, mas obtendo média do tempo de execução 33 vezes mais rápida que a média do algoritmo ARFReg.

Dado que os resultados do trabalho foram considerados promissores, sua proposta foi submetida com artigo completo para o 11th Brazilian Conference on Intelligent Systems 2022 (BRACIS). Até o momento de escrita desse trabalho a submissão está em processo de revisão pelo evento.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros, oportunidades para estudo se abrem sobre as limitações envolvendo a regressão/classificação e possíveis atrasos do classificador se atualizar, ou seja, o intervalo até a sua próxima substituição.

Ainda, pretende-se explorar modelos AFXGBReg com aprendizagem semi-supervisionada. Com relação a mudança de conceito, pretende-se estudar mais sobre outras estratégias de detecção além da ativa, bem como explorar o algoritmo AFXGBReg para torná-lo cada vez mais eficiente para diferentes tipos de mudança de conceito.

REFERÊNCIAS

ABBASZADEH, Omid; AMIRI, Ali; KHANTEYMOORI, Ali Reza. An ensemble method for data stream classification in the presence of concept drift. **Frontiers of Information Technology & Electronic Engineering**, Springer, v. 16, n. 12, p. 1059–1068, 2015. Citado na página 25.

ALTMAN, Naomi S. An introduction to kernel and nearest-neighbor nonparametric regression. **The American Statistician**, Taylor & Francis, v. 46, n. 3, p. 175–185, 1992. Citado 3 vezes nas páginas 20, 32 e 34.

BARDDAL, Jean Paul. Vertical and horizontal partitioning in data stream regression ensembles. In: IEEE. **2019 International Joint Conference on Neural Networks (IJCNN)**. Curitiba, BR, 2019. p. 1–8. Citado 2 vezes nas páginas 22 e 25.

BASSEVILLE, Michele; NIKIFOROV, Igor V et al. **Detection of abrupt changes: theory and application**. France: prentice Hall Englewood Cliffs, 1993. v. 104. Citado na página 22.

BERGSTRA, James; BENGIO, Yoshua. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. 2, 2012. Citado na página 29.

BIFET, Albert. Adaptive learning and mining for data streams and frequent patterns. **ACM SIGKDD Explorations Newsletter**, ACM New York, NY, USA, v. 11, n. 1, p. 55–56, 2009. Citado na página 20.

BIFET, Albert; GAVALDA, Ricard. Learning from time-changing data with adaptive windowing. In: SIAM. **Proceedings of the 2007 SIAM international conference on data mining**. Catalunha, Espanha, 2007. p. 443–448. Citado na página 24.

BONASSA, Gustavo. **Adaptação de classificador utilizando a biblioteca XGBoost para classificação rápida de fluxos de dados parcialmente classificados com mudança de conceito**. 2021. Citado 12 vezes nas páginas 13, 14, 30, 34, 35, 37, 38, 41, 42, 44, 45 e 56.

BOSE, RP Jagadeesh Chandra et al. Dealing with concept drifts in process mining. **IEEE transactions on neural networks and learning systems**, IEEE, v. 25, n. 1, p. 154–171, 2013. Citado na página 22.

BOULEGANE, Dihia et al. Streaming time series forecasting using multi-target regression with dynamic ensemble selection. In: IEEE. **2020 IEEE International Conference on Big Data (Big Data)**. Atlanta, GA, USA, 2020. p. 2170–2179. Citado 3 vezes nas páginas 31, 34 e 35.

BREIMAN, Leo. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 17.

CHEN, Tianqi; GUESTRIN, Carlos. Xgboost: A scalable tree boosting system. In: ACM. **Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining**. Washington, USA, 2016. p. 785–794. Citado 3 vezes nas páginas 18, 30 e 34.

CHEN, Tianqi et al. Xgboost: extreme gradient boosting. **R package version 0.4-2**, v. 1, n. 4, p. 1–4, 2015. Citado na página 13.

CHEN, Wenbin et al. Radar emitter classification for large data set based on weighted-xgboost. **IET**, 2016. Citado na página 35.

CHEN, Yixin et al. Multi-dimensional regression analysis of time-series data streams. In: ELSEVIER. **VLDB'02: Proceedings of the 28th International Conference on Very Large Databases**. Wright State University, USA, 2002. p. 323–334. Citado na página 20.

CHICCO, Davide; JURMAN, Giuseppe. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. 2020. Citado na página 27.

DARGAN, Shaveta et al. A survey of deep learning and its applications: a new paradigm to machine learning. **Archives of Computational Methods in Engineering**, Springer, p. 1–22, 2019. Citado na página 15.

DEMŠAR, Janez. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, JMLR. org, v. 7, p. 1–30, 2006. Citado na página 46.

DITZLER, Gregory et al. Learning in nonstationary environments: A survey. **Comp. Intell. Mag.**, IEEE Press, v. 10, n. 4, p. 12–25, nov. 2015. ISSN 1556-603X. Disponível em: <https://doi.org/10.1109/MCI.2015.2471196>. Citado na página 12.

DOMINGOS, Pedro; HULTEN, Geoff. Mining high-speed data streams. In: **Kdd**. [S.l.: s.n.], 2000. v. 2, p. 4. Citado 2 vezes nas páginas 32 e 34.

DONG, Xibin et al. A survey on ensemble learning. **Frontiers of Computer Science**, Springer, v. 14, n. 2, p. 241–258, 2020. Citado 2 vezes nas páginas 17 e 19.

DUARTE, João; GAMA, João; BIFET, Albert. Adaptive model rules from high-speed data streams. **ACM Trans. Knowl. Discov. Data**, Association for Computing Machinery, New York, NY, USA, v. 10, n. 3, jan 2016. ISSN 1556-4681. Disponível em: <https://doi.org/10.1145/2829955>. Citado na página 20.

ELWELL, Ryan; POLIKAR, Robi. Incremental learning of concept drift in nonstationary environments. **IEEE Transactions on Neural Networks**, IEEE, v. 22, n. 10, p. 1517–1531, 2011. Citado na página 12.

FAFALIOS, Stefanos; CHARONYKTAKIS, Pavlos; TSAMARDINOS, Ioannis. Gradient boosting trees. 2020. Citado na página 19.

GALAR, Mikel et al. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, IEEE, v. 42, n. 4, p. 463–484, 2011. Citado na página 17.

GAMA, João et al. A survey on concept drift adaptation. **ACM computing surveys (CSUR)**, ACM, v. 46, n. 4, p. 44, 2014. Citado na página 25.

GAMA, João et al. A survey on concept drift adaptation. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 46, n. 4, mar. 2014. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2523813>. Citado na página 12.

GAMAGE, Sunanda; PREMARATNE, Upeka. Detecting and adapting to concept drift in continually evolving stochastic processes. In: **Proceedings of the International Conference on Big Data and Internet of Thing**. New York, NY, USA: Association for Computing Machinery, 2017. (BDIOT2017), p. 109–114. ISBN 9781450354301. Disponível em: <https://doi.org/10.1145/3175684.3175723>. Citado na página 11.

GOLLAPUDI, Sunila. **Practical machine learning**. EUA: Packt Publishing Ltd, 2016. Citado na página 17.

GOMES, Heitor Murilo et al. Adaptive random forests for data stream regression. In: IEEE. **ESANN**. Curitiba, Paraná, Brasil, 2018. Citado 4 vezes nas páginas 20, 31, 34 e 35.

GONÇALVES-JR, Paulo M et al. A comparative study on concept drift detectors. **Expert Systems with Applications**, Elsevier, v. 41, n. 18, p. 8144–8156, 2014. Citado na página 24.

GRULICH, Philipp M et al. Scalable detection of concept drifts on data streams with parallel adaptive windowing. In: EDBT. **EDBT**. Berlim, Alemanha, 2018. p. 477–480. Citado na página 23.

HAN, Jiawei; PEI, Jian; KAMBER, Micheline. **Data mining: concepts and techniques**. Illinois, EUA: Elsevier, 2011. 460-500 p. Citado 2 vezes nas páginas 16 e 20.

HO, Tin Kam. Complexity of classification problems and comparative advantages of combined classifiers. In: SPRINGER. **International workshop on multiple classifier systems**. Murray Hill, USA, 2000. p. 97–106. Citado na página 17.

HOENS, T Ryan; POLIKAR, Robi; CHAWLA, Nitesh V. Learning from streaming data with concept drift and imbalance: an overview. **Progress in Artificial Intelligence**, Springer, v. 1, n. 1, p. 89–101, 2012. Citado na página 20.

IKONOMOVSKA, Elena; GAMA, João; DŽEROSKI, Sašo. Online tree-based ensembles and option trees for regression on evolving data streams. **Neurocomputing**, Elsevier, v. 150, p. 458–470, 2015. Citado na página 20.

JUNIOR, Paulo Gonçalves et al. A comparative study on concept drift detectors. **Expert Systems with Applications**, v. 41, p. 8144–8156, 12 2014. Citado 2 vezes nas páginas 32 e 35.

KADIYALA, Akhil; KUMAR, Ashok. Applications of python to evaluate the performance of decision tree-based boosting algorithms. **Environmental Progress & Sustainable Energy**, Wiley Online Library, v. 37, n. 2, p. 618–623, 2018. Citado na página 42.

KOEHRSEN, Will. Overfitting vs. underfitting: A complete example. **Towards Data Science**, 2018. Citado na página 15.

KRAWCZYK, Bartosz et al. Ensemble learning for data stream analysis: A survey. **Information Fusion**, v. 37, p. 132 – 156, 2017. ISSN 1566-2535. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253516302329>. Citado 4 vezes nas páginas 11, 17, 22 e 27.

KRAWCZYK, Bartosz et al. Ensemble learning for data stream analysis: A survey. **Information Fusion**, v. 37, p. 132–156, 2017. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253516302329>. Citado na página 25.

KRAWCZYK, Bartosz; PFAHRINGER, Bernhard; WOŹNIAK, Michał. Combining active learning with concept drift detection for data stream mining. In: IEEE. **2018 IEEE International Conference on Big Data (Big Data)**. Hamilton, New Zealand, 2018. p. 2239–2244. Citado na página 17.

KROGH, Anders; VEDELSBY, Jesper et al. Neural network ensembles, cross validation, and active learning. **Advances in neural information processing systems**, Morgan Kaufmann Publishers, v. 7, p. 231–238, 1995. Citado na página 17.

KWON, Donghwoon et al. A survey of deep learning-based network anomaly detection. **Cluster Computing**, Springer, v. 22, n. 1, p. 949–961, 2019. Citado na página 16.

LANEY, Douglas. **3D Data Management: Controlling Data Volume, Velocity, and Variety**. EUA, 2001. Disponível em: <http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>. Citado na página 11.

LARSON, Deanne; CHANG, Victor. A review and future direction of agile, business intelligence, analytics and data science. **International Journal of Information Management**, Elsevier, v. 36, n. 5, p. 700–710, 2016. Citado na página 11.

LI, Changgeng; QIU, Zhengyang; LIU, Changtong. An improved weighted k-nearest neighbor algorithm for indoor positioning. **Wireless Personal Communications**, Springer, v. 96, n. 2, p. 2239–2251, 2017. Citado na página 32.

LIAO, Zhenwei; WANG, Yongheng. Rival learner algorithm with drift adaptation for online data stream regression. In: **Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence**. New York, NY, USA: Association for Computing Machinery, 2018. (ACAI 2018). ISBN 9781450366250. Disponível em: <https://doi.org/10.1145/3302425.3302475>. Citado 5 vezes nas páginas 12, 20, 31, 34 e 35.

LIASHCHYNSKYI, Petro; LIASHCHYNSKYI, Pavlo. Grid search, random search, genetic algorithm: A big comparison for nas. **arXiv preprint arXiv:1912.06059**, 2019. Citado 2 vezes nas páginas 28 e 29.

LOPES, Raul HC; REID, ID; HOBSON, Peter R. The two-dimensional kolmogorov-smirnov test. Proceedings of Science, 2007. Citado na página 24.

LU, Jie et al. Learning under concept drift: A review. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 31, n. 12, p. 2346–2363, 2018. Citado 4 vezes nas páginas 11, 22, 23 e 25.

Lu, J. et al. Learning under concept drift: A review. **IEEE Transactions on Knowledge and Data Engineering**, v. 31, n. 12, p. 2346–2363, 2018. Citado na página 21.

MAHDI, Osama A et al. Fast reaction to sudden concept drift in the absence of class labels. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 10, n. 2, p. 606, 2020. Citado na página 22.

MANICKASWAMY, Thangam; BHUVANESWARI, A. Concept drift in data stream classification using ensemble methods: Types, methods and challenges. **INFOCOMP Journal of Computer Science**, v. 19, n. 2, p. 163–174, 2020. Citado na página 23.

MARSLAND, Stephen. **Machine learning: an algorithmic perspective**. Berlin, Germany: Chapman and Hall/CRC, 2011. Citado na página 12.

MAULUD, Dastan; ABDULAZEEZ, Adnan M. A review on linear regression comprehensive in machine learning. **Journal of Applied Science and Technology Trends**, v. 1, n. 4, p. 140–147, 2020. Citado na página 17.

MAYR, Andreas et al. The evolution of boosting algorithms. **Methods of information in medicine**, Schattauer GmbH, v. 53, n. 06, p. 419–427, 2014. Citado na página 13.

MEHMOOD, Hassan et al. Concept drift adaptation techniques in distributed environment for real-world data streams. **Smart Cities**, Multidisciplinary Digital Publishing Institute, v. 4, n. 1, p. 349–371, 2021. Citado na página 12.

MONTIEL, Jacob et al. Adaptive xgboost for evolving data streams. In: IEEE. **2020 International Joint Conference on Neural Networks (IJCNN)**. Hamilton, New Zealand, 2020. p. 1–8. Citado 6 vezes nas páginas 13, 30, 34, 35, 37 e 44.

MONTIEL, Jacob et al. Scikit-multiflow: A multi-output streaming framework. **Journal of Machine Learning Research**, v. 19, n. 72, p. 1–5, 2018. Disponível em: <http://jmlr.org/papers/v19/18-251.html>. Citado 2 vezes nas páginas 32 e 44.

NIKAM, Sagar S. A comparative study of classification techniques in data mining algorithms. **Oriental journal of computer science & technology**, v. 8, n. 1, p. 13–19, 2015. Citado na página 16.

NUNES, André Luís. Um estudo investigativo de algoritmos de regressão para data streams. Universidade do Vale do Rio dos Sinos, 2017. Citado na página 21.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citado na página 32.

PINAGÉ, Felipe Azevedo et al. Handling concept drift based on data similarity and dynamic classifier selection. Universidade Federal do Amazonas, 2017. Citado na página 20.

PROVOST, Foster; KOHAVI, R. Glossary of terms. **Journal of Machine Learning**, v. 30, n. 2-3, p. 271–274, 1998. Citado na página 15.

RAMRAJ, S et al. Experimenting xgboost algorithm for prediction and classification of different datasets. **International Journal of Control Theory and Applications**, v. 9, p. 651–662, 2016. Citado na página 13.

REIS, Denis Moreira dos. **Classificação de fluxos de dados com mudança de conceito e latência de verificação**. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado na página 19.

SANTOS, Silas Garrido Teixeira de Carvalho. **Avaliação criteriosa dos algoritmos de detecção de concept drifts**. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2015. Citado 3 vezes nas páginas 32, 34 e 35.

SCHAPIRE, Robert E. The strength of weak learnability. **Machine learning**, Springer, v. 5, n. 2, p. 197–227, 1990. Citado na página 17.

SCHAPIRE, Robert E. The boosting approach to machine learning: An overview. **Nonlinear estimation and classification**, Springer, p. 149–171, 2003. Citado na página 12.

SOUZA, F. M. de. **ADAPTAÇÃO DA BIBLIOTECA AXGBOOST PARA REGRESSÃO EM FLUXOS DE DADOS COM DETECÇÃO E RESPOSTA À MUDANÇA DE CONCEITOS REALIZADA POR MÚLTIPLOS DETECTORES JOINVILLE 2022**. 63 p. — Universidade do Estado de Santa Catarina, Joinville, 2022. Citado 2 vezes nas páginas 33 e 34.

TOGBE, Maurras Ulbricht et al. Anomalies detection using isolation in concept-drifting data streams. **Computers**, Multidisciplinary Digital Publishing Institute, v. 10, n. 1, p. 13, 2021. Citado na página 24.

VERBRAEKEN, Joost et al. A survey on distributed machine learning. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 53, n. 2, p. 1–33, 2020. Citado na página 15.

WIDMER, Gerhard. Combining robustness and flexibility in learning drifting concepts. In: CITESEER. **ECAI**. Vienna, Austria, 1994. p. 468–472. Citado na página 20.

YAN, Myuu Myuu Wai. Accurate detecting concept drift in evolving data streams. **ICT Express**, Elsevier, v. 6, n. 4, p. 332–338, 2020. Citado na página 25.

YANG, Li; MANIAS, Dimitrios Michael; SHAMI, Abdallah. Pwpae: An ensemble framework for concept drift adaptation in iot data streams. **arXiv preprint arXiv:2109.05013**, 2021. Citado 5 vezes nas páginas 23, 24, 31, 34 e 35.

YU, Hang; LU, Jie; ZHANG, Guangquan. Morstreaming: A multioutput regression system for streaming data. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, p. 1–13, 2021. Citado 2 vezes nas páginas 11 e 12.

ZENOBI, Gabriele; CUNNINGHAM, Padraig. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In: SPRINGER. **European Conference on Machine Learning**. Dublin, Ireland, 2001. p. 576–587. Citado na página 17.

ŽLIOBAITĖ, Indrė. Learning under concept drift: an overview. **arXiv preprint arXiv:1010.4784**, 2010. Citado na página 21.