UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Juan Isidro González Hidalgo

Ajuste Dinâmico de Parâmetros: Enfoques, Estratégias e Experimentações Aplicadas na Aprendizagem em Fluxos de Dados com Mudanças de Conceitos.

Recife

2022

Juan Isidro González Hidalgo

Ajuste Dinâmico de Parâmetros: Enfoques, Estratégias e Experimentações Aplicadas na Aprendizagem em Fluxos de Dados com Mudanças de Conceitos.

Tese de Doutorado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador: Prof. Dr. Roberto Souto Maior de Barros

Coorientador: Dr. Silas Garrido Teixeira de Carvalho Santos

Recife

2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

G643a González Hidalgo, Juan Isidro
Ajuste dinâmico de parâmetros: enfoques, estratégias e experimentações aplicadas na aprendizagem em fluxos de dados com mudanças de conceitos / Juan Isidro González Hidalgo. – 2022.
142 f.: il., fig., tab.

Orientador: Roberto Souto Maior de Barros.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências e apêndice.

1. Inteligência computacional. 2. Detecção de mudança de conceito. 3. Comitê. 4. Aprendizado online. 5. Fluxo de dados. I. Barros, Roberto Souto Maior de (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2023 – 006

Juan Isidro González Hidalgo

“Ajuste Dinâmico de Parâmetros: Enfoques, Estratégias e Experimentações Aplicadas na Aprendizagem em Fluxos de Dados com Mudanças de Conceitos”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 28/09/2022.

_____

Orientador: Prof. Dr. Roberto Souto Maior de Barros

BANCA EXAMINADORA

_____

Prof. Dr. Cleber Zanchettin
Centro de Informática / UFPE

_____

Prof. Dr. Germano Crispim Vasconcelos
Centro de Informática / UFPE

_____

Prof. Dr. Paulo Maurício Gonçalves Júnior
Instituto Federal de Pernambuco / Campus Recife

_____

Prof. Dr. Francisco Madeiro Bernardino Junior
Escola UNICAP ICAM-TECH / UNICAP

_____

Prof. Dr. João Roberto Bertini Junior
Faculdade de Tecnologia / UNICAMP

"Dedico esta tese de doutorado em especial ao meu filho Theo Gabriel González Terto e a todas as pessoas que me apoiaram de forma incondicional para lograr o sonho de me tornar Doutor em Ciência da Computação."

AGRADECIMENTOS

Agradeço a toda minha família pela força, ajuda e os bons conselhos que sempre me deram em todos os momentos da minha vida estudantil e profissional.

Ao meu orientador, Professor Roberto Souto Maior de Barros, pelos aprendizados, experiências, por acreditar em mim e me dar o voto de confiança em todos os momentos que passamos durante os meus estudos de mestrado e doutorado. Muito obrigado senhor, sou muito grato a você.

Agradeço ao meu coorientador Silas Garrido Teixeira de Carvalho Santos, pela parceria, amizade e sobretudo pelos ensinamentos, especialmente durante o período do doutorado.

Agradeço ao meu amigo Bruno Maciel pela motivação, ajuda e colaboração durante todos esse anos de investigação.

Ao time de desenvolvedores da Foco Aluguel de Carros. Muito obrigado a todos vocês meus colegas de trabalho pela parceria e suporte em todo momento.

Agradeço a todos os professores do programa de pós-graduação do Centro de Informática (CIn-UFPE), em específico aqueles que tive contato na sala de aula.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de estudos durante a maior parte do Doutorado.

"Acredite que você pode, assim você já está no meio do caminho andado" (PEDROSA, 2015).

RESUMO

O processo de trabalho com fluxos de dados exige novas demandas e tarefas desafiadoras na área de mineração de dados e aprendizagem de máquina. Esse fluxo pode ser categorizado como um sistema que gera muitos dados ao longo do tempo. Dessa forma, quando a distribuição de probabilidade dentro desse fluxo varia, estamos com um problema comumente conhecido como mudança de conceito (*Concept Drift*). O processo de implementação de novos métodos para lidar com fluxos de dados contendo mudanças de conceito requer algoritmos que sejam capazes de se adaptar a diferentes situações para, assim, melhorar sua performance. Nesse sentido, o ajuste dinâmico de parâmetros é um contexto pouco explorado nas implementações e experimentações das pesquisas da área, requerendo uma especial atenção, sobretudo para que estes métodos consigam se adaptar melhor aos diferentes ambientes onde são aplicados. Nesta pesquisa são propostos vários enfoques e estratégias para ajustar parâmetros de forma dinâmica em vários algoritmos de classificação existentes. Desse modo, primeiramente são apresentadas várias versões do Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN, PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$), um novo método de classificação em par que utiliza diferentes procedimentos para ajustar de forma dinâmica e incremental o número de vizinhos k. Todas as versões são aplicados ao processo de aprendizagem online em fluxo de dados com mudanças de conceitos. A outra proposta desta tese é o Parameter Estimation Procedure (PEP), um método genérico para o ajuste dinâmico de parâmetros que é aplicado ao parâmetro de diversidade λ (lambda), comum a vários comitês de classificadores utilizados na área. Com essa finalidade, o método proposto (PEP) foi utilizado para criar versões alternativas de três comitês já existentes: BOLE-PE, OABM1-PE e OzaBag-PE. Para validá-los, foram realizados experimentos com conjuntos de dados artificiais e reais e os resultados foram avaliados usando a métrica de acurácia e o teste de Friedman com o pós-teste de Nemenyi. Os resultados dos testes com PL-kNN e suas versões mostram que estas contribuições melhoraram o desempenho do K-Nearest Neighbors (k-NN) com valores fixos de k na maior parte dos cenários testados em termos de acurácia. Já os resultados das versões usando PEP evidenciaram que a estimação dinâmica do λ é capaz de produzir bons resultados de acurácia na maioria dos ambientes experimentados.

Palavras-chaves: detecção de mudança de conceito; comitê; aprendizado online; fluxo de dados; classificação; parametrização dinâmica.

ABSTRACT

The working process for dealing with data streams requires new demands and challenging tasks in the area of data mining and machine learning. A stream can be categorized as a system that generates huge amounts of data. Thus, when the probability distribution within this data stream varies, we have a problem commonly known as *Concept Drift*. The process of implementation of new methods to deal with data streams containing concept drifts requires algorithms that can adapt to different situations to improve their performance. In this sense, the dynamic adjustment of parameters is a seldon explored context in the research implementations and experiments in the area, requiring special attention, especially to permit these methods to better adapt to the different environments where they are applied. In this research, several approaches and strategies are proposed to dynamically adjust parameters in several existing classification algorithms. In this way, several versions of Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN, PL-kNN$_2$, PL-kNN$_3$ and PL-kNN$_4$) first are presented. They are variations of a new pairing method that uses different procedures to dynamically and incrementally adjust the number of neighbors k. All versions are applied to the online learning process in data streams with concept drifts. The other proposal of this thesis is the Parameter Estimation Procedure (PEP), a generic method for the dynamic adjustment of parameters that is applied to the diversity parameter λ (lambda), common to several ensemble classifiers used in the area. For this purpose, the proposed estimation method (PEP) was used to create alternative versions of three existing ensembles: Boosting-like Online Learning Ensemble with Parameter Estimation (BOLE-PE), Online AdaBoost-based M1 with Parameter Estimation (OABM1-PE) and Oza and Russell's Online Bagging with Parameter Estimation (OzaBag-PE). To validate them, experiments were performed with artificial and real-world datasets and the results were evaluated using the accuracy metric and the Friedman test with the Nemenyi post-hoc test. The results of the tests with PL-kNN and its versions show that these contributions improved the performance of k-NN with fixed values of k in most tested scenarios in terms of accuracy. The results of the versions using PEP (BOLE-PE, OABM1-PE and OzaBag-PE) showed that the dynamic estimation of λ is capable of producing good accuracy results in most tested environments.

Keywords: concept drift detection; ensemble; on-line learning; classification; data stream; dynamic parameterization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AdaBoost.M1	Adaptive Boosting for Multiclass Learning 1
ADOB	Adaptable Diversity-based Online Boosting
Adwin	Adaptive Windowing
Adwin OzaBoost	Adwin Online Boosting
BOLE	Boosting-like Online Learning Ensemble
BOLE-PE	Boosting-like Online Learning Ensemble with Parameter Estimation
CSDD	Cosine Similarity Drift Detector
CTs	Clustering Trees
DDM	Drift Detection Method
FTDD	Fisher Test Drift Detector
HDDM	Hoeffding-based Drift Detection Method
HDDM$_A$	Hoeffding-based Drift Detection Method A-test
HT	Hoeffding Tree
k-NN	K-Nearest Neighbors
LevBag	Leveraging Bagging
MLSC	Multi-Label Stream Classification
MOA	Massive Online Analysis
OABM1	Online AdaBoost-based M1
OABM1-PE	Online AdaBoost-based M1 with Parameter Estimation
OzaBag	Oza and Russell's Online Bagging
OzaBag-PE	Oza and Russell's Online Bagging with Parameter Estimation
OzaBoost	Oza and Russell's Online Boosting
PEP	Parameter Estimation Procedure
PL-kNN$_4$	Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 4

PL-kNN$_3$	Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 3
PL-kNN$_2$	Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 2
PL-kNN	Paired k-NN Learners with Dynamically Adjusted Number of Neighbors
RDDM	Reactive Drift Detection Method
SAM-kNN	Self Adjusting Memory Model for k-NN
STEPD	Statistical Test of Equal Proportions
USDD	Ultimately Simple Drift Detector
WSTD	Wilcoxon Rank Sum Test Drift Detector

LISTA DE SÍMBOLOS

k	Número de vizinhos
k	Instâncias mais próximas selecionadas
Λ	Parâmetro de diversidade de diferentes comitês de classificação
$f(x) = y$	Função f que tem como entrada e saída os conjuntos x e y, nessa ordem.
x	Conjunto de atributos
y_i	Valor pertencente a classe Y
$\in$	Pertence
S	Conjunto de instâncias de treinamento
p	Classificador arbitrário
W	Buffer
δ	Delta
ε	Epsilon
$\bar{r}$	Valor médio
n	Observações
n	Parâmetro que determina o número mínimo de instâncias
R	Limite utilizado na equação de Hoeffding bound
R	Conjunto de números reais
B	Função arbitrária B
ω	Elemento omega
Ω	Espaço amostral
N	Conjunto de números naturais
p_i	Taxa de erro

s_i	Desvio padrão
w_r	Número de erros da janela recente
r_r	Número de acertos da janela recente
w_p	Número de predições corretas da janela antiga
r_p	Número de predições incorretas da janela antiga
$H(x)$	Comitê de classificadores
K	Variável aleatória
π	Constante Pi
Φ	Phi
Ξ	Xi
Θ	Teta

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

Na era da informação, muitos sistemas geram uma grande quantidade de dados ao longo do tempo, o que é conhecido como fluxo contínuo de dados (*Data Streams*). A abordagem tradicional de mineração de dados pode permitir que grandes conjuntos de dados sejam usados em computadores modernos, mas ainda não resolve o problema desse fornecimento contínuo de dados. Aplicações reais como monitoramento de tráfego de rede, detecção de fraudes em cartões de crédito, fluxo de informações em smartphones, redes sociais, entre outros, são responsáveis pela geração de grandes quantidades de dados. Normalmente, um modelo que foi induzido anteriormente não pode ser atualizado quando novas informações chegam. Em vez disso, todo o processo de treinamento deve ser repetido com os novos exemplos incluídos. Existem situações, entretanto, em que esta limitação é desfavorável e provavelmente ineficiente (WU; LI; HU, 2012; SIVARAJAH et al., 2017). Desse modo, o paradigma do fluxo de dados surgiu recentemente em resposta ao problema de dados contínuos.

A ideia do processamento de fluxo de dados parte do pressuposto que os exemplos de treinamento podem ser inspecionados brevemente uma única vez, ou seja, eles chegam em alta velocidade e devem ser descartados (mais cedo ou mais tarde) para abrir espaço aos exemplos subsequentes. Além disso, o algoritmo que processa o fluxo não tem controle sobre a ordem dos exemplos vistos e deve atualizar seu modelo de forma incremental conforme cada exemplo é inspecionado (BIFET; KIRKBY, 2009).

O processo de trabalho com fluxos de dados exige novas demandas e tarefas desafiadoras na área de mineração e aprendizagem de máquina. Requerimentos ou restrições como utilizar um pequeno tempo constante e/ou uma quantidade fixa de memória, independentemente do número total de exemplos, e construir um modelo de decisão usando uma única exploração nos dados de treinamento são algumas das características indicadas aos sistemas de aprendizagem para extrair informação de forma eficiente em fluxos de dados contínuos com grandes volumes (HULTEN; SPENCER; DOMINGOS, 2001).

Um exemplo atual onde esses sistemas inteligentes lidam com grandes volumes de dados pode ser encontrado nas plataformas de streaming de vídeo, como Netflix, Youtube, Amazon Prime, etc. Nessa caso, os algoritmos podem inferir a partir das informações geradas pelos usuários em tempo real e utilizá-las, por exemplo, para sugerir algum tipo de conteúdo

específico (SANTOS, 2019).

Da mesma forma, o número de ataques cibernéticos aumenta a cada ano de maneira enorme. Assim, o monitoramento e o controle desses dados para detectar atividades e comportamentos anômalos, fora do controle, têm especial relevância. Algumas soluções propostas visam coletar eventos de segurança, analisá-los, avaliar o risco que eles trazem e informar o administrador sobre eles, a fim de tomar a decisão apropriada para mitigar o problema na segurança potencial (HERMANOWSKI, 2015).

Para fins de exemplificação, considere uma rede de computadores de uma empresa, que é protegida e monitorada por um sistema de segurança OSSIM (COPPOLINO et al., 2011). Trata-se de um sistema de gerenciamento de informações e eventos de segurança de código aberto, integrando seleção de ferramentas gratuitas projetadas para suportar administradores de rede em segurança de computadores, detecção e prevenção de intrusão (HERMANOWSKI, 2015). Um de seus sensores, responsável por coletar registros, monitorar pacotes de rede e varrer hosts, mostra um desempenho irregular relatando alguns alarmes que podem supor uma violação da segurança da rede. Tal circunstância é frequentemente formulada como uma tarefa de detecção, onde o ponto que mostrou a atividade suspeita deve ser sinalizado.

Outro exemplo do mundo real está relacionado às mudanças nas preferências de um usuário em relação às notícias online, onde as notícias recebidas sobre imóveis são classificadas como *Relevante* ou *Não Relevante*. Supondo que o usuário esteja disposto a comprar uma nova casa/apartamento, qualquer notícia sobre domicílios seria *Relevante* e o resto seria *Não Relevante*. Sempre que o usuário realmente compre uma casa, seus interesses provavelmente mudariam e as residências se tornariam *Não Relevante*. A partir desse ponto, talvez as casas de férias se tornem *Relevante*. Esta mudança de preferência é outro exemplo inerente à tarefa de detecção (MAHDI et al., 2020).

No aprendizado de máquina, problemas similares são comumente estudados como mudanças de conceito (*Concept Drift*). Neste sentido, os dados não estacionários desempenham um papel fundamental nos ambientes dinâmicos especialmente quando a distribuição de probabilidade dos dados muda ao longo do tempo (BRZEZIŃSKI, 2010). Uma das categorizações das mudanças de conceito refere-se à velocidade de mudança de um conceito para outro no fluxo de dados, que pode ser abrupta ou gradual (MINKU; WHITE; YAO, 2010). Outra acontece no contexto onde tais mudanças provocam reações nas distribuições dos dados e que são nomeadas reais ou virtuais (WIDMER; KUBAT, 1993; GONÇALVES JR.; BARROS, 2013).

Em geral, as abordagens para lidar com as mudanças de conceito podem ser classificadas

em duas categorias: (i) abordagens que adaptam um classificador em intervalos regulares sem considerar se as mudanças realmente ocorreram; (ii) abordagens que primeiro detectam mudanças de conceito e, em seguida, o classificador é adaptado a essas mudanças (GAMA et al., 2004a). Estes últimos detectam explicitamente mudanças usando mecanismos de acionamento (*drift detection*) para situações como as que foram expostas anteriormente. Desse modo, as variações de mudanças de conceito alteram os resultados dos classificadores ao longo do tempo, fazendo com que os modelos se tornem obsoletos e menos precisos no processo de aprendizagem. A detecção dessas mudanças de conceito (BARROS; SANTOS, 2018) ajuda a atualizar o classificador de forma eficaz e, portanto, obter bons resultados de acurácia na classificação (GABER; ZASLAVSKY; KRISHNASWAMY, 2007).

Além disso, na Inteligencia Artificial existem os métodos de aprendizagem preguiçosa (*Lazy Learning*) (ZHENG; WEBB, 2000; ZHANG; ZHOU, 2007; AHA, 2013) e aprendizagem ansiosa (*Eager Learning*) (SOLOMATINE; MASKEY; SHRESTHA, 2006; DONG; CHENG; SHANG, 2012; WEI, 2015). Os algoritmos de aprendizagem preguiçosa como o K-Nearest Neighbors (k-NN) tardam o processamento de suas entradas até receber o pedido de informações, ou seja, armazenam as entradas para utilizar posteriormente. Em seguida, respondem ao pedido/solicitação de informações utilizando esses dados armazenados (treinamento). Por último, eles descartam a resposta construída e os resultados intermediários. Estas particularidades tornam esses métodos diferentes de outros algoritmos de aprendizado, como os de aprendizagem ansiosa.

Os algoritmos ansiosos selecionam suas entradas gananciosamente mediante uma descrição de conceito intencional, ou seja, esse processo pode ser representado por um conjunto de regras, árvore de decisão ou rede neural, onde as entradas são descartadas. Eles respondem ao pedido de informações usando esse procedimento a priori e o retem para fazer futuras solicitações. Embora o custo computacional dos algoritmos preguiçosos seja menor do que o custo computacional dos métodos ansiosos na etapa de treinamento, os requisitos para armazenar os dados nos algoritmos preguiçosos são maiores e, geralmente, na etapa de responder aos pedidos de informação, o custo computacional do algoritmo preguiçoso é mais alto (SIMOUDIS; AHA, 1997).

Ainda sobre os algoritmos preguiçosos (lazy), eles oferecem uma perspectiva alternativa sobre como resolver tarefas de aprendizagem. Nesse sentido, tais métodos costumam usar abordagens locais (BOTTOU; VAPNIK, 1992), que geram um comportamento altamente adaptativo geralmente não encontrado em algoritmos ansiosos, e os algoritmos preguiçosos são a base de muitas aplicações industriais (NGUYEN; CZERWINSKI; LEE, 1993). Uma média a priori

de dados de entrada, resultados intermediários de armazenamento em cache (por exemplo, configurações de parâmetros para uma função específica) e o acompanhamento de feedback de desempenho para modificar o comportamento de aprendizagem são algumas das tendências que surgiram para diminuir os custos computacionais, aumentar o escopo de aplicabilidade, ou melhorar alguma outra medida de desempenho dessas abordagens (SIMOUDIS; AHA, 1997).

Diversas extensões dos algoritmos k-NN surgiram em tópicos de interesse particular na pesquisa de aprendizado de máquina. Investigações sobre como melhorar o k-NN em ambientes de conceitos e mudança de amostra (sample shift); caracterizar situações em que discretizar recursos contínuos aumenta a acurácia preditiva; e avaliar um algoritmo que seleciona de forma inteligente os k vizinhos mais próximos de uma consulta são algumas das contribuições onde estas extensões de algoritmos com suas diferentes peculiaridades fizeram parte (ATKESON; MOORE; SCHAAL, 1997; SALGANICOFF, 1997; SIMOUDIS; AHA, 1997). Com o passar dos anos, esses aplicativos e extensões do k-NN ganharam maior uso no aprendizado de fluxo de dados com mudanças de conceito. Atualmente, este algoritmo de classificação também é usado em conjunto com detectores de mudanças de conceito (BARROS; SANTOS, 2018) ou como parte de comitês de classificação para melhorar o processo de decisão e o desempenho da aprendizagem. Esses detectores e comitês de classificadores, assim como o algoritmo k-NN, estão disponíveis gratuitamente no Massive Online Analysis (MOA) framework (BIFET et al., 2010), uma estrutura de implementação desenvolvida para lidar com fluxos de dados na área de aprendizado de máquina.

O k-NN tradicional foi criado com base na similaridade de recursos e pertence à família de algoritmos preguiçosos mais simples. Também conhecido como uma abordagem não paramétrica, o algoritmo armazena todos os casos disponíveis e classifica novos casos com base na métrica de similaridade. Para classificar dados vetoriais dentro de um espaço de recurso (dispersão), ele normalmente usa a distância Euclidiana para calcular a distância entre pontos no espaço de recurso (CAI; JI; CAI, 2010). O número de vizinhos k define o ponto de corte para determinar qual é a classe de um objeto vetorial a ser predito dentro do espaço de recursos e selecionar o valor k correto é um processo chamado ajuste de parâmetro e é importante para uma melhor acurácia (SRIVAS, 2019).

Nas avaliações experimentais, o parâmetro k do k-NN pode ser usado com diferentes valores. Este valor é fixo e determinado antes de iniciar o processo de execução do método. Neste trabalho, argumenta-se que um valor diferente de k, menor ou maior, pode ser muito eficaz no resultado final dos experimentos, dependendo do conjunto de dados utilizado. O

principal problema em ter um k fixo é que nem sempre ele é eficiente ou adequado para o cenário em questão. Além disso, por ser um algoritmo simples e usado por vários autores da área, este método foi escolhido nesta pesquisa para aplicar diferentes estratégias e implementar vários métodos baseados no k-NN, ajustando de forma dinâmica o parâmetro k pertencente a este método de classificação e pretendendo melhorar o seu desempenho nos resultados experimentais.

Por outro lado, a aprendizagem usando comitês (*Ensemble Learning*) é uma aplicação muito experimentada dentro das tarefas da classificação. A classificação tenta inferir uma função que mapeia valores de característica em classes de dados de treinamento e aplica a função a dados com rótulos de classes desconhecidas. Essa função é chamada de modelo ou classificador. Visto dessa maneira, um algoritmo de classificação geralmente faz certas suposições sobre o espaço de hipóteses e, assim, define um espaço de hipótese para procurar o modelo correto. O algoritmo também define um determinado critério para medir a qualidade do modelo para que o que tenha a melhor medida no espaço de hipóteses seja retornado. Portanto, as abordagens de comitês podem ser consideradas como uma família de algoritmos de classificação, desenvolvida para melhorar as capacidades de generalização dos classificadores. É difícil conseguir uma única classificação modelo com boa capacidade de generalização, que é chamado de classificador forte, mas pode-se transformar um conjunto de classificadores fracos em um forte por meio de sua combinação (AGGARWAL, 2014).

Um exemplo simples, considerando um cenário de previsão do tempo, mostra uma possível aplicação destes algoritmos de classificação assumindo que todos os rótulos estarão disponíveis. A ideia é que o algoritmo de classificação indique se vai ou não chover no dia seguinte, utilizando como atributos medições meteorológicas fornecidas do dia atual (umidade, vento, temperatura, etc.). No dia seguinte, o método vai ter as respostas (choveu ou não choveu) para determinar as suas próximas previsões e aprender com seus erros/acertos (SHALEV-SHWARTZ et al., 2012; SANTOS, 2019).

Diversas técnicas têm sido usadas para gerar um bom conjunto de classificadores base. Entre elas encontram-se: (i) obter diferentes exemplos de bootstrap do conjunto de treinamento e treinar um classificador em cada exemplo de bootstrap (BREIMAN, 1996); (ii) extrair diferentes subconjuntos de exemplos ou subconjuntos de recursos ou características e treinar um classificador em cada subconjunto; (iii) aplicar diferentes algoritmos de aprendizagem no conjunto de treinamento; (iv) incorporar aleatoriedade no processo de um algoritmo de aprendizagem particular ou usar parametrização diferente para obter resultados de predição

diferentes (AGGARWAL, 2014). As técnicas (i) e (iv) são as estratégias relacionadas com este tópico de aprendizagem de comitês utilizadas nesta pesquisa para melhorar o desempenho dos diferentes comitês de classificação introduzidos na área.

Como já foi dito nas especificações anteriores, no aprendizado incremental em ambientes com mudanças de conceito, também existem comitês de classificadores que relacionam o procedimento de classificação em conjunto com a detecção de mudanças (FRÍAS-BLANCO et al., 2016). Com este propósito, eles usam os detectores de mudanças de conceito para melhorar a performance dos métodos e o desempenho global da classificação quando ocorrem estas mudanças na distribuição dos dados.

Embora existam comitês que apresentam boas performances gerais (BARROS; SANTOS, 2019; BARROS; SANTOS; GONÇALVES JR., 2016; FRÍAS-BLANCO et al., 2016), não está claro o que poderia acontecer com o desempenho de tais métodos de classificação em termos de acurácia se novas estratégias para ajustar dinamicamente seus parâmetros fossem adotadas. Por outro lado, é válido ressaltar que este tema de ajuste dinâmico de parâmetros tem sido pouco aprofundado nos trabalhos da área, pelo que ainda existem vários desafios e soluções a descobrir que podem ser consideradas nessa direção.

Até então, não é possível encontrar respostas definitivas para os pontos levantados. Para tentar dar mais um passo nesse sentido, a pesquisa atual propõe-se responder a seguinte questão, que é o *problema de investigação*: a aplicação de novas estratégias de ajuste dinâmico de parâmetros aos métodos de classificação existentes na área, poderia melhorar o desempenho das acurácias dos métodos na avaliação experimental de fluxo de dados onde acontecem mudanças de conceitos?

1.2 OBJETIVOS

Levando em consideração os pontos mencionados anteriormente, o *objetivo geral* deste trabalho é: Aplicar estratégias de implementação de ajuste dinâmico de parâmetros aos métodos de classificação existentes na área para melhorar os seus resultados experimentais testados em ambientes onde acontecem mudanças de conceitos na distribuição dos dados.

Como *objetivos específicos* a serem obtidos durante e após o desenvolvimento deste trabalho, estão:

- Investigar e detalhar as estratégias pertencentes à aplicação do ajuste de parâmetros de

forma dinâmica nos métodos de classificação escolhidos neste trabalho.

- Detalhar as principais particularidades de implementação dos métodos de classificação apresentados nesta pesquisa.
- Explicar o processo de configuração experimental selecionado para realizar os testes iniciais do estudo empírico desta tese de doutorado.
- Obter melhores resultados aplicando as novas propostas de ajuste dinâmico de parâmetros nos métodos de classificação definidos de acordo com a métrica de avaliação escolhida no momento de medir o desempenho dos algoritmos em diferentes cenários de avaliação.

1.3 CONTRIBUIÇÕES

Como parte complementar da formulação do problema e a explicação dos objetivos, a seguir apresentam-se as seguintes contribuições desta tese.

- Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN), uma abordagem baseada em k-NN que adapta de forma dinâmica e incremental os valores k usados pelos classificadores pareados k-NN no processo de aprendizagem de fluxos de dados com presença de mudanças de conceito.
- PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$ são versões de PL-kNN que também são apresentadas neste trabalho e utilizam estratégias de implementação diferentes para ajustar os valores k do par de classificadores k-NN usados nos algoritmos propostos.
- PEP, uma técnica genérica para ajustar parâmetros dinamicamente, que é aplicada e testada no parâmetro de diversidade (λ) de diferentes comitês de classificação, visando melhorar suas acurácias de classificação global em cenários com mudanças de conceito. Para este fim, o método de estimação proposto (PEP) foi usado para criar BOLE-PE, OABM1-PE e OzaBag-PE com base nos comitês existentes BOLE, OABM1 e OzaBag, respectivamente. Apesar de sua aplicação específica para comitês neste trabalho, a ideia do PEP é abrangente o suficiente para ser adaptada a outros contextos, por exemplo, otimização dos parâmetros nos classificadores ou detectores de mudanças de conceito.

1.4 METODOLOGIA

Para atingir os objetivos desejados, foram seguidas as metodologias de pesquisa teórica e empírica. Neste sentido, foram analisados trabalhos científicos relacionados, também foram formulados e discutidos os conceitos de pesquisa teórica e empírica da investigação, e finalmente foram feitas avaliações empíricas dos novos métodos introduzidos no trabalho. Na análise da investigação relacionada, identificam-se as desvantagens ou necessidades dos métodos existentes. As soluções propostas são validadas experimentalmente usando uma grande variedade de dados relevantes.

Desta forma, são apresentadas as seguintes *tarefas de investigação* no decorrer do projeto:

- Estudo do estado de arte da classificação de dados em ambientes dinâmicos, tendo maior ênfase nos dados não estacionários onde acontecem mudanças de conceito.
- Revisão bibliográfica sobre detectores de mudanças de conceitos em fluxo de dados.
- Revisão bibliográfica sobre comitês de classificadores em fluxo de dados.
- Investigação dos testes estatísticos não paramétricos mais usados para aplicar aos métodos propostos.
- Levantamento bibliográfico sobre os geradores de bases de dados artificiais e bases de dados reais que contenham mudanças de conceitos em fluxos de dados.
- Realização dos experimentos, utilizando o ambiente MOA (BIFET; HOLMES; PFAHRINGER, 2010), usando conjuntos de dados sintéticos e reais com mudanças de conceito, utilizando os métodos de classificação na distribuição dos dados não estacionários formados em cada base de dados.
- Análise dos resultados dos experimentos e conclusões da pesquisa.

1.5 ORGANIZAÇÃO DO TRABALHO

Os próximos capítulos da tese de doutorado estão organizados da seguinte forma:

- **Capítulo 2**: O escopo deste capítulo é fundamentar os principais conceitos da classificação em fluxo de dados introduzindo os algoritmos de classificação k-NN e HT usados

no trabalho, assim como fazer uma breve revisão conceitual do desempenho da detecção de mudanças de conceitos e caracterização dos recentes trabalhos da área.

- **Capítulo 3**: O propósito deste capítulo é fornecer e explicar os métodos propostos nesta pesquisa relacionados com o ajuste dinâmico de parâmetros. Com essa finalidade, são apresentadas as particularidades de implementação das contribuições: (i) PL-kNN e as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, métodos pareados que usam várias estratégias para adaptar de forma dinâmica os valores do parâmetro k usados pelos membros de classificação k-NN presentes nos enfoques; (ii) PEP a técnica proposta para ajustar dinamicamente parâmetros de métodos relacionados a fluxo de dados e, os comitês BOLE-PE, OABM1-PE e OzaBag-PE que foram o resultado da utilização de PEP para estimar o parâmetro da diversidade (λ) dinamicamente nos enfoques.

- **Capítulo 4**: Este capítulo descreve todas as informações relevantes sobre os experimentos desenvolvidos no trabalho, incluindo também uma breve descrição dos conjuntos de dados artificiais e do mundo real usados nos testes. Também são mostradas as particularidades relacionadas à configuração de parametrização dos métodos introduzidos nesta tese de doutorado.

- **Capítulo 5**: O objetivo deste capítulo é exibir os resultados obtidos dos experimentos usando PL-kNN, as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, assim como também diferentes parametrizações da versão tradicional do k-NN.

- **Capítulo 6**: Nesta parte, são apresentados os resultados dos experimentos relacionados com a aplicação do método PEP em diversos comitês de classificadores. BOLE-PE, OABM1-PE e OzaBag-PE foram os métodos resultantes da aplicação dessa estratégia.

- **Capítulo 7**: Por último este capítulo apresenta as conclusões referentes à pesquisa realizada nesta tese de doutorado e também são apresentadas as perspectivas para trabalhos futuros.

2 REVISÃO DA LITERATURA

O aprendizado de máquina online é um método de aprendizado onde os dados postos à disposição de forma sequencial são usados para fazer previsões futuras em cada etapa. Essas previsões são feitas por médio da atualização de um preditor ou classificador. Desse modo, este tipo de aprendizado é usado em situações onde é preciso que esse preditor se adapte dinamicamente a novos padrões nos dados, ou quando estes dados são gerados em função do tempo (SHALEV-SHWARTZ et al., 2012; FONTENLA-ROMERO et al., 2013).

Um tema que tem vital importância no aprendizado online é a classificação em fluxo de dados. Neste capítulo denota-se a abordagem conceitual da classificação em fluxo de dados. Nesse contexto, apresenta-se uma breve descrição de algoritmos de classificação, detectores e comitês de classificadores utilizados nos experimentos da tese de doutorado para avaliar os métodos apresentados nesta investigação.

2.1 CLASSIFICAÇÃO

A classificação é uma forma de análise de dados onde um modelo ou classificador é construído para predizer ou prever as classes (categóricas) rotuladas ou etiquetadas. Essas categorias podem ser representadas por valores discretos, onde a ordenação entre eles não tem nenhuma importância (HAN; PEI; KAMBER, 2011).

O processo de trabalho da classificação é dividido em duas etapas (NGUYEN; ARMITAGE, 2008; SUN, 2014):

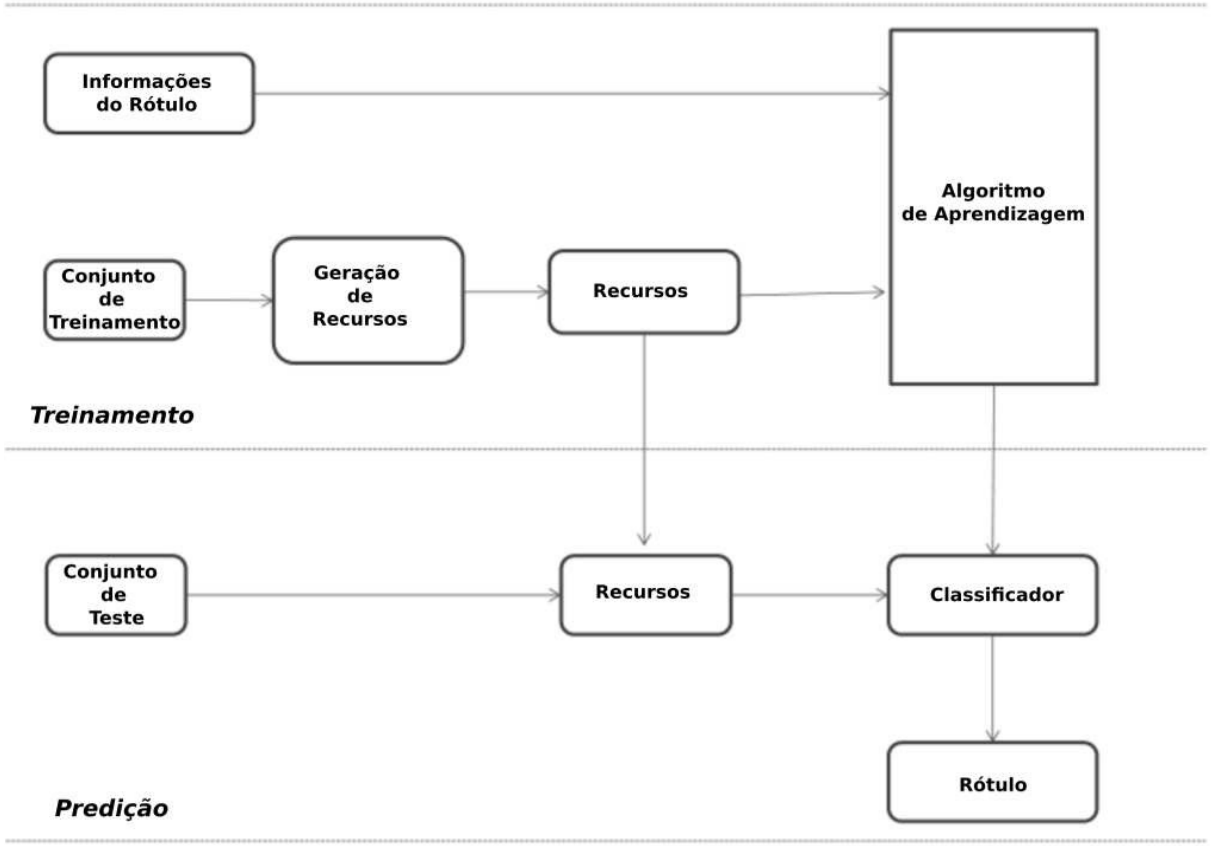
- **Construção do modelo**: Nesta primeira etapa descreve-se um conjunto de classes predeterminadas. Cada exemplo pertence a uma classe predefinida, conforme determinado pelo atributo rótulo da classe. O conjunto de exemplos usado para construir o modelo é o conjunto de treinamento. Esse modelo é representado por regras de classificação, árvores de decisão, ou fórmulas matemáticas.

- **Uso do modelo**: Este segundo passo é efetuado para a classificação dos objetos futuros ou desconhecidos. É aqui onde a acurácia do modelo é estimada. Para fazer a estimação, o rótulo conhecido do exemplo de teste é comparado com o resultado classificado do modelo. Desse modo, a taxa de acurácia é a porcentagem do conjunto de exemplos que são corretamente classificados pelo modelo. Nesta etapa o conjunto de teste é

independente do conjunto de treinamento. Finalmente, se a acurácia for aceitável é conveniente fazer uso do modelo para classificar novos dados.

Na Figura 1 ilustra-se o processo geral de classificação de dados apresentando as duas etapas descritas anteriormente.

Figura 1 – Processo geral de classificação de dados.



Fonte: O autor (2022)

No âmbito de fluxos de dados, a classificação é uma variação da aprendizagem de máquina supervisionada tradicional (FACELI et al., 2011; GAMA et al., 2004b) conjuntamente com as formas de classificação. Ambas formas estão relacionadas com o problema de predizer um valor nominal de uma instância não rotulada representada por um vetor de características. A principal diferença entre essas tarefas é que, em cenários de fluxo contínuo, as instâncias não estão disponíveis para o classificador, não sendo parte de um extenso conjunto de dados estáticos. Ao invés disso, as instâncias são fornecidas sequencial e rapidamente ao longo do tempo como um fluxo de dados contínuo. Portanto, um classificador de fluxo de dados deve estar preparado para lidar com um grande número de instâncias, de modo que cada instância

só pode ser inspecionada uma vez ou armazenada por apenas um curto período de tempo (GOMES et al., 2017).

No problema abordado na tese de doutorado, as entradas para os algoritmos são dadas como um fluxo de dados. Cada instância é da forma $(x_1, y_1), ..., (x_N, y_N)$, com uma função desconhecida $f(x) = y$. Os x_i são vetores da forma $(x_{i1}, x_{i2}, ..., x_{iM})$, com valores discretos, em que x_{ij} refere-se ao valor do j-ésimo atributo, chamado X_j, de instância T_i. Os valores y_i referem-se aos valores do atributo Y, a classe. O atributo de classe y_i é discreto, ou seja, $y_i \in C_1, C_2, ..., C_{N_{Cl}}$. A partir do conjunto de instâncias de treinamento S, um classificador p é induzido pelo sistema de aprendizagem.

2.2 ALGORITMOS DE CLASSIFICAÇÃO

Como foi visto anteriormente, os algoritmos de classificação são úteis no processo de avaliação de aprendizagem num fluxo de dados. Árvores de decisão, regras de associação, redes neurais artificiais, máquina de vetores de suporte e redes bayesianas são uma série de abordagens em que os classificadores desempenham um papel fundamental (AGGARWAL, 2014).

Para colocar este trabalho em prática, o próximo passo foi a seleção dos classificadores k-NN e HT que foram usados no trabalho. No caso de k-NN, o método forma parte de base de estudo e implementação nesta pesquisa na utilização de estratégias para ajustar parâmetros do enfoque. Além disso, os algoritmos também foram escolhidos porque são populares na área de classificação online e estão livremente disponíveis no ambiente MOA.

2.2.1 K-Nearest Neighbors (k-NN)

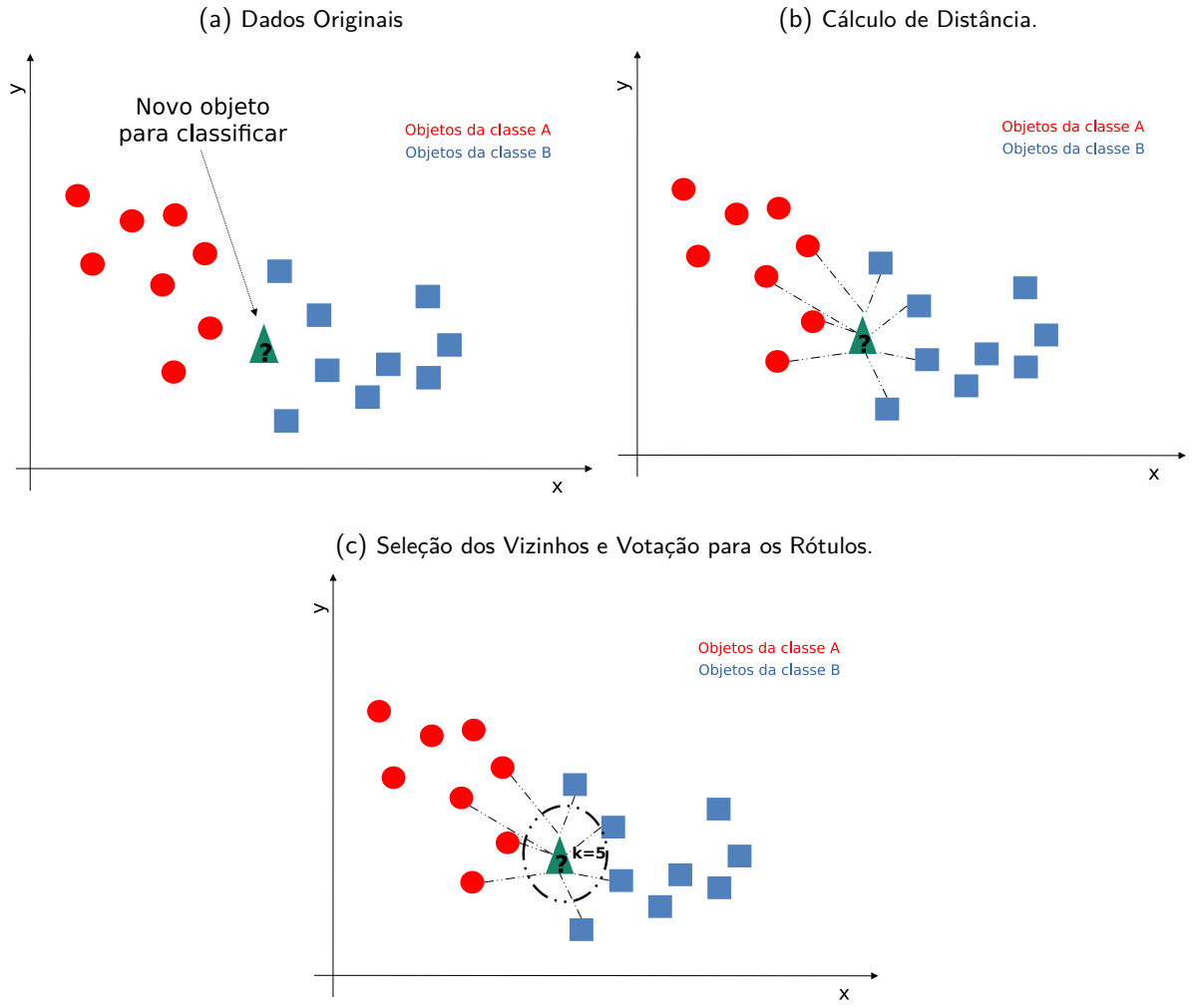
K-Nearest Neighbors (k-NN) é um método de classificação simples, que tem sido utilizado na área de aprendizado de máquina. Este algoritmo considera a proximidade entre os dados para fazer suas predições com base na distância. O fundamento deste classificador está baseado no argumento de que os dados similares ou parecidos entre si têm uma tendencia a se concentrar na mesma região no espaço de dispersão de dados (KRAMER, 2013).

O espaço (diagrama) de dispersão de dados é um gráfico representado por pares de valores correspondentes a duas variáveis em que os pontos ilustrados nesta região mostram o comportamento conjunto dessas variáveis (SOUSA, 2019). Desse modo, a dispersão destes pontos apontam uma relação entre objetos ou atributos (ESCOVEDO; KOSHIYAMA, 2020). Portanto, a

intuição de k-NN é baseada no fato de que objetos relacionados ao mesmo conceito são semelhantes entre si. Além disso, tem a vantagem de que, além de ser utilizado para classificação, também pode ser utilizado para regressão (ZHANG et al., 2017; CAI et al., 2020).

A Figura 2 ilustra o processo de operação do k-NN em geral. As etapas executadas pelo algoritmo são representadas graficamente em planos cartesianos. Observe que o exemplo ilustra um modelo binário (apenas duas classes) em que os círculos vermelhos representam objetos da Classe A, os quadrados representam objetos da Classe B e o triângulo com o ponto de interrogação é o novo objeto a ser classificado. Finalmente, o número de vizinhos k que define o ponto de corte para determinar a classe do objeto a ser classificado no espaço de dispersão é representado na etapa final (Figura 2c) é igual a 5.

Figura 2 – Etapas realizadas pelo algoritmo de classificação k-NN com um número de vizinhos k = 5.



Fonte: O autor (2022)

No âmbito dos fluxos de dados, este método preguiçoso classifica as instâncias de acordo com as k instâncias mais próximas. Essa proximidade é determinada pelo cálculo de uma

medida de distância que permite saber quão semelhantes ou diferentes são duas instâncias. A medida de distância mais usada é a distância Euclidiana (BARDDAL et al., 2016). Dada uma nova instância a ser classificada, k-NN executa as seguintes etapas:

1. Calcula-se a distância entre a instância a ser classificada e as demais do conjunto.
2. As k instâncias mais próximas são selecionadas.
3. A instância é classificada em uma determinada categoria, de acordo com o k número de vizinhos em um buffer de tamanho W.

Diversas contribuições aparecem na literatura relacionadas ao uso de especificações teóricas e implementação do algoritmo de classificação k-NN. Tudo isso para criar novas técnicas e métodos inovadores que resultam do grande interesse nos diferentes cenários e áreas onde essas abordagens são aplicadas. Nesse sentido, em termos de classificação, Liao e Vemuri (LIAO; VEMURI, 2002) propõem um método baseado em k-NN para classificar o comportamento do programa (representado pelas frequências de chamadas do sistema) como normal ou intrusivo. Na análise preliminar desta pesquisa, os autores mostram que o classificador k-NN detectou ataques intrusivos de forma satisfatória e obteve um baixo índice de falsos positivos.

Outra abordagem de aprendizagem preguiçosa inspirada em k-NN é ML-KNN (ZHANG; ZHOU, 2007). Neste método, os k vizinhos mais próximos dos dados pertencentes a cada instância invisível no conjunto de treinamento são identificados primeiro. Então, o método de máxima verossimilhança que pode ser usado para estimar uma série de parâmetros desconhecidos, relacionados a uma dada amostra e conhecido pelo princípio do máximo a posteriori (MAP) é usado para determinar o conjunto de rótulos para a instância invisível usando o número de instâncias vizinhas pertencentes a cada classe possível.

Outra proposta diz respeito a um método de categorização de texto k-NN baseado no vizinho mais próximo compartilhado, combinando as informações de vizinhança das amostras com o cálculo de similaridade BM25 para garantir a eficácia do método na classificação de patentes (CAI; JI; CAI, 2010). O algoritmo de classificação k-NN também foi utilizado na avaliação de desempenho de esquemas de classificação de imagens de satélite, por estar entre as técnicas de classificação mais populares, geralmente utilizadas na classificação de imagens (SRIVAS, 2019).

Adicionalmente, com o objetivo de melhorar a performance do k-NN em termos de classificação ajustando hiperparâmetros, o trabalho de Bahri et al. (2020) propõe a versão RP-kNN

que aplica uma técnica de redução de dimensão interna (DR) por meio de projeção aleatória (RP) ao fluxo a fim de reduzir o uso de recursos e usa um sistema de monitoramento automático que ajusta dinamicamente a configuração do algoritmo k-NN e o tamanho da dimensão de saída com fluxo de dados de alta dimensionalidade. Nesse processo eles usam o enfoque AutoML que dinamicamente ajusta hiperparâmetros dos classificadores e, principalmente usá-lo com o algoritmo RP-kNN.

Por outro lado, também existem trabalhos onde o algoritmo de classificação k-NN é utilizado em ambientes de fluxos de dados com mudanças de conceito. Exemplos incluem o estudo do problema Multi-Label Stream Classification (MLSC) (XIOUFIS et al., 2011), onde múltiplas mudanças de conceito, distorção de rótulo e expansão do conjunto de rótulos são identificados. Os autores desenvolvem uma implementação k-NN para instanciar a abordagem de janela múltipla proposta. O método faz uma única passagem pelos exemplos alocados em um buffer compartilhado e, a seguir, calcula a distância entre a instância do rótulo e a expansão do conjunto de rótulos. Finalmente, os votos para cada rótulo são analisados em uma lista para encontrar os vizinhos mais próximos.

No framework RCD (GONÇALVES JR.; BARROS, 2013), k-NN é usado como um teste estatístico multivariado não paramétrico configurável para comparar amostras de dados e identificar se conceitos novos ou antigos estão ocorrendo. Classificadores usados anteriormente em conceitos recorrentes são reutilizados. Desta forma, a estrutura lida com mudanças de conceito recorrentes em ambientes de aprendizagem online.

O Self Adjusting Memory Model for k-NN (SAM-kNN) (LOSING; HAMMER; WERSING, 2018) lida com diferentes tipos de mudanças de conceito. Este método foi implementado com a ideia de construir modelos dedicados aos conceitos atuais e anteriores para poder aplicá-los de acordo com os requisitos de uma dada situação, sem a necessidade de otimizar meta-parâmetros. Neste caso, o k número de vizinhos *não* é um parâmetro a ser ajustado na fase de adaptação do método.

O vizinho mais próximo (NN-DVI) (LIU et al., 2018) foi proposto para detectar mudanças de conceito com base na estimativa da densidade regional. Entre seus componentes, há um esquema de particionamento de espaço baseado no vizinho k mais próximo (NNPS), que transforma instâncias de dados discretos não medidos em um conjunto de subespaços compartilhados para estimação de densidade. Nesse sentido, o valor k de k-NN é de vital importância para monitorar a resolução que faz parte da construção do método.

O framework Dynse (ALMEIDA et al., 2018) usa Seleção Dinâmica de Classificadores (DCS)

para lidar com mudanças de conceito. Na implementação do método, as funcionalidades k-NN são utilizadas para selecionar as k instâncias mais próximas de uma janela de estimativa de acurácia W representada em uma região local definida pela vizinhança do número de instâncias presentes na janela. Finalmente, é importante notar que k-NN também foi usado como um classificador base nas comparações experimentais de várias investigações que fazem avaliações em diferentes cenários de fluxos de dados com mudanças de conceito (KOYCHEV, 2007; LU; ZHANG; LU, 2014; BARDDAL et al., 2016; LU et al., 2016).

2.2.2 Hoeffding Tree (HT)

HT é um algoritmo de classificação incremental popular para lidar com as mudanças de conceito. Suporta a indução de árvores de decisão e fornece soluções para os seguintes desafios:

- A incerteza no tempo de aprendizagem. Em HT, o tempo de aprendizado é constante por exemplo (instância) e isso significa que o classificador é adequado para a mineração de fluxos de dados.
- As árvores resultantes são quase idênticas às árvores construídas pela aprendizagem em lote convencional (MARRÓN et al., 2016), desde que receba instâncias suficientes para treinar e construir as árvores.

HT assume que a geração de distribuição dos exemplos não é constante e explora o fato de que uma pequena amostra pode ser suficiente para escolher um atributo com boa separação entre as classes, o que é matematicamente suportado pelo conceito de Hoeffding bound (HOEFFDING, 1963; MARON; MOORE, 1994). Este conceito afirma que, com probabilidade $1-\delta$, a verdadeira média da variável é de pelo menos $\bar{r}-\varepsilon$, onde $\bar{r}$ é o valor médio calculado a partir de n observações independentes de uma variável aleatória de valor real r com limite R e ε está determinado pela equação 2.1.

$$\varepsilon = \sqrt{\frac{R^2(\ln 1/\delta)}{2n}} \tag{2.1}$$

Hoeffding bound quantifica o número de observações quando é necessário estimar o quão bom um atributo é (DOMINGOS; HULTEN, 2000). O que torna a Hoeffding bound atrativo é a sua capacidade de dar os mesmos resultados independentemente da distribuição de probabilidade, gerando assim as observações. No entanto, o número de observações necessárias para atingir certos valores de δ e ε são diferentes entre as distribuições de probabilidade.

Em relação às contribuições deste popular algoritmo de classificação, na literatura existem diversos trabalhos enaltecedores que têm sido desenvolvidos no transcurso dos anos com base nas particularidades de classificação do tradicional HT. Entre eles está a pesquisa de Kumar, Kaur e Sharma (2015) onde foi feita uma comparação aprofundada em termos teóricos entre os algoritmos Hoeffding Tree, Streaming Random Forest e a versão rápida Concept Adapting Very Fast Decision Tree (CVFDT) (DOMINGOS; HULTEN, 2000) que são usados para a classificação de fluxo de dados. Nesse trabalho são detalhadas as particularidades dos algoritmos assim como o desempenho deles em diferentes ambientes onde são utilizados.

Além disso, também foi apresentado o Vertical Hoeffding Tree (VHT) (KOURTELLIS et al., 2016), algoritmo de classificação de streaming distribuído para a aprendizagem com árvores de decisão. Aqui os autores vinculam as fundamentações de implementação de algoritmos distribuídos e de streaming para desenvolver o método distribuindo árvores de decisão por meio de uma estratégia de paralelismo vertical.

No estudo de Pham et al. (2017) os autores apresentam uma abordagem baseada em comitês para o aprendizado de máquina online onde vários HT classificam e são atualizados nos dados projetados de dimensão inferior gerados a partir da originalidade por projeções aleatórias. Uma vez que a projeção aleatória é instável, a partir de um exemplo, diversos dados são usados para treinar o conjunto de classificadores HT. Nos experimentos realizados eles demonstram que a abordagem proposta tem um desempenho significativamente melhor do que o original enfoque HT.

Também, foi proposto o FAHT (Fairness-Aware Hoeffding Tree) (ZHANG; NTOUTSI, 2019) outra extensão do conhecido algoritmo de classificação HT para indução de árvore de decisão em fluxos de dados que leva em consideração a justiça ou equidade na classificação online em sistemas automatizados de tomada de decisão baseados ou orientados por dados. Nos experimentos realizados neste trabalho foi evidenciado que o FAHT foi capaz de lidar com a discriminação em ambientes de streaming mantendo um desempenho moderado.

De outro ponto de vista, relacionando a classificação de HT com as mudanças de conceito, também existem vários trabalhos desenvolvidos neste sentido. A fim de apoiar a classificação dos fluxos de dados de sensores, o artigo de Song et al. (2016b) propõe uma estratégia de análise de fluxos de dados de sensores massivos baseado em HT com mudanças de conceitos para aplicação de monitoramento de eventos no sistema Hadoop. Na proposta, os autores utilizam uma plataforma de redução de mapas do próprio sistema Hadoop na classificação de fluxo de dados sensoriais, demonstrando com os resultados da simulação que a estratégia

consegue mais economia de energia e também garante poucas quantidades de dados do sensor retidos na memória.

O Hoeffding Anytime Tree é um dos recentes algoritmos de árvores de decisão incremental apresentados por Manapragada, Webb e Salehi (2018) os quais aludem que a proposta é resiliente a mudanças de conceitos e pode ser usado como um substituto de maior acurácia para o tradicional HT em diversos cenários. Este método apresenta um pequeno custo computacional em relação ao seu comparativo tradicional, mas os autores somente apresentaram evidencias experimentais utilizando bases de dados do mundo real do repositório UCI (FRANK, 2010) e não levaram em consideração o desempenho do método em ambientes experimentais com bases artificiais para demonstrar quão resistente pode ser quando acontecem mudanças na distribuição dos dados.

Também existem pesquisas que enaltecem as contribuições do tradicional HT e sua relação com as mudanças de conceitos. Um exemplo que demonstra esta afirmação é o trabalho de Muallem et al. (2017) onde apresenta-se uma visão geral ampla e bem construída do uso do aprendizado de máquina para o problema das mudanças de conceito em fluxo de dados, considerando o fornecimento da eficácia do uso de HT como uma solução. Especificamente, os autores categorizam os métodos propostos de árvores de decisão com base na construção de características para lidar com a detecção das mudanças nas distribuições dos dados. Esses modelos podem ser uma árvore de decisão, árvores baseadas no limite de Hoeffding, árvores baseadas em janelas, peso ou um comitê de classificadores HT.

2.3 DETECÇÃO DE MUDANÇAS DE CONCEITO

A função de um detector de mudanças de conceito é identificar e detectar o momento preciso onde ocorrem as mudanças na distribuição de probabilidades dos dados. A metodologia Prequential, independente da abordagem adotada para o cálculo estatístico, tem a função de avaliar a tarefa de classificação e, para isso, cada instância é repassada ao classificador, esperando-se uma resposta (HIDALGO; MACIEL; BARROS, 2019). As respostas das previsões do classificador são disponibilizadas aos algoritmos de detecção de mudanças de conceitos.

Os detectores trabalham com os acertos e erros das classificações realizadas pelo classificador. As respostas do classificador estão representadas como uma função B que associa cada elemento ω do espaço amostral Ω com um valor $b \in R\ N(0,1)$. São admitidos apenas dois valores 0 (se ocorrer sucesso) ou 1 (se ocorrer fracasso). Portanto, os detectores trabalham

com base em dados binários e precisam detectar a ocorrência de mudanças de conceitos.

Diferentes técnicas são usadas para detectar mudanças de conceitos e uma das mais comuns é baseada em duas janelas: geralmente há uma janela que possui dados recentes e outra que tem dados mais antigos. Teoricamente, os tamanhos das janelas devem de ser maiores que 30 observações para respeitar o teorema central do limite (JOHNSON, 2004), no qual afirma-se que dado um conjunto de amostras maior ou igual a 30, é possível realizar uma estimação de aproximação em relação à distribuição de normal dos dados. Sendo assim, as amostras assumem a normalidade dos dados de uma população.

Tradicionalmente, as variabilidades das distribuições são calculadas usando alguma medida estatística. Portanto, a baixa correlação de similaridade das distribuições sugere a ocorrência de mudança de conceito. No framework MOA, os dados que formam as distribuições são formados por sequências de zeros e uns $(0/1)$ em um número crescente de instâncias ao longo do tempo.

2.3.1 Detectores de Mudanças de Conceito

Como foi visto anteriormente, é evidente a estreita relação dos métodos ou detectores de mudanças de conceito conjuntamente com os classificadores para trabalhar no processo de aprendizagem com fluxo de dados.

Em geral, os detectores analisam as respostas das predições do classificador base e adotam um modelo de decisão para detectar as mudanças na distribuição dos dados. Em especifico, estes métodos trabalham com dois níveis de alarmes: $warning$ e $drift$ (ATTAR et al., 2012, p. 158), onde, o $drift$ representa um nível maior de mudança na distribuição analisada e simboliza que, de fato, ocorreu uma modificação de conceito. Assim sendo, quando o nível de $warning$ é sinalizado, uma nova instância do classificador base é criada e mantida em paralelo com o classificador antigo. Caso o nível de $drift$ seja alcançado, o detector exclui o antigo classificador e mantém apenas o novo. Por outro lado, caso o sinal de $warning$ passe a ser considerado um alarme falso, a nova instância do classificador é excluída.

Nas próximas subseções apresentam-se alguns detectores de mudanças de conceito, que pela particularidade de suas características são utilizados nas experimentações deste trabalho.

2.3.1.1 Drift Detection Method (DDM)

O Drift Detection Method (DDM) (GAMA et al., 2004b) assume que, de acordo com o modelo de aprendizagem provavelmente aproximadamente correta (PAC) (MITCHELL, 1997, p. 201–223), se um determinado algoritmo preditor estiver recebendo – como entradas – exemplos de uma distribuição estacionária, então a sua taxa de erro irá decrescer quando o número de exemplos aumentar. Assim, um significativo aumento nos erros do classificador sugere uma mudança na distribuição aprendida e que o modelo de aprendizagem atual não é mais apropriado.

Uma vez que o detector foi projetado para trabalhar de forma conjunta com um classificador supervisionado, o método supõe que os erros e acertos do preditor seguem a forma de uma distribuição binomial, onde, para cada ponto i da sequência de exemplos aprendida, a taxa de erro é representada por p_i e o seu desvio padrão é dado por: $s_i = \sqrt{p_i \times (1 - p_i) \ / \ i}$.

O algoritmo de detecção proposto gerencia duas variáveis – p_{min} e s_{min} – durante o treinamento. Propositalmente, o método inicializa p_{min} e s_{min} com um número real de alto valor. Assim, a cada instância avaliada, é comparado se $p_i + s_i < p_{min} + s_{min}$. Caso a expressão anterior seja verdadeira, p_{min} e s_{min} recebem, respectivamente, os valores de p_i e s_i.

Conforme mencionado anteriormente, é comum que os detectores de mudanças de conceitos trabalhem com dois níveis de alarmes ($warning$ e $drift$). Para o DDM, os níveis de $warning$ e $drift$ são alcançados, respectivamente, quando $p_i + s_i \geq p_{min} + w \times s_{min}$ e $p_i + s_i \geq p_{min} + d \times s_{min}$. Onde, w e d são dois parâmetros configurados pelo usuário, os quais – por padrão – são 2 e 3, respectivamente. Além disso, o detector também possui o parâmetro n, o qual é responsável por determinar o número mínimo de instâncias que devem ser analisadas antes que o método passe a realizar as comparações citadas.

2.3.1.2 Fisher Test Drift Detector (FTDD)

Fisher Test Drift Detector (FTDD) é o detector mais transparente dos três detectores de mudanças de conceito (CABRAL; BARROS, 2018) baseado em uma implementação eficiente do teste exato de Fisher (FISHER, 1934; YATES, 1934).

Independentemente do teste exato de Fisher ser, geralmente, indicado apenas para a análise de pequenas amostras, a sua implementação permite que o FTDD seja aplicado em conjuntos de dados de quaisquer tamanhos. Para isso, o método utiliza uma janela de dados antigos com

tamanho reduzido para o valor proporcional ao número de erros e acertos contidos na janela antiga original.

Com a simplicidade do cálculo do valor de p, é possível verificar que o método em questão realiza o teste estatístico utilizando o número de erros – e acertos – da janela recente (w_r e r_r) e os valores proporcionais ao número de predições corretas, e incorretas, contidas na janela antiga (w_p e r_p).

Por fim é valido ressaltar que o FTDD baseia-se em Statistical Test of Equal Proportions (STEPD)(NISHIDA; YAMAUCHI, 2007) e na deficiência do seu teste estatístico de proporções iguais em situações em que as amostras de dados são pequenas ou desbalanceadas.

2.3.1.3 Hoeffding-based Drift Detection Method (HDDM)

De acordo com seus autores (FRÍAS-BLANCO et al., 2015), Hoeffding-based Drift Detection Method (HDDM) é uma família de métodos que propõem monitorar as métricas de desempenho medidas durante o processo de aprendizagem e sinalizar as mudanças quando uma variação significativa é detectada. Os métodos aplicam desigualdades de probabilidade que assumem apenas variáveis aleatórias independentes, uni-variadas e delimitadas para obter garantias teóricas para as detecções. Duas versões principais são descritas:

- **ATest** é implementado mediante modificações realizadas a um corolário proposto por Hoeffding (1963) que pode ser aplicado à detecção de mudanças significativas nas médias móveis dos valores de um fluxo de dados, surgindo assim um teste estatístico bicaudal onde são detectadas as mudanças na média da população e por conseguinte, permite monitorar a diferença entre as médias abordadas do teste utilizando uma sequência de variáveis aleatórias $X_1, ..., X_n$, $Y_1, ..., Y_m$. Os autores sugerem que o melhor comportamento desta versão é nas detecções de mudanças de conceito abruptas.

- **WTest** é baseado em um teste estatístico mais geral que usa médias móveis ponderadas, embora igualmente eficiente e simples. Neste caso, os valores recentes no fluxo de dados têm mais peso do que os mais antigos, assumindo que eles têm maior probabilidade de ocorrência. O teste McDiarmid (1989) é uma generalização do conceito de Hoeffding bound para variáveis aleatórias dependentes. Finalmente, este método é mais recomendável para a utilização das execuções com as mudanças de conceito graduais.

2.3.1.4 Reactive Drift Detection Method (RDDM)

O Reactive Drift Detection Method (RDDM) (BARROS et al., 2017) é um eficiente detector inspirado em DDM. Entre outras modificações heurísticas, foi proposto com o objetivo de superar/reduzir um problema de performance do DDM quando os conceitos são muito grandes. Neste cenário, existe perda de sensibilidade e muitas instâncias passam a ser exigidas para realização das detecções. Assim, o RDDM adiciona explicitamente um mecanismo para descartar instâncias antigas em conceitos grandes e periodicamente recalcula as estatísticas responsáveis por detectar os níveis de alerta e de mudanças de conceito, melhorando a precisão de suas detecções e especialmente a acurácia final.

Outra finalidade do método é a realização de uma detecção forçada quando o número de instâncias em estado de alerta atinge um limite parametrizável (*warnLimit*). Ter longos períodos de alerta é uma situação comum sobretudo em mudanças graduais, e que muitas vezes é seguido de um falso negativo. Assim, a ideia e efeito prático dessa funcionalidade é tornar o detector mais eficiente na detecção de mudanças graduais.

Finalmente, seus autores afirmam que o RDDM alcança, em termos gerais, uma acurácia superior ao DDM, especialmente em mudanças graduais. Apesar de aumentar um pouco o consumo de memória e a quantidade de falsos positivos, as detecções geralmente são realizadas rapidamente.

2.3.2 Comitês de Classificadores (Ensembles)

Formalmente, um classificador P aprende de um conjunto de dados de treinamento S : $\{h_1(x), ..., h_P(x)\}$, cada um dos quais mapeia as características dos valores de x em um rótulo de classe y. Uma vez que são combinados em um comitê (ensemble) de classificadores $H(x)$, a expectativa é de que o desempenho da classificação seja melhor.

Há dois fatores principais que contribuem para o sucesso do aprendizado de comitês. Primeiro, a análise teórica e a prática real mostram que o erro esperado de um modelo de ensemble é menor do que o de um único modelo. Intuitivamente, se existisse a informação a priori de que $h_1(x)$ tem o melhor desempenho de predição em dados futuros, então sem dúvida os outros classificadores deveriam ser descartados e a escolha seria $h_1(x)$ para as predições futuras. Portanto, o mais correto seria escolher a previsão obtida pela combinação de vários modelos (AGGARWAL, 2014).

Outra vantagem do aprendizado de comitês é sua capacidade de superar a limitação da hipótese de espaço de hipóteses feita por modelos únicos. Como discutido anteriormente, um classificador de modelo único geralmente procura o melhor modelo dentro de um espaço de hipótese que é assumido pelo algoritmo de aprendizado específico. É muito provável que o modelo verdadeiro não resida no espaço de hipóteses, e então é impossível obter o modelo verdadeiro pelo algoritmo de aprendizado. No entanto, ao combinar vários classificadores, as abordagens de ensemble podem simular com sucesso o limite verdadeiro. A razão é que os métodos de aprendizagem combinam hipóteses diferentes e a hipótese final não está necessariamente contida no espaço da hipótese original. Portanto, os métodos de ensemble têm mais flexibilidade na hipótese que eles poderiam representar (BISHOP, 2006; SENI; ELDER, 2010).

Em sentido geral, os métodos de comitês surgiram como uma técnica poderosa para melhorar a robustez e a acurácia dos modelos básicos. Na atualidade existem inúmeros estudos sobre o problema de combinar modelos concorrentes em um comitê, e o sucesso de técnicas de comitês foi observado em múltiplas disciplinas e aplicações (WANG et al., 2003; SEWELL, 2008; ZHOU, 2012; LARGE; LINES; BAGNALL, 2017).

Nas próximas subseções apresentam-se os comitês de classificadores, que pela particularidade de suas características e o bom desempenho na área de aprendizagem em fluxo de dados são utilizados como base referencial para criar novos métodos e estabelecer comparações experimentais neste trabalho.

2.3.2.1 Bagging e Boosting

Bagging (BREIMAN, 1996) e Boosting (FREUND; SCHAPIRE, 1997) são os métodos tradicionais usados para melhorar a acurácia de outros algoritmos (*weak* learners). Ambos usam um conjunto de classificadores e os combinam em um ensemble, agregando suas respostas com o objetivo de oferecer melhores resultados.

Bagging adota a reamostragem do conjunto de treinamento com repetições para gerar diferentes amostras de bootstrap de forma aleatória para treinamento. De forma explicita, o comitê treina os classificadores de forma independente por vários conjuntos de treinamento por meio do método de inicialização. Para construí-los é preciso montar m conjuntos iguais e consequentemente replicar esses dados de treinamento aleatoriamente para gerar m amostras independentes por reamostragem com reposição. Seguidamente, é utilizado o método de combinação por maioria de votos para agregar as m amostras.

Diferentemente, o Boosting varia a quantidade de diversidade dada ao treinamento de cada membro, dependendo das predições anteriores, como meio de gerar distribuições diferentes sobre os dados de treinamento. Dessa forma, a relevância do voto é ponderado com base no desempenho de cada modelo, em vez da atribuição de mesmo peso para todos os votos. Especialmente, esse procedimento permite aumentar o desempenho de um limiar arbitrário simplesmente adicionando classificadores (learners) mais fracos.

2.3.2.2 Oza and Russell's Online Bagging (OzaBag) e Oza and Russell's Online Boosting (OzaBoost)

Oza and Russell's Online Bagging (OzaBag) e Oza and Russell's Online Boosting (OzaBoost) (OZA; RUSSELL, 2001) são propostas que adotaram uma distribuição de Poisson para simular o comportamento de seus correspondentes algoritmos offline em ambientes online.

O OzaBag simula o processo de bootstrap enviando K cópias de cada novo exemplo para atualizar cada modelo base. A variável K é aleatória e definida de acordo com a distribuição de Poisson. Esta estratégia produz um comportamento de aprendizado semelhante ao do bagging em lote (batch bagging).

Por outro lado, o OzaBoost gera uma série de modelos $h_1....h_M$. Cada modelo h_m é aprendido a partir de um conjunto de treinamento ponderado cujos pesos são determinados pelos erros de classificação do modelo anterior $h_m - 1$. Em específico, os exemplos classificados incorretamente por $h_m - 1$ recebem mais peso no conjunto de treinamento para h_m, de modo que os pesos de todos os exemplos classificados incorretamente constituem a metade do peso total do conjunto de treinamento. Neste método também é usada a distribuição de Poisson para aproximar o algoritmo de reponderação.

Além disso, esses dois métodos são frequentemente usados como abordagens de linha de base em muitas comparações experimentais na área de mineração de fluxo de dados.

2.3.2.3 Adaptable Diversity-based Online Boosting (ADOB)

Adaptable Diversity-based Online Boosting (ADOB) (SANTOS et al., 2014) é uma variação do método Adwin Online Boosting (Adwin OzaBoost) (OZA; RUSSELL, 2001). Basicamente, o ADOB propõe a distribuição de instâncias de forma mais eficiente entre os especialistas para adaptar-se mais rapidamente às situações onde ocorrem mudanças de conceitos com

frequência. Nesse sentido, a distribuição é realizada controlando-se a diversidade (λ) através da acurácia dos especialistas, que são usados para reduzir o erro inicial com a finalidade de aumentar o foco em casos de classificação difícil e acelerar a diversidade.

De forma geral, o método classifica os especialistas por acurácia antes de processar cada instância. Isso afeta a forma como a diversidade é distribuída aos classificadores e tende a melhorar levemente a acurácia do ensemble logo após a mudança de conceito, especialmente quando esses drifts são abruptos.

Quando chega uma instância, inicialmente o especialista com menos acurácia será selecionado. Por conseguinte, se essa instância for classificada corretamente, supõe-se que os outros especialistas (mais precisos) também tenham boas chances de classificados corretamente, considerando dessa maneira que um erro é improvável.

Além disso, a diversidade λ também é diminuída quando a classificação está correta e aumentada se estiver incorreta. Este procedimento faz com que a influência de um erro improvável em λ diminua à medida que mais instâncias sejam processadas, porque o próximo especialista selecionado será aquele com a melhor acurácia.

2.3.2.4 Boosting-like Online Learning Ensemble (BOLE)

Boosting-like Online Learning Ensemble (BOLE) (BARROS; SANTOS; GONÇALVES JR., 2016) é um dos recentes comitês com um desempenho eficiente nos trabalhos relacionados da área. O método foi implementado com base nas modificações heurísticas do anteriormente explicado ADOB, investigando empiricamente os efeitos de: (i) enfraquecer os requisitos para permitir que os especialistas votem e (ii) alterar o detector de mudanças de conceito usado internamente, com o objetivo de melhorar a acurácia do método.

Com base nestas observações, os autores de BOLE decidiram modificar ADOB (e de fato também OzaBoost) para tentar algumas estratégias diferentes destinadas a melhorar a acurácia dos ensembles correspondentes. Portanto, foram investigadas duas possibilidades para enfraquecer a tradicional estratégia de votação e forçar mais especialistas a votar.

No primeiro cenário, manter o requisito de erro abaixo de 50% para votar, mas aceitar os votos de todos os classificadores no ensemble que atendem a essa condição de 50%. No segundo cenário (mais permissivo), foi adotado um limite de erro maior para aceitar os votos dos classificadores e uma estratégia ligeiramente diferente para agregá-los. Finalmente a terceira adaptação examinada foi a substituição do detector de mudanças de conceito utilizado

internamente em ambos os métodos, que foi Adaptive Windowing (Adwin) (BIFET; GAVALDÀ, 2007).

A implementação de BOLE foi desenvolvida para melhorar o método java *getVotesForInstance* que implementa o cálculo de votação original de ambos os algoritmos de reforço (Adwin OzaBoost, ADOB) testados conforme implementado no MOA estrutura. Este método tem como desvantagens que só permite um classificador (hipótese fraca) para votar se o seu erro é de até 50% e por conseguinte, que uma instância seja processada pelos classificadores restantes depois que um deles falhar na condição 50%.

Além disso, BOLE oferece um desempenho muito forte na maioria das situações, independentemente do método auxiliar de mudança de conceito usado (BARROS; SANTOS, 2019), porém os autores inicialmente consideraram na comparação de detectores de mudanças de conceito que DDM era o melhor método de detecção padrão. Assim, a variante usada (*BOLE4*) para realizar o experimento, incorpora DDM como mecanismo de detecção padrão. Mas as recentes experimentações feitas com o método RDDM alegam que BOLE tem melhores resultados usando este detector de mudanças de conceito.

2.3.2.5 Online AdaBoost-based M1 (OABM1)

O Online AdaBoost-based M1 (OABM1) (SANTOS; BARROS, 2019) foi proposto com o objetivo de manter as características da versão tradicional do Adaptive Boosting for Multiclass Learning 1 (AdaBoost.M1) (FREUND; SCHAPIRE, 1996) e realizar uma melhor distribuição do peso das instâncias para combinar com o comportamento do método tradicional.

Para isso, os autores propõem na implementação do método uma nova função de atualização que se comporta da maneira esperada em todos os casos para atingir o comportamento inesperado de OzaBoost em alguns casos quando: é reduzido o peso das instâncias que são classificadas incorretamente e aumentado o peso das que são classificadas corretamente.

Na implementação do método, cada um dos classificadores é treinado de acordo com os pesos atuais representados por um vetor, os quais são utilizados numa distribuição de Poisson (OZA; RUSSELL, 2001), com o objetivo de manter essa característica similar à versão tradicional do AdaBoost.M1. Nesse sentido, as mesmas restrições da versão oficial também foram mantidas no OABM1.

Finalmente é válido ressaltar que as experimentações feitas pelos autores após a criação do método demostraram que a proposta do algoritmo mantém altas acurácias, superando os

outros métodos testados. Além disso, foram usados argumentos teóricos para demonstrar a convergência de OABM1.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo, foram apresentadas as particularidades referentes à classificação em fluxo de dados com o aprendizado online. Especificamente, foram descritos os algoritmos de classificação k-NN e HT utilizados nesta tese de doutorado, assim como algum dos trabalhos relacionados dos enfoques referentes aos seus vínculos com a classificação e a detecção de mudanças de conceitos. Além disso, também foi ressaltada a importância dos detectores e comitês de classificadores em relação à suas utilidades no processo de aprendizagem online com fluxo de dados.

3 AJUSTE DINÂMICO DE PARÂMETROS. MÉTODOS PROPOSTOS

O processo de implementação de novos métodos para tratar com fluxo de dados onde acontecem mudanças de conceitos requer algoritmos que sejam capazes de se adaptar a diferentes situações para, assim, melhorar a performance nas diferentes avaliações experimentais onde são testados. Por isso, é importante desenvolver algoritmos que sejam capazes de lidar com situações em que os algoritmos clássicos de mineração de dados não apresentam um desempenho satisfatório. Com base nesses desafios, nas próximas seções deste capítulo são apresentados os métodos introduzidos neste trabalho para realizar ajustes dinâmicos nos parâmetros de vários métodos de classificação introduzidos na área.

3.1 PAIRED K-NN LEARNERS WITH DYNAMICALLY ADJUSTED NUMBER OF NEIGHBORS (PL-KNN)

O número de vizinhos k é um dos parâmetros mais importantes do k-NN, impactando diretamente na classificação final na maioria das avaliações experimentais onde é utilizado. Geralmente, o valor k é fixo, determinado antes do processo de aprendizagem, e uma escolha errada pode influenciar negativamente nos resultados finais do classificador, pois diferentes fluxos de dados possuem características distintas e valores ótimos para k.

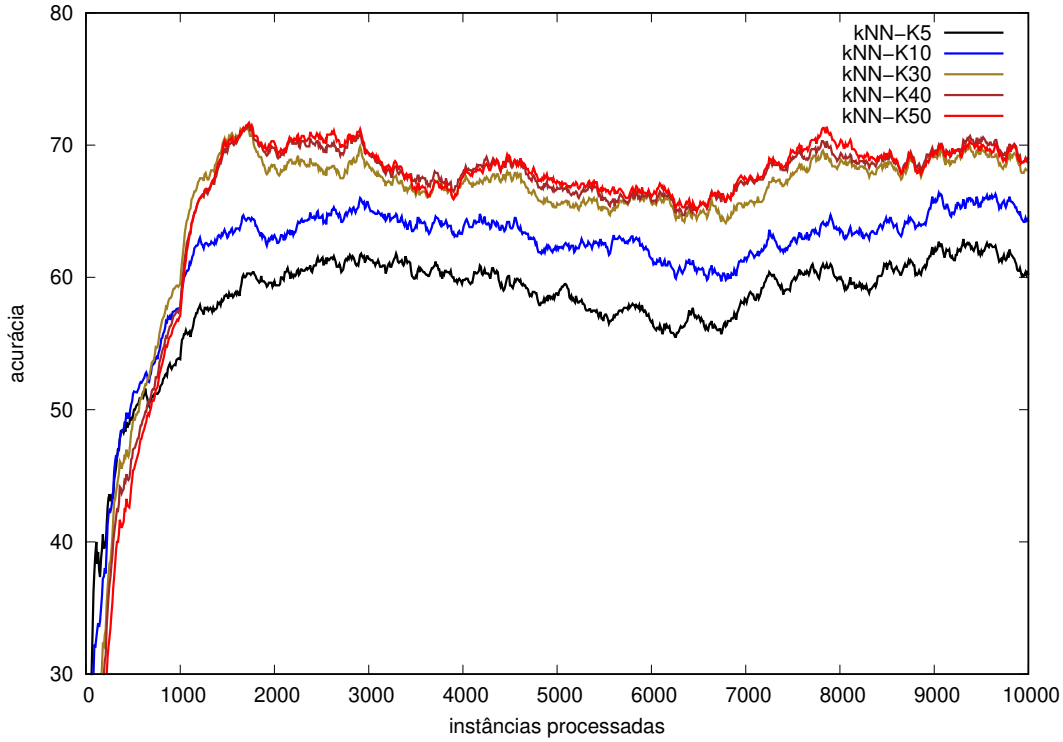
Um ponto de partida motivador desta investigação foi a expectativa de que, mesmo nos conjuntos de dados onde k maior é melhor para a classificação, com poucas instâncias processadas, um valor pequeno de k poderia ser benéfico no início do processo de aprendizagem. Se confirmado, isso significaria que um aumento dinâmico em k deve melhorar a acurácia de tais conjuntos de dados.

A Figura 3 ilustra a influência de k em k-NN. Ela mostra os resultados de acurácia de k-NN em dois conjuntos de dados artificiais com 10,000 instâncias, a saber *LED* (Figura 3a) e *Mixed* (Figura 3b), com diferentes valores de k: 5, 10, 30, 40 e 50.

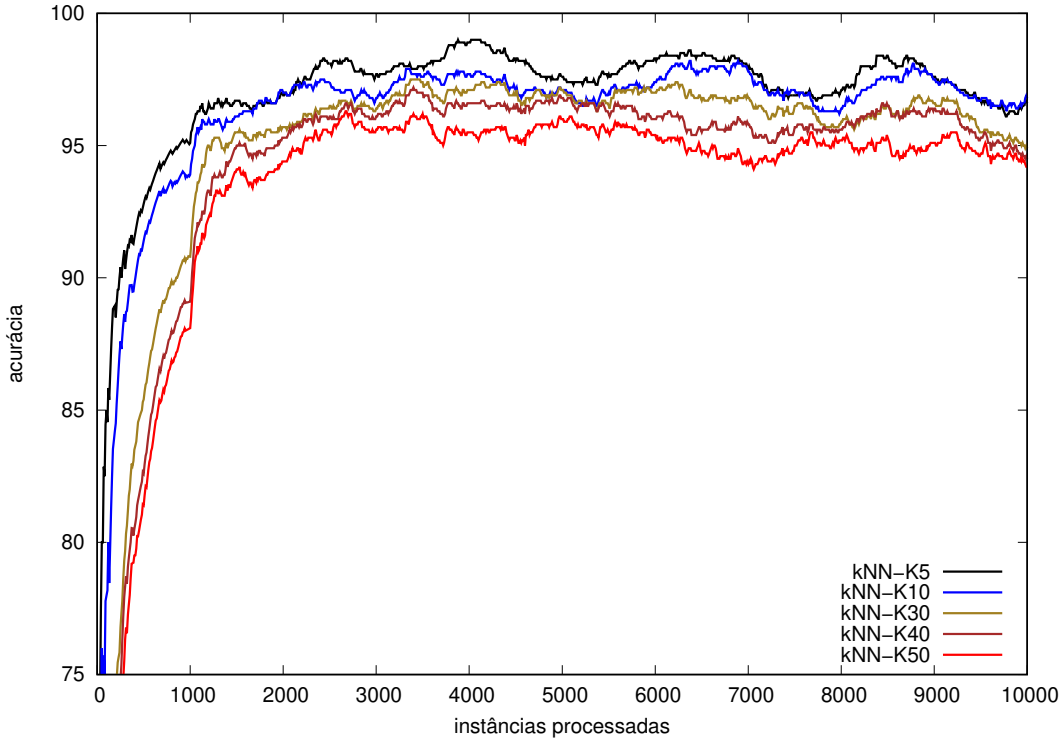
Os resultados nestes testes confirmaram que não existe um melhor valor predefinido para k, como esperado, exceto no início do processo de aprendizagem, onde os menores valores de k provaram experimentalmente ser as melhores opções, independentemente do cenário. Esta informação pode ser confirmada em ambas as figuras. Depois de um certo número de instâncias, porém, os padrões tornam-se muito diferentes: enquanto no conjunto de dados *LED*

Figura 3 – Acurácias do classificador k-NN usando diferentes valores k com os conjuntos de dados artificiais: a) LED e b) Mixed.

(a)



(b)



Fonte: O autor (2022)

valores maiores de k são progressivamente mais eficazes, em *Mixed* os valores menores de k permanecem como as melhores opções. É importante ressaltar que vários outros conjuntos de dados também foram testados e a maioria dos resultados seguiu o padrão de um dos dois casos apresentados.

Levando em consideração esta característica de imprevisibilidade do parâmetro k, nesta tese de doutorado também é proposto o Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN), um método que utiliza classificadores k-NN emparelhados com o ajuste dinâmico de k em um de seus membros. No início do processo de treinamento, ambos os classificadores usarão um pequeno valor de k. Esta decisão é baseada nos experimentos de exploração anteriormente mencionados, que confirmaram que um pequeno número de vizinhos nos estágios iniciais torna o k-NN mais preciso. Então, conforme mais instâncias são processadas, o par começa a diferir: o primeiro membro permanece inalterado, com k fixado em um valor baixo para o treinamento, enquanto o valor de k no segundo membro é aumentado progressivamente em intervalos regulares.

O último detalhe de PL-kNN refere-se ao processo de classificação: o método compara o desempenho atual de seus dois membros, levando em consideração as instâncias já processadas, e o membro com maior acurácia até este ponto fornecerá a resposta do par para a instância atual. Esse procedimento visa maximizar (aproximadamente) a resposta do método em um grande número de cenários, uma vez que não há um k ótimo em geral. A justificativa para esta decisão foi a expectativa de que PL-kNN pudesse fornecer acurácias competitivas na maioria dos conjuntos de dados, votando com k progressivamente maiores em conjuntos de dados onde k-NN se comporta como em *LED* e com um pequeno k em conjuntos de dados onde se comporta como em *Mixed*.

O algoritmo 1 apresenta o pseudo-código abstrato que descreve a implementação de PL-kNN. Além do fluxo de dados (ds) e dos classificadores k-NN emparelhados (pl), PL-kNN recebe como entradas os seguintes parâmetros, com os respectivos valores padrão: o valor mínimo k para o primeiro membro de pl ($k\_low = 5$); uma lista de valores k para o segundo membro de pl (k_list = [5, 10, 15, 20, 30, 40, 50]); e o intervalo em instâncias para as atualizações em k_2 ($interv$ = 100). Todos esses valores de parametrização padrão foram definidos com base nos experimentos preliminares.

A parte principal do código é a função *PL-kNN*, começando na linha 12. Sua primeira parte (linhas 13–16) descreve todo o processo de treinamento para os classificadores pareados. A linha 13 representa o loop que itera os classificadores dentro de PL-kNN para cada instância

Algoritmo 1: Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN)

Input: **ds**: fluxo de dados; **pl**: classificadores k-NN pareados; **k_low**: valor mínimo para o parâmetro k do primeiro membro de pl;
k_list: lista de valores para k_2; **interv**: intervalo em instâncias para atualizar k_2

acc : vetor contendo acurácias de k-NN
pl_k : vetor contendo os parâmetros k iniciais
instances: controla o número total de instâncias
idx: controla a variação do parâmetro k

Function `initialize()`**:**
    instances ← 0
    idx ← 1
    pl_k ← [k_low, k_list[0]]
    for $m = 1$ ***to*** 2 **do**
        $acc_m \leftarrow 0$
        $pl_m \leftarrow$ kNN(pl_k_m)

Function `PL-kNN(`*instance* ***x***`)`**:**
    for $m = 1$ ***to*** 2 **do**
        if $pl_m(x)$ *foi classificada corretamente* **then**
            Atualizar acurácia do membro acc_m
        Train $pl_m(x)$
    instances ← instances + 1
    if *(*$idx <$ *size of* k_list*)* ***and*** *(instances* ***mod*** *interv* $== 0$*)* **then**
        $k_2 \leftarrow k\_list_{idx}$ // O k do segundo classificador pl_2 é modificado para o próximo elemento de k_list
        $idx \leftarrow idx + 1$
    Return predição do membro com a maior acurácia

Program `()`**:**
    initialize()
    foreach *instance* x ***in*** ds **do**
        PL-kNN(x)
        $driftDetected \leftarrow$ ChangeDetector(x) // Detector acusa mudança de conceito?
        if *driftDetected* **then**
            initialize()

que chega no processo de execução. Logo, desde que a instância seja classificada corretamente (linha 14), a acurácia de cada membro do par é atualizada (linha 15). Observe que os resultados de acurácia são usados no processo de decisão do método. Na linha 16, os classificadores são treinados.

A parte mais importante do método proposto é descrita nas linhas 17 -21. Inicialmente, a variável $instances$ atualiza o número de instâncias processadas (linha 17). Os comandos que representam a mudança no valor de k_2, o k do segundo membro, são representados nas linhas 18 - 20 do pseudocódigo. A modificação dinâmica de k_2 será realizada a cada $interv$ instâncias até chegar ao final de k_list. Aqui pode-se observar que:

- apenas o segundo classificador (pl_2) terá seu k modificado.
- o resultado da função é a predição do membro de PL-kNN com melhor desempenho de acurácia de classificação até a última instância processada, chegando à sua decisão final (linha 21).

Para uma melhor compreensão, é valido considerar o seguinte exemplo. Assumindo que os parâmetros têm valores padrão, ou seja, k_list = [5, 10, 15, 20, 30, 40, 50] e $interv$ = 100, na instância 100 k_2 será alterado para 10, o segundo elemento de k_list, visto que o primeiro era seu valor inicial. Após outras 100 instâncias ($instances$ = 200), este procedimento acontece novamente e k_2 é alterado para 15, e isso é repetido até que k_2 se torne 50, o último elemento de k_list.

Finalmente, é importante destacar que o fluxo de execução do algoritmo em geral, é controlado pelo programa principal detalhado no último trecho do pseudocódigo entre as linhas 22 e 28. Para cada instância do fluxo de dados (linha 24), o programa chama repetidamente a função *PL-kNN* (linha 25). Adicionalmente, ele controla a execução de um método de detecção de mudança de conceito (linha 26) e, quando uma mudança de conceito é detectada (linha 27), todas as variáveis são inicializadas por uma ativação extra (linha 28) da função *initialize*, descrita nas linhas 5–11, e, assim, o processo de atualização começa do zero para cada novo conceito no fluxo de dados.

Além disso, nesta pesquisa também são introduzidos os métodos Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 2 (PL-kNN$_2$), PL-kNN$_3$ e PL-kNN$_4$, versões de PL-kNN que utilizam estratégias de implementação diferentes para ajustar os valores k do par de classificadores k-NN usados nos algoritmos propostos.

3.1.1 Métodos baseados em Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN)

Como foi mencionado na seção anterior, a seguir também são apresentados outras versões do PL-kNN, desta vez utilizando outras estratégias de implementação com base no ajuste dinâmico dos k vizinhos em ambos os membros k-NN usados no método de classificação pareado. Neste sentido, estas modificações foram feitas com o intuito de considerar outras situações de ajuste do parâmetro k onde a performance do método pode mudar e, por conseguinte, obter resultados benéficos em outros contextos de teste experimental.

Além disso, é válido destacar que nos pseudocódigos dos algoritmos referentes às outras versões do PL-kNN, apresentados nas próximas subseções, as diferenças em relação ao algoritmo original foram marcadas com um `tipo de fonte diferente`.

3.1.1.1 Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 2 (PL-kNN$_2$)

Diferente de PL-kNN, em PL-kNN$_2$ além de ajustar o parâmetro k do segundo membro de classificação no método, a ideia seria variar também de forma dinâmica o k do primeiro classificador do ensemble para o k anterior pertencente ao segundo classificador, seguindo o critério de desempenho entre as acurácias dos membros no método (somente se a acurácia do primeiro membro entre os classificadores k-NN for menor do que a mesma métrica do segundo membro).

No Algoritmo 2 apresenta-se o pseudocódigo abstrato do PL-kNN$_2$. Inicialmente, o método recebe como entrada os parâmetros que foram detalhados e inicializados com os mesmos valores anteriormente descritos no Algoritmo 1 em PL-kNN. Na função *PL-kNN2* (linhas 12 - 23) são descritas as novas estratégias de implementação de PL-kNN$_2$. Como pode se apreciar, entre as linhas 13 e 16 é descrito o processo de treinamento e atualização das acurácias dos classificadores pareados a cada instância que chega no processo de execução.

Posteriormente, na parte mais relevante da função (linhas 17 - 23), inicialmente são atualizadas o número de instâncias processadas (linha 17). Note que entre as linhas 18 -22 são descritas as sintaxes que representam as mudanças nos valores k_1 e k_2 de ambos os membros pareados do método (pl_1 e pl_2). Aqui é válido ressaltar que as mudanças de k nos classificadores é realizado em cada intervalo de instâncias $interv$ até o final de k_list (linha 18). O ajuste dinâmico de k_1 (k do primeiro membro) acontece se a acurácia pertencente ao seu respectivo

classificador acc_{pl_1} for menor do que a acurácia do segundo membro acc_{pl_2} (linha 19). Nesse sentido, o valor k_1 de pl_1 varia para o item anterior ao k_2 (k do segundo membro) como pode ser apreciado na linha 20. Seguidamente, a modificação dinâmica de k_2 é realizada da mesma forma como acontece no Algoritmo 1 em PL-kNN, o seu valor é modificado para o próximo elemento de k_list (linha 21). Na parte final da função (linha 23), o critério de decisão retorna o membro entre os classificadores pareados com melhor desempenho de acurácia.

Considerando os valores padrão do exemplo de PL-kNN explicado na seção 3.1 e assumindo que na instância 200 a acurácia do primeiro membro acc_{pl_1} é menor do que a acurácia de segundo membro acc_{pl_2}, quando o valor de $k_2 = 10$ for alterado para 15 (o terceiro elemento de k_list), o valor de k_1 nesta instância também será alterado para 10, o elemento anterior ao k_2. Após outras 100 instâncias ($instances = 300$) se o procedimento se comportar da mesma forma o valor de k_2 vai ser 20 e o valor de k_1 será alterado para 15, e esse processo em que os valores de k_1 e k_2 variam acontece até k_2 obter o último elemento de k_list ($k_2 = 50$).

Por último, note que a diferença no fluxo de execução do algoritmo geral monitorado pelo programa entre as linhas 24- 30 em comparação com o fluxo na descrição do Algoritmo 1 , é a chamada da função $PL\text{-}kNN2(x)$ na linha 27.

3.1.1.2 Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 3 (PL-kNN$_3$)

Em Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 3 (PL-kNN$_3$) também são ajustados de forma dinâmica e incremental ambos os parâmetros k dos membros de classificação pareados no método. A diferença neste caso é a variação do valor do número de vizinhos k pertencente ao primeiro membro (k_1) entre os k-NN pareados que é feita de um em um seguindo o igual critério condicional entre as acurácias apresentado no Algoritmo 2.

No Algoritmo 3 é exibido o pseudocódigo abstrato deste método. Na parte inicial relacionada com a entrada de parâmetros, além dos vetores acc, pl_k e a variável $instances$ que controla o número total de instâncias, note que nas linhas 4 e 5 foram adicionadas idx_fisrt e idx_second com objetivo de controlar mais adiante na função principal *PL-kNN3* a variação do k em cada um dos membros pl_1 e pl_2. Estas variáveis, também são inicializadas posteriormente dentro da função $initialize$ (linhas 6 - 12).

Além disso, aludindo a função *PL-kNN3* descrita entre as linhas 13 e 25, após o processo de treinamento e atualização das acurácias pl_1 e pl_2 que atende as considerações da condição

Algoritmo 2: Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 2 (PL-kNN$_2$)

Input: **ds**: fluxo de dados; **pl**: classificadores k-NN pareados; **k_low**: valor mínimo para o parâmetro k dos membros de pl;
k_list: lista de valores para k_1 e k_2; **interv**: intervalo em instâncias para atualizar k_1 e k_2

acc : vetor contendo acurácias de k-NN
pl_k : vetor contendo os parâmetros k iniciais
instances: controla o número total de instâncias
idx: controla a variação do parâmetro k

Function `initialize()`:
    instances ← 0
    idx ← 1
    pl_k ← [k_low, k_list[0]]
    for $m = 1$ ***to*** 2 **do**
        $acc_m \leftarrow 0$
        $pl_m \leftarrow$ kNN(pl_k_m)

Function `PL-kNN2(`*instance* ***x***`)`:
    for $m = 1$ ***to*** 2 **do**
        if $pl_m(x)$ *foi classificada corretamente* **then**
            Atualizar acurácia do membro acc_m
        Train $pl_m(x)$
    instances ← instances + 1
    if *(*$idx <$ *size of* k_list*)* ***and*** *(instances* ***mod*** *interv* $==$ *0)* **then**
        if *(*$\texttt{acc}_{\texttt{pl}_1} < \texttt{acc}_{\texttt{pl}_2}$*)* **then**
            $\texttt{k}_1 \leftarrow \texttt{k\_list}_{\texttt{idx}-1}$ `// O k do primeiro classificador` $\texttt{pl}_1$ `é modificado para o elemento anterior de` $\texttt{k}_2$ `localizado em` $\texttt{k\_list}_{\texttt{idx}-1}$
        $k_2 \leftarrow k\_list_{idx}$ // O k do segundo classificador pl_2 é modificado para o próximo elemento de k_list
        $idx \leftarrow idx + 1$
    Return predição do membro com a maior acurácia

Program `()`:
    initialize()
    foreach *instance* x ***in*** ds **do**
        `PL-kNN2(`x`)`
        $driftDetected \leftarrow$ ChangeDetector(x) // Detector acusa mudança de conceito?
        if *driftDetected* **then**
            initialize()

de classificação das instâncias (se $pl_m(x)$ for corretamente classificada) localizado entre as linhas 14 e 17, e a atualização do número de instâncias processadas na linha 18, note que o critério de parada para variar os valores de k nos membros vai até o final de k_list quando o k_2 pertencente ao segundo membro de classificação do método obtenha o valor do último elemento da lista de valores no intervalo de instâncias $interv$ na linha 19.

Depois, nas linhas seguintes onde acontecem as variações dos k números de vizinhos dos membros de classificadores, o k_1 do primeiro classificador pl_1 é modificado para o próximo elemento de $k\_list_{idx\_first}$ e seguidamente o valor de idx_fisrt é incrementado. Assim, o k_2 pertencente a pl_2 é variado para o próximo elemento de $k\_list_{idx\_second}$ e, a variável idx_second é atualizada a seguir (linhas 23 e 24). Já na parte final da função (linha 25), é retornado o membro do método com o melhor desempenho segundo a acurácia de classificação.

No exemplo a seguir pode se obter um melhor entendimento. Atribuindo os mesmos valores padrão dos exemplos anteriores aos parâmetros $k\_list = [5, 10, 15, 20, 30, 40, 50]$ e $interv =$ 100, na instância 100 supondo que ($acc_{pl_1} < acc_{pl_2}$), o k_1 que tinha o valor do primeiro elemento de k_list ($k_1 = 5$) vai ser modificado para o segundo elemento de k_list, ou seja, $k_1 = 10$ e o k_2 também varia para esse mesmo valor do segundo elemento de k_list. Após outras 100 instâncias ($instances = 200$), se acc_{pl_1} for novamente menor que acc_{pl_2}, ambos os valores de k_1 e k_2 serão alterados para 15. Caso contrario, o k_2 do segundo membro vai ser o único modificado para 15 e o k_1 ficaria com o valor de 10. Essas variações vão continuar no processo de execução até o k_2 receber 50, o último elemento de k_list.

Por outro lado, igualmente como acontece nos algoritmos das subseções anteriores, no fluxo de execução geral (linhas 26 – 32) é feita a chamada da função principal (neste caso a chamada de *PL-kNN3* na linha 29).

3.1.1.3 Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 4 (PL-kNN$_4$)

Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 4 (PL-kNN$_4$) foi implementado de forma semelhante ao PL-kNN$_3$, ambos os k_1 e k_2 pertencentes aos membros k-NN do método são ajustados de forma dinâmica. No caso de k_1, a variação é feita um a um levando em consideração o critério entre as acurácias dos membros pareados de PL-kNN$_4$ caso a acurácia do primeiro membro entre os classificadores seja menor do que a acurácia do segundo membro como acontece em PL-kNN$_2$ e PL-kNN$_3$, métodos anteriormente explicados nos Algoritmos 2 e 3. Porém, a particularidade que torna o PL-kNN$_4$ diferente de PL-kNN$_3$ é

Algoritmo 3: Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 3 (PL-kNN$_3$)

Input: ds: fluxo de dados; **pl**: classificadores k-NN pareados; **k_low**: valor mínimo para o parâmetro k dos membros de pl;
k_list: lista de valores para k_1 e k_2; **interv**: intervalo em instâncias para atualizar k_1 e k_2

acc : vetor contendo acurácias de k-NN
pl_k : vetor contendo os parâmetros k iniciais
instances: controla o número total de instâncias
`idx_first`: controla a variação do parâmetro k do primeiro classificador $\texttt{pl}_1$
`idx_second`: controla a variação do parâmetro k do segundo classificador $\texttt{pl}_2$

Function `initialize()`:
    instances ← 0
    `idx_first, idx_second` ← 1
    pl_k ← [k_low, k_list[0]]
    for $m = 1$ ***to*** 2 **do**
        $acc_m \leftarrow 0$
        $pl_m \leftarrow \text{kNN}(pl\_k_m)$

Function `PL-kNN3(`*instance* ***x***`)`:
    for $m = 1$ ***to*** 2 **do**
        if $pl_m(x)$ *foi classificada corretamente* **then**
            Atualizar acurácia do membro acc_m
        Train $pl_m(x)$
    instances ← instances + 1
    if *(*`idx_second` $<$ *size of* k_list*)* ***and*** *(instances* ***mod*** *interv* $== 0$*)* **then**
        if *(*$\texttt{acc}_{\texttt{pl}_1} < \texttt{acc}_{\texttt{pl}_2}$*)* **then**
            $\texttt{k}_1 \leftarrow \texttt{k\_list}_{\texttt{idx\_first}}$ `// O k do primeiro classificador` $\texttt{pl}_1$ `é modificado para o próximo elemento de` $\texttt{k\_list}_{\texttt{idx\_first}}$
            `idx_first` ← `idx_first` + 1
        $\texttt{k}_2 \leftarrow \texttt{k\_list}_{\texttt{idx\_second}}$ `// O k do segundo classificador` $\texttt{pl}_2$ `é modificado para o próximo elemento de` $\texttt{k\_list}_{\texttt{idx\_second}}$
        `idx_second` ← `idx_second` + 1
    Return predição do membro com a maior acurácia

Program `()`:
    initialize()
    foreach *instance* x ***in*** ds **do**
        `PL-kNN3(`x`)`
        $driftDetected \leftarrow$ ChangeDetector(x) // Detector acusa mudança de conceito?
        if *driftDetected* **then**
            initialize()

o critério de parada no intervalo onde os valores dos k vizinhos pertencentes aos membros do método pareado variam. Nesse sentido, as variações de k_1 são as que determinam a parada do processo quando chegar à penúltima posição da lista de valores para os k vizinhos determinada antes do inicio da execução do método. O propósito principal neste contexto é experimentar outra estratégia de execução devido a que as mudanças de k podem demorar mais a terminar neste caso e consequentemente, influenciar no resultado final do método.

A seguir, no pseudocódigo do Algoritmo 4, são detalhados os argumentos anteriormente explicados. Inicialmente, como nos algoritmos anteriores o método recebe como entrada o fluxo de dados ds, os classificadores pareados k-NN em pl, k_low que representa o menor valor para o parâmetro k dos membros de pl, k_list com os valores para k_1 e k_2 e o intervalo em instâncias $interv$ para realizar os ajustes dinâmicos em k_1 e k_2. Aqui também foram adicionadas as variáveis idx_first e idx_second que controlam a variação do parâmetro k nos membros pl_1 e pl_2 na função principal *PL-kNN4*.

Nesta função *PL-kNN4* detalhada entre as linhas 13 e 26, primeiramente é descrita a parte do treinamento dos classificadores considerando a classificação da instância processada para depois atualizar a acurácia de cada membro do enfoque pareado (linhas 14 – 17). Seguidamente, $instance$ é atualizada na linha 18. Depois é descrita a parte mais importante da função (linhas 19 até 25). Aqui é válido observar a expressão condicional da linha 19 que fundamenta a diferença do termo de parada de PL-kNN$_4$ em relação ao PL-kNN$_3$. Assim, a variação dos membros k_1 e k_2 acontecem até k_1 obter o valor da penúltima posição da lista de valores (k_list - 1) demorando cada vez mais o ajuste dinâmico dos k vizinhos pertencentes ao par de classificadores k-NN do método. Na parte final da função é retornado o membro entre os classificadores pareados de PL-kNN$_4$ com a maior acurácia (linha 26).

Por último, entre as linhas 27 e 33 denota se o processo de execução do PL-kNN$_4$ em geral monitorado pelo programa principal como acontece nos métodos anteriores chamando repetidamente a função principal *PL-kNN4* e considerando o procedimento de atualização do método para cada mudança de conceito detectada no fluxo de dados.

3.1.2 Complexidade Computacional do PL-kNN

Para medir a complexidade computacional do PL-kNN e as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, é válido ressaltar que a implementação do algoritmo de classificação k-NN tradicional é linear e adaptada para o aprendizado com fluxo de dados (BIFET et al., 2018). Esta

Algoritmo 4: Paired k-NN Learners with Dynamically Adjusted Number of Neighbors 4 (PL-kNN$_4$)

Input: **ds**: fluxo de dados; **pl**: classificadores k-NN pareados; **k_low**: valor mínimo para o parâmetro k dos membros de pl;
k_list: lista de valores para k_1 e k_2; **interv**: intervalo em instâncias para atualizar k_1 e k_2

acc : vetor contendo acurácias de k-NN
pl_k : vetor contendo os parâmetros k iniciais
instances: controla o número total de instâncias
`idx_first`: controla a variação do parâmetro k do primeiro classificador $\texttt{pl}_1$
`idx_second`: controla a variação do parâmetro k do segundo classificador $\texttt{pl}_2$

Function `initialize()`:
    instances ← 0
    `idx_first, idx_second` ← 1
    pl_k ← [k_low, k_list[0]]
    for $m = 1$ ***to*** 2 **do**
        $acc_m \leftarrow 0$
        $pl_m \leftarrow \text{kNN}(pl\_k_m)$

Function `PL-kNN4(`*instance* ***x***`)`:
    for $m = 1$ ***to*** 2 **do**
        if $pl_m(x)$ *foi classificada corretamente* **then**
            Atualizar acurácia do membro acc_m
        Train $pl_m(x)$
    instances ← instances + 1
    if *(*`idx_first` $<$ `size of k_list` $- 1$*)* ***and*** *(instances* ***mod*** *interv* $== 0$*)* **then**
        if *(*$\texttt{acc}_{\texttt{pl}_1} < \texttt{acc}_{\texttt{pl}_2}$*)* **then**
            $\texttt{k}_1 \leftarrow \texttt{k\_list}_{\texttt{idx\_first}}$ `// O k do primeiro classificador` $\texttt{pl}_1$ `é modificado para o próximo elemento de` $\texttt{k\_list}_{\texttt{idx\_first}}$
            `idx_first` ← `idx_first` + 1
        if *(*`idx_second` $<$ `size of k_list`*)* **then**
            $\texttt{k}_2 \leftarrow \texttt{k\_list}_{\texttt{idx\_second}}$ `// O k do segundo classificador` $\texttt{pl}_2$ `é modificado para o próximo elemento de` $\texttt{k\_list}_{\texttt{idx\_second}}$
            `idx_second` ← `idx_second` + 1
    Return predição do membro com a maior acurácia

Program `()`:
    initialize()
    foreach *instance* x ***in*** ds **do**
        `PL-kNN4(`x`)`
        $driftDetected \leftarrow$ ChangeDetector(x) // Detector acusa mudança de conceito?
        if *driftDetected* **then**
            initialize()

implementação também usa uma *heap* para manter as k distâncias mais curtas à medida que são calculadas. Neste cenário, k-NN terá uma complexidade computacional na ordem de $\mathcal{O}(w \times (d + log\ k))$ para a classificação e $\mathcal{O}(1)$ para treinamento, onde w é o tamanho da janela deslizante, d é o número de atributos e k é o número de vizinhos mais próximos considerados.

Desta forma, sabendo que as iterações de PL-kNN e suas versões acontecem através do par de classificadores para atualizar a acurácia, a classificação nestes métodos terá um desempenho assintótico de $\mathcal{O}(2 \times w \times (d + log\ k))$. Por outro lado, como no k-NN, o treinamento terá uma complexidade computacional constante.

3.2 PARAMETER ESTIMATION PROCEDURE (PEP)

O ajuste dinâmico de parâmetros em comitês de classificadores tem sido abordado na literatura de diferentes maneiras, incluindo propostas de ambiente. Os exemplos incluem votos ponderados dinamicamente de classificadores individuais em um ensemble (RIJN et al., 2018); classificação de fluxos textuais com base em uma série de Clustering Trees (CTs), usando uma estratégia de conjunto adaptativo para escolher dinamicamente CTs de acordo com as propriedades de um fluxo de dados (SONG et al., 2016a); encontrar e adaptar automaticamente os parâmetros ideais para fluxos de dados em algoritmos de agrupamento (CARNEIN et al., 2019); e aprendizagem ativa com uma estratégia de rotulagem híbrida para ajustar dinamicamente o limite de decisão de uma estratégia de incerteza após mudanças de conceito (SHAN et al., 2018).

De forma explícita, a maioria desses trabalhos concentra seus esforços no ajuste de um parâmetro específico, ou seja, esse ajuste dinâmico geralmente *não* é realizado de forma genérica que poderia ser adaptada para estimar ou ajustar outros parâmetros de diferentes abordagens. A proposta a seguir, por outro lado, implementa um método genérico que pode ser usado para ajustar outros parâmetros em métodos diferentes, não apenas ensembles, usando a mesma estratégia de implementação.

Esta seção descreve Parameter Estimation Procedure (PEP), a estratégia proposta para ajustar dinamicamente parâmetros de métodos relacionados a fluxos de dados. Como parte fundamental desta investigação, este procedimento é aplicado em três comitês de classificadores para ajustar o parâmetro de diversidade (λ), comum em várias abordagens de comitês.

A ideia principal do PEP é usar uma amostra inicial de cada conceito do fluxo de dados para executar um método testando diferentes valores de parâmetros e escolher aquele que tem

melhor desempenho para ser usado após este período até que uma mudança de conceito seja detectada.

Nos casos de estudo com comitês de classificadores, a amostra inicial de cada conceito é usada para treinar T conjuntos de classificadores, cada conjunto com um valor diferente de λ - o parâmetro escolhido para ser otimizado. O valor λ associado com o melhor desempenho do comitê neste período é selecionado e usado nas instâncias restantes do conceito atual.

O algoritmo 5 detalha como essa ideia foi implementada. Inicialmente, ele recebe como entrada o classificador c, os limites padrão (λ^{def}), inferior (λ^{min}) e superior (λ^{max}) para λ, definindo o intervalo escolhido usado na estimação, o número de comitês (T) que serão testados com diferentes valores de λ e, por fim, o número de instâncias de amostra (si) usado para medir o desempenho dos comitês.

Algoritmo 5: Specific Parameter Estimation Procedure (PEP) for λ optimization

Input: c: classificador; λ^{def}: valor padrão para diversidade (λ); λ^{min}: λ inferior; λ^{max}: λ superior; T: número de comitês com diferentes valores de λ; si: período de instâncias

1 H: vetor do comitê de classificadores; n: instâncias processadas; λ^{aux}: diversidade

2 **Function** `initialize()`**:**

3 $n \leftarrow 0$

4 $\lambda^{aux} \leftarrow \lambda^{min}$

5 **for** $t \leftarrow 1$ **to** T **do**

6 ▷ **Cria comitês de classificadores c usando diferentes valores de λ**

7 $H_t \leftarrow$ createEnsemble(c, λ^{aux})

8 $\lambda^{aux} \leftarrow \lambda^{aux} + (\lambda^{max} - \lambda^{min} + 1) \,/\, T$

9 **Function** `getBestLambda`(x: *instance*)**:**

10 $n \leftarrow n + 1$

11 **if** $n < si$ **then**

12 **for** $t \leftarrow 1$ **to** T **do**

13 Train $H_t(x)$ ▷ **Cada comitê está associado a um valor diferente de λ**

14 **return** λ^{def}

15 **else**

16 **return** *o valor λ associado ao membro H mais preciso*

Observe que a linha 1 cria o vetor H do comitê de classificadores e variáveis auxiliares para controlar o número de instâncias e a diversidade dos comitês. As linhas 2 a 8 descrevem a função *initialize*, responsável pela inicialização de PEP. É valido ressaltar que cada posição de H armazena um comitê de classificadores usando diferentes valores lambda (linha 7) e que

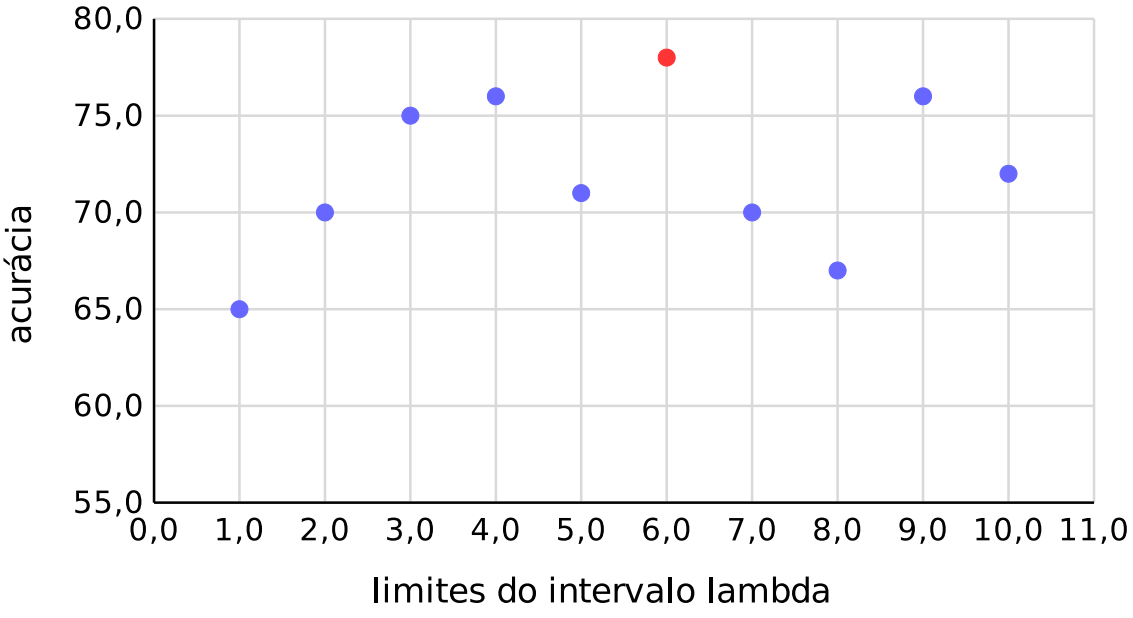
o código da linha 8 garante que esses valores lambda sejam distribuídos proporcionalmente dentro do intervalo delimitado por λ^{min} e λ^{max}.

A parte mais importante da implementação é detalhada nas linhas 9 - 16. Observe que a função $getBestLambda$ recebe sucessivamente as instâncias processadas (x) em seu parâmetro, uma instância por vez, e é responsável por retornar o melhor valor λ. Ao longo de T iterações (linha 12), os comitê são treinados com a instância atualmente processada (linha 13). Importante enfatizar que cada comitê (H_t) está associado a diferentes valores λ, conforme definido na função *initialize*.

A definição do melhor λ só ocorre após um determinado período (si instâncias), como pode ser visto na condição da linha 11. Após este período, nenhum processamento adicional será realizado e o valor λ retornado será aquele associado ao comitê com a melhor acurácia (linha 16). É importante destacar que, enquanto o melhor valor estimado não é encontrado, o valor considerado *default* para λ no comitê modificado (λ^{def}) é retornado (linha 14).

Outra maneira mais explícita de entender a implementação de PEP é ilustrada na representação gráfica de dispersão mostrada na Figura 4. Esta figura representa os resultados do procedimento usando 10 comitês com diferentes valores λ variando de 1,0 a 10,0 no processo de aprendizagem: a acurácia do comitê associado a cada λ é representada por um círculo e o círculo vermelho ilustrado com o valor 6,0 representa o λ retornado por PEP, que é o que tem a melhor acurácia.

Figura 4 – Representação gráfica da execução de PEP, usando os limites: $\lambda^{min} = 1.0$ e $\lambda^{max} = 10.0$



Fonte: O autor (2022)

As subseções a seguir detalham o uso de PEP em diferentes comitês de classificadores. Como mencionado anteriormente, BOLE$_5$, OABM1, e OzaBag foram escolhidos para estimar dinamicamente a diversidade λ, gerando suas respectivas novas versões BOLE-PE, OABM1-PE, e OzaBag-PE, usadas nos testes do Capítulo 6.

3.2.1 Boosting-like Online Learning Ensemble with Parameter Estimation (BOLE-PE)

BOLE-PE é basicamente BOLE com a inserção de PEP para estimar o parâmetro λ dinamicamente, em vez de usar seu valor padrão original 1.0 no primeiro membro processado do comitê em cada instância. No Algoritmo 6 apresenta-se o pseudocódigo referente à implementação do BOLE-PE.

Algoritmo 6: Boosting-like Online Learning Ensemble with Parameter Estimation (BOLE-PE)

Input: h: comitê; M: tamanho do comitê; x: instância; N: número de instâncias processadas;

```
1  minPos ← 1; maxPos ← M; correct ← false
2  λ^sc ← λ^sw ← 0.0
3  ordenar h por acurácia em ordem ascendente
4  λ ← PEP.getBestLambda(x)
5  for m ← 1 to M do
6      if correct then
7          pos ← maxPos
8          maxPos ← maxPos - 1
9      else
10         pos ← minPos
11         minPos ← minPos + 1
12     k ← Poisson(λ)
13     Train h_pos(x) using k
14     λ_m^sc ← λ_m^sc + λ × [[h_pos(x) = y]]
15     λ_m^sw ← λ_m^sw + λ × [[h_pos(x) ≠ y]]
16     λ ← λ (N/2.λ_m^sc) × [[h_pos(x) = y]] + λ (N/2.λ_m^sw) × [[h_pos(x) ≠ y]]
17     correct ← [[h_pos(x) = y]]
   Final Hypothesis Output:
18 H(x) ← arg max_{y∈Y} Σ_{m=1}^{M} (log 1/β_m) × [[h_m(x) = y]]
```

$[[h_{pos}(x) = y]]$ e $[[h_{pos}(x) \neq y]]$ são booleanos que retornarão 1 ou 0 se suas condições forem verdadeiras ou falsas, respectivamente.

Inicialmente, o comitê de classificadores (h) é ordenado em ordem crescente pela acurácia. Antes, várias variáveis são inicializadas, incluindo $minPos$ e $maxPos$ com valores que repre-

sentam o classificador com a pior e a melhor acurácia, respectivamente, bem como λ^{sc}, λ^{sw} e *correct* (linhas 1-2). Posteriormente, o valor de λ, responsável pela diversidade em BOLE-PE, recebe o valor retornado pela função $getBestLambda$ de PEP com a instância atual (x) passada como parâmetro (linha 4).

Os próximos passos do algoritmo permanecem os mesmos que foram originalmente propostos em (BARROS; SANTOS; GONÇALVES JR., 2016). Em geral, quando uma nova instância chega, o especialista com a pior acurácia é inicialmente selecionado. Se a instância for classificada corretamente, os outros especialistas, que são mais precisos, também terão uma boa chance de classificá-la corretamente. Assim, o erro de outro especialista com melhor acurácia é considerado um erro improvável (linhas 6 a 17). Uma vez que λ será reduzido quando a classificação for feita corretamente e aumentado quando for incorreta, para minimizar as consequências de um erro improvável, o próximo especialista selecionado será aquele com a melhor acurácia e assim por diante (linhas 14 a 17).

3.2.2 Online AdaBoost-based M1 with Parameter Estimation (OABM1-PE)

Similarmente, OABM1-PE é baseado em OABM1, aprendendo com novas instâncias sem perder as características originais de sua versão offline correspondente AdaBoost.M1 (FREUND; SCHAPIRE, 1997). O método continua com a ideia de aumentar ou diminuir o peso das instâncias conforme as predições falham ou são corretas, respectivamente, mas OABM1-PE inclui a estratégia PEP, descrita no Algoritmo 5, para ajustar o parâmetro de diversidade (λ) dinamicamente. O pseudocódigo de OABM1-PE é mostrado no Algoritmo 7 e a estimativa de λ é feita na linha 2 – em OABM1, este parâmetro de diversidade é definido como 1 por padrão.

A partir da linha 3 em diante, as características originais do método (SANTOS; BARROS, 2019) são mantidas. A abordagem visa ser o mais próximo possível de AdaBoost.M1. Resumidamente, cada um dos classificadores do ensemble é visitado e treinado de acordo com o peso atual (λ): quanto maior o λ, maior a intensidade de treinamento no classificador h_m.

3.2.3 Oza and Russell's Online Bagging with Parameter Estimation (OzaBag-PE)

O processo de aprendizagem em OzaBag-PE é realizado de forma semelhante ao OzaBag original. O ensemble adota uma distribuição de Poisson para simular o comportamento do tradicional bagging (versão batch). Como nos dois métodos anteriores, o treinamento do

Algoritmo 7: Online AdaBoost-based M1 with Parameter Estimation (OABM1-PE)

Input: h: comitê; M: tamanho do comitê; x: instância

1 $\lambda^{sc} \leftarrow \lambda^{sw} \leftarrow 0.0$
2 $\lambda \leftarrow$ **PEP**.getBestLambda(x)
3 **for** $m = 1$ ***to*** M **do**
4 $\quad k \leftarrow \text{Poisson}(\lambda)$
5 $\quad$ Train $h_m(x)$ using k
6 $\quad \lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda \times [[h_m(x) = y]]$
7 $\quad \lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda \times [[h_m(x) \neq y]]$
8 $\quad cw_m \leftarrow \lambda_m^{sc} + \lambda_m^{sw}$
9 $\quad \varepsilon_m \leftarrow \lambda_m^{sw} / cw_m$
10 $\quad$ **if** $\varepsilon_m > 1/2$ **then**
11 $\quad\quad$ break
12 $\quad \beta_m \leftarrow \varepsilon_m / (1 - \varepsilon_m)$
13 $\quad \lambda \leftarrow (\lambda \times \beta_m^{1-[[h_m(x) \neq y]]}) / (\frac{\lambda_m^{sc}}{cw_m} \times \beta_m^1 + \frac{\lambda_m^{sw}}{cw_m} \times \beta_m^0)$

Final Hypothesis Output:

14 $\mathcal{H}(x) \leftarrow arg\max_{y \in Y} \sum_{m=1}^{M} \left(log \frac{1}{\beta_m} \right) \times [[h_m(x) = y]]$

[[$h_m(x) = y$]] e [[$h_m(x) \neq y$]] são booleanos que retornarão 1 ou 0 se suas condições forem verdadeiras ou falsas, respectivamente.

comitê usa a distribuição de Poisson e seu valor inicial é definido por PEP. No entanto, por ser uma abordagem baseada em bagging, o valor de λ *não* variará ao longo da iteração em M. Essas características são detalhadas no Algoritmo 8.

Algoritmo 8: Oza and Russell's Online Bagging with Parameter Estimation (OzaBag-PE)

Input: h: comitê; M: tamanho do comitê; x: instância

1 $\lambda \leftarrow$ **PEP**.getBestLambda(x)
2 **for** $m = 1$ ***to*** M **do**
3 $\quad k \leftarrow \text{Poisson}(\lambda)$
4 $\quad$ Train $h_m(x)$ using k

Final Hypothesis Output:

5 $\mathcal{H}(x) \leftarrow arg\max_{y \in Y} \sum_{m=1}^{M} [[h_m(x) = y]]$

[[$h_m(x) = y$]] é um booleano que retornará 1 ou 0 se sua condição for verdadeira ou falsa, respectivamente.

Outra diferença em relação às outras abordagens está associada à saída. Como pode ser visto na linha 5, a votação dos membros do ensemble é feita pela estratégia do voto majoritário. Em outras palavras, tanto em OzaBag-PE quanto em sua versão original, a votação ponderada não é realizada como é o caso com as abordagens de boosting explicadas anteriormente.

3.2.4 Complexidade Computacional do PEP

Neste trabalho, a analise da complexidade computacional da técnica genérica PEP é detalhada mediante as duas fases do processo de execução do método: treinamento e classificação. Primeiramente, para avaliar a complexidade de treinamento do método, é preciso descrever a complexidade computacional do classificador base HT usado no treinamento dos comitês de classificadores com diferentes valores de lambda. Desse modo, a complexidade de treinamento de HT é $\mathcal{O}(C \times D \times V)$, onde C é quantidade de classes, D é quantidade de atributos e V o número máximo de valores por atributo (BARROS; SANTOS; BARDDAL, 2022). Porém, considerando que outros classificadores podem ser utilizados pelo método, para fins de exemplificação vamos considerar que a complexidade do treinamento de um classificador arbitrário é de $\mathcal{O}(\Phi)$.

Além do classificador, para definir a complexidade do PEP é necessário também que seja analisada a complexidade dos comitês de classificadores (BOLE, OABM1, OzaBag) utilizados para ajustar de forma dinâmica a diversidade λ, parâmetro comum destes métodos. Logo, a complexidade de treinamento dos três comitês é $\mathcal{O}(M)$, sendo M as iterações que cada ensemble faz para treinar cada um de seus classificadores membros (SANTOS, 2019). No entanto, é possível que outros comitês sejam utilizados pelo PEP. Assim, de forma semelhante ao que foi considerado no classificador, definimos também uma complexidade arbitrária para o treinamento de um comitê como $\mathcal{O}(\Xi)$. Portanto, a complexidade do comitê incluindo o classificador seria $\mathcal{O}(\Xi \times \Phi)$.

Uma vez detalhadas as complexidades do classificador base e os comitês de classificadores, podemos dizer que a complexidade geral de treinamento do PEP é $\mathcal{O}(T \times \Xi \times \Phi)$, onde T é a quantidade de iterações realizadas por PEP para treinar cada um dos comitês de classificadores usando diferentes valores de λ. Porém, na maioria dos cenários, o PEP terá uma complexidade de $\Theta(\Xi \times \Phi)$, quando não estiver no período onde o método faz a escolha do melhor lambda.

Por outro lado, para analisar a complexidade de classificação de PEP, também é preciso considerar as complexidades do classificador HT e os comitês BOLE, OABM1, OzaBag. Nesse sentido, sabendo que o HT foi utilizado para classificação, a complexidade dele é $\mathcal{O}(C \times D)$. Porém, considerando que qualquer outro classificador pode ser utilizado com outras complexidades, pode-se definir de forma genérica que a complexidade de classificação é $\mathcal{O}(\Phi)$.

No casos dos comitês BOLE, OABM1, OzaBag a complexidade de classificação de cada um deles é $\mathcal{O}(M)$ sendo M a quantidade de membros de cada um dos comitês de classificadores que serão iterados para a atribuição dos pesos (linhas 18, 14 e 5 dos Algoritmos 6, 7 e 8,

respectivamente) (SANTOS; BARROS, 2019). Porém, considerando que outros comitês podem ser utilizados com particularidades de complexidades diferentes, pode-se determinar que a complexidade genérica da classificação de um comitê é O(Ψ).

Por fim, uma vez analisadas as complexidades do classificador e dos comitês, pode-se definir a complexidade de classificação do PEP. No primeiro cenário, quando o método estará no período da escolha do melhor lambda a ser usado no processo de aprendizagem, o PEP simplesmente fará a classificação com o comitê definido com seus parâmetros padrões. Já no segundo cenário, após conhecido o melhor valor da diversidade lambda, o PEP fará a classificação com o comitê contendo o valor de lambda atualizado. Dessa forma, nota-se que em ambos os casos o PEP terá uma complexidade computacional constante. Assim, o custo computacional de classificação será representado apenas pela complexidade do comitê e do classificador, podendo ser definida como $\mathcal{O}(\Phi \times \Psi)$.

3.3 CONSIDERAÇÕES FINAIS

Neste capítulo foram detalhadas as estratégias referentes à aplicação do ajuste de parâmetros de forma dinâmica nos métodos de classificação escolhidos neste trabalho bem como as principais particularidades de suas implementações.

Primeiramente, foram explicadas as argumentações e implementações relacionadas com PL-kNN, a abordagem baseada em k-NN que ajusta de forma dinâmica e incrementalmente os vizinhos k usados pelos classificadores pareados k-NN. Nesta parte, também foram apresentadas as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, enfoques que utilizam estratégias de implementação diferentes para ajustar os valores k dos classificadores pareados dos métodos mencionados.

Seguidamente, foram descritas as implementações de PEP, uma técnica genérica que ajusta parâmetros dinamicamente, que foi aplicada e testada no parâmetro de diversidade (λ) de diversos comitês de classificadores. Para este propósito, o método de estimação descrito foi usado para criar BOLE-PE, OABM1-PE e OzaBag-PE, enfoques que também foram explicitamente apresentados e explicados com todos os seus argumentos de implementação, os quais foram baseados nos comitês existentes BOLE, OABM1 e OzaBag-PE, respectivamente.

4 ESTUDO EMPÍRICO

Este capítulo descreve todas as informações importantes sobre a preparação e as configurações dos experimentos realizados como parte desta tese de doutorado. Da mesma forma, são reveladas as particularidades fundamentais dos conjuntos de dados artificiais e reais usados nas experimentações. Por conseguinte, são descritas as considerações referentes à configuração de parametrização dos métodos introduzidos nesta pesquisa.

Os experimentos foram executados utilizando um computador com um processador Intel Core i9 9900K, com 32GB de memória DDR4 2400MHz RAM e um SSD, rodando o sistema operacional Ubuntu Desktop 18.10 LTS 64-bit. A configuração dos scripts para extrair e analisar os resultados dos experimentos foi feita com a ferramenta MOAManager [1] (MACIEL; SANTOS; BARROS, 2020). Esta aplicação da web de código aberto auxilia na criação, execução, extração e formatação dos resultados de experimentos em ambientes online usando a estrutura MOA (BIFET et al., 2010).

Além disto, como foi aludido na seção 2.1, k-NN e HT foram escolhidos para serem usados como classificadores base nos experimentos, pois são classificadores que são muito utilizados na área do fluxo de dados e suas implementações estão disponíveis gratuitamente no MOA framework.

A acurácia dos métodos testados foi medida usando a metodologia $Prequential$ (DAWID, 1984) com uma janela deslizante de tamanho 1000 como seu mecanismo de esquecimento (HIDALGO; MACIEL; BARROS, 2019). Desta forma, cada instância é usada para teste antes de ser usada para treinamento e, portanto, a acurácia é atualizada de forma incremental. Isso garante que todas as instâncias sejam usadas para teste e treinamento, e nenhum treinamento acontece antes do teste em qualquer instância.

Com o intuito de verificar a existência ou não de diferenças estatísticas entre os enfoques avaliados, foi aplicado o teste de *Friedman* sobre os métodos para identificar qual deles apresenta melhor desempenho em termos de acurácia na avaliação dos resultados em cenários onde acontecem mudanças de conceitos na distribuição dos dados. Os testes foram aplicados com um intervalo de confiança de 95%. Assim, o resultado dos ranks estabelecidos pelo método estatístico proporcionam o critério para avaliar os métodos de classificação introduzidos no trabalho, levando em consideração que quanto menor seja o valor do rank calculado, melhor

[1] MOAManager está disponível gratuitamente em <https://github.com/brunom4ciel/moamanager/>

vai ser o resultado nessa abordagem.

Além disso, como a rejeição da hipótese nula do método de *Friedman* indica uma ocorrência de diferenças estatísticas entre os métodos mas não define quais deles são estatisticamente diferentes, foi aplicado o pós-teste de *Nemenyi* (NEMENYI, 1963) para fazer comparações múltiplas entre os métodos testados e distinguir as diferenças significativas, conforme prescrito por Demsar (DEMSAR, 2006).

4.1 CONJUNTO DE DADOS

Esta seção descreve todos os conjuntos de dados artificiais e do mundo real escolhidos para os experimentos descritos na seção anterior.

Em sentido geral, para realizar os testes e medir o desempenho das novas propostas ((i) PL-kNN e as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$; (ii) PEP e as aplicações BOLE-PE, OABM1-PE e OzaBag-PE), foram selecionados seis geradores de conjuntos de dados artificiais e construídos conjuntos de dados abruptos e graduais com 10.000, 20.000, 50.000, 100.000, 500.000 e 1.000.000 instâncias configuradas com 4 mudanças de conceitos distribuídas em intervalos regulares.

Em todos os conjuntos de dados graduais, a duração das mudanças de conceito foi definida para 500 instâncias. Em particular, nos testes experimentais relacionados com PEP, os conjuntos de dados $Agrawal_1$ e Sea foram definidos com 5% de ruído em cada um dos seus atributos numéricos. Para calcular a acurácia final dos métodos testados, os experimentos foram executados 30 vezes e os resultados médios foram calculados em conjunto com os intervalos de confiança de 95%.

Além disso, foram selecionados alguns conjuntos de dados do mundo real com um número muito diferente de instâncias e complexidade, que foram usados em trabalhos anteriores na área. Nesses conjuntos de dados, o número e a posição das mudanças de conceitos são desconhecidos.

Os próximos tópicos descrevem as principais particularidades de cada uma das bases de dados selecionadas no trabalho. Igualmente, é apresentado um resumo destes conjuntos de dados localizados na Tabela 1.

4.1.1 Bases Artificiais

4.1.1.1 Agrawal

Agrawal (AGRAWAL; IMIELINSKI; SWAMI, 1993; SANTOS et al., 2014) é um gerador que combina um conjunto de atributos referentes a uma série informações pessoais de alguns indivíduos, com o intuito de realizar uma análise de crédito.

Cada indivíduo é representado por um conjunto de 9 atributos, sendo eles: salário, comissão, idade, escolaridade, possui carro, código postal, valor do imóvel, quantidade de anos que vive em casa própria, e o valor do empréstimo desejado. Desses, 3 são categóricos e 6 numéricos.

De acordo com a combinação dos atributos, os indivíduos podem ser classificados em dois grupos: A ou B. Uma vez que a classificação pode ser realizada com até dez funções diferentes, o rótulo de cada instância poderá variar de acordo com a função utilizada. Duas versões deste gerador foram configurados nesta tese de doutorado: $Agrawal_1$, com funções F1 a F5, e $Agrawal_2$, com funções F6 a F10, semelhantes às utilizadas em (BARROS; SANTOS, 2018; BARROS; SANTOS, 2019).

4.1.1.2 LED

LED (BIFET; HOLMES; PFAHRINGER, 2010; GONÇALVES JR.; BARROS, 2013) produz uma base com 24 atributos categóricos. Desses, 17 são irrelevantes. Ademais, o conjunto de dados possui 10 possíveis classes, também categóricas.

O problema representado por este gerador refere-se a tarefa de predição do dígito apresentando por um visor LED com 7 segmentos. Cada atributo do conjunto de dados gerado, por padrão, possui 10% de chance de ser invertido (nível padrão de ruído).

Além disso, as mudanças de conceitos são simuladas através de modificações nas posições dos atributos relevantes. No trabalho foi utilizada a versão de LED disponível no ambiente MOA.

4.1.1.3 Mixed

Mixed (GAMA et al., 2004b; BARROS; SANTOS; GONÇALVES JR., 2016) cria um conjunto de dados composto por 4 atributos, sendo 2 deles booleanos (v e w) e 2 numéricos (x e y).

De acordo com os valores de seus atributos, as instâncias são classificadas em positivas ou negativas.

Para que uma instância seja rotulada como positiva, pelo menos duas das seguintes condições devem ser satisfeitas: os valores de v e w devem ser verdadeiros e $y < 0,5+0,3sin(3\pi x)$.

Para realizar a simulação de mudanças de conceitos, este gerador efetua a inversão dos rótulos, onde, para que uma instância seja considerada positiva, pelo menos duas das seguintes condições devem ser satisfeitas: os valores de v e w deverão ser falsos e $y \geq 0,5+0,3\sin(3\pi x)$.

4.1.1.4 Random RBF

Radial Basis Function (RBF) (BIFET et al., 2009; SANTOS; BARROS; GONÇALVES JR., 2015) é um gerador onde os conjuntos de dados são criados com n centroides. Cada centroide tem seu centro, rótulo e peso, definido aleatoriamente. Além disso, cada instância criada por este gerador possui m atributos que variam de acordo com:

1. A posição de cada centro;
2. Um número gerado aleatoriamente, indo de -1 a $+1$, o qual será multiplicado por um valor baseado no desvio padrão de cada centro e na distribuição Gaussiana.

A simulação das mudanças de conceitos é realizada através da modificação do número e da posição dos centroides.

4.1.1.5 SEA

O gerador *SEA* (AGRAWAL; IMIELINSKI; SWAMI, 1993; STREET; KIM, 2001) é composto por três atributos dos quais apenas $f1$ e $f2$ são relevantes. O limite de decisão de suas duas classes é dado por $f1 + f2 = \theta$, onde θ é um limite pré-definido. Nos conjuntos de dados criados para os experimentos, 5% de ruído também foi incluído.

4.1.1.6 Sine

Em *Sine* (GAMA et al., 2004b; SANTOS et al., 2014), as bases de dados criadas possuem dois atributos relevantes (x e y). Cada um dos atributos possui valores uniformemente distribuídos no intervalo $[0, 1]$.

Este gerador cria conjuntos de dados com dois possíveis conceitos. São eles: Sine1 e Sine2. No primeiro contexto, todos os pontos abaixo da curva $y = \sin(x)$ são classificados como positivos.

Já no segundo contexto, a inequação $y < 0,5 + 0,3\sin(3\pi x)$ deve ser verdadeira para que uma determinada instância seja considerada como positiva. Nos dois contextos citados, as mudanças de conceitos são simuladas pela inversão das condições ou pela alternância entre os contextos Sine1 e Sine2.

4.1.1.7 WaveForm

WaveForm (BIFET; HOLMES; PFAHRINGER, 2010; MACIEL; SANTOS; BARROS, 2015) é composto por três classes e 40 atributos numéricos, onde os últimos 19 são utilizados para adicionar ruído. Neste gerador cada classe é gerada a partir de uma combinação de duas dentre 3 ondas base.

Para realizar alterações, as posições dos atributos, que representam um determinado contexto, são trocadas. Cada instância é gerada com ruído adicional (média 0, variância 1) em cada atributo.

Não obstante, é válido constatar que neste conjunto de dados pode ser derivada uma expressão analítica através da taxa do erro de Bayes. O uso desta regra em uma amostra de teste de tamanho 5000, dá uma taxa de erro igual a 14.

4.1.2 Bases Reais

4.1.2.1 Census

Census (OZA; RUSSELL, 2001; DUA; GRAFF, 2017) é um conjunto de dados que contém dados censitários ponderados com variáveis demográficas relacionadas ao emprego retiradas das Pesquisas Populacionais Atuais (Current Population Surveys) em 1994 e 1995 conduzidas pelo U.S. Census Bureau. Esse conjunto de dados binários tem 299.285 instâncias e 40 atributos.

4.1.2.2 Connect4

Connect4 (FRANK, 2010) contém todas as posições legais no jogo de *Connect-4* em que nenhum dos jogadores ganhou ainda, e em que a próxima jogada não é forçado. Este conjunto de dados está composto por 42 atributos e 67.557 instâncias. A classe de resultado é o valor teórico do jogo para o primeiro jogador: ganhar, perder ou empatar.

4.1.2.3 Covertype

Covertype (BIFET et al., 2009) contém dados do Sistemas de Informações de Recursos da Região 2 do US Forest Service (USFS). Os atributos desta base de dados possuem informações que correspondem a uma área florestal de 30 $\times$ 30 metros. Este conjunto de dados conta com 581.012 instâncias. Cada instância possui 54 atributos, entre numéricos e categóricos. A tarefa de classificação é predizer o tipo de cobertura florestal com base nas diversas variáveis cartográficas.

Nesta proposta também foi utilizada uma versão ordenada pelo atributo elevação nomeada **Covertype Sorted** (IENCO et al., 2013). Dessa forma, esta base induz mudanças de conceitos graduais nas distribuições das classes. Isso ocorre porque, dependendo da elevação, alguns tipos de vegetações desaparecem, enquanto outras começam a aparecer.

4.1.2.4 Electricity

Electricity (GAMA et al., 2004a) é outro conjunto de dados coletado a partir de um mercado australiano de eletricidade (Australian New South Wales Electricity Market). Essa base de dados possui uma série de preços ajustados de acordo com a demanda e a oferta.

Esses preços sofrem variações a cada 5 minutos e o objetivo do problema é predizer se este irá aumentar ou diminuir em relação à movimentação média das últimas 24 horas. Electricity compõe-se de 2 classes, 8 atributos e 45.312 instâncias.

4.1.2.5 Letter Recognition

Letter Recognition (FERN; BRODLEY, 2004) visa identificar as letras maiúsculas do alfabeto inglês em displays de pixels retangulares preto e branco. As imagens dos personagens

foram baseadas em 20 fontes diferentes e cada letra dentro dessas 20 fontes foi distorcida aleatoriamente. O conjunto de dados contém 20.000 instâncias e 16 atributos numéricos primitivos.

4.1.2.6 Nomao

Nomao (CANDILLIER; LEMAIRE, 2012) é um conjunto de dados binários com 34.654 instâncias e 118 atributos. Representa um buscador de lugares, agregando informações de diversas fontes na web para propor informações completas relacionadas a um lugar.

Para isso, o objetivo é primeiro detectar quais dados se referem ao mesmo local e, em seguida, construir o modelo preditivo que decide se dois registros devem ser mesclados ou não.

4.1.2.7 Pokerhand

Pokerhand (BIFET; HOLMES; PFAHRINGER, 2010; GONÇALVES JR.; BARROS, 2013) é um conjunto de dados representado pela problemática de identificar o valor das 5 cartas numa mão do jogo de Poker. Dessa forma, cada instância contém 5 atributos numéricos e 5 categóricos. Além disso, as classes são determinadas por 10 possíveis rótulos categóricos.

Para este trabalho foram utilizadas: (i) a versão original da $Pokerhand$, com 1.000.000 de instâncias e cartas não ordenadas assim como, (ii) a versão modificada disponível no Site MOA onde as cartas são classificadas por classificação e naipe e as duplicatas foram removidas, resultando em 829.201 instâncias.

4.1.2.8 Rialto

Rialto (LOSING; HAMMER; WERSING, 2016) representa outro conjunto de dados referente à codificação de dez edifícios coloridos ao lado da famosa ponte Rialto em Veneza em um histograma RGB normalizado de 27 dimensões. As gravações cobrem 20 dias consecutivos de maio a junho de 2016. Esta base contém 82.250 instâncias descritas por 27 atributos e 10 classes.

4.1.2.9 Sensor

Sensor (ZHU, 2010) possui dados coletados a partir de 54 sensores introduzidos pelo laboratório de pesquisa $Intel\ Berkeley$. Esses sensores medem informações referentes a temperatura, umidade, luz e tensão. Também, a identificação do sensor representa a classe e deve ser identificado com base nos dados coletados. Além disso, os dados foram gravados por um período de 2 meses e as leituras foram realizadas a cada 1–3 minutos. Finalmente, esta base conta com 2.219.803 instâncias, 5 atributos e 54 classes.

4.1.2.10 Spam

Spam (KATAKIS; TSOUMAKAS; VLAHAVAS, 2010) consiste de 9324 instâncias e 500 atributos (palavras derivadas após a seleção de características). Esta base de dados é baseada nas mensagens de e-mail da Coleção Spam Assassin (*Spam Assassin Collection*) e está disponível em <http://mlkd.csd.auth.gr/conceptdrift.html>. Além disso, os recursos dessas mensagens de spam mudam gradualmente com o tempo, ou seja, esse conjunto de dados contém mudanças de conceitos graduais.

4.1.2.11 Weather

Weather foi introduzida por (ELWELL; POLIKAR, 2011). No período de 1949-1999, oito características diferentes, como temperatura, pressão, velocidade do vento, etc. foram medidas na Base Aérea de Offutt, em Bellevue, Nebraska.

O objetivo é prever se vai chover em um determinado dia ou não. O conjunto de dados contém 18.159 instâncias com um desbalanceamento em direção a nenhuma chuva (69%) (LOSING; HAMMER; WERSING, 2016).

4.1.2.12 WineRed e WineWhite

Wine-Quality-Red e o Wine-Quality-White são dois conjuntos de dados baseados em dados de vinhos (tinto e branco), apresentados por (CORTEZ et al., 2009; COETZEE; JAARSVELD; VANHAECKE, 2014).

Devido a questões de privacidade e logística, eles contêm apenas as variáveis físico-químicas

(entradas) e sensoriais (saída); os dados sobre tipos de uvas, marcas de vinho, preços, etc. não estão incluídos.

Ambos os conjuntos de dados têm 4898 instâncias e 11 atributos mais a saída, que é baseada em dados sensoriais (mediana de pelo menos 3 avaliações feitas por especialistas em vinho), onde cada especialista classificou a qualidade do vinho entre 0 (muito ruim) e 10 (excelente). O objetivo é modelar a qualidade do vinho com base em testes físico-químicos.

Tabela 1 – Características dos conjuntos de dados artificiais e reais

Bases artificiais				Bases reais			
Bases	#Inst.	#Atrib.	#Classes	Bases	#Inst.	#Atrib.	#Classes
Agrawal	10K;20K;50K;100K;500K;1M	9	2	Census	299,285	40	2
Led	10K;20K;50K;100K;500K;1M	24	10	Connect4	67,557	42	3
Mixed	10K;20K;50K;100K;500K;1M	4	2	Covert. Sorted	581,012	54	7
RBF	10K;20K;50K;100K;500K;1M	10	2	Electricity	45,312	8	2
SEA	10K;20K;50K;100K;500K;1M	3	2	LetterRecog.	20,000	16	26
Sine	10K;20K;50K;100K;500K;1M	2	2	Nomao	34,465	118	2
Wave.	10K;20K;50K;100K;500K;1M	40	3	Pokerhand	829,201	10	10
				Pokerhand1M	1,025,009	10	10
				Rialto	82,250	27	10
				Sensor	2,219,803	5	54
				Spam	9,324	500	2
				Weather	18,159	8	2
				WineWhite	4,898	12	9

Fonte: O autor (2022)

4.2 PARAMETRIZAÇÃO DOS MÉTODOS

Anteriormente, na implementação de PL-kNN na seção 3.1, foi descrita a configuração dos parâmetros do método para ajustar o número de vizinhos k de forma dinâmica. Da mesma forma, foram configurados os parâmetros das versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$. Estes métodos também usaram: (i) o valor mínimo k para os membros pareados de pl ($k\_low = 5$), (ii) uma lista de valores k para os membros pareados de pl (k_list = [5, 10, 15, 20, 30, 40, 50]), e (iii) o intervalo em instâncias para as atualizações em k_1 e k_2 ($interv = 100$). Aqui é valido ressaltar que estes valores de parametrização foram definidos com base nos experimentos preliminares apresentados nessa seção.

O classificador tradicional k-NN, conforme implementado no MOA, foi configurado com diferentes valores de k e essas versões foram comparadas com os métodos propostos em diferentes situações. Os valores de k escolhidos para configurar k-NN foram 5, 10 (seu valor

padrão usual) e 50, e as versões parametrizadas foram nomeadas kNN-k5, kNN-k10 e kNN-k50, respectivamente.

Por outro lado, relacionado com as parametrizações da técnica genérica (PEP) para ajustar parâmetros dinamicamente, que é aplicada e testada no parâmetro de diversidade (λ) de diferentes comitês de classificação, o BOLE, OABM1 e OzaBag foram testados usando seus parâmetros padrão propostos por seus respectivos autores, exceto na utilização do detector de mudanças de conceito auxiliar.

BOLE-PE, OABM1-PE e OzaBag-PE usaram os mesmos valores nos parâmetros comuns e foram adicionados parâmetros específicos relacionados ao PEP. Os limites inferior e superior (λ^{min}=1,0 e λ^{max}=10,0), usados por PEP como o intervalo de valores possíveis de lambda nos comitês, o número de comitês (T=10) com valores diferentes de λ e o número de instâncias de amostra (si=100) para aplicar PEP foram todos simplesmente escolhidos usando o bom senso. Além disso, o valor padrão definido como λ^{def} foi 1.0 nos três métodos, mas apenas em OABM1 é um parâmetro formal: nos outros dois ensembles, foi escolhido permanentemente no código em variáveis internas.

Adicionalmente, inspirados em Leveraging Bagging (LevBag) (BIFET; HOLMES; PFAHRINGER, 2010), esses comitês de classificadores também foram testados com λ definido como 6.0, para observar seu comportamento com um valor maior de λ e entender melhor o impacto do ajuste dinâmico de λ em PEP. Essas versões foram nomeadas BOLE-λ6, OABM1-λ6, e OzaBag-λ6, respectivamente. Além disso, observe que OzaBag-λ6 é, de fato, o mesmo que LevBag com sua configuração padrão, exceto pela mudança de seu detector de mudanças. Nesse sentido, RDDM e HDDM_A foram os detectores de mudanças de conceitos utilizados como auxiliares nos testes experimentais destes métodos.

4.3 CONSIDERAÇÕES FINAIS

Como foi mostrado neste capítulo, foram apresentadas detalhadamente as configurações experimentais que representam vital importância para o desenvolvimento do trabalho. Especificamente, foram descritas as configurações do computador onde foram executados os testes e as particularidades da configuração dos scripts para extrair e analisar os resultados. Também foram mencionados os classificadores escolhidos, bem como a metodologia utilizada para medir a acurácia dos métodos testados e seguidamente foram descritos os os testes de análise estatístico com que foram feitas as comparações entre os métodos testados.

Por outro lado, as principais descrições das bases de dados artificiais e reais em conjunto com a configuração dos parâmetros utilizados nos métodos escolhidos para este trabalho, dão a conhecer como se caracterizou o estudo empírico da investigação para prosseguir com a análise dos resultados que serão apresentadas nos próximos capítulos.

5 ANÁLISE DOS RESULTADOS EXPERIMENTAIS DE PL-KNN E VERSÕES

Neste capítulo são exibidas as comparações e análise dos resultados dos experimentos usando PL-kNN, as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, bem como diferentes parametrizações da versão tradicional do k-NN. Adicionalmente, para permitir que os classificadores se adaptem às mudanças de conceito, os detectores de mudanças de conceito RDDM (BARROS et al., 2017) e FTDD (CABRAL; BARROS, 2018) foram usados em conjunto com os classificadores. Esta escolha foi baseada nos resultados reportados em (BARROS; SANTOS, 2018).

As Tabelas 2 e 3 mostram os resultados de acurácia dos diferentes métodos testados com o detector RDDM nos conjuntos de dados artificiais em cenários de mudanças de conceito abruptas e graduais, respectivamente. De igual forma nas Tabelas 4 e 5, também são apresentados estes valores resultantes das acurácias dos métodos testados, mas neste caso, utilizando o detector de mudanças de conceitos FTDD.

Adicionalmente, para entender melhor o desempenho dos métodos testados, estes resultados de acurácia também são mostrados graficamente. Nas Figuras 5 e 6 são apresentados os gráficos dos testes utilizando RDDM como detector auxiliar nos métodos em cenários de mudanças de conceitos abruptas e graduais, respectivamente. De igual forma, nas Figuras 7 e 8 são mostrados os gráficos dos testes relacionados com o detector interno FTDD usado nos métodos experimentados. Nestas tabelas e figuras, pode-se apreciar o bom desempenho obtido por PL-kNN em geral quando comparado com os outros métodos testados.

Especificamente, nos ambientes de mudanças abruptas com RDDM (Tabela 2), PL-kNN obteve os melhores resultados em 16 dos 35 conjuntos de dados, especialmente em $Agrawal_1$, LED, $Sine$ e $Waveform$, com resultados próximos dos melhores nos conjuntos de dados restantes. Note que na maioria dos casos os resultados de PL-kNN$_2$ e PL-kNN$_4$ são próximos entre eles e também quando comparados com PL-kNN. Já o PL-kNN$_3$ só obteve melhor acurácia igualmente do que PL-kNN e PL-kNN$_4$ com o gerador LED nos testes com 500K. Nos geradores $Mixed$ e RBF, kNN-k5 apresentou os melhores resultados, enquanto que nos conjuntos de dados *Agrawal*$_2$ kNN-k50 apresentou as melhores acurácias.

Nos testes com mudanças graduais exibidos na Tabela 3, PL-kNN não foi tão eficaz em alcançar as melhores acurácias, sendo o melhor método em 10 dos 35 conjuntos de dados artificiais, principalmente nas bases LED e $Sine$ e em *Agrawal*$_1$ nos testes com o maior tamanho do gerador. No entanto, vale ressaltar que o PL-kNN e os métodos baseados nele

(PL-kNN$_2$, PL-kNN$_3$, PL-kNN$_4$), foram igualmente eficientes em entregar resultados próximos dos melhores em todos os conjuntos de dados testados.

Tabela 2 – Média das acurácias em porcentagem usando RDDM como detector auxiliar nos métodos, com intervalos de confiança de 95% em cenários de mudanças de conceito abruptas com conjuntos de dados artificiais.

		Reactive Drift Detection Method (RDDM)						
		Mudanças de conceito abruptas						
#Inst.	Bases	kNN-k5	kNN-k10	kNN-k50	PL-kNN	PL-kNN$_2$	PL-kNN$_3$	PL-kNN$_4$
10K	Agrawal$_1$	62,49±0,26	62,61±0,26	**67,03±0,20**	66,95±0,23	66,91±0,20	66,90±0,20	66,87±0,20
	Agrawal$_2$	75,44±0,30	74,99±0,23	**81,13±0,31**	80,90±0,34	80,87±0,32	80,90±0,33	80,89±0,33
	LED	55,80±0,28	59,26±0,33	62,04±0,38	**63,29±0,39**	63,28±0,39	63,25±0,39	63,28±0,39
	Mixed	**95,27±0,16**	94,29±0,17	89,19±0,25	95,18±0,17	95,14±0,18	95,14±0,18	95,14±0,18
	RBF	**36,70±0,35**	35,72±0,31	29,49±0,34	36,59±0,33	36,05±0,53	36,16±0,54	36,10±0,53
	Sine	90,53±0,16	90,20±0,18	87,83±0,21	**90,78±0,19**	90,51±0,23	90,58±0,22	90,58±0,22
	Waveform	79,95±0,31	80,94±0,31	81,57±0,31	**82,53±0,30**	82,48±0,30	82,51±0,30	82,50±0,30
20K	Agrawal$_1$	63,92±0,17	63,02±0,14	68,37±0,12	**68,43±0,13**	68,40±0,14	68,39±0,12	68,34±0,14
	Agrawal$_2$	78,46±0,13	78,11±0,11	**83,72±0,13**	83,65±0,14	83,68±0,15	83,66±0,14	83,65±0,14
	LED	57,99±0,19	61,61±0,23	65,54±0,28	**66,12±0,28**	66,11±0,28	66,10±0,28	**66,12±0,28**
	Mixed	**96,39±0,11**	95,68±0,10	91,95±0,13	96,34±0,11	96,27±0,13	96,27±0,13	96,27±0,13
	RBF	**36,87±0,18**	36,33±0,20	30,17±0,29	36,77±0,17	36,32±0,35	36,40±0,32	36,36±0,32
	Sine	91,64±0,13	91,47±0,14	89,75±0,14	**91,96±0,13**	91,71±0,19	91,82±0,16	91,78±0,17
	Waveform	80,60±0,19	81,76±0,21	83,08±0,17	**83,50±0,17**	83,48±0,18	83,47±0,18	83,45±0,19
50K	Agrawal$_1$	64,87±0,11	63,41±0,10	69,35±0,09	**69,38±0,10**	69,34±0,10	69,32±0,09	69,32±0,10
	Agrawal$_2$	80,42±0,10	79,99±0,08	**85,29±0,07**	85,23±0,06	85,24±0,06	85,23±0,06	85,23±0,06
	LED	59,20±0,16	63,00±0,18	67,72±0,23	**67,94±0,23**	67,92±0,22	67,92±0,22	67,92±0,22
	Mixed	**97,05±0,05**	96,53±0,05	93,68±0,08	97,03±0,05	96,85±0,20	96,85±0,20	96,85±0,20
	RBF	**36,86±0,13**	36,65±0,12	30,48±0,18	36,82±0,13	36,31±0,29	36,53±0,24	36,39±0,28
	Sine	92,29±0,08	92,22±0,07	90,87±0,07	**92,69±0,07**	92,45±0,15	92,54±0,12	92,53±0,12
	Waveform	80,94±0,14	82,14±0,14	83,88±0,13	84,04±0,13	**84,07±0,12**	84,02±0,13	**84,07±0,12**
100K	Agrawal$_1$	65,21±0,08	63,62±0,08	69,68±0,06	**69,71±0,07**	69,66±0,08	69,65±0,07	69,65±0,07
	Agrawal$_2$	81,15±0,07	80,63±0,05	**85,93±0,05**	85,91±0,05	85,90±0,05	85,90±0,05	85,90±0,05
	LED	59,69±0,14	63,54±0,15	68,52±0,15	68,61±0,16	**68,62±0,16**	68,61±0,16	**68,62±0,16**
	Mixed	**97,25±0,03**	96,79±0,04	94,21±0,04	97,24±0,03	97,12±0,10	97,12±0,10	97,12±0,10
	RBF	**36,96±0,08**	36,81±0,09	30,64±0,12	36,93±0,08	36,44±0,22	36,54±0,18	36,43±0,21
	Sine	92,48±0,05	92,49±0,04	91,22±0,05	**92,89±0,05**	92,63±0,14	92,73±0,09	92,71±0,11
	Waveform	80,99±0,11	82,34±0,09	84,17±0,07	84,24±0,07	84,24±0,07	84,23±0,07	**84,25±0,07**
500K	Agrawal$_1$	65,47±0,07	63,78±0,07	69,93±0,04	**70,07±0,04**	69,95±0,04	69,95±0,06	69,95±0,04
	Agrawal$_2$	81,74±0,07	81,16±0,07	**86,35±0,06**	86,33±0,05	86,32±0,05	86,32±0,05	86,33±0,06
	LED	59,98±0,12	63,87±0,11	69,05±0,11	**69,07±0,14**	69,06±0,14	**69,07±0,13**	**69,07±0,13**
	Mixed	**97,47±0,03**	97,03±0,03	94,63±0,05	97,46±0,03	97,36±0,10	97,34±0,11	97,35±0,11
	RBF	**36,93±0,07**	36,83±0,09	30,71±0,12	36,91±0,06	36,48±0,20	36,52±0,25	36,45±0,24
	Sine	92,69±0,05	92,72±0,04	91,54±0,03	**93,12±0,05**	92,86±0,20	92,94±0,13	92,95±0,16
	Waveform	81,05±0,10	82,35±0,12	84,33±0,07	**84,35±0,06**	84,34±0,06	84,34±0,05	**84,35±0,05**

Fonte: O autor (2022)

Por outro lado, ao usar FTDD como detector auxiliar nas comparações dos métodos apresentados nas Tabelas 4 e 5 com os tipos de mudanças abruptas e graduais nessa ordem, PL-kNN e suas versões foram geralmente os melhores métodos. Nesse sentido, PL-kNN teve o

maior destaque nos conjuntos de dados abruptos quanto nos graduais, com 17 e 15 melhores resultados, respectivamente.

Tabela 3 – Média das acurácias em porcentagem usando RDDM como detector auxiliar nos métodos, com intervalos de confiança de 95% em cenários de mudanças de conceito graduais com conjuntos de dados artificiais.

		Reactive Drift Detection Method (RDDM)						
		Mudanças de conceito graduais						
#Inst.	Bases	kNN-k5	kNN-k10	kNN-k50	PL-kNN	PL-kNN$_2$	PL-kNN$_3$	PL-kNN$_4$
10K	Agrawal$_1$	61,69±0,23	61,54±0,23	**66,17±0,19**	66,05±0,21	66,10±0,21	66,06±0,21	66,07±0,21
	Agrawal$_2$	73,51±0,25	73,74±0,22	**79,32±0,24**	79,11±0,26	79,12±0,25	79,11±0,24	79,10±0,25
	LED	54,63±0,30	58,11±0,34	61,09±0,34	62,05±0,36	62,04±0,35	62,05±0,35	**62,08±0,36**
	Mixed	86,06±0,25	**86,45±0,17**	84,03±0,22	86,18±0,21	86,22±0,20	86,40±0,19	86,22±0,19
	RBF	**36,80±0,34**	35,85±0,31	29,47±0,34	36,59±0,35	35,97±0,52	36,00±0,52	35,99±0,52
	Sine	83,17±0,23	83,50±0,23	82,40±0,17	**83,91±0,23**	83,76±0,21	83,85±0,21	83,78±0,20
	Waveform	79,98±0,31	81,00±0,30	81,69±0,27	82,42±0,28	82,43±0,29	82,43±0,29	**82,44±0,28**
20K	Agrawal$_1$	63,60±0,17	62,67±0,14	**68,13±0,13**	68,07±0,13	67,99±0,13	68,01±0,13	68,00±0,13
	Agrawal$_2$	77,83±0,11	77,62±0,12	**82,98±0,12**	82,86±0,13	82,87±0,13	82,87±0,12	82,87±0,12
	LED	57,42±0,19	61,00±0,21	64,97±0,30	**65,50±0,27**	65,44±0,28	65,43±0,28	65,45±0,27
	Mixed	**92,12±0,15**	91,93±0,13	89,33±0,13	92,00±0,16	91,46±0,17	91,71±0,14	91,57±0,17
	RBF	**36,87±0,18**	36,28±0,19	30,11±0,27	36,73±0,16	36,11±0,41	36,32±0,36	36,26±0,38
	Sine	87,92±0,16	88,14±0,14	87,12±0,15	**88,56±0,14**	88,29±0,14	88,44±0,12	88,31±0,14
	Waveform	80,60±0,20	81,81±0,21	83,11±0,17	**83,49±0,17**	83,47±0,17	83,47±0,17	83,48±0,17
50K	Agrawal$_1$	64,79±0,10	63,33±0,10	**69,29±0,09**	69,27±0,10	69,19±0,09	69,20±0,09	69,21±0,09
	Agrawal$_2$	80,23±0,10	79,85±0,08	**85,05±0,07**	85,00±0,06	85,00±0,07	85,00±0,06	85,00±0,07
	LED	59,03±0,15	62,83±0,18	67,54±0,23	**67,75±0,23**	67,74±0,23	67,74±0,23	67,74±0,23
	Mixed	**95,39±0,06**	95,13±0,05	92,75±0,08	95,35±0,05	94,35±0,11	94,62±0,13	94,33±0,12
	RBF	**36,88±0,13**	36,64±0,13	30,47±0,19	36,83±0,14	36,37±0,26	36,53±0,22	36,47±0,23
	Sine	90,99±0,08	91,05±0,07	89,96±0,07	**91,42±0,08**	90,97±0,12	91,18±0,10	90,94±0,12
	Waveform	80,94±0,15	82,14±0,14	83,86±0,13	84,01±0,12	**84,06±0,13**	84,04±0,13	84,05±0,12
100K	Agrawal$_1$	65,18±0,08	63,59±0,07	**69,66±0,06**	69,65±0,06	69,63±0,06	69,60±0,06	69,60±0,06
	Agrawal$_2$	81,03±0,07	80,54±0,05	**85,78±0,05**	85,75±0,05	85,75±0,05	85,75±0,05	85,75±0,05
	LED	59,64±0,13	63,47±0,15	68,47±0,15	68,55±0,15	**68,56±0,15**	**68,56±0,15**	**68,56±0,15**
	Mixed	**96,47±0,03**	96,12±0,04	93,79±0,04	96,45±0,03	95,30±0,14	95,65±0,12	95,28±0,14
	RBF	**36,96±0,08**	36,81±0,09	30,64±0,12	36,93±0,08	36,19±0,31	36,43±0,24	36,35±0,23
	Sine	91,91±0,05	91,96±0,04	90,80±0,05	**92,34±0,05**	91,91±0,10	92,03±0,09	91,85±0,09
	Waveform	80,99±0,11	82,34±0,09	84,13±0,08	84,23±0,07	84,23±0,07	84,23±0,07	**84,24±0,07**
500K	Agrawal$_1$	65,46±0,07	63,77±0,06	69,92±0,05	**70,01±0,06**	69,94±0,04	69,95±0,05	69,94±0,04
	Agrawal$_2$	81,71±0,06	81,14±0,07	**86,32±0,05**	86,30±0,05	86,28±0,05	86,29±0,05	86,29±0,05
	LED	59,97±0,13	63,87±0,11	69,05±0,12	**69,06±0,14**	**69,06±0,14**	69,05±0,14	**69,06±0,15**
	Mixed	**97,30±0,02**	96,89±0,02	94,54±0,06	97,29±0,03	96,49±0,23	96,80±0,15	96,51±0,19
	RBF	**36,93±0,07**	36,83±0,08	30,71±0,12	36,91±0,07	36,39±0,27	36,66±0,19	36,57±0,22
	Sine	92,57±0,06	92,61±0,04	91,44±0,03	**93,00±0,05**	92,60±0,14	92,73±0,16	92,49±0,11
	Waveform	81,05±0,10	82,35±0,12	84,32±0,07	84,34±0,06	84,34±0,06	84,34±0,05	**84,35±0,05**

Fonte: O autor (2022)

No caso dos enfoques PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, os resultados deles de forma geral ficaram na maioria das bases testadas bem próximos ou iguais àqueles do PL-kNN. As exceções em específico onde cada um deles obteve os melhores resultados na maioria dos testes quando

comparados ao resto dos métodos foram nos geradores artificiais LED e $Waveform$. Também vale a pena acrescentar que PL-kNN, PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$ todos eles usando FTDD como detector auxiliar no testes, foram melhores do que kNN-k10 (a versão padrão com k =10) em todos os cenários experimentados. Observe que nestas tabelas e nas próximas, o melhor resultado de cada conjunto de dados é mostrado em **negrito**.

Tabela 4 – Média das acurácias em porcentagem usando FTDD como detector auxiliar nos métodos, com intervalos de confiança de 95% em cenários de mudanças de conceito abruptas com conjuntos de dados artificiais.

		Fisher Test Drift Detector (FTDD)						
		Mudanças de conceito abruptas						
#Inst.	Bases	kNN-k5	kNN-k10	kNN-k50	PL-kNN	PL-kNN$_2$	PL-kNN$_3$	PL-kNN$_4$
10K	Agrawal$_1$	61,91±0,25	61,71±0,28	**66,92±0,28**	66,66±0,29	66,67±0,28	66,66±0,28	66,65±0,28
	Agrawal$_2$	75,08±0,41	75,17±0,27	81,30±0,31	**81,57±0,24**	81,47±0,24	81,52±0,25	81,48±0,23
	LED	55,60±0,34	58,75±0,40	58,69±1,04	62,80±0,46	**62,93±0,46**	62,87±0,46	62,85±0,45
	Mixed	**95,25±0,17**	94,24±0,19	88,44±0,47	95,17±0,17	95,14±0,17	95,14±0,17	95,14±0,17
	RBF	**36,99±0,34**	36,26±0,31	29,78±0,36	36,90±0,34	36,46±0,57	36,48±0,57	36,45±0,57
	Sine	90,66±0,17	90,26±0,17	86,34±0,48	**90,89±0,18**	90,64±0,22	90,70±0,21	90,71±0,21
	Waveform	79,95±0,32	80,90±0,31	79,37±0,68	**82,55±0,29**	82,53±0,30	82,52±0,30	82,51±0,29
20K	Agrawal$_1$	63,69±0,18	62,71±0,17	**68,42±0,15**	68,29±0,19	68,28±0,20	68,29±0,19	68,28±0,18
	Agrawal$_2$	78,69±0,21	78,34±0,14	84,02±0,09	**84,14±0,11**	84,10±0,10	84,10±0,11	84,10±0,11
	LED	57,97±0,21	61,44±0,21	63,23±0,73	65,92±0,34	**65,95±0,35**	**65,95±0,35**	**65,95±0,35**
	Mixed	**96,43±0,12**	95,70±0,11	91,66±0,24	96,39±0,12	96,32±0,14	96,32±0,14	96,32±0,14
	RBF	**36,91±0,20**	36,55±0,19	30,28±0,30	36,85±0,20	36,24±0,49	36,43±0,44	36,34±0,45
	Sine	91,79±0,12	91,57±0,13	89,17±0,23	**92,11±0,13**	91,86±0,20	91,94±0,18	91,94±0,18
	Waveform	80,60±0,19	81,74±0,21	81,87±0,39	83,58±0,16	83,58±0,16	**83,59±0,15**	83,58±0,16
50K	Agrawal$_1$	64,84±0,09	63,41±0,11	69,40±0,09	**69,41±0,10**	69,38±0,09	69,38±0,09	69,37±0,10
	Agrawal$_2$	80,79±0,11	80,13±0,09	85,52±0,06	**85,55±0,06**	85,54±0,06	85,54±0,06	85,54±0,06
	LED	59,17±0,16	62,87±0,23	66,52±0,42	67,80±0,24	**67,82±0,25**	**67,82±0,25**	**67,82±0,25**
	Mixed	**97,09±0,06**	96,58±0,05	93,54±0,15	97,08±0,06	97,04±0,07	97,04±0,07	97,04±0,07
	RBF	**36,85±0,13**	36,66±0,12	30,51±0,19	36,82±0,13	36,19±0,35	36,30±0,31	36,17±0,34
	Sine	92,41±0,07	92,32±0,07	90,68±0,11	**92,80±0,07**	92,51±0,17	92,60±0,14	92,61±0,14
	Waveform	80,96±0,14	82,15±0,14	83,40±0,18	84,08±0,13	**84,12±0,12**	84,10±0,13	**84,12±0,12**
100K	Agrawal$_1$	65,20±0,08	63,62±0,07	69,71±0,06	**69,78±0,06**	69,70±0,06	69,70±0,06	69,69±0,06
	Agrawal$_2$	81,31±0,07	80,71±0,07	85,99±0,05	**86,01±0,05**	**86,01±0,05**	**86,01±0,05**	**86,01±0,05**
	LED	59,69±0,13	63,51±0,18	67,83±0,35	68,51±0,20	**68,53±0,19**	68,52±0,20	**68,53±0,19**
	Mixed	**97,30±0,03**	96,85±0,04	94,19±0,07	97,29±0,03	97,27±0,05	97,27±0,05	97,27±0,05
	RBF	**36,94±0,08**	36,79±0,08	30,63±0,13	36,92±0,08	36,13±0,37	36,19±0,34	36,07±0,38
	Sine	92,59±0,05	92,58±0,03	91,17±0,05	**93,00±0,05**	92,71±0,14	92,80±0,12	92,80±0,12
	Waveform	81,01±0,11	82,36±0,08	83,95±0,10	84,30±0,07	84,30±0,07	84,30±0,07	**84,31±0,07**
500K	Agrawal$_1$	65,50±0,07	63,79±0,06	69,96±0,04	**70,13±0,06**	69,96±0,05	69,97±0,05	69,98±0,04
	Agrawal$_2$	81,80±0,06	81,20±0,07	**86,39±0,06**	**86,39±0,06**	**86,39±0,06**	**86,39±0,06**	**86,39±0,06**
	LED	60,02±0,13	63,90±0,14	68,96±0,18	**69,16±0,14**	**69,16±0,14**	**69,16±0,14**	**69,16±0,14**
	Mixed	**97,50±0,02**	97,07±0,03	94,71±0,05	**97,50±0,02**	**97,50±0,02**	**97,50±0,02**	**97,50±0,02**
	RBF	**36,93±0,07**	36,84±0,08	30,73±0,13	36,92±0,07	35,98±0,36	36,04±0,32	35,87±0,52
	Sine	92,75±0,06	92,79±0,03	91,58±0,05	**93,18±0,05**	92,92±0,24	93,01±0,18	93,01±0,18
	Waveform	81,07±0,10	82,38±0,12	84,35±0,08	**84,41±0,06**	**84,41±0,06**	**84,41±0,06**	84,40±0,06

Fonte: O autor (2022)

Agora, analisando os resultados dos conjuntos de dados do mundo real apresentados nas Tabelas 6 e 7 usando RDDM e FTDD como detectores auxiliares nos métodos, respectivamente, PL-kNN foi o melhor método em 5 de 10 conjuntos de dados testados com RDDM, seguido de kNN-k5 que foi o segundo melhor método em 4 do total de bases reais.

Tabela 5 – Média das acurácias em porcentagem usando FTDD como detector auxiliar nos métodos, com intervalos de confiança de 95% em cenários de mudanças de conceito graduais com conjuntos de dados artificiais.

		Fisher Test Drift Detector (FTDD)						
		Mudanças de conceito graduais						
#Inst.	Bases	kNN-k5	kNN-k10	kNN-k50	PL-kNN	PL-kNN$_2$	PL-kNN$_3$	PL-kNN$_4$
10K	Agrawal$_1$	61,39±0,24	60,99±0,24	**66,27±0,20**	66,08±0,21	66,05±0,21	66,05±0,20	66,04±0,20
	Agrawal$_2$	72,72±0,17	73,59±0,17	78,28±0,28	**78,30±0,25**	78,27±0,24	78,24±0,24	78,25±0,24
	LED	54,38±0,30	57,64±0,39	58,24±0,80	61,68±0,46	61,71±0,47	61,72±0,46	**61,73±0,46**
	Mixed	84,97±0,31	85,02±0,25	82,99±0,45	**86,01±0,21**	85,93±0,22	**86,01±0,24**	**86,01±0,23**
	RBF	**37,03±0,34**	36,23±0,29	29,74±0,34	36,93±0,35	36,47±0,60	36,53±0,60	36,51±0,60
	Sine	83,23±0,24	83,49±0,25	81,26±0,43	83,79±0,26	83,66±0,26	**83,80±0,22**	83,69±0,20
	Waveform	79,98±0,31	80,96±0,31	80,25±0,42	82,55±0,27	**82,56±0,29**	82,55±0,28	**82,56±0,28**
20K	Agrawal$_1$	63,41±0,18	62,37±0,16	**68,08±0,14**	68,01±0,14	68,01±0,14	68,02±0,14	68,01±0,14
	Agrawal$_2$	77,31±0,11	77,43±0,12	**82,50±0,11**	82,49±0,11	82,48±0,11	82,47±0,11	82,48±0,11
	LED	57,23±0,21	60,78±0,21	63,13±0,69	65,16±0,38	**65,27±0,38**	65,26±0,36	65,23±0,38
	Mixed	91,12±0,17	90,94±0,16	88,40±0,23	**91,70±0,13**	91,55±0,13	91,64±0,12	91,60±0,12
	RBF	**36,94±0,18**	36,56±0,18	30,24±0,31	36,88±0,18	36,38±0,45	36,48±0,45	36,45±0,47
	Sine	87,84±0,13	87,96±0,11	86,63±0,24	**88,45±0,11**	88,18±0,12	88,31±0,13	88,16±0,15
	Waveform	80,62±0,20	81,79±0,21	82,40±0,24	83,56±0,16	83,56±0,17	83,57±0,16	**83,59±0,16**
50K	Agrawal$_1$	64,73±0,10	63,23±0,10	**69,25±0,08**	**69,25±0,09**	69,20±0,09	69,21±0,10	69,20±0,09
	Agrawal$_2$	80,17±0,11	79,81±0,08	**84,92±0,07**	84,90±0,06	84,89±0,06	84,89±0,06	84,89±0,06
	LED	58,90±0,15	62,64±0,23	66,55±0,40	67,52±0,26	**67,56±0,26**	67,54±0,26	67,53±0,26
	Mixed	94,85±0,10	94,71±0,08	92,31±0,13	**95,14±0,06**	94,91±0,10	94,97±0,09	94,90±0,11
	RBF	**36,87±0,13**	36,65±0,11	30,50±0,19	36,83±0,13	36,21±0,38	36,29±0,36	36,16±0,38
	Sine	90,89±0,08	90,90±0,08	89,69±0,10	**91,36±0,08**	91,04±0,14	91,11±0,11	90,95±0,11
	Waveform	80,97±0,14	82,16±0,14	83,60±0,15	84,07±0,13	84,10±0,12	84,07±0,13	**84,12±0,12**
100K	Agrawal$_1$	65,14±0,08	63,52±0,07	69,62±0,06	**69,70±0,06**	69,62±0,07	69,63±0,07	69,61±0,06
	Agrawal$_2$	80,98±0,07	80,50±0,05	**85,70±0,05**	**85,70±0,05**	85,69±0,05	85,69±0,05	85,69±0,05
	LED	59,57±0,13	63,39±0,18	67,86±0,33	68,41±0,19	**68,43±0,19**	68,41±0,19	**68,43±0,19**
	Mixed	96,20±0,04	95,83±0,05	93,46±0,07	**96,33±0,04**	96,04±0,11	96,13±0,09	96,10±0,09
	RBF	**36,94±0,08**	36,79±0,08	30,63±0,13	36,92±0,08	36,14±0,34	36,28±0,33	36,16±0,37
	Sine	91,83±0,05	91,87±0,04	90,69±0,05	**92,29±0,06**	91,92±0,13	92,09±0,10	92,02±0,11
	Waveform	81,01±0,11	82,37±0,08	84,05±0,08	84,28±0,08	84,29±0,07	84,28±0,07	**84,31±0,07**
500K	Agrawal$_1$	65,48±0,07	63,77±0,06	69,94±0,04	**70,11±0,06**	69,94±0,05	69,94±0,05	69,96±0,04
	Agrawal$_2$	81,74±0,06	81,15±0,06	**86,33±0,05**	86,32±0,05	86,32±0,05	86,32±0,05	86,32±0,05
	LED	59,99±0,13	63,87±0,14	69,02±0,16	69,13±0,14	**69,14±0,14**	69,13±0,14	69,13±0,14
	Mixed	97,27±0,02	96,86±0,03	94,56±0,05	**97,29±0,02**	96,95±0,23	97,11±0,11	97,02±0,15
	RBF	**36,93±0,07**	36,84±0,08	30,73±0,13	36,92±0,07	36,06±0,35	36,17±0,33	35,96±0,50
	Sine	92,58±0,06	92,63±0,03	91,47±0,04	**93,03±0,04**	92,59±0,20	92,73±0,17	92,57±0,17
	Waveform	81,07±0,10	82,38±0,12	84,36±0,07	**84,41±0,06**	**84,41±0,06**	**84,41±0,06**	**84,41±0,06**

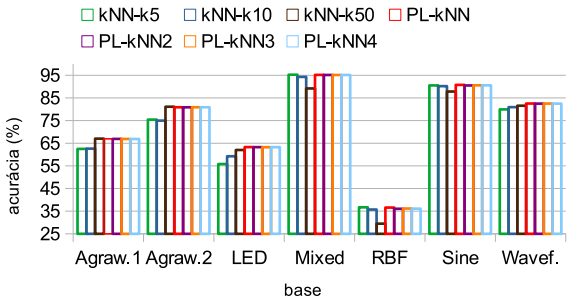
Fonte: O autor (2022)

Já nos testes com FTDD, o kNN-k5 foi o melhor, obtendo os maiores valores de acurácia

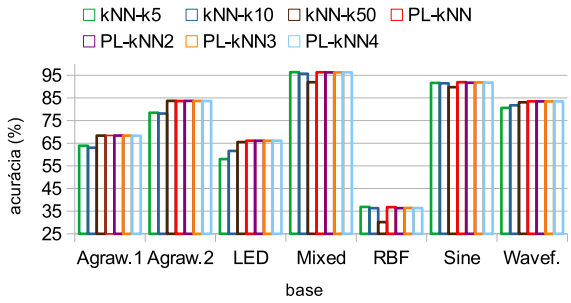
em 6 dos 10 conjuntos testados. Apesar disso, observe que os valores de acurácia de PL-kNN e suas versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$, assim como os valores de kNN-k5, mostram que o desempenho desses métodos foi muito próximo e, além disso, também é evidente que eles foram melhores que kNN-k10 e kNN-k50 na maioria dos cenários. Além disso, esses resultados da média das acurácias com ambos detectores (RDDM e FTDD) como auxiliares nos métodos, também foram mostrados nos gráficos da Figura 9.

Figura 5 – Ilustração da média das acurácias usando RDDM como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 500.000 instâncias.

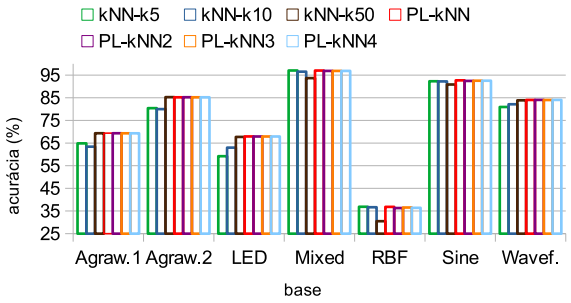
(a) 10.000 instâncias



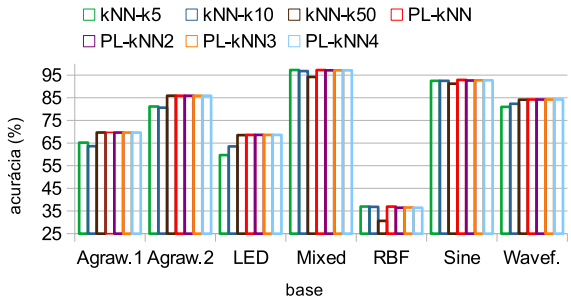
(b) 20.000 instâncias



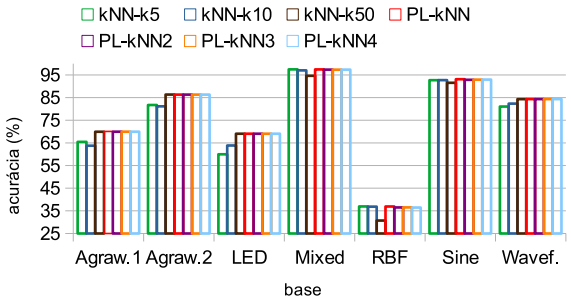
(c) 50.000 instâncias



(d) 100.000 instâncias



(e) 500.000 instâncias



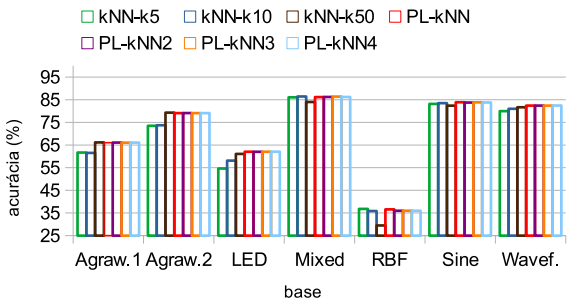
Fonte: O autor (2022)

De outra perspectiva, a Tabela 8 mostra os ranks dos métodos referentes ao teste de Friedman em todos os tipos de mudanças de conceito com os detectores RDDM e FTDD.

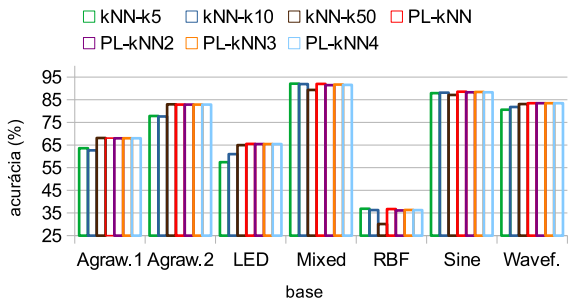
Esses resultados mostram que PL-kNN foi o método mais bem ranqueado em todos os cenários com ambos os detectores, geralmente seguido por PL-kNN$_3$, PL-kNN$_4$, PL-kNN$_2$, kNN-k5, kNN-k50 e kNN-k10. No entanto, nos conjuntos de dados graduais, kNN-k50 foi melhor que kNN-k5. Observe que nos testes com os conjuntos de dados do mundo real usando ambos os detectores, a ordem dos métodos ranqueados depois do melhor (PL-kNN) foi um tanto diferente, o kNN-k5 foi o segundo melhor classificado seguido de PL-kNN$_2$, PL-kNN$_4$, PL-kNN$_3$, kNN-k10 e por ultimo o kNN-k50.

Figura 6 – Ilustração da média das acurácias usando RDDM como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 500.000 instâncias.

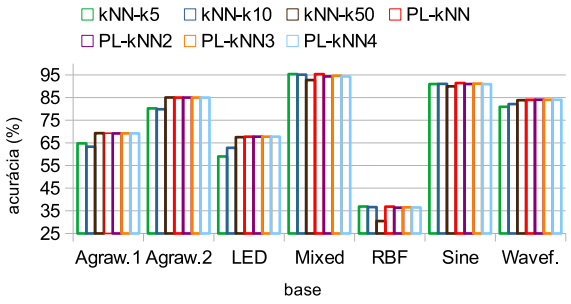
(a) 10.000 instâncias



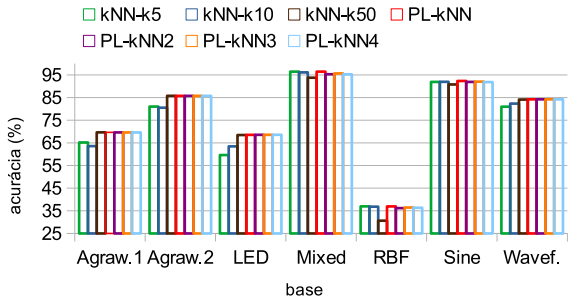
(b) 20.000 instâncias



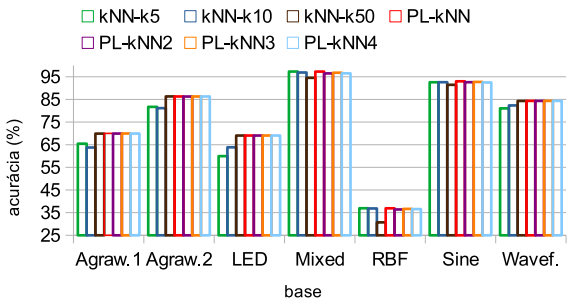
(c) 50.000 instâncias



(d) 100.000 instâncias



(e) 500.000 instâncias



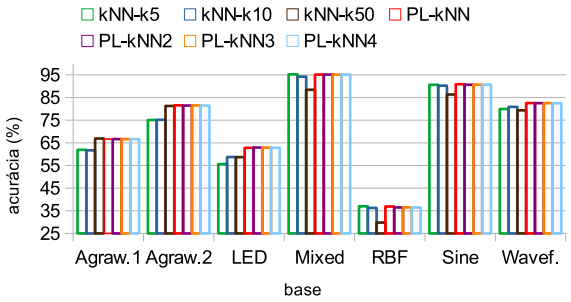
Fonte: O autor (2022)

Além dos resultados mostrados até agora, as Figuras 10 e 11 ilustram os gráficos rela-

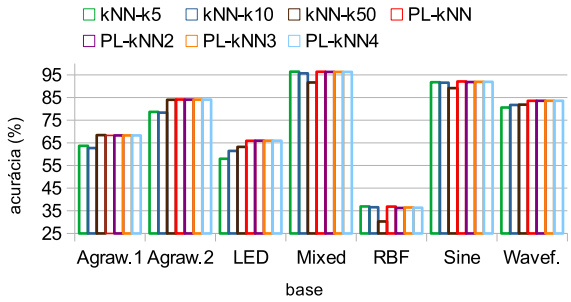
cionados ao teste de Friedman e o pós-teste de Nemenyi nos conjuntos de dados abruptos, graduais e reais testados usando RDDM e FTDD como detectores auxiliares nas comparações entre os métodos relacionados com o pós-teste de Nemenyi que define quais métodos são estatisticamente diferentes. Da mesma forma, a Figura 12 exibe os gráficos com todos os conjuntos de dados experimentados utilizando ambos os detectores auxiliares RDDM (Figura 12a) e FTDD (Figura 12b) nos métodos comparados. Note que os resultados dos ranks usam uma representação gráfica em que a diferença crítica (*CD*) é ilustrada por uma barra e os métodos conectados por essa barra são estatisticamente equivalentes.

Figura 7 – Ilustração da média das acurácias usando FTDD como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 500.000 instâncias.

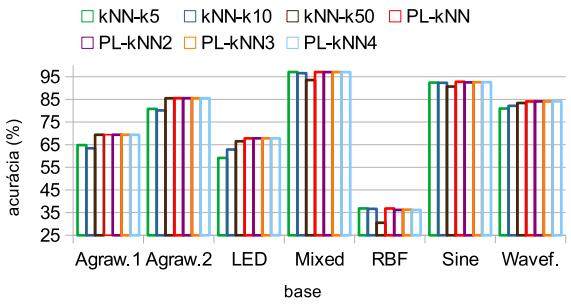
(a) 10.000 instâncias



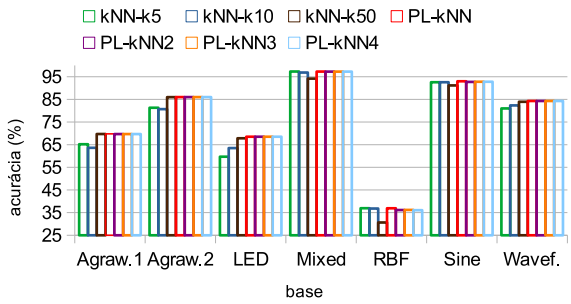
(b) 20.000 instâncias



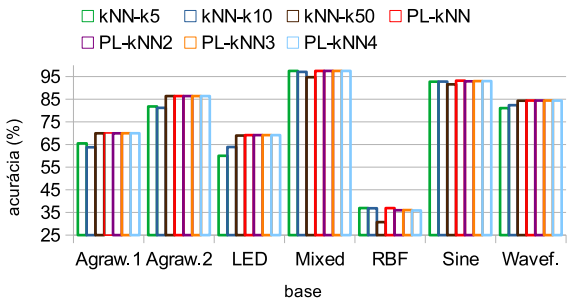
(c) 50.000 instâncias



(d) 100.000 instâncias



(e) 500.000 instâncias



Fonte: O autor (2022)

Nas ilustrações da Figura 10 do teste com RDDM como detector auxiliar nas comparações nos conjuntos de dados artificiais e reais, PL-kNN foi o melhor método de forma geral. Especificamente, nos testes das bases abruptas (Figura 10a), PL-kNN ficou como primeiro colocado e apresentou diferenças significativas em comparação com as suas versões e os métodos kNN-k5, kNN-50 e kNN-k10 que foram os últimos posicionados nessa ordem. Nas comparações das bases graduais (Figura 10b) não houve diferenças estatísticas entre PL-kNN, PL-kNN$_3$ e PL-kNN$_4$, os enfoques de melhor desempenho. PL-kNN$_3$ e PL-kNN$_4$ também tiveram um comportamento semelhante em relação a PL-kNN$_2$ devido a que não apresentaram diferenças significativas entres eles bem como entre PL-kNN$_2$, kNN-k50, kNN-k5 e kNN-k10, respectivamente.

Já nos testes com as bases reais (Figura 10c), PL-kNN foi novamente o melhor método seguido de kNN-k5, PL-kNN$_2$, PL-kNN$_4$, PL-kNN$_3$, kNN-k10 e kNN-k50 na ordem. Note que PL-kNN, kNN-k5 e PL-kNN$_2$ apresentaram diferenças significativas em relação a kNN-k10 e kNN-k50 que ficaram nas ultimas posições do gráfico.

Por outro lado, na comparação estatística utilizando FTDD como detector auxiliar nos conjuntos de dados artificiais e reais da Figura 11, os resultados também foram semelhantes, outra vez PL-kNN destacou-se entre os outros métodos comparados sendo o melhor tanto nos conjuntos de dados abruptos (Figura 11a) e graduais (Figura 11b) bem como nos conjuntos de dados do mundo real (Figura 11c).

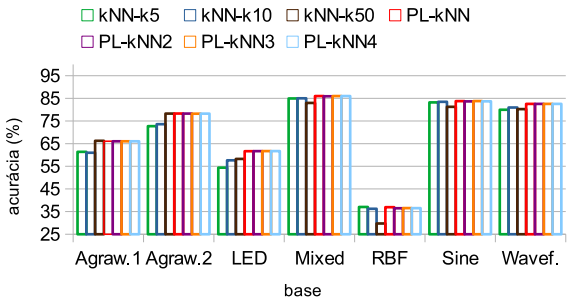
Tabela 6 – Média das acurácias em porcentagem usando RDDM como detector auxiliar nos métodos, com conjuntos de dados reais.

	Reactive Drift Detection Method (RDDM)						
Bases	kNN-k5	kNN-k10	kNN-k50	PL-kNN	PL-kNN$_2$	PL-kNN$_3$	PL-kNN$_4$
Census	94,63	94,49	93,54	**94,68**	94,66	94,65	94,66
Connect4	**77,64**	75,98	73,01	77,39	76,70	76,83	76,65
Cover.Sorted	77,11	75,65	72,75	**77,13**	76,68	76,36	76,66
Electricity	**82,60**	78,54	70,90	82,38	82,43	82,26	82,26
LetterRecog.	**66,26**	61,13	46,54	66,25	66,25	66,25	66,25
Nomao	96,92	96,73	95,54	**96,98**	96,93	96,93	96,93
Pokerhand1M	48,83	49,53	**49,89**	49,86	49,85	49,81	49,83
Rialto	**77,44**	71,03	49,19	77,34	76,82	76,82	76,82
Spam	92,24	90,71	84,68	**92,25**	91,93	91,86	91,88
WineWhite	50,30	50,06	50,82	**51,51**	51,37	51,37	51,37

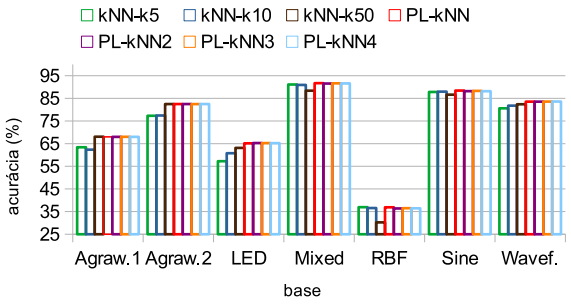
Fonte: O autor (2022)

Figura 8 – Ilustração da média das acurácias usando FTDD como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 500.000 instâncias.

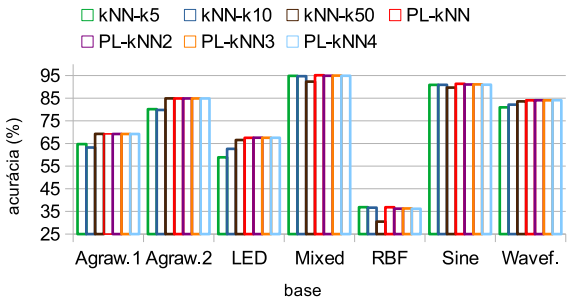
(a) 10.000 instâncias



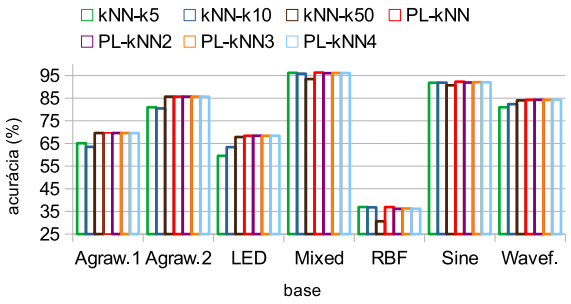
(b) 20.000 instâncias



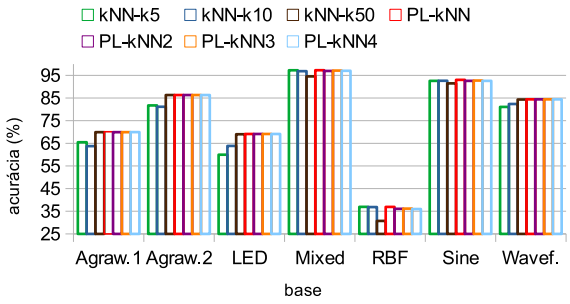
(c) 50.000 instâncias



(d) 100.000 instâncias



(e) 500.000 instâncias



Fonte: O autor (2022)

Tabela 7 – Média das acurácias em porcentagem usando FTDD como detector auxiliar nos métodos, com conjuntos de dados reais.

	Fisher Test Drift Detector (FTDD)						
Bases	kNN-k5	kNN-k10	kNN-k50	PL-kNN	PL-kNN$_2$	PL-kNN$_3$	PL-kNN$_4$
Census	94,60	94,53	93,65	**94,68**	94,62	94,63	94,64
Connect4	**77,75**	76,16	73,04	77,67	76,76	76,74	76,74
Cover.Sorted	**77,28**	75,83	72,81	**77,28**	76,40	76,57	76,29
Electricity	**82,02**	78,27	68,51	82,01	82,01	82,01	82,01
LetterRecog.	**66,26**	61,13	46,54	66,25	66,25	66,25	66,25
Nomao	**96,96**	96,74	95,04	96,86	96,70	96,70	96,70
Pokerhand1M	48,83	49,53	**49,90**	49,89	49,89	49,88	49,89
Rialto	59,99	39,92	29,23	**62,76**	62,44	62,44	62,44
Spam	**92,88**	90,94	82,49	92,54	91,92	91,92	91,92
WineWhite	50,30	50,00	51,52	51,59	**51,80**	**51,80**	**51,80**

Fonte: O autor (2022)

Nos gráficos da Figura 12 relacionados aos testes com todos os conjuntos de dados utilizando RDDM como auxiliar (Figura 12a), PL-kNN apresentou diferenças significativas em relação às demais abordagens, sendo o método de melhor desempenho. Não houve diferenças significativas entre PL-kNN$_3$, PL-kNN$_4$, e PL-kNN$_2$, bem como entre PL-kNN$_4$, PL-kNN$_2$ e kNN-k5. De igual forma aconteceu entre os métodos kNN-k5, kNN-k50 e kNN-k10, todos eles tiveram um performance semelhante, não apresentando diferenças significativas entre eles.

Tabela 8 – Ranks dos métodos em todos os tipos de conjuntos de dados com os detectores RDDM e FTDD.

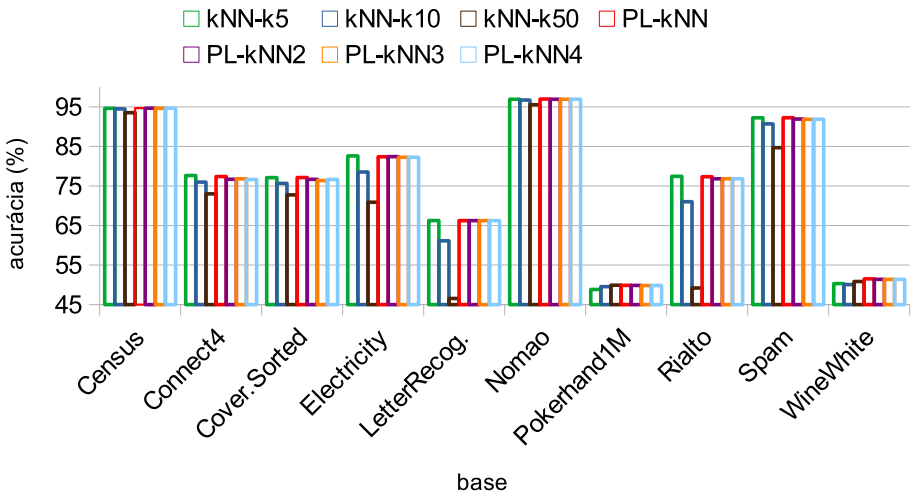
	RDDM				FTDD			
Métodos	Abrupto	Gradual	Real	Todos	Abrupto	Gradual	Real	Todos
kNN-k5	4,7714	4,9000	3,1000	4,6188	4,7714	5,3143	2,9500	4,7813
kNN-k10	5,9143	5,1429	6,1000	5,6000	5,8286	5,6286	5,8000	5,7375
kNN-k50	4,9714	4,8143	6,2000	5,0563	5,4714	4,9571	6,2000	5,3375
PL-kNN	**1,8429**	**2,3714**	**1,7500**	**2,0625**	**2,1429**	**2,1857**	**2,3500**	**2,1875**
PL-kNN$_2$	3,6000	3,9571	3,1000	3,6938	3,4000	3,5000	3,5000	3,4563
PL-kNN$_3$	3,4429	3,1857	4,0000	3,4000	2,9857	2,9571	3,6500	3,0563
PL-kNN$_4$	3,4571	3,6286	3,7500	3,5688	3,4000	3,4571	3,5500	3,4438

Fonte: O autor (2022)

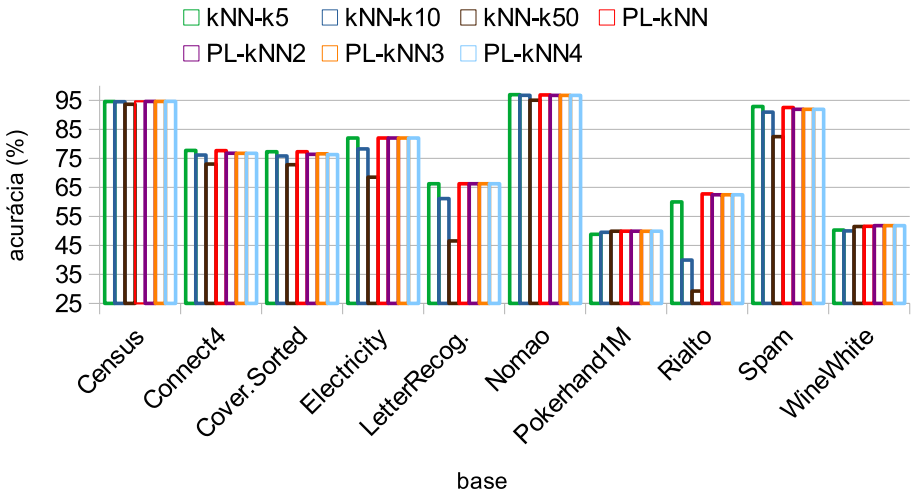
Finalmente, nos testes com FTDD (Figura 12b), PL-kNN, PL-kNN$_3$, PL-kNN$_4$ e PL-kNN$_2$ obtiveram os melhores desempenhos nessa ordem, sendo estatisticamente superiores aos demais métodos. Ao contrario do resultado com RDDM, pode se observar que PL-kNN

Figura 9 – Ilustração da média das acurácias usando RDDM e FTDD como detectores auxiliares nos métodos, com conjuntos de dados do mundo real.

(a) RDDM



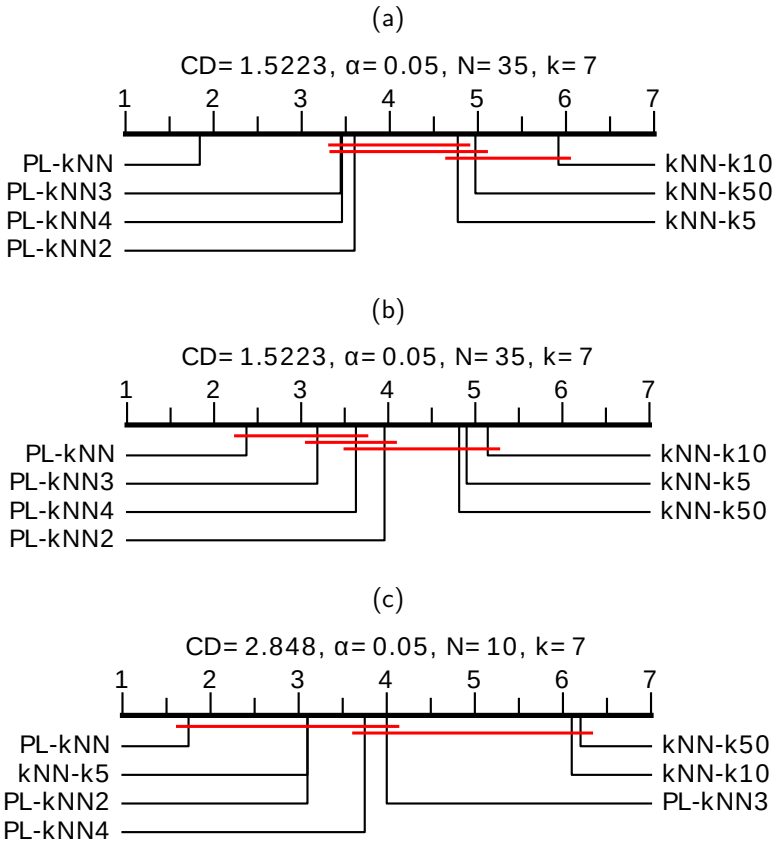
(b) FTDD



Fonte: O autor (2022)

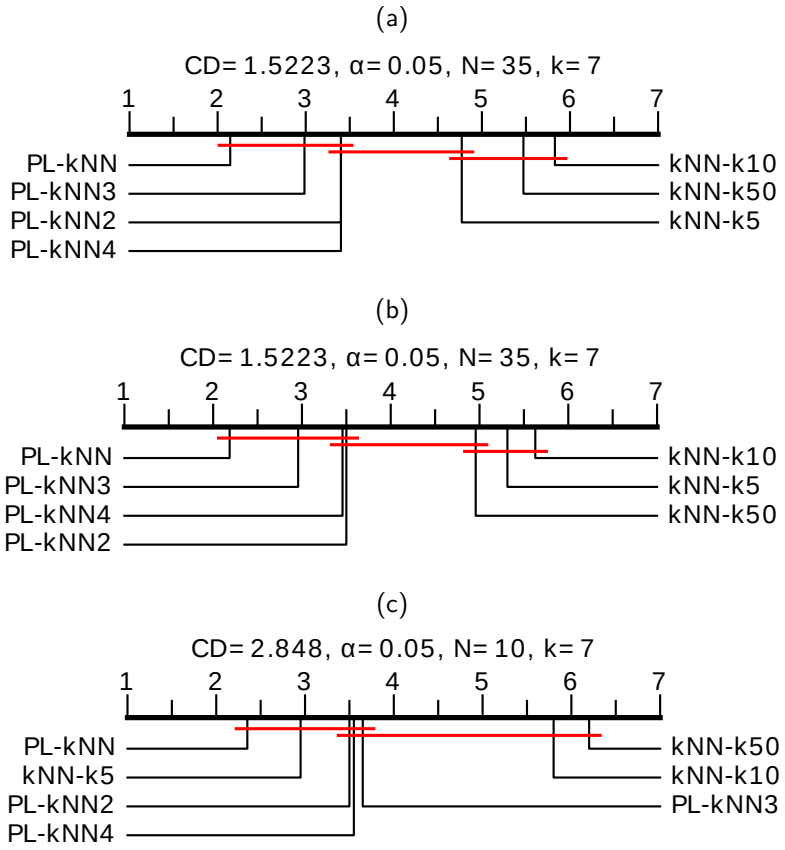
não teve diferenças significativas em relação a PL-kNN$_3$ e , por conseguinte, este último comportou-se de igual forma quando comparado com PL-kNN$_4$ e PL-kNN$_2$. Também pode se observar que entre os demais métodos (kNN-k5, kNN-k50, kNN-k10) não houve diferenças estatísticas.

Figura 10 – Comparação estatística da acurácia dos métodos usando o teste de Friedman e o pós-teste de Nemenyi com RDDM como detector auxiliar nos conjuntos de dados: (a) Abruptos, (b) Graduais e (c) Reais



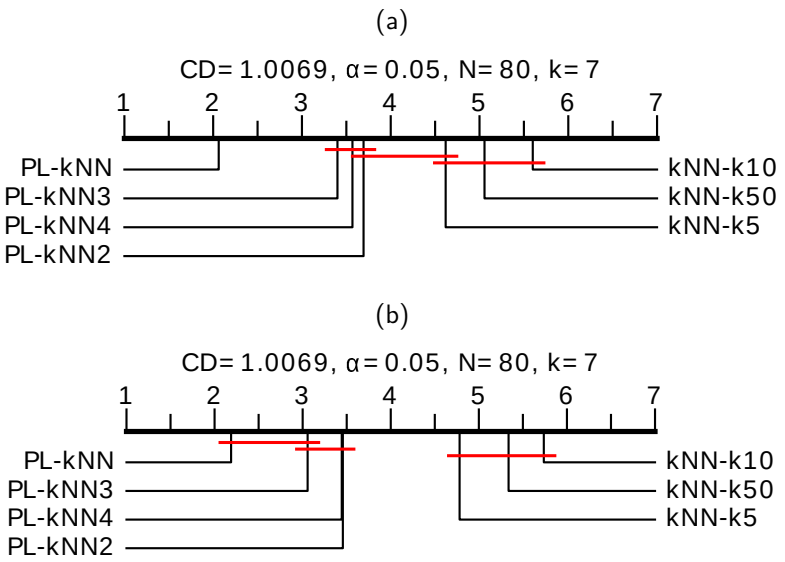
Fonte: O autor (2022)

Figura 11 – Comparação estatística da acurácia dos métodos usando o teste de Friedman e o pós-teste de Nemenyi com FTDD como detector auxiliar nos conjuntos de dados: (a) Abruptos, (b) Graduais e (c) Reais



Fonte: O autor (2022)

Figura 12 – Comparação estatística da acurácia dos métodos usando o teste de Friedman e o pós-teste de Nemenyi para comparações em todos os conjuntos de dados testados com os detectores auxiliares: (a) RDDM e (b) FTDD



Fonte: O autor (2022)

5.1 CONSIDERAÇÕES FINAIS

Neste capítulo foram detalhadas as avaliações dos resultados experimentais de PL-kNN e as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$. Nos testes realizados, foram usadas 35 bases artificiais e 10 reais com os detectores internos RDDM e FTDD nos métodos.

Estes resultados demostraram que as estratégias de implementação usadas nas abordagens apresentadas para otimizar de forma dinâmica o parâmetro k presente em cada um dos classificadores k-NN nos membros dos métodos foram benéficas na maioria dos cenários testados. O método que apresentou melhor performance em relação ao resto das abordagens comparadas foi o PL-kNN.

Especificamente, nos experimentos das comparações entre as médias das acurácias com as bases artificiais utilizando RDDM, o PL-kNN obteve os melhores resultados. As versões de PL-kNN foram igualmente eficientes, entregando resultados próximos dos melhores enfoques. Já nas mesmas comparações usando FTDD como detector interno nos métodos, PL-kNN e as versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$ foram os melhores métodos.

Visando valores resultantes dos experimentos com as bases reais, as novas abordagens obtiveram resultados satisfatórios nos testes com RDDM, mas com FTDD, o kNN-k5 foi o melhor método. No entanto, as novas propostas mostraram um desempenho muito próximo e também foi evidente a melhora destes em comparação com kNN-k10 e kNN-k50.

Por outro lado, nos resultados do teste estatístico de Friedman, o PL-kNN foi o método melhor ranqueado em todos os cenários, e por conseguinte, nas gráficas relacionadas com este teste e o pós-teste de Nemenyi usando RDDM como auxiliar nas comparações dos enfoques, o PL-kNN ficou como o primeiro colocado apresentando diferenças significativas em relação às outras abordagens comparadas. Por último, nas ilustrações utilizando FTDD, o PL-kNN e as versões PL-kNN$_3$, PL-kNN$_4$ e PL-kNN$_2$ apresentaram os melhores resultados nessa ordem e, também foram significativamente diferentes em relação a kNN-k5, kNN-k50 e kNN-k10.

Finalmente, após apresentados os resultados obtidos de todos os experimentos com as comparações entre as novas contribuições (PL-kNN e versões) e os restantes métodos (kNN-k5, kNN-k10 e kNN-k50), e considerando o uso de cada um deles em futuras avaliações experimentais da área, de forma geral o método sugerido para usar em qualquer cenário de mudanças de conceito usando bases artificiais e reais é o PL-kNN. Além disso, as versões PL-kNN$_3$ e PL-kNN$_4$ também podem ser utilizadas nas avaliações experimentais com conjuntos de dados artificiais em cenários de mudanças graduais.

6 ANÁLISE DOS RESULTADOS EXPERIMENTAIS DA APLICAÇÃO DO MÉTODO PEP EM COMITÊS DE CLASSIFICADORES

Este capítulo resume os resultados dos experimentos que avaliaram os comitês de classificadores BOLE-PE, OABM1-PE e OzaBag-PE, as versões obtidas após a aplicação da estratégia PEP para ajustar dinamicamente a diversidade λ, contra seus respectivos métodos originais, BOLE, OABM1 e OzaBag, bem como contra as versões configuradas com λ definido como 6, ou seja, BOLE-λ6 , OABM1-λ6 e OzaBag-λ6.

Para observar melhor o desempenho dos algoritmos de comitês de classificação modificados em diferentes situações, os nove métodos foram testados usando RDDM e HDDM_A, dois dos melhores detectores de mudanças de conceito disponíveis para configurar comitês (BARROS; SANTOS, 2019).

Os resultados dos experimentos foram separados por métodos básicos: primeiro BOLE e versões (BOLE-λ6 e BOLE-PE), depois OABM1 e versões (OABM1-λ6 e OABM1-PE), e, finalmente, OzaBag e versões (OzaBag-λ6 e OzaBag-PE).

As Tabelas 9, 10 e 11 apresentam os resultados dos métodos nos conjuntos de dados artificiais e do mundo real usando o algoritmo de classificação HT, configurado com RDDM como detector auxiliar de mudanças de conceito. No entanto, os resultados utilizando HDDM_A como detector interno nos métodos são mostrados nas Tabelas 12, 13 e 14.

Além disso, observe que essas tabelas também incluem as diferenças nos desempenhos dos métodos modificados em relação aos respectivos métodos originais, bem como os valores dos ranks resultantes do teste de Friedman considerando todos os conjuntos de dados testados. A melhor média de acurácia em cada conjunto de dados é mostrada em **negrito**. De outro modo, os resultados das acurácias dos métodos testados com ambos os detectores RDDM e HDDM_A utilizados como auxiliares nos métodos também são mostrados de forma gráfica (Apêndice A).

Nos resultados da Tabela 9, em relação aos conjuntos de dados artificiais, BOLE-PE obteve a melhor perfomance, alcançando os melhores valores de acurácia em 20 dos 40 cenários (50.00%), especificamente na maioria das versões de SEA e em algumas versões de $Agrawal_1$ e RBF, com ambos os tipos de mudanças de conceito. BOLE-λ6 ficou em segundo lugar nesses conjuntos de dados, fornecendo a melhor acurácia em 16 dos 40 conjuntos de dados (40.00%), e foi especialmente eficaz na maioria das versões dos conjuntos de dados $Waveform$.

Diretamente comparando as diferenças entre BOLE-PE e BOLE (Diff1), BOLE-PE foi

superior na maioria dos conjuntos de dados. As exceções foram os conjuntos de dados abruptos $Agrawal_1$ de 10K e 50K, bem como os conjuntos de dados $Agrawal_1$ e SEA de 50K e em $Agrawal_1$ de 100K com mudanças graduais, mas nesses casos as diferenças foram pequenas: 0.3% ou menores.

No caso de BOLE-PE versus BOLE-λ6 (Diff6), embora o número de melhores resultados seja semelhante (22 e 17), as diferenças são normalmente maiores quando BOLE-PE supera BOLE-λ6.

Por outro lado, em relação aos resultados nos conjuntos de dados do mundo real, observe que BOLE-PE obteve os melhores resultados de acurácia em 5 dos 9 conjuntos de dados testados: foi a melhor versão em todos os conjuntos de dados binários, enquanto os melhores resultados em conjuntos de dados multi-classe foram divididos pelas três versões.

Os resultados apresentados na Tabela 10 referem-se a OABM1, OABM1-λ6 e OABM1-PE e foram bem diferentes. Nos conjuntos de dados artificiais, OABM1 geralmente obteve as melhores acurácias nos conjuntos de dados $Agrawal_1$ e SEA, enquanto OABM1-λ6 foi o melhor na maioria dos conjuntos de dados RBF e $Waveform$. OABM1-PE foi o segundo melhor método na maioria dos conjuntos de dados, sendo o melhor nos geradores RBF e SEA de 100K e em $Waveform$ de 1M nas mudanças abruptas. Nos geradores com mudanças graduais, OABM1-PE também obteve os melhores valores resultantes em RBF e $Waveform$ de 50K e em $Waveform$ de 1M. Por outro lado, nos conjuntos de dados do mundo real, OABM1-PE foi superior aos outros métodos em 7 dos 9 conjuntos de dados experimentados (77.77%).

Considerando os valores de acurácia obtidos por OzaBag, OzaBag-λ6, e OzaBag-PE, apresentados na Tabela 11, os resultados nos conjuntos de dados artificiais mostram que OzaBag geralmente teve melhor desempenho nos conjuntos de dados $Agrawal_1$ e SEA, OzaBag-λ6 foi muito eficaz em conjuntos de dados RBF e $Waveform$, mas apresentou resultados muito ruins em todas as versões do conjunto de dados $Agrawal_1$, enquanto OzaBag-PE mostrou um desempenho mais proporcional em todos os conjuntos de dados testados.

Nesse sentido, observe que OzaBag-PE foi o melhor método em alguns casos nos geradores $Agrawal_1$, RBF e $Waveform$ com os tipos de dados abruptos, já nas bases graduais foi o melhor em $Agrawal_1$ de 10K e seguidamente em RBF e $Waveform$ de 20K. Além disso, é valido ressaltar que, na maioria dos resultados das acurácias nos testes com estes geradores artificiais, o OzaBag-PE foi o segundo melhor método em relação ao enfoque que obteve a melhor acurácia, sendo o mais estável em relação a resto das abordagens de forma geral nos

Tabela 9 – Média das acurácias de BOLE, BOLE-λ6 e BOLE-PE em porcentagem usando HT e RDDM, como detector auxiliar, com intervalos de confiança de 95%, em conjuntos de dados artificiais e reais.

Tipo-Inst.	Bases	BOLE	BOLE-λ6	BOLE-PE	Diff1	Diff6
Abrupt-10K	$Agrawal_1$	**70,60±0,43**	70,03±0,49	70,47±0,72	-0,13	0,44
	RBF	31,33±0,51	**31,62±0,44**	31,48±0,42	0,15	-0,14
	SEA	88,78±0,00	88,95±0,00	**89,41±0,00**	0,63	0,46
	Waveform	79,84±0,38	**80,74±0,32**	80,50±0,39	0,66	-0,24
Abrupt-20K	$Agrawal_1$	72,12±0,37	72,69±0,32	**72,89±0,52**	0,77	0,20
	RBF	31,60±0,44	31,97±0,40	**32,07±0,37**	0,47	0,10
	SEA	90,08±0,00	90,08±0,00	**90,53±0,00**	0,45	0,45
	Waveform	80,54±0,25	**81,95±0,21**	81,79±0,27	1,25	-0,16
Abrupt-50K	$Agrawal_1$	**75,70±0,23**	74,96±0,15	75,40±0,28	-0,30	0,44
	RBF	32,10±0,28	32,48±0,30	**32,63±0,24**	0,53	0,15
	SEA	90,81±0,00	**91,28±0,00**	91,02±0,00	0,21	-0,26
	Waveform	81,56±0,20	**82,99±0,10**	82,83±0,16	1,27	-0,16
Abrupt-100K	$Agrawal_1$	**76,90±0,16**	76,59±0,20	76,44±0,18	-0,46	-0,15
	RBF	32,40±0,22	32,76±0,25	**32,92±0,24**	0,52	0,16
	SEA	90,94±0,00	91,81±0,00	**91,99±0,00**	1,05	0,18
	Waveform	82,20±0,18	**83,56±0,11**	83,41±0,14	1,21	-0,15
Abrupt-1M	$Agrawal_1$	78,13±0,08	78,61±0,12	**78,80±0,21**	0,67	0,19
	RBF	34,77±0,19	35,18±0,17	**35,21±0,15**	0,44	0,03
	SEA	92,37±0,00	**93,39±0,00**	93,20±0,00	0,83	-0,19
	Waveform	83,03±0,05	**84,16±0,03**	84,04±0,06	1,01	-0,12
Gradual-10K	$Agrawal_1$	68,02±0,39	67,64±0,42	**68,64±0,63**	0,62	1,00
	RBF	31,31±0,39	**31,69±0,41**	31,59±0,46	0,28	-0,10
	SEA	88,22±0,00	88,58±0,00	**88,96±0,00**	0,74	0,38
	Waveform	79,23±0,33	**80,19±0,36**	80,11±0,34	0,88	-0,08
Gradual-20K	$Agrawal_1$	71,29±0,49	71,49±0,26	**72,00±0,39**	0,71	0,51
	RBF	31,88±0,42	**32,10±0,38**	31,98±0,36	0,10	-0,12
	SEA	89,97±0,00	89,73±0,00	**90,09±0,00**	0,12	0,36
	Waveform	80,26±0,23	81,87±0,24	**81,90±0,21**	1,64	0,03
Gradual-50K	$Agrawal_1$	**75,21±0,22**	74,71±0,16	74,94±0,24	-0,27	0,23
	RBF	32,02±0,33	32,47±0,33	**32,53±0,27**	0,51	0,06
	SEA	90,80±0,00	**90,89±0,00**	90,77±0,00	-0,03	-0,12
	Waveform	81,50±0,15	**82,96±0,13**	82,80±0,21	1,30	-0,16
Gradual-100K	$Agrawal_1$	**76,76±0,13**	76,52±0,13	76,42±0,17	-0,34	-0,10
	RBF	32,41±0,23	32,74±0,22	**32,91±0,20**	0,50	0,17
	SEA	91,03±0,00	91,46±0,00	**91,83±0,00**	0,80	0,37
	Waveform	82,24±0,18	**83,59±0,10**	83,51±0,13	1,27	-0,08
Gradual-1M	$Agrawal_1$	78,22±0,17	**78,75±0,14**	**78,75±0,17**	0,53	0,00
	RBF	34,93±0,15	35,08±0,16	**35,24±0,14**	0,31	0,16
	SEA	92,22±0,00	93,24±0,00	**93,27±0,00**	1,05	0,03
	Waveform	83,07±0,05	**84,16±0,03**	84,08±0,05	1,01	-0,08
Real	Census	94,74	94,73	**94,88**	0,14	0,15
	Connect4	**77,04**	76,82	76,64	-0,40	-0,18
	Cover,Sort,	74,72	**76,72**	76,69	1,97	-0,03
	Electricity	90,23	90,18	**90,80**	0,57	0,62
	Pokerhand	**78,30**	78,08	78,20	-0,10	0,12
	Sensor	89,81	**89,88**	89,72	-0,09	-0,16
	Spam	97,17	97,16	**97,50**	0,33	0,34
	Weather	73,37	73,48	**73,54**	0,17	0,06
	WineWhite	45,73	45,34	**45,76**	0,03	0,42
Rank		2,5408	1,8571	**1,6200**		

Fonte: O autor (2022)

Tabela 10 – Média das acurácias de OABM1, OABM1-λ6 e OABM1-PE em porcentagem usando HT e RDDM, como detector auxiliar, com intervalos de confiança de 95%, em conjuntos de dados artificiais e reais.

Tipo-Inst.	Bases	OABM1	OABM1-λ6	OABM1-PE	Diff1	Diff6
Abrupt-10K	$Agrawal_1$	**68,42±0,72**	60,93±1,26	67,72±0,94	-0,70	6,79
	RBF	20,12±1,59	**21,34±0,98**	20,43±1,32	0,31	-0,91
	SEA	**90,22±0,00**	88,85±0,00	89,46±0,00	-0,76	0,61
	Waveform	80,26±0,42	**81,56±0,29**	80,99±0,25	0,73	-0,57
Abrupt-20K	$Agrawal_1$	**72,14±0,55**	63,96±1,23	71,01±1,09	-1,13	7,05
	RBF	19,95±1,50	**21,24±1,08**	20,92±1,10	0,97	-0,32
	SEA	**90,96±0,00**	89,36±0,00	90,15±0,00	-0,81	0,79
	Waveform	81,07±0,24	**82,38±0,19**	82,18±0,20	1,11	-0,20
Abrupt-50K	$Agrawal_1$	**75,25±0,31**	69,65±0,62	74,08±0,52	-1,17	4,43
	RBF	20,10±1,30	**20,98±1,01**	20,95±1,19	0,85	-0,03
	SEA	91,13±0,00	**91,40±0,00**	91,01±0,00	-0,12	-0,39
	Waveform	81,86±0,12	**82,83±0,13**	82,80±0,13	0,94	-0,03
Abrupt-100K	$Agrawal_1$	**77,19±0,27**	72,41±0,55	75,86±0,46	-1,33	3,45
	RBF	19,03±1,03	19,95±1,17	**20,28±0,87**	1,25	0,33
	SEA	90,73±0,00	92,03±0,00	**92,14±0,00**	1,41	0,11
	Waveform	82,48±0,13	**83,08±0,09**	83,06±0,09	0,58	-0,02
Abrupt-1M	$Agrawal_1$	**80,13±0,22**	77,86±0,12	79,16±0,22	-0,97	1,30
	RBF	18,67±0,49	**20,65±0,44**	19,74±0,52	1,07	-0,91
	SEA	92,14±0,00	**93,48±0,00**	93,36±0,00	1,22	-0,12
	Waveform	83,20±0,04	83,29±0,03	**83,36±0,04**	0,16	0,07
Gradual-10K	$Agrawal_1$	**66,50±0,52**	58,32±0,66	64,94±1,06	-1,56	6,62
	RBF	19,22±1,56	**21,63±0,88**	20,78±1,30	1,56	-0,85
	SEA	**89,09±0,00**	88,33±0,00	88,66±0,00	-0,43	0,33
	Waveform	79,55±0,38	**81,40±0,27**	80,92±0,32	1,37	-0,48
Gradual-20K	$Agrawal_1$	**70,88±0,45**	62,86±0,71	70,10±1,02	-0,78	7,24
	RBF	19,34±1,50	**20,82±0,92**	20,47±1,23	1,13	-0,35
	SEA	**90,37±0,00**	89,71±0,00	89,55±0,00	-0,82	-0,16
	Waveform	80,82±0,26	**82,30±0,19**	81,97±0,21	1,15	-0,33
Gradual-50K	$Agrawal_1$	**74,70±0,31**	69,08±0,75	72,93±0,53	-1,77	3,85
	RBF	20,07±1,21	20,51±1,03	**21,12±1,23**	1,05	0,61
	SEA	91,02±0,00	**91,31±0,00**	90,92±0,00	-0,10	-0,39
	Waveform	81,74±0,15	82,78±0,09	**82,81±0,13**	1,07	0,03
Gradual-100K	$Agrawal_1$	**77,07±0,31**	71,71±0,44	75,78±0,41	-1,29	4,07
	RBF	18,71±1,05	20,49±0,96	**20,57±1,11**	1,86	0,08
	SEA	91,03±0,00	**92,18±0,00**	92,04±0,00	1,01	-0,14
	Waveform	82,33±0,14	83,00±0,08	**83,04±0,10**	0,71	0,04
Gradual-1M	$Agrawal_1$	**80,24±0,26**	78,04±0,16	79,09±0,21	-1,15	1,05
	RBF	18,33±0,68	**20,88±0,51**	19,68±0,49	1,35	-1,20
	SEA	92,41±0,00	**93,53±0,00**	93,41±0,00	1,00	-0,12
	Waveform	83,19±0,05	83,28±0,03	**83,34±0,04**	0,15	0,06
Real	Census	94,92	94,92	**95,18**	0,26	0,26
	Connect4	75,61	74,80	**76,08**	0,47	1,28
	Cover,Sort,	76,93	86,72	**88,16**	11,23	1,44
	Electricity	90,25	91,70	**91,93**	1,68	0,23
	Pokerhand	78,07	89,37	**90,28**	12,21	0,91
	Sensor	90,47	92,72	**92,77**	2,30	0,05
	Spam	**97,68**	97,30	97,55	-0,13	0,25
	Weather	74,73	**75,07**	74,00	-0,73	-1,07
	WineWhite	39,34	42,75	**43,48**	4,14	0,73
Rank		2,2959	1,9286	**1,7755**		

Fonte: O autor (2022)

Tabela 11 – Média das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE em porcentagem usando HT e RDDM, como detector auxiliar, com intervalos de confiança de 95%, em conjuntos de dados artificiais e reais.

Tipo-Inst.	Bases	OzaBag	OzaBag-λ6	OzaBag-PE	Diff1	Diff6
Abrupt-10K	$Agrawal_1$	69,00±0,39	64,46±1,31	**69,39±0,95**	0,39	4,93
	RBF	31,84±0,41	**32,03±0,36**	32,02±0,44	0,18	-0,01
	SEA	**90,42±0,00**	89,00±0,00	89,91±0,00	-0,51	0,91
	Waveform	79,08±0,47	80,27±0,39	**80,48±0,50**	1,40	0,21
Abrupt-20K	$Agrawal_1$	**73,04±0,43**	68,08±1,22	72,16±0,80	-0,88	4,08
	RBF	32,11±0,36	32,44±0,31	**32,83±0,41**	0,72	0,39
	SEA	**91,55±0,00**	90,96±0,00	90,56±0,00	-0,99	-0,40
	Waveform	79,95±0,27	81,38±0,30	**81,82±0,41**	1,87	0,44
Abrupt-50K	$Agrawal_1$	**77,29±0,29**	72,55±0,92	76,28±0,63	-1,01	3,73
	RBF	32,54±0,24	**34,60±0,27**	33,93±0,36	1,39	-0,67
	SEA	91,48±0,00	92,17±0,00	**92,35±0,00**	0,87	0,18
	Waveform	80,82±0,17	**83,10±0,20**	82,93±0,32	2,11	-0,17
Abrupt-100K	$Agrawal_1$	**79,61±0,28**	74,73±0,53	77,66±0,61	-1,95	2,93
	RBF	32,91±0,24	**35,72±0,18**	35,05±0,41	2,14	-0,67
	SEA	91,80±0,00	**93,09±0,00**	93,02±0,00	1,22	-0,07
	Waveform	81,05±0,17	**83,80±0,13**	83,23±0,22	2,18	-0,57
Abrupt-1M	$Agrawal_1$	**82,81±0,24**	79,90±0,22	81,54±0,22	-1,27	1,64
	RBF	34,92±0,15	**38,06±0,09**	37,39±0,18	2,47	-0,67
	SEA	92,99±0,00	**94,10±0,00**	94,07±0,00	1,08	-0,03
	Waveform	82,52±0,12	**84,68±0,05**	84,36±0,08	1,84	-0,32
Gradual-10K	$Agrawal_1$	67,05±0,32	64,19±1,10	**67,09±1,26**	0,04	2,90
	RBF	31,78±0,39	**31,97±0,37**	31,65±0,43	-0,13	-0,32
	SEA	**90,40±0,00**	88,54±0,00	89,65±0,00	-0,75	1,11
	Waveform	78,31±0,42	**80,40±0,34**	80,30±0,50	1,99	-0,10
Gradual-20K	$Agrawal_1$	**71,84±0,31**	67,55±1,29	70,83±0,81	-1,01	3,28
	RBF	32,17±0,30	32,66±0,33	**32,73±0,39**	0,56	0,07
	SEA	**91,16±0,00**	91,08±0,00	90,09±0,00	-1,07	-0,99
	Waveform	79,53±0,27	81,61±0,26	**81,87±0,49**	2,34	0,26
Gradual-50K	$Agrawal_1$	**76,45±0,27**	72,50±0,76	75,00±0,59	-1,45	2,50
	RBF	32,48±0,27	**34,57±0,28**	34,13±0,37	1,65	-0,44
	SEA	91,37±0,00	**92,50±0,00**	91,69±0,00	0,32	-0,81
	Waveform	80,83±0,14	**83,18±0,18**	82,94±0,27	2,11	-0,24
Gradual-100K	$Agrawal_1$	**78,91±0,31**	75,05±0,51	77,38±0,39	-1,53	2,33
	RBF	32,85±0,21	**35,85±0,17**	35,36±0,34	2,51	-0,49
	SEA	91,58±0,00	**93,13±0,00**	92,74±0,00	1,16	-0,39
	Waveform	80,91±0,14	**83,78±0,14**	83,54±0,19	2,63	-0,24
Gradual-1M	$Agrawal_1$	**82,44±0,24**	79,82±0,17	81,25±0,27	-1,19	1,43
	RBF	34,88±0,16	**38,07±0,07**	37,42±0,19	2,54	-0,65
	SEA	93,03±0,00	**94,13±0,00**	94,01±0,00	0,98	-0,12
	Waveform	82,53±0,11	**84,69±0,05**	84,39±0,08	1,86	-0,30
Real	Census	94,07	94,25	**94,34**	0,27	0,09
	Connect4	75,05	**75,88**	75,38	0,33	-0,50
	Cover,Sort,	74,22	**82,13**	81,93	7,71	-0,20
	Electricity	84,64	88,05	**88,22**	3,58	0,17
	Pokerhand	76,81	78,90	**82,28**	5,47	3,38
	Sensor	86,56	87,76	**88,27**	1,71	0,51
	Spam	92,24	94,52	**95,57**	3,33	1,05
	Weather	72,88	**75,34**	74,30	1,42	-1,04
	WineWhite	47,50	48,91	**49,63**	2,13	0,72
Rank		2,4490	**1,7755**	**1,7755**		

Fonte: O autor (2022)

Tabela 12 – Média das acurácias de BOLE, BOLE-λ6 e BOLE-PE em porcentagem usando HT e $HDDM_A$, como detector auxiliar, com intervalos de confiança de 95%, em conjuntos de dados artificiais e reais.

Tipo-Inst.	Bases	BOLE	BOLE-λ6	BOLE-PE	Diff1	Diff6
Abrupt-10K	$Agrawal_1$	69,84±0,47	69,77±0,49	**70,22±0,66**	0,38	0,45
	RBF	31,36±0,40	**31,84±0,45**	31,47±0,47	0,11	-0,37
	SEA	88,68±0,00	88,68±0,00	**89,02±0,00**	0,34	0,34
	Waveform	79,29±0,42	**80,72±0,37**	80,34±0,46	1,05	-0,38
Abrupt-20K	$Agrawal_1$	72,06±0,40	72,14±0,34	**72,43±0,55**	0,37	0,29
	RBF	31,88±0,35	**31,99±0,44**	31,65±0,41	-0,23	-0,34
	SEA	89,99±0,00	**90,45±0,00**	90,18±0,00	0,19	-0,27
	Waveform	80,45±0,24	**82,00±0,24**	81,75±0,29	1,30	-0,25
Abrupt-50K	$Agrawal_1$	75,06±0,45	74,98±0,15	**75,42±0,25**	0,36	0,44
	RBF	32,01±0,34	**32,69±0,29**	32,52±0,36	0,51	-0,17
	SEA	90,68±0,00	91,24±0,00	**91,53±0,00**	0,85	0,29
	Waveform	81,74±0,17	**82,95±0,12**	82,82±0,19	1,08	-0,13
Abrupt-100K	$Agrawal_1$	76,30±0,50	76,61±0,35	**76,72±0,22**	0,42	0,11
	RBF	32,32±0,31	**33,22±0,29**	33,09±0,21	0,77	-0,13
	SEA	91,57±0,00	**92,13±0,00**	91,72±0,00	0,15	-0,41
	Waveform	82,53±0,18	**83,80±0,09**	83,63±0,16	1,10	-0,17
Abrupt-1M	$Agrawal_1$	78,76±0,17	**79,23±0,16**	79,22±0,24	0,46	-0,01
	RBF	35,97±0,20	**36,10±0,25**	36,08±0,26	0,11	-0,02
	SEA	93,45±0,00	**93,97±0,00**	93,95±0,00	0,50	-0,02
	Waveform	83,82±0,06	**84,53±0,03**	84,45±0,08	0,63	-0,08
Gradual-10K	$Agrawal_1$	**68,25±0,45**	67,24±0,65	67,54±0,67	-0,71	0,30
	RBF	31,56±0,54	**31,64±0,58**	31,42±0,48	-0,14	-0,22
	SEA	88,23±0,00	88,40±0,00	**88,42±0,00**	0,19	0,02
	Waveform	78,61±0,41	**80,49±0,43**	80,22±0,50	1,61	-0,27
Gradual-20K	$Agrawal_1$	**71,13±0,24**	70,93±0,49	71,11±0,56	-0,02	0,18
	RBF	31,55±0,48	**32,07±0,43**	31,93±0,36	0,38	-0,14
	SEA	89,91±0,00	**89,99±0,00**	89,97±0,00	0,06	-0,02
	Waveform	80,05±0,22	81,88±0,23	**81,90±0,30**	1,85	0,02
Gradual-50K	$Agrawal_1$	74,51±0,43	**74,67±0,14**	74,63±0,30	0,12	-0,04
	RBF	31,98±0,30	**32,44±0,31**	32,39±0,28	0,41	-0,05
	SEA	90,88±0,00	**91,42±0,00**	91,13±0,00	0,25	-0,29
	Waveform	81,69±0,21	**83,13±0,16**	82,93±0,23	1,24	-0,20
Gradual-100K	$Agrawal_1$	76,32±0,41	**76,55±0,12**	76,52±0,18	0,20	-0,03
	RBF	32,45±0,30	32,90±0,26	**33,13±0,27**	0,68	0,23
	SEA	91,47±0,00	92,02±0,00	**92,04±0,00**	0,57	0,02
	Waveform	82,70±0,19	**83,79±0,10**	83,65±0,16	0,95	-0,14
Gradual-1M	$Agrawal_1$	78,63±0,11	**79,29±0,19**	79,20±0,15	0,57	-0,09
	RBF	36,09±0,26	**36,10±0,30**	35,92±0,26	-0,17	-0,18
	SEA	93,48±0,00	**93,96±0,00**	93,85±0,00	0,37	-0,11
	Waveform	83,81±0,05	**84,52±0,03**	84,45±0,07	0,64	-0,07
Real	Census	94,59	**94,68**	94,65	0,06	-0,03
	Connect4	76,89	**76,95**	76,69	-0,20	-0,26
	Cover,Sort,	72,73	72,73	**79,38**	6,65	6,65
	Electricity	89,35	**90,05**	89,75	0,40	-0,30
	Pokerhand	77,50	77,47	**77,54**	0,04	0,07
	Sensor	89,79	89,43	**90,01**	0,22	0,58
	Spam	97,13	**97,59**	97,55	0,42	-0,04
	Weather	73,24	**73,70**	73,19	-0,05	-0,51
	WineWhite	45,68	44,63	**47,25**	1,57	2,62
Rank		2,6939	**1,4898**	1,8163		

Fonte: O autor (2022)

Tabela 13 – Média das acurácias de OABM1, OABM1-λ6 e OABM1-PE em porcentagem usando HT e $HDDM_A$, como detector auxiliar, com intervalos de confiança de 95%, em conjuntos de dados artificiais e reais.

Tipo-Inst.	Bases	OABM1	OABM1-λ6	OABM1-PE	Diff1	Diff6
Abrupt-10K	$Agrawal_1$	**68,10±0,75**	58,86±0,73	67,97±1,06	-0,13	9,11
	RBF	19,30±1,99	20,87±1,37	**21,01±1,47**	1,71	0,14
	SEA	88,52±0,00	88,86±0,00	**89,74±0,00**	1,22	0,88
	Waveform	79,98±0,46	**81,49±0,29**	81,13±0,30	1,15	-0,36
Abrupt-20K	$Agrawal_1$	**71,92±0,55**	62,89±0,94	70,41±1,02	-1,51	7,52
	RBF	18,73±1,91	20,45±1,29	**20,51±1,41**	1,78	0,06
	SEA	**90,71±0,00**	89,55±0,00	89,86±0,00	-0,85	0,31
	Waveform	80,97±0,25	**82,32±0,20**	81,98±0,22	1,01	-0,34
Abrupt-50K	$Agrawal_1$	**75,27±0,37**	68,60±0,84	73,76±0,67	-1,51	5,16
	RBF	18,38±1,76	20,05±1,25	**20,49±1,28**	2,11	0,44
	SEA	91,23±0,00	**91,65±0,00**	91,25±0,00	0,02	-0,40
	Waveform	81,99±0,15	**83,04±0,12**	82,92±0,14	0,93	-0,12
Abrupt-100K	$Agrawal_1$	**77,26±0,40**	72,14±0,58	75,65±0,48	-1,61	3,51
	RBF	18,39±1,62	20,50±1,02	**20,56±1,23**	2,17	0,06
	SEA	91,68±0,00	**92,37±0,00**	91,94±0,00	0,26	-0,43
	Waveform	82,56±0,12	**83,25±0,12**	83,22±0,12	0,66	-0,03
Abrupt-1M	$Agrawal_1$	**81,42±0,35**	78,85±0,23	80,15±0,35	-1,27	1,30
	RBF	18,87±1,11	20,05±0,83	**21,16±1,01**	2,29	1,11
	SEA	93,75±0,00	**94,02±0,00**	94,01±0,00	0,26	-0,01
	Waveform	**83,95±0,04**	83,70±0,04	83,75±0,06	-0,20	0,05
Gradual-10K	$Agrawal_1$	**66,35±0,71**	58,21±0,52	63,61±0,77	-2,74	5,40
	RBF	19,10±2,07	20,44±1,31	**20,67±1,57**	1,57	0,23
	SEA	88,29±0,00	88,30±0,00	**89,10±0,00**	0,81	0,80
	Waveform	79,22±0,39	**81,42±0,27**	80,90±0,29	1,68	-0,52
Gradual-20K	$Agrawal_1$	**70,55±0,61**	61,56±0,96	67,78±1,18	-2,77	6,22
	RBF	18,50±1,91	19,95±1,27	**20,57±1,54**	2,07	0,62
	SEA	**90,38±0,00**	90,03±0,00	89,89±0,00	-0,49	-0,14
	Waveform	80,46±0,23	**82,37±0,16**	82,01±0,19	1,55	-0,36
Gradual-50K	$Agrawal_1$	**74,30±0,35**	68,35±0,81	72,68±0,83	-1,62	4,33
	RBF	18,06±1,63	20,22±1,26	**20,59±1,24**	2,53	0,37
	SEA	90,98±0,00	**91,42±0,00**	91,12±0,00	0,14	-0,30
	Waveform	81,95±0,16	**83,03±0,10**	83,01±0,12	1,06	-0,02
Gradual-100K	$Agrawal_1$	**76,63±0,22**	71,58±0,51	75,20±0,46	-1,43	3,62
	RBF	18,26±1,74	**20,18±1,29**	19,58±1,38	1,32	-0,60
	SEA	91,44±0,00	**92,34±0,00**	92,11±0,00	0,67	-0,23
	Waveform	82,68±0,12	**83,32±0,10**	**83,32±0,11**	0,64	0,00
Gradual-1M	$Agrawal_1$	**81,26±0,38**	78,41±0,27	79,41±0,28	-1,85	1,00
	RBF	19,32±1,21	20,24±1,01	**20,78±1,22**	1,46	0,54
	SEA	93,73±0,00	**94,05±0,00**	93,99±0,00	0,26	-0,06
	Waveform	**83,92±0,05**	83,69±0,05	83,78±0,05	-0,14	0,09
Real	Census	94,65	**94,96**	94,90	0,25	-0,06
	Connect4	74,92	74,81	**75,80**	0,88	0,99
	Cover,Sort,	73,16	90,09	**90,68**	17,52	0,59
	Electricity	89,48	91,05	**91,14**	1,66	0,09
	Pokerhand	78,00	**87,22**	85,89	7,89	-1,33
	Sensor	90,43	92,20	**92,64**	2,21	0,44
	Spam	**97,12**	96,98	96,96	-0,16	-0,02
	Weather	74,52	73,76	**74,53**	0,01	0,77
	WineWhite	35,16	32,95	**37,98**	2,82	5,03
Rank		2,3265	1,9898	**1,6837**		

Fonte: O autor (2022)

Tabela 14 – Média das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE em porcentagem usando HT e $HDDM_A$, como detector auxiliar, com intervalos de confiança de 95%, em conjuntos de dados artificiais e reais.

Tipo-Inst.	Bases	OzaBag	OzaBag-λ6	OzaBag-PE	Diff1	Diff6
Abrupt-10K	$Agrawal_1$	68,64±0,44	65,15±1,25	**68,91±1,08**	0,27	3,76
	RBF	31,82±0,43	32,16±0,41	**32,19±0,48**	0,37	0,03
	SEA	89,17±0,00	88,44±0,00	**89,22±0,00**	0,05	0,78
	Waveform	78,74±0,49	80,32±0,36	**80,72±0,48**	1,98	0,40
Abrupt-20K	$Agrawal_1$	**73,07±0,40**	68,45±1,21	72,73±0,80	-0,34	4,28
	RBF	32,23±0,39	**33,16±0,42**	32,97±0,44	0,74	-0,19
	SEA	**91,02±0,00**	90,35±0,00	90,23±0,00	-0,79	-0,12
	Waveform	79,84±0,27	81,32±0,31	**81,64±0,38**	1,80	0,32
Abrupt-50K	$Agrawal_1$	**77,27±0,26**	72,09±0,90	75,80±0,83	-1,47	3,71
	RBF	32,54±0,30	**34,52±0,22**	33,86±0,42	1,32	-0,66
	SEA	91,22±0,00	**92,32±0,00**	91,74±0,00	0,52	-0,58
	Waveform	80,78±0,17	**83,17±0,19**	82,72±0,40	1,94	-0,45
Abrupt-100K	$Agrawal_1$	**79,47±0,30**	74,77±0,69	77,70±0,54	-1,77	2,93
	RBF	33,10±0,29	**35,66±0,25**	34,83±0,36	1,73	-0,83
	SEA	91,92±0,00	**93,04±0,00**	92,74±0,00	0,82	-0,30
	Waveform	81,18±0,16	**83,90±0,14**	83,71±0,28	2,53	-0,19
Abrupt-1M	$Agrawal_1$	**83,59±0,20**	80,46±0,22	81,95±0,36	-1,64	1,49
	RBF	35,77±0,26	**38,80±0,10**	38,15±0,28	2,38	-0,65
	SEA	94,05±0,00	**94,48±0,00**	94,40±0,00	0,35	-0,08
	Waveform	84,11±0,10	**85,25±0,04**	85,14±0,07	1,03	-0,11
Gradual-10K	$Agrawal_1$	**66,59±0,39**	61,98±0,88	65,97±0,93	-0,62	3,99
	RBF	31,86±0,47	32,25±0,44	**32,27±0,60**	0,41	0,02
	SEA	88,02±0,00	88,36±0,00	**88,62±0,00**	0,60	0,26
	Waveform	77,91±0,45	80,37±0,36	**80,53±0,45**	2,62	0,16
Gradual-20K	$Agrawal_1$	**71,39±0,30**	66,59±1,18	70,93±0,67	-0,46	4,34
	RBF	32,31±0,37	**33,46±0,33**	32,84±0,42	0,53	-0,62
	SEA	**90,63±0,00**	90,24±0,00	90,20±0,00	-0,43	-0,04
	Waveform	79,39±0,24	82,02±0,25	**82,17±0,43**	2,78	0,15
Gradual-50K	$Agrawal_1$	**75,76±0,57**	71,22±1,02	74,62±0,68	que-1,14	3,40
	RBF	32,64±0,32	**34,51±0,27**	33,90±0,41	1,26	-0,61
	SEA	91,38±0,00	**92,19±0,00**	91,80±0,00	0,42	-0,39
	Waveform	80,71±0,19	**83,53±0,18**	83,19±0,33	2,48	-0,34
Gradual-100K	$Agrawal_1$	**78,59±0,28**	74,10±0,53	77,07±0,55	-1,52	2,97
	RBF	33,09±0,26	**35,64±0,23**	35,02±0,41	1,93	-0,62
	SEA	91,96±0,00	**92,86±0,00**	92,73±0,00	0,77	-0,13
	Waveform	81,09±0,16	**83,99±0,15**	83,65±0,32	2,56	-0,34
Gradual-1M	$Agrawal_1$	**83,09±0,35**	80,36±0,22	82,02±0,26	-1,07	1,66
	RBF	35,52±0,16	**38,80±0,11**	38,12±0,28	2,60	-0,68
	SEA	94,03±0,00	**94,50±0,00**	94,48±0,00	0,45	-0,02
	Waveform	84,09±0,08	**85,29±0,04**	85,09±0,06	1,00	-0,20
Real	Census	94,35	94,48	**94,59**	0,24	0,11
	Connect4	74,94	75,64	**75,72**	0,78	0,08
	Cover,Sort,	73,66	77,51	**87,20**	13,54	9,69
	Electricity	85,67	**88,21**	88,14	2,47	-0,07
	Pokerhand	76,36	78,71	**79,69**	3,33	0,98
	Sensor	86,64	88,04	**88,27**	1,63	0,23
	Spam	92,63	96,30	**97,22**	4,59	0,92
	Weather	72,37	**75,00**	74,20	1,83	-0,80
	WineWhite	47,13	**47,45**	46,13	-1,00	-1,32
Rank		2,4898	**1,7551**	**1,7551**		

Fonte: O autor (2022)

testes.

Em relação aos conjuntos de dados do mundo real, OzaBag-PE foi o melhor método em 6 dos 9 cenários testados (66,66%) e OzaBag-λ6 apresentou os melhores resultados nos três conjuntos de dados restantes.

Já os resultados da Tabela 12 onde foi utilizado $HDDM_A$ como detector auxiliar nos métodos, o BOLE-λ6 obteve o melhor desempenho sendo superior do que as outras abordagens principalmente nos testes com as bases abruptas e graduais de 1M. Os valores resultantes das acurácias de BOLE-PE somente foram superiores na maioria dos testes com $Agrawal_1$ nas bases abruptas e, nas bases graduais em alguns casos com SEA e $Waveform$.

Por conseguinte, analisando os valores das diferenças entre BOLE-PE e BOLE (Diff1), BOLE-PE foi melhor na maioria das bases experimentadas. Nas comparações entre BOLE-PE e BOLE-λ6, este último obteve os melhores resultados embora as diferenças de BOLE-PE não tenham sido significativas.

De outro lado, nos resultados com as bases reais, BOLE-λ6 e BOLE-PE foram os melhores métodos obtendo os maiores valores de acurácia em 5 do total de bases testadas no caso de BOLE-λ6 e 4 dos 9 conjuntos reais experimentados no caso de BOLE-PE.

Seguidamente, os valores das acurácias de OABM1, OABM1-λ6 e OABM1-PE da Tabela 13, usando o detector auxiliar $HDDM_A$ evidenciaram que OABM1 foi superior em todos os conjuntos artificiais de $Agrawal_1$ e em poucos casos com SEA e $Waveform$. Nestas duas bases, OABM1-λ6 obteve a melhor perfomance na maioria dos testes com estas bases artificiais. Já o OABM1-PE foi superior nos testes com RBF.

Em relação às diferenças (Diff1) entre OABM1-PE e OABM1, pode se observar que em muitos casos os valores de OABM1 foram melhores em $Agrawal_1$ apresentando diferenças relevantes. Mas essas diferenças foram ainda maiores em OABM1-PE quando comparado com OABM1 (Diff6) nos testes com essa mesma base artificial. Finalmente nos conjuntos de dados reais, OABM1-PE foi novamente superior obtendo as melhores acurácias em 6 das 9 bases testadas.

Na Tabela 14 onde podem se observar os valores das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE com $HDDM_A$ como detector interno nos métodos, os resultados foram semelhantes aos apresentados com o detector RDDM (Tabela 11).

Adicionalmente, as versões de comitês de classificadores testadas também foram avaliadas usando o teste *Friedman*, conforme prescrito por Demsar (DEMSAR, 2006): os ranks resultantes também foram incluídos nas tabelas 9, 10 e 11 relacionadas com os testes usando o detector

auxiliar RDDM. Estes ranks mostram que as variações BOLE-PE, OABM1-PE e OzaBag-PE foram as de melhor desempenho, seguidas pelas versões com λ igual a 6, e as versões originais (tabelas 9 e 10). Note que na tabela 11, ambos os métodos OzaBag-PE e OzaBag-PE-λ6 foram igualmente ranqueados com valores de 1.7755, sendo os de melhor performance seguidos pela versão original OzaBag com um valor resultante de 2.4490.

Por outro lado, os ranks resultantes também foram inseridos nas tabelas 12, 13, e 14 pertencentes aos testes com o detector interno $HDDM_A$ nos métodos. Em específico, no final da Tabela 12 pode se apreciar que BOLE-λ6 foi o melhor ranqueado seguido de BOLE-PE e BOLE. No entanto, os ranks resultantes do teste estatístico nas avaliações de OABM1, OABM1-λ6 e OABM1 (Tabela 13), mostram que OABM1-PE foi o melhor enfoque. Já nos testes de OzaBag, a versão com λ igual a 6 e OzaBag-PE (Tabela 14), OzaBag-PE e OzaBag-λ6 foram os de melhor desempenho obtendo ranks iguais entre eles.

Para descobrir quais versões de comitês são estatisticamente diferentes nos experimentos, também foi aplicado o pós-teste *Nemenyi* (DEMSAR, 2006). A Figura 13 representa graficamente os resultados com RDDM como detector auxiliar, usando todos os conjuntos de dados separados pelo comitê base, ou seja, BOLE e suas versões (Figura 13a), OABM1 e suas versões (13b), e OzaBag e suas versões (13c).

Seguidamente, a Figura 14 ilustra os resultados com todos os métodos por tipo de conjunto de dados, ou seja, artificial (14a) , bases do mundo real (14b) e todos os conjuntos de dados (14c).

Além do mais, a Figura 15 também ilustra de forma gráfica os resultados com o método de detecção de mudanças $HDDM_A$ como auxiliar, usando todos os conjuntos de dados separados pelo comitê base como explicado acima. Da mesma forma, são apresentadas as ilustrações com os resultados deste detector auxiliar com todos os métodos também por tipo de conjunto de dados na Figura 16. Nesses gráficos, os ranks dos métodos são marcados na régua, a diferença crítica (CD) é representada por barras vermelhas e os métodos conectados por uma barra são estatisticamente semelhantes.

Os resultados dos testes separados por métodos (Figura 13) mostram que as versões modificadas BOLE-PE, OABM1-PE e OzaBag-PE foram os comitês mais bem ranqueados em cada gráfico. Especificamente, tanto BOLE-PE quanto OABM1-PE e OzaBag-PE foram estatisticamente melhores que seus respectivos métodos originais (BOLE, OABM1 e OzaBag), sem diferença significativa para as versões com λ definido como 6. Note que na ilustração 13c referente aos testes com OzaBag e versões, o OzaBag-λ6 obteve o valor do rank igual a

OzaBag-PE, sendo ambos os métodos os que apresentaram o melhor desempenho. Na avaliação do OABM1 e suas versões, também não houve diferenças significativas entre OABM1-λ6 e OABM1.

Nos resultados dos testes com todos os métodos nos conjuntos de dados artificiais (Figura 14a), OzaBag-PE foi o melhor conjunto seguido por OzaBag-λ6, BOLE-PE, BOLE-λ6 e OzaBag, sem diferenças significativas entre eles. No entanto, observe que apenas OzaBag-PE e OzaBag-λ6 foram estatisticamente melhores que OABM1-PE, OABM1-λ6, OABM1 e BOLE, que foram as versões que obtiveram os ranks mais baixos. Também é valido ressaltar que entre estas versões que obtiveram os menores resultados, não houve diferenças significativas.

Por outro lado, nos testes com as bases do mundo real (Figura 14b), os métodos mais bem ranqueados foram OABM1-PE e OABM1-λ6 e ambos foram estatisticamente melhores que OzaBag, sem outras diferenças estatísticas entre os restantes métodos testados.

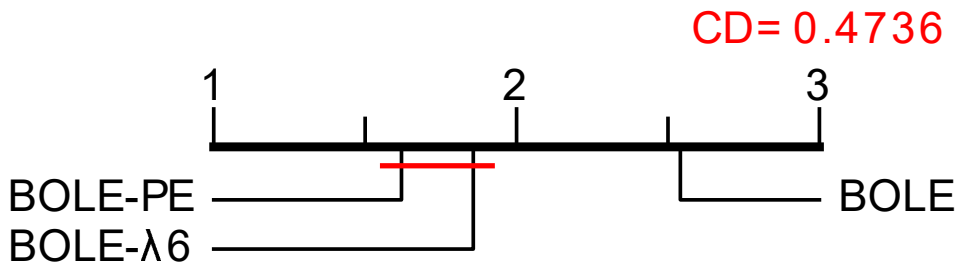
Por último, na avaliação usando todos os conjuntos de dados (Figura 14c), pode-se observar que as versões OzaBag-PE, OzaBag-λ6 e BOLE-PE foram as mais bem ranqueadas, seguidas por BOLE-λ6, OzaBag, OABM1-PE, OABM1-λ6, OABM1 e BOLE. Nesta avaliação, OzaBag-PE foi significativamente melhor do que OzaBag, OABM1-PE, OABM1-λ6, OABM1 e BOLE. Além disso, OzaBag-λ6 também foi superior a OABM1-λ6, OABM1 e BOLE. Já o BOLE-PE teve diferenças significativas em relação a OABM1 e BOLE, os comitês de menores ranks neste cenário.

Sob a avaliação referente aos resultados dos testes separados por métodos (Figura 15) usando HDDM_A como detector auxiliar nos comitês de classificadores é valido ressaltar que estes não foram muito diferentes dos apresentados acima usando RDDM internamente nos métodos. A exceção foi no gráfico 15a relacionado com as versões do Bole. Nesta parte, BOLE-λ6 foi o melhor método seguido de BOLE-PE. Porém é válido ressaltar que ambos enfoques tiveram um desempenho semelhante devido a que não houve diferenças significativas entre eles. Já o BOLE foi o terceiro posicionado e BOLE-λ6 bem como BOLE-PE foram estatisticamente superiores do que ele.

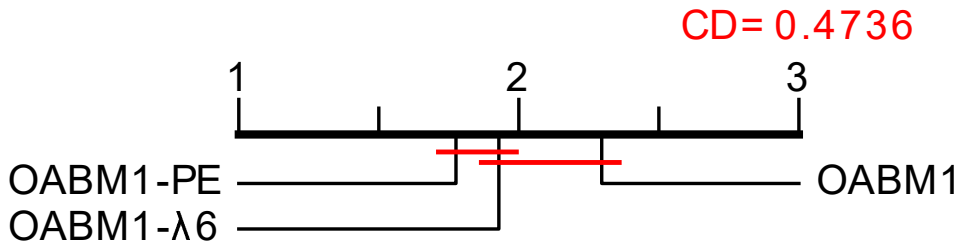
Finalmente, nos resultados do teste de Nemenyi com todos os métodos nos conjuntos de dados artificiais (Figura 16a), OzaBag-PE, OzaBag-λ6 e BOLE-λ6 foram os melhores enfoques na ordem das posições. Nos conjuntos de dados reais (Figura 16b), OABM1-PE obteve o melhor resultado e apresentou diferenças significativas em relação a OzaBag. Na ilustração da Figura 16c com todos os conjuntos, OzaBag-PE foi o método melhor ranqueado sendo estatisticamente superior aos método OzaBag, OABM1-PE, OABM1-λ6, BOLE e OABM1.

Figura 13 – Comparação estatística da acurácia dos métodos (RDDM como detector) usando o teste de Friedman em combinação com o pós-teste de Nemenyi com todos os conjuntos de dados.

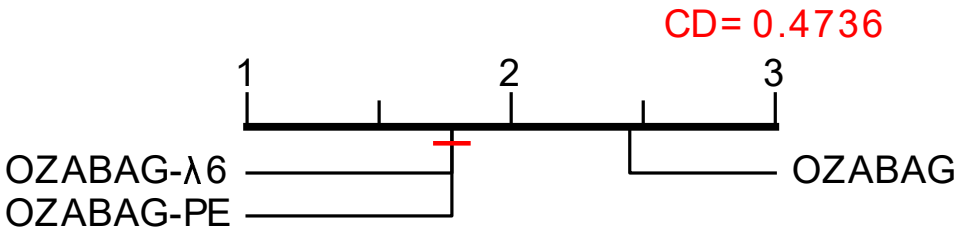
(a) Versões do Bole.



(b) Versões do OABM1.



(c) Versões do OzaBag.

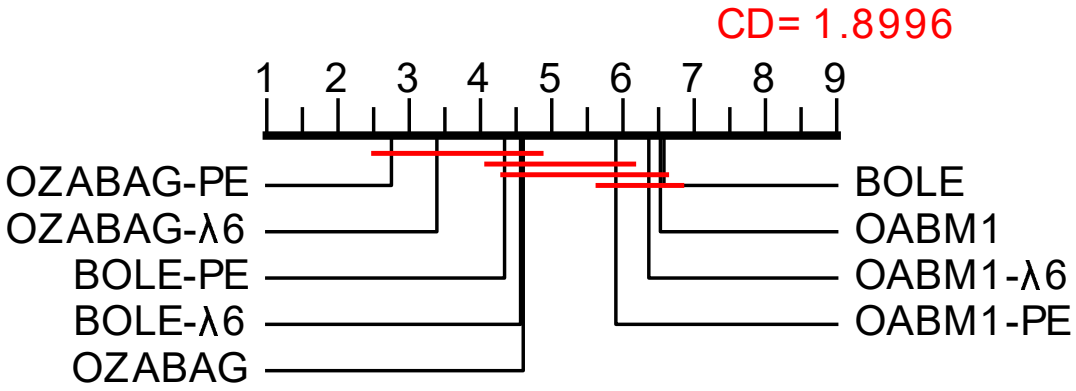


Fonte: O autor (2022)

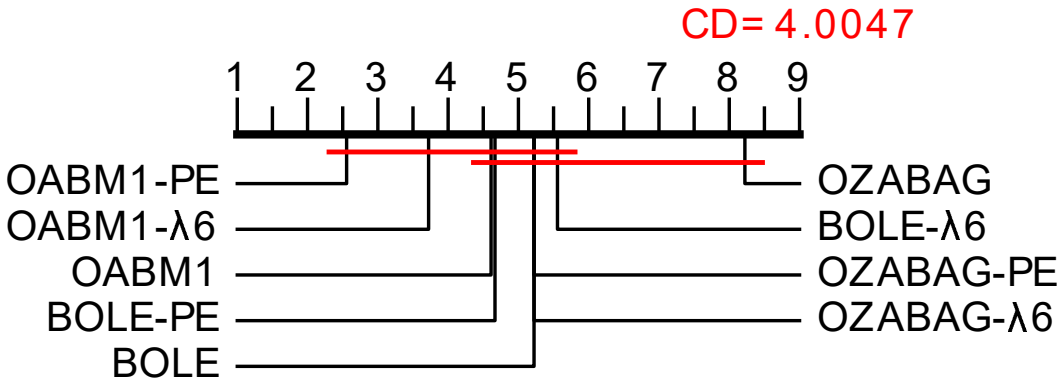
Para concluir esta análise dos experimentos, foram incluídas ilustrações de como PEP funciona ao longo do fluxo. As Figuras 17, 18 e 19 mostram a trajetória dos valores lambda retornados por BOLE-PE, OABM1-PE e OzaBag-PE no processo de aprendizado com os conjuntos de dados $CovertypeSorted$ e $Electricity$. Os diferentes valores retornados ao longo do processo de aprendizagem mostram como o ajuste dinâmico da diversidade aplicado aos comitês de classificadores é importante e impacta no resultado final da avaliação dos métodos. Esses gráficos também ratificam que escolher um λ fixo antes do processo de aprendizagem pode não ser a melhor escolha, independentemente de ser pequeno ou grande o valor deste parâmetro.

Figura 14 – Comparação estatística da acurácia dos métodos (RDDM como detector) usando o teste de Friedman em combinação com o pós-teste de Nemenyi com bases de dados artificiais, reais e todos os conjuntos.

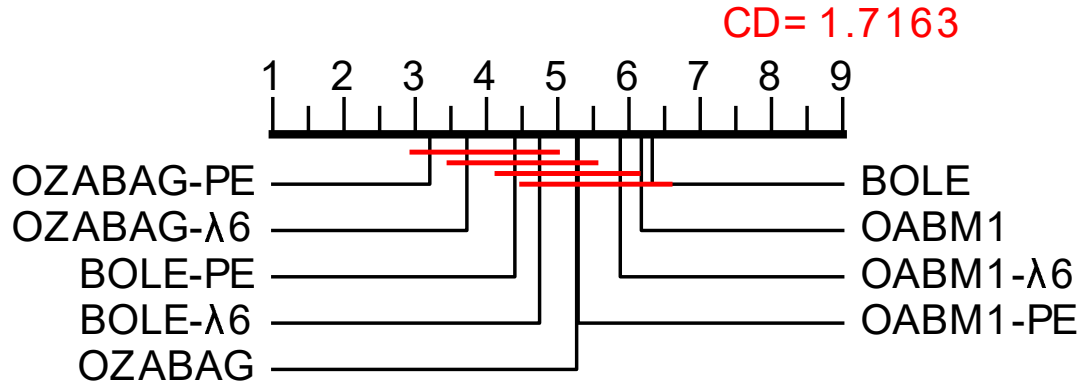
(a) Conjuntos de dados artificiais



(b) Conjuntos de dados reais



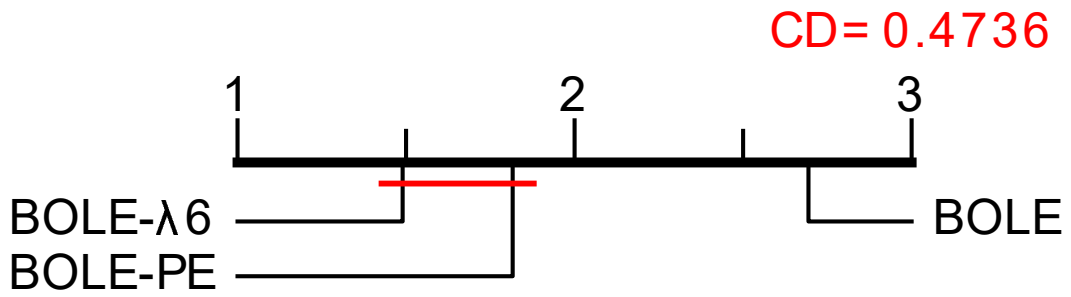
(c) Todos os conjuntos de dados



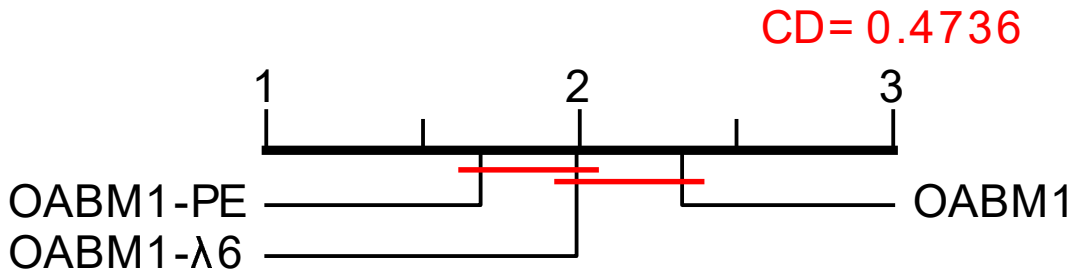
Fonte: O autor (2022)

Figura 15 – Comparação estatística da acurácia dos métodos ($HDDM_A$ como detector) usando o teste de Friedman em combinação com o pós-teste de Nemenyi com todos os conjuntos de dados.

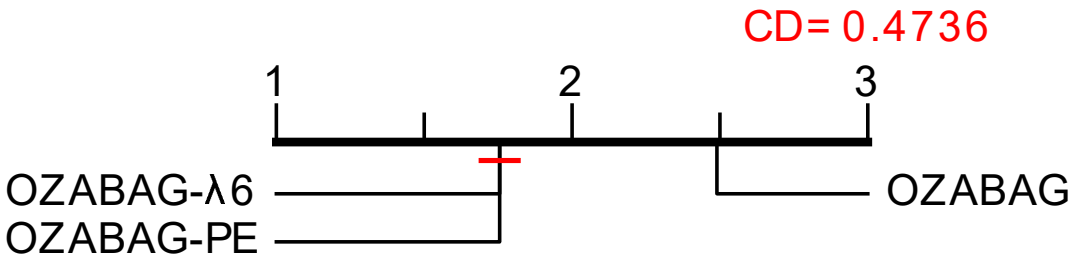
(a) Versões do Bole.



(b) Versões do OABM1.



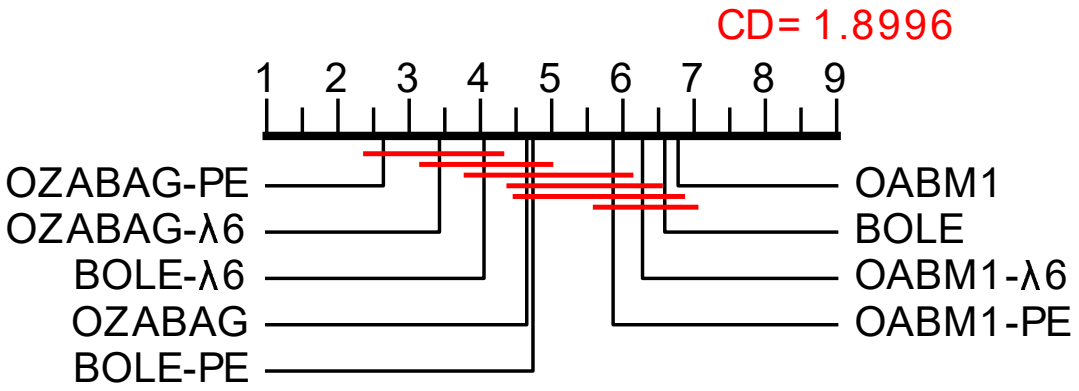
(c) Versões do OzaBag.



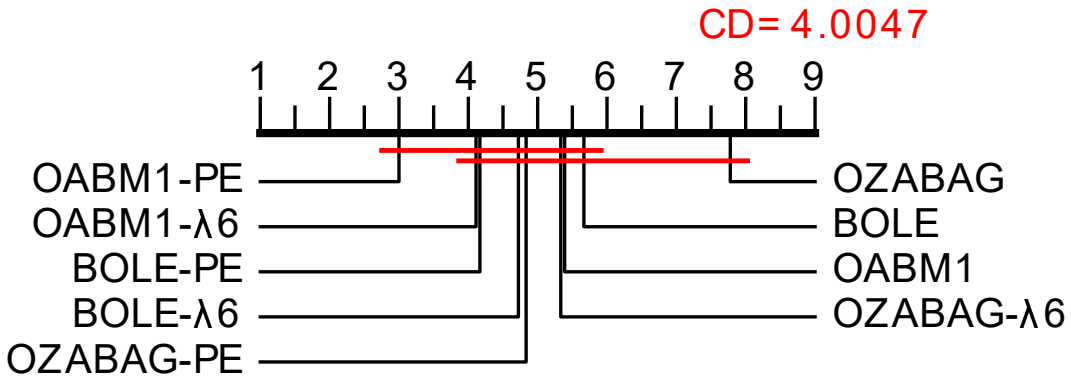
Fonte: O autor (2022)

Figura 16 – Comparação estatística da acurácia dos métodos ($HDDM_A$ como detector) usando o teste de Friedman em combinação com o pós-teste de Nemenyi com bases de dados artificiais, reais e todos os conjuntos.

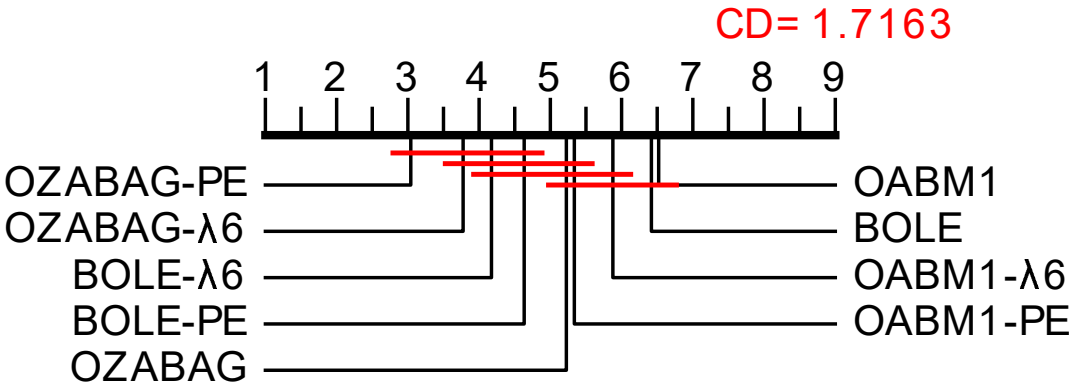
(a) Conjuntos de dados artificiais



(b) Conjuntos de dados reais



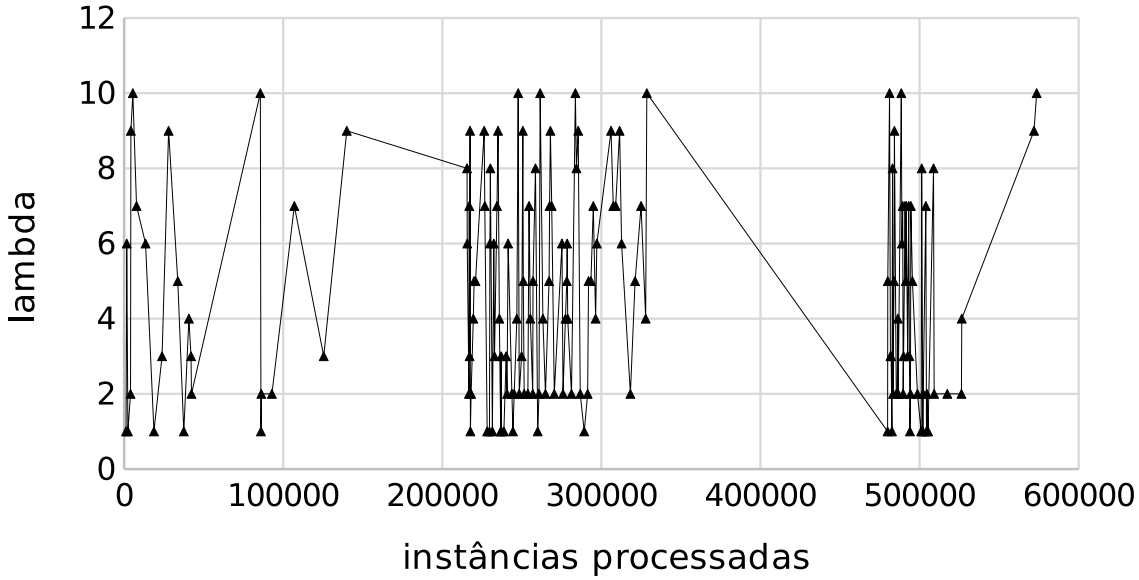
(c) Todos os conjuntos de dados



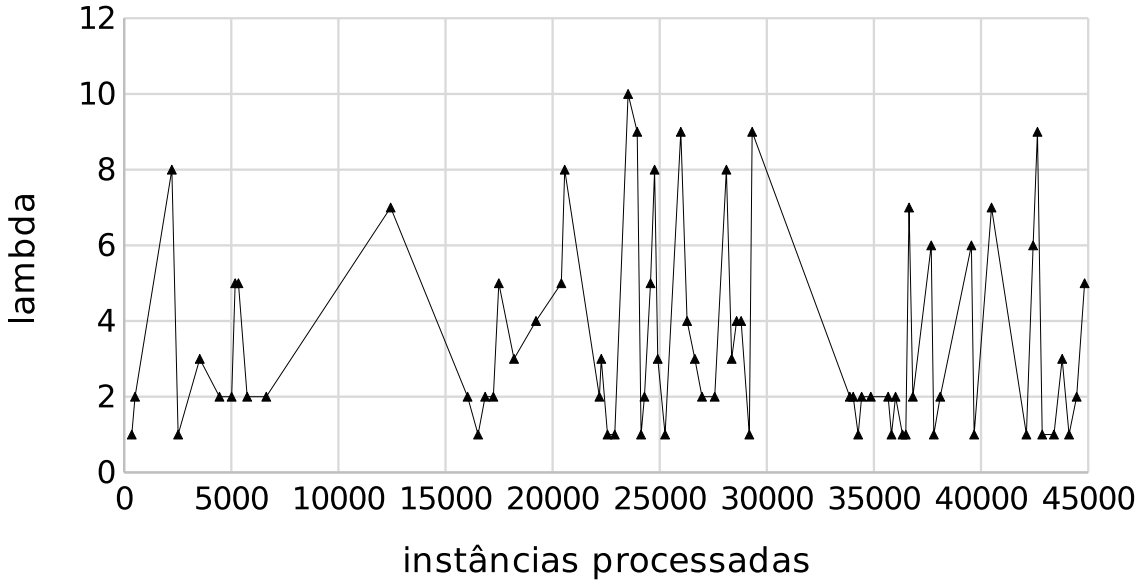
Fonte: O autor (2022)

Figura 17 – Ilustração da trajetória *lambda* retornada pela variação de BOLE-PE no processo de aprendizagem com os conjuntos de dados Covertype Sorted e Electricity.

(a) BOLE-PE CovertypeSorted



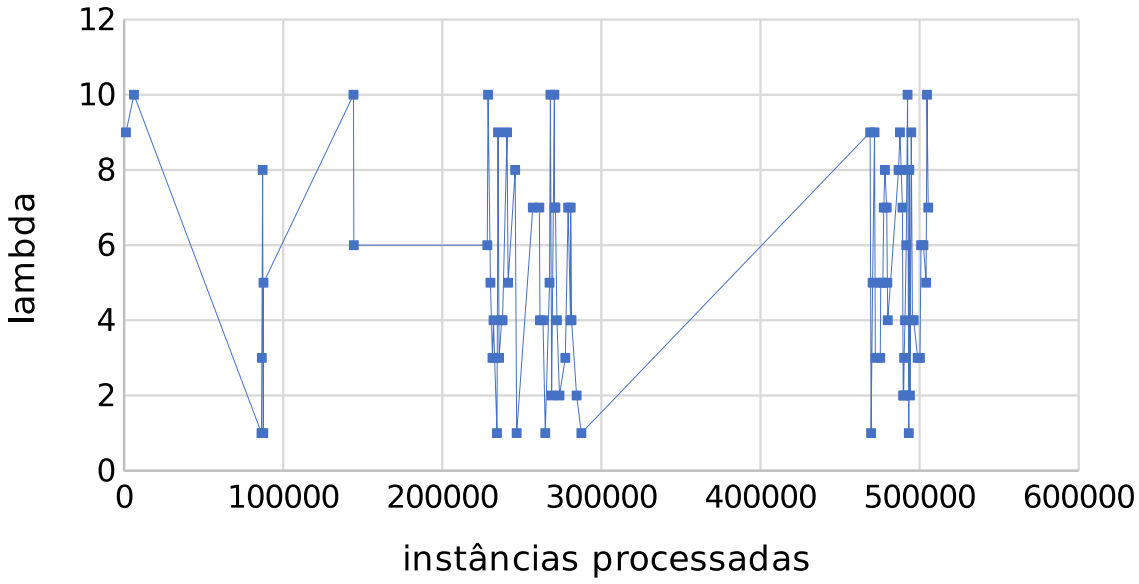
(b) BOLE-PE Electricity



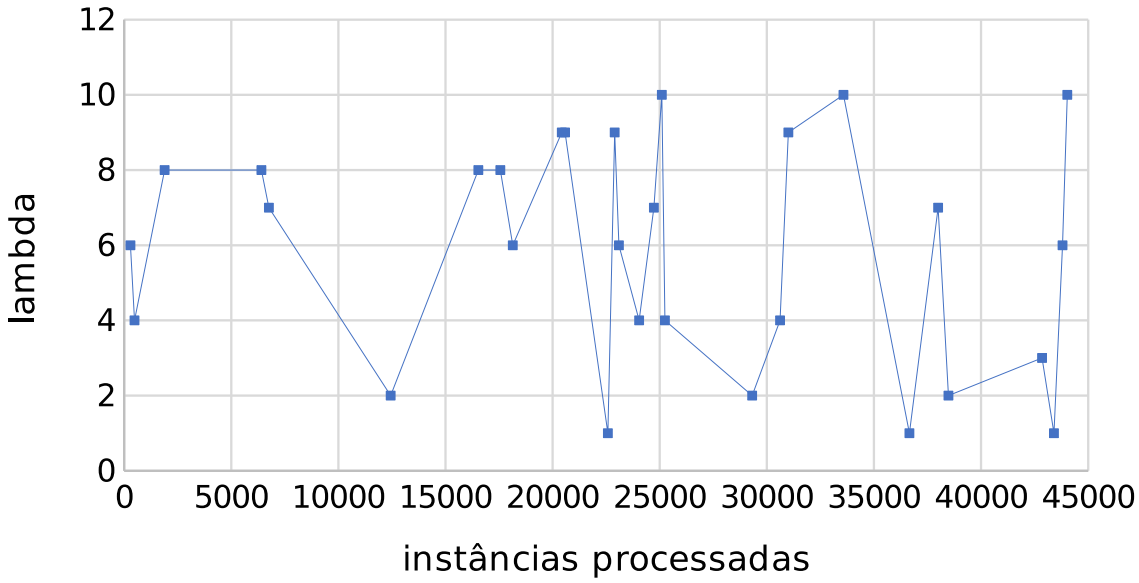
Fonte: O autor (2022)

Figura 18 – Ilustração da trajetória *lambda* retornada pela variação de OABM1-PE no processo de aprendizagem com os conjuntos de dados Covertype Sorted e Electricity.

(a) OABM1-PE CovertypeSorted



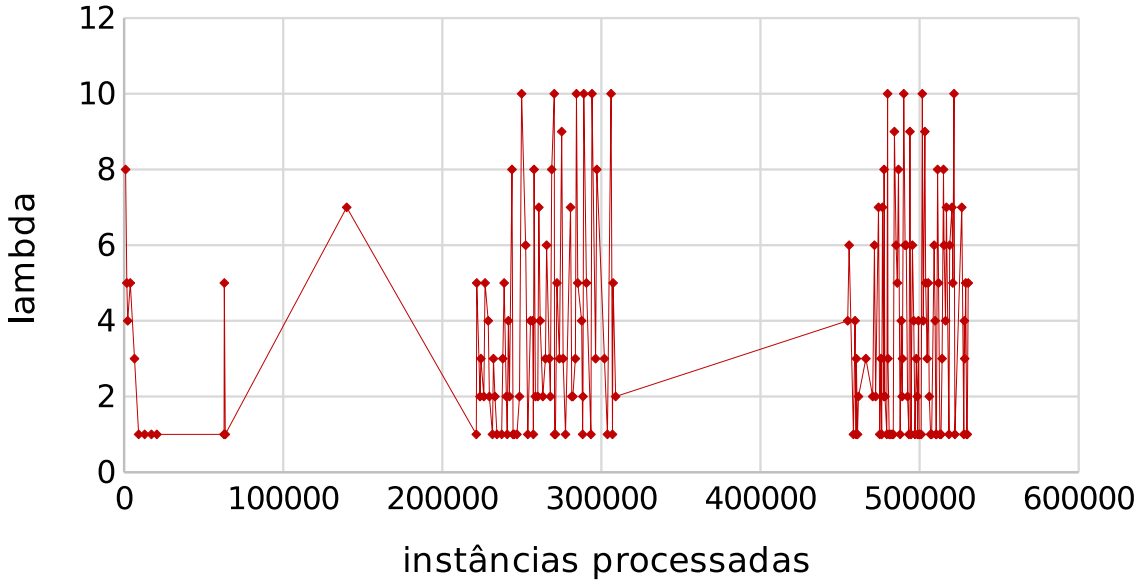
(b) OABM1-PE Electricity



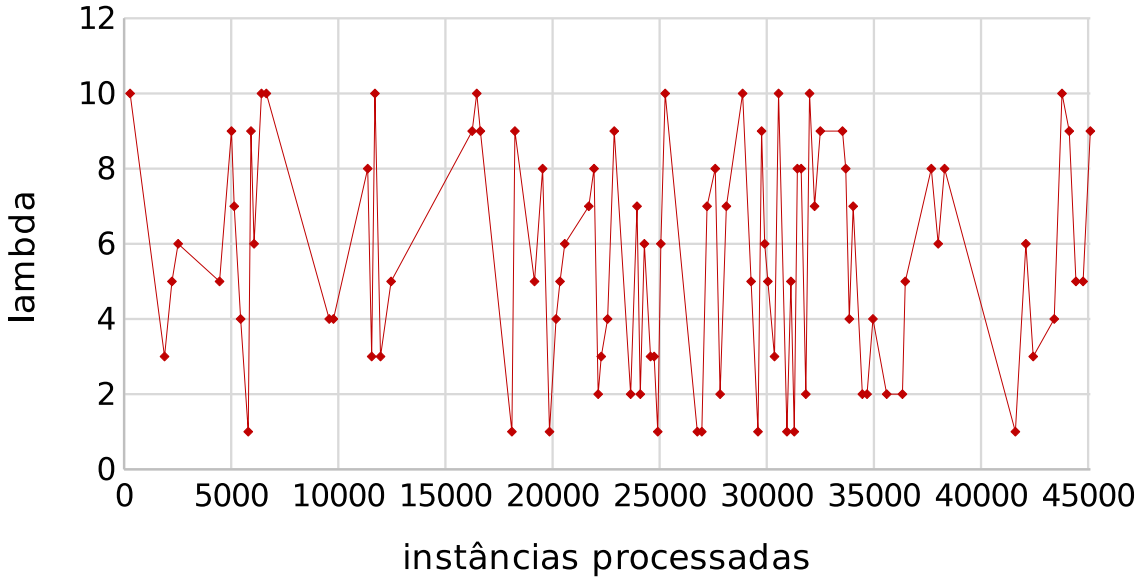
Fonte: O autor (2022)

Figura 19 – Ilustração da trajetória *lambda* retornada pela variação de OzaBag-PE no processo de aprendizagem com os conjuntos de dados Covertype Sorted e Electricity.

(a) OzaBag-PE CovertypeSorted



(b) OzaBag-PE Electricity



Fonte: O autor (2022)

6.1 CONSIDERAÇÕES FINAIS

Como já foi evidenciado anteriormente, neste capítulo foi apresentada a análise dos resultados experimentais relacionada com a aplicação do método PEP em comitês de classificadores. Nesse sentido, foram avaliadas as versões BOLE-PE, OABM1-PE e OzaBag-PE, abordagens resultantes da aplicação do PEP nos comitês BOLE, OABM1 e OzaBag. Estas versões modificadas incorporam a característica de ajustar dinamicamente o parâmetro λ desses comitês.

Os testes foram realizados com 40 bases artificiais e 9 bases do mundo real com os detectores internos RDDM e HDDM_A nos métodos comparados. Também foi usado o classificador base HT.

Estes resultados experimentais demonstraram que as abordagens BOLE-PE, OABM1 e OzaBag tiveram resultados vantajosos na maioria dos cenários testados. Em específico, nos testes apresentando os valores das médias de acurácia e as diferenças nos desempenhos dos métodos modificados em relação aos originais nos conjuntos de dados artificiais, BOLE-PE obteve o melhor desempenho quando comparado com BOLE-λ6 e BOLE. Já OABM1-PE em relação a OABM1-λ-6 e OABM1, foi o segundo melhor método na maioria dos conjuntos de dados, despontando com os melhores resultados nos geradores RBF, SEA e $Waveform$. Nos testes de OzaBag e versões, o OzaBag-PE apresentou um desempenho mais proporcional.

Além disso, nas avaliações de dados do mundo real, as três versões apresentaram os melhores desempenhos, sendo superiores em relação ao resto dos métodos na maioria dos testes com este tipo de conjunto de dados.

Por outro lado, os resultados das avaliações do teste de $Friedman$ mostraram que estas versões também foram as que obtiveram a melhor performance, seguidas pelas versões com λ igual a 6, e as versões originais. Adicionalmente, nas avaliações dos pós teste $Nemenyi$ que identifica se existe diferenças significativas entre os enfoques comparados, os resultados também foram semelhantes. Aqui as versões modificadas foram as mais bem ranqueadas na maioria das ilustrações apresentadas sendo estatisticamente diferentes do que os seus respectivos métodos originais.

Uma vez, realizada a avaliação experimental aos métodos propostos (BOLE-PE, OABM1-PE e OzaBag-PE) neste capítulo, após ter observado o desempenho deles nos cenários experimentados, e olhando a possibilidade de usá-los em pesquisas da área em cenários de mudanças de conceitos abruptas e graduais com conjuntos de dados artificiais, a sugestão dos métodos a usar são OzaBag-PE e BOLE-PE. Nas avaliações com bases reais, o OABM1-PE é o melhor

a ser usado. Já nos cenários com conjuntos de dados artificiais e reais, o OzaBag-PE é o mais recomendado.

Por último, nos gráficos que mostraram a trajetória dos valores de $lambda$ e, por conseguinte o funcionamento desta estratégia PEP aplicada a estas versões no longo do fluxo de dados, pode se apreciar o quanto benéfico é este ajuste dinâmico da diversidade aplicado aos comitês de classificadores testados e a importância no impacto dos resultados finais da avaliação dos enfoques.

7 CONCLUSÕES

Nesta tese de doutorado foram propostas novas estratégias de ajuste dinâmico de parâmetros que foram aplicadas a diversos métodos de classificação existentes na área de aprendizado de máquina, visando com esse propósito, melhorar o desempenho dos enfoques no que se refere à métrica de acurácia na avaliação experimental de fluxo de dados onde acontecem mudanças de conceitos nas distribuições. Nesse sentido, foram aplicados vários procedimentos com o propósito de melhorar a performance dos atuais trabalhos relacionados.

Também foi realizado um estudo compreensivo das singularidades referentes à classificação em fluxo de dados online e denotadas as características dos preditores Hoeffding Tree (HT) e K-Nearest Neighbors (k-NN). Este último usado como método base nesta pesquisa para implementar estratégias de ajuste de parâmetros no enfoque. Também foi apresentada uma revisão conceitual do desempenho da detecção de mudanças de conceito e, por conseguinte, foram caracterizados os recentes trabalhos da área (detectores e comitês de classificadores) que foram utilizados nesta tese de doutorado.

Depois foram fornecidos os métodos introduzidos neste trabalho para realizar ajustes dinâmicos nos parâmetros de vários algoritmos de classificação introduzidos na área. Nesse sentido, primeiramente foram implementados os métodos pareados PL-kNN, PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$. Estas abordagens utilizam diferentes procedimentos para ajustar de forma dinâmica e incremental o número de vizinhos k, parâmetro pertencente aos classificadores k-NN dos enfoques pareados. Seguidamente, foram descritas a implementações da técnica genérica que ajusta parâmetros dinamicamente PEP e as versões BOLE-PE, OABM1-PE, e OzaBag-PE, respectivamente. Essas versões representam o resultado da aplicação de PEP neles para ajustar dinamicamente o parâmetro da diversidade λ destes comitês de classificadores baseados nos enfoques existentes BOLE, OABM1 e OzaBag-PE nessa ordem.

O relato dos diferentes aspectos utilizados na proposta para realizar as execuções dos experimentos foi apresentado no capítulo 4, envolvendo as novas abordagens, utilizando diferentes estratégias de experimentação. Os testes foram realizados no MOA e os procedimentos de experimentação em termos de acurácia de forma geral, foram testados junto com os classificadores k-NN e HT. De igual forma, foram usados os detectores de mudanças de conceitos HDDM$_\text{A}$, RDDM e FTDD como auxiliares nos testes dos métodos. Para realizar os experimentos de todas as propostas introduzidas neste trabalho, foram selecionados sete geradores

de bases de dados artificiais e construídas diferentes configurações de mudanças de conceitos abruptas e graduais. Também foram escolhidos treze conjuntos de dados do mundo real. Por conseguinte, foram descritas as considerações referentes à configuração de parametrização dos métodos introduzidos nesta pesquisa.

Os resultados experimentais relacionados com os algoritmos de classificação pareados mostraram que, com base na acurácia global, o PL-kNN apresentou forte desempenho na grande maioria dos conjuntos de dados. As versões PL-kNN$_2$, PL-kNN$_3$ e PL-kNN$_4$ foram igualmente eficientes, entregando resultados próximos dos melhores enfoques. Nesse sentido foram atingidos os objetivos propostos, PL-kNN e suas demais versões tiveram um desempenho satisfatório sendo os mais eficazes ou próximos das melhores configurações testadas do k-NN em quase todos os cenários.

Em termos estatísticos, os resultados do teste de *Friedman* e do pós-teste *Nemenyi* confirmaram a superioridade global de PL-kNN. Mais especificamente, ao considerar todos os conjuntos de dados testados, foi estatisticamente melhor do que todas as três configurações testadas de k-NN, independentemente do detector de mudanças de conceito utilizado. As versões PL-kNN$_3$, PL-kNN$_4$ e PL-kNN$_2$ foram os métodos que obtiveram os restantes melhores resultados nessa ordem depois do PL-kNN.

Por outro lado, os testes resultantes das avaliações experimentais em termos de acurácia da aplicação do método PEP em comitês de classificadores mostraram que o ajuste dinâmico de λ usando PEP foi benéfico na maioria dos cenários, ou seja, os métodos modificados apresentaram desempenhos superiores quando comparados às abordagens originais sem ajuste dinâmico. Essas melhorias foram observadas em cenários artificiais e do mundo real, com pequenas diferenças na eficácia entre os conjuntos de base.

Mais especificamente, de acordo com os resultados do teste *Friedman* em combinação com o pós-teste *Nemenyi*, BOLE-PE, OABM1-PE e OzaBag-PE alcançaram os melhores ranks nas comparações separadas por comitê base, e as três versões foram significativamente melhores do que suas respectivas contrapartes originais ao considerar todo o conjunto de bases de dados testadas.

Complementando essas avaliações estatísticas, também foram comparados todos os métodos juntos por tipo de conjunto de dados. OzaBag-PE foi o comitê com melhor desempenho em conjuntos de dados artificiais, sendo estatisticamente melhor que OABM1-PE. Nos conjuntos de dados do mundo real, OABM1-PE foi a primeira abordagem ranqueada e não houve diferenças significativas entre os três conjuntos modificados. Já em todos os conjuntos de dados,

OzaBag-PE foi novamente o melhor e, neste cenário o método também foi estatisticamente melhor que OABM1-PE.

7.1 PUBLICAÇÕES

Os argumentos apresentados nesta tese de doutorado relacionados com o algoritmo de classificação pareado Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN) fizeram parte da escrita do artigo, que foi recentemente aceito para publicação no periódico *Knowledge and Information Systems*.

- **Juan I. G. Hidalgo, Silas G. T. C. Santos, Roberto S. M. Barros**. "*Paired k-NN Learners with Dynamically Adjusted Number of Neighbors for Classification of Drifting Data Streams*".

Além disso, os fundamentos presentes nesta pesquisa referentes à técnica genérica Parameter Estimation Procedure (PEP) e as versões resultantes da aplicação deste método (Boosting-like Online Learning Ensemble with Parameter Estimation (BOLE-PE), Online AdaBoost-based M1 with Parameter Estimation (OABM1-PE) e Oza and Russell's Online Bagging with Parameter Estimation (OzaBag-PE)), complementaram a escrita do artigo publicado no periódico *ACM Transactions on Knowledge Discovery from Data*.

- **Juan I. G. Hidalgo, Silas G. T. C. Santos, Roberto S. M. Barros**. "*Dynamically Adjusting Diversity in Ensembles for the Classification of Data Streams with Concept Drift*". In: ACM: *Trans. Knowl. Discov. Data*. 16, 2, Article 31 (July 2021) p. 20.

Por outro lado, no inicio do meu doutorado, foi publicado no periódico *Neurocomputing* o artigo relacionado com o detector de mudanças de conceitos Wilcoxon Rank Sum Test Drift Detector (WSTD), trabalho no qual participei da escrita e que forma parte dos estudos durante a minha formação no curso de mestrado.

- **Roberto S. M. Barros, Juan I. G. Hidalgo, Danilo R. L. Cabral**. "*Wilcoxon Rank Sum Test Drift Detector*". *Neurocomputing*. v.275. p.1954-1963. 2018.

Da mesma forma, durante este período foi publicado no periódico *Computational Intelligence*, o artigo relacionado com a pesquisa da minha defesa de dissertação de mestrado.

- **Juan I. G. Hidalgo, Bruno I. F. Maciel, Roberto S. M. Barros**. "*Experimenting with prequential variations for data stream learning evaluation*". *Computational Intelligence*, Wiley Online Library, 2019.

Além dos trabalhos submetidos e publicados em periódicos, também foram publicados varios trabalhos de conferências. Durante o meu doutorado foi implementado o detector Cosine Similarity Drift Detector (CSDD). As particularidades deste método de detecção de mudanças de conceitos geraram a escrita de um artigo publicado na *28th International Conference on Artificial Neural Networks* (ICANN19), realizado de 17 a 19 de setembro de 2019 na Universidade Técnica de Munique, Alemanha.

- **Juan I. G. Hidalgo, Laura M. P. Mariño, Roberto S. M. Barros**. "*Cosine Similarity Drift Detector*". In: SPRINGER: *International Conference on Artificial Neural Networks*. 2019. p. 669-685.

Do mesmo modo, nesse decorrer da minha pesquisa também participei da escrita do artigo referente aos comitês *Fast Stacking of Ensembles boosting the Best* ($FASEB$) e *Fast Ensemble boosting the Best with Combined Weighting Voting* ($FASE_{wv}$), adaptações do comitê de classificador *FASE*, que foi publicado no *International Joint Conference on Neural Networks (IJCNN) 2019*.

- **Laura M. P. Mariño, Juan I. G. Hidalgo, Roberto S. M. Barros, Germano C. Vasconcelos**. "*Improving Fast Adaptive Stacking of Ensembles*".In: *International Joint Conference on Neural Networks (IJCNN)*. 2019. Budapest.

Por último, também participei da escrita do artigo referente ao detector Ultimately Simple Drift Detector (USDD), que foi publicado no *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, realizado entre os dias 17 e 20 de outubro de 2021 em Melbourne, Australia.

- **Bruno I. F. Maciel, Juan I. G. Hidalgo, Roberto S. M. Barros**. "*An Ultimately Simple Concept Drift Detector for Data Streams*". In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2021. pp. 625-630. Melbourne.

7.2 PERSPECTIVAS PARA TRABALHOS FUTUROS

Em sentido geral, as avaliações realizadas nesta tese de doutorado com as estratégias para ajustar parâmetros de forma dinâmica nos métodos de classificação escolhidos demostraram que a aplicação destes procedimentos foi benéfica para melhorar o desempenho experimental dos métodos na maioria dos cenários onde foram testados no que se refere à métrica de acurácia. Mas nesta área de pesquisa, ainda existem muitos desafios.

Em relação ao método Paired k-NN Learners with Dynamically Adjusted Number of Neighbors (PL-kNN) e as versões propostas para ajustar dinamicamente o número de vizinhos k presentes nos membros k-NN dos algoritmos pareados, algumas direções podem ser consideradas como trabalhos futuros:

- Experimentar a estratégia proposta usando outros classificadores e/ou parâmetros básicos.
- Experimentar a potencialidade de um comitê com mais membros, explorando diferentes estratégias de manipulação do parâmetro ajustado.
- Testar outras funções de agregação para realizar a classificação do comitê, por exemplo, média ponderada de acurácia.
- Estudar a viabilidade de ajustar dinamicamente mais de um parâmetro ao mesmo tempo.

Por outro lado, no que se refere às avaliações realizadas com a técnica Parameter Estimation Procedure (PEP) aplicada nos comitês de classificadores escolhidos, algumas direções como trabalho futuro podem ser seguidas:

- Ampliar a análise das versões propostas utilizando outras métricas de avaliação como tempo de execução e consumo de memória.
- Verificar o impacto que um aumento ou diminuição do tamanho da amostra utilizada para fazer o ajuste dos parâmetros causaria no desempenho geral dos comitês.
- Incluir um período de estabilidade dos métodos antes que a amostra seja utilizada para escolher o melhor valor de parâmetro, aumentando a probabilidade de que essa escolha seja feita com um nível de convergência satisfatório no conceito atual.

- Testar PEP com outros métodos e usando outros parâmetros, pois é uma abordagem genérica que pode ser adaptada sem grandes dificuldades.

REFERÊNCIAS

AGGARWAL, C. C. *Data classification: algorithms and applications*. [S.l.]: CRC Press, 2014.

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. N. Database mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, v. 5, n. 6, p. 914–925, 1993.

AHA, D. W. *Lazy learning*. [S.l.]: Springer Science & Business Media, 2013.

ALMEIDA, P. R.; OLIVEIRA, L. S.; JR, A. S. B.; SABOURIN, R. Adapting dynamic classifier selection for concept drift. *Expert Systems with Applications*, Elsevier, v. 104, p. 67–85, 2018.

ATKESON, C. G.; MOORE, A. W.; SCHAAL, S. Locally weighted learning. *Artificial Intelligence Review*, Springer, v. 11, n. 1-5, p. 11–73, 1997.

ATTAR, V.; CHAUDHARY, P.; RAHAGUDE, S.; CHAUDHARI, G.; SINHA, P. An instance-window based classification algorithm for handling gradual concept drifts. In: CAO, L.; BAZZAN, A. L. C.; SYMEONIDIS, A. L.; GORODETSKY, V. I.; WEISS, G.; YU, P. S. (Ed.). *Agents and Data Mining Interaction: 7th International Workshop on Agents and Data Mining Interation, ADMI 2011, Taipei, Taiwan, May 2-6, 2011, Revised Selected Papers*. Berlin, Heidelberg: Springer, 2012. p. 156–172. ISBN 978-3-642-27609-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-27609-5_11>.

BAHRI, M.; VELOSO, B.; BIFET, A.; GAMA, J. Automl for stream k-nearest neighbors classification. In: IEEE. *2020 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2020. p. 597–602.

BARDDAL, J. P.; GOMES, H. M.; GRANATYR, J.; BRITTO, A. de S.; ENEMBRECK, F. Overcoming feature drifts via dynamic feature weighted k-nearest neighbor learning. In: *Proceedings of 23rd IEEE International Conference on Pattern Recognition (ICPR)*. [S.l.: s.n.], 2016. p. 2186–2191.

BARROS, R. S.; CABRAL, D. R.; GONÇALVES JR., P. M.; SANTOS, S. G. RDDM: Reactive drift detection method. *Expert Systems with Applications*, Elsevier, v. 90, p. 344–355, 2017.

BARROS, R. S.; SANTOS, S. G.; GONÇALVES JR., P. M. A boosting-like online learning ensemble. In: IEEE. *2016 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2016. p. 1871–1878.

BARROS, R. S. M.; SANTOS, S. G. T. C. A large-scale comparison of concept drift detectors. *Information Sciences*, Elsevier, v. 451-452, n. C, p. 348–370, 2018. Doi: 10.1016/j.ins.2018.04.014.

BARROS, R. S. M.; SANTOS, S. G. T. C. An overview and comprehensive comparison of ensembles for concept drift. *Information Fusion*, Elsevier, v. 52, n. C, p. 213–244, 2019.

BARROS, R. S. M.; SANTOS, S. G. T. C.; BARDDAL, J. P. Evaluating k-nn in the classification of data streams with concept drift. *arXiv preprint arXiv:2210.03119*, 2022.

BIFET, A.; GAVALDÀ, R. Learning from time-changing data with adaptive windowing. In: *Proceedings of 7th SIAM International Conference on Data Mining (SDM'07)*. Minneapolis, MN, USA: [s.n.], 2007. p. 443–448.

BIFET, A.; GAVALDÀ, R.; HOLMES, G.; PFAHRINGER, B. *Machine Learning for Data Streams with Practical Examples in MOA*. [S.l.]: MIT Press, 2018. <https://moa.cms.waikato.ac.nz/book/>.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. MOA: Massive online analysis. *Journal of Machine Learning Research*, MIT Press, v. 11, p. 1601–1604, 2010.

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: *Machine Learning and Knowledge Discovery in Databases*. [S.l.]: Springer, 2010, (LNCS, v. 6321). p. 135–150.

BIFET, A.; HOLMES, G.; PFAHRINGER, B.; KIRKBY, R.; GAVALDÀ, R. New ensemble methods for evolving data streams. In: *Proceedings of 15th ACM International Conference on Knowledge Discovery and Data Mining (KDD'09)*. Paris, France: [s.n.], 2009. p. 139–148.

BIFET, A.; KIRKBY, R. *Data stream mining a practical approach*. [S.l.]: Citeseer, 2009.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006.

BOTTOU, L.; VAPNIK, V. Local learning algorithms. *Neural Computation*, MIT Press, Cambridge, MA, USA, v. 4, n. 6, p. 888–900, nov. 1992. ISSN 0899-7667.

BREIMAN, L. Bagging predictors. *Machine Learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

BRZEZIŃSKI, D. *Mining data streams with concept drift*. Dissertação (Mestrado) — Master's thesis, Poznan University of Technology, 2010.

CABRAL, D. R. L.; BARROS, R. S. M. Concept drift detection based on Fisher's Exact test. *Information Sciences*, Elsevier, v. 442, p. 220–234, 2018.

CAI, L.; YU, Y.; ZHANG, S.; SONG, Y.; XIONG, Z.; ZHOU, T. A sample-rebalanced outlier-rejected k-nearest neighbor regression model for short-term traffic flow forecasting. *IEEE access*, IEEE, v. 8, p. 22686–22696, 2020.

CAI, Y.-l.; JI, D.; CAI, D. A knn research paper classification method based on shared nearest neighbor. In: *Proceedings of NTCIR-8 Workshop Meeting*. Tokyo, Japan: [s.n.], 2010. p. 336–340.

CANDILLIER, L.; LEMAIRE, V. Design and analysis of the nomao challenge active learning in the real-world. In: *Proceedings of the ALRA: Active Learning in Real-world Applications, Workshop ECML-PKDD*. [S.l.: s.n.], 2012. p. 1–15.

CARNEIN, M.; TRAUTMANN, H.; BIFET, A.; PFAHRINGER, B. Towards automated configuration of stream clustering algorithms. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Cham: Springer, 2019. p. 137–143.

COETZEE, P.; JAARSVELD, F. V.; VANHAECKE, F. Intraregional classification of wine via icp-ms elemental fingerprinting. *Food Chemistry*, Elsevier, v. 164, p. 485–492, 2014. Doi: 10.1016/j.foodchem.2014.05.027.

COPPOLINO, L.; D'ANTONIO, S.; FORMICOLA, V.; ROMANO, L. Integration of a system for critical infrastructure protection with the ossim siem platform: A dam case study. In: SPRINGER. *International Conference on Computer Safety, Reliability, and Security*. [S.l.], 2011. p. 199–212.

CORTEZ, P.; CERDEIRA, A.; ALMEIDA, F.; MATOS, T.; REIS, J. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, v. 47, n. 4, p. 547–553, 2009.

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)*, JSTOR, p. 278–292, 1984.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, JMLR.org, v. 7, p. 1–30, 2006.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: *Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, USA: ACM, 2000. (KDD '00), p. 71–80.

DONG, T.; CHENG, W.; SHANG, W. The research of knn text categorization algorithm based on eager learning. In: IEEE. *2012 International Conference on Industrial Control and Electronics Engineering*. [S.l.], 2012. p. 1120–1123.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

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

ESCOVEDO, T.; KOSHIYAMA, A. *Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise*. [S.l.]: Casa do Código, 2020.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. Inteligência artificial: Uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, v. 2, p. 192, 2011.

FERN, X.; BRODLEY, C. *Cluster ensembles for high dimensional clustering: An empirical study*. [S.l.], 2004. Disponível em: <http://hdl.handle.net/1957/35655>.

FISHER, R. *Statistical Methods for Research Workers*. London, England: Oliver and Boyd, 1934. (Biological Monographs and Manuals).

FONTENLA-ROMERO, Ó.; GUIJARRO-BERDIÑAS, B.; MARTINEZ-REGO, D.; PÉREZ-SÁNCHEZ, B.; PETEIRO-BARRAL, D. Online machine learning. In: *Efficiency and Scalability Methods for Computational Intellect*. [S.l.]: IGI Global, 2013. p. 27–54.

FRANK, A. UCI machine learning repository. *http://archive.ics.uci.edu/ml*, University of California, School of Information and Computer Science, 2010.

FREUND, Y.; SCHAPIRE, R. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, Elsevier, Orlando, FL, USA, v. 55, n. 1, p. 119–139, 1997.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: *International Conference on Machine Learning*. [S.l.: s.n.], 1996. v. 96, p. 148–156.

FRÍAS-BLANCO, I.; CAMPO-ÁVILA, J. del; RAMOS-JIMÉNEZ, G.; MORALES-BUENO, R.; ORTIZ-DÍAZ, A.; CABALLERO-MOTA, Y. 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.

FRÍAS-BLANCO, I.; VERDECIA-CABRERA, A.; ORTIZ-DÍAZ, A.; CARVALHO, A. Fast adaptive stacking of ensembles. In: ACM. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. [S.l.], 2016. p. 929–934.

GABER, M. M.; ZASLAVSKY, A.; KRISHNASWAMY, S. *A Survey of Classification Methods in Data Streams*. Boston, MA: Springer, 2007. 39–59 p. ISBN 978-0-387-47534-9. Disponível em: <https://doi.org/10.1007/978-0-387-47534-9_3>.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: *Advances in Artificial Intelligence: SBIA 2004*. [S.l.]: Springer, 2004, (LNCS, v. 3171). p. 286–295.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: SPRINGER. *Brazilian Symposium on Artificial Intelligence*. [S.l.], 2004. p. 286–295.

GOMES, H. M.; BARDDAL, J. P.; ENEMBRECK, F.; BIFET, A. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, ACM, v. 50, n. 2, p. 23, 2017.

GONÇALVES JR., P. M.; BARROS, R. S. M. RCD: A recurring concept drift framework. *Pattern Recognition Letters*, Elsevier, v. 34, n. 9, p. 1018–1025, 2013.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011.

HERMANOWSKI, D. Open source security information management system supporting it security audit. In: IEEE. *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*. [S.l.], 2015. p. 336–341.

HIDALGO, J. I. G.; MACIEL, B. I. F.; BARROS, R. S. M. Experimenting with prequential variations for data stream learning evaluation. *Computational Intelligence*, Wiley, v. 35, p. 670–692, 2019.

HOEFFDING, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, v. 58, p. 13–30, 1963.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, USA: [s.n.], 2001. (KDD '01), p. 97–106.

IENCO, D.; BIFET, A.; ŽLIOBAITĖ, I.; PFAHRINGER, B. Clustering based active learning for evolving data streams. In: SPRINGER. *International Conference on Discovery Science*. [S.l.], 2013. p. 79–93.

JOHNSON, O. *Information Theory and the Central Limit Theorem*. [S.l.]: Imperial College Press, London, 2004.

KATAKIS, I.; TSOUMAKAS, G.; VLAHAVAS, I. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, Springer, v. 22, p. 371–391, 2010.

KOURTELLIS, N.; MORALES, G. D. F.; BIFET, A.; MURDOPO, A. VHT: Vertical Hoeffding tree. In: *2016 IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2016. p. 915–922.

KOYCHEV, I. Experiments with two approaches for tracking drifting concepts. *Serdica Journal of Computing*, v. 1, n. 1, p. 27–44, 2007.

KRAMER, O. K-nearest neighbor. In: _____. *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Berlin, Heidelberg: Springer, 2013. p. 13–23. ISBN 978-3-642-38652-7.

KUMAR, A.; KAUR, P.; SHARMA, P. A survey on Hoeffding tree stream data classification algorithms. *CPUH-Research Journal*, v. 1, n. 2, p. 28–32, 2015.

LARGE, J.; LINES, J.; BAGNALL, A. The heterogeneous ensembles of standard classification algorithms (HESCA): the whole is greater than the sum of its parts. *arXiv preprint arXiv:1710.09220*, 2017.

LIAO, Y.; VEMURI, V. Use of k-nearest neighbor classifier for intrusion detection. *Computers & Security*, Elsevier Advanced Technology Publications, GBR, v. 21, n. 5, p. 439–448, October 2002. ISSN 0167-4048.

LIU, A.; LU, J.; LIU, F.; ZHANG, G. Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognition*, Elsevier, v. 76, p. 256–272, 2018.

LOSING, V.; HAMMER, B.; WERSING, H. KNN classifier with self adjusting memory for heterogeneous concept drift. In: *IEEE 16th International Conference on Data Mining (ICDM)*. [S.l.: s.n.], 2016. p. 291–300. Doi: 10.1109/ICDM.2016.0040.

LOSING, V.; HAMMER, B.; WERSING, H. Tackling heterogeneous concept drift with the self-adjusting memory (sam). *Knowledge and Information Systems*, Springer, v. 54, n. 1, p. 171–201, 2018.

LU, N.; LU, J.; ZHANG, G.; MANTARAS, R. Lopez de. A concept drift-tolerant case-base editing technique. *Artificial Intelligence*, Elsevier, GBR, v. 230, n. C, p. 108–133, jan. 2016. ISSN 0004-3702.

LU, N.; ZHANG, G.; LU, J. Concept drift detection via competence models. *Artificial Intelligence*, v. 209, p. 11–28, 2014. ISSN 0004-3702.

MACIEL, B. I. F.; SANTOS, S. G. T. C.; BARROS, R. S. M. A lightweight concept drift detection ensemble. In: *Proceedings of 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'15)*. Vietri sul Mare, Italy: [s.n.], 2015. p. 1061–1068.

MACIEL, B. I. F.; SANTOS, S. G. T. C.; BARROS, R. S. M. MOAManager: a tool to support data stream experiments. *Software: Practice and Experience*, Wiley, v. 50, n. 4, p. 325–334, 2020.

MAHDI, O. A.; PARDEDE, E.; ALI, N.; CAO, J. Diversity measure as a new drift detection method in data streaming. *Knowledge-Based Systems*, Elsevier, v. 191, p. 12, 2020.

MANAPRAGADA, C.; WEBB, G. I.; SALEHI, M. Extremely fast decision tree. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*. New York, NY, USA: [s.n.], 2018. (KDD '18), p. 1953–1962. ISBN 9781450355520. Disponível em: <https://doi.org/10.1145/3219819.3220005>.

MARON, O.; MOORE, A. Hoeffding races: Accelerating model selection search for classification and function approximation. In: COWAN, J. D.; TESAURO, G.; ALSPECTOR, J. (Ed.). *Advances in Neural Information Processing Systems 6*. San Mateo, California: Elsevier, 1994.

MARRÓN, D.; READ, J.; BIFET, A.; ABDESSALEM, T.; AYGUADÉ, E.; HERRERO, J. R. Echo state Hoeffding tree learning. In: *Proceedings of The 8th Asian Conference on Machine Learning*. [S.l.: s.n.], 2016. p. 382–397.

MCDIARMID, C. On the method of bounded differences. *Surveys in combinatorics*, v. 141, n. 1, p. 148–188, 1989.

MINKU, L. L.; WHITE, A. P.; YAO, X. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, v. 22, n. 5, p. 730–742, 2010.

MITCHELL, T. *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.

MUALLEM, A.; SHETTY, S.; PAN, J. W.; ZHAO, J.; BISWAL, B. Hoeffding tree algorithms for anomaly detection in streaming datasets: A survey. *Journal of Information Security*, v. 8, n. 4, 2017.

NEMENYI, P. B. Distribution-free multiple comparisons. *Princeton University*, 1963.

NGUYEN, T.; CZERWINSKI, M.; LEE, D. Compaq quicksource: Providing the consumer with the power of artificial intelligence. In: *Proceedings of the The Fifth Conference on Innovative Applications of Artificial Intelligence*. [S.l.]: AAAI Press, 1993. (IAAI '93), p. 142–151. ISBN 0929280466.

NGUYEN, T. T.; ARMITAGE, G. A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, IEEE, v. 10, n. 4, p. 56–76, 2008.

NISHIDA, K.; YAMAUCHI, K. Detecting concept drift using statistical testing. In: *Proceedings of 10th International Conference on Discovery Science (DS'07)*. [S.l.]: Springer, 2007. (LNCS, v. 4755), p. 264–269.

OZA, N. C.; RUSSELL, S. Online bagging and boosting. In: *Artificial Intelligence and Statistics*. [S.l.]: Morgan Kaufmann, 2001. p. 105–112.

PEDROSA, J. *Pensamentos edificantes*. [S.l.]: Clube de Autores, 2015.

PHAM, X. C.; DANG, M. T.; DINH, S. V.; HOANG, S.; NGUYEN, T. T.; LIEW, A. W.-C. Learning from data stream based on random projection and Hoeffding tree classifier. In: IEEE. *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. [S.l.], 2017. p. 1–8.

RIJN, J. N. van; HOLMES, G.; PFAHRINGER, B.; VANSCHOREN, J. The online performance estimation framework: heterogeneous ensemble learning for data streams. *Machine Learning*, Springer, v. 107, n. 1, p. 149–176, 2018.

SALGANICOFF, M. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, Kluwer Academic Publishers, USA, v. 11, n. 1–5, p. 133–155, fev. 1997.

SANTOS, S. G. T. C. *Online boosting para problemas multiclasse*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2019. Disponível em: <https://repositorio.ufpe.br/handle/123456789/34465>.

SANTOS, S. G. T. C.; BARROS, R. S. M. Online adaboost-based methods for multiclass problems. *Artificial Intelligence Review*, Springer, p. 1–30, 2019.

SANTOS, S. G. T. C.; BARROS, R. S. M.; GONÇALVES JR., P. M. Optimizing the parameters of drift detection methods using a genetic algorithm. In: *Proceedings of 27th IEEE Internatational Conference on Tools with Artificial Intelligence (ICTAI'15)*. Vietri sul Mare, Italy: [s.n.], 2015. p. 1077–1084.

SANTOS, S. G. T. C.; GONÇALVES JR., P. M.; SILVA, G.; BARROS, R. S. M. Speeding up recovery from concept drifts. In: *Machine Learning and Knowledge Discovery in Databases*. [S.l.]: Springer, 2014, (LNCS, v. 8726). p. 179–194.

SENI, G.; ELDER, J. F. Ensemble methods in data mining: improving accuracy through combining predictions. *Synthesis lectures on data mining and knowledge discovery*, Morgan & Claypool Publishers, v. 2, n. 1, p. 1–126, 2010.

SEWELL, M. Ensemble learning. *RN*, v. 11, n. 02, 2008.

SHALEV-SHWARTZ, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, Now Publishers, Inc., v. 4, n. 2, p. 107–194, 2012.

SHAN, J.; ZHANG, H.; LIU, W.; LIU, Q. Online active learning ensemble framework for drifted data streams. *IEEE transactions on neural networks and learning systems*, v. 30, n. 2, p. 486–498, 2018.

SIMOUDIS, E.; AHA, D. W. Special issue on lazy learning. *Artificial Intelligence Review*, Kluwer Academic Publishers, v. 11, n. 1—5, p. 7–10, 1997. ISSN 0269–2821.

SIVARAJAH, U.; KAMAL, M. M.; IRANI, Z.; WEERAKKODY, V. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, Elsevier, v. 70, p. 263–286, 2017.

SOLOMATINE, D. P.; MASKEY, M.; SHRESTHA, D. L. Eager and lazy learning methods in the context of hydrologic forecasting. In: IEEE. *IEEE International Joint Conference on Neural Network Proceedings*. [S.l.], 2006. p. 4847–4853.

SONG, G.; YE, Y.; ZHANG, H.; XU, X.; LAU, R. Y.; LIU, F. Dynamic clustering forest: an ensemble framework to efficiently classify textual data stream with concept drift. *Information Sciences*, Elsevier, v. 357, p. 125–143, 2016.

SONG, X.; HE, H.; NIU, S.; GAO, J. A data streams analysis strategy based on Hoeffding tree with concept drift on hadoop system. In: *2016 International Conference on Advanced Cloud and Big Data (CBD)*. [S.l.: s.n.], 2016. p. 45–48.

SOUSA, Á. Diagrama de dispersão, correlação e regressão linear. *Correio dos Açores*, Gráfica Açoreana, Lda, p. 16–16, 2019.

SRIVAS, P. G. K. S. Performance evaluation of MOA v/s KNN classification schemes: Case study of major cities in the world. *International Journal of Computer Sciences and Engineering*, IJCSE, Indore, INDIA, v. 7, p. 489–495, 4 2019. ISSN 2347-2693.

STREET, W. N.; KIM, Y. A streaming ensemble algorithm (SEA) for large-scale classification. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2001. (KDD '01), p. 377–382.

SUN, Y. *Data mining techniques*. 2014. Disponível em: <http://web.cs.ucla.edu/~yzsun/classes/2014Fall_CS6220/Slides/04Matrix_Data_Classification_1.pdf>.

WANG, H.; FAN, W.; YU, P. S.; HAN, J. Mining concept-drifting data streams using ensemble classifiers. In: ACM. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2003. p. 226–235.

WEI, C.-C. Comparing lazy and eager learning models for water level forecasting in river-reservoir basins of inundation regions. *Environmental Modelling & Software*, Elsevier, v. 63, p. 137–155, 2015.

WIDMER, G.; KUBAT, M. Effective learning in dynamic environments by explicit context tracking. In: *Machine learning: ECML-93*. [S.l.]: Springer, 1993. p. 227–243.

WU, X.; LI, P.; HU, X. Learning from concept drifting data streams with unlabeled data. *Neurocomputing*, Elsevier, v. 92, p. 145–155, 2012.

XIOUFIS, E. S.; SPILIOPOULOU, M.; TSOUMAKAS, G.; VLAHAVAS, I. Dealing with concept drift and class imbalance in multi-label stream classification. In: *Proceedings of 22nd International Joint Conference on Artificial Intelligence*. Barcelona, Spain: [s.n.], 2011. (IJCAI'11), p. 1583–1588.

YATES, F. Contingency tables involving small numbers and the χ 2 test. *Supplement to the Journal of the Royal Statistical Society*, JSTOR, v. 1, n. 2, p. 217–235, 1934.

ZHANG, M.-L.; ZHOU, Z.-H. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, v. 40, n. 7, p. 2038–2048, 2007. ISSN 0031-3203.

ZHANG, S.; LI, X.; ZONG, M.; ZHU, X.; CHENG, D. Learning k for kNN classification. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 8, n. 3, p. 1–19, 2017.

ZHANG, W.; NTOUTSI, E. FAHT: An adaptive fairness-aware decision tree classifier. *arXiv preprint arXiv:1907.07237*, 2019.

ZHENG, Z.; WEBB, G. I. Lazy learning of bayesian rules. *Machine Learning*, Springer, v. 41, n. 1, p. 53–84, 2000.

ZHOU, Z.-H. *Ensemble methods: foundations and algorithms*. [S.l.]: Chapman and Hall/CRC, 2012.

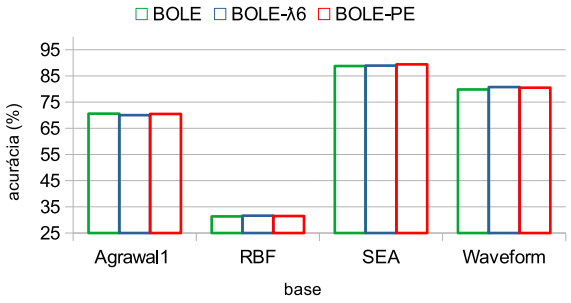
ZHU, X. *Stream Data Mining Repository*. 2010. Online. <http://www.cse.fau.edu/~xqzhu/stream.html>.

APÊNDICE A – RESULTADOS ADICIONAIS REFERENTES A APLICAÇÃO DO MÉTODO PEP EM COMITÊS DE CLASSIFICADORES

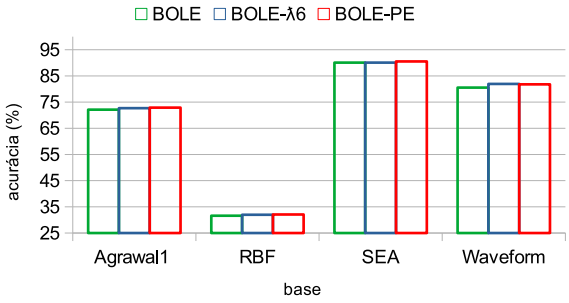
A.1 EXPERIMENTOS USANDO REACTIVE DRIFT DETECTION METHOD (RDDM) COMO DETECTOR AUXILIAR NOS MÉTODOS.

Figura 20 – Ilustração da média das acurácias de BOLE, BOLE-λ6 e BOLE-PE usando HT e RDDM como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

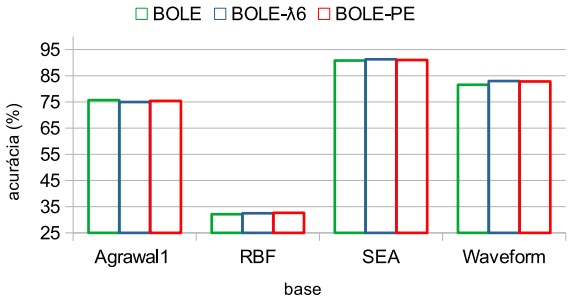
(a) 10.000 instâncias



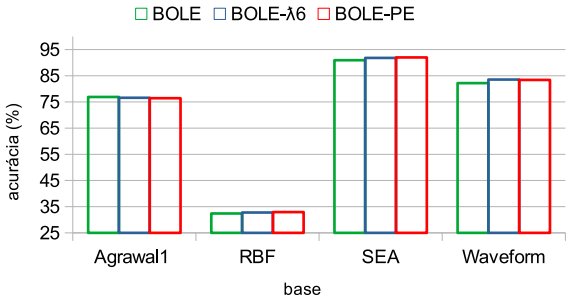
(b) 20.000 instâncias



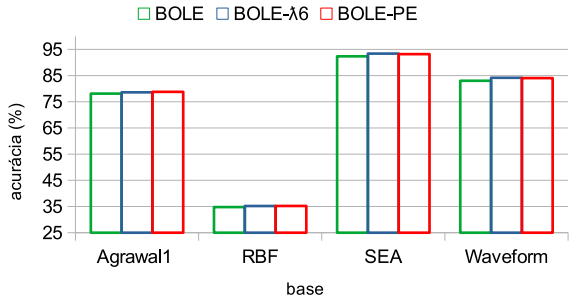
(c) 50.000 instâncias



(d) 100.000 instâncias



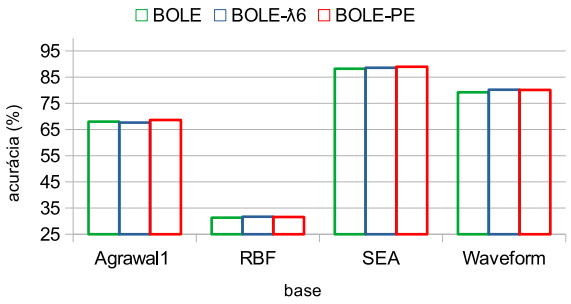
(e) 1.000.000 instâncias



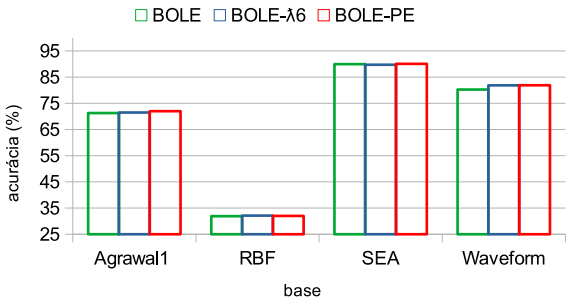
Fonte: O autor (2022)

Figura 21 – Ilustração da média das acurácias de BOLE, BOLE-λ6 e BOLE-PE usando HT e RDDM como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

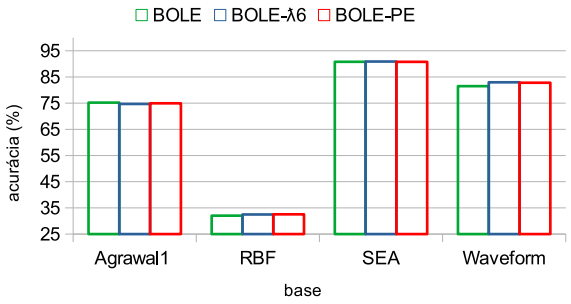
(a) 10.000 instâncias



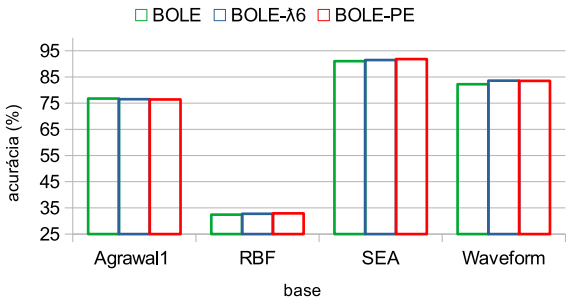
(b) 20.000 instâncias



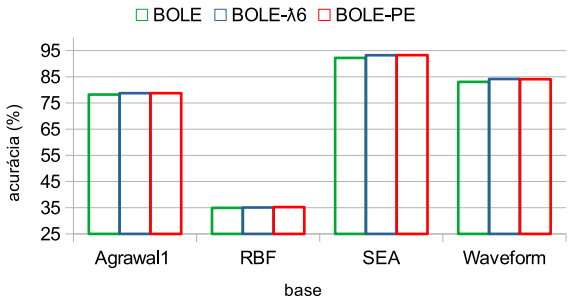
(c) 50.000 instâncias



(d) 100.000 instâncias



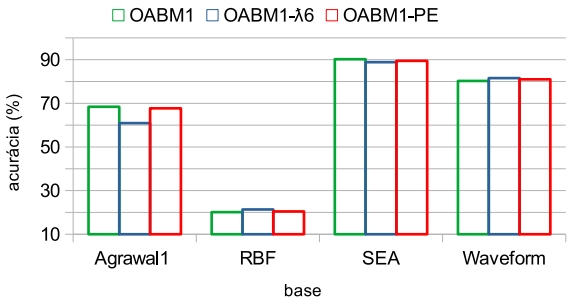
(e) 1.000.000 instâncias



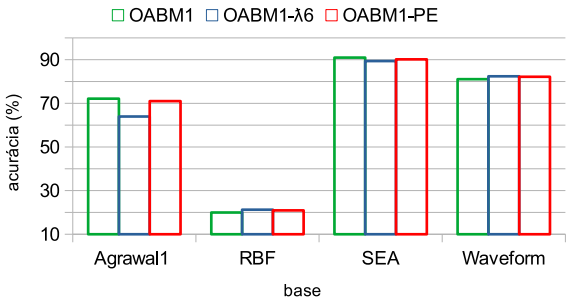
Fonte: O autor (2022)

Figura 22 – Ilustração da média das acurácias de OABM1, OABM1-λ6 e OABM1-PE usando HT e RDDM como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

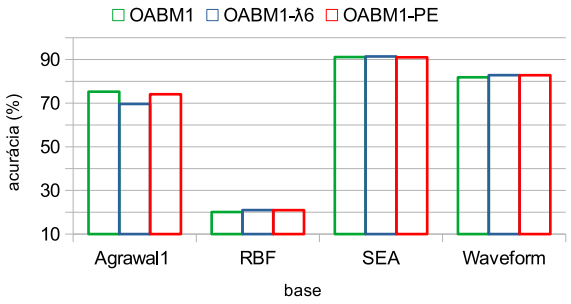
(a) 10.000 instâncias



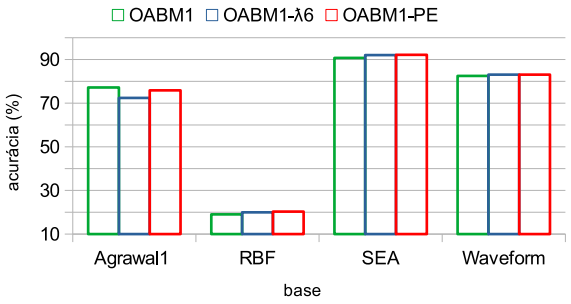
(b) 20.000 instâncias



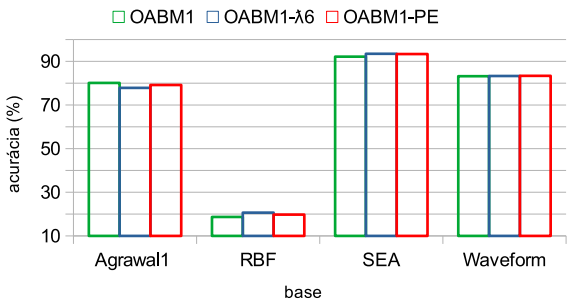
(c) 50.000 instâncias



(d) 100.000 instâncias



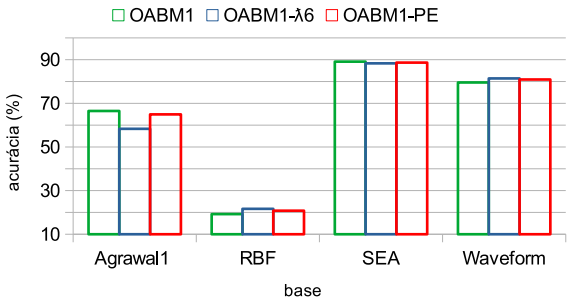
(e) 1.000.000 instâncias



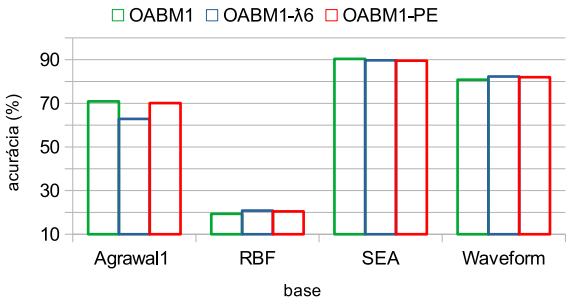
Fonte: O autor (2022)

Figura 23 – Ilustração da média das acurácias de OABM1, OABM1-λ6 e OABM1-PE usando HT e RDDM como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

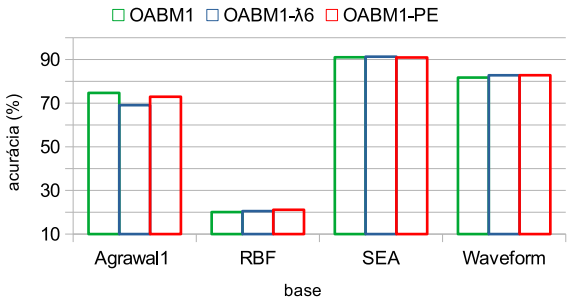
(a) 10.000 instâncias



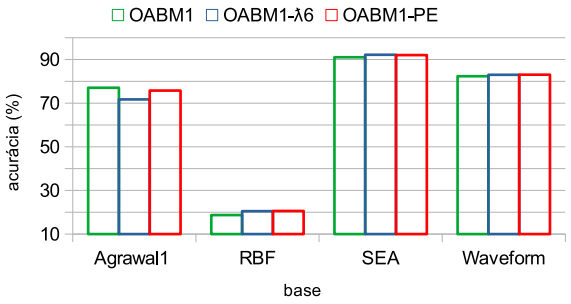
(b) 20.000 instâncias



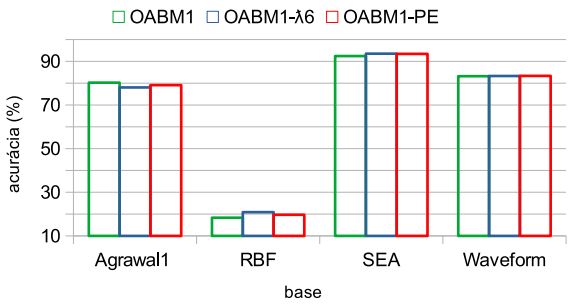
(c) 50.000 instâncias



(d) 100.000 instâncias



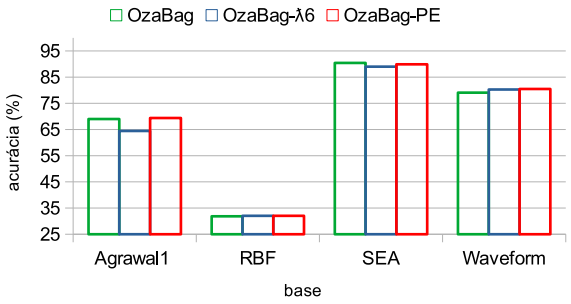
(e) 1.000.000 instâncias



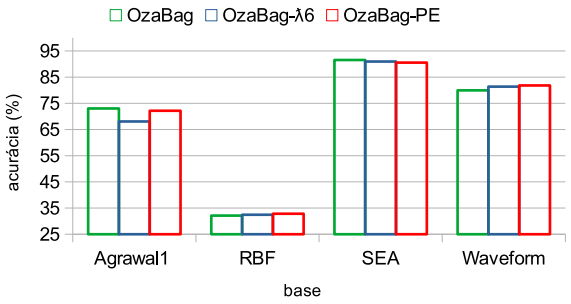
Fonte: O autor (2022)

Figura 24 – Ilustração da média das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE usando HT e RDDM como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

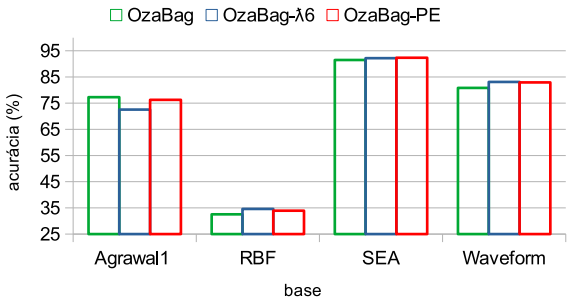
(a) 10.000 instâncias



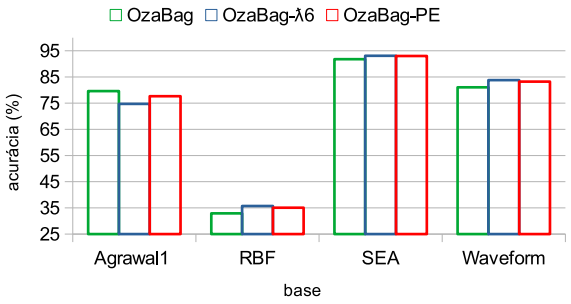
(b) 20.000 instâncias



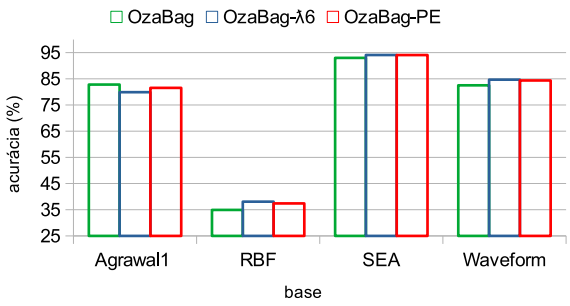
(c) 50.000 instâncias



(d) 100.000 instâncias



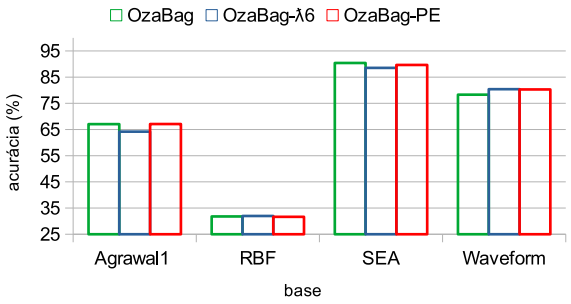
(e) 1.000.000 instâncias



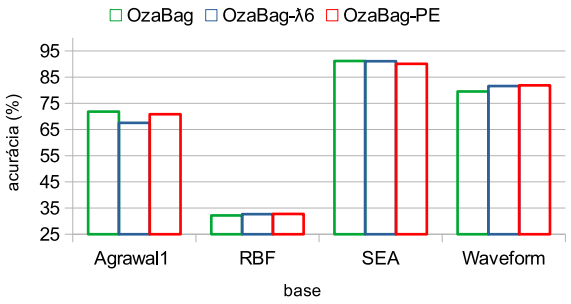
Fonte: O autor (2022)

Figura 25 – Ilustração da média das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE usando HT e RDDM como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

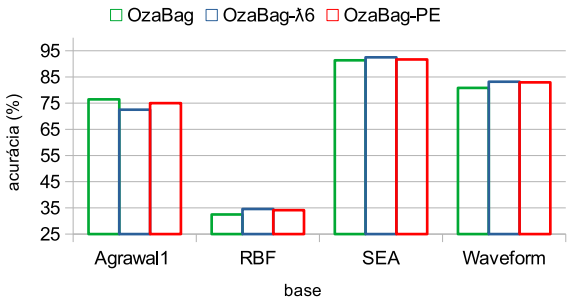
(a) 10.000 instâncias



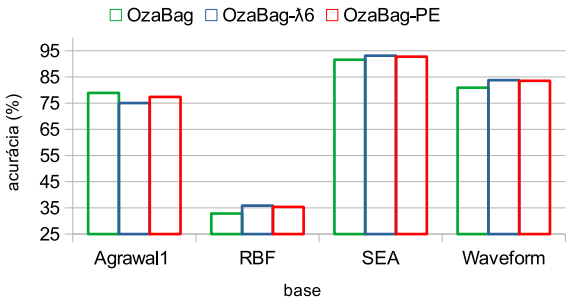
(b) 20.000 instâncias



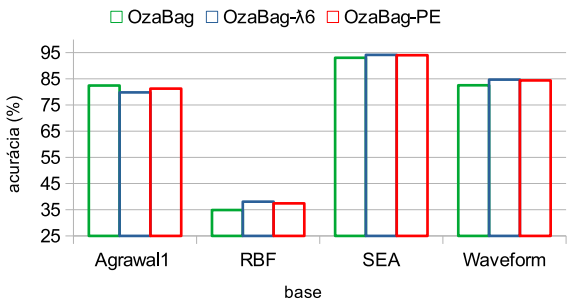
(c) 50.000 instâncias



(d) 100.000 instâncias



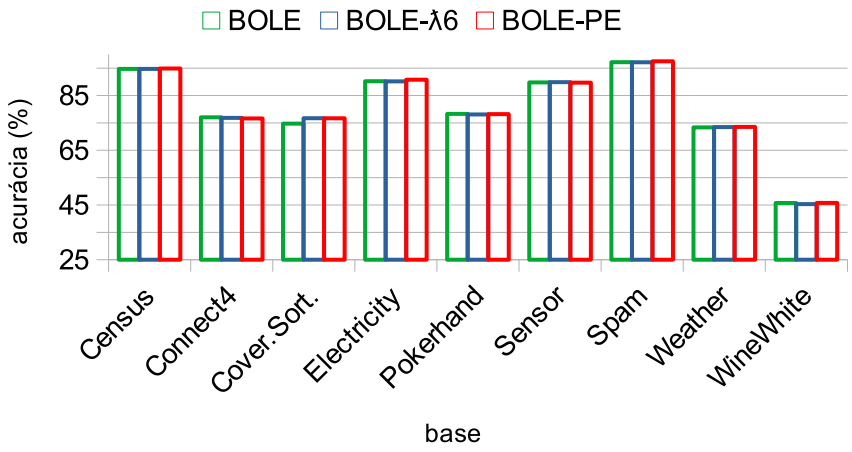
(e) 1.000.000 instâncias



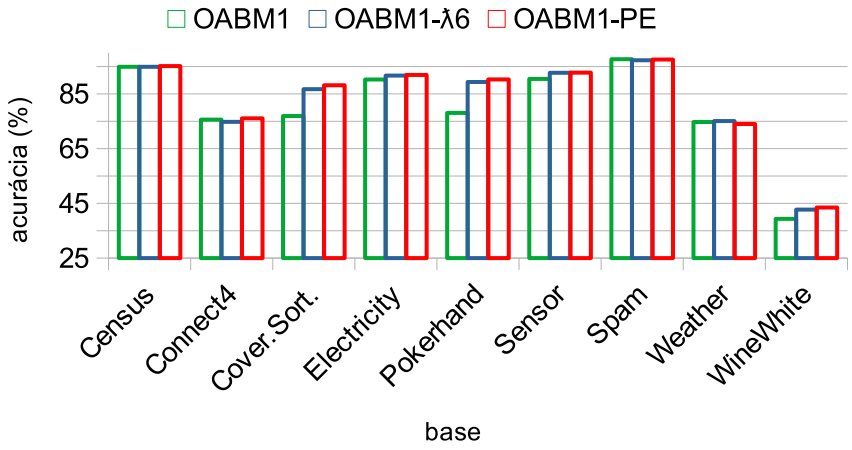
Fonte: O autor (2022)

Figura 26 – Ilustração da média das acurácias dos métodos usando HT e RDDM como detector auxiliar, com conjuntos de dados do mundo real.

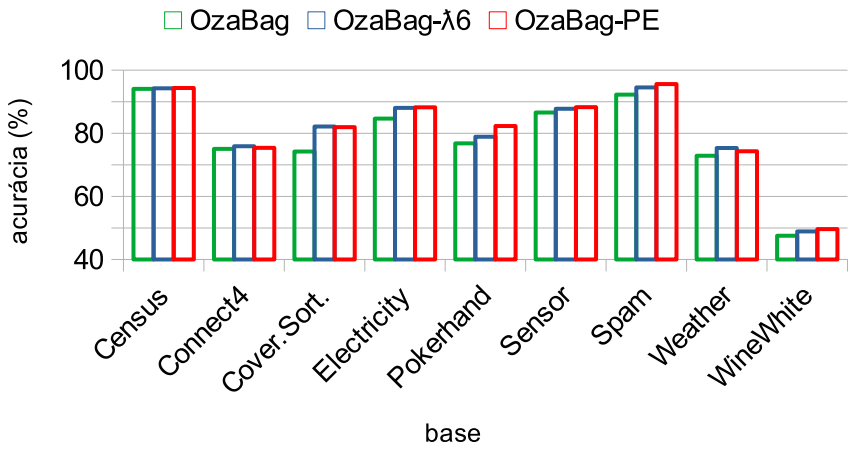
(a) BOLE e versões



(b) OABM1 e versões



(c) OzaBag e versões

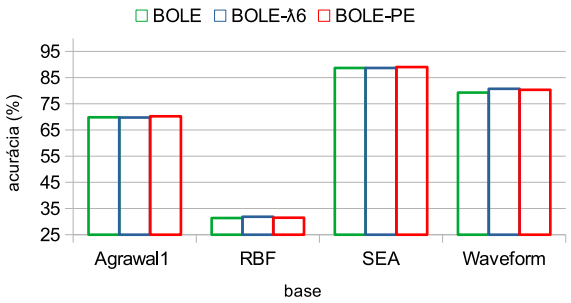


Fonte: O autor (2022)

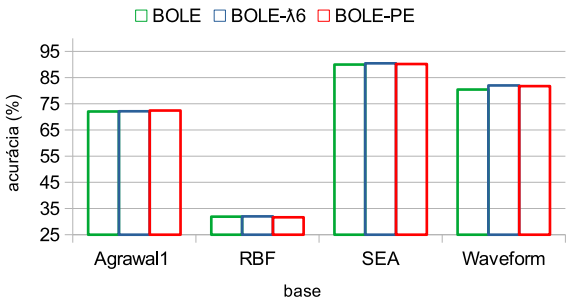
A.2 EXPERIMENTOS USANDO HOEFFDING-BASED DRIFT DETECTION METHOD A-TEST ($HDDM_A$) COMO DETECTOR AUXILIAR NOS MÉTODOS.

Figura 27 – Ilustração da média das acurácias de BOLE, BOLE-λ6 e BOLE-PE usando HT e $HDDM_A$ como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

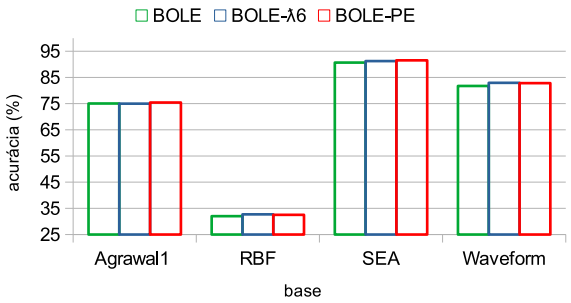
(a) 10.000 instâncias



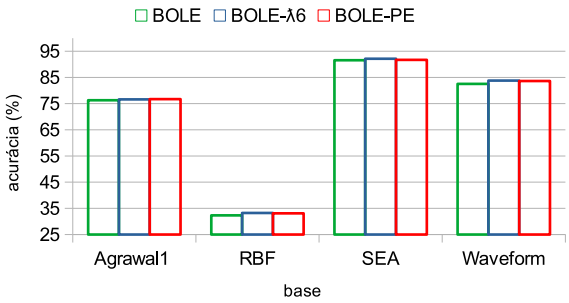
(b) 20.000 instâncias



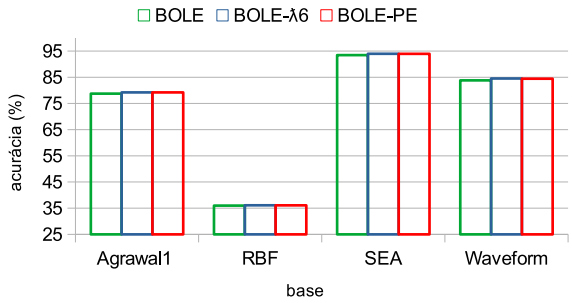
(c) 50.000 instâncias



(d) 100.000 instâncias



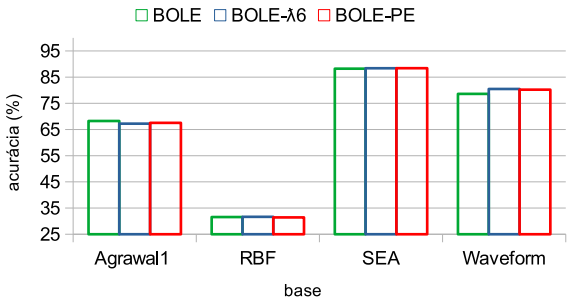
(e) 1.000.000 instâncias



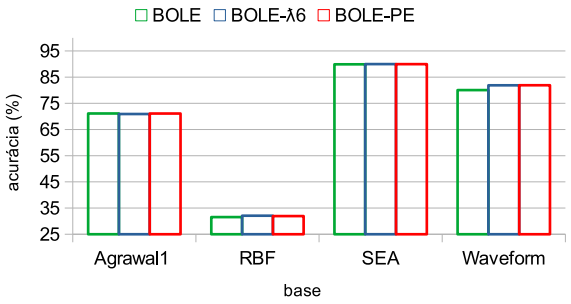
Fonte: O autor (2022)

Figura 28 – Ilustração da média das acurácias de BOLE, BOLE-λ6 e BOLE-PE usando HT e $HDDM_A$ como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

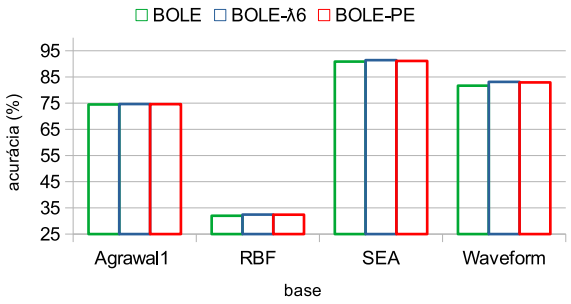
(a) 10.000 instâncias



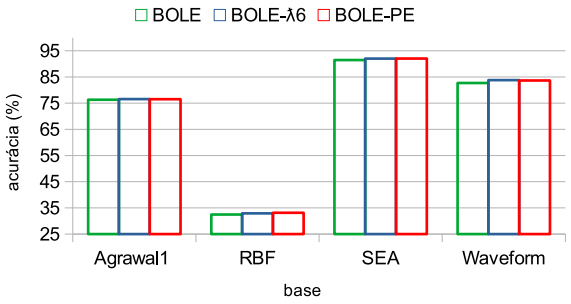
(b) 20.000 instâncias



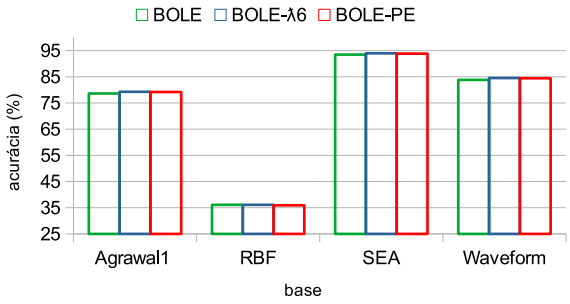
(c) 50.000 instâncias



(d) 100.000 instâncias



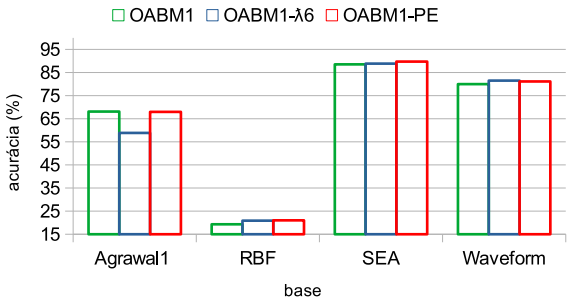
(e) 1.000.000 instâncias



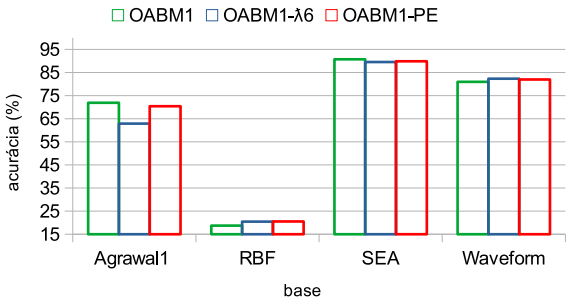
Fonte: O autor (2022)

Figura 29 – Ilustração da média das acurácias de OABM1, OABM1-λ6 e OABM1-PE usando HT e HDDM$_A$ como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

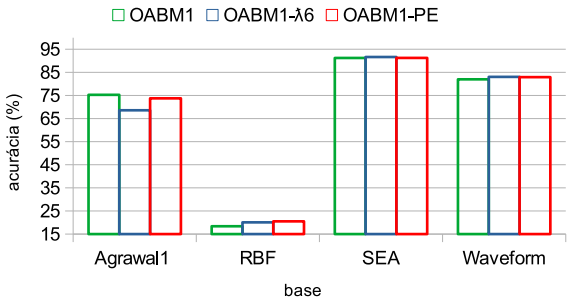
(a) 10.000 instâncias



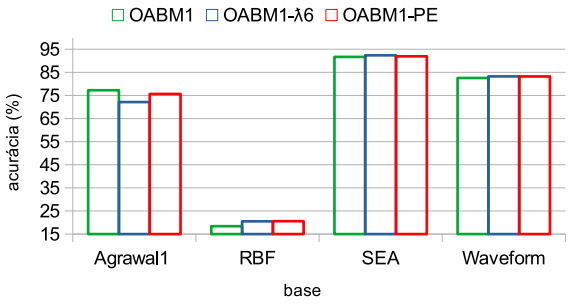
(b) 20.000 instâncias



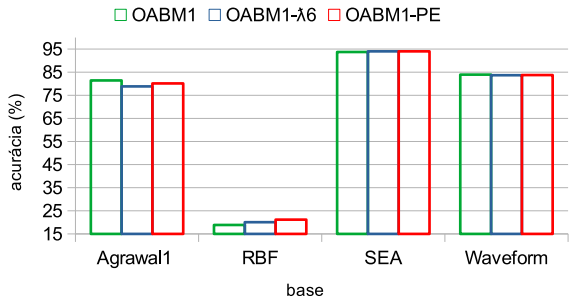
(c) 50.000 instâncias



(d) 100.000 instâncias



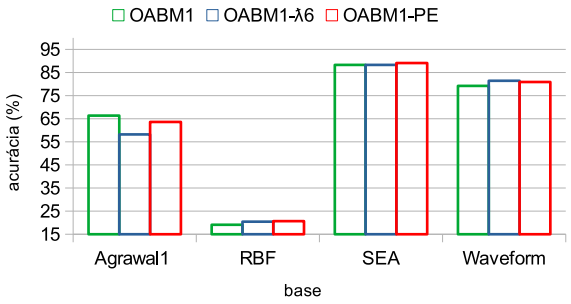
(e) 1.000.000 instâncias



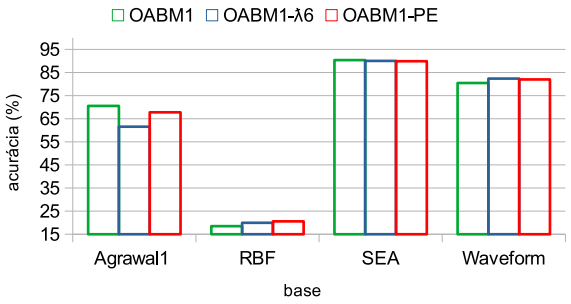
Fonte: O autor (2022)

Figura 30 – Ilustração da média das acurácias de OABM1, OABM1-λ6 e OABM1-PE usando HT e $HDDM_A$ como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

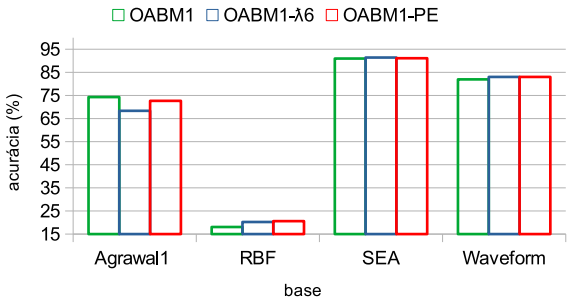
(a) 10.000 instâncias



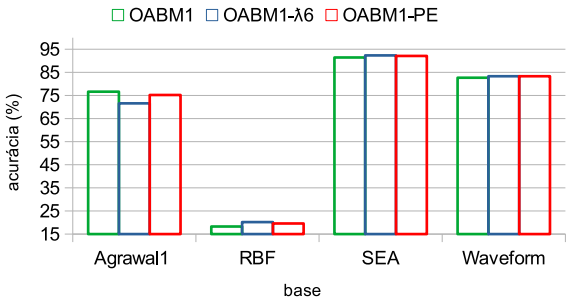
(b) 20.000 instâncias



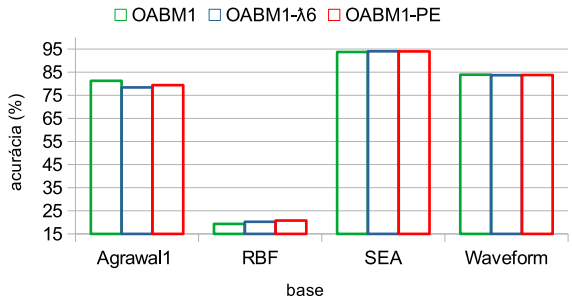
(c) 50.000 instâncias



(d) 100.000 instâncias



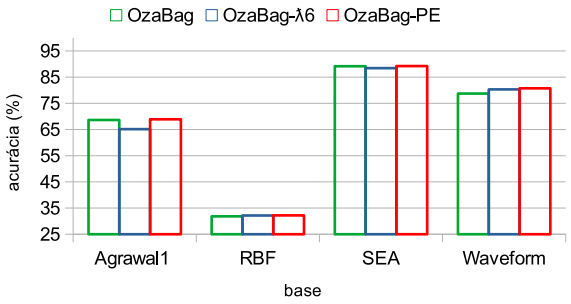
(e) 1.000.000 instâncias



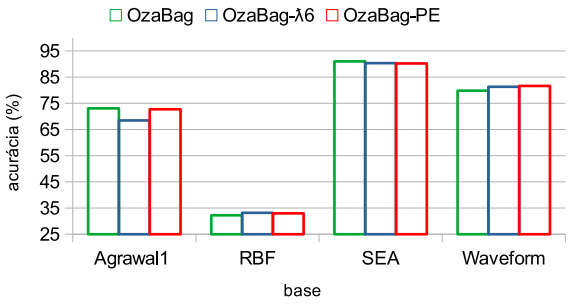
Fonte: O autor (2022)

Figura 31 – Ilustração da média das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE usando HT e $HDDM_A$ como detector auxiliar nos métodos, em cenários de mudanças abruptas com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

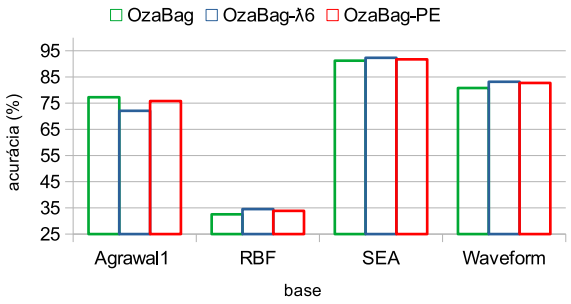
(a) 10.000 instâncias



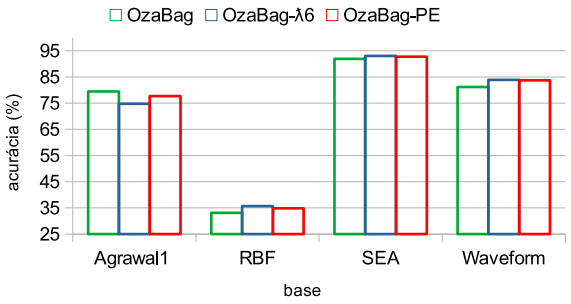
(b) 20.000 instâncias



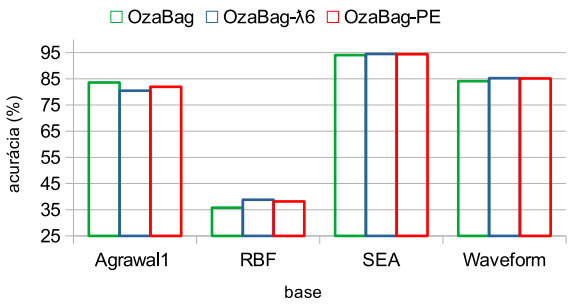
(c) 50.000 instâncias



(d) 100.000 instâncias



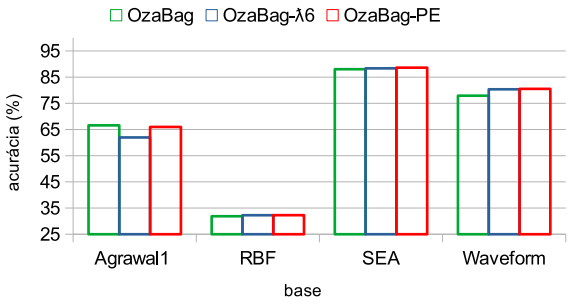
(e) 1.000.000 instâncias



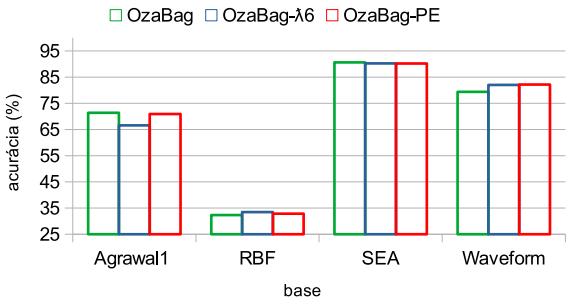
Fonte: O autor (2022)

Figura 32 – Ilustração da média das acurácias de OzaBag, OzaBag-λ6 e OzaBag-PE usando HT e $HDDM_A$ como detector auxiliar nos métodos, em cenários de mudanças graduais com conjuntos de dados artificiais construídos com 10.000, 20.000, 50.000, 100.000 e 1.000.000 instâncias.

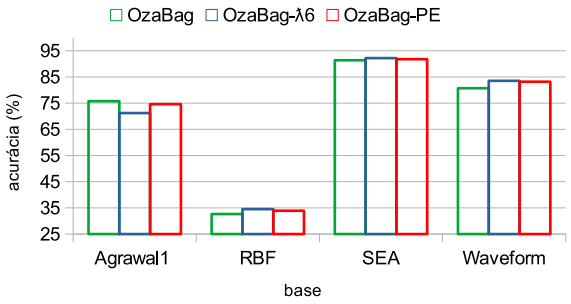
(a) 10.000 instâncias



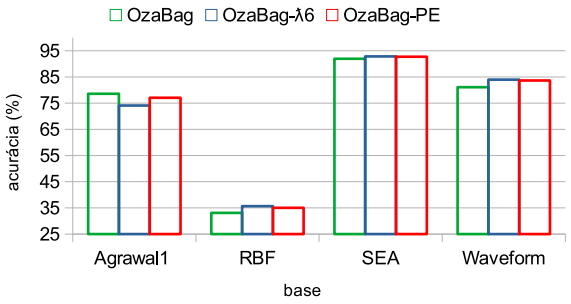
(b) 20.000 instâncias



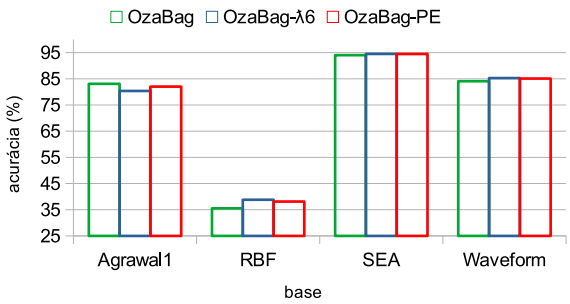
(c) 50.000 instâncias



(d) 100.000 instâncias



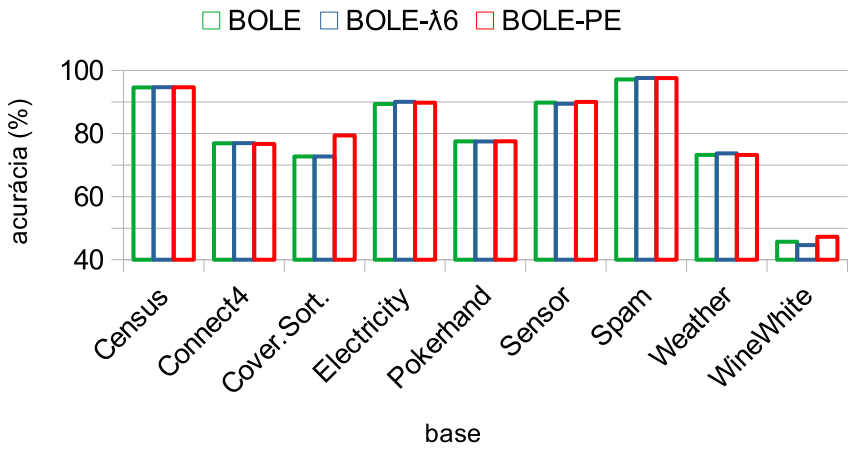
(e) 1.000.000 instâncias



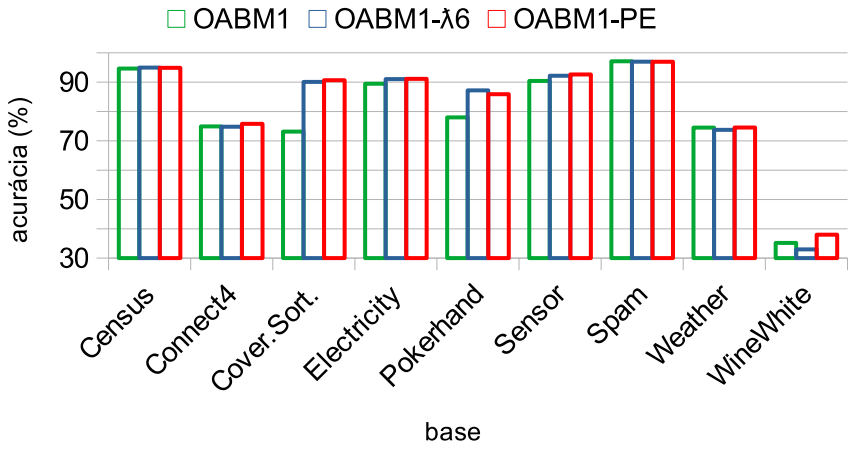
Fonte: O autor (2022)

Figura 33 – Ilustração da média das acurácias dos métodos usando HT e $HDDM_A$ como detector auxiliar, com conjuntos de dados do mundo real.

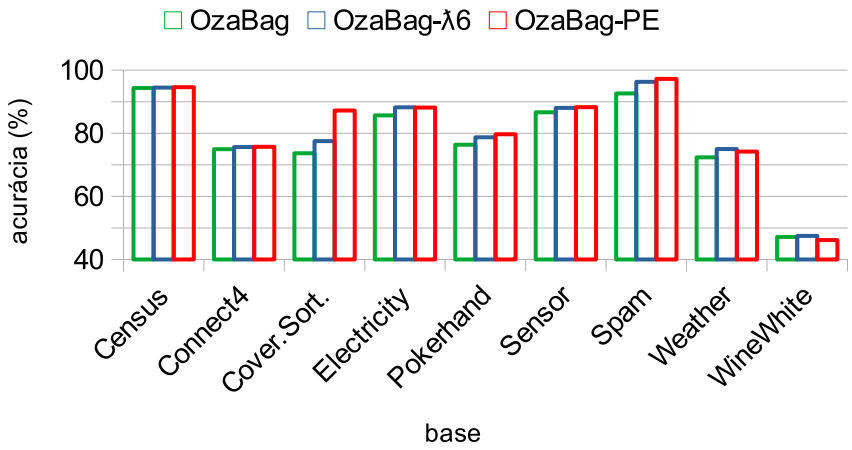
(a) BOLE e versões



(b) OABM1 e versões



(c) OzaBag e versões



Fonte: O autor (2022)