PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

Fernando Pereira dos Santos

Seleção dinâmica de conjuntos de classificadores para mineração de fluxos de dados baseada em dominância local-global e amostragem online variável

MESTRADO EM
CIÊNCIA DA COMPUTAÇÃO
PUCPR

CURITIBA
2026

FERNANDO PEREIRA DOS SANTOS

Seleção dinâmica de conjuntos de classificadores para mineração de fluxos de dados baseada em dominância local-global e amostragem *online* variável

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.
Campo de Concentração: Ciência da Computação.

Pontifícia Universidade Católica do Paraná - PUCPR

Programa de Pós-Graduação em Informática - PPGIa

Orientador: FABRICIO ENEMBRECK

Curitiba - PR, Brasil

2026

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR – Biblioteca Central

Santos, Fernando Pereira dos
S237s Seleção dinâmica de conjuntos de classificadores para mineração de fluxos
2025 de dados baseada em dominância local-global e amostragem online variável / Fernando Pereira dos Santos ; orientador: Fabricio Enembreck. -- 2025
98 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2025.
Bibliografia: f. 87-98

1. Informática. 2. Fluxo de dados (Computadores). 3. Processo decisório por critério múltiplo. 4. Aprendizado do computador. 5. Aprendizagem de árvore de decisão. I. Enembreck, Fabricio. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título

CDD 20. ed. – 004

Bibliotecária: Edilene de Oliveira dos Santos CRB 9 / 1636

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 13 de fevereiro de 2026.

05-2026

DECLARAÇÃO

Declaro para os devidos fins, que **FERNANDO PEREIRA DOS SANTOS** defendeu a dissertação de Mestrado intitulada "**Seleção dinâmica de conjuntos de classificadores para mineração de fluxos de dados baseada em dominância local-global e amostragem *online* variável"**, na área de concentração Ciência da Computação no dia 18 de novembro de 2025, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

gov.br
Documento assinado digitalmente
JEAN PAUL BARDDAL
Data: 23/02/2026 10:42:27-0300
Verifique em https://validar.iti.gov.br

_____
Prof. Dr. Jean Paul Barddal
Coordenador do Programa de Pós-Graduação em Informática

Dedico esse trabalho a todos que um dia sonharam em se tornar cientistas.

Agradecimentos

Gostaria de agradecer a todos que contribuíram para a realização desta dissertação de mestrado. Este trabalho não seria possível sem o apoio, a compreensão e a colaboração de todos que estiveram ao meu lado ao longo dessa jornada.

Agradeço primeiramente ao meu orientador, Fabrício Enembreck, pelo conhecimento compartilhado, pela paciência e pela orientação cuidadosa, que foram fundamentais para o desenvolvimento deste estudo.

Aos colegas e amigos que estiveram ao meu lado nos momentos de incerteza, ofereço minha gratidão por todas as palavras de incentivo, pelas trocas de ideias e pela motivação constante. O apoio emocional e o incentivo de cada um de vocês foram fundamentais para que eu mantivesse o foco e a determinação ao longo do processo.

Agradeço especialmente à minha família, pelo carinho, compreensão e apoio incondicional. Vocês foram essenciais para que eu pudesse enfrentar os desafios e persistir em cada etapa desta trajetória.

Por fim, sou grato a todas as pessoas e instituições que, direta ou indiretamente, contribuíram para que este trabalho fosse concluído com sucesso. A cada um de vocês, meu sincero e profundo agradecimento.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"O universo é assimétrico e estou persuadido de que a vida,
como nós a conhecemos
é resultado direto da assimetria do Universo
ou de suas consequências indiretas."

Louis Pasteur

Resumo

A Aprendizagem de Máquina tem se destacado em diferentes áreas por viabilizar a descoberta de padrões a partir de grandes volumes de dados. Em alguns cenários, essa tarefa se torna mais desafiadora, pois é necessário identificar e adaptar padrões continuamente à medida que novos dados são gerados em forma de fluxo. Em cenários de mineração de fluxos de dados, técnicas baseadas em conjuntos de classificadores (*ensembles*) têm se destacado. Entretanto, assim como em cenários de aprendizagem *batch*, devido ao processo de geração dos membros do *ensemble*, nem todos classificadores são aptos a classificar todas as instâncias de teste disponíveis e, portanto, técnicas de seleção de classificadores se tornam necessárias. A técnica de seleção dinâmica de classificadores tem se mostrado eficaz para melhorar o desempenho de algoritmos de aprendizagem baseados em *ensembles*, mas sua aplicação no contexto de fluxo de dados e aprendizado *online* começou a ser explorada apenas recentemente. Este trabalho apresenta uma nova técnica para a seleção dinâmica de conjuntos de classificadores para fluxos de dados baseada em um critério de seleção de dominância local-global, onde os modelos locais (*base learners*) selecionados são aqueles que apresentam performance individual de curto prazo superior à performance média global do *ensemble*. Essa estratégia de seleção permite uma adaptação da constante de *Poisson* no processo de treinamento dos modelos base, tendo em vista a segmentação do *ensemble* entre modelos selecionados e não selecionados. Dessa forma, é possível reduzir a intensidade de treinamento no conjunto selecionado e aumentar no conjunto não selecionado e vice-versa. A abordagem visa melhorar a precisão do *ensemble* e aprimorar questões de desempenho (tempo de processamento e uso de memória) tendo em vista que *ensembles* podem depender de uso excessivo de poder computacional e as técnicas de seleção dinâmica para fluxos de dados existentes na literatura ainda acrescentam *overhead* significativo de processamento. Nos experimentos foram utilizados 34 datasets, sendo 17 reais e 17 sintéticos. As implementações propostas obtiveram os melhores resultados gerais em relação ao ARF (*Adaptive Random Forest*), um dos algoritmos de referência de estado da arte em fluxos de dados contínuos, tanto nas bases reais como nas sintéticas. Os experimentos também mostram melhores resultados do que outras abordagens recentemente publicadas sobre seleção dinâmica de classificadores para fluxos de dados.

Palavras-chave: Fluxos de Dados, Conjuntos de Classificadores, Seleção Dinâmica de Classificadores, Adaptive Random Forests.

Abstract

Machine Learning has stood out in various fields by enabling the discovery of patterns from large volumes of data. In some scenarios, this task becomes more challenging because it is necessary to continuously identify and adapt to patterns as new data are generated in a streaming form. Techniques based on classifier ensembles have shown great promise in data stream mining scenarios. However, as in batch learning scenarios, owing to the process of generating ensemble members, not all of them are capable of classifying all available test instances. Therefore, classifier-selection techniques have become necessary. Dynamic classifier selection has proven effective in improving the performance of ensemble-based learning algorithms; however, its application in the context of data streams and online learning has only recently been explored in the literature. This study presents a new technique for the dynamic selection of classifier ensembles for data streams based on a local-global dominance selection criterion. In this approach, the selected local models (base learners) exhibit individual short-term performances that are superior to the average global performance of the ensemble. This selection strategy allows the adaptation of the Poisson constant in the training process of the base models, considering the segmentation of the ensemble between the selected and non-selected models. In this way, it becomes possible to reduce the training intensity for the selected set and increase it for the non-selected set and vice versa. The proposed approach aims to improve ensemble accuracy and enhance performance aspects (processing time and memory usage), considering that ensembles may rely on extensive computational power and that existing dynamic selection techniques for data streams still introduce significant processing overheads. In the experiments, 34 datasets were used: 17 real datasets and 17 synthetic datasets. The proposed implementations achieved the best overall results compared to the ARF (Adaptive Random Forest), one of the state-of-the-art reference algorithms for continuous data streams, on both real and synthetic datasets. The experiments also show better results than other recently published approaches for dynamic classifier selection for data streams.

Keywords: Data Streams, Ensembles of Classifiers, Dynamic Classifier Selection, Adaptive Random Forests.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADWIN	Adaptive Windowing
ARE	Adaptive Regularized Ensemble
ARF	Adaptive Random Forest
ARTE	Adaptive Random Tree Ensemble
ARE	Adaptive Relularized Ensemble
OLA	Overall local accuracy
LCA	Local class accuracy
META-DES	Meta-learning para seleção dinânica de classificadores
HaO-DES	Hardness-aware Oracle with Dynamic Ensemble Selection
DES-A	Dynamic ensemble selection based on the algorithm Shapley
DYNSE	Dynamic Selection Based Drift Handler
DESDD	Dynamic Ensemble Selectin for Drift Detection
DCS-LA	Dynamic Classifier Selection wtih Local Accuracy
DCS-MCB	Dynamic Classifier Selection based on Multiple Classifer Behavior
DDCS	Double Dynamic Classifier Classifier Selection
DESW-ID	Dynamic ensemble selection based on window over imbalanced drift data stream
MSTS	Mass-based Short Term Selection
MSTS	Mass-based Short Term Selection with Adaptive Random Forest
SRP	Sreaming random patches

Sumário

1 Introdução

A quantidade de informação disponível em diversas plataformas tecnológicas na internet é imensa, e esse volume continuará a crescer, formando um fluxo de dados contínuo e de alta frequência. A análise dessa informação em tempo real deve ser atualizada constantemente e se adaptar em virtude da velocidade desse fluxo de dados, uma vez que podem ocorrer alterações nas distribuições dos dados e em suas classes ao longo do tempo (PéREZ, 2018).

Atualmente, existem diversos modelos de aprendizagem de máquina, sendo que os mais tradicionais processam os dados de forma *offline*, também conhecida como processamento em *batch*. Esses modelos de processamento *offline* treinam e processam dados utilizando todo o conjunto de informações disponível de uma só vez, o que os torna altamente custosos em termos computacionais (HULTEN; SPENCER; DOMINGOS, 2001). Em resposta a esse cenário, observou-se o surgimento do processamento/mineração de fluxo de dados de forma *online* (em tempo real). Este tipo de processamento busca equilibrar o uso dos recursos computacionais, mantendo o alto desempenho do modelo, de modo que o algoritmo não precise processar uma grande quantidade de dados simultaneamente.

Dentre os vários algoritmos de aprendizagem de máquina (*offline* e *online*) orientados à classificação, uma das estratégias mais utilizadas é a baseada em conjuntos de classificadores (*ensembles*), onde a decisão geral do *ensemble* é dada pela combinação da decisão de cada membro do conjunto de classificadores. Para realizar essa combinação, geralmente é utilizado o voto majoritário ou o voto ponderado, com eles obtendo maior precisão que um algoritmo de classificação monolítico (OZA, 2005), devido à maior diversidade de pontos de vista sobre o mesmo problema de aprendizagem.

Um dos algoritmos de *ensembles* de estado da arte mais empregados para o processamento de dados em tempo real é o *Adaptive Random Forests* (ARF) (GOMES et al., 2017), que constitui uma adaptação do tradicional *Random Forest* (BREIMAN, 2001). Este algoritmo utiliza um método robusto de reamostragem inspirado no *Online Bagging* (OZA, 2005), juntamente com uma estratégia adaptativa de atualização para ambientes de fluxos de dados. Tal estratégia é implementada através da constante de distribuição de *Poisson*, a qual define a quantidade média de vezes que cada instância será utilizada para o treinamento do classificador base (*base learner*), promovendo diversidade entre os *learners* que são treinados com instâncias diferentes e em quantidades variadas de vezes.

Diversos estudos na área têm sido conduzidos para desenvolver novos métodos de *ensembles* com o intuito de aprimorar a performance e o desempenho, sendo a Seleção Dinâmica de Classificadores (BRITTO; SABOURIN; OLIVEIRA, 2014), uma das áreas de

pesquisa mais proeminentes. A premissa dessa abordagem é que nem todos os classificadores do *ensemble* são eficazes para prever todas as instâncias do fluxo de dados. Assim, para cada nova instância de teste, os classificadores mais adequados devem ser selecionados e posteriormente combinados para a decisão final do *ensemble*. Conforme evidenciado na literatura, algumas técnicas de seleção dinâmica, como o DYNSE (ALMEIDA et al., 2016) e o DESDD (ALBUQUERQUE et al., 2019), tendem a melhorar a performance dos *ensembles*, mas introduzem uma sobrecarga significativa de processamento, sendo este *trade-off* o principal fator motivador para esta pesquisa.

1.1 Objetivos

Este estudo sugere estratégias para a seleção dinâmica de classificadores e as implementa inicialmente no algoritmo ARF, com o objetivo de aprimorar sua performance e eficiência.

Dessa forma o presente projeto visa avaliar se o monitoramento de desempenho de curto prazo tanto dos *base learners* quanto do *ensemble*, em conjunto com o ajuste dos valores da distribuição da constante de *Poisson*, pode resultar em melhorias de performance e eficiência em *ensembles* orientados a fluxos de dados. As duas estratégias principais a serem investigadas são denominadas (i) dominância local-global e (ii) amostragem *online* variável. A dominância local-global assume que em alguns contextos a performance local de classificadores base pode ser superior à do *ensemble* (performance global), sendo que nesses casos é melhor selecionar apenas esses *learners* para votação. A amostragem *online* assume que alguns *learners* selecionados já estão aptos a classificar determinadas instâncias, não sendo necessário treiná-los repetidamente com elas, ou seja, é possível reduzir o valor da constante de *Poisson* para economizar recursos sem comprometer o desempenho. Os objetivos específicos incluem:

- Avaliar o impacto da estratégia de seleção baseada em dominância local-global nos resultados do *ensemble*.
- Avaliar o impacto do uso combinado da estratégia de seleção e da variação do valor *Poisson* nos resultados do *ensemble*.
- Realizar experimentos em contextos variados de forma a caracterizar os benefícios ou deficiências das abordagens implementadas em relação ao *ensemble* original e às estratégias de seleção dinâmica de classificadores disponíveis na literatura de mineração de fluxos de dados.

1.2 Hipótese

A abordagem de dominância local-global emprega uma janela deslizante de curto prazo para a extração de estatísticas, selecionando-se os classificadores cuja performance local supera a performance global do *ensemble*. Tanto para os classificadores selecionados quanto para os não selecionados, são aplicadas estratégias de redução de treinamento, variando-se o valor médio da constante de *Poisson* utilizada no processo de reamostragem. A combinação dessas estratégias visa aprimorar a performance e o desempenho do *ensemble*.

A hipótese principal defendida neste estudo é a seguinte:

- A aplicação conjunta da estratégia de seleção por Dominância Local-Global e de valores adaptados da constante *Poisson* resulta na produção de *ensembles* que são mais precisos e eficientes do que aqueles gerados pelo algoritmo ARF, bem como em comparação com as técnicas de seleção de conjuntos de classificadores disponíveis na literatura no contexto de mineração de fluxos de classificadores.

1.3 Contribuições

O projeto tem as seguintes contribuições:

- Criação de uma nova técnica de seleção dinâmica de *ensembles* com a estratégia de dominância local-global para o algoritmo ARF;
- Criação de estratégias dinâmicas de variação da quantidade de treinamento induzido pelo valor médio da constante de *Poisson* para o processo de reamostragem que o ARF utiliza;
- Estudos comparativos entre as abordagens desenvolvidas, os ensembles originais induzidos pelo ARF e as técnicas de seleção dinâmica da literatura em bases de dados reais e sintéticas.

1.4 Visão Geral

Este trabalho está estruturado da seguinte maneira: o Capítulo 2 introduz conceitos fundamentais de mineração em fluxo de dados, bem como as métricas para avaliação de desempenho de modelos preditivos. O Capítulo 3 aborda definições relacionadas à seleção dinâmica de classificadores, discutindo também alguns algoritmos de ponta utilizados tanto em ambientes *batch* quanto em ambientes *online*. O Capítulo 4 apresenta o método proposto, detalhando seus principais conceitos. O Capítulo 5 expõe os resultados obtidos a partir de uma série de experimentos realizados em diversos conjuntos de dados. Por fim,

o Capítulo 6 discute as principais conclusões da pesquisa e sugere possíveis direções para trabalhos futuros.

2 Mineração em Fluxos de Dados

A mineração de fluxo de dados constitui um processo amplamente utilizado para a extração e análise contínua de informações, visando a obtenção de conhecimentos úteis a partir de dados em diversas aplicações. Exemplos notáveis incluem a detecção de fraudes em tempo real, a análise de redes sociais e outras aplicações em larga escala.

Este capítulo oferece uma introdução aos principais conceitos relacionados à Mineração de Fluxo de Dados, essenciais para a compreensão deste estudo, incluindo a descrição de alguns algoritmos e suas métricas de avaliação de desempenho.

2.1 Classificação em Fluxos de Dados

Os algoritmos de Aprendizagem de Máquina Supervisionada têm como objetivo identificar padrões a partir de um conjunto de exemplos rotulados. Esses exemplos, também denominados instâncias, sendo (x_1, y_2), (x_2, y_2), ..., (x_n, y_n) (CAVALHEIRO et al., 2021), onde a variável x denota um conjunto de características descritivas e y seus respectivos rótulos. O propósito dessa técnica é a construção de um modelo preditivo de classificação ou regressão, cuja saída consiste na previsão do rótulo y para o conjunto de características (BURKOV, 2019).

Os modelos de mineração em *batch* são treinados utilizando a base de dados completa. Com o tempo, esses modelos podem se tornar desatualizados, exigindo a reconstrução de um novo modelo, uma vez que novos dados podem apresentar padrões distintos daqueles observados no conjunto de treinamento original. Em contraste, os modelos de aprendizagem *online* processam continuamente os dados gerados em fluxo, sendo treinados e atualizados de forma contínua para se adaptarem às mudanças em sua distribuição (BIFET; KIRKBY, 2019).

Devido a essa análise contínua, com treinos e atualizações, os algoritmos desenvolvidos para trabalhar com fluxos de dados contínuos devem atender a quatro requisitos importantes (BIFET; KIRKBY, 2019):

- **O processamento de uma única instância por vez**

A principal característica dos fluxos de dados está na chegada contínua dos dados e em seu processamento imediato. O processamento pode ocorrer com uma instância por vez, ou em mini-lotes de tamanhos fixos ou variáveis, mas em geral é assumido que eles não podem ser reprocessados uma vez que eles não são armazenados integralmente. Caso

o algoritmo necessite armazenar pequenas amostras de dados temporariamente para a reutilização posterior, ele deve prever estratégias de controle interno dos recursos utilizados.

- **Limitação do uso de memória**

Trata do uso eficiente de memória, visto que o algoritmo precisa tratar quantidade possivelmente infinita de dados, sendo impossível armazenar todos eles em RAM.

- **Limite de tempo para o processamento em tempo real**

O algoritmo precisa trabalhar em tempo real, processando as instâncias do fluxo de dados de forma mais rápida que o processo de geração de dados, caso contrário, é possível haver perda de dados.

- **Realização da predição em tempo real**

A produção de um modelo preciso que permanece disponível para previsões contínuas à medida que novas instâncias são introduzidas.

A aplicação desses requisitos por algoritmos de fluxos de dados contínuos para o tratamento de grandes volumes de dados deve ser gerida de maneira responsável, considerando os elevados custos de processamento computacional. (BIFET; KIRKBY, 2019).

2.1.1 Métricas de Avaliação de Performance

Para a avaliação dos classificadores, pode-se empregar um conjunto de métricas existentes, as quais são dependentes do tipo de análise que se pretende realizar, expressando o desempenho do modelo durante a classificação das instâncias do fluxo de dados. A seguir, são apresentadas algumas métricas importantes que serão posteriormente utilizadas ao longo deste trabalho.

A acurácia é a métrica mais amplamente reconhecida e simples, empregada na avaliação da frequência de previsões corretas, sendo definida pela proporção de previsões corretas em relação ao total de previsões realizadas (equação 2.1) (BISHOP, 2006).

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Sendo:

TP o número de verdadeiros positivos

TN o número de verdadeiros negativos

FP o número de falsos positivos

FN o número de falsos negativos

A Estatística Kappa, também conhecida como coeficiente Kappa de Cohen, é usada para avaliar a concordância entre as previsões do modelo e as observações reais das instâncias (COHEN, 1960), sendo ela dada pela equação 2.2.

$$\text{Kappa} = \frac{f_O - f_E}{N - f_E} \tag{2.2}$$

Onde:

f_O a frequência observada

f_E a frequência esperada

N o número total de observações

Neste estudo, essa métrica é empregada durante o processo de seleção dos classificadores, justificando-se por ser a mais utilizada na literatura atual para trabalhar com bases de dados desbalanceadas (BRZEZINSKI; STEFANOWSKI, 2018).

O Tempo de Processamento quantifica a duração do processamento do modelo realizado pela CPU, expresso em segundos, e pode ser representado pela equação 2.3 (GAMA; GABER, 2007).

$$T_{proc} = \frac{\sum_{i=1}^{N} t_i}{N} \tag{2.3}$$

Sendo:

T_{proc} o tempo médio de processamento por amostra

N o número total de amostras processadas

t_i tempo de processamento da i-ésima amostra

O Custo do Modelo quantifica a dimensão ou o custo de memória consumida durante o processamento do modelo, expresso em *RAM-Hours*, conforme representado pela equação 2.4 (BIFET et al., 2018).

$$C_{modelo} = \alpha \cdot T_{proc} + \beta \cdot M \tag{2.4}$$

onde:

C_{modelo} o custo total do modelo

T_{proc} o tempo médio de processamento por amostra, dado em horas

M uso médio de memória, dado em Gigabytes (G) da memória RAM

α **e** β os coeficientes que demonstram a importância do tempo de processamento e do uso de memória no cálculo do custo.

2.1.2 Procedimentos de Avaliação de Classificadores

O procedimento de avaliação de um classificador constitui uma das tarefas mais cruciais no desenvolvimento de modelos de aprendizagem de máquina, devendo sua escolha ser realizada em conformidade com o problema a ser abordado. Este procedimento determina quais instâncias serão utilizadas nas fases de treino e teste do algoritmo, possibilitando a extração de métricas a partir de uma janela temporal (BIFET et al., 2010).

O processamento tradicional (*batch*) frequentemente utiliza técnicas como *validação cruzada* e *holdout*, que dividem a base de dados em partições de treinamento e teste para a avaliação do modelo. No contexto de fluxos contínuos de dados, a chegada ilimitada de dados apresenta desafios distintos para os procedimentos de avaliação, uma vez que não é possível estabelecer *a priori* partições de treinamento e teste. Para viabilizar a avaliação de algoritmos de aprendizagem neste contexto, foi desenvolvida a técnica *Prequential*, ou *Interleaved Test-Then-Train*. Esta técnica utiliza cada instância que chega para a avaliação do modelo, ou seja, a instância é primeiramente utilizada para o teste e, posteriormente, para o treino do modelo (princípio *Test-Then-Train*), mantendo as métricas de desempenho sempre atualizadas ao longo do tempo (DAWID, 1984).

Outro método empregado é o *Holdout* Periódico (*Periodic Holdout*), que consiste em aplicar o tradicional protocolo de *Holdout* de forma repetida. No início do processamento do fluxo de dados, as primeiras N instâncias são utilizadas exclusivamente para o treino do modelo. Subsequentemente, as próximas M instâncias são utilizadas exclusivamente para o teste do modelo (CAVALHEIRO et al., 2021). Este processo é repetido enquanto houver instâncias a serem processadas.

Para os experimentos conduzidos neste estudo, foi empregada a técnica *Prequential*, dado que é amplamente utilizada na literatura, o que facilita comparações e reprodutibilidade. Ademais, as métricas *Acurácia* e *Kappa* foram analisadas para a apresentação dos resultados de performance, enquanto as métricas *Tempo de Processamento* e *Custo do Modelo* foram utilizadas para a avaliação do desempenho.

2.1.3 Classificadores Monolíticos para Fluxos de Dados

Conforme mencionado em seções anteriores, alguns algoritmos empregam conjuntos de classificadores para realizar predições. Cada membro desse conjunto é um modelo individual e contribui com um peso específico na decisão final do conjunto. Os algoritmos monolíticos mais explorados na literatura são discutidos brevemente nas seções seguintes.

2.1.3.1 Naive Bayes

Este algoritmo foi desenvolvido com base no *Teorema de Bayes*, conforme a equação 2.5, sendo idêntico ao utilizado na versão em *batch* para tarefas de classificação. Ele pressupõe que todas as características do fluxo de dados são independentes entre si; no entanto, essa suposição pode não ser sempre válida devido a vieses entre as características das instâncias (BISHOP, 2006).

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)} \tag{2.5}$$

Essa simplificação da independência aumenta a eficiência, permitindo seu funcionamento em diversos cenários práticos, mesmo quando tais características apresentam alta correlação e a independência total não é observada.

2.1.3.2 Hoeffding Tree

A *Hoeffding Tree* é um algoritmo de classificação baseado em árvore de decisão, adaptado para ambientes online. Este algoritmo realiza aprendizado contínuo a cada instância do fluxo de dados, gerando estruturas semelhantes às árvores de decisão utilizadas no processamento convencional em *batch*, desde que haja uma quantidade suficiente de instâncias. Ademais, este algoritmo não requer o armazenamento completo dos dados na memória. Ele se baseia em decisões periódicas de expansão da árvore, utilizando um resultado estatístico conhecido como limites de *Hoeffding* (*Hoeffding Bound*) (DOMINGOS; HULTEN, 2000), representados pela equação 2.6.

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \tag{2.6}$$

Onde R é o intervalo da métrica avaliada, δ o nível de confiança, e n o número de instâncias observadas.

A presente equação permite determinar, com uma probabilidade mínima de erro δ, quando a diferença entre dois atributos é estatisticamente significativa para justificar uma divisão no nó da árvore. Esta equação é utilizada durante o treinamento do modelo para decidir quando e a partir de qual atributo a árvore deve iniciar sua expansão.

2.2 Conjuntos de classificadores

Um conjunto de classificadores, ou *ensemble of classifiers*, representa uma alternativa aos algoritmos que empregam um único classificador. Este conjunto é constituído por um grupo de classificadores, cada um dos quais possui uma decisão individual.

Durante este processo, é possível aplicar algum tipo de seleção nos classificadores, seguido pela sua combinação, seja por voto majoritário ou ponderado, resultando no resultado global do algoritmo.

Os algoritmos apresentados na seção 2.1.3 são alguns dos tipos mais utilizados para a construção de conjuntos de classificadores. Conforme discutido no trabalho (DIETTERICH, 2000), existem três motivos fundamentais para a utilização de conjuntos de classificadores:

- **Estatística**: A quantidade insuficiente de dados para o treinamento do modelo, em comparação com o espaço de possibilidades que o algoritmo está explorando, pode resultar em soluções distintas para o problema. Posteriormente, a combinação dos resultados de cada membro é realizada, obtendo-se um valor geral mais robusto.

- **Computacional**: A dificuldade que os algoritmos enfrentam para encontrar a melhor solução devido ao seu processamento é notável. Muitos utilizam algum tipo de busca local que pode comprometer computacionalmente o resultado, levando cada algoritmo a convergir para soluções diferentes. A combinação dos resultados do conjunto consegue, assim, encontrar uma maior aproximação com o resultado real.

- **Representacional**: Um modelo unitário pode não identificar uma solução que atenda ao problema proposto. Isso pode ocorrer devido a alguma limitação do algoritmo ou inconsistência nos dados. A combinação de resultados pode, no entanto, produzir bons indicadores, mesmo quando os membros do conjunto não apresentam desempenho excepcional de forma isolada.

Por outro lado, pode-se afirmar que, quando o conjunto de classificadores é composto por membros que são tanto precisos quanto diversos, ele supera qualquer membro individual do conjunto. Dois classificadores são considerados diversos se cometem erros diferentes nos mesmos pontos de dados (DIETTERICH, 2000).

2.3 Mudanças de Conceito

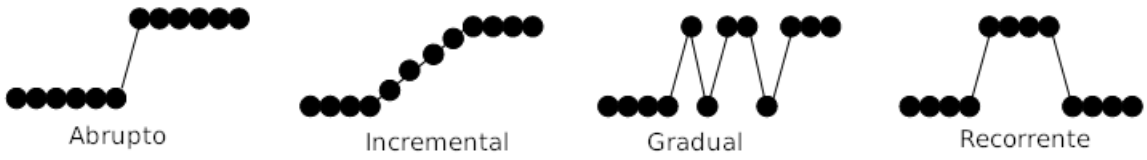
Mudanças de conceito (*Concept Drifts*) são eventos que podem ocorrer em qualquer momento e impactar a dinâmica dos fluxos de dados (SCHILIMMER; GRANGER, 1986). Um exemplo concreto é o impacto nos dados gerados pelas variações diárias do mercado

financeiro ou pela influência de temas populares nas redes sociais. Essas mudanças podem manifestar-se em diferentes velocidades, de maneira abrupta ou gradual, com uma frequência que evolui ou se repete ao longo do tempo.

Detectar mudanças de conceitos pode ser desafiador em muitos problemas reais, pois elas podem refletir contextos ocultos e externos aos dados (*Hidden Context*). Portanto, é essencial que os algoritmos de fluxos de dados sejam capazes de identificá-las e adaptar-se a elas (TSYMBAL, 2005).

Elas podem ocorrer em diferentes velocidades, sendo conceitualmente classificadas em quatro tipos, conforme ilustrado na Figura 1. Mudanças abruptas ocorrem quando as instâncias alteram rapidamente sua distribuição de um conceito para outro. A mudança de conceito incremental ocorre quando a frequência de instâncias de um conceito aumenta gradualmente ao longo do tempo. As mudanças graduais ocorrem quando a velocidade diminui gradualmente, à medida que instâncias do fluxo de dados de determinado conceito se tornam menos frequentes ao longo do tempo. Por fim, a mudança recorrente acontece em ciclos temporais regulares (TSYMBAL, 2005).

Figura 1 – Tipos de mudanças de conceito



Fonte: Adaptado de (GAMA et al., 2014)

Os algoritmos de classificação em fluxo de dados devem operar nesse contexto com mecanismos para detectar e se adaptar aos dados que chegam no decorrer do tempo, devendo atender aos seguintes requisitos (GAMA et al., 2014):

- Detecção da mudança de conceito o mais rápido possível.
- Saber diferenciar as mudanças de conceito de *outliers* ou ruídos no fluxo de dados.
- Processar as instâncias no menor tempo possível utilizando uma quantidade fixa de memória.

2.4 Detectores de Mudanças

Os detectores são técnicas empregadas para identificar alterações de desvio de conceitos em fluxos de dados, sendo amplamente discutidos na literatura contemporânea. Além disso, eles podem ser aplicados de outras maneiras, como em estratégias para a seleção de conjuntos de classificadores, destacando-se por sua capacidade de indicar e sinalizar mudanças significativas na distribuição dos dados ao longo do tempo (GAMA et al., 2014).

Diversos métodos para a detecção de mudanças são descritos na literatura, como o *CUSUM (Cumulative Sum)*. Esta técnica é caracterizada pela análise sequencial dos dados, detectando mudanças de conceito quando a soma cumulativa de erros ocorridos durante a predição atinge um determinado limite (PAGE, 1954). Outro método, conhecido como *PH (Page-Hinkley)*, semelhante ao CUSUM, realiza a detecção de mudanças por meio do monitoramento do aumento significativo da média da soma cumulativa dos erros ocorridos durante a previsão (PAGE, 1954).

O *Adaptive Windowing* (ADWIN) monitora uma janela deslizante de tamanho variável de instâncias. Esta janela é automaticamente expandida quando nenhuma alteração é detectada e reduzida quando são identificadas mudanças no fluxo de dados, indicadas por uma variação significativa de erro entre duas partições adjacentes da janela, conforme determinado pelo teste de *Hoeffding* (BIFET; GAVALDà, 2007).

No estudo (GAMA; MEDAS; RODRIGUES, 2004), é apresentada a técnica *Drift Detection Method (DDM)*, na qual a detecção é realizada por meio do monitoramento do número de erros durante o processo de previsão. Caso esse número de erros aumente significativamente em relação à média e ao desvio padrão esperados, uma mudança de conceito é detectada.

Em (BAENA-GARCíA et al., 2006), é introduzido o *Early Drift Detection Method (EDDM)*, que se assemelha ao algoritmo DDM. Este método calcula as distâncias entre as instâncias classificadas como erro durante o processo de detecção. Ao longo do monitoramento, uma mudança é identificada se o valor dessas distâncias aumentar significativamente.

É relevante destacar que o presente estudo não pretende se aprofundar nos métodos de detecção de mudanças, considerando a extensa literatura existente sobre o tema e a ampla variedade de métodos disponíveis na área. O leitor interessado pode consultar alguns estudos comparativos recentes disponíveis em (BARROS et al., 2017; FRíAS-BLANCO et al., 2015; BARDDAL et al., 2017; GAMA et al., 2014).

2.5 Conjuntos de Classificadores para fluxos contínuos de dados

Conforme apresentado na Seção 2.2, os conjuntos de classificadores consistem em uma combinação de múltiplos modelos, cada um especializado em diferentes aspectos de um problema, o que geralmente resulta em uma solução superior àquela obtida por um único modelo. A diversidade dentro de um conjunto de classificadores aumenta a probabilidade de que um membro complemente os outros, melhorando assim as métricas de desempenho do conjunto.(CAVALHEIRO et al., 2021).

Cada membro do conjunto contribui com um voto para a predição do problema, e esses votos são então combinados em uma única predição. O método mais comum para essa combinação é o voto majoritário, no qual a classe mais votada pelos membros do conjunto é selecionada. A estratégia de voto ponderado atribui um peso ao voto de cada classificador, considerando uma métrica individual de desempenho, como a acurácia ou a estatística Kappa.

Nas subseções seguintes, são apresentados alguns algoritmos utilizados para a construção de conjuntos de classificadores em fluxos de dados. No entanto, este trabalho visa investigar exaustivamente o estado da arte. Os métodos apresentados possuem alguma similaridade conceitual ou estrutural com a proposta deste estudo.

2.5.1 Online Bagging

O *Online Bagging* (OZA, 2005) constitui uma adaptação da estratégia tradicional de *Bagging* (BREIMAN, 1996), originalmente desenvolvida para ambientes *offline*, para contextos *online*. Este método é detalhado no Algoritmo 1, onde, durante o treinamento do *ensemble*, é gerado um conjunto de amostras de tamanho N (*bags*) de instâncias para o treinamento de cada membro do conjunto. O tamanho dessas amostras é regulado por uma variável k, que segue uma distribuição de *Poisson* ($\lambda = 1$), conforme indicado na linha 4 do Algoritmo 1. Esta variável determina o número de vezes que a instância será utilizada no treinamento dos classificadores base, sendo que uma ou mais cópias das instâncias são empregadas em aproximadamente 63% das vezes.

A classificação de uma instância é realizada por meio de uma votação não ponderada, onde cada membro do conjunto possui o mesmo peso na decisão final, conforme descrito na linha 14 do Algoritmo 1.

Algorithm 1 Online Bagging (OZA, 2005)

Require: Base learning algorithm L, number of models M, data stream D
Initialize M base models $h_1, h_2, \ldots, h_M$ using L
for each incoming example (x, y) in D **do**
    for $m \leftarrow 1$ to M **do**
        $k \leftarrow Poisson(1)$
        for $t \leftarrow 1$ to k **do**
            Update model $h_m \leftarrow L_o(h_m, (x, y))$
        end for
    end for
end for

function PREDICT(x)
    for $m \leftarrow 1$ to M **do**
        $p_m \leftarrow h_m(x)$
    end for
    Combine predictions $\{p_1, p_2, \ldots, p_M\}$ by majority vote (for classification) or average (for regression)
end function

2.5.2 Adaptive Random Forest

O *Adaptive Random Forest* (ARF), conforme apresentado em (GOMES et al., 2017), constitui uma adaptação do algoritmo de aprendizagem de máquina mais clássico e amplamente utilizado, o *Random Forest (RF)* (BREIMAN, 2001). Este algoritmo promove a criação e o desenvolvimento de árvores utilizando diferentes subconjuntos de instâncias e características, permitindo sua execução em paralelo sem comprometer o desempenho de classificação, conforme demonstrado nos Algoritmos 2 e 3.

O ARF incorpora um método eficaz de reamostragem (*resampling*) e operadores adaptativos capazes de lidar com diversos tipos de mudanças de conceito sem a necessidade de otimizações complexas para diferentes tipos de dados. Este algoritmo tem se mostrado robusto em variados problemas, combinando as características de algoritmos de processamento *off-line* (*batch*) com métodos de atualização dinâmica para gerenciar fluxos de dados em evolução.

Como o ARF é um algoritmo de *ensembles*, cada membro do conjunto é uma *Hoeffding Tree*, sendo que cada árvore é treinada com subconjuntos de instâncias processadas mais de uma vez, de maneira semelhante ao *Online Bagging*, apresentado na seção 2.5.1. O crescimento das árvores é controlado por divisões baseadas na escolha do melhor atributo selecionado a partir de um subconjunto aleatório de m características em cada nó folha.

O algoritmo incorpora dois detectores de mudanças para cada árvore criada. O primeiro realiza a detecção de possíveis alterações no fluxo de dados, gerando assim um alerta, enquanto o segundo confirma a detecção de mudança. Quando um alerta é

detectado, inicia-se o treinamento de novas árvores em segundo plano e, caso a mudança seja confirmada, ocorre a substituição da árvore pelo classificador de segundo plano. O algoritmo não se limita a um único detector de mudança; contudo, é comumente associado ao ADWIN (BIFET; GAVALDà, 2007) com dois intervalos de confiança (um para cada nível).

Como mencionado anteriormente, a amostragem aleatória empregada pelo ARF baseia-se no *Online Bagging* (OzaBag) (OZA, 2005), distinguindo-se na escolha do parâmetro λ da função de $Poisson(\lambda)$. No ARF, $\lambda = 6$, enquanto o OzaBag utiliza $\lambda = 1$. Essa abordagem possibilita a utilização de um número maior de instâncias no treinamento dos classificadores base, promovendo maior diversidade no *ensemble*.

A classificação das instâncias é realizada por meio da ponderação dos votos, utilizando como base a acurácia obtida no *test-and-train* de cada árvore. As árvores que apresentarem melhor desempenho terão uma influência mais significativa na decisão final, embora outras estratégias de ponderação possam ser empregadas.

Algorithm 2 RFTree Train. **Symbols:** λ is a fixed parameter of the Poisson distribution; GP is the grace period before recalculating heuristics for the split test. (GOMES et al., 2017)

```
1: function RFTreeTrain(m, t, x, y)
2: k ← Poisson(λ = 6)
3: if k > 0 then
4:     l ← FindLeaf(t, x)
5:     UpdateLeafCounts(l, x, k)
6:     if InstancesSeen(l) ≥ GP then
7:         AttemptSplit(l)
8:         if DidSplit(l) then
9:             CreateChildren(l, m)
10:        end if
11:    end if
12: end if
13: end function
```

Algorithm 3 Adaptive Random Forests - **Symbols:** m is the maximum number of eatures evaluated per split; n is the total number of trees ($n = |T|$); δ_w is the warning threshold; δ_d is the drift threshold; $C(\cdot)$ is the change detection method; S is the data stream; B is the set of background trees; $W(t)$ is the weight of tree t; $P(\cdot)$ is the learning performance estimation function. (GOMES et al., 2017)

function AdaptiveRandomForests(m, n, δ_w, δ_d)
$T \leftarrow$ CreateTrees(n)
$W \leftarrow$ InitWeights(n)
$B \leftarrow \emptyset$
while HasNext(S) **do**
    $(x, y) \leftarrow$ next(S)
    for all $t \in T$ **do**
        $\hat{y} \leftarrow$ predict(t, x)
        $W(t) \leftarrow P(W(t), \hat{y}, y)$
        RFTreeTrain(m, t, x, y) ▷ Train t on the current instance (x, y)
        if $C(\delta_w, t, x, y)$ **then**
            $b \leftarrow$ CreateTree()
            $B(t) \leftarrow b$ ▷ Init background tree
        end if
        if $C(\delta_d, t, x, y)$ **then**
            $t \leftarrow B(t)$ ▷ Replace t by its background tree
        end if
    end for
    for all $b \in B$ **do**
        RFTreeTrain(m, b, x, y) ▷ Train each background tree
    end for
end while
end function

2.5.3 Streaming Random Patches

O *Streaming Random Patches* (SRP) integra a seleção de subespaços aleatórios de características (*Random Subspaces*) e instâncias. Similar ao ARF, ele emprega um processo de reamostragem de instâncias inspirado no *Online Bagging* (OZA, 2005) para aprimorar o desempenho preditivo em comparação com outros métodos que utilizam *ensembles*, evidenciando que essa combinação pode ser uma alternativa eficaz (GOMES; READ; BIFET, 2019), com o pseudocódigo do algoritmo apresentado em 4.

A seleção aleatória de instâncias é idêntica à utilizada pelo ARF configurado para empregar a função de *Poisson* com $\lambda = 6$. Este valor tende a melhorar o desempenho preditivo dos *ensembles* à medida que os modelos-base são treinados, além de induzir maior diversidade; contudo, tende a demandar mais recursos computacionais devido ao aumento na quantidade de treinamento.

Com a implementação da seleção aleatória de características, o algoritmo introduz um método adicional de indução de diversidade nos modelos. No entanto, essa escolha

acarreta um risco ao algoritmo, uma vez que pode resultar na seleção de características irrelevantes, levando a uma predição de baixo desempenho em determinadas situações. Para mitigar esses riscos, foram incorporados dois mecanismos: (i) a reinicialização dos subespaços de características quando ocorre um desvio de conceito e (ii) a atribuição de pesos aos votos dos modelos com base em seu desempenho preditivo.

Algorithm 4 Streaming Random Patches. **Symbols:** k is the maximum number of features per subset; λ is the Poisson distribution parameter; M is the total number of models ($M = |L|$); δ_w is the warning threshold; δ_d is the drift threshold; S is the data stream; B is the set of background models; $W(l)$ is the weight of model l; $P(\cdot)$ is the model predictive performance estimation function; $d(\cdot)$ is the drift detection method. (GOMES; READ; BIFET, 2019)

function TrainSRP(k, n, δ_w, δ_d)
  $L \leftarrow$ CreateBaseModels(k, n) ▷ Assign random subspaces of size k to each base model
  $W \leftarrow$ InitWeights(n)
  $B \leftarrow \emptyset$
  while HasNext(S) **do**
    $(x, y) \leftarrow$ next(S)
    for all $l \in L$ **do**
      $\hat{y} \leftarrow$ predict(l, x)
      $W(l) \leftarrow P(W(l), \hat{y}, y)$
      Train(k, l, x, y)
      if $d(\delta_w, l, x, y)$ **then**
        $B(l) \leftarrow$ CreateBkgModel(k) ▷ Warning detected?
      end if
      if $d(\delta_d, l, x, y)$ **then** ▷ Drift detected?
        $l \leftarrow B(l)$ ▷ Replace l by background learner
      end if
    end for
    for all $b \in B$ **do**
      Train(k, b, x, y) ▷ Train each background learner
    end for
  end while
end function

2.6 Considerações Finais

Este capítulo tem como objetivo apresentar os conceitos e técnicas atualmente utilizados na literatura, aplicados a fluxos contínuos de dados, que foram empregados no desenvolvimento deste trabalho. O termo conjuntos de classificadores (*ensembles methods*) é frequentemente mencionado, sendo uma opção amplamente utilizada na mineração de fluxos de dados, oferecendo diversas vantagens para diferentes problemas de classificação de dados.

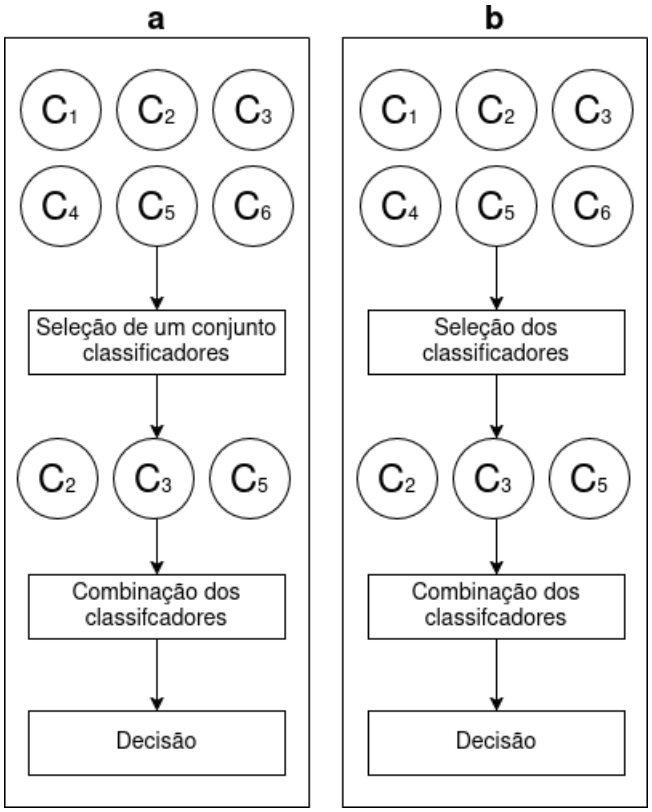
Esses conceitos servem como fundamentação teórica para a proposta deste trabalho, que será apresentada nos capítulos subsequentes, além de permitir a compreensão do funcionamento de alguns dos conjuntos de classificadores mais utilizados na literatura. As técnicas discutidas inspiraram o desenvolvimento de operações de seleção de classificadores e adaptação no processo de treinamento, na tentativa de obter resultados superiores ao algoritmo *Adaptive Random Forest*.

3 Seleção Dinâmica de Classificadores

Conforme discutido em capítulos anteriores, um dos fatores mais significativos para o uso de conjuntos de classificadores é a integração de cada membro do conjunto em uma única predição. Devido à diversidade presente nos conjuntos de classificadores, espera-se que seus membros cometam erros em diferentes instâncias. Assim, assume-se *a priori* que nem todos os membros do conjunto são adequados para a classificação de todas as instâncias. Isso motivou estudos sobre a aplicação da seleção de classificadores em *ensembles*, com o objetivo principal de selecionar apenas os membros mais competentes para a predição geral do conjunto.

Os métodos de seleção de classificadores podem ser classificados em dois tipos, conforme a literatura atual, ambos ilustrados na Figura 2. O primeiro tipo é a Seleção Estática (a), que ocorre uma única vez quando o mesmo padrão de seleção é repetido, e o subconjunto de classificadores selecionados é combinado para obter o voto final. O segundo método é a Seleção Dinâmica (b), na qual, para a predição de cada instância, é selecionado dinamicamente um subconjunto específico de classificadores, utilizando diferentes padrões de seleção. Este subconjunto pode conter um único membro, um subconjunto ou o conjunto completo de classificadores, e posteriormente ocorre a combinação dos votos.(CAVALHEIRO et al., 2021).

Figura 2 – Seleção e combinação de votos de classificadores, (a) representa a seleção estática de classificadores, ocorrendo uma única vez e a seleção dinâmica de classificadores (b), onde o processo se repete cada vez que uma nova instância chega.



Fonte: Adaptado de (KO; SABOURIN; JR., 2008)

Quanto à avaliação da competência de um classificador base para ser selecionado, a literatura atual apresenta diversas estratégias de *ranking* fundamentadas em acurácia local, informações probabilísticas, comportamento do classificador, representação do Oráculo (modelo conceitual ideal), entre outras (BRITTO; SABOURIN; OLIVEIRA, 2014). Em geral, essas técnicas avaliam a performance dos classificadores localmente, utilizando um conjunto de instâncias de referência conhecido como região de competência.

Este estudo concentra-se em métodos de seleção dinâmica de classificadores, considerando que, em um fluxo de dados, as instâncias são processadas individualmente. Assim, nas seções subsequentes, são apresentados métodos desenvolvidos tanto para ambientes em *batch* (ou *offline*) quanto para *online*.

3.1 Seleção dinâmica de classificadores em ambientes *offline*

A seleção dinâmica para ambientes *offline* é amplamente discutida na literatura. Entre suas características, destaca-se o processo de divisão da base de dados em conjuntos de treino e validação, onde os algoritmos são treinados com o conjunto de treino. Durante a fase de predição, o algoritmo *K-Nearest Neighbors* (FIX; HODGES, 1989) é empregado para identificar as instâncias mais semelhantes nos dados de validação, definindo assim uma região de competência. Os classificadores selecionados são aqueles que apresentam maior acurácia na classificação dentro dessa região de competência.

Essa característica, no entanto, limita a aplicação desses algoritmos em ambientes de fluxo de dados contínuos. Contudo, alguns algoritmos foram adaptados para ambientes *online*, como é o caso do *Bagging*, que evoluiu para o *Online Bagging* (OZA, 2005), e do *Random Forest*, que foi adaptado para o ARF (GOMES et al., 2017).

3.1.1 Overall Local Accuracy

O algoritmo *Overall Local Accuracy* (OLA), conforme apresentado em (WOODS; JR.; BOWYER, 1997), constitui um método de seleção que se baseia na estimativa de precisão em regiões locais, definidas por um espaço restrito de características, para selecionar o classificador mais preciso localmente. Esta região local é determinada pela utilização dos K-vizinhos mais próximos durante a fase de treinamento do modelo. A seleção do classificador é realizada por meio da comparação da acurácia local entre todos os classificadores disponíveis na região do espaço de características, optando-se por aquele que apresenta o maior percentual de acertos nessa região. Em caso de empate, e se os classificadores pertencerem a classes diferentes, o algoritmo resolve o impasse por meio de uma votação, selecionando a classe que recebeu o maior número de votos entre os classificadores empatados. A principal desvantagem do algoritmo reside na baixa estimativa de desempenho durante a seleção do melhor classificador para uma amostra.

3.1.2 Local Class Accuracy

Conforme apresentado em (WOODS; JR.; BOWYER, 1997), o algoritmo *LCA* emprega os mesmos princípios do algoritmo *OLA*, diferenciando-se pelo cálculo da porcentagem de exemplos corretamente atribuídos a cada classe. Durante o processo de seleção, cada membro do conjunto atribui individualmente um rótulo de classe a uma amostra desconhecida. Caso haja consenso total, a decisão é aceita; em caso de discordância, utiliza-se o método dos K-vizinhos mais próximos no conjunto de treinamento para estimar a competência de cada membro. Para cada membro que previu uma classe C_i, calcula-se a porcentagem de amostras nessa vizinhança local que foram corretamente rotuladas para a classe específica. Posteriormente, seleciona-se o classificador com a maior acurácia estimada,

resolvendo possíveis empates através das classes que apresentam maior frequência entre os classificadores com desempenho equivalente naquela região.

Em termos de resultados, o algoritmo LCA demonstrou desempenho superior ao algoritmo OLA em todos os conjuntos de dados utilizados, resultando em maior precisão na classificação. Isso evidencia que a distribuição de classes dentro de uma região local é o principal fator de sua superioridade (WOODS; JR.; BOWYER, 1997).

3.1.3 META-DES

O estudo apresentado em (CRUZ et al., 2015) propõe um esquema de seleção dinâmica de classificadores que se distingue dos métodos previamente apresentados. Este processo de seleção é denominado *meta-learning*, uma vez que é abordado como um problema de classificação distinto. O meta-aprendiz gerado possui a capacidade de avaliar a competência de um *base learner* para ser incorporado no *ensemble*.

O processo é dividido em três fases. A primeira fase é a superprodução (*overproduction*), durante a qual os conjuntos de classificadores são treinados. A segunda fase, o meta-treinamento (*meta-training*), envolve a extração de cinco meta-características (*meta-features*) dos dados de treino e o treinamento de meta-classificadores (*meta-classifiers*) que serão empregados na seleção dinâmica.

A terceira etapa, a generalização, envolve a extração das meta-características das instâncias de teste, seguida pela estimativa da competência do classificador em relação à instância de teste.

Cada grupo de meta-características captura diferentes propriedades do comportamento de um classificador base. A seleção é realizada por meio da observação do desempenho de classificação em uma região local do espaço de características e da confiança do classificador no resultado obtido na amostra de teste.

Os experimentos realizados demonstraram um alto grau de acurácia nos resultados de classificação na maioria dos conjuntos de dados utilizados (em 60

3.1.4 Dynamic Ensemble Algorithm Post-Selection Using Hardness-Aware Oracle

O algoritmo, *Hardness-aware Oracle with Dynamic Ensemble Selection* (HaO-DES), apresentado em (CORDEIRO; CAVALCANTI; CRUZ, 2023), introduz uma abordagem inovadora para aprimorar o desempenho das técnicas de seleção dinâmica de classificadores. Este aprimoramento é alcançado por meio da seleção e avaliação do conjunto mais adequado de classificadores para cada instância da base de dados, atuando como uma estratégia de pós-seleção. A avaliação dos conjuntos gerados por diferentes técnicas é realizada utilizando

uma nova métrica denominada *Hardness-aware Oracle* (HaO).

Na primeira fase (i), é empregada a amostragem por *bootstrap*, na qual são gerados múltiplos conjuntos de treinamento, em um processo análogo à abordagem de *bagging*, o que potencializa a diversidade no conjunto. Subsequentemente, os classificadores são treinados utilizando esses conjuntos de treino gerados, com a aplicação de algum classificador heterogêneo para promover o aumento da diversidade (por exemplo, Perceptron, Regressão Linear ou Naive Bayes). Finalmente, é iniciada a técnica de seleção dinâmica utilizando um conjunto de dados de validação. Esta última etapa depende de uma configuração específica, sendo crucial a adoção de diferentes critérios de seleção para alcançar a seleção de *ensembles* diversos.

Na segunda fase (ii), ocorre a seleção dinâmica, que se inicia com a definição da região de competência para cada amostra, utilizando-se métodos como k-NN ou *clustering*. Em seguida, procede-se à geração do *ensemble* por meio de uma técnica de seleção dinâmica previamente configurada. A seleção faz uso da mesma região de competência estabelecida anteriormente, e então calcula-se, para cada conjunto gerado, a métrica HaO. Esta métrica avalia a proporção de classificadores competentes selecionados por uma técnica de seleção dinâmica em relação ao número total de classificadores selecionados, sendo escolhido o *ensemble* que apresenta o maior valor da métrica HaO.

Na terceira fase (iii), procede-se à integração dos classificadores por meio da técnica de votação majoritária.

3.1.5 Meta-learning-based sample discrimination framework for improving dynamic selection of classifiers under label noise

O estudo apresentado em (XU et al., 2024) propõe um novo algoritmo que utiliza uma estrutura para a discriminação de amostras baseada em *meta-learning* (MSD), com o objetivo de melhorar o desempenho na seleção dinâmica de classificadores na presença de ruídos de rótulos. O algoritmo é constituído por três fases. A primeira fase (i) emprega uma amostra de treino em uma região local, utilizando uma combinação com o algoritmo K-vizinhos mais próximos (KNN), um conjunto de dados de validação e um conjunto de rótulos associados a essa região. Essa combinação visa determinar a aplicabilidade de técnicas de seleção dinâmica no tratamento de amostras de treinamento a partir da perspectiva do *meta-learning*.

Consequentemente, são extraídos da região local nove conjuntos distintos de características-meta, que servem como indicadores indiretos de desempenho para cada classificador base na classificação da amostra, capturando diferentes aspectos da região local definida. As seis primeiras características-meta são determinadas por meio de medidas de complexidade do problema de classificação, enquanto as três últimas fornecem informações

sobre o equilíbrio entre as classes e identificam amostras frequentemente classificadas incorretamente.

Além disso, é estabelecido um conjunto de rótulos-meta com base na distribuição das classes na região local de uma amostra. Caso a amostra contenha instâncias de mais de uma classe, o rótulo-meta será atribuído o valor 1, indicando que a seleção dinâmica não é recomendada devido à presença de ruídos nos rótulos. Por outro lado, o valor 0 para o rótulo-meta sugere que a seleção dinâmica pode ser realizada.

O conjunto de dados de *meta-learnings* consiste em pares de vetores de características-meta e rótulos-meta.

Na segunda fase (ii), realiza-se o treinamento do conjunto de *meta-learnings*, com a seleção e treinamento de um classificador base utilizando o conjunto construído na primeira fase (i), visando a obtenção de um *meta-learner*. Este processo é projetado para que o *meta-learner* selecionado seja capaz de capturar a relação entre as características das regiões locais e a aplicabilidade dos métodos de seleção dinâmica. Assim, ele poderá avaliar a adequação de uma técnica de seleção específica.

Na terceira fase (iii) é realizada a seleção dinâmica, baseada na saída do único *meta-learner* selecionado em (ii). Se ele indicar que a seleção é aplicável (rótulos-meta com valor zero), qualquer método de seleção disponível pode ser utilizado para a classificação. Caso contrário, é aplicado um processo de seleção dinâmica baseado em algoritmo genético (GA), para se mitigar o impacto negativo de ruídos no rótulo.

Este algoritmo empregou oito métodos de seleção dinâmica como base de avaliação, os quais foram divididos em duas categorias: (i) a seleção dinâmica de classificadores (DCS), com três tipos distintos de algoritmos, e (ii) a seleção dinâmica de conjuntos (DES), com cinco algoritmos diferentes.

Os resultados apresentados indicam melhorias em aproximadamente 83,3% das comparações no desempenho da seleção dinâmica em bases de dados com diferentes níveis de ruídos em rótulos, em comparação com métodos específicos de seleção dinâmica.

3.1.6 DES-A: Dynamic ensemble selection based on algorithm Shapley

O algoritmo DES-A (ZHANG; ZHU; LUO, 2024) introduz um método inovador de seleção dinâmica de classificadores, fundamentado na competência de sinergia entre os classificadores, utilizando uma variação do algoritmo de Shapley, comumente empregado na teoria dos jogos, para avaliar essa competência. Este método visa superar as limitações das métricas tradicionais de diversidade, que não consideram tal competência.

No estágio inicial, é realizada a construção de um conjunto de classificadores candidatos, que podem ser gerados de maneira homogênea ou heterogênea, com o intuito

de alcançar um determinado grau de diversidade no conjunto. Na abordagem homogênea, esse objetivo é atingido por meio da aplicação do mesmo algoritmo de classificação em diferentes conjuntos de características. Em cenários heterogêneos, são utilizados diferentes algoritmos de classificação ou diferentes parâmetros dentro do mesmo algoritmo.

Na segunda fase, ocorre o processo de seleção, que se baseia em uma seleção em grupo denominada competência de sinergia, na qual se mede a capacidade de colaboração entre os classificadores do conjunto. O algoritmo *Shapley* (SHAPLEY, 1953) é utilizado para obter essa medida, atribuindo valores de forma justa e de acordo com o desempenho cooperativo do classificador. Durante esse cálculo, a simulação de Monte Carlo truncada é empregada para reduzir a complexidade computacional e aproximar os valores. Posteriormente, os membros do conjunto que obtiverem um valor de *Shapley* positivo serão selecionados.

Na fase final, ocorre o processo de combinação para se obter a decisão final, com o peso de cada membro do conjunto calculado com base no seu valor do *Shapley* normalizado por votação majoritária ponderada.

Como resultado, o método demonstrou ser eficaz e robusto para a seleção dinâmica de conjuntos de classificadores, superando outros algoritmos de estado da arte na área, como o META-DES, OLA e *Bagging*, evidenciando a aplicabilidade da competência de sinergia medida pelo algoritmo *Shapley*.

3.1.7 Dynamic Post-Hoc Neural Ensemblers

Este trabalho, conforme apresentado em (ARANGO et al., 2024), propõe um método de *ensemble* pós-hoc dinâmico que emprega redes neurais para combinar as previsões de múltiplos classificadores base, com foco em três aspectos principais: *Ensembles* Dinâmicos, Regularização por *Dropout* e Arquitetura de Redes Neurais.

O primeiro aspecto refere-se à criação de uma rede neural denominada *Neural Ensembler*, que atribui pesos de forma dinâmica a cada instância da base de dados, em vez de utilizar pesos fixos. Essa abordagem permite uma adaptação às características específicas de cada instância.

A regularização por *Dropout* é empregada para mitigar o *overfitting* e promover a diversidade no *ensemble* durante o treinamento. Esta técnica descarta algumas previsões de determinados classificadores base do conjunto, fazendo com que o *Neural Ensembler* dependa de diferentes classificadores.

Além disso, a arquitetura de redes neurais pode operar de duas maneiras distintas. No modo *Stacking*, a rede neural recebe as previsões dos classificadores e realiza uma estimativa final da previsão. No modo de média de modelos, a rede gera pesos para cada classificador base, que são utilizados para a combinação das previsões. Este processo utiliza uma arquitetura baseada em *Deep Sets* para a geração de pesos não normalizados, que são

posteriormente normalizados por meio do *SoftMax*.

O estudo apresenta resultados que indicam a eficácia da abordagem para *ensembles* pós-hoc, demonstrando a capacidade de adaptação dinâmica às instâncias, ao mesmo tempo em que mitiga o *overfitting* por meio da técnica de *dropout*.

Considerando que o objetivo deste estudo é o desenvolvimento de técnicas de seleção dinâmica no contexto *online*, uma revisão mais abrangente sobre o tema de seleção dinâmica de classificadores no contexto *batch* pode ser encontrada nos seguintes trabalhos: (BRITTO; SABOURIN; OLIVEIRA, 2014) e (ALMEIDA et al., 2016).

3.2 Seleção dinâmica de classificadores em ambientes *online*

As pesquisas em seleção dinâmica e ambientes *online* são recentes, destacando-se principalmente pela forma como os modelos são treinados. Esses modelos são treinados de maneira incremental, ou seja, são continuamente atualizados sem a necessidade de um retreino completo do modelo desde o início.

Nas seções subsequentes, são apresentadas diversas técnicas que empregam algoritmos de seleção dinâmica de classificadores em ambientes *online*. Algumas dessas técnicas realizam a seleção em combinação com outros métodos, como a detecção de desvios de conceitos, enquanto outras utilizam a seleção dinâmica para aprimorar os resultados dos conjuntos de classificadores como um todo.

3.2.1 Dynamic Selection Based Drift Handler

O algoritmo *Dynamic Selection Based Drift Handler* (DYNSE) (ALMEIDA et al., 2016) foi desenvolvido com o objetivo de abordar problemas de mudanças de conceito por meio da seleção dinâmica de classificadores. A metodologia deste algoritmo consiste em manter o maior número possível de classificadores no *ensemble*, com cada membro sendo treinado com dados coletados em diferentes momentos e contextos, o que resulta em um comportamento dinâmico tanto na geração do conjunto de classificadores quanto durante a seleção dos mesmos.

O esquema de seleção empregado baseia-se no cálculo da vizinhança mais próxima, utilizando um número de k-vizinhos que representa a região local onde o esquema de seleção dinâmica será aplicado. No estudo, é utilizado o algoritmo *KNORA-E* (KO; SABOURIN; JR., 2008), com algumas modificações para o tratamento de ruídos nas bases de dados, como método de seleção.

Durante a fase de testes, é elaborado um conjunto de validação em *batch* contendo um número de n instâncias. Cada *batch* criado é utilizado para treinar os novos classifica-

dores. Caso o tamanho n de instâncias definido não seja suficiente, outros *batches* podem ser acumulados antes do treinamento de um novo classificador (ALMEIDA et al., 2016).

Este estudo apresentou resultados superiores, variando entre 77,8% e 90,8%, em comparação com outros métodos de seleção de estado da arte atuais, como o DDM e o EDDM (BAENA-GARCíA et al., 2006), considerando os valores médios obtidos em todos os testes realizados.

3.2.2 A Decision-Based Dynamic Ensemble Selection Method for Concept Drift

O algoritmo *Dynamic Ensemble Selection for Drift Detection* (DESDD), conforme apresentado em (ALBUQUERQUE et al., 2019), constitui um método para a detecção de desvios de conceitos (*concept drift*). Este algoritmo propõe a formação de um conjunto diversificado de múltiplos sub-conjuntos de classificadores e, por meio da aplicação de seleção dinâmica baseada na predição global, obtém um conjunto mais adequado para cada instância.

Essa predição global representa a precisão acumulada do conjunto em todas as instâncias processadas, sendo atualizada a cada nova instância classificada e utilizando a acurácia *prequential* acumulada, que permite a adaptação às mudanças de distribuição dos dados que podem ocorrer ao longo do tempo.

O estudo emprega os algoritmos *Online Bagging* e *Hoeffding Tree* em combinação. O primeiro algoritmo é responsável por diversificar o conjunto de classificadores por meio da variação do parâmetro da constante de $Poisson(\lambda)$. O segundo algoritmo, utilizado como classificador base na árvore de decisão (seção 2.1.3.2), é escolhido por sua capacidade de aprendizado incremental.

Os resultados apresentados demonstraram ser superiores em comparação com outros algoritmos utilizados como referência nos testes realizados, evidenciando uma melhoria na eficácia da detecção de desvios de conceito, com a redução de atrasos durante o processo de detecção. Ademais, comprovou-se que a utilização combinada de diversidade e seleção resulta em melhorias em relação ao *ensemble*.

3.2.3 A drift detection method based on dynamic classifier selection

O método de detecção de mudança de conceito baseado na seleção dinâmica de classificadores (PINAGé; SANTOS; GAMA, 2019) é semissupervisionado e se destina a cenários práticos nos quais mudanças de conceitos ocorrem frequentemente e onde o monitoramento da precisão da detecção resulta na diminuição do desempenho do sistema. Embora o objetivo principal da técnica seja a detecção de mudança, os conceitos empregados

são derivados das abordagens de seleção dinâmica de classificadores discutidas nas seções anteriores.

Este estudo apresenta uma abordagem que busca evitar o monitoramento da precisão por meio das previsões de um conjunto de classificadores selecionados de forma dinâmica, atribuindo rótulos a cada instância do fluxo de dados recebido. Com base nesses rótulos, um detector de mudanças é aplicado para identificar a mudança de conceito.

O método é estruturado em três componentes principais: a geração do conjunto, a seleção dinâmica do classificador e a detecção de mudanças. A seleção dinâmica é implementada para identificar o classificador mais competente dentro do *ensemble*, considerando que cada membro do conjunto possui especialização em uma região específica de competência. Para tal, são empregadas duas estratégias:

- A DCS-LA (*Dynamic Classifier Selection with Local Accuracy*) realiza a avaliação da precisão local de cada classificador, considerando os vizinhos mais próximos. O classificador com maior precisão é selecionado para a agregação do voto final.
- A DS-MCB (*Dynamic Classifier Selection based on Multiple Classifier Behavior*) realiza a análise do comportamento de múltiplos classificadores, medindo a similaridade entre os rótulos atribuídos por cada classificador base aos seus vizinhos mais próximos. Um novo conjunto é gerado com base nessa similaridade, e o classificador mais preciso nesse conjunto é selecionado para a agregação do voto final.

O estudo revelou resultados promissores em termos de precisão geral de classificação e na detecção de mudanças de conceito, quando comparado a outras abordagens supervisionadas e semissupervisionadas.

3.2.4 Dynamically Selected Ensemble for Data Stream Classification

Em (CAVALHEIRO et al., 2021), os autores apresentam um método inovador denominado *Double Dynamic Classifier Selection* (DDCS), desenvolvido para abordar a seleção dinâmica de classificadores com ênfase na eficiência temporal e de memória, constituindo uma alternativa eficaz para lidar com problemas de *concept drift*.

O algoritmo processa os dados de maneira semelhante ao algoritmo DYNSE (3.2.1), utilizando *mini-batches* ou *chunks* sequenciais, onde cada *chunk* é empregado para treinar um novo classificador base, que é subsequentemente adicionado ao conjunto.

Durante o processo de predição, para cada instância do fluxo de dados, é construída uma árvore utilizando o algoritmo KD-Tree, com o objetivo de identificar o vizinho mais próximo dentro de um espaço de características. A partir dessa etapa, aplica-se um algoritmo de seleção dinâmica de classificadores para determinar os mais competentes,

utilizando-se os algoritmos KNORA-E, KNORA-U ou optando-se por não empregar nenhum método de seleção. Nesta última opção, realiza-se a votação majoritária para a combinação direta das predições do *ensemble*.

No que tange à otimização, a implementação do algoritmo KD-Tree resultou em uma redução significativa da sobrecarga computacional, demonstrando grande flexibilidade. Este algoritmo pode ser empregado com diferentes métodos de seleção dinâmica de classificadores, com a possibilidade de execução com ou sem o *online bagging*.

Em termos de resultados, a abordagem é apresentada como uma alternativa eficaz para a seleção dinâmica de classificadores em fluxos de dados, demonstrando um desempenho competitivo em termos de acurácia e resultados superiores no que diz respeito ao tempo de processamento e ao consumo de memória.

3.2.5 Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams

O trabalho apresentado em (ZYBLEWSKI; SABOURIN; WOźNIAK, 2021) concentra-se no problema de dados desbalanceados, sendo semelhante ao algoritmo DYNSE (Seção 3.2.1). Neste estudo, propõe-se a criação de um *framework* que combina o pré-processamento de dados com a seleção dinâmica de classificadores, visando aprimorar a classificação em conjuntos de dados altamente desbalanceados. A abordagem empregada envolve o treinamento de múltiplos classificadores, dos quais um subconjunto é selecionado dinamicamente durante a fase de testes, com base em sua competência local.

O fluxo de dados é processado como uma série de blocos de tamanho fixo. Para cada bloco, é realizado o treinamento de um classificador de *bagging* estratificado, que é então incorporado a um conjunto de classificadores. Esta técnica é considerada uma abordagem para lidar com o desbalanceamento entre classes, sendo tratada por meio de uma amostragem com substituição separada das classes minoritárias e majoritárias, assegurando assim a manutenção da proporção original de classes.

A seleção dinâmica proposta pode ocorrer em dois níveis: (i) no nível dos classificadores *bagging*, com até 5 membros no conjunto, e (ii) no nível que abrange todos os classificadores do conjunto, limitado a um tamanho de 50 membros.

O estudo proposto apresentou resultados que superaram os métodos de última geração em termos de desempenho de classificação, especialmente em cenários caracterizados por um alto grau de desbalanceamento nos dados.

3.2.6 Dynamic ensemble selection classification algorithm based on window over imbalanced drift data stream

Os autores em (HAN et al., 2023) apresentam o algoritmo *Dynamic ensemble selection based on window over imbalanced drift data stream* (DESW-ID), que, similar ao algoritmo discutido na Seção 3.2.5, também aborda o problema de dados desbalanceados.

O estudo propõe uma combinação de técnicas, introduzindo um novo método de amostragem baseado em janela, utilizando a função de *Poisson* (λ) para tratar o desbalanceamento entre as classes.

Dois tipos de janelas deslizantes são empregados. A primeira armazena as instâncias do fluxo de dados à medida que chegam, representando, assim, o conceito mais recente dos dados. Esta janela possui um tamanho dinâmico, aumentando na ausência de desvios de conceito. O segundo tipo de janela é responsável por armazenar as instâncias de classes minoritárias, assegurando um número suficiente de exemplos representativos das classes minoritárias para o treinamento do classificador, mesmo na ausência de desvios de conceitos.

A seleção dinâmica dos classificadores é implementada por meio de quatro etapas:

- Os membros da lista de classificadores são ordenados em ordem crescente de erros durante a fase de treinamento.
- Realiza-se uma busca inversa, começando pelo classificador com o pior desempenho. Durante essa busca, é efetuada uma comparação entre dois classificadores consecutivos.
- A determinação do tamanho do conjunto é realizada por meio da verificação da diferença entre esses dois classificadores; se essa diferença for menor que um determinado limiar, o algoritmo seleciona o classificador com melhor desempenho.
- Caso a diferença de erro não seja menor que o limiar estabelecido, seleciona-se metade dos classificadores da lista ordenada.

O algoritmo demonstrou eficácia em várias métricas, bem como adaptabilidade a diferentes mudanças de conceito, além de eficiência em termos de tempo de execução.

3.2.7 Adaptive regularized ensemble for evolving data stream classification

O algoritmo *Adaptive Regularized Ensemble* (ARE) para classificação em fluxos de dados é apresentado em (PAIN; ENEMBRECK, 2024). Este algoritmo busca alcançar alta precisão preditiva com baixo custo computacional, empregando árvores de decisão treinadas em subespaços de atributos de tamanhos aleatórios.

Para atingir esse objetivo, o algoritmo utiliza subespaços de atributos de tamanhos aleatórios para cada classificador base, promovendo assim um aumento na diversidade entre os classificadores. Posteriormente, realiza-se uma regularização do treinamento por meio da seleção de instâncias do fluxo de dados, utilizando apenas as instâncias classificadas incorretamente, o que otimiza o tempo de processamento.

A seleção dinâmica de classificadores é implementada por meio de uma janela deslizante que monitora as últimas N instâncias do fluxo de dados. Este mecanismo avalia o nível de competência de cada classificador, selecionando apenas aqueles considerados mais competentes, ou seja, aqueles cujo desempenho supera a precisão média dentro da janela.

Os resultados apresentados evidenciam o impacto do limite de rejeição, com o controle da frequência em que as instâncias corretamente classificadas não são utilizadas no treinamento. Este comportamento demonstrou aprimorar a precisão geral do *ensemble*. Em relação aos resultados competitivos com outros algoritmos, observou-se melhorias significativas em comparação com alguns dos algoritmos de última geração, destacando-se também no desempenho computacional.

3.2.8 Mass-Based Short Term Selection of Classifiers in Data Streams

O estudo apresentado em (ASSIS; ENEMBRECK; BARDDAL, 2023) introduz o método *Mass-based Short Term Selection* (MSTS). Este método propõe a utilização de uma janela deslizante para monitorar o desempenho recente dos classificadores e selecionar aqueles com maior probabilidade de acerto. As melhorias sugeridas foram implementadas nos algoritmos *Adaptive Random Forest* (ARF) (GOMES et al., 2017), Streaming Random Patches (SRP) (GOMES; READ; BIFET, 2019) e Bossting-like Online Learning (BOLE) (BARROS; SANTOS; JúNIOR, 2016).

A seleção proposta tem como objetivo aprimorar a precisão dos modelos de conjunto (*ensembles*) por meio da seleção dos classificadores mais adequados, conforme sua competência para cada instância do fluxo de dados individualmente, em vez de considerar o conjunto como um todo. Para isso, a seleção é realizada com base na observação do desempenho de curto prazo, utilizando uma janela deslizante configurada para monitorar previsões corretas nas últimas N instâncias do fluxo de dados.

São utilizadas três estratégias com variações de limiares para a seleção:

- **Limiar fixo**, onde um classificador é selecionado apenas se um número de previsões corretas ocorrer dentro de um limiar fixo na janela deslizante.
- **Limiar fixo e Média**, uma especialização do primeiro, onde se considera a previsão média correta entre todos os classificadores nas instâncias mais recentes, com um

classificador sendo selecionado somente se suas previsões corretas excederem o limiar fixo e a média.

- **Limiar fixo e Moda**, semelhante ao anterior, mas considerando a moda das previsões corretas entre os classificadores nas instâncias mais recentes, com o classificador sendo selecionado se suas previsões corretas excederem o limiar fixo e a moda.

Além disso, é empregada uma Função de Massa de Previsão Correta (MFRP) no processo, a qual é responsável por contabilizar o número de classificadores básicos que apresentam um determinado número de previsões corretas dentro da janela deslizante.

Como os resultados apresentados, é possível destacar melhorias com as alterações propostas nos algoritmos ARF e BOLE, superando o algoritmo SRP em sua versão original quanto às métricas de custo de processamento e uso de memória, caracterizando o MSTS como o estado da arte para seleção dinâmica de classificadores para fluxos de dados de propósito geral.

3.2.9 Adaptive random tree ensemble for evolving data stream classification

Em (PAIM; ENEMBRECK, 2025), os autores introduzem o algoritmo *Adaptive Random Tree Ensemble* (ARTE) para a classificação de fluxo de dados, explorando diversas abordagens para alcançar alta precisão preditiva e eficiência computacional.

As principais abordagens empregadas pelo ARTE são:

- A utilização de um subespaço de características de tamanho aleatório para cada membro do conjunto de classificadores visa mitigar o problema de selecionar um subconjunto de atributos que seja limitado ou excessivo.

- A aplicação do *bagging online*, similar ao algoritmo *Adaptive Random Forest* (ARF) com $\lambda = 6$, busca aumentar a probabilidade de utilizar um maior número de instâncias para o treinamento, promovendo assim uma maior diversidade no conjunto de classificadores.

- A determinação de um ponto de corte para a divisão dos nós das árvores tem como objetivo reduzir o tempo de processamento sem comprometer a precisão do modelo, além de aumentar a diversidade nas árvores.

- A seleção de classificadores para a votação final do conjunto é realizada de modo que apenas aqueles com precisão individual superior à média do conjunto em uma janela deslizante de curto prazo sejam escolhidos, eliminando assim os classificadores fracos. Esta janela é configurada com o tamanho de 500.

- A possibilidade de utilização de diversos algoritmos de detecção de drift é considerada, sendo o algoritmo padrão o ADWIN. Cada classificador base possui um detector independente que, ao identificar uma mudança, substitui a árvore existente por uma nova.

A determinação da previsão final é realizada por meio da combinação dos votos através da votação majoritária, uma vez que apenas os classificadores de melhor desempenho dentro da janela deslizante foram selecionados.

3.3 Considerações Finais

Esse capítulo apresentou uma revisão da literatura atual sobre seleção dinâmica de classificadores em ambientes em *batch* e *online*. Em alguns desses trabalhos, a seleção dinâmica foi utilizada em conjunto com outros métodos para a detecção de desvios de conceitos e aprimoramento da performance geral do ensemble, não sendo utilizada somente como método para seleção dos membros mais competentes de um conjunto.

Isso demonstra a vasta gama de aplicações que a seleção dinâmica de classificadores pode ter, indicando também que ainda existe um grande espaço para melhorias nos algoritmos existentes e para a criação de novos algoritmos voltados para a resolução dos diversos desafios e adaptações aos ambientes voláteis atuais.

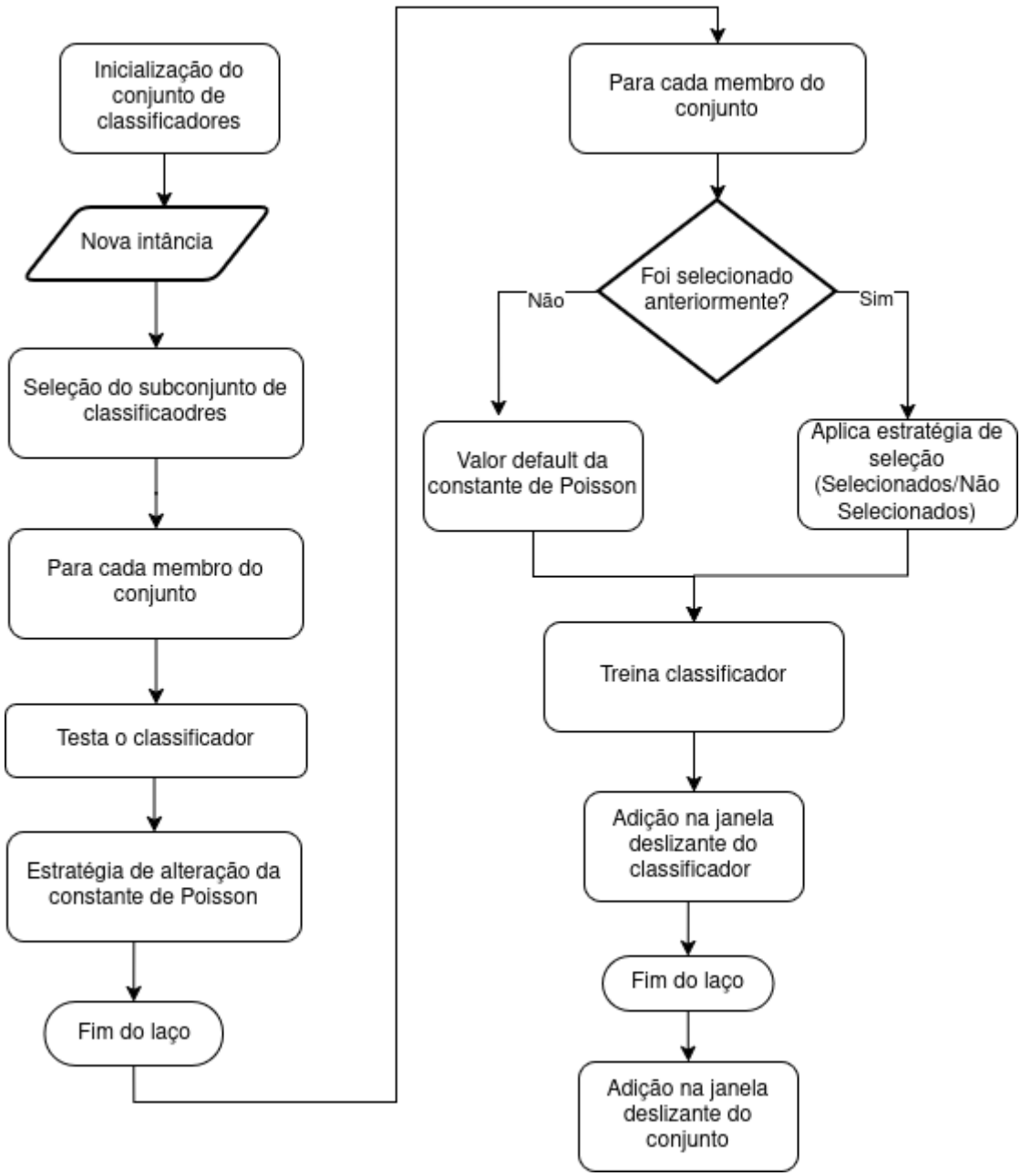
Os próximos capítulos apresentam um método de seleção dinâmica de classificadores adaptado para o algoritmo *Adaptive Random Forest*, sendo a investigação de aplicabilidade do método sobre outros *ensembles* e etapas futuras não contempladas pela presente pesquisa. As técnicas propostas foram inspiradas em vários dos algoritmos apresentados no presente capítulo e foram pensadas para se aplicar a diferentes problemas de propósito geral, e não em contextos específicos como fluxos fortemente desbalanceados ou ruidosos.

4 Método proposto

Na literatura atual, diversos métodos visam aprimorar o processo de seleção dinâmica de classificadores, alcançando frequentemente bons resultados em termos de taxa de acerto (acurácia). No entanto, esses métodos frequentemente enfrentam desafios relacionados ao consumo de recursos computacionais. O equilíbrio entre essas métricas é crucial em ambientes de fluxos de dados *online*. O método proposto neste estudo busca equilibrar a precisão e o uso de recursos computacionais. Propomos uma nova técnica que explora a seleção por meio da *Dominância Local-Global* (LGDS), onde os classificadores base (*base learners*) selecionados são aqueles que demonstram desempenho individual de curto prazo superior ao desempenho médio global do *ensemble*.

A Figura 3 apresenta o fluxograma do algoritmo com a aplicação da abordagem proposta.

Figura 3 – Fluxograma do algoritmo da fase de treino com a abordagem proposta



Fonte: Autoria Própria

Com a aplicação da seleção dinâmica, observa-se a variação da constante de *Poisson* no *online bagging*, conforme ilustrado na Figura 3. Este processo envolve a escolha de valores específicos de λ para os classificadores selecionados e não selecionados. Assim, busca-se reduzir a intensidade de treinamento em subconjuntos de classificadores, mitigando o possível *overhead* de processamento introduzido pelo processo de seleção dinâmica.

4.1 Conceitos principais do método proposto

O método proposto fundamenta-se em conceitos amplamente utilizados na literatura contemporânea e, de modo geral, apoia-se em três conceitos principais, os quais são detalhados nas seções subsequentes.

4.1.1 Janela Deslizante de Curto Prazo

Durante o processo de predição, à medida que as instâncias do fluxo de dados são recebidas, podem ocorrer desvios de conceito que resultam na degradação do modelo em treinamento. Para mitigar esse fenômeno, algoritmos de aprendizagem em tempo real podem empregar técnicas de retenção temporária de dados, permitindo a adaptação aos desvios de conceitos. Essa capacidade é também referida como o dilema da estabilidade-plasticidade, onde a plasticidade se refere à habilidade de aprender novos conceitos, enquanto a estabilidade diz respeito à retenção do conhecimento previamente adquirido (GAMA et al., 2014; GOMES et al., 2017).

A utilização de janelas deslizantes de curto prazo possibilita o equilíbrio entre estabilidade e plasticidade, ao mesmo tempo que permite a adaptação a novos desvios de conceitos. Essas janelas podem ter um tamanho fixo n, determinado por um parâmetro configurável. Uma vez que a janela é preenchida, os dados mais antigos são removidos, podendo seguir uma regra pré-definida.

Neste estudo, a técnica é empregada para a extração de estatísticas de desempenho relevantes em dois processos: (i) monitoramento do desempenho do *ensemble*, conforme ilustrado no Algoritmo 5 (linha 25), e (ii) monitoramento do desempenho do *base learner*, como demonstrado no Algoritmo 6 (linha 25). Em ambos os casos, a métrica de validação de desempenho utilizada foi a *Kappa Statistic* (COHEN, 1960) (fórmula 2.2), que mede o nível de concordância entre a decisão esperada de um classificador e os dados observados. A escolha dessa métrica é justificada por seus resultados serem menos afetados em cenários com dados desbalanceados, promovendo previsões corretas de instâncias de classes minoritárias.

Nos experimentos, foram utilizados valores de configuração para as janelas de 10, 50, 100 e 200. Esses valores foram selecionados para analisar dois aspectos do método: (i) o uso de recursos computacionais, como tempo e memória, uma vez que a responsividade a mudanças e o desempenho podem ser alcançados com janelas menores; e (ii) a estabilidade dos indicadores de desempenho para a métrica de seleção dos classificadores, pois janelas maiores tendem a gerar resultados mais confiáveis devido à maior quantidade de informações disponíveis. A variação do tamanho das janelas tem sido estudada na literatura, como em (GAMA et al., 2014), onde os autores discutem que o uso de janelas curtas pode prejudicar o desempenho dos algoritmos durante períodos estáveis, sem mudanças significativas, mas

também pode assegurar uma adaptação mais rápida durante a ocorrência de mudanças de conceito. Por outro lado, janelas longas apresentam melhor desempenho em períodos estáveis, mas reagem lentamente às mudanças de conceito.

4.1.2 Seleção Dinâmica de Subconjuntos de Classificadores

Conforme discutido anteriormente, uma das funções desempenhadas pelos algoritmos de seleção dinâmica é a combinação dos votos do subconjunto selecionado no *ensemble*, modificando a maneira como a instância do fluxo de dados é processada, dado que pode haver variação no número de membros utilizados para a combinação dos votos.

Neste estudo, o processo é realizado por meio da comparação entre o desempenho preditivo individual do classificador base (local) e o desempenho médio geral do *ensemble* (global) nas últimas n instâncias processadas, conforme descrito no Algoritmo 7 (linha 6). Essas características foram resumidas no termo *Dominância Local-Global*, uma vez que os classificadores selecionados são aqueles cujo desempenho local é equivalente ou superior ao do *ensemble*, utilizando a estatística Kappa como métrica de desempenho, conforme discutido na Seção 4.1.1. Essa escolha é justificada pela natureza dinâmica dos fluxos de dados na busca pela redução de erros de classificação.

A seleção é dada pela equação 4.1.

$$C^* = \{C_i \mid Kappa(C_i) >= Kappa(C), \forall C_i \in C\}\} \tag{4.1}$$

onde C_i é o classificador base, C o *ensemble* e C^* é o subconjunto dos classificadores base que possuem o valor da estatística *Kappa* maior ou igual ao valor do *ensemble*.

O cálculo da estatística *Kappa* é realizado por meio da aplicação da Equação 2.2, sendo sua obtenção efetuada através das duas janelas deslizantes incorporadas nos algoritmos do classificador base (*base learner*) e outra no conjunto de classificadores (*ensemble*).

4.1.3 Amostragem *online* variável

O processo de monitoramento e seleção dos classificadores, conforme discutido nas seções anteriores, pode introduzir processamento adicional e comprometer o desempenho computacional. Para mitigar esse problema, foi implementada a aplicação da mudança da média de *Poisson* utilizada pelo *Online Bagging* para simular o processo de amostragem. Considerando que as estratégias de seleção dinâmica de classificadores são capazes de determinar quais classificadores são mais adequados para classificar uma instância específica, optou-se por desenvolver estratégias que visam a segmentação do *ensemble* entre os modelos selecionados e não selecionados, conforme ilustrado no algoritmo 5 (linhas 5-9), aplicando

valores diferentes da média de *Poisson* a esses subconjuntos. Isso pode resultar em ganhos em termos de recursos computacionais, uma vez que menos treinamento pode ser necessário para uma instância cujos classificadores selecionados acertam sua classificação, e em ganhos de desempenho, dado que o treinamento excessivo de classificadores com uma instância pode ser prejudicial a longo prazo.

Esta abordagem visa analisar o comportamento geral tanto em termos de desempenho dos recursos computacionais quanto em relação à precisão, considerando a redução ou o aumento da intensidade do treino nos conjuntos selecionados e não selecionados.

Os valores atribuídos a λ são detalhados na seção 5.1.2. É importante destacar que, na implementação original do ARF, a constante de *Poisson* é fixada em 6 ($\lambda = 6$), o que eleva a probabilidade de atribuir pesos mais elevados às instâncias durante o treinamento dos classificadores base, resultando em um aumento no tamanho das árvores.

Com a aplicação desta abordagem de variação, a estratégia de seleção torna-se ativa, modificando o treinamento do *ensemble* como um todo, variando conforme a proporção de classificadores selecionados e não selecionados. Até o momento, não se encontrou na literatura qualquer estudo que empregue a estratégia de seleção de forma ativa para modificar o treinamento de *ensembles*.

Esses pontos serão abordados posteriormente na apresentação dos resultados deste estudo no Capítulo 5.

4.2 Detalhamento do método proposto

As abordagens de dominância *local-global* e adaptação do *Poisson* são aplicáveis a qualquer *ensemble* baseado em *Online Bagging*. No entanto, devido à necessidade de customizações nos algoritmos de *ensembles* para sua implementação, optou-se por modificar o algoritmo ARF (*Adaptive Random Forest*), nomeando o como LGD-ARF[1]. O Algoritmo 5 apresenta a função de treino do conjunto, seguindo o fluxo original do ARF (GOMES et al., 2017), com a diferença da inclusão de uma janela de curto prazo na linha 25. Este mesmo processo é replicado no *ensemble* com a adição de outra janela de curto prazo (Algoritmo 6, linha 25), onde os valores são adicionados para cada instância do fluxo de dados.

Após este processo, realiza-se a seleção *local-global* por meio da comparação dos valores individuais das métricas das janelas de cada um dos classificadores base (*local*) com a janela geral do *ensemble* (*global*), conforme descrito no Algoritmo 7 (linhas 6-8).

Durante o treinamento do modelo, é implementada uma estratégia de modificação

[1] <https://github.com/fernandopsan/moa_lgd/blob/main/moa/src/main/java/moa/classifiers/meta/LGDARF.java>

do parâmetro λ de *Poisson*, conforme descrito no Algoritmo 5 (linhas 5-9). A estratégia de amostragem variável aplica diferentes valores de λ para os subconjuntos selecionados e não selecionados, mantendo o valor *default* de λ no caso de não haver classificadores selecionados, como indicado na linha 11 do Algoritmo 5. Os valores de λ são especificados na Tabela 3. Assim, a estratégia de seleção evolui de uma abordagem passiva de fusão de decisão para uma abordagem ativa, que altera a forma como o *ensemble* se desenvolve ao longo do treinamento.

Algorithm 5 LGDRFTreeTrain with local global selection proposed changes - **Symbols:** λ: Fixed parameter to Poisson distribution; λs: Poisson distribution value for selected subsets; λn: Poisson distribution value for not selected subsets; GP: Grace period before recalculating heuristics for split test; w: The sliding windows size for the tree; $P(\cdot)$: Learning performance estimation function; sw_t: Sliding window evaluator for trees; $W(t)$: Tree weighted vote; E: Selected trees

1: **function** LGDRFTreeTrain(m, t, x, y, w, E, W)
2: $sw_t \leftarrow WindowPerformanceEvaluator(w)$
3: $\lambda s, \lambda n \leftarrow LambdaValues()$ ▷ Initialize λ values
4: **if** $|E| > 0$ **then**
5: **if** $TreeExistsInSelectedTreeList(t, E)$ **then**
6: $k \leftarrow Poisson(\lambda = \lambda s)$ ▷ Trees selected
7: **else**
8: $k \leftarrow Poisson(\lambda = \lambda n)$ ▷ Trees not selected
9: **end if**
10: **else**
11: $k \leftarrow Poisson(\lambda)$ ▷ Original value of the $\lambda = 6$
12: **end if**
13: **if** $k > 0$ **then** ▷ Original implementation
14: $l \leftarrow$ FindLeaf(t, x)
15: UpdateLeafCounts(l, x, k)
16: **if** InstancesSeen(l) $\geq$ GP **then**
17: AttemptSplit(l)
18: **if** DidSplit(l) **then**
19: CreateChildren(l, m)
20: **end if**
21: **end if**
22: **end if**
23: $\hat{y} \leftarrow predict(t, x)$
24: $W(t) \leftarrow P(\hat{y}, y)$ ▷ Weighted vote
25: $sw_t.Add(x, W(t))$ ▷ Add value to sliding window evaluator
26: **end function**

Algorithm 6 Local-Global Dominance Adaptive Random Forests - **Symbols:** m: maximum features evaluated per split; n: total number of trees ($n = |T|$); δw: warning threshold; δd : drift threshold; w: sliding windows size; $C(\cdot)$: change detection method; S: Data stream; B: Set of background trees; $W(t)$: Tree weight vote; $P(\cdot)$: Learning performance estimation function; sw_e: Sliding window evaluator for the ensemble

function LGDARF(m, n, δw, δd, w)
  $T \leftarrow CreatesTree(n)$
  $W \leftarrow InitWeights(n)$
  $B \leftarrow \emptyset$
  $sw_e \leftarrow WindowPerformanceEvaluator(w)$
  while $HasNext(S)$ **do**
    $(x, y) \leftarrow next(S)$
    $E \leftarrow GetSelectedTrees(T)$
    for all $t \in T$ **do**
      $\hat{y} \leftarrow predict(t, x)$
      $W(t) \leftarrow P(W(t), \hat{y}, y)$ ▷ Weighted votes
      $LGDSRFTreeTrain(m, t, x, y, w, E, W)$ ▷ Train t on the current instance
      if $C(\delta w, t, x, y)$ **then** ▷ Warning detect?
        $b \leftarrow CreateTree()$ ▷ Init background tree
        $B(t) \leftarrow b$
      end if
      if $C(\delta d, t, x, y)$ **then** ▷ Drift detect?
        $t \leftarrow B(t)$ ▷ Replace t by its background tree
      end if
    end for
    for all $b \in B$ **do**
      $LGDRFTreeTrain(m, b, x, y, w, E, W)$ ▷ Train each background tree
    end for
  end while
  $sw_e.Add(x, W(t))$ ▷ Add value to sliding window evaluator
end function

Algorithm 7 GetSelectedTrees - **Symbols:** T: Ensemble of Trees; E: List of the selected trees; e_v: Ensemble metric vote based on the window evaluator; t_v: Tree metric vote based on the window evaluator

```
1: function GetSelectedTrees(T)
2:     E ← ∅                                   ▷ Initialize the list of the selected base learners
3:     e_v ← GetEnsembleWindowMetricVote()      ▷ Ensemble weighted average vote(Kappa Statistic)
4:     for all t ϵ T do
5:         t_v ← GetTreeWindowMetricVote()      ▷ The tree average vote(Kappa Statistic)
6:         if t_v >= e_v then                   ▷ Is tree performance greater than ensemble performance
7:             E ← E ∪ {t_v}                    ▷ Fill the list with the selected trees
8:         end if
9:     end for
       return E
10: end function
```

A agregação dos votos por meio da média ponderada do conjunto para cada instância é expressa pela Equação 4.2 (conforme o Algoritmo 7, linha 3 para o *ensemble* e linha 5 para o *base learner*).

$$\hat{y}_t = \arg\max_{y_k \in \mathcal{Y}} \sum_{i=1}^{M} w_i(t) \cdot P_i(y_k \mid x_t), \tag{4.2}$$

Onde:

$\hat{y}_t$ Classe predita pelo conjunto de classificadores no instante t.

x_t Instância de entrada observada no tempo t.

$\mathcal{Y}$ Conjunto de classes possíveis.

y_k Classe candidata utilizada no operador arg max.

M Número de árvores no conjunto de classificadores.

h_i i-ésima árvore do conjunto de classificadores.

$P_i(y_k \mid x_t)$ Probabilidade estimada por h_i para a classe y_k.

$w_i(t)$ Peso dinâmico associado à árvore h_i.

$\epsilon_i(t)$ Estimativa do erro recente da árvore h_i.

α Fator de esquecimento exponencial.

δ Constante positiva para evitar divisão por zero.

As técnicas empregadas no método proposto têm como objetivo selecionar apenas os classificadores adequados para a classificação das instâncias, ao mesmo tempo em que reduzem o treinamento em situações desnecessárias. A seleção é utilizada como um meio de segregação do conjunto, indicando que os selecionados possuem maior aptidão para classificar a instância, enquanto os não selecionados apresentam menor aptidão. Contudo, não está claro como essa segregação deve influenciar o processo de treinamento. Por exemplo, ao treinar mais frequentemente os classificadores selecionados, é possível desenvolver classificadores altamente especializados, entretanto, a performance dos não selecionados pode diminuir devido à menor frequência de treinamento. Por outro lado, ao treinar mais frequentemente os classificadores não selecionados, pode ocorrer uma uniformização das decisões, reduzindo a diversidade do *ensemble*. O próximo capítulo apresenta resultados que contribuem para a análise dessas hipóteses, incluindo também o impacto delas no desempenho computacional.

O código fonte da abordagem desse trabalho está disponível no repositório [1].

4.3 Considerações Finais

Este capítulo apresentou o método proposto de seleção *local-global* de classificadores e a aplicação da abordagem de alteração da constante de *Poisson*, com a variação da quantidade de treino do modelo de acordo com a seleção realizada. As técnicas foram implementadas como uma extensão do algoritmo ARF.

O próximo capítulo investiga o comportamento do método proposto em diferentes conjuntos de dados utilizados nos experimentos, com o objetivo de demonstrar, por meio de uma ampla variedade de problemas, o comportamento das alterações propostas no ARF, bem como aprofundar o entendimento das estratégias de seleção ativa discutidas neste capítulo.

[1] <https://github.com/fernandopsan/moa_lgd/blob/main/moa/src/main/java/moa/classifiers/meta/LGDARF.java>

5 Resultados

Como discutido em seções anteriores (Seções 1.1 e 1.2), os experimentos visam aprimorar a performance e o desempenho dos conjuntos de classificadores gerados pelo ARF. Assim, neste capítulo, serão apresentados os resultados obtidos com a utilização da abordagem proposta. Para tanto, são apresentados resultados separados para *datasets* gerados a partir de problemas reais e *datasets* construídos a partir de geradores sintéticos.

5.1 Protocolo Experimental

Para a avaliação do método proposto neste projeto, foi empregado o software *Massive Online Analysis* (MOA) (BIFET et al., 2010), na versão 2023.04. A configuração computacional utilizada inclui uma CPU Intel Xeon Gold 6238 de 2,10 GHz (8 núcleos), acompanhada por 32 GB de memória. O MOA possibilita a análise de diversos algoritmos voltados para fluxos de dados e é implementado na linguagem de programação Java, incorporando métodos para a execução de tarefas de classificação, entre outras.

A seção seguinte apresenta os detalhes referentes às bases de dados utilizadas nas avaliações, abrangendo tanto aquelas derivadas de problemas reais quanto as bases de dados sintéticas, geradas com o auxílio de geradores.

Os experimentos foram divididos em duas fases. Na primeira fase, foram realizados experimentos com as 32 configurações possíveis utilizando a abordagem proposta, que inclui o uso de janelas deslizantes descritas na Seção 4.1.1 e a aplicação da Amostragem *online* variável, conforme detalhado na Seção 4.1.3. Além disso, foram aplicados os testes de Friedman e Nemenyi em uma fase de *tunning* com alguns conjuntos de dados selecionados arbitrariamente (*Airlines*, *Nomao*, *Rialto*, *HYPER* e *SINE*), com o objetivo de identificar configurações genéricas que possam apresentar bom desempenho em diferentes problemas e garantir uma comparação justa com os demais algoritmos, que não foram ajustados e são explorados em configurações *default* nas seções subsequentes. Nesta fase, foram selecionadas duas configurações que obtiveram as melhores posições no ranking médio entre as métricas de acurácia e estatística kappa.

Os resultados são apresentados utilizando os testes de Friedman e de Nemenyi, um teste *post-hoc* amplamente empregado na literatura contemporânea, com o objetivo de verificar se o desempenho dos classificadores difere estatisticamente (DEMSAR, 2006). Na apresentação, são utilizados gráficos de distância crítica (*critical distance* - CD) que ilustram visualmente a diferença entre os classificadores.

Na segunda fase, foram conduzidos experimentos com as duas melhores configu-

rações selecionadas, utilizando o algoritmo ARF em sua configuração original e outro algoritmo de estado da arte, o *Mass-Based Short Term Selection* (MSTS) (Seção 3.2.8), empregando uma das variações apresentadas no estudo, o MSTS-ARF. Outros algoritmos mencionados na revisão da literatura na Seção 3.2 não foram utilizados devido a fatores como o uso de métricas de distância e do KNN (K-Visinhos mais próximos), que afetam seu desempenho em fluxos de dados, conforme discutido em (BRITTO; SABOURIN; OLIVEIRA, 2014), como é o caso dos algoritmos DYNSE, DCS-LA, DDCS e *Preprocessed dynamic classifier ensemble selection*. Além disso, outros algoritmos foram excluídos por integrarem diferentes estratégias de geração, atualização e fusão, sendo a seleção apenas uma dessas estratégias, como ocorre com os algoritmos ARE e ARTE.

Destaca-se que, em ambas as fases mencionadas anteriormente, foi realizada a modificação do componente de aleatoriedade do ARF, do MSTS-ARF e das melhores configurações, com a execução de 10 repetições utilizando diferentes valores de *seeds*, sendo apresentados os valores médios e o desvio padrão.

O modelo de validação *Prequential* (GAMA; SEBASTIAO; RODRIGUES, 2013) foi empregado durante os experimentos, também conhecido como teste e treino, conforme discutido na Seção 2.1.2. A seguir, são apresentadas tabelas com os valores médios individuais por conjunto de dados e os resultados das métricas de acurácia, estatística kappa, *CPU-Time* e *Model Cost*. Posteriormente, são apresentados os testes de Friedman e Nemenyi com os rankings médios da acurácia e da estatística kappa, tanto para identificar as melhores configurações para a abordagem aqui proposta quanto para a comparação com os algoritmos de estado da arte.

5.1.1 Datasets

Para a avaliação do método proposto, foram selecionados 34 *datasets*, dos quais 17 contêm dados reais e 17 foram gerados sinteticamente. Os conjuntos sintéticos foram criados com 500.000 instâncias, apresentando uma mudança de conceito (abrupta e gradual) a cada 125.000 instâncias, utilizando o software MOA.

As Tabelas 1 e 2 apresentam as propriedades dos *datasets* utilizados, incluindo o número total de instâncias (#Inst), número de atributos numéricos (#Num), número de atributos nominais (#Nom), número de classes (#Classes) e o tipo de mudança de conceito (Drift). Adicionalmente, descrições detalhadas de cada dataset estão incluídas no Apêndice A.

Tabela 1 – Propriedades dos *datasets* sintéticos

Dataset	#Inst	#Num	#Nom	#Classes	Drift
LED-A (BREIMAN et al., 1984)	500000	0	10	10	Abrupta
LED-G (BREIMAN et al., 1984)	500000	0	10	10	Gradual
Agrawal-G (AGRAWAL; IMIELINSKI; SWANI, 1993)	500000	6	3	2	Gradual
Agrawal-A (AGRAWAL; IMIELINSKI; SWANI, 1993)	500000	6	3	2	Abrupta
RTG-A (DOMINGOS; HULTEN, 2000)	500000	5	5	2	Abrupta
RTG-G (DOMINGOS; HULTEN, 2000)	500000	5	5	2	Gradual
RTG (DOMINGOS; HULTEN, 2000)	500000	5	5	2	Nenhum
RBF-F (HULTEN; SPENCER; DOMINGOS, 2001)	500000	10	0	5	Abrupta
RBF-M (HULTEN; SPENCER; DOMINGOS, 2001)	500000	10	0	5	Gradual
HYPER (HULTEN; SPENCER; DOMINGOS, 2001)	500000	10	0	2	Gradual
SEA-A (STREET; KIM, 2001)	500000	3	0	2	Abrupta
SEA-G (STREET; KIM, 2001)	500000	3	0	2	Gradual
Mixed-Balanced (GAMA; MEDAS; RODRIGUES, 2004)	500000	2	2	2	Abrupta
Mixed-Imbalanced (GAMA; MEDAS; RODRIGUES, 2004)	500000	2	2	2	Abrupta
SINE (BAENA-GARCíA et al., 2006)	500000	4	0	2	Nenhum
WAVEFORM-G (ASUNCIóN; NEWMAN, 2007)	500000	40	0	2	Gradual
WAVEFORM (ASUNCIóN; NEWMAN, 2007)	500000	40	4	2	Nenhum

Tabela 2 – Propriedades dos *datasets* reais

Dataset	#Inst	#Num	#Nom	#Classes	Drift
Connect-4 (TROMP, 1995)	67557	0	42	2	Desconhecido
Forest Cover Type (BLACKARD; DEAN, 1999)	581012	10	44	7	Desconhecido
Electricity (HARRIES; WALES, 1999)	45312	7	1	2	Desconhecido
KDD (STOLFO et al., 2000)	4898431	34	6	2	Desconhecido
KDD99CUP (STOLFO et al., 2000)	494021	34	6	2	Desconhecido
Pokerhand (CATTRAL; OPPACHER, 2002)	82201	5	5	10	Desconhecido
Ozone (YUAN et al., 2008)	2534	72	0	2	Desconhecido
Airlines (IKONOMOVSKA; BIFET, 2009)	539383	3	4	2	Desconhecido
Keystroke (MAXION; KILLOURHY, 2010)	1600	10	0	4	Desconhecido
GMSC (FUSION, 2011)	150000	10	0	2	Desconhecido
NOOA (ELWELL; POLIKAR, 2011)	18154	8	0	2	Desconhecido
Gas Sensor (VERGARA et al., 2013)	13910	128	0	6	Desconhecido
Nomao (CANDILLIER; LEMAIRE, 2013)	9844	89	29	2	Desconhecido
Powersupply (LICHMAN, 2013)	29928	2	0	24	Desconhecido
Luxembourg (CODECA; FRANK; ENGEL, 2015)	1901	16	15	2	Desconhecido
Outdoor (LOSING; HAMMER; WERSING, 2016)	4000	21	0	40	Desconhecido
Rialto (LOSING; HAMMER; WERSING, 2016)	82250	27	0	10	Desconhecido

5.1.2 Variações da constante de *Poisson*

Conforme apresentado na seção 4.2, foram empregados valores predeterminados para a variação da constante de *Poisson*. A Tabela 3 relaciona os valores tanto para os classificadores base (*base learners*) selecionados quanto para os não selecionados.

Tabela 3 – Variações utilizadas

Variação	**Selecionados** (λs)	**Não Selecionados** (λn)
1	3	1
2	1	3
3	6	1
4	1	6
5	10	1
6	1	10
7	12	1
8	1	12

Como não é possível prever *a priori* a taxa de seleção dos classificadores (quantos classificadores serão selecionados) e considerando que o processo de treinamento diferenciado influencia a performance, foram realizados experimentos com cada uma das 8 possibilidades listadas na Tabela 3. O objetivo foi analisar o impacto do aumento ou redução do treinamento nos subconjuntos selecionados e não selecionados sobre os resultados e a diversidade do *ensemble*.

5.1.3 Métricas utilizadas

Os resultados apresentados foram obtidos por meio de quatro métricas disponíveis no *software* MOA, cujas equações foram previamente apresentadas na Seção 2.1.1:

1. Acurácia: Esta métrica é amplamente utilizada para a avaliação de algoritmos, tanto em ambientes *on-line* quanto em ambientes *off-line*, demonstrando as taxas de previsões corretas realizadas.

2. Estatística Kappa: Empregada para avaliar a concordância entre as previsões do modelo e as observações reais, é particularmente útil em conjuntos de dados com alto grau de desbalanceamento, razão pela qual é utilizada neste estudo.

3. Tempo de Avaliação do Modelo, ou *CPU Seconds*: Mede a quantidade de tempo (em segundos) em que o processo do algoritmo permanece ativo.

4. Custo do Modelo (*RAM Hours*): Avalia a quantidade de gigabytes por hora consumidos pelo classificador.

5.1.4 Testes estatísticos

Para a comparação de desempenhos entre as configurações utilizadas e os algoritmos de estado da arte, foram empregados os testes de Friedman e Nemenyi. O teste de Friedman, um teste não-paramétrico, é utilizado para detectar diferenças significativas

entre os algoritmos avaliados em diferentes bases de dados (DEMSAR, 2006), conforme representado pela Equação 5.1.

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{5.1}$$

Onde χ_F^2 é a estatística do teste de Friedman, N representa o número de bases de dados, k é o número de algoritmos comparados e R_j é a soma dos ranks atribuídos ao algoritmo j.

Adicionalmente, o teste de Nemenyi, amplamente utilizado para a comparação entre múltiplos classificadores, avalia se o desempenho é significativamente distinto, com as classificações médias diferindo por pelo menos uma diferença estatística (CD) (DEMSAR, 2006), conforme representado pela Equação 5.2.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{5.2}$$

A Distância Crítica *(Critical Distance)* - CD é empregada para avaliar se dois algoritmos apresentam diferenças estatisticamente significativas em seu desempenho. O valor q_α representa o valor crítico derivado da distribuição *Studentized range* para um nível de significância específico α. O parâmetro k denota o número total de algoritmos comparados, enquanto N refere-se ao número de bases de dados utilizadas na comparação.

5.2 Melhor configuração

As diferentes configurações do LGDS seguem um padrão de codificação para facilitar a compreensão. Por exemplo, $ARF\_sw\{A\}\_S\{B\}\_NS\{C\}$ indica que o LGDS foi aplicado sobre o ARF com uma janela de tamanho w, utilizando valores λs e λn de λ para os conjuntos de classificadores selecionados e não selecionados, respectivamente. Com os *datasets* previamente descritos para a fase de ajuste do algoritmo, foram utilizadas apenas as métricas de acurácia e estatística Kappa para a seleção das melhores configurações.

Figura 4 – Distância crítica para os valores médios obtidos para métrica acurácia em todas configurações

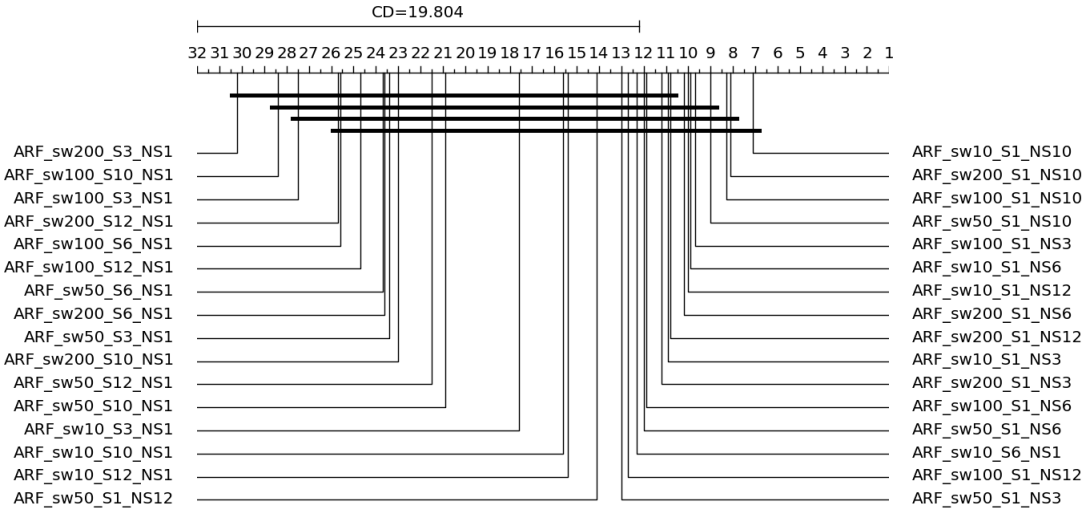
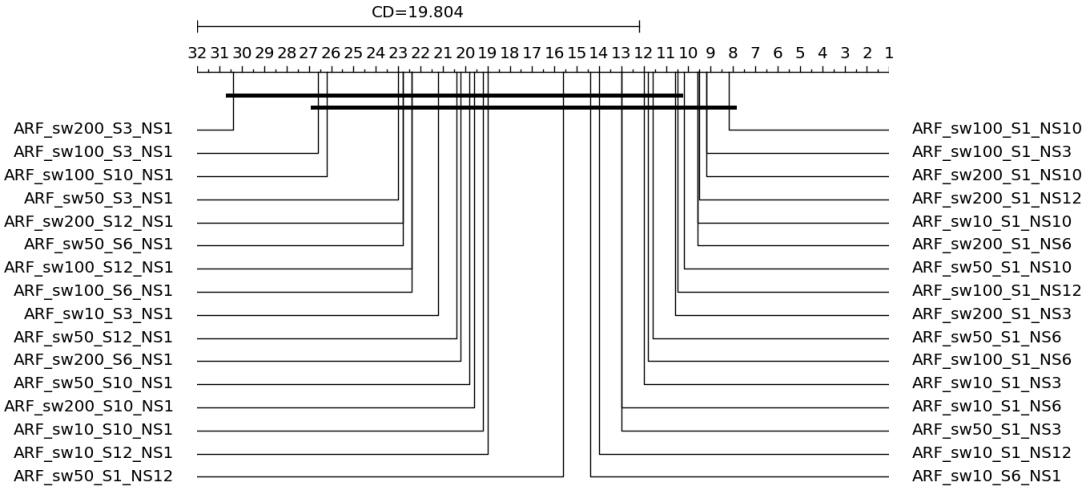


Figura 5 – Distância crítica para os valores médios obtidos para métrica kappa em todas configurações



Os resultados do teste de Friedman indicaram um *p-value* de $1,7e^{-7}$ para a acurácia, e os resultados com a distância crítica estão ilustrados na Figura 4. Para a métrica estatística Kappa, foi obtido um *p-value* de $1,4e^{-4}$, com o gráfico de distância crítica apresentado na Figura 5. Esta análise fundamentou a seleção de duas configurações para comparação com os demais algoritmos, sendo escolhidas as configurações *ARF_sw100_S1_NS10* e *ARF_sw200_S1_NS10*, que se destacaram entre as três melhores no ranking médio, considerando tanto a estatística Kappa quanto a acurácia. Ademais, as Figuras 4 e 5

revelam um padrão consistente que sugere que os melhores resultados são alcançados com menos treinamento no conjunto de classificadores selecionados (S1) e mais treinamento nos classificadores não selecionados (NS10, NS3, NS12, NS6).

5.3 Comparação com outros algoritmos

Nesta seção, são apresentados os resultados comparativos entre as duas configurações descritas na Seção 5.2 e os dois algoritmos de estado da arte, avaliados em termos de desempenho preditivo e consumo de recursos computacionais. Destaca-se que, para ambos, também foi aplicada a modificação do componente aleatório, com a coluna *Original* referenciando os resultados do ARF em sua versão original. As tabelas são acompanhadas pelas figuras com o teste *post-hoc* de Nemenyi correspondente.

Os resultados também incluem as métricas de acurácia e a estatística kappa do oráculo correspondente para as configurações selecionadas, sendo este uma referência comparativa que representa um modelo com desempenho teórico, ou seja, com a melhor seleção possível. O oráculo seleciona um classificador capaz de fornecer uma resposta correta em uma amostra (KUNCHEVA, 2002). Por ser considerado um modelo de seleção ideal, ele constitui um limite superior para as técnicas analisadas neste estudo. É importante destacar que, no contexto de mineração de fluxo de dados, os membros do *ensemble* evoluem ao longo do tempo, diferentemente do cenário *batch*, onde os classificadores são estáticos e não sofrem alterações. Isso implica que, com a chegada de novos dados e as modificações nos classificadores, o oráculo também se adapta ao longo do tempo. Portanto, ele deve acompanhar a evolução do *ensemble* e torna-se sensível aos parâmetros das configurações utilizadas nos experimentos, como nas colunas *Oraculo_SW100* e *Oraculo_SW200* da Tabela 4. Assim, mesmo que os fluxos sejam idênticos, os resultados dos oráculos gerados para diferentes configurações podem variar.

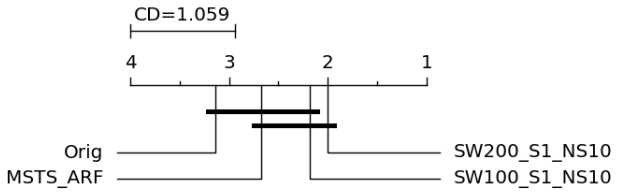
Em ambas as configurações apresentadas na Tabela 4 e comparadas com seus respectivos oráculos, observamos que algumas bases demonstram uma aproximação ao modelo ideal dos resultados, particularmente nas configurações dos *datasets convertype*, *eletricity*, *gmsc*, *kdd99*, *kdd99cup*, *luxembourg* e *ozone*. Nos demais casos, identificamos oportunidades de melhorias nos experimentos para que os resultados se aproximem ainda mais do ideal.

Tabela 4 – Acurácia média para os *datasets* reais

	Acurácia					
	Original	**SW100_S1_NS10**	**Oraculo_SW100**	**SW200_S1_NS10**	**Oraculo_SW200**	**MSTS_ARF**
connect_4	$62,94 \pm 1,60$	$\mathbf{69,18 \pm 1,58}$	$97,07 \pm 0,75$	$67,96 \pm 2,51$	96,21 ± 1,44	$64,63 \pm 3,18$
covertype	$96,96 \pm 1,00$	$\mathbf{98,57 \pm 0,29}$	$99,81 \pm 0,01$	$91,13 \pm 11,13$	99,78 ± 0,02	$96,72 \pm 0,55$
electricity	$85,95 \pm 2,22$	$\mathbf{90,12 \pm 2,09}$	$91,23 \pm 9,66$	$89,71 \pm 1,88$	91,25 ± 9,67	$87,99 \pm 1,23$
gassensor	$89,85 \pm 1,43$	$90,79 \pm 1,78$	$96,55 \pm 2,76$	$\mathbf{90,82 \pm 4,14}$	96,35 ± 3,13	$88,87 \pm 0,21$
gmsc	$91,40 \pm 0,51$	$91,39 \pm 0,45$	$95,29 \pm 2,01$	$91,67 \pm 0,28$	93,93 ± 3,77	$\mathbf{92,19 \pm 0,62}$
kdd99	$99,98 \pm 0,04$	$\mathbf{100,00 \pm 0,00}$	$100,00 \pm 0,00$	$\mathbf{100,00 \pm 0,00}$	100,00 ± 0,00	$99,99 \pm 0,03$
kddcup99	$99,87 \pm 0,11$	$\mathbf{99,95 \pm 5,00}$	$100,00 \pm 0,00$	$\mathbf{99,95 \pm 0,05}$	100,00 ± 0,10	$99,77 \pm 0,20$
keystroke	$86,23 \pm 1,24$	$\mathbf{88,80 \pm 2,43}$	$91,42 \pm 7,31$	$88,41 \pm 2,38$	91,34 ± 7,25	$86,14 \pm 1,66$
luxembourg	$\mathbf{100,00 \pm 0,00}$	$\mathbf{100,00 \pm 0,00}$	$99,50 \pm 0,07$	$\mathbf{100,00 \pm 0,00}$	99,50 ± 0,07	$\mathbf{100,00 \pm 0,00}$
NOAA	$68,03 \pm 1,88$	$71,75 \pm 1,35$	$92,55 \pm 7,01$	$\mathbf{71,90 \pm 1,41}$	92,55 ± 7,01	$71,10 \pm 2,40$
outdoor	$59,68 \pm 2,12$	$60,06 \pm 3,36$	$75,24 \pm 0,66$	$60,39 \pm 2,63$	74,24 ± 3,30	$\mathbf{61,22 \pm 2,31}$
ozone	$95,10 \pm 0,35$	$95,00 \pm 0,30$	$95,57 \pm 0,92$	$95,10 \pm 0,41$	95,56 ± 0,88	$\mathbf{95,77 \pm 0,19}$
pokerhand	$74,63 \pm 7,14$	$79,20 \pm 14,06$	$97,68 \pm 3,16$	$\mathbf{83,97 \pm 6,82}$	88,31 ± 13,25	$71,18 \pm 5,05$
power_supply	$14,11 \pm 1,34$	$13,28 \pm 0,94$	$29,32 \pm 15,06$	$13,62 \pm 1,44$	18,12 ± 7,81=	$\mathbf{15,98 \pm 1,14}$
Rank médio	3,14	2,18		**2,00**		2,68

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 6 – Gráfico de distância crítica para os valores médios obtidos para métrica acurácia nos *datasets* reais



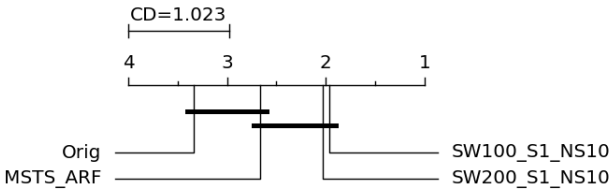
Para os resultados apresentados na Tabela 4, o *p-value* obtido foi de 0,06, indicando que não houve diferença estatisticamente significativa em relação ao MSTS-ARF nos *datasets* reais.

Tabela 5 – Acurácia média para os *datasets* sintéticos

	Acurácia					
	Original	**SW100_S1_NS10**	**Oraculo_SW100**	**SW200_S1_NS10**	**Oraculo_SW200**	**MSTS_ARF**
AGR_A	$79,16 \pm 2,08$	$\mathbf{81,59 \pm 2,27}$	$99,59 \pm 0,16$	$79,72 \pm 5,21$	$99,53 \pm 0,21$	$79,01 \pm 2,54$
AGR_G	$71,34 \pm 3,43$	$73,05 \pm 5,75$	$99,53 \pm 0,17$	$\mathbf{75,15 \pm 3,77}$	$99,42 \pm 0,21$	$74,54 \pm 3,41$
LED_A	$\mathbf{71,58 \pm 0,89}$	$69,63 \pm 3,44$	$83,11 \pm 6,99$	$70,06 \pm 3,54$	$82,80 \pm 6,70$	$71,27 \pm 0,94$
LED_G	$71,86 \pm 1,17$	$71,70 \pm 1,76$	$83,32 \pm 8,97$	$71,24 \pm 3,83$	$82,48 \pm 8,28$	$\mathbf{72,21 \pm 1,00}$
mixedBalac	$98,47 \pm 0,29$	$\mathbf{98,85 \pm 0,41}$	$98,82 \pm 1,38$	$98,80 \pm 0,41$	$98,82 \pm 1,38$	$97,87 \pm 0,47$
mixedImbalac	$98,79 \pm 0,19$	$\mathbf{99,14 \pm 0,36}$	$98,92 \pm 1,26$	$99,10 \pm 0,24$	$98,92 \pm 1,26$	$98,39 \pm 0,61$
RBF_F	$50,72 \pm 3,12$	$59,75 \pm 3,36$	$80,77 \pm 23,16$	$\mathbf{60,78 \pm 4,01}$	$80,65 \pm 23,06$	$52,64 \pm 4,02$
RBF_M	$71,08 \pm 1,85$	$\mathbf{77,04 \pm 4,41}$	$88,12 \pm 14,21$	$75,57 \pm 3,23$	$88,10 \pm 14,19$	$63,24 \pm 8,62$
RTG	$90,20 \pm 1,30$	$91,28 \pm 1,09$	$94,06 \pm 6,92$	$91,54 \pm 1,24$	$94,08 \pm 6,23$	$\mathbf{92,17 \pm 0,67}$
RTG_A	$86,93 \pm 0,78$	$88,34 \pm 1,54$	$90,87 \pm 10,87$	$\mathbf{88,57 \pm 1,63}$	$90,82 \pm 10,83$	$86,25 \pm 1,82$
RTG_G	$83,77 \pm 2,32$	$\mathbf{85,68 \pm 1,92}$	$87,57 \pm 14,95$	$85,34 \pm 2,10$	$87,54 \pm 14,93$	$85,31 \pm 2,01$
SEA_A	$87,80 \pm 1,51$	$90,26 \pm 0,41$	$95,50 \pm 0,79$	$\mathbf{90,44 \pm 0,20}$	$95,54 \pm 0,82$	$89,37 \pm 0,36$
SEA_G	$87,35 \pm 1,84$	$\mathbf{89,86 \pm 0,41}$	$95,90 \pm 0,92$	$89,82 \pm 0,44$	$95,70 \pm 0,98$	$89,37 \pm 0,29$
WAVEFORM	$79,28 \pm 1,67$	$79,71 \pm 0,98$	$90,42 \pm 11,38$	$79,71 \pm 0,84$	$90,41 \pm 11,37$	$\mathbf{82,72 \pm 0,52}$
WAVEFORM_DRIFT	$80,17 \pm 1,68$	$81,40 \pm 1,81$	$99,64 \pm 0,10$	$81,37 \pm 1,48$	$99,55 \pm 0,14$	$\mathbf{83,54 \pm 0,74}$
Rank médio	3,33	**1,97**		2,03		2,67

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 7 – Gráfico de distância crítica para os valores médios obtidos para métrica acurácia nos *datasets* sintéticos



A Tabela 5 apresenta os resultados de acurácia para os dados sintéticos, com um valor de *p-value* de 0,01, indicando uma diferença significativa. Ambas as configurações ocupam a primeira e a segunda posição no ranking médio, mas não apresentam diferença crítica em relação ao algoritmo MSTS-ARF, conforme ilustrado na Figura 7. Nos dois *datasets* gerados com a base *RBF*, observam-se resultados superiores aos dos outros algoritmos, demonstrando que a técnica utilizada facilita a recuperação de desempenho diante dos dois tipos diferentes de desvios de conceito apresentados em ambas as bases.

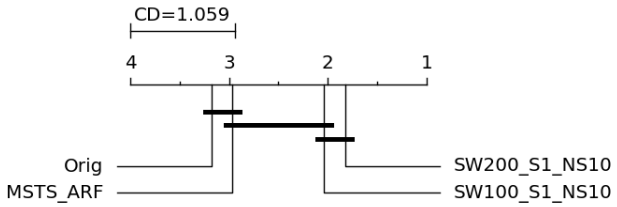
Em relação aos oráculos, em ambas as configurações apresentadas, observamos que algumas bases se aproximam do modelo ideal nos *datasets mixedBalanc*, *mixedImbalac*, *RTG_A* e *RTG_G*. Nos demais casos, identificamos a necessidade de melhorias para que os resultados se aproximem do ideal.

Tabela 6 – Estatística Kappa média para os *datasets* reais

	Kappa					
	Original	**SW100_S1_NS10**	**Oraculo_SW100**	**SW200_S1_NS10**	**Oraculo_SW200**	**MSTS_ARF**
connect_4	$32,33 \pm 2,88$	$\mathbf{43,49 \pm 3,26}$	$97,66 \pm 0,74$	$40,98 \pm 4,69$	$96,20 \pm 1,44$	$33,66 \pm 8,02$
covertype	$86,60 \pm 4,48$	$\mathbf{93,71 \pm 1,28}$	$99,81 \pm 0,00$	$75,43 \pm 25,7$	$99,77 \pm 0,02$	$85,71 \pm 2,18$
electricity	$71,90 \pm 4,50$	$\mathbf{80,24 \pm 4,17}$	$91,23 \pm 9,65$	$79,43 \pm 3,74$	$91,24 \pm 9,67$	$75,93 \pm 2,49$
gassensor	$87,59 \pm 1,73$	$88.71 \pm 2,18$	$96,55 \pm 2,76$	$\mathbf{88,79 \pm 4.98}$	$96,34 \pm 3,13$	$86,38 \pm 0,25$
gmsc	$18,01 \pm 5,82$	$20,56 \pm 5,90$	$95,29 \pm 2,01$	$\mathbf{22,93 \pm 5,80}$	$93,92 \pm 3,77$	$21,77 \pm 6,26$
kdd99	$80,00 \pm 40,00$	$\mathbf{100,00 \pm 0,00}$	$99,98 \pm 0,00$	$\mathbf{100,00 \pm 0,00}$	$99,98 \pm 0,00$	$90,00 \pm 30,00$
kddcup99	$99,78 \pm 0,19$	$\mathbf{99,91 \pm 0,09}$	$99,80 \pm 0,11$	$\mathbf{99,91 \pm 0,90}$	$99,79 \pm 0,10$	$99,61 \pm 0,33$
keystroke	$81,64 \pm 1,65$	$\mathbf{85,07 \pm 3,23}$	$91,41 \pm 7,30$	$84,55 \pm 3.17$	$91,34 \pm 7,25$	$81,52 \pm 2,21$
luxembourg	$\mathbf{100,00 \pm 0,0}$	$\mathbf{100,00 \pm 0,00}$	$99,50 \pm 0,07$	$\mathbf{100,00 \pm 0,00}$	$99,50 \pm 0,07$	$\mathbf{100,00 \pm 0,0}$
NOAA	$32,03 \pm 5,01$	$39,07 \pm 2,47$	$92,55 \pm 7,00$	$\mathbf{39,51 \pm 2,65}$	$92,55 \pm 7,01$	$37,92 \pm 7,23$
outdoor	$58,33 \pm 2,19$	$58,76 \pm 3,46$	$75,23 \pm 0,66$	$59,10 \pm 2,71$	$74,24 \pm 3,30$	$\mathbf{59,92 \pm 2,39}$
ozone	$9,5 \pm 5,53$	$7,25 \pm 4,66$	$95,56 \pm 0,91$	$\mathbf{9,43 \pm 6,08}$	$95,56 \pm 0,88$	$3,57 \pm 5,08$
pokerhand	$53,74 \pm 15,16$	$66,12 \pm 19,30$	$97,56 \pm 0,91$	$\mathbf{71,57 \pm 12,43}$	$88,31 \pm 13,25$	$40,61 \pm 14,54$
power_supply	$10,40 \pm 1,39$	$9,53 \pm 0,98$	$29,31 \pm 15,55$	$9,88 \pm 1,49$	$18,12 \pm 7,81$	$\mathbf{12,34 \pm 1,19}$
Rank médio	3,18	2,04		**1,82**		2,96

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 8 – Gráfico de distância crítica para os valores médios obtidos para métrica Estatística Kappa nos *datasets* reais



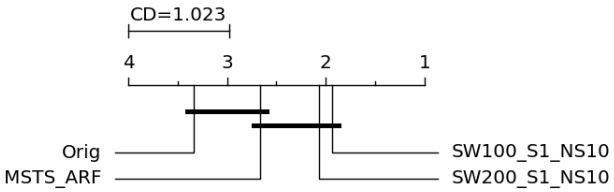
Ao analisar a estatística kappa nos *datasets* reais, conforme apresentado na Tabela 6, observa-se uma diferença entre a configuração que ocupa a primeira posição no ranking médio, SW200_S1_NS10, e os demais algoritmos. Esta configuração obteve melhores resultados individuais em 8 de um total de 14 *datasets*, destacando-se particularmente no desempenho no *dataset pokerhand*, com um valor de kappa de 71,57%. Este resultado apresenta uma diferença crítica em relação aos outros algoritmos utilizados na comparação, conforme ilustrado na Figura 8. Além disso, o valor de *p-value* de 0,06 não indica uma diferença significativa em comparação com os outros algoritmos, acompanhando o desempenho observado na métrica de acurácia.

Tabela 7 – Estatística Kappa média para os *datasets* sintéticos

	Kappa					
	Original	**SW100_S1_NS10**	**Oraculo_SW100**	**SW200_S1_NS10**	**Oraculo_SW200**	**MSTS_ARF**
AGR_A	$56,96 \pm 4,0$	$\mathbf{61,85 \pm 4,27}$	99.59 ± 0.16	$58,22 \pm 9,74$	$99.53 \pm 0,21$	$56,74 \pm 5,24$
AGR_G	$42,04 \pm 6,88$	$45,73 \pm 9,85$	99.53 ± 0.17	$\mathbf{49,16 \pm 6,81}$	$99,42 \pm 0,21$	$48,25 \pm 6,94$
LED_A	$\mathbf{68,4 \pm 0,99}$	$66,23 \pm 3,82$	$83,11 \pm 7,95$	$66,71 \pm 3,93$	$82,80 \pm 6,70$	$68,07 \pm 1,04$
LED_G	$68,70 \pm 1,29$	$68,52 \pm 1,96$	$83,32 \pm 8.97$	$68,0 \pm 4,28$	$82,48 \pm 8.28$	$\mathbf{69,09 \pm 1,11}$
mixedBalac	$96,94 \pm 0,58$	$\mathbf{97,7 \pm 0,81}$	98.82 ± 1.38	$97,6 \pm 0,83$	$98,82 \pm 1,38$	$95,74 \pm 0,94$
mixedImbalac	$97,58 \pm 0,38$	$\mathbf{98,28 \pm 0,72}$	$98,92 \pm 1.26$	$98,20 \pm 0,47$	$98.92 \pm 1,26$	$96,78 \pm 1,22$
RBF_F	$35,52 \pm 4,51$	$47,44 \pm 4,27$	$80,77 \pm 23,16$	$\mathbf{48,81 \pm 5,07}$	80.65 ± 23.06	$37,5 \pm 5,66$
RBF_M	$62,35 \pm 2,36$	$\mathbf{70,14 \pm 5,72}$	$88,12 \pm 14,21$	$68,23 \pm 4,19$	$88,10 \pm 14,19$	$51,88 \pm 11,70$
RTG	$79,72 \pm 2,62$	$81,94 \pm 2,22$	$94,06 \pm 6.92$	$82,47 \pm 2,52$	$94,08 \pm 6,93$	$\mathbf{83,71 \pm 1,40}$
RTG_A	$71,59 \pm 1,71$	$74,66 \pm 3,27$	$90,87 \pm 10.87$	$\mathbf{75,29 \pm 3,56}$	$90,82 \pm 10,83$	$70,14 \pm 3,97$
RTG_G	$64,51 \pm 5,05$	$\mathbf{68,8 \pm 4,34}$	87.57 ± 14.95	$68,19 \pm 4,75$	$87,54 \pm 14,93$	$67,74 \pm 4,57$
SEA_A	$75,42 \pm 2,98$	$80,31 \pm 0,83$	$95,50 \pm 0.79$	$\mathbf{80,67 \pm 0,4}$	$95,54 \pm 0,82$	$78,52 \pm 0,73$
SEA_G	$74,48 \pm 3,67$	$\mathbf{79,48 \pm 0,81}$	95.90 ± 0.92	$79,39 \pm 0,88$	$95,70 \pm 0,98$	$78,5 \pm 0,57$
WAVEFORM	$68,91 \pm 2,49$	$69,55 \pm 1,47$	$90,42 \pm 0.92$	$69,55 \pm 1,26$	$90,41 \pm 11,37$	$\mathbf{74,06 \pm 0,78}$
WAVEFORM_DRIFT	$70,26 \pm 2,52$	$72,1 \pm 2,72$	99.64 ± 0.10	$72,06 \pm 2,23$	$99,55 \pm 0,14$	$\mathbf{75,31 \pm 1,11}$
Rank médio	3,33	**1,93**		2,07		2,67

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 9 – Gráfico de distância crítica para os valores médios obtidos para métrica Estatística Kappa nos *datasets* sintéticos



Nos *datasets* sintéticos, o *p-value* foi igual a 0,01, indicando uma diferença estatisticamente significativa. No entanto, em relação à distância crítica no contexto do MSTS_ARF, não se observa uma diferença crítica, conforme ilustrado na Figura 9. Ao analisar individualmente a configuração SW100_S1_NS10, verificou-se que ela apresentou melhorias em 6 *datasets*. Na Tabela 7, destacam-se os *datasets RBF_f*, *RBF_m* e *RTG_a*, que repetiram o desempenho observado na métrica de acurácia.

Em contraste com os resultados anteriores, as métricas relacionadas ao consumo de recursos computacionais não evidenciaram melhorias, conforme discutido a seguir. Esse comportamento pode ser atribuído a uma combinação de fatores, como a quantidade de treino realizada pelos classificadores e o tamanho das janelas deslizantes utilizadas.

Tabela 8 – Tempo de Processamento para os *datasets* reais

	Tempo de Processamento (s)			
	Original	**SW100_S1_NS10**	**SW200_S1_NS10**	**MSTS_ARF**
connect_4	$463,59 \pm 318,77$	$522,28 \pm 207,43$	$327,50 \pm 106,83$	$\mathbf{53,67 \pm 10,85}$
covertype	$2839,22 \pm 1451,12$	$2930,59 \pm 977,18$	$1841,07 \pm 297,91$	$\mathbf{950,90 \pm 85,44}$
electricity	$117,78 \pm 64,74$	$116,73 \pm 45,90$	$78,86 \pm 14,06$	$\mathbf{26,35 \pm 5,49}$
gassensor	$171,48 \pm 89,47$	$168,63 \pm 59,51$	$128,61 \pm 23,30$	$\mathbf{53,25 \pm 4,98}$
gmsc	$267,95 \pm 156,75$	$299,35 \pm 115,79$	$188,83 \pm 40,13$	$\mathbf{56,28 \pm 11,55}$
kdd99	$6038,63 \pm 3175,96$	$6727,11 \pm 1739,44$	$4290,62 \pm 823,66$	$\mathbf{2671,24 \pm 144,05}$
kddcup99	$654,31 \pm 201,87$	$865,84 \pm 174,82$	$572,54 \pm 17,51$	$\mathbf{497,62 \pm 76,97}$
keystroke	$3,78 \pm 1,91$	$4,03 \pm 2,59$	$2,77 \pm 0,48$	$\mathbf{1,37 \pm 0,15}$
luxembourg	$2,12 \pm 0,81$	$1,70 \pm 0,76$	$1,38 \pm 0,06$	$\mathbf{1,19 \pm 0,07}$
NOAA	$51,11 \pm 33,58$	$57,65 \pm 24,79$	$34,94 \pm 8,99$	$\mathbf{7,65 \pm 1,62}$
outdoor	$42,04 \pm 18,18$	$50,95 \pm 16,42$	$36,46 \pm 2,70$	$\mathbf{18,65 \pm 0,68}$
ozone	$24,09 \pm 14,72$	$24,13 \pm 10,75$	$17,08 \pm 3,33$	$\mathbf{4,77 \pm 1,26}$
pokerhand	$2728,33 \pm 1759,47$	$3013,82 \pm 1165,12$	$1820,27 \pm 370,34$	$\mathbf{391,93 \pm 77,97}$
power_supply	$102,96 \pm 67,09$	$125,36 \pm 66,42$	$78,44 \pm 21,77$	$\mathbf{18,10 \pm 3,28}$
Rank médio	3,21	3,79	2,00	**1,00**

Valores em **negrito** indicam o melhor resultado por *datasets*

Quanto ao tempo de processamento nos *datasets* reais, as configurações não superaram o MSTS. Na Tabela 8, é possível observar valores próximos entre o ARF. Entretanto, não houve diferença estatística entre o MSTS_ARF e a configuração SW200_S1_NS10,

conforme ilustrado na Figura 10. Em relação à versão original do ARF, a mesma configuração, assim como o MSTS, apresentou ganhos com diferença estatística significativa.

Figura 10 – Gráfico da distância crítica para os valores obtidos na métrica tempo de processamento nos *datasets* reais

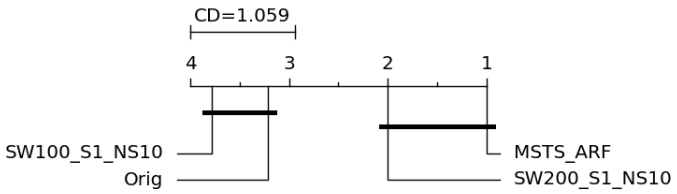
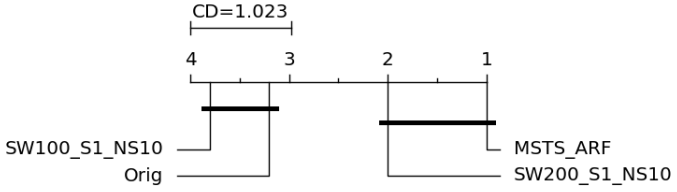


Tabela 9 – Tempo de Processamento para os *datasets* sintéticos

	Tempo de Processamento (s)			
	Original	**SW100_S1_NS10**	**SW200_S1_NS10**	**MSTS_ARF**
AGR_A	$1383,70 \pm 913,85$	$1431,52 \pm 677,00$	$913,58 \pm 201,60$	$\mathbf{215,87 \pm 52,36}$
AGR_G	$1818,01 \pm 1184,49$	$1583,95 \pm 661,99$	$1052,14 \pm 217,94$	$\mathbf{261,70 \pm 62,26}$
LED_A	$887,13 \pm 398,75$	$1036,54 \pm 422,66$	$664,83 \pm 84,78$	$\mathbf{393,92 \pm 39,86}$
LED_G	$891,58 \pm 380,02$	$1139,68 \pm 457,44$	$682,96 \pm 72,94$	$\mathbf{344,63 \pm 62,46}$
mixedBalac	$177,68 \pm 82,13$	$208,38 \pm 87,60$	$149,89 \pm 10,99$	$\mathbf{73,27 \pm 5,10}$
mixedImbalac	$178,56 \pm 82,72$	$227,50 \pm 75,12$	$150,80 \pm 11,71$	$\mathbf{73,68 \pm 5,13}$
RBF_F	$1198,73 \pm 730,13$	$1243,58 \pm 498,16$	$862,86 \pm 153,80$	$\mathbf{254,68 \pm 33,59}$
RBF_M	$1078,73 \pm 595,21$	$1161,74 \pm 446,34$	$775,30 \pm 119,82$	$\mathbf{267,25 \pm 47,49}$
RTG	$1061,99 \pm 669,87$	$1155,83 \pm 525,71$	$719,31 \pm 157,68$	$\mathbf{209,12 \pm 39,16}$
RTG_A	$1465,69 \pm 877,95$	$1463,01 \pm 665,62$	$956,53 \pm 180,51$	$\mathbf{245,08 \pm 64,38}$
RTG_G	$1819,71 \pm 1127,42$	$1798,74 \pm 866,85$	$1126,24 \pm 205,93$	$\mathbf{271,52 \pm 67,59}$
SEA_A	$565,72 \pm 344,52$	$728,21 \pm 341,14$	$455,17 \pm 100,09$	$\mathbf{111,67 \pm 19,27}$
SEA_G	$569,36 \pm 348,96$	$748,64 \pm 354,33$	$455,94 \pm 105,66$	$\mathbf{109,64 \pm 19,88}$
WAVEFORM	$3239,92 \pm 2007,27$	$3413,06 \pm 1672,05$	$2357,05 \pm 591,32$	$\mathbf{660,03 \pm 181,43}$
WAVEFORM_DRIFT	$3270,33 \pm 1924,43$	$3421,31 \pm 1650,24$	$2308,48 \pm 601,84$	$\mathbf{660,25 \pm 120,44}$
Rank médio	3,20	3,80	2,00	**1,00**

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 11 – Gráfico da distância crítica para os valores obtidos na métrica tempo de processamento nos *datasets* sintéticos



O mesmo comportamento é observado nos *datasets* sintéticos, conforme demonstrado na Tabela 9 e na Figura 11. A configuração SW200_S1_NS10 apresenta uma diferença

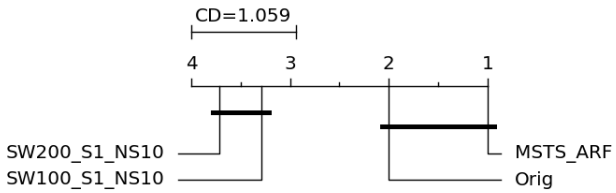
estatisticamente significativa em relação à configuração original do ARF, enquanto não há diferença estatística em relação ao MSTS_ARF.

Tabela 10 – Custo do Modelo nos *datasets* reais

	Custo do Modelo			
	Original	**SW100_S1_NS10**	**SW200_S1_NS10**	**MSTS_ARF**
connect_4	1,0e−4 ± 8,8e−5	2,9e−4 ± 1,2e−4	2,9e−4 ± 9,4e−5	**8,3e−6 ± 2,0e−6**
covertype	8,7e−4 ± 5,1e−4	2,9e−3 ± 9,6e−4	3,0e−3 ± 4,9e−4	**2,3e−4 ± 5,8e−5**
electricity	3,3e−5 ± 2,1e−5	6,3e−5 ± 2,3e−5	6,1e−5 ± 9,9e−6	**5,6e−6 ± 1,7e−6**
gassensor	3,9e−5 ± 2,7e−5	1,4e−4 ± 4,9e−5	1,8e−4 ± 3,6e−5	**1,1e−5 ± 1,0e−6**
gmsc	8,1e−5 ± 5,1e−5	1,6e−4 ± 6,5e−5	1,4e−4 ± 3,1e−5	**1,6e−5 ± 4,8e−6**
kdd99	2,4e−3 ± 1,5e−3	1,7e−2 ± 4,4e−3	2,0e−2 ± 3,9e−3	**9,4e−4 ± 1,9e−4**
kddcup99	2,2e−4 ± 6,9e−5	2,2e−3 ± 4,4e−4	2,6e−3 ± 7,8e−5	**1,6e−4 ± 2,4e−5**
keystroke	7,4e−7 ± 3,8e−7	2,6e−6 ± 1,7e−6	2,9e−6 ± 4,9e−7	**2,7e−7 ± 3,0e−8**
luxembourg	4,0e−7 ± 1,5e−7	7,6e−7 ± 3,4e−7	9,3e−7 ± 4,1e−8	**2,3e−7 ± 1,3e−8**
NOAA	9,6e−6 ± 5,7e−6	2,7e−5 ± 1,1e−5	2,4e−5 ± 5,7e−6	**2,0e−6 ± 5,1e−7**
outdoor	1,3e−5 ± 5,4e−6	2,1e−4 ± 6,7e−5	2,7e−4 ± 2,0e−5	**5,6e−6 ± 2,2e−7**
ozone	4,7e−6 ± 2,6e−6	1,1e−5 ± 4,8e−6	1,2e−5 ± 2,1e−6	**1,0e−6 ± 4,1e−7**
pokerhand	8,3e−4 ± 5,5e−4	3,8e−3 ± 1,5e−3	4,0e−3 ± 8,4e−4	**1,2e−4 ± 4,3e−5**
power_supply	3,4e−5 ± 2,6e−5	3,2e−4 ± 1,7e−4	3,6e−4 ± 1,0e−4	**7,2e−6 ± 2,6e−6**
Rank médio	2,00	3,29	3,71	**1,00**

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 12 – Gráfico de distância crítica para os valores obtidos na métrica custo do modelo nos *datasets* reais



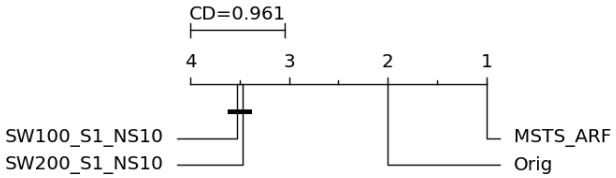
Na avaliação do custo do modelo em *datasets* reais, as configurações selecionadas do LGD demonstraram um maior consumo de memória, conforme evidenciado na Tabela 10 e no gráfico de distância crítica da Figura 12.

Tabela 11 – Custo do Modelo nos *datasets* sintéticos

	Custo do Modelo			
	Original	**SW100_S1_NS10**	**SW200_S1_NS10**	**MSTS_ARF**
AGR_A	3,5e−4 ± 2,4e−4	7,7e−4 ± 3,6e−4	6,9e−4 ± 1,5e−4	**6,6e−5 ± 1,8e−5**
AGR_G	5,1e−4 ± 3,6e−4	8,2e−4 ± 3,2e−4	7,8e−4 ± 1,5e−4	**7,9e−5 ± 2,6e−5**
LED_A	3,3e−4 ± 1,7e−4	1,4e−3 ± 5,7e−4	1,5e−3 ± 2,1e−4	**1,5e−4 ± 2,7e−5**
LED_G	3,0e−4 ± 1,5e−4	1,5e−3 ± 6,1e−4	1,5e−3 ± 1,7e−4	**1,2e−4 ± 4,3e−5**
mixedBalac	4,2e−5 ± 2,0e−5	1,0e−4 ± 4,3e−5	1,1e−4 ± 7,9e−6	**1,7e−5 ± 1,4e−6**
mixedImbalac	4,1e−5 ± 1,9e−5	1,1e−4 ± 3,8e−5	1,1e−4 ± 9,0e−6	**1,7e−5 ± 1,2e−6**
RBF_F	3,1e−4 ± 2,3e−4	9,6e−4 ± 3,8e−4	1,1e−3 ± 1,9e−4	**5,8e−5 ± 1,9e−5**
RBF_M	3,0e−4 ± 1,6e−4	9,4e−4 ± 3,8e−4	9,8e−4 ± 1,5e−4	**6,3e−5 ± 1,7e−5**
RTG	2,6e−4 ± 1,7e−4	5,8e−4 ± 2,7e−4	5,3e−4 ± 1,2e−4	**5,8e−5 ± 1,7e−5**
RTG_A	3,4e−4 ± 2,0e−4	7,2e−4 ± 3,3e−4	6,9e−4 ± 1,3e−4	**6,5e−5 ± 2,8e−5**
RTG_G	4,3e−4 ± 2,6e−4	9,3e−4 ± 4,7e−4	8,3e−4 ± 1,6e−4	**6,7e−5 ± 2,0e−5**
SEA_A	1,4e−4 ± 8,3e−5	3,7e−4 ± 1,7e−4	3,3e−4 ± 7,3e−5	**2,7e−5 ± 4,8e−6**
SEA_G	1,4e−4 ± 8,4e−5	3,7e−4 ± 1,8e−4	3,3e−4 ± 7,7e−5	**2,7e−5 ± 4,9e−6**
WAVEFORM	9,3e−4 ± 5,6e−4	2,2e−3 ± 1,1e−3	2,2e−3 ± 5,6e−4	**2,2e−4 ± 9,1e−5**
WAVEFORM_DRIFT	8,8e−4 ± 4,8e−4	2,2e−3 ± 1,1e−3	2,2e−3 ± 5,8e−4	**2,0e−4 ± 6,4e−5**
Rank médio	2,00	3,53	3,47	**1,00**

Valores em **negrito** indicam o melhor resultado por *datasets*

Figura 13 – Gráfico de distância crítica para os valores obtidos na métrica custo do modelo nos *datasets* sintéticos



O comportamento observado no uso de memória foi semelhante ao considerar os *datasets* sintéticos, conforme ilustrado na Tabela 11 e no gráfico de distância crítica da Figura 13.

Conforme mencionado em parágrafos anteriores, a estratégia de amostragem variável das configurações selecionadas emprega o valor da constante de *Poisson* igual a 10 para os classificadores não selecionados e o valor 1 para os selecionados, resultando em um maior treinamento na proporção de classificadores não selecionados. Assim, o crescimento das árvores é diretamente influenciado pela proporção de classificadores selecionados e não selecionados. Dessa forma, quando poucos classificadores são selecionados, ocorre um aumento no treinamento. A Tabela 12 apresenta a quantidade média de classificadores selecionados em cada dataset real. Em alguns casos extremos, como na base *kdd99*, foi selecionado, em média, próximo de um classificador, impactando significativamente a complexidade das árvores e o uso de memória, como pode ser observado na Tabela 10.

Tabela 12 – Quantidade média de classificadores selecionados nos *datasets* reais (membros do conjunto = 100)

	Configurações	
	SW100_S1_NS10	**SW200_S1_NS10**
connect_4	26.22 ± 10.30	25.72 ± 13.31
covertype	29.32 ± 3.57	25.41 ± 6.21
electricity	76.19 ± 19.49	64.55 ± 29.07
gassensor	74.50 ± 3.46	77.93 ± 6.22
gmsc	89.64 ± 8.32	81.14 ± 15.58
kdd99	0.78 ± 0.03	1.27 ± 0.06
kddcup99	99.84 ± 0.14	99.66 ± 0.29
keystroke	88.43 ± 9.99	84.84 ± 17.16
luxembourg	99.79 ± 0.04	99.75 ± 0.09
NOAA	58.52 ± 34.18	49.33 ± 41.70
outdoor	83.14 ± 3.68	77.29 ± 7.75
ozone	79.40 ± 2.78	91.94 ± 4.35
pokerhand	30.29 ± 26.81	45.10 ± 44.64
power_supply	40.71 ± 48.13	39.91 ± 48.51
Média	62,62	61,70

Resultados semelhantes foram observados com os *datasets* sintéticos. A Tabela 13 demonstra que a proporção de classificadores não selecionados é superior na grande maioria dos *datasets*. Isso indica que a estratégia de seleção é eficaz, identificando adequadamente os classificadores que devem ser empregados para a classificação das instâncias. No entanto, essa abordagem gera um *overhead* de uso de memória quando associada à estratégia seletiva de treinamento.

Tabela 13 – Número médio de classificadores selecionados nos *datasets* sintéticos (membros do conjunto = 100)

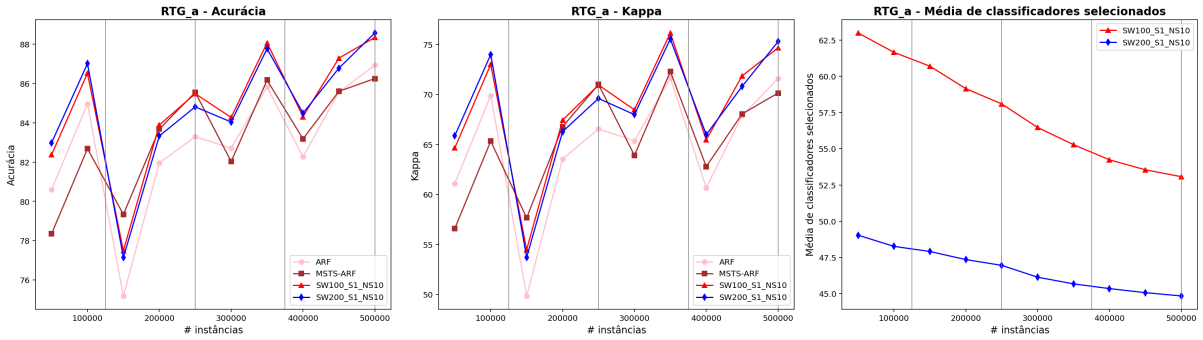
	Configurações	
	SW100_S1_NS10	**SW200_S1_NS10**
AGR_A	32.62 ± 22.09	34.21 ± 31.65
AGR_G	21.22 ± 20.15	24.62 ± 30.73
LED_A	45.37 ± 42.48	42.05 ± 43.45
LED_G	44.62 ± 42.67	41.30 ± 43.21
mixedBalac	91.03 ± 7.36	83.04 ± 13.91
mixedImbalac	88.69 ± 9.33	80.39 ± 16.27
RBF_F	40.82 ± 48.32	41.35 ± 47.89
RBF_M	42.02 ± 47.34	41.96 ± 47.40
RTG	65.87 ± 28.41	48.95 ± 41.84
RTG_A	53.07 ± 39.07	44.84 ± 45.06
RTG_G	45.44 ± 44.57	42.26 ± 47.15
SEA_A	58.61 ± 2.72	48.34 ± 3.58
SEA_G	54.11 ± 2.32	44.28 ± 2.72
WAVEFORM	44.79 ± 45.10	42.62 ± 46.86
WAVEFORM_DRIFT	14.77 ± 9.37	7.06 ± 3.91
Média	49,53	44,48

O aumento do treinamento na maioria dos classificadores promove sua especialização durante o processo de aprendizagem; consequentemente, em cenários onde muitos classificadores não são selecionados, a diversidade no conjunto tende a ser reduzida.

Para uma análise detalhada do comportamento da seleção ao longo do fluxo de dados, foram selecionados quatro *datasets*: dois nos quais foram observados ganhos (*RTG_a* e *AGR_a*) em relação à capacidade preditiva e dois onde não houve ganhos

(*WAVEFORM_DRIFT* e *LED_A*). As Figuras 14, 15, 16 e 17 apresentam a proporção de classificadores selecionados ao longo do treinamento para esses *datasets*. Observa-se que a proporção de selecionados diminui ao longo do tempo.

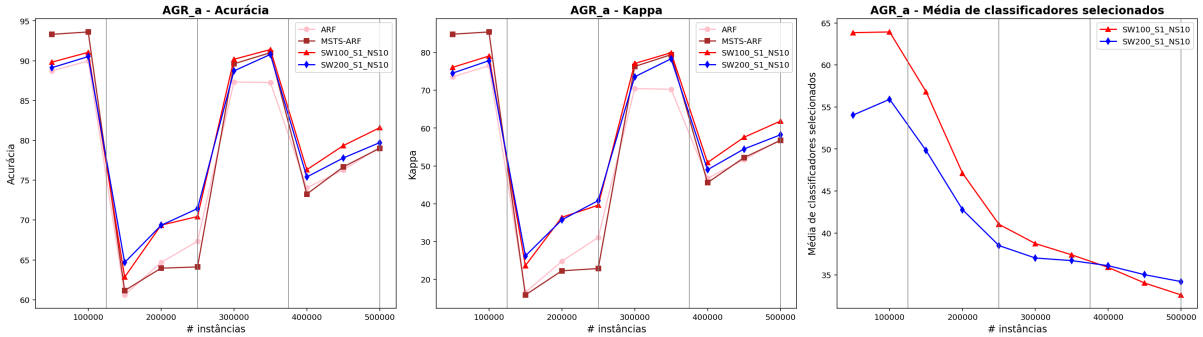
Figura 14 – Acurácia, Kappa e a Proporção de *learners* selecionados no *dataset* RTG_a para as duas melhores configurações. As linhas verticais na cor cinza indicam os desvios de conceitos



Para o dataset *RTG_a*, onde foram observados ganhos, nota-se que ambas as configurações apresentaram curvas similares em termos de acurácia e kappa. No entanto, a configuração com a janela deslizante de valor 100 (SW100_S1_NS10) demonstrou um desempenho superior, conforme evidenciado pelos resultados individuais apresentados nas Tabelas 5 e 7.

Em relação à proporção de selecionados, verifica-se que essa configuração alcançou taxas de seleção mais elevadas em comparação com a janela de valor 200. O impacto dos desvios de conceitos na janela deslizante de maior valor é perceptível não apenas na Figura 14, mas também nas figuras subsequentes. Ao analisar as Tabelas 12 e 13, observa-se um comportamento semelhante para os demais *datasets* utilizados nos experimentos.

Figura 15 – Acurácia, Kappa e Proporção de learners selecionados no dataset AGR_a para as duas melhores configurações. As linhas verticais na cor cinza indicam os desvios de conceitos



Semelhante ao *dataset Agrawal* com desvio de conceito abrupto (Figura 15), a configuração de janela deslizante de valor 100 funcionou melhor durante os períodos de desvios de conceito e com média maior de seleção de classificadores.

Figura 16 – Acurácia, Kappa e Proporção de *learners* selecionados no dataset WAVEFORM_DRIFT para as duas melhores configurações. As linhas verticais na cor cinza indicam os desvios de conceitos

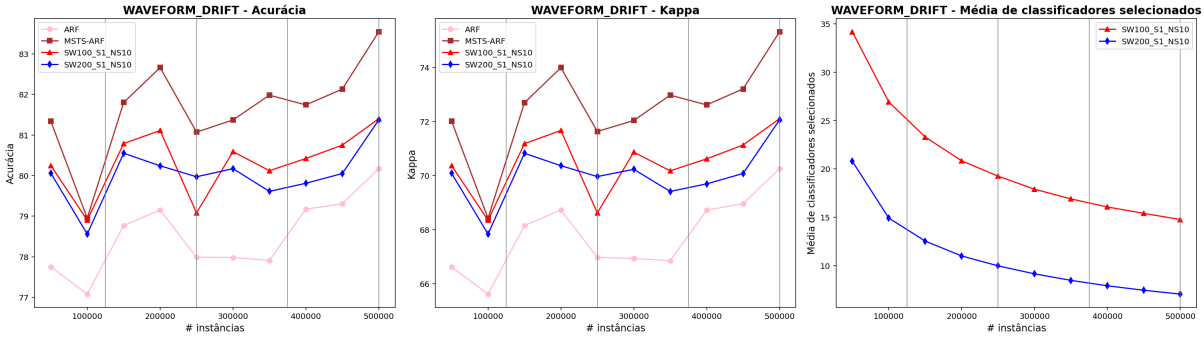
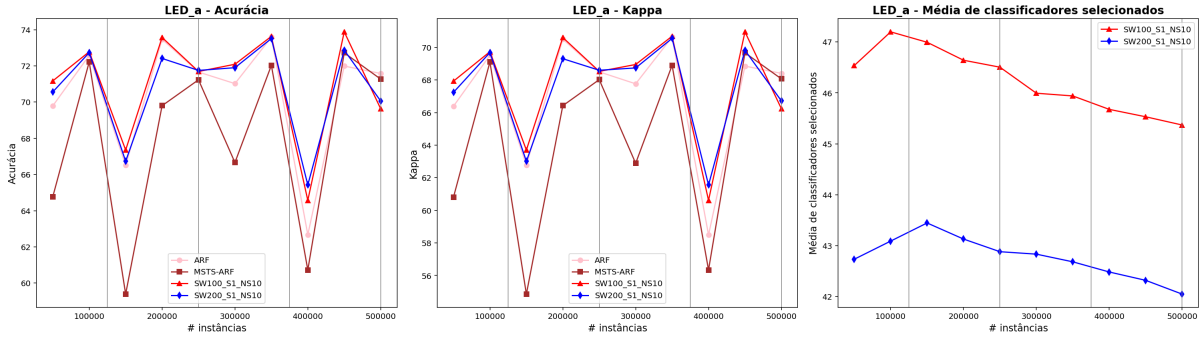


Figura 17 – Acurácia, Kappa e Proporção de *learners* selecionados no *datasets* LED_a para as duas melhores configurações. As linhas verticais na cor cinza indicam os desvios de conceitos



Nos quatro casos apresentados, a configuração da janela de tamanho 100 seleciona um número maior de classificadores em comparação com a configuração de janela longa (200). Isso se deve à maior estabilidade das estatísticas extraídas em janelas mais longas, que são mais confiáveis, reduzindo as diferenças entre o desempenho dos classificadores base e do *ensemble*. Assim, com a seleção de menos classificadores na configuração de janela de 200, há um crescimento mais frequente para uma grande quantidade de árvores, impactando o uso de memória.

5.4 Considerações Finais

Este capítulo apresentou os resultados das duas melhores configurações com a abordagem LGD aplicada ao *Adaptive Random Forest*. De modo geral, observou-se que esta abordagem é competitiva em relação a alguns algoritmos de estado da arte atuais, sendo avaliada pela perspectiva dos resultados de acurácia e da estatística kappa, bem como em relação ao modelo conceitual do oráculo. Isso demonstra que a seleção de classificadores é eficaz e que a redução da amostragem online, com a aplicação de valores menores para o *lambda* nos classificadores selecionados, melhora os resultados dessas métricas.

Em contrapartida, o método demonstrou variação no consumo de recursos computacionais, apresentando melhorias em relação ao *baseline* (algoritmo ARF), mas exibindo resultados ora comparáveis, ora inferiores ao MSTS. De modo geral, o método evidencia um bom equilíbrio entre desempenho preditivo e consumo de recursos computacionais.

A proposta de utilizar uma ampla variedade de *datasets* com comportamentos diversos apresentou um desafio significativo na busca por melhores resultados. No entanto, as configurações selecionadas demonstraram-se eficazes e adequadas para a maioria dos *datasets* empregados.

6 Conclusão

Diversos estudos na literatura têm demonstrado que a aplicação da seleção dinâmica de classificadores resulta em melhorias na performance dos algoritmos de aprendizagem de máquina que utilizam conjuntos de classificadores. Neste estudo, a *dominância local-global*, aplicada ao ARF juntamente com a abordagem de adaptação da constante de *Poisson*, foi empregada com o objetivo de aprimorar a performance do algoritmo original e verificar se os resultados são competitivos em relação a outras técnicas de ponta.

Com os resultados apresentados, constatou-se que essa abordagem pode ser considerada agressiva. Por exemplo, na maioria das bases sintéticas utilizadas, a média de classificadores selecionados ficou abaixo de 50%, enquanto nas bases reais, essa média foi mais elevada, em torno de 60%. Isso evidencia que o critério de seleção é eficaz, pois aprimora a capacidade preditiva do *ensemble*.

A variação do valor da constante de *Poisson* possibilitou a concretização do segundo objetivo do estudo, demonstrando que, apesar da redução da diversidade do *ensemble* devido às múltiplas atualizações dos modelos não selecionados, ainda há ganhos de desempenho (tempo e memória) em relação ao algoritmo original. Portanto, é possível afirmar que a hipótese levantada sobre os ganhos de performance preditiva e desempenho computacional foi confirmada pelos resultados apresentados.

Como contribuição, é igualmente viável correlacionar a diversidade dos experimentos realizados, o que possibilitou a avaliação da generalidade das abordagens propostas em diferentes problemas. As análises indicam que, embora os resultados variem entre os *datasets*, existem configurações que obtêm resultados satisfatórios para a maioria deles, aproximando-se inclusive do oráculo.

6.1 Trabalhos Futuros

Uma extensão natural deste trabalho seria aplicar a abordagem de seleção proposta a outros algoritmos de *ensemble*, como o *Streaming Random Patches* (SRP), *Adaptive Regularized Ensemble* (ARE), *Adaptive Random Tree Ensemble* (ARTE) e *Dynamic ensemble selection based on windows over imbalanced drift data stream* (DESW-ID), investigando o impacto da seleção ativa de classificadores em cada um deles.

Outra possível melhoria no método consiste em tornar dinâmica a variação dos valores da constante de *Poisson*, considerando a proporção de classificadores selecionados. Os resultados obtidos indicam que quanto mais treinamento é aplicado na proporção de classificadores não selecionados, melhor é o desempenho preditivo alcançado. Este estudo

poderia, assim, substituir os valores fixos apresentados na Tabela 3, buscando uma variação dinâmica da estratégia de seleção que seria mais adaptável e mais eficaz em diferentes tipos de fluxos de dados que sofrem diferentes tipos de mudanças.

Uma terceira direção de pesquisa derivada do presente trabalho seria a análise e adaptação dos algoritmos propostos para permitir a seleção dinâmica de conjuntos de classificadores heterogêneos, considerando que eles podem variar significativamente em comportamento ao longo do fluxo, gerando novas observações e relações entre a quantidade de treinamento e a estratégia de seleção.

Referências

AGRAWAL, R.; IMIELINSKI, T.; SWANI, A. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Enginnering*, n. 5, p. 914–925, 1993. Citado 2 vezes nas páginas 69 e 95.

ALBUQUERQUE, R. A. S. et al. A decision-based dynamic ensemble selection for concept drift. 2019. Citado 2 vezes nas páginas 24 e 49.

ALMEIDA, P. R. L. de et al. Handling concept drifts using dynamic selectin of classifiers. *IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, n. 1, p. 989–995, 2016. Citado 3 vezes nas páginas 24, 48 e 49.

ARANGO, S. P. et al. *Dynamic Post-Hoc Neural Ensemblers*. 2024. Disponível em: <https://arxiv.org/abs/2410.04520>. Citado na página 47.

ASSIS, D. N.; ENEMBRECK, F.; BARDDAL, J. P. Mass-based short term selection of classifiers in data streams. *2023 International Joint Conference on Neural networks*, p. 1–8, 2023. Citado na página 53.

ASUNCIóN, A.; NEWMAN, J. D. *UCI Machine Learning Repository*, 2007. Citado 2 vezes nas páginas 69 e 96.

BAENA-GARCíA, M. et al. Early drift detectin method. *ECML PKDD 2006 workshop on knowledge discorety from data streams*, 2006. Citado 4 vezes nas páginas 34, 49, 69 e 96.

BARDDAL, J. P. et al. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, v. 127, p. 278–294, maio 2017. ISSN 01641212. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0164121216301030>. Citado na página 34.

BARROS, R. S. et al. Rddm: Reactive drift detection method. *Expert Systems with Applications*, v. 90, p. 344–355, 2017. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417417305614>. Citado na página 34.

BARROS, R. S. M. de; SANTOS, S. G. T. de C.; JúNIOR, P. M. G. A bossting-lie online learning ensemble. *2016 International Joint Conference on Neural Networks (IJCNN)*, p. 1871–1878, 2016. Citado na página 53.

BIFET, A.; GAVALDà, R. Learning from time-changing data with adaptive windowing. *Proceedings of the Seventh SIAM International Conference on Data Mining*, v. 1, n. 26-28, p. 443–448, 2007. Citado 2 vezes nas páginas 34 e 37.

BIFET, A. et al. *MOA: Massive Online analysis, a Framework for Stream Classificatin and Clustering*. [S.l.], 2010. Citado 2 vezes nas páginas 30 e 67.

BIFET, A. et al. *Machine Learning for Data Streams with Practical Examples in MOA*. [S.l.], 2018. Citado na página 29.

BIFET, A.; KIRKBY, R. *Data Stream Mining a pratical approach*. [S.l.], 2019. Citado 2 vezes nas páginas 27 e 28.

BISHOP, C. M. *Pattern recognition and machine learning*. 1. ed. Berlin, Heidelbert: Springer, 2006. Acesso em: 13 jun 2024. Citado 2 vezes nas páginas 28 e 31.

BLACKARD, J. A.; DEAN, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computer and Electronics in Agriculture*, v. 24, n. 3, p. 131–151, 1999. Citado 2 vezes nas páginas 69 e 97.

BREIMAN, L. *Bagging predictors*. 1. ed. [S.l.]: Springer, 1996. Acesso em: 09 jul 2024. Citado na página 35.

BREIMAN, L. *Random forests*. 1. ed. [S.l.]: Springer, 2001. Acesso em: 04 jul 2024. Citado 2 vezes nas páginas 23 e 36.

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: CRC Press, 1984. Acesso em: 09 ago 2024. Citado 2 vezes nas páginas 69 e 95.

BRITTO, A. S.; SABOURIN, R.; OLIVEIRA, L. E. Ensemble learning for data stream analysis: A survey. *Pattern Recognition*, v. 1, n. 47, p. 3665–3680, 2014. Citado 4 vezes nas páginas 23, 42, 48 e 68.

BRZEZINSKI, D.; STEFANOWSKI, J. Ensemble classifiers for imbalanced and evolving data streams. In: _____. *Data Mining in Time Series and Streaming Databases*. [s.n.], 2018. cap. Chapter 3, p. 44–68. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/9789813228047_0003>. Citado na página 29.

BURKOV, A. *The Hundred-page Machine Learning Book*. [S.l.: s.n.], 2019. Citado na página 27.

CANDILLIER, L.; LEMAIRE, V. Design and analysis of the nomao challenge active learning in the real-world. In: . [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 69 e 97.

CATTRAL, R.; OPPACHER, F. *Poker Hand*. 2002. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5KW38. Citado 2 vezes nas páginas 69 e 98.

CAVALHEIRO, L. P. et al. Dynamically selected ensemble for data stream classification. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2021. p. 1–7. Citado 5 vezes nas páginas 27, 30, 35, 41 e 50.

CODECA, L.; FRANK, R.; ENGEL, T. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In: *2015 IEEE Vehicular Networking Conference (VNC)*. [S.l.: s.n.], 2015. p. 1–8. Citado 2 vezes nas páginas 69 e 98.

COHEN, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, v. 20, n. 1, p. 37–46, 1960. Disponível em: <https://doi.org/10.1177/001316446002000104>. Citado 2 vezes nas páginas 29 e 59.

CORDEIRO, P.; CAVALCANTI, G.; CRUZ, R. Dynamic ensemble algorithm post-selection using hardness-aware oracle. *IEEE Access*, PP, p. 1–1, 01 2023. Citado na página 44.

CRUZ, R. M. O. et al. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, v. 48, n. 1, 2015. Citado na página 44.

DAWID, A. P. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, [Royal Statistical Society, Oxford University Press], v. 147, n. 2, p. 278–292, 1984. ISSN 00359238, 23972327. Disponível em: <http://www.jstor.org/stable/2981683>. Citado na página 30.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, JMLR.org, v. 7, p. 1–30, dez. 2006. ISSN 1532-4435. Citado 2 vezes nas páginas 67 e 71.

DIETTERICH, T. G. Ensemble methods in machine learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, v. 1, n. 1, p. 1–15, 2000. Citado na página 32.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, v. 1, n. 1, p. 71–80, 2000. Disponível em: <https://dl.acm.org/doi/10.1145/347090.347107>. Citado 3 vezes nas páginas 31, 69 e 96.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. *Neural Networks, IEEE Transactions on*, v. 22, p. 1517 – 1531, 11 2011. Citado 2 vezes nas páginas 69 e 97.

FIX, E.; HODGES, J. L. *Discriminatory analysis. nonparametric discrimination: Consistency properties.* 3. ed. [S.l.]: International Statistical Review, 1989. Acesso em: 30 jul 2024. Citado na página 43.

FRíAS-BLANCO, I. et al. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, v. 27, n. 3, p. 810–823, 2015. Citado na página 34.

FUSION, W. C. C. *Give Me Some Credit.* Kaggle, 2011. Disponível em: <https://kaggle.com/competitions/GiveMeSomeCredit>. Citado 2 vezes nas páginas 69 e 97.

GAMA, J.; GABER, M. M. *Learning from Data Streams: Processing Techniques in Sensor Networks.* 1. ed. [S.l.]: Spring Natrue, 2007. ISBN 3540736786. Citado na página 29.

GAMA, J.; MEDAS, G. C. P.; RODRIGUES, P. Learning with drift detection. *Brazilian Symposium on Artificial Intelligence*, n. 41, p. 1286–295, 2004. Citado 3 vezes nas páginas 34, 69 e 95.

GAMA, J.; SEBASTIAO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Mach. Learn.*, Kluwer Academic Publishers, USA, v. 90, n. 3, p. 317–346, mar. 2013. ISSN 0885-6125. Disponível em: <https://doi.org/10.1007/s10994-012-5320-9>. Citado na página 68.

GAMA, J. et al. A survey on concept drift adaptation. *ACM Computing Surveys*, v. 46, n. 4, p. 1–37, abr. 2014. ISSN 0360-0300, 1557-7341. Disponível em: <https://dl.acm.org/doi/10.1145/2523813>. Citado 3 vezes nas páginas 33, 34 e 59.

GOMES, H. M. et al. *Adaptive random forests for envolving data stream classification*. [S.l.], 2017. Citado 8 vezes nas páginas 23, 36, 37, 38, 43, 53, 59 e 61.

GOMES, H. M.; READ, J.; BIFET, A. Streaming random patches for evolving data stream classification. *IEEE International Conference on Data mining (ICDM)*, v. 1, n. 1, p. 240–249, 2019. Citado 3 vezes nas páginas 38, 39 e 53.

HAN, M. et al. Dynamic ensemble selection classification algorithm based on window over imbalanced drift data stream. *Knowledge and Information Systems*, v. 65, p. 1105–1128, 2023. Citado na página 52.

HARRIES, M.; WALES. Splice-2 comparative evaluation: Electricity pricing. *Technical report, The University of South Wales*, 1999. Citado 2 vezes nas páginas 69 e 97.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*, p. 97–106, 2001. Citado 4 vezes nas páginas 23, 69, 95 e 96.

IKONOMOVSKA, E.; BIFET, A. *Airlines dataset*. [S.l.], 2009. Disponível em: <https://www.openml.org/search?type=data&sort=runs&id=42493&status=active>. Citado 2 vezes nas páginas 69 e 96.

KO, A. H.; SABOURIN, R.; JR., A. S. B. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, v. 48, n. 5, p. 1718–1731, 2008. Citado 2 vezes nas páginas 42 e 48.

KUNCHEVA, L. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 2, p. 281–286, 2002. Citado na página 73.

LICHMAN, M. *UCI Machine Learning Repository*. 2013. Accessed: 2024-11-03. Disponível em: <http://archive.ics.uci.edu/ml>. Citado 2 vezes nas páginas 69 e 98.

LOSING, V.; HAMMER, B.; WERSING, H. Knn classifier with self adjusting memory for heterogeneous concept drift. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. [S.l.: s.n.], 2016. p. 291–300. Citado 2 vezes nas páginas 69 e 97.

MAXION, R. A.; KILLOURHY, K. S. Keystroke biometrics with number-pad input. In: *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*. [S.l.: s.n.], 2010. p. 201–210. Citado 2 vezes nas páginas 69 e 97.

OZA, N. Online bagging and boosting. In: *2005 IEEE International Conference on Systems, Man and Cybernetics*. [S.l.: s.n.], 2005. v. 3, p. 2340–2345 Vol. 3. Citado 6 vezes nas páginas 23, 35, 36, 37, 38 e 43.

PAGE, E. S. Continuous inspection schemes. *Biometrika*, v. 41, n. 1-2, p. 100–115, 06 1954. ISSN 0006-3444. Disponível em: <https://doi.org/10.1093/biomet/41.1-2.100>. Citado na página 34.

PAIM, A. M.; ENEMBRECK, F. Adaptive random tree ensemble for evolving data stream classification. *Know.-Based Syst.*, Elsevier Science Publishers B. V., NLD, v. 309, n. C, jan. 2025. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2024.112830>. Citado na página 54.

PAIN, A. M.; ENEMBRECK, F. Adaptive regularized ensemble for envolving data stream classification. *Pattern Recognition Letters*, v. 180, p. 55–61, 2024. Citado na página 52.

PINAGé, F.; SANTOS, E. M. dos; GAMA, J. A drift detection method based on dynamic classifier selection. *Data Mining and Knowledge Discovery*, v. 34, p. 50–74, 2019. Citado na página 49.

PéREZ, J. L. M. *Comitê de métodos estatísticos para detecção de mudanças de conceito*. [S.l.], 2018. Disponível em: <https://repositorio.ufpe.br/bitstream/123456789/29990/1/DISSERTA%c3%87%c3%83O%20Jos%c3%a9%20Luis%20Mart%c3%adnez%20P%c3%a9rez.pdf>. Citado na página 23.

SCHILIMMER, J. C.; GRANGER, J. R. H. Incremental learning from noisy data. *Kluver Academic Publishers*, v. 5, n. 1, p. 317–354, 1986. Citado na página 32.

SHAPLEY, L. S. A value for n-person games. In: KUHN, H. W.; TUCKER, A. W. (Ed.). *Contributions to the Theory of Games II*. Princeton, NJ: Princeton University Press, 1953, (Annals of Mathematics Studies, v. 28). p. 307–317. Citado na página 47.

STOLFO, S. et al. Cost-based modeling for fraud and intrusion detection: results from the jam project. In: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*. [S.l.: s.n.], 2000. v. 2, p. 130–144 vol.2. Citado 2 vezes nas páginas 69 e 98.

STREET, N. W.; KIM, Y. S. A streaming ensemble algorithm (sea) for large-scale classification. *Preceedings of the sevnth ACM SIGKDD international conference on Knowledge discorevery and data mining*, p. 377–382, 2001. Citado 2 vezes nas páginas 69 e 96.

TROMP, J. Solving connect-4 on medium board sizes. *ICGA Journal*, v. 31, n. 2, p. 110–112, 1995. Citado 2 vezes nas páginas 69 e 96.

TSYMBAL, A. The problem of concept drift: Definitions and related work. v. 1, n. 5, 2005. Citado na página 33.

VERGARA, A. et al. On the performance of gas sensor arrays in open sampling systems using inhibitory support vector machines. *Sensors and Actuators B: Chemical*, v. 185, p. 462–477, 2013. Citado 2 vezes nas páginas 69 e 97.

WOODS, K.; JR., W. P. K.; BOWYER, K. Combinatin of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Matchine Intelligence*, v. 19, n. 4, p. 405–410, 1997. Citado 2 vezes nas páginas 43 e 44.

XU, C. et al. Meta-learning-based sample discrimination framework for improving dynamic selection of classifiers under label noise. *Know.-Based Syst.*, Elsevier Science Publishers B. V., NLD, v. 295, n. C, jul. 2024. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2024.111811>. Citado na página 45.

YUAN, K. et al. Hydrogen-bonded complex between ozone and thioperoxy radical in gas-phase. *Acta Physico-Chimica Sinica*, Editorial Office of Acta Physico-Chimica Sinica, v. 24, n. 11, p. 2065–2070, 2008. Citado 2 vezes nas páginas 69 e 97.

ZHANG, Z.-L.; ZHU, Y.-H.; LUO, X.-G. Des - a: Dynamic ensemble selection based on algorithm shapley. *Elsevier ScienceDirect Journals*, v. 157, p. 110899, 2024. Citado na página 46.

ZYBLEWSKI, P.; SABOURIN, R.; WOźNIAK, M. Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. *Information Fusion*, v. 66, p. 138–154, 2021. Citado na página 51.

Apêndices

APÊNDICE A – Descrição dos datasets

A.1 Datasets sintéticos

- **Randon Basis Function (RBF)** (HULTEN; SPENCER; DOMINGOS, 2001), esse dataset possui um número pré-definido de centroides, com uma posição central randômica, com seus rótulos e pesos associados a um desvio padrão. Ele possui 5 classes, 10 atributos numéricos e nenhum atributo nominal. Sendo utilizadas duas configurações:

  RBF_f : Configuração com mudança abrupta do deslocamento do centroide.

  RBF_m : Configuração com mudança gradual do deslocamento do centroide.

- **Agrawal (AGR)**, introduzido por (AGRAWAL; IMIELINSKI; SWANI, 1993), simula a situação da aprovação de um empréstimo bancário para um cliente. Ele contém 10 funções de empréstimo para a geração dos dados, com o mapeamento de 2 classes possíveis, 6 atributos numéricos e 3 nominais, com as seguintes configurações:

  AGR_g : Configuração com a alteração de desvio de fluxo com variação gradual.

  AGR_a : Configuração com alterações de desvio de fluxo com variação abrupta.

- **LED**, introduzido por (BREIMAN et al., 1984), ela cria dados para a simulação de dígitos apresentados em um display de LED de 7 segmentos, tendo 10 classes, nenhum atributo numérico e 10 atributos nominais, sendo utilizada nas seguintes configurações:

  LED_a : Simula uma mudança de conceito abrupta.

  LED_g : Simula uma mudança de conceito gradual.

- **Mixed**, apresentado em (GAMA; MEDAS; RODRIGUES, 2004), esse gerador cria dados para a simulação de cenários com desvio de conceito abrupto. A base gerada contém 2 classes, 2 atributos numéricos e 2 atributos nominais. As configurações utilizadas são:

  $MIXED - BALANCED$: Configuração que utiliza classes balanceadas.

  $MIXED - IMBALANCED$: Configuração que utiliza classes desbalanceadas.

- **Random Tree Generator (RTG)**: Esse gerador cria uma árvore de decisão através da seleção aleatória de atributos e da alocação de classes aleatórias em cada nó da folha. Com isso, novas instâncias são geradas através da atribuição de valores

aleatórios distribuídos para cada atributo. A base gerada possui 2 classes, 5 atributos numéricos e 5 atributos nominais (DOMINGOS; HULTEN, 2000). As configurações utilizadas:

RTG_a : Configuração com a mudança de conceito abrupta.

RTG_g : Configuração com a mudança de conceito gradual.

RTG : Configuração original, sem nenhuma mudança de conceito.

- **Rotating Hyperplane (HYPER)**: Gerador apresentado em (HULTEN; SPENCER; DOMINGOS, 2001), utiliza hiperplanos para criar fluxos de dados com mudanças de conceito a partir da rotação e movimentação suave do hiperplano. Permite simular o aumento da velocidade de desvio incremental. Ela possui 2 classes, 10 atributos numéricos e nenhum atributo nominal.

- **SEA**: Introduzido em (STREET; KIM, 2001), esse gerador de fluxos de dados inclui três atributos contínuos, 2 classes e nenhum atributo nominal, permitindo ser customizado.

  SEA_a : Simulação de mudança de conceito abrupta.

  SEA_g : Simulação de conceito gradual.

- **SINE**: Os dados gerados simulam coordenadas x e y em diferentes contextos. O gerador foi apresentado em (BAENA-GARCíA et al., 2006) e os dados utilizados nesse trabalho não utilizam a opção de desvio de conceito, contendo 2 classes, 4 atributos numéricos e nenhum atributo nominal.

- **WAVEFORM**: Esse gerador, apresentado em (ASUNCIóN; NEWMAN, 2007), permite analisar um problema de previsão de três tipos de formas de onda, tendo a mesma origem do conjunto de dados LED. Ela possui 2 classes com 40 atributos numéricos e nenhum nominal.

  $WAVEFORM_g$: Configuração com mudança de conceito gradual.

  $WAVEFORM$: Configuração sem desvio de conceito.

A.2 Datasets Reais

- **Airlines**: Esse dataset foi apresentado em (IKONOMOVSKA; BIFET, 2009) e armazena dados utilizados para se prever a ocorrência de atrasos de voos, contendo um total de 539.383 instâncias, 2 classes, 3 atributos numéricos e 4 atributos nominais.

- **Connect-4**: Introduzido em (TROMP, 1995), apresenta uma visão abrangente de situações que ocorrem no jogo Connect-4, contendo um total de 67.557 instâncias, 2 classes, nenhum atributo numérico e 42 atributos nominais.

- **Forest Cover type (Covertype)**: Apresentado em (BLACKARD; DEAN, 1999), apresenta dados florestais em regiões, obtidos do Sistema de Informação de Recursos da 2ª. Região do serviço florestal dos EUA, informando o tipo de cobertura florestal de cada instância. Ele contém 581.012 instâncias, 7 classes, 10 atributos numéricos e 44 atributos nominais.

- **Electricity**: Dataset apresentado por (HARRIES; WALES, 1999), contém dados do consumo de eletricidade de New South Wales, na Austrália. Contém 45.312 instâncias, com 2 classes e 7 atributos numéricos e 1 atributo nominal.

- **Gas Sensor**: Introduzido em (VERGARA et al., 2013), ele contém dados coletados sobre o nível de concentração de sensores em contato com produtos como amoníaco, acetaldeído, acetona, etileno, etanol e tolueno. Ele contém 13910 instâncias, 6 classes, 128 atributos numéricos e nenhum atributo nominal.

- **Give Me Some Credit (GMSC)**: Esse dataset (FUSION, 2011) apresenta dados utilizados para avaliar a pontuação de crédito para a tarefa de tomada de decisão na concessão de empréstimos, contendo 150.000 instâncias, 2 classes, 10 atributos numéricos e nenhum atributo nominal.

- **Keystroke**: Dataset apresentado em (MAXION; KILLOURHY, 2010) que contém dados sobre o tempo de digitação de 51 datilógrafos, com um total de 1.600 instâncias, 4 classes, 10 atributos numéricos e nenhum atributo nominal.

- **NOOA**: Apresentado em (ELWELL; POLIKAR, 2011), o dataset contém dados de medições meteorológicas diárias coletadas em diversas estações ao redor do mundo. Possui 18.154 instâncias, 2 classes, 8 atributos numéricos e nenhum atributo nominal.

- **Nomao**: Apresentado em (CANDILLIER; LEMAIRE, 2013), esse dataset contém dados sobre um mecanismo de busca que se propõe a buscar informações completas relacionadas a lugares com a agregação de diversas fontes encontradas na internet, como nome, telefone, localização, etc. Apresentando um total de 9.844 instâncias.

- **Outdoor**: Introduzido em (LOSING; HAMMER; WERSING, 2016), os dados contêm imagens capturadas por celular em ambientes externos, sendo essas imagens representadas por histogramas RGB normalizados. O dataset contém um total de 4.000 instâncias, 40 classes, 21 atributos numéricos e nenhum atributo nominal.

- **Ozone**: Esse dataset, apresentado em (YUAN et al., 2008), contém dados de duas séries de dados referentes aos níveis de ozônio no solo, possuindo 2.534 instâncias, 2 classes, 72 atributos numéricos e nenhum atributo nominal.

- **Rialto**: Apresentado por (LOSING; HAMMER; WERSING, 2016), apresenta histogramas coloridos de imagens de edifícios adjacentes à ponte de Rialto em Veneza,

possuindo 82.250 instâncias, 10 classes, 27 atributos numéricos e nenhum atributo nominal.

- **KDD99 e KDD99CUP**: São conjuntos de dados utilizados para detecção de invasões em redes de computadores, apresentados em (STOLFO et al., 2000) para a competição KDD Cup de 1999. Ambas apresentam os mesmos atributos, mas se diferenciam pela quantidade de instâncias, sendo a base KDD99Cup uma amostra de 10% da base KDD99 original.

- **Luxembourg**: Apresentada em (CODECA; FRANK; ENGEL, 2015), foi utilizada para cenários em que simulam 24 horas do monitoramento realista do tráfego na cidade de Luxemburgo. A base possui 1.905 instâncias, 2 classes, 16 atributos numéricos e 15 atributos nominais.

- **Pokerhand**: Base de dados que propõe a simulação de jogadas do popular jogo de baralho, onde cada instância representa um exemplo de mão com cinco cartas de um baralho com um total de 52 cartas. Apresentada em (CATTRAL; OPPACHER, 2002), a base possui um total de 829.201 instâncias, 10 classes, 5 atributos numéricos e 5 atributos nominais.

- **Powersupply**: Conjunto de dados utilizado em problemas relacionados à previsão e falhas de sistemas de fornecimento de energia, ele contém 29.928 instâncias e foi apresentado em (LICHMAN, 2013), contendo 24 classes, 2 atributos numéricos e nenhum atributo nominal.