UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

DEIVIDY AMORIM POLICARPO

SSAFXGB: CLASSIFICAÇÃO SEMI-SUPERVISIONADA MULTI-CLASSE DE FLUXO DE DADOS COM A UTILIZAÇÃO DO XGBOOST

JOINVILLE

2024

DEIVIDY AMORIM POLICARPO

SSAFXGB: CLASSIFICAÇÃO SEMI-SUPERVISIONADA MULTI-CLASSE DE FLUXO DE DADOS COM A UTILIZAÇÃO DO XGBOOST

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Dr. Fabiano Baldo

JOINVILLE

2024

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Universitária Udesc,

com os dados fornecidos pelo(a) autor(a)

Policarpo, Deividy Amorim
SSAFXGB: Classificação Semi-Supervisionada Multi-Classe de Fluxo de Dados com a Utilização do XGBOOST / Deividy Amorim Policarpo. -- 2024.
86 p.

Orientador: Fabiano Baldo
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação , Joinville, 2024.

1. classificação multi-classe. 2. semi-supervisionado. 3. fluxo de dados. 4. pseudo rotulagem. 5. xgboost. I. Baldo, Fabiano. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação . III. Título.

DEIVIDY AMORIM POLICARPO

SSAFXGB: CLASSIFICAÇÃO SEMI-SUPERVISIONADA MULTI-CLASSE DE FLUXO DE DADOS COM A UTILIZAÇÃO DO XGBOOST

Esta dissertação foi julgada adequada para a obtenção do título de **Mestre em Computação Aplicada** área de concentração em "Ciência da Computação", e aprovada em sua forma final pelo Curso de Mestrado em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina.

Banca Examinadora:

Dr. Fabiano Baldo
CCT/UDESC
Presidente/Orientador

Dr. Paulo Ricardo Lisboa de Almeida
UFPR

Dr. Yuri Kaszubowski Lopes
CCT/UDESC

Joinville, 09 de Maio de 2024

AGRADECIMENTOS

Primeiramente, gostaria de expressar minha profunda gratidão a Deus, por me guiar e sustentar em cada passo deste caminho.

À minha amada família, em especial à minha esposa Daiana e minha filha Elisa Valentina, por seu amor incondicional, paciência e constante incentivo. Seu apoio inabalável foi essencial para superar os desafios e perseverar durante este período de estudo.

Gostaria de expressar minha sincera gratidão ao meu orientador, Professor Dr. Fabiano Baldo, por sua orientação, apoio e valiosos *insights* ao longo deste trabalho. Sua orientação foi fundamental para o desenvolvimento deste projeto e para o meu crescimento acadêmico e profissional. Agradeço também aos membros da banca examinadora pelas valiosas contribuições e sugestões para o aprimoramento do trabalho.

Agradeço à Universidade do Estado de Santa Catarina CEAVI pela facilitação da realização deste trabalho, ao CCT por fornecer recursos e ambiente propício para o desenvolvimento desta pesquisa, e também ao Programa de Pós-Graduação em Computação Aplicada pela oportunidade. Agradeço também aos colegas Júlia e Yuji por suas contribuições e apoio ao longo deste processo.

Por fim, expresso meu profundo agradecimento a todos que de alguma forma contribuíram para este trabalho, direta ou indiretamente. Suas contribuições foram fundamentais para a conclusão desta dissertação.

“Duvidar de tudo ou acreditar em tudo são duas atitudes igualmente cômodas, que nos dispensam ambas de refletir.”
Henri Poincaré (1854–1912), físico e matemático francês.

RESUMO

O surgimento de novas tecnologias estão aumentando o volume de dados criados através de fluxos (*data streaming*), apresentando características, como instâncias não rotuladas e mudança de conceitos, que requerem o desenvolvimento de novas abordagens para realizar sua análise. No entanto, existem poucos estudos que propuseram algoritmos para lidar com a classificação multi-classe, semi-supervisionada, em fluxos de dados com mudança de conceito. As propostas existentes possuem baixo desempenho quanto ao seu tempo de execução. Portanto, este trabalho propõe o SSAFXGB, um algoritmo adaptativo semi-supervisionado para classificação multi-classe de fluxos de dados com instâncias parcialmente rotuladas e com suporte à mudança de conceito. O *XGBoost* foi usado como classificador base, e foi estendido para suportar o aprendizado semi-supervisionado por meio de *wrappers*. Neste contexto, foram implementados os métodos 1NN e KNN, com a capacidade adicional de incorporar outros classificadores conforme necessário para atender às demandas específicas de diferentes problemas. Além disso, foi integrado suporte para geração de super amostragem como uma estratégia alternativa para abordar o desbalanceamento de classes, potencializando a eficácia do modelo em cenários com distribuições de classe desiguais. Os resultados mostram que a solução proposta apresentou melhor acurácia em todos os *datataset's* avaliados com média de quase 20 pontos percentuais melhor do que seu equivalente da literatura, e em comparação com os métodos clássicos para fluxo a presente proposta ficou consistentemente em primeiro colocado no *ranking* das métricas avaliadas (Acurácia, *Kappa* e F1) em todos *dataset's* e cenários elaborados, com boa performance computacional.

Palavras-chaves: classificação multi-classe, semi-supervisionado, fluxo de dados, dados não estacionários, pseudo rotulagem, *wrapper*, xgboost, *ensemble*.

ABSTRACT

The emergence of new technologies is increasing the volume of data generated through streaming, presenting characteristics such as unlabeled instances and concept drift, which require the development of new approaches to perform their analysis. However, there are few studies that have proposed algorithms to handle multi-class, semi-supervised classification in data streams with concept drift. The existing proposals have low performance in terms of their execution time. Therefore, this work proposes SSAFXGB, an adaptive semi-supervised algorithm for multi-class classification of data streams with partially labeled instances and support for concept drift. XGBoost was used as the base classifier and was extended to support semi-supervised learning through wrappers. In this context, 1NN and KNN methods were implemented, with the additional capability to incorporate other classifiers as needed to meet the specific demands of different problems. Furthermore, support for oversampling generation was integrated as an alternative strategy to address class imbalance, enhancing the model's effectiveness in scenarios with imbalanced class distributions. The results show that the proposed solution presented better accuracy across all evaluated datasets, with an average of almost 20 percentage points better than its literature equivalent. Compared to classic methods for data streams, the present proposal consistently ranked first in the evaluated metrics (Accuracy, Kappa and F1) across all datasets and scenarios, with good computational performance.

Key-words: classification, multi-class, semi-supervised, data stream, data streaming, nonstationary data, ensemble, wrapper, xgboost, pseudo labeling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1NN	*First Nearest Neighbor* (Primeiro Vizinho Mais Próximo)
ADASYN	Adaptive Synthetic Sampling (Amostragem Sintética Adaptativa)
ADWIN	*Adaptive Windowing* (Janela Adaptativa)
AFXGB	*Adaptive Fast eXtreme Gradient Boosting*
AFXGB-MC	*AFXGB, Multi-Class (Multi-Classe)*
API	*Application Programming Interface* (Interface de Programação de Aplicativos)
AXGB	*Adaptive eXtreme Gradient Boosting*
CD	*Critical Difference* (Diferença Crítica)
BN	*Binary* (Binária)
DDM	*Drift Detection Method* (Método de Detecção de Mudanças de Conceito)
FIFO	First In, First Out (Primeiro a entrar, primeiro a sair)
IoT	*Internet of Things* (Internet das Coisas)
KNN	*K-Nearest Neighbor* (K-vizinho mais Próximo)
MC	*Multi-Class* (Multi-Classe)
PROP	Proporcional
PROPMEAN	Proporcional com Média
ROS	Random OverSampling (Sobre amostragem Aleatória)
RP	*Prevalence Ratio* (Razão de prevalência)
SMOTE	Synthetic Minority Over-sampling Technique (Técnica de Sobre amostragem Sintética da Minoria)
SSL	*Semi-Supervised Learning* (Aprendizado Semi-Supervisionado)
SSAFXGB	Classificação semi-supervisionada multi-classe de fluxo de dados com XGBOOST
SSAFXGB-1NN	SSAFXGB, com *wrapper* 1NN
SSAFXGB-KNN	SSAFXGB, com *wrapper* KNN
XGBoost	*eXtreme Gradient Boosting*

SUMÁRIO

1 INTRODUÇÃO

A crescente adoção e utilização de dispositivos eletrônicos conectados, dotados de sensores e interatividade social, impulsionada pela Internet das Coisas (IoT, sigla em Inglês) e pela tecnologias 5G, tem proporcionado a geração cada vez maior e mais frequente de dados. Como consequência dessa situação, tem-se uma abundância de informações disponíveis para análise e apoio à tomada de decisão. A análise desse conjunto massivo de dados traz oportunidades para a geração de informações valiosas para a tomada de decisão. Entretanto, essa análise não é um processo trivial, o que demanda do auxílio de técnicas computacionais capazes de extrair padrões de grandes volumes de dados. Não obstante, devido ao crescente aumento da quantidade e da diversidade de dados que estão sendo gerados, percebe-se a necessidade de aperfeiçoamento das técnicas de aprendizado de máquina tradicionais (HABASHY et al., 2023).

Dentre as mais recentes formas de geração de dados, destacam-se os fluxos de dados (*data streaming*), que acontecem em cenários onde os dados são gerados de forma contínua, ininterrupta e não estacionária. Fluxos de dados não estacionários são aqueles em que as propriedades estatísticas da variável dependente que o modelo está tentando prever mudam ao longo do tempo de maneiras imprevistas (BARROS; SANTOS, 2019). Essa característica também é conhecida como mudança de conceito. Quando ocorrem mudanças de conceito o modelo treinado perde sua eficácia na predição de novos dados, levando a resultados de decisão ruins (LU et al., 2019). Ainda, por ser ininterrupto, não é possível extrair uma base de treinamento prévia capaz de representar todas as situações/condições presentes nos dados para serem utilizadas no treinamento do modelo. Portanto, os exemplos são apresentados sequencialmente durante o treinamento do modelo à medida que são coletados, o que faz com que os modelos precisem se adaptar constantemente (OSOJNIK; PANOV; DŽEROSKI, 2017).

Por outro lado, instâncias não rotuladas estão comumente presentes em fluxos de dados devido ao fato de que o processo de rotulagem tende a ser caro e demorado (WANG; LI; ZHOU, 2019). A escassez de rótulos em tempo real em fluxos de dados apresenta um desafio significativo quando apenas um pequeno número de exemplos rotulados está disponível (CHEN et al., 2021). Uma solução possível é usar técnicas de treinamento semi-supervisionadas, que utilizam dados não rotulados para aprimorar o treinamento do modelo e assim predizer com maior precisão pontos em um fluxo com uma pequena quantidade de exemplos rotulados (WAGNER et al., 2018).

A escassez de dados rotulados e o alto custo de rotulação manual são desa-

fios importantes nessa área de aprendizado de máquina (WANG; LI; ZHOU, 2019), e por isso um dos maiores motivadores por trás dessa pesquisa. A abordagem semi-supervisionada surge como uma alternativa promissora para lidar com essa limitação, pois permite que modelos sejam treinados com menos dados rotulados do que o normalmente necessário em abordagens puramente supervisionadas.

Entretanto, apesar do crescente surgimento de novas fontes geradoras de fluxos de dados não estacionários, a maior parte das pesquisas no campo do aprendizado de máquina está centrada na construção de modelos para fontes de dados estacionárias. Estes modelos são treinados com um conjunto de dados específico e, posteriormente, são utilizados para a predição de novos dados. Essa abordagem tem sido amplamente explorada no contexto da Ciência de Dados e permite a utilização de modelos pré-treinados para tomar decisões e fazer previsões em diferentes cenários (AGRAHARI; SINGH, 2022). Contudo, as técnicas tradicionais de aprendizado supervisionado não são adequadas para os cenários de fluxos de dados parcialmente rotulados, pois não se adaptam a mudança de conceito e não aprendem com instâncias não rotuladas, condição que faz com que o aprendizado supervisionado não seja suficiente para este cenário (HU; KANTARDZIC; SETHI, 2020).

A adaptação à mudança de conceito é uma tarefa desafiadora na análise de fluxos de dados. Nos métodos tradicionais, um modelo de classificação é treinado com um subconjunto de dados que se presume representar completamente o problema, e limites de decisão são descobertos para gerar predições futuras a partir da mesma distribuição dos dados de treinamento. Entretanto, em um fluxo de dados não estacionário, as características dos dados e os limites de decisão estão sujeitos a alterações, exigindo que os modelos sejam atualizados para se adaptarem às mudanças. O simples reconhecimento da ocorrência de mudanças é desafiador, assim como a necessidade de reagir e se adaptar a elas.

Outro desafio no contexto da análise de fluxos de dados é que o algoritmo deve suportar atualizações rápidas, pois o fluxo pode ter alta frequência de geração de dados. Considerando a velocidade de processamento de dados, um algoritmo relevante neste aspecto é o *eXtreme Gradient Boosting* (XGBoost). O XGBoost é um algoritmo baseado em *boosting* que cria cada árvore individual em paralelo, reduzindo assim o tempo de treinamento dos modelos (CHEN; GUESTRIN, 2016a). Algoritmos de *boosting* combinam modelos fracos que são induzidos iterativamente, sendo que o próximo modelo tenta resolver os erros do anterior, melhorando o resultado final da previsão (MAYR et al., 2014). Embora o XGBoost não suporte aprendizado semi-supervisionado, há na literatura algumas abordagens semi-supervisionadas que poderiam ser incorporadas a ele, tais como, o autotreinamento, o co-treinamento e a pseudo-rotulagem por meio de métodos *wrapper* (ENGELEN; HOOS, 2020).

No contexto de classificação existe uma distinção entre dois tipos principais de classificação, a chamada classificação binária (variável dicotômica) e a classificação multi-classe (variável categórica). Na classificação binária o modelo precisa predizer a classe tendo apenas duas opções, por exemplo, “verdadeiro” ou “falso”, “0” ou “1”, “quente” ou “frio”, etc. Já na classificação multi-classe o modelo precisa predizer uma classe dentre um conjunto finito de classes que pode variar de 3 a dezenas de classes. O segundo caso é o mais desafiador pois, como os modelos de predição definem a classe do dado baseado em probabilidades, quanto maior a quantidade de classes, mas difícil se torna obter classificações com alto grau de probabilidade. Portanto, partindo da condição de que o processo de aprendizado semi-supervisionado em classificação binária é uma tarefa complexa, aplicar esse tipo e aprendizado à classificação multi-classe se torna uma tarefa ainda mais complexa (BARBER; BOTEV, 2016). A literatura apresenta poucas iniciativas para solução deste tipo de problema, dentre elas, a que merece maior destaque é o IOE (*Improved Online Ensembles*) proposto por Vafaie, Viktor e Michalowski (2020), apresentado em detalhes na Seção 2.7.

O problema do desbalanceamento de classes é outro aspecto que intensifica as dificuldades na análise de fluxos de dados, onde é comum que algumas classes sejam representadas por uma quantidade significativamente maior de dados do que outras, resultando em um modelo que tende a favorecer a classe majoritária, enquanto negligencia as minoritárias. Esse desbalanceamento pode prejudicar consideravelmente a precisão do modelo em relação às classes menos representadas, uma vez que as decisões tendem a ser enviesadas em favor das classes com mais exemplos (HE; GARCIA, 2009). Técnicas como a re-amostragem dos dados, tanto por subamostragem da classe majoritária quanto por super amostragem da minoritária, podem mitigar esses efeitos, entretanto, sua aplicação em fluxos de dados pode ser desafiadora devido à natureza dinâmica e contínua deles (LOSING; HAMMER; WERSING, 2018).

Além disso, a presença de mudança de conceito em fluxos de dados torna o desbalanceamento de classes ainda mais problemático, pois as distribuições das classes podem variar ao longo do tempo, exacerbando ou atenuando o desbalanceamento inicial. Isso requer que os modelos de classificação não apenas lidem com o desbalanceamento, mas também se adaptem continuamente às novas distribuições de classes que surgem conforme os dados evoluem. Gama et al. (2014) propõem equilibrar a necessidade de adaptação ao novo conceitos por meio da utilização de *ensembles* que ajustam seus classificadores periodicamente para refletir as mudanças nas distribuições das classes e no próprio conceito dos dados.

Dados os desafios da classificação semi-supervisionada de fluxos de dados, o presente trabalho tem como pergunta de pesquisa: Como resolver o problema de

classificação de fluxos de dados multi-classe parcialmente rotulados, desbalanceados e com presença de mudança de conceito buscando alto grau de acurácia e baixo tempo de processamento?

Levando em consideração esta pergunta de pesquisa, o presente trabalho apresenta a premissa de que o XGBoost pode ser uma alternativa relevante a ser utilizada como classificador base para o desenvolvimento do método de classificação multi-classe semi-supervisionado, principalmente por ser conhecidamente um dos algoritmos de indução de modelos mais rápidos presentes na literatura (CHEN; GUESTRIN, 2016b). Ainda, dentre as alternativas para o suporte ao aprendizado semi-supervisionado, a técnica de *wrapper* se apresenta como uma alternativa relevante, pois não demanda de alterações internas no algoritmo base, sendo executada externamente, como um pré-processamento, que realiza a chamada pseudo-rotulagem das instâncias que são posteriormente utilizadas no treinamento do modelo pelo classificador base. A pseudo-rotulagem é uma técnica que atribui um rótulo/classe temporária para o dado que precisa ser avaliada antes de ser considerada correta (LI; YIN; CHEN, 2022).

1.1 OBJETIVO

Este trabalho tem por objetivo propor uma solução para a classificação multi-classe semi-supervisionada de fluxos de dados utilizando o XGBoost como algoritmo base e a técnica *wrapper* com estratégia de pseudo-rotulagem. O algoritmo proposto é denominado *Semi-Supervised Adaptive Fast XGBoost for Multiclass Stream Classification* (SSAFXGB), uma adaptação de Baldo et al. (2023) originalmente projetada para classificação multi-classe supervisionada.

1.1.1 Objetivos Específicos

O presente trabalho apresenta os seguintes objetivos específicos:

1. Projetar uma estratégia para resolver a pseudo-rotulagem baseada na técnica *wrapper*;
2. Propor uma alternativa para melhorar o balanceamento das classes utilizando técnicas de re-amostragem dos dados;
3. Criar um protótipo da solução e avaliar os resultados utilizando métricas reconhecidas pela literatura.

1.2 METODOLOGIA DE PESQUISA

Nesta seção, fornecemos uma visão geral da caracterização metodológica adotada nesta pesquisa, juntamente com uma descrição detalhada do procedimento utilizado para o desenvolvimento deste estudo.

1.2.1 Caracterização Metodológica

Com base nas classificações estabelecidas por Wazlawick (2017), este estudo é classificado como de natureza primária, pois tem como objetivo gerar novo conhecimento por meio da construção e experimentação de uma abordagem de aprendizado de máquina com características diferenciadas ao abordado anteriormente na literatura, pois trata-se de uma classificação semi-supervisionada de fluxo de dados não estacionários, com tratamento ao desbalanceamento de classes, e redução de recursos computacionais. O foco desta pesquisa é agilizar o desafio de classificação multi-classe em fluxos de dados não estacionários, mantendo ou melhorando a qualidade das predições apresentadas na literatura.

Quanto ao objetivo desta pesquisa, ela pode ser classificada como explicativa, pois apresenta dados sobre os experimentos e analisa o comportamento do algoritmo proposto em diferentes conjuntos de dados. Em relação ao procedimento técnico, pode ser categorizada como experimental, uma vez que envolve a manipulação dos hiperparâmetros do algoritmo proposto e a realização de medições e observações dos resultados para ratificar as hipóteses estabelecidas, utilizando análises estatísticas.

1.2.2 Procedimento Metodológico

O procedimento metodológico adotado nesta pesquisa inicia com uma revisão bibliográfica dos conceitos fundamentais relacionados à classificação de dados, aprendizado semi-supervisionado, conjuntos de árvores de decisão, fluxos de dados, pseudo-rotulagem e técnicas para tratamento de dados com classes desbalanceadas. Além disso, é realizado um estudo abrangente sobre os trabalhos relacionados encontrados na literatura.

Com base nesses conceitos fundamentais, é realizado o projeto da arquitetura base do método de classificação multi-classe semi-supervisionado proposto. Essa arquitetura é inspirada nos trabalhos desenvolvidos por Baldo et al. (2023), que propuseram classificação supervisionada multi-classe, e por Vafaie, Viktor e Michalowski (2020), que propuseram uma solução para a classificação multi-classe semi-supervisionada.

Assim como em Vafaie, Viktor e Michalowski (2020), o método proposto implementa o aprendizado semi-supervisionado por meio de uma abordagem baseada

em *wrapper*, que realiza a pseudo-rotulagem dos dados não rotulados recebidos pelo fluxo. Para aferir a qualidade da pseudo-rotulagem é utilizada uma métrica que calcula a margem de incerteza. Além disso, também são incorporadas ao método proposto alternativas de re-amostragem para tratar o desbalanceamento de classes dos dados recebidos no fluxo.

Com base no método proposto, é implementado um protótipo em linguagem `Python` utilizando o suporte das bibliotecas de código aberto. Esse protótipo é utilizado na realização de experimentos para analisar o desempenho do método proposto utilizando as métricas de acurácia, *Kappa* de Cohen, F1-score e tempo total de execução em diversos cenários com quantidades variadas de dados não rotulados.

As análises são realizadas executando o algoritmo implementado sobre conjuntos de dados reais e sintéticos simulados como fluxos. A utilização de bibliotecas especializadas para a geração de fluxos de dados sintéticos permite a realização de experimentos em ambientes controlados, onde é possível incluir estrategicamente pontos de mudança de conceito. Essas análises têm como objetivo fornecer uma avaliação objetiva e comparativa do desempenho do algoritmo proposto em diferentes cenários de fluxo de dados.

Por fim, é realizada uma avaliação comparativa entre o presente trabalho e a solução proposta por Vafaie, Viktor e Michalowski (2020), algoritmo mais recente que aborda o mesmo problema de aprendizado de máquina, além de comparações com algoritmos clássicos especializados na análise de fluxo de dados. Essas comparações tem por objetivo identificar se o método proposto traz contribuições na geração de modelos com maior adaptabilidade e generalização, com diminuição no tempo de execução.

1.3 ESTRUTURA DO TRABALHO

A estrutura deste trabalho está organizada da seguinte maneira. No Capítulo 2 são apresentados os fundamentos teóricos da pesquisa, abrangendo conceitos de aprendizado de máquina, árvores de decisão, conjuntos de classificadores (*ensembles*), fluxo de dados e mudança de conceito, além da revisão dos trabalhos relacionados. No Capítulo 3 é apresentada a solução proposta, incluindo o fluxo e o pseudo-código de implementam o método proposto e os hiper-parâmetros de configuração utilizados. O Capítulo 4 descreve os resultados alcançados e faz uma comparação com o principal trabalho relacionado da literatura. Finalmente, o Capítulo 5 traz as conclusões e sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo é fornecida uma base teórica dos conceitos abordados neste trabalho, que se concentram no aprendizado de máquina utilizando conjuntos de árvores de decisão, especificamente aplicados na classificação multi-classe de fluxos de dados, incluindo a ocorrência de mudança de conceito. Além disso, são apresentados os trabalhos relacionados encontrados na literatura, fazendo uma contextualização do estado da arte na classificação multi-classe e semi-supervisionada de fluxos de dados não estacionários.

2.1 APRENDIZADO DE MÁQUINA

O conceito de Aprendizado de Máquina (AM) surgiu na década de 1950, como um ramo da inteligência artificial. O termo em si foi cunhado por Samuel (1959), um dos pioneiros nesta área de estudo. Ele definiu o AM como a capacidade de um sistema de computador de aprender automaticamente sem ser explicitamente programado. Mitchell et al. (2007) trouxeram uma visão mais pragmática que definem AM como a construção de programas que melhoram seu desempenho por meio de uma grande quantidade de exemplos utilizados para gerar o conhecimento do computador, que são hipóteses geradas a partir dos dados.

Goodfellow, Bengio e Courville (2016) afirmam que AM é um ramo que se concentra no desenvolvimento de algoritmos e modelos computacionais capazes de generalizar, aprender e melhorar automaticamente a partir de dados. Os modelos de AM são treinados usando dados de entrada para identificar padrões, fazer previsões ou tomar decisões.

2.1.1 Tarefas de aprendizado

A classificação e a regressão são as principais tarefas desempenhadas pelos algoritmos de AM. Ambas estão relacionadas à previsão ou inferência de valores, mas se diferenciam conceitualmente em termos das saídas desejadas e da natureza dos problemas que resolvem (MURPHY, 2012).

O objetivo da classificação é atribuir uma classe a cada instância de uma determinada amostra predefinida a partir de um conjunto discreto de classes possíveis, visando construir um modelo que possa aprender a mapear as características das instâncias da amostra para suas respectivas classes. O resultado da classificação é a atribuição de uma classe/categoria para cada instância da amostra. Já a regressão tem por finalidade prever um valor numérico ou contínuo com base no conjunto de

características das instâncias da amostra. Portanto, diferentemente da classificação, em que a saída é um valor qualitativo, a regressão envolve a previsão de um valor quantitativo.

Como mencionado por Hastie et al. (2009), “na classificação a resposta é discreta enquanto que na regressão ela é contínua”. Essa distinção fundamental entre os dois conceitos é crucial para a compreensão e aplicação adequada dessas tarefas no campo do aprendizado de máquina.

2.1.2 Métricas de avaliação de desempenho

Tarefas de AM possuem diversas métricas que são utilizadas para avaliar o desempenho dos modelos, e são essenciais para definir o grau de confiança que pode ser depositado no modelo e garante a segurança nas decisões baseadas em tal resultado (MENDITTO; PATRIARCA; MAGNUSSON, 2007).

Algumas das principais métricas de classificação segundo Grandini, Bagli e Visani (2020) são:

- **Acurácia (ou exatidão[1])** é uma métrica de avaliação de desempenho comum que calcula a proporção de previsões corretas em relação ao total de previsões realizadas, definida como o quociente entre o número de classificações precisas e o número total de exemplos. Amplamente utilizada devido à sua simplicidade (SOKOLOVA; LAPALME, 2009).

  A acurácia é calculada como a proporção de predições corretas (verdadeiros positivos e verdadeiros negativos) em relação ao total de casos examinados:

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

  Onde TP refere-se aos verdadeiros positivos, TN são os verdadeiros negativos, FP são falsos positivos e FN os falsos negativos.

  Precisão é uma métrica que mede a proporção de verdadeiros positivos entre todos os resultados positivos. Ela é útil quando o custo de falsos positivos é alto (GIRALDO-FORERO et al., 2015). Sua fórmula de calculo é:

  $$\text{Precisão} = \frac{TP}{TP + FP} \tag{2.2}$$

  Sensibilidade/**Revocação** (*Recall*): é a razão entre o número de verdadeiros positivos e o número total de elementos que realmente são positivos (verdadeiros positivos mais falsos negativos). Mede a capacidade do modelo de identificar

[1] Acurácia e exatidão são termos sinônimos que se referem à mesma métrica. No entanto, a literatura recente parece favorecer o uso do termo “acurácia” e portanto, será o vocábulo adotado daqui em diante.

todos os elementos relevantes de uma classe específica, sendo particularmente útil quando a falha em identificar positivos é crítica (SOKOLOVA; LAPALME, 2009). A fórmula para calculo é:

$$\textit{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

- **Kappa de Cohen** é uma métrica que avalia a concordância entre as previsões do classificador e as verdadeiras classes, ajustando para a concordância esperada ao acaso. Ele é especialmente útil em datasets desbalanceados, onde a acurácia pode ser enganosa. O Kappa varia de -1 a 1, onde 1 indica perfeita concordância, 0 indica concordância ao nível do acaso e valores negativos indicam concordância pior que o acaso (Grandini et al., 2020). A fórmula para calculo é:

$$\textit{Kappa} = \frac{p_o - p_e}{1 - p_e} \tag{2.4}$$

  Onde p_o é a proporção de concordância observada e p_e é a proporção de concordância esperada ao acaso.

- **F1 e F1 Macro**: O F1-Score é a média harmônica entre precisão e recall, fornecendo um único valor que equilibra as duas métricas. É útil quando há um desequilíbrio entre as classes positivas e negativas, pois considera tanto falsos positivos quanto falsos negativos. O F1-Score varia de 0 a 1, onde 1 indica a melhor performance possível. A fórmula para calcular o F1-score é:

$$\textit{F1-score} = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}} \tag{2.5}$$

  O F1 Macro é uma extensão do F1-Score para classificação multi-classe, onde o F1-Score é calculado para cada classe individualmente e depois se faz a média desses valores, dando igual peso a cada classe. Isso é útil para avaliar o desempenho de um modelo em conjuntos de dados desbalanceados, onde algumas classes podem ter menos exemplos. Em contraste, o F1 Micro trata todas as instâncias igualmente, somando os verdadeiros positivos, falsos negativos e falsos positivos de todas as classes antes de calcular o F1-Score, o que pode favorecer classes majoritárias (GRANDINI; BAGLI; VISANI, 2020). A fórmula para calcular o F1 Macro é:

$$\textit{F1 Macro} = \frac{2 \cdot \sum_{i=1}^{C} \frac{\textit{Precision}_i \cdot \textit{Recall}_i}{\textit{Precision}_i + \textit{Recall}_i}}{C} \tag{2.6}$$

  Onde C é o número de classes.

Fatourechi et al. (2008) propõe uma estrutura para comparar métricas de avaliação em aplicações de classificação com *datasets* desbalanceados. Eles demonstram

que, embora o F1-Score seja importante, a Acurácia e o *Kappa* também são relevantes, pois fornecem diferentes perspectivas sobre a performance do classificador. Em particular, o Kappa é destacado como mais adequado para avaliar a concordância ajustada ao acaso em comparação com a Acurácia.

Wardhani et al. (2019) comparam várias métricas para avaliar a performance de classificação em dados desbalanceados. Os autores concluíram que o uso de métricas múltiplas, incluindo Acurácia, F1-Score e *Kappa*, proporciona uma avaliação mais completa e robusta da performance dos classificadores, especialmente em dados desbalanceados.

Em contraponto, Chicco et al. (2021) destacam uma limitação importante do *Kappa* de Cohen em situações específicas, particularmente quando as probabilidades marginais são muito pequenas, o que pode levar a resultados enganosos. Enquanto Takahashi et al. (2022) explora a confiabilidade dos intervalos de confiança para F1-Scores micro e macro-*averaged* em problemas de classificação binária e multi-classe. Eles enfatizam que as propriedades estatísticas dos F1 não são frequentemente discutidas, e que isso pode levar a uma interpretação inadequada da performance do classificador.

Ou seja, o F1-Score embora útil, possui limitações, especialmente em situações de baixa prevalência e quando se considera a confiabilidade e a incerteza do desempenho futuro. Portanto, faz sentido utilizar múltiplas métricas, como Acurácia e *Kappa*, além do F1, para obter uma avaliação mais robusta e abrangente da performance dos modelos de classificação. A Acurácia fornece uma visão geral, o F1 equilibra precisão e *recall*, e o *Kappa* ajusta para a concordância ao acaso. Juntas, essas métricas proporcionam uma avaliação mais robusta e abrangente, essencial em projetos de classificação.

2.1.3 Categorias de aprendizado

Embora não haja consenso na comunidade científica sobre a divisão das subcategorias do AM, a maioria dos autores subdividem-os nas seguintes categorias (SARKER, 2021):

- **Supervisionado**: é a abordagem em que um modelo é treinado para fazer previsões ou tomar decisões com base em exemplos rotulados previamente. O objetivo é fazer com que o modelo aproxime uma função $f(X) = y$ que mapeie as entradas X para as saídas y desejadas, com base nos dados de treinamento disponíveis.

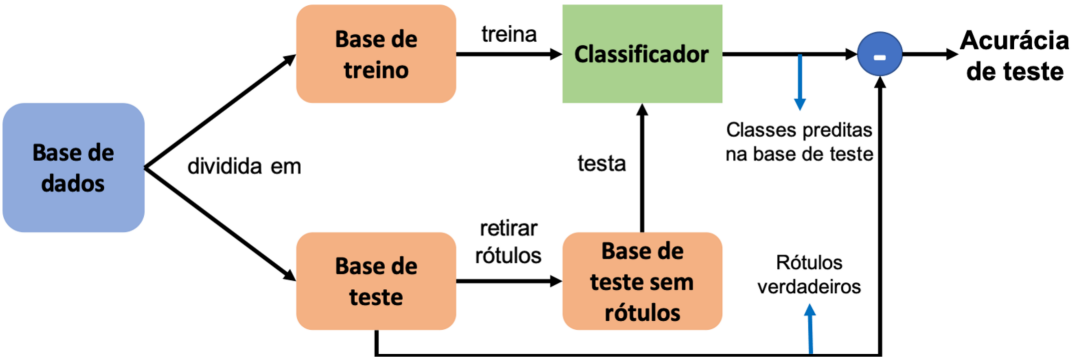
  O conjunto de dados de treinamento consiste em pares (X, y), onde "X" são as características ou atributos das amostras e "y" são os rótulos correspondentes

que representam as respostas desejadas. Durante o processo de treinamento, o modelo aprende a generalizar a partir desses exemplos rotulados, para que possa fazer previsões em novos dados não vistos anteriormente.

O treinamento do modelo ocorre por meio de algoritmos que buscam minimizar a diferença entre as previsões do modelo e os rótulos reais fornecidos nos dados de treinamento. Isso é feito ajustando os parâmetros internos do modelo com base em uma função de perda, que quantifica a discrepância entre as previsões e os rótulos reais. O objetivo é encontrar os parâmetros que minimizam essa função de perda, permitindo que o modelo generalize para novos dados e faça previsões (HASTIE et al., 2009).

A Figura 1 apresenta o fluxograma resumido da classificação supervisionada. Nesse exemplo, os dados podem ser segmentados randomicamente entre treinamento e teste, e após o treinamento, utilizando os dados de teste, é possível verificar se o modelo é eficaz ao realizar as predições comparando os resultados da predição com os rótulos reais dos dados do segmento de teste.

Figura 1 – Fluxo resumido de classificação supervisionada



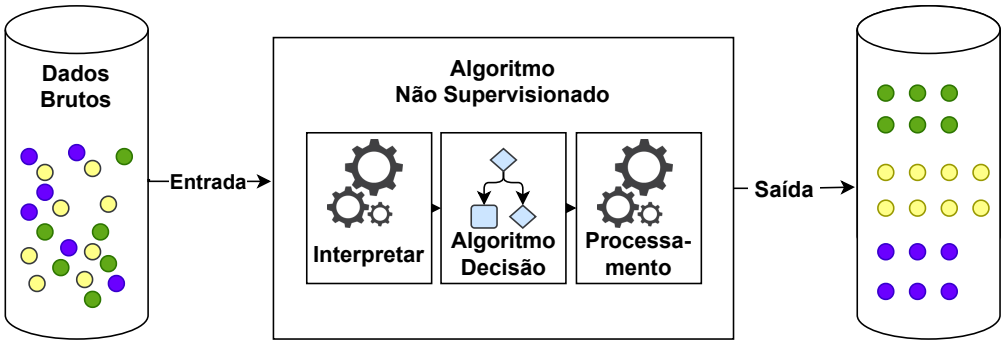
Fonte: (ESCOVEDO; KOSHIYAMA, 2020).

- **Não supervisionado**: é uma abordagem utilizada em conjuntos de dados não rotulados que busca encontrar padrões, estruturas ou agrupamentos intrínsecos nos dados. Ao contrário do aprendizado supervisionado, não há saídas ou respostas desejadas fornecidas durante o treinamento e também não há geração de modelos. O algoritmo utiliza um conjunto de dados que não possuem rótulos ou categorias conhecidas. O objetivo é descobrir informações úteis a partir desses dados, como grupos naturais, relações, tendências ou características latentes. O modelo busca aprender a estrutura subjacente dos dados e encontrar padrões significativos, sem qualquer informação prévia sobre as classes ou categorias (BISHOP; NASRABADI, 2006).

  A Figura 2 demonstra um exemplo do processo executado por um algoritmo não-supervisionado. Como pode ser visto, ele tem por objetivo apenas extrair padrões

de agrupamento dos dados com base nos valores de suas propriedades, sem levar em consideração o rótulo da variável dependendo, pois essa informação não está disponível.

Figura 2 – Fluxo do Aprendizado Não-Supervisionado



Fonte: O autor.

- **Semi-Supervisionado**: é uma abordagem que combina elementos de aprendizado supervisionado e não supervisionado. Nesse tipo de aprendizado, um modelo é treinado usando um conjunto de dados que contém tanto exemplos rotulados quanto não rotulados (para o caso de classificação). Os exemplos rotulados são utilizados para guiar o processo de aprendizado, enquanto os exemplos não rotulados fornecem informações adicionais que podem melhorar o desempenho do modelo (ZHU, 2010).

- **por Reforço**: diferente do aprendizado supervisionado, onde exemplos rotulados são usados para treinar o modelo, e do aprendizado não supervisionado, onde não há rótulos disponíveis, o aprendizado por reforço é baseado em um sistema de recompensas e penalidades. Nesse tipo de aprendizado, um agente interage com um ambiente dinâmico e observa o estado atual do ambiente. Com base nessa observação, o agente toma uma ação e o ambiente responde com uma nova observação e uma recompensa. O objetivo do agente é aprender uma política, ou seja, uma estratégia de ação, que maximize a recompensa acumulada ao longo do tempo (SUTTON; BARTO, 2018).

2.2 ÁRVORES DE DECISÃO

Árvore de decisão é uma técnica de aprendizado supervisionado que visa construir um conjunto de regras de decisão hierarquicamente organizadas na forma de uma estrutura de árvore. Essa estrutura é composta por nós internos, que representam testes em atributos, e nós folha, que resultam em classes ou valores de saída com base em múltiplas variáveis de entrada (KOTSIANTIS, 2013).

Uma árvore de decisão começa com um nó raiz que representa o conjunto de dados completo. Em seguida, são feitas análises sobre os atributos das instâncias de cada nó interno para dividir o conjunto de dados em subconjuntos menores. Essas análises são feitas com base em métricas que buscam maximizar a separação entre as classes ou otimizar a predição dos valores de saída. Essa divisão continua até que sejam alcançados os critérios de parada pré-definidos, como uma pureza mínima nas folhas ou uma profundidade máxima da árvore.

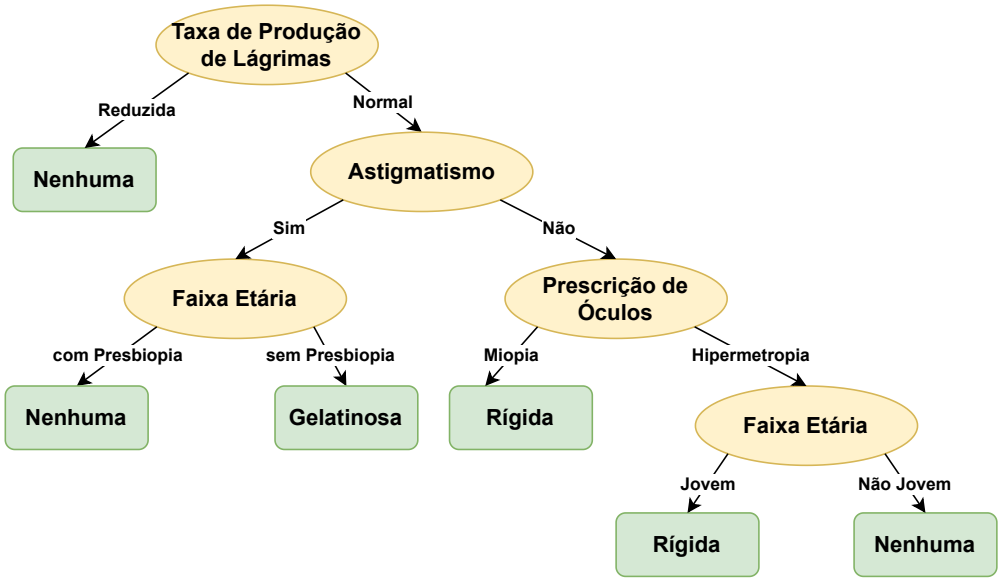
Uma vez construída a árvore de decisão, ela pode ser utilizada para fazer previsões sobre novos exemplos de dados. Para isso, basta percorrer a árvore, seguindo as respostas às perguntas até chegar a uma folha, que contém a classe ou o valor de saída previsto. As árvores de decisão são amplamente utilizadas em diferentes áreas, nas tarefas de classificação, regressão e seleção de recursos, e podem ser facilmente aplicadas em problemas do mundo real.

A Figura 3 apresenta um exemplo de árvore de decisão. Neste caso, o objetivo é prever a classe (tipo de lente) que uma pessoa pode utilizar, e para tal pode-se avaliar quatro atributos, representados por elipses amarelas. Os "nós" de decisão são criados de acordo com a relevância dos atributos na predição da classe, e as classes estão representadas pelos retângulos verdes com os seguintes valores: **Nenhuma**, **Gelatinosa** ou **Rígida**. Os atributos utilizados nesse exemplo são: "Taxa de produção de lágrimas", onde os valores possíveis são `Reduzida` ou `Normal`, "Astigmatismo", cujos valores podem ser `Sim` ou `Não`, "Faixa Etária", que pode conter quatro valores possíveis `com/sem Presbiopia` ou `Jovem/Não Jovem` e a "Prescrição de óculos", que pode ser `Miopia` ou `Hipermetropia`. Nesse exemplo, uma pessoa com "Taxa de produção de lágrimas" `reduzida` não pode utilizar utilizar lentes (classe **Nenhuma**), mas uma pessoa com "Taxa de produção de lágrimas" `normal`, que `não` possua "Astigmatismo", e que sua "Prescrição de Óculos" seja `miopia`, poderá utilizar lente do tipo Rígida (classe **Rígida**).

A construção de uma árvore de decisão envolve a seleção cuidadosa de atributos e a criação de regras de divisão para segmentar os dados de treinamento em subconjuntos cada vez mais puros (HUANG; HOA, 2009). De forma geral, algumas etapas-chave para construir uma árvore de decisão são:

- **Seleção do atributo**: o primeiro passo é selecionar o atributo mais relevante para dividir os dados. Existem várias métricas utilizadas para medir a relevância, como entropia, índice de Gini e taxa de erro de classificação. O objetivo é escolher o atributo que resulte em maior ganho de informação ou redução da impureza.
- **Divisão dos dados**: com base no atributo selecionado, os dados de treinamento

Figura 3 – Árvore de decisão para classificação multi-classe



Fonte: Adaptado de CMU (2023).

são divididos em subconjuntos com base nos valores desse atributo. Cada subconjunto representa um ramo na árvore.

- **Recursividade**: o processo de seleção do atributo e divisão dos dados é aplicado de forma recursiva para cada subconjunto até que uma condição de parada seja atingida. As condições de parada podem incluir a pureza dos subconjuntos, um limite no número de níveis da árvore ou um critério de parada pré-definido.
- **Definição dos nós folha**: quando a construção da árvore é concluída, cada folha representa uma decisão ou um valor previsto. Dependendo do tipo de problema, os nós folha podem ser usados para classificação (por exemplo, rótulos de classe) ou regressão (por exemplo, valores numéricos).

É importante observar que as árvores de decisão podem sofrer de *overfitting*, ou seja, uma superespecialização nos dados de treinamento e, portanto, podem não generalizar bem para dados não vistos. Para mitigar isso, técnicas como poda da árvore (RASTOGI; SHIM, 2000), limite de profundidade máxima e uso de algoritmos de aprendizado *ensemble*, como *Random Forests* e *Gradient Boosting*, podem ser aplicadas.

2.2.1 *Ensembles* ou Conjuntos de Árvores de Decisão

A ideia central por trás dos *ensembles* é combinar vários modelos preditivos fracos para formar um modelo mais poderoso e geralmente mais preciso. Um modelo

preditivo fraco é um modelo que tem um desempenho ligeiramente melhor do que uma previsão aleatória, mas ainda não é capaz de realizar previsões precisas o suficiente para o problema em questão. Portanto, esses modelos podem ser simples, ter uma capacidade de generalização limitada ou ser sensíveis a variações nos dados de treinamento. Ao combinar múltiplos modelos preditivos fracos, os *ensembles* aproveitam a diversidade dos modelos individuais para superar as limitações de cada um por meio de mecanismos de votação para obter melhor qualidade no modelo do que as obtidas por qualquer abordagem de único modelo (ZHOU, 2012). Essa diversidade é obtida de diferentes maneiras, dependendo da técnica de *ensemble* utilizada.

Jurek et al. (2014) apresentam os principais tipos de *ensembles* contidos na literatura:

- ***Bagging*** (BREIMAN, 1996): é uma técnica baseada no particionamento do conjunto de dados de treinamento em várias amostras por meio da reamostragem sobre o conjunto de dados original. Cada amostra é usada para treinar uma árvore de decisão individual, e as previsões de todas as árvores são combinadas pela média, no caso da regressão, ou pela votação da maioria, no caso da classificação, para obter a resposta final. É uma técnica onde os modelos podem ser construídos em paralelo e que apresenta baixa diversidade de classificadores de base, o que não melhora modelos estáveis.

- ***AdaBoost*** (SCHAPIRE, 1999): é uma evolução do algoritmo de *Boosting* (SCHAPIRE, 1990) que se destaca por sua abordagem adaptativa que ajusta os pesos das instâncias durante o treinamento. Portanto, ela gera um peso maior para as instâncias que foram classificadas incorretamente pelos modelos anteriores, concentrando-se nas instâncias mais difíceis de se classificar. Produz alterações maiores nos dados de treinamento em comparação ao *bagging*, o que resulta em maior diversidade entre os classificadores base, que podem ser aplicados com modelos instáveis. Porém, esses modelos não podem ser construídos em paralelo.

- ***Random Forest*** (BREIMAN, 2001): é uma extensão ao *bagging* que introduz aleatoriedades durante a construção de cada árvore, selecionando um subconjunto aleatório de recursos (atributos) em cada divisão. Essa aleatoriedade visa aumentar a diversidade entre as árvores e reduzir a correlação entre elas, permitindo melhorar a precisão, robustez e a capacidade de generalização do ensemble.

- ***Gradient Boosting*** (FRIEDMAN, 2001): é uma técnica que pertence à família de algoritmos de *boosting*. Entretanto, diferentemente do *AdaBoost*, ela utiliza

uma abordagem baseada em gradientes para ajustar sequencialmente os modelos do *ensemble*, construindo modelos em série, onde cada modelo é treinado para corrigir os erros residuais cometidos pelos modelos anteriores e utiliza uma função de perda diferenciável para calcular os gradientes dos erros residuais.

2.3 XGBOOST - *EXTREME GRADIENT BOOSTING*

O XGBoost é uma biblioteca proposta por Chen et al. (2015) que utiliza a estratégia de *Gradient Boosting* para a construção de conjuntos de árvores de decisão (CHEN; GUESTRIN, 2016b). Ela apresenta resultados consistentes de desempenho iguais ou superiores em termos de precisão e velocidade em comparação com outras abordagens ensembles presentes na literatura (GOHIYA; LOHIYA; PATIDAR, 2018).

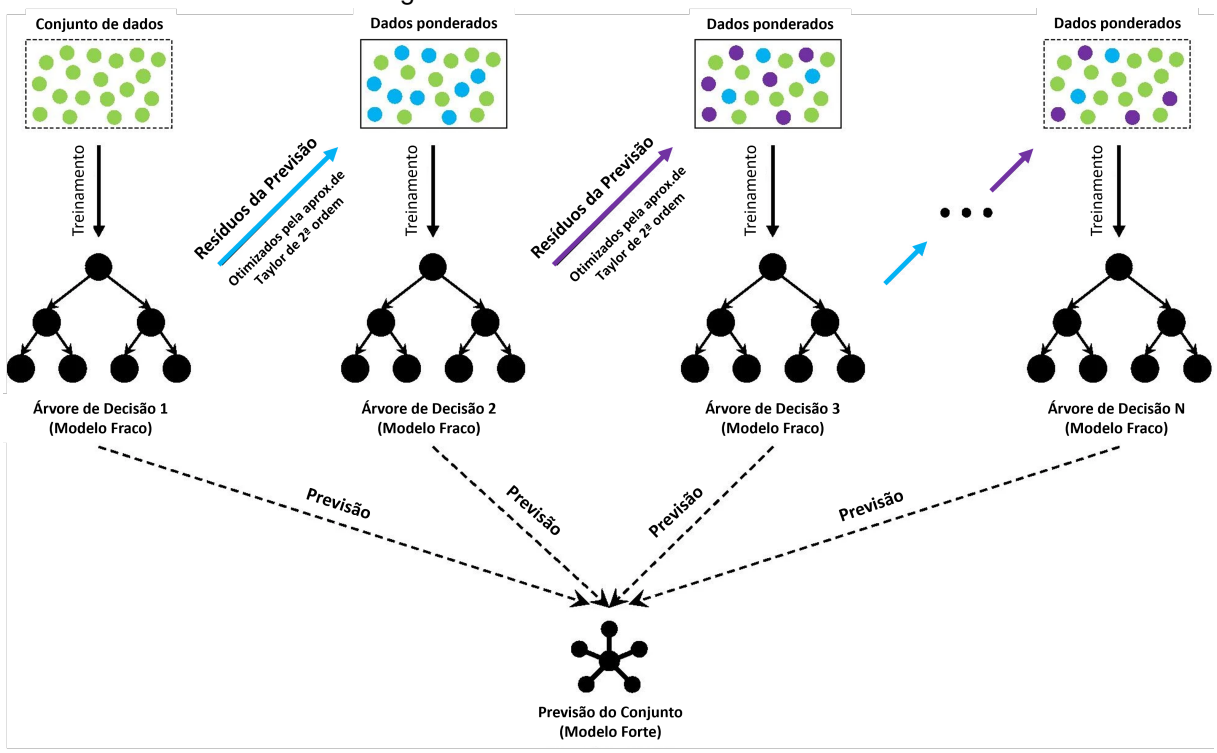
O funcionamento do XGBoost ocorre em um processo iterativo. Inicialmente, é treina uma árvore de decisão e em seguida são calculados os resíduos entre as previsões iniciais e os valores reais do conjunto de treinamento. Então, uma nova árvore é treinada e adicionada ao modelo para capturar os padrões nos resíduos restantes. Essas etapas são repetidas várias vezes, onde cada nova árvore é ajustada para corrigir os erros cometidos pelas árvores anteriores, conforme ilustrado na Figura 4. Durante o processo de treinamento, os pesos das amostras de treinamento são aumentados com base na dificuldade em prevê-los corretamente, portanto, atribuindo mais peso às amostras mal classificadas.

Além disso, o XGBoost utiliza técnicas de regularização, como a restrição do tamanho das árvores e a aplicação de penalidades nas funções de perda, para evitar o *overfitting* e aumentar a generalização do modelo. Durante a etapa de previsão, as previsões de todas as árvores são combinadas para obter a previsão final. O XGBoost também oferece recursos avançados como o tratamento de dados ausentes e a capacidade de lidar com conjuntos de dados esparsos, geralmente superando outras técnicas de *boosting* em termos de velocidade e eficiência pelas seguintes razões:

- **Processamento Paralelo**: foi projetado para explorar as capacidades de multiprocessadores, permitindo que ele execute mais rapidamente por meio do cálculo paralelo. Essa característica acelera significativamente o processo de treinamento em comparação com o *AdaBoost*, que normalmente processa sequencialmente e não suporta nativamente execução paralela (WEN et al., 2019);
- **Otimização de Hardware**: o XGBoost pode ser otimizado para infraestruturas de hardware e nuvem, tornando-o mais adaptável a vários ambientes computacionais. Essa adaptabilidade inclui uma melhor utilização dos recursos de hardware, o que contribui para sua vantagem de velocidade (JIANG et al., 2017);

- **Melhorias Algorítmicas**: incorpora várias melhorias algorítmicas sobre o *AdaBoost*, como poda de árvores mais eficiente, tratamento de dados ausentes e fornecimento de um mecanismo robusto para lidar com *overfitting* por meio de técnicas de regularização. Essas melhorias não apenas tornam o XGBoost mais rápido, mas também mais eficaz em muitos cenários (FERRARIO; HäMMERLI, 2019);

Figura 4 – Funcionamento do XGBoost



Fonte: Adaptado de (DENG et al., 2021)

2.4 FLUXO DE DADOS

O fluxo de dados, também conhecido como *data streaming*, são fontes ininterruptas de dados que podem variar a distribuição estatística dos valores dos seu atributos e que precisam ser processadas a fim de identificar padrões relevantes para a tomada de decisão. Portanto, diferentemente das abordagens tradicionais de processamento de dados, em que os dados são armazenados primeiro para posterior análise, o fluxo de dados envolve o processamento imediato de dados à medida que são gerados. Isso permite a obtenção de informações em tempo real, possibilitando ações rápidas e decisões em tempo hábil.

Alguns exemplos de aplicações de análise de fontes de fluxos de dados são: Segurança de fronteira usando sensores, monitoramento automático de câmeras de vigilância, rastreamento de dispositivos de internet das coisas (IoT), rastreamento de

pacientes em tempo real, análise do mercado de ações, detecção de intrusão de rede e sistemas de previsão de terremotos (YAHYAOUI et al., 2021).

Agrahari e Singh (2022) definiram “fluxo de dados” como qualquer sequência de dados geradas ao longo do tempo, que contém um vasto volume de instâncias variando em diferentes níveis de velocidade. Além de serem geradas de diferentes fontes, possuem as seguintes características:

- Tamanho ilimitado;
- Alta frequência de geração de dados e velocidade variável;
- Podem mudar ao longo do tempo (mudança de conceito);
- O processamento de dados tem restrições de tempo e/ou memória.

Portanto, um fluxo de dados S é uma sequência ordenada e ilimitada de instâncias $S = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N^t, y_N^t)\}$ onde (x_i^t, y_i^t) é a i-ésima instância de dado, que é um vetor de características d-dimensional, e N representa o número de instâncias disponíveis (até o momento t) e que tende ao infinito. O fluxo de dados difere das fontes tradicionais de dados de treinamento pois, na maioria dos casos, nem todas as instâncias possuem classe e nem todas as classes são previamente definidas (ZUBAROĞLU; ATALAY, 2021).

Dentro de um fluxo de dados pode ocorrer um fenômeno chamado mudança de conceito. Ele se refere à mudança na distribuição dos valores dos atributos ao longo do tempo, ou quando as relações entre as variáveis de entrada e saída evoluem ao longo do tempo, levando a uma degradação no desempenho dos modelos treinados inicialmente. É uma situação comum e desafiadora, e pode gerar modelos menos assertivos ou até mesmo irrelevantes para novos dados (DITZLER et al., 2015).

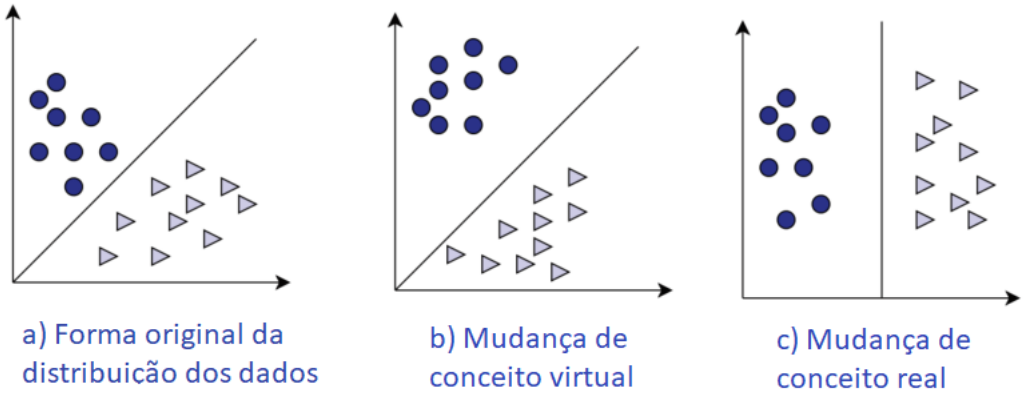
2.4.1 Mudança de Conceito - *Concept Drift*

O “*concept drift*” se refere à mudança na distribuição dos dados ao longo do tempo, o que leva a uma alteração no relacionamento entre as variáveis e, consequentemente, nos conceitos subjacentes. Em outras palavras, os modelos treinados em um determinado momento podem se tornar menos eficazes à medida que os dados evoluem e o conceito subjacente muda. Isso pode ocorrer em várias áreas, como aprendizado de máquina, mineração de dados e reconhecimento de padrões (WEBB et al., 2016).

As mudanças de conceito em fluxos de dados podem variar em termos de velocidade, gravidade e distribuição. Identificar as abordagens de detecção mais adequadas para cada ambiente de aplicação específico de classificação de fluxo é cru-

cial. Essas abordagens devem considerar características como a presença de dados rotulados ou não rotulados, a velocidade da mudança, a ocorrência de mudanças repetidas ou não repetidas, entre outros fatores. Devido a essa diversidade de cenários, é desafiador estabelecer uma estratégia de propósito geral para lidar com a mudança de conceito em fluxos de dados. Cada contexto requer uma análise cuidadosa e a seleção de abordagens de detecção apropriadas (WEBB et al., 2016). Em complemento, Agrahari e Singh (2022) classificaram os diferentes tipos de mudança de conceito e velocidade da mudança de conceito conforme ilustrado nas Figuras 5 e 6, respectivamente. Os tipos de mudança de conceito são:

Figura 5 – Ilustração dos tipos de mudança de conceito.



Fonte: Adaptado de Agrahari e Singh (2022).

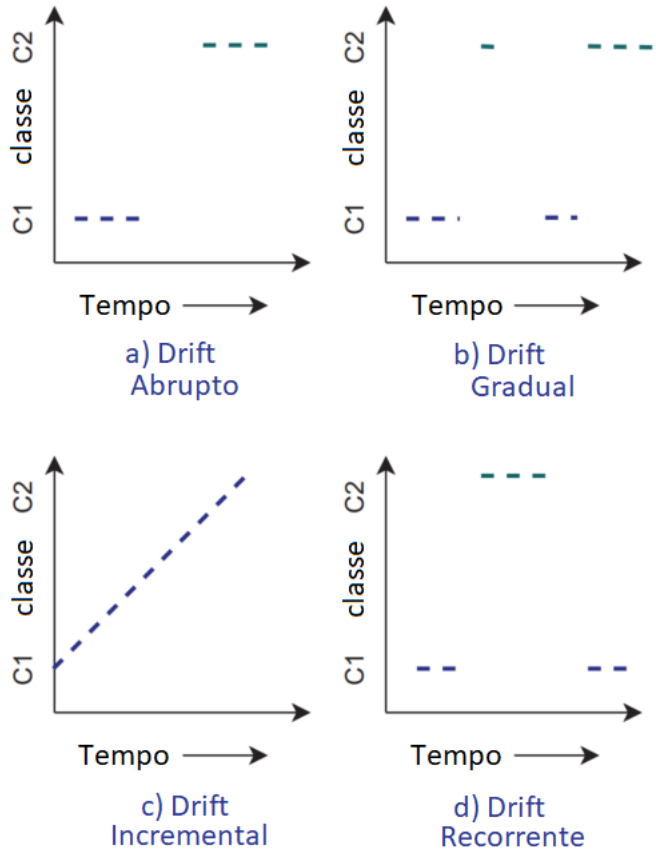
- **Virtual**: na mudança de conceito virtual, apesar do limite de decisão permanece o mesmo, a distribuição de dados dentro da mesma classe é afetada. Isso significa que as características dos dados mudam, mas a forma como essas características se relacionam com a saída não muda. Por exemplo: um modelo de previsão meteorológica, um mudança de conceito virtual pode ocorrer com mudanças sazonais nas condições climáticas, como temperaturas e padrões de precipitação que mudam com as estações do ano. Embora as condições meteorológicas se modifiquem, a relação entre essas condições e a ocorrência de um determinado evento climático, como chuva, não necessariamente muda (ver Figura 5b).

- **Real**: a mudança de conceito real representa uma mudança na classe de destino utilizando os mesmos valores das características apresentados anteriormente. Portanto, os limites de decisão são afetados e, com isso, diminui o desempenho do modelo. Em um sistema de detecção de fraudes, um mudança de conceito real pode ocorrer quando os fraudadores mudam suas estratégias e, portanto, a forma como os indicadores de fraude se relacionam com as transações fraudulentas realmente muda. Por exemplo, se inicialmente a fraude estava fortemente

associada a transações de alto valor feitas à noite, mas os fraudadores mudam de tática e começam a fazer várias pequenas transações durante o dia, o modelo precisa adaptar-se a essa nova realidade (ver Figura 5c).

Quanto a velocidade de mudança de conceito os fluxos podem ser categorizados como:

Figura 6 – Ilustração dos tipos de mudança de conceito em termos de velocidade.



Fonte: Adaptado de Agrahari e Singh (2022).

- **Abrupto**: acontece quando um novo conceito substitui repentinamente um conceito existente. Nestes caso, é fundamental a utilização de uma técnica de detecção de mudança de conceito capaz de identificar quando a precisão do classificador é degradada repentinamente e, a partir desse ponto, fazer com que o modelo aprenda com as características do novo conceito e, consequentemente, as mudanças sejam absorvidas (ver Figura 6a).
- **Gradual**: acontece quando a duração da mudança de conceito é relativamente longa em comparação com a mudança abrupta. Por exemplo, uma mudança gradual é vista quando os fenômenos do mercado mudam devido à inflação ou à recessão. Este tipo de mudança apresenta conceitos sobrepostos que, após um

determinado tempo, é substituído integralmente pelo novo conceito (ver Figura 6b).

- **Incremental**: a mudança também acontece gradualmente, entretanto, neste caso, o conceito vai se transformando ao longo do tempo sem apresentar um período de transição claramente visível (ver Figura6c).
- **Recorrente**: neste tipo de mudança de conceito, um conceito existente reaparece após um período de tempo. Portanto, ele apresenta um comportamento cíclico e acíclico. O fenômeno cíclico se aplica em uma situação onde ocorrem variações sazonais. Por exemplo, a venda de sorvete aumenta devido ao verão. Já o fenômeno acíclico pode ser visto no exemplo onde o preço da eletricidade aumenta devido ao aumento do preço da gasolina, mas quanto o preço da gasolina diminui a eletricidade volta ao preço anterior (ver Figura6d). É importante mencionar que a mudança recorrente de conceito pode ocorrer de forma Abrupta, Gradual ou Incremental.

2.4.2 Desbalanceamento de classes

O problema do desbalanceamento de classes em tarefas de classificação apresenta um desafio significativo, onde a prevalência de uma classe sobre outras pode levar a um viés nos modelos de aprendizado de máquina, prejudicando sua generalização e acurácia. Em cenários de fluxo de dados, este problema pode ser ainda mais exacerbado devido à natureza dinâmica e evolutiva dos dados, onde as distribuições das classes podem variar significativamente ao longo do tempo e não é possível construir um conjunto de treinamento balanceado *a priori* (SILVA et al., 2021). O desbalanceamento de classes ocorre quando algumas das classes presentes no conjunto de dados tem menor representatividade numérica em relação a outras. Isso resulta em uma dificuldade para o algoritmo de aprendizado em reconhecer os padrões das classes minoritárias, levando frequentemente a um desempenho inadequado.

No contexto de fluxos de dados, o desbalanceamento de classes pode aumentar à medida que novos dados chegam pelo fluxo. Portanto, o fenômeno do desbalanceamento de classes em fluxos de dados não estacionários desafia ainda mais os algoritmos de treinamento devido ao processo de adaptação contínua do modelo realizado por eles. Técnicas como re-amostragem dinâmica ou ajuste de pesos das classes tornam-se complexos de se implementar sem incorrer em perda de informação ou sobrecarga computacional (GAMA et al., 2004). Além disso, a natureza incremental do aprendizado em fluxos de dados exige que os modelos sejam capazes de aprender com quantidades limitadas de dados disponíveis para cada classe em momentos específicos, dificultando ainda mais a atualização do um modelo.

Uma das formas de mensurar o desbalanceamento dos dados é utilizando a razão de prevalência (PR), ou razão de desbalanceamento. Uma medida de associação frequentemente utilizada em estudos de classificação para quantificar a relação entre diferentes classes. Diferente da razão de chances (*odds ratio*), que pode ser menos intuitiva, a razão de prevalência oferece uma interpretação mais direta e compreensível para a maioria das pessoas. De acordo com Deddens e Petersen (2008), a PR é preferível à razão de chances em estudos de classificação com resultados comuns, pois indica quantas vezes mais provável é a ocorrência de uma determinada classe em comparação com outra. Thompson, Myers e Kriebel (1998) destacam que a PR é mais conservadora e consistente, sendo também mais interpretável. A vantagem principal da razão de prevalência reside em sua capacidade de fornecer uma medida direta e intuitiva de comparação entre classes, facilitando a interpretação dos resultados por analistas, pesquisadores e o público em geral. A razão de prevalência pode ser calculada conforme fórmula a seguir:

$$RP = \frac{\text{frequência da classe majoritária}}{\text{frequência da classe minoritária}} \tag{2.7}$$

2.4.3 Dados Rotulados e Não Rotulados

A coleta de grandes volumes de dados rotulados continua sendo uma barreira fundamental na realização do pleno potencial dos algoritmos de AM. Essa tarefa é demorada, custosa e, em certos casos, inviável, especialmente em situações que envolvam condições ou eventos raros (CHEN et al., 2021). Para superar esses desafios, a literatura apresenta uma variedade de algoritmos de aprendizado semi-supervisionado (SSL), que buscam complementar um conjunto limitado de dados rotulados disponíveis com um conjunto maior de exemplos não rotulados durante o treinamento. Essa abordagem visa mitigar os problemas associados à falta de dados rotulados, permitindo que o aprendizado seja realizado de maneira mais eficiente e eficaz (BANITALEBI-DEHKORDI; GUJJAR; ZHANG, 2022).

Em cenários com poucos dados rotulados, os métodos de classificação semi-supervisionados são especialmente úteis. Em tais casos, construir um classificador supervisionado confiável pode ser um desafio significativo. Essa situação é frequentemente observada na maioria dos domínios de aplicação nos quais a obtenção de dados rotulados é dispendiosa ou de difícil acesso. No entanto, quando há um número suficiente de dados não rotulados disponíveis e determinadas suposições podem ser feitas sobre a distribuição desses dados, é possível utilizar esses dados não rotulados para auxiliar no aprimoramento do treinamento do modelo. Na verdade, os métodos de aprendizado semi-supervisionados têm sido aplicados mesmo em situações nas quais não há uma escassez significativa de dados rotulados, pois os dados não rotulados podem fornecer informações adicionais relevantes para o processo de treinamento,

e serem aproveitados para melhorar o desempenho do modelo (ENGELEN; HOOS, 2020).

O aproveitamento desses dados pode trazer benefícios significativos para os negócios existentes, permitindo uma tomada de decisão mais precisa e em tempo real. Exemplos desses benefícios incluem o fornecimento de *insights* valiosos sobre o comportamento do cliente, padrões de compra, monitoramentos, tendências de mercado, segurança da informação entre outros. A análise em tempo real desses dados pode ser utilizada para personalizar produtos, otimizar processos, melhorar a experiência do cliente e impulsionar a inovação, ou ainda pode revelar novos padrões, tendências emergentes e necessidades do mercado que antes não eram conhecidas.

2.5 TÉCNICAS DE APRENDIZADO SEMI-SUPERVISIONADO

Conforme mencionado anteriormente, o aprendizado semi-supervisionado tem por objetivo utilizar o conhecimento contigo nos dados não rotulados para melhorar o treinamento do modelo. Entretanto, existem algumas técnicas propostas na literatura para a realização desse tipo de aprendizado. Algumas das principais técnicas apresentadas na literatura são:

- *Co-Training*: É uma técnica de aprendizado semi-supervisionado proposta por Blum e Mitchell (1998). Esta técnica assume que cada instância no conjunto de dados pode ser descrita por dois conjuntos distintos de características, que são suficientemente independentes e cada subconjunto é suficiente para treinar um bom classificador. O processo envolve dois classificadores que são treinados separadamente em diferentes visões do conjunto de dados. Cada classificador, então, rotula exemplos não rotulados para o outro, o que ajuda na melhoria iterativa dos modelos.

- *Self-Training*: É um método simples de aprendizado semi-supervisionado em que um classificador inicial é treinado com um pequeno conjunto de dados rotulado. O classificador então é usado para rotular dados não rotulados, e as predições mais confiáveis são adicionadas ao conjunto de treinamento. Este ciclo é repetido várias vezes (AMINI et al., 2022).

- *Wrapper*: Uma maneira de estender algoritmos supervisionados para a tarefa de aprendizado semi-supervisionado consiste em acoplar a ele uma outra estratégia de aprendizado, seja ela realizada por um algoritmo de aprendizado supervisionado ou não supervisionado que realize a chamada pseudo-rotulagem das instâncias de dados sem rótulo. Após essa pseudo-rotulagem, os dados com maior confiança em sua rotulagem correta podem ser inseridos no conjunto de trei-

namento e, assim, tanto os dados rotulados originais quanto os dados pseudo-rotulados podem ser utilizados para treinar o modelo que efetivamente será utilizado para as futuras predições de instâncias não rotuladas (ZHU, 2008). Entretanto, antes de incluir a instância pseudo-rotulada no conjunto de treinamento é necessário avaliar o nível de certeza com que aquele rótulo foi atribuído. Para isso, são aplicadas métricas probabilísticas tais como a métrica de confiança ou de incerteza.

Uma das principais vantagens dos métodos *wrapper* é a sua aplicabilidade a quase todos os algoritmos supervisionados existentes, pois o algoritmo base supervisionado não precisa ser alterado para utilizar essa abordagem, já que este simplesmente fornece amostras pseudo-rotuladas ao algoritmo base como se fossem amostras originalmente rotuladas. Embora alguns métodos *wrapper* exijam que o modelo base forneça previsões probabilísticas, muitos outros métodos *wrapper* que dependem de múltiplos modelos base não possuem essa restrição. Para cada método *wrapper* específico as suposições subjacentes de aprendizado semi-supervisionado são dependentes dos algoritmos base utilizados (ENGELEN; HOOS, 2020). Portanto, um método *wrapper* não pode ser considerado um método de aprendizado completo por si só, ele se torna um método de aprendizado completo apenas quando combinado com um conjunto específico de algoritmos base. Uma pesquisa abrangente sobre métodos *wrapper* foi publicada por Triguero, García e Herrera (2015).

Dentro dessa abordagem, duas técnicas em particular merecem atenção: o k-vizinhos mais próximos (KNN) e sua variação o 1-vizinho mais próximo (1NN). Ambas são técnicas de classificação simples, mas poderosas, que se baseiam na proximidade dos pontos de dados no espaço de características para realizar previsões:

- **KNN**: O algoritmo *k-Nearest Neighbors* (KNN) é um método supervisionado de AM onde o valor k representa o número de vizinhos mais próximos que serão considerados para classificar ou prever um novo exemplo. O algoritmo KNN é flexível e não faz suposições sobre a distribuição dos dados, permitindo sua a adaptação a diferentes tipos de problemas (TAUNK et al., 2019). Entretanto, por ser sensível à dimensionalidade dos atributos, é necessário normalizar os valores dos atributos antes de utilizá-lo para garantir que todas as dimensões tenham uma influência equilibrada na medida de distância utilizada (ALI, 2022).

  O KNN funciona da seguinte maneira: a) Armazena um conjunto de dados de treinamento contendo exemplos previamente rotulados; b) Quando um novo exemplo precisa ser classificado, o algoritmo calcula a distância entre o novo exemplo e todos os exemplos de treinamento; c) Identifica os k exemplos de

treinamento mais próximos com base na medida de distância (geralmente a distância euclidiana); d) No caso da classificação, a classe mais frequente entre os k vizinhos mais próximos é atribuída ao novo exemplo. No caso da regressão, o valor médio (ou mediano) dos rótulos dos k vizinhos mais próximos é usado como valor de previsão para o novo exemplo.

- **1NN** : O algoritmo 1NN (primeiro vizinho mais próximo) é uma variação mais simples do KNN, em que o valor de k é igual a 1 (um). Isso significa que, para cada instância a ser predita, o algoritmo encontra o vizinho mais próximo do conjunto de treinamento e atribui o rótulo desse vizinho à instância de teste. É um algoritmo simples e intuitivo que pode ser usado para tarefas de classificação e regressão (GWEON; YU, 2021).

2.6 ALGORITMOS CLÁSSICOS

A classificação de fluxos de dados é uma área emergente e importante em AM, portanto, foram desenvolvidos ou aperfeiçoados vários algoritmos para lidar com eles. Dentre os algoritmos propostos, os que mais se destacam são: *Leverage Bagging*, *Adaptive Random Forest* (ARF), *Hoeffding Adaptive Tree* (HAT) e OzaBagging. Cada um desses algoritmos oferece abordagens únicas para melhorar a qualidade e o desempenho da classificação e a adaptabilidade para lidar com fluxos de dados dinâmicos e em constante mudança.

Leverage Bagging é uma extensão do algoritmo tradicional de *bagging*, adaptada para lidar com fluxos de dados contínuos e variáveis. Nessa abordagem, várias cópias dos classificadores de base são mantidas, e cada um deles é treinado em diferentes subconjuntos de dados de treinamento gerados por amostragem com substituição. A principal inovação em relação ao *bagging* tradicional é a introdução de pesos para as instâncias de dados, permitindo que instâncias mais recentes tenham uma influência maior no modelo, melhorando assim a capacidade do algoritmo de se adaptar a mudanças nos dados ao longo do tempo (BIFET; HOLMES; PFAHRINGER, 2010).

Adaptive Random Forest (ARF) é um método avançado para a classificação de fluxos de dados, que combina a robustez das florestas aleatórias com técnicas adaptativas para lidar com mudanças nos dados. ARF cria uma floresta de árvores de decisão, onde cada árvore é treinada com uma amostra diferente dos dados, e cada árvore pode adaptar-se de forma independente às mudanças nos dados. Além disso, o ARF implementa mecanismos de detecção de mudança e esquecimento de dados, permitindo que o modelo não apenas reconheça quando os dados estão mudando, mas também ajuste sua estrutura para manter a qualidade da classificação (GOMES et al., 2017).

Hoeffding Adaptive Tree (HAT) é uma variação do algoritmo *Hoeffding Tree* projetado especificamente para fluxos de dados. A característica distintiva do HAT é sua capacidade de adaptar-se a mudanças nos dados ao longo do tempo. Utilizando o limite de Hoeffding, o HAT decide quando dividir os nós da árvore com base em um número estatisticamente significativo de exemplos, garantindo que as decisões sejam feitas com um alto grau de confiança. Além disso, o HAT incorpora mecanismos para detectar e reagir a mudanças nos dados, como o uso de janelas deslizantes que avaliam continuamente o desempenho do modelo e ajustam a estrutura da árvore conforme necessário (BIFET; GAVALDA, 2009).

OzaBagging é uma adaptação do algoritmo de *bagging* para fluxos de dados. *OzaBagging* simula o processo de amostragem com reposição utilizado no *bagging* convencional de forma online, onde cada nova instância de dados é utilizada para atualizar os classificadores de base de maneira incremental. Essa abordagem permite que *OzaBagging* mantenha a eficiência computacional e a capacidade de adaptação a novas informações, características essenciais para a classificação de fluxos de dados (OZA; RUSSELL, 2001).

2.7 TRABALHOS RELACIONADOS

Esta seção apresenta as principais referências de trabalhos relacionados ao trabalho em questão. As pesquisas abrangem algoritmos de classificação multi-classe em fluxos de dados que empregam modelos baseados em árvores, bem como estudos que exploram o uso do XGBoost para lidar com fluxos de dados.

O IOE (*Improved Online Ensembles*), proposto por Vafaie, Viktor e Michalowski (2020), combina técnicas de conjuntos (*ensembles*) e amostragem com base nas frequências das instâncias e nas taxas de cobertura (*recall*), ao mesmo tempo em que rastreia o *recall* para cada uma das classes minoritárias para lidar com a mudança de conceito e classes desbalanceadas de dados. Além disso, aplica uma abordagem 1NN para implementar o aprendizado semi-supervisionado em instâncias não rotuladas. Se houver uma alteração na distribuição da razão de confiança, é definido um "marco" de mudança de conceito e um novo modelo é treinado com base nos novos dados. O método de auto-treinamento, que combina pseudo-rotulagem, pode ser considerado um método *wrapper*. Devido à utilização de uma técnica de treinamento de *ensemble* baseado em amostragens, ele requer bastante tempo para treinar novos modelos e atualizar o *ensemble*.

O trabalho de Masud et al. (2008) propõe uma abordagem para classificar fluxos de dados com dados rotulados limitados. O artigo apresenta um modelo com técnica de classificação de conjuntos que usa *microclusters* (micro aglomeração) para

representar o fluxo de dados em evolução e construir um conjunto de classificadores baseados em árvore de decisão usando um esquema de votação ponderada. No processo de classificação, o algoritmo KNN é aplicado para encontrar os *microclusters* mais próximos a partir de novas instâncias de dados, e a classe com a maior frequência de dados rotulados nesses *microclusters* é então escolhida como a predição final para aquela instância. A etapa de agrupamento combina dados rotulados e não rotulados para melhorar a acurácia.

Ahmadi e Beigy (2012) apresenta o SSEL, algoritmo de aprendizado semi-supervisionado para fluxos de dados. O algoritmo utiliza as instâncias rotuladas do fluxo de dados e usa um método de conjuntos (*ensemble*) para rotular as instâncias não rotuladas restantes por meio de treinamento adicional por pseudo rotulagem e re-amostragem estatística. A técnica usada é a votação por maioria, onde vários modelos estão envolvidos (*weak learners*) e cada modelo gera uma previsão para cada instância não rotulada. Isso ajuda a reduzir a taxa de erro de classificação, mesmo na presença de instâncias rotuladas de forma ruidosa, desde que haja um número suficiente de instâncias rotuladas corretamente.

Liu et al. (2013) propuseram o algoritmo SSEA (*Semi-Supervised Ensemble Approach*), que é uma abordagem de aprendizado semi-supervisionado para fluxos de dados que combina modelos supervisionados e não supervisionados para melhorar o desempenho da classificação. Ele constrói um modelo de conjunto usando agrupamento K-Means em blocos de dados rotulados, com cada classe representada por *micro-clusters*. O algoritmo divide os dados do fluxo em blocos rotulados e não rotulados. O modelo é treinado com um bloco de dados rotulados e atualizado a partir de novos dados rotulados provenientes do fluxo, enquanto os dados não rotulados são usados para construir um modelo não supervisionado. As classes são fornecidas com base em um consenso de maximização entre os modelos supervisionados e não supervisionados.

A abordagem SE-PLS, proposta por Sethi et al. (2014), para classificação binária semi-supervisionada, combina agrupamento de densidade de grade e classificação de conjunto para lidar com mudança de conceito em fluxos de dados. Ela rastreia mudanças na distribuição de dados espaciais e atualiza conjuntos locais com base na acurácia de cada *cluster* ao longo do tempo. A proposta usa uma abordagem de amostragem de grade (*Dynamic Grid Density Clustering*) para distribuir amostras rotuladas com base na densidade do fluxo de dados, permitindo um aprendizado a partir de dados parcialmente rotulados. Porém, nos resultados apresentados, não foi utilizado nenhuma taxa com dados rotulados abaixo de 10%.

SPASC (*Semi-Supervised Pool and Accuracy-Based Stream Classification*), proposto por Hosseini, Gholipour e Beigy (2016), é um algoritmo de conjunto utili-

zado para classificar instâncias de fluxos de dados não estacionários em um ambiente semi-supervisionado. Cada modelo determina um conceito específico, permitindo que o SPASC manipule conceitos recorrentes em fluxos de dados e se adapte às mudanças de conceito ao longo do tempo. Para conseguir isso, o método agrupa os dados usando um algoritmo de *clustering* e treina um classificador para cada *cluster* usando dados rotulados e não rotulados. Durante o teste, as instâncias são atribuídas ao *cluster* mais próximo sem exigir rotulagem explícita.

Montiel et al. (2020) propuseram o *Adaptive XGBoost* (AXGB), um algoritmo de classificação binária que utiliza um conjunto de modelos XGBoost como base. O AXGB adota uma abordagem em que os modelos XGBoost mais antigos são substituídos por modelos mais recentes, treinados com dados mais recentes do fluxo de entrada. O conjunto mantém um número máximo definido de modelos, que quando cheio precisa eliminar os classificadores mais antigos para treinar novos, aplicando uma estratégia de janela deslizante adaptativa. A adaptabilidade da janela deslizante é estabelecida por meio de limites mínimo e máximo e, a cada iteração de treinamento, o tamanho da janela é atualizado, aumentando exponencialmente até atingir o valor máximo. Isso permite um início rápido do treinamento, pois a janela está pequena. No entanto, à medida que o treinamento progride, a janela é expandida para reduzir a frequência de criação de novos classificadores. Para acelerar ainda mais a adaptação do modelo a novos conceitos, o detector ADWIN também foi implementado no AXGB.

Souza, Grando e Baldo (2022) apresentaram o AFXGB-Reg, uma adaptação para regressão de fluxos de dados do AXGB, que utiliza apenas dois modelos XGBoost em vez de um conjunto de modelos como abordado no AXGB. Para detecção de mudança de conceito, em vez de utilizar somente o ADWIN, foi proposto um vetor de detectores que eram utilizados para melhor identificar uma mudança de conceito no fluxo. Os detectores utilizados neste trabalho foram o ADWIN, o KSWIN e o DDM. Este trabalho mostrou uma precisão equivalente aos algoritmos estado da arte da literatura para a regressão de fluxos de dados, entretanto, com tempo de processamento consideravelmente inferior a eles.

O AFXGB-MC, proposto por Baldo et al. (2023), adapta o algoritmo proposto por Baldo et al. (2022) para a realização de classificação multiclasse. Assim, o AFXGB-MC mantém boa eficiência na velocidade de processamento e bons níveis de acurácia em comparação com outras abordagens presentes na literatura. Ele também utiliza uma estratégia de janela deslizante, cadenciando o treinamento de dois modelos XGBoost simultaneamente a cada novo dado processado, até que o modelo mais antigo comece a ficar lento demais, sendo substituído pelo mais recente. Dessa forma, o segundo modelo assume o papel de principal e um novo segundo modelo começa a ser treinado.

O LAX-Reg (GRANDO, 2023) foi inspirado no algoritmo AFXGB-Reg (SOUZA; GRANDO; BALDO, 2022), entretanto ele retirou a necessidade do treinamento de um segundo modelo XGBoost. Dessa forma, na abordagem do LAX-Reg não é mais necessária a alternância entre modelos principal e secundário, mantendo um único modelo sendo treinado e atualizado. Este ajuste simplifica e acelera o processo de treinamento do modelo, pois concentra os esforços de aprendizado e detecção de mudança de conceito em um único modelo.

2.7.1 Considerações sobre os trabalhos relacionados

A Tabela 1 sintetiza os trabalhos descritos na Seção 2.7. Nela é possível observar que a maioria das abordagens para classificação semi-supervisionada tem usado principalmente métodos baseados em conjuntos (*ensemble*). Esses métodos podem obter melhor precisão do que modelos individuais, mas geralmente são mais lentos, pois precisam treinar vários modelos. Além disso, as técnicas de *wrapper* têm se mostrado uma abordagem comum na literatura para lidar com instâncias não rotuladas. Essas técnicas geralmente aplicam *clustering* para pseudo-rotular grupos de instâncias semelhantes não classificadas e usá-las para treinar/atualizar o modelo induzido pelo algoritmo de AM base. Além disso, o XGBoost é conhecido por seu alto índice de acurácia e baixo tempo de processamento, uma vez que é paralelizável. Portanto, sabendo que a literatura ainda não explorou a utilização do XGBoost para a classificação multi-classe semi-supervisionada, este trabalho visa propor uma estratégia que consiga explorar o potencial associado ao baixo tempo de processamento e ao alto grau de acurácia provido pelo XGBoost no problema de classificação multi-classe semi-supervisionada.

Outro ponto a ser destacado diz respeito a comparação com os trabalhos relacionados. A escolha de realizar a comparação direta apenas com o IOE (VAFAIE; VIKTOR; MICHALOWSKI, 2020) se baseou no fato dele ser o mais recente e de ter obtido os melhores resultados comparados com a literatura. Além disso, ele tem código-fonte disponível, fato importante caso seja necessário rodar experimentos comparativos em ambos no mesmo ambiente de execução, com as mesmas fontes de dados, o que facilita a comparação direta do desempenho entre eles.

Tabela 1 – Resumo dos trabalhos relacionados

Trabalho	Objetivo	Modelo Base	Classif.Semi	Sigla	Fluxo	Tem Fonte
Masud et al. (2008)	Classificação MC	*Micro Clusters*	KNN	MASUD	Sim	Não
Ahmadi e Beigy (2012)	Classificação MC	*Ensemble Voting*	*Self*	SSEL	Sim	Não
Liu et al. (2013)	Classificação MC	*K-Means*	*Self*	SSEA	Sim	Não
Sethi et al. (2014)	Classificação Binária	*Grid Clustering*	*Cluster Ensemble*	SE-PLS	Sim	Não
Hosseini, Gholipour e Beigy (2016)	Classificação MC	*Clustering*	*Bayesian*, *Heuristic*	SPASC	Sim	Não
Vafaie, Viktor e Michalowski (2020)	Classificação MC	*Hoeffding trees*	1NN, *Self*	IOE	Sim	Sim
Montiel et al. (2020)	Classificação Binária	XGBoost	Não	AXGB	Sim	Sim
Souza, Grando e Baldo (2022)	Regressão	XGBoost	Não	AFXGB-Reg	Sim	Sim
Grando (2023)	Regressão	XGBoost	Não	LAX-Reg	Sim	Sim
Baldo et al. (2023)	Classificação MC	XGBoost	Não	AFXGB-MC	Sim	Sim
Presente Pesquisa	Classificação MC	XGBoost	KNN, 1NN Extensível	SSAFXGB	Sim	Sim

Fonte: O autor.

3 MÉTODO PROPOSTO

Este trabalho propõe o SSAFXGB (*Semi-Supervised Adaptive Fast XGBoost for Multiclass Stream Classification*), que visa realização da classificação multi-classe semi-supervisionada de fluxo de dados não estacionários com presença de mudança de conceito. Para tal, é proposta uma adaptação ao AFXGB-MC (BALDO et al., 2023) a fim de torná-lo adapto a realização da classificação multi-classe semi-supervisionada de fluxos não estacionários parcialmente rotulados.

Na Seção 3.1 são discutidos aspectos fundamentais dessa proposta, enquanto a Seção 3.2 detalha as soluções propostas nesta pesquisa. A Seção 3.3 ilustra este estudo no formato de fluxograma, complementado pela Seção 3.4 que descreve o pseudocódigo desse fluxo. A Seção 3.5 aborda os principais aspectos sobre sua implementação. Finalmente, a Seção 3.5.1 detalha os hiperparâmetros necessários para executá-lo.

3.1 FUNDAMENTOS DA PROPOSTA

Um dos pontos chave da proposta de Baldo et al. (2023) é a abordagem para lidar com fluxos de dados não estacionários. A janela deslizante, uma técnica que envolve a análise contínua de um conjunto de dados em movimento, onde novos dados são adicionados à janela enquanto os dados mais antigos são removidos, permitindo que o algoritmo se adapte a mudanças na distribuição dos dados ao longo do tempo, facilitando a detecção de mudanças de conceito e a atualização dos modelos de classificação.

Ao trabalhar com fluxo de dados parcialmente rotulados, é importante considerar que a quantidade de dados não rotulados geralmente supera significativamente a quantidade de dados rotulados recebidos. Nesse contexto, o desafio inicial reside em aproveitar a informação implícita contida nos dados não rotulados para aprimorar o desempenho do modelo de classificação principal. A utilização eficaz desses dados não rotulados pode contribuir para a adaptação contínua do modelo às mudanças na distribuição dos dados ao longo do tempo.

Portanto, para obter vantagem na utilização dos dados não rotulados recebidos no fluxo, ao invés de descartá-los, eles são mantidos temporariamente no processo formando assim dois grupos de dados: rotulados e não rotulados. Em seguida, um processo de pseudo-rotulagem é aplicado aos dados não rotulados, utilizando um modelo treinado com os dados rotulados. Este processo atribui pseudo rótulos aos dados não rotulados com base nas previsões do modelo. Após a pseudo-rotulagem,

é necessário calcular a medida de confiança para cada rótulo atribuído. Os dados que tiverem alta confiança na pseudo-rotulagem são agregados ao conjunto de dados rotulados utilizados para a atualização do modelo principal. Já os dados com baixa confiança são descartados.

Finalmente, o modelo de classificação principal é atualizado com este conjunto de dados, que agora inclui tanto os dados originalmente rotulados quanto os dados pseudo-rotulados com alta confiança. Este processo permite que o modelo aprenda não apenas com os dados rotulados, mas também com a informação valiosa contida nos dados não rotulados, potencializando a melhoria do desempenho do modelo.

3.2 DETALHAMENTO DA SOLUÇÃO PROPOSTA

A proposta para pseudo-rotulagem dos dados sem rótulo adotada neste trabalho é baseada na técnica de *wrappers* (ver Seção 2.5), utilizando o KNN ou o 1NN como estratégias de pseudo-rotulagem. A ideia básica por trás deste enfoque consiste em aproveitar informações disponíveis nos dados rotulados para inferir pseudo-rótulos sobre instâncias não rotuladas, permitindo assim que o modelo de classificação aprenda mais rápido e com maior precisão (HIGUCHI et al., 2022). Além disso, a possibilidade de selecionar diferentes *wrappers* (KNN ou 1NN) dependendo do cenários oferece flexibilidade a cada necessidades específica.

Para evitar que o classificador base seja treinado com dados pseudo-rotulados incorretamente, é possível configurar um limite de confiabilidade da pseudo-rotulagem (*threshold*). Um meio de medir a confiança nas pseudo-rotulagens pode ser obtido por meio do cálculo da margem de incerteza (QUINONERO-CANDELA et al., 2006). Para o KNN, é utilizado o método de predição de probabilidades que lista todas as classes com suas respectivas probabilidades de acerto. Para calcular o nível de confiança na pseudo-rotulagem é calculada a diferente entre o maior nível de probabilidade e o segundo maior nível de probabilidade. Se a diferença entre eles for **maior** que o limite estabelecido pelo *threshold*, então a pseudo-rotulagem é considerada confiável e a instância é incluída no conjunto de dados que atualizará o classificador base. Para o caso do 1NN, é utilizado o método de predição direta, e posteriormente calculada a escala de distância até seu vizinho mais próximo, que deve ser **menor** que o *threshold* para que a instância seja incluída no conjunto de dados que atualizará o classificador base, caso contrário, a instância será descartada. O valor do *threshold* é definido via hiperparâmetro.

Conforme ocorre a entrada dos dados do fluxo, estes são segregados em dois *buffers*: rotulados e não rotulados. Esses *buffers* precisam ter seu crescimento gerenciado, principalmente para conter a utilização de memória e, portanto, manter

o uso racional dos recursos computacionais. A capacidade de configurar o tamanho do *buffer* de dados rotulados para a pseudo-rotulagem com o *wrapper* é uma característica que permite equilibrar a qualidade da pseudo-rotulagem e a complexidade computacional do processamento. Quando o tamanho do *buffer* é pequeno, a pseudo-rotulagem pode ser imprecisa devido à falta de diversidade nos dados. Por outro lado, um *buffer* excessivamente grande pode levar a um aumento na complexidade computacional e consumo de memória sem necessariamente melhorar a qualidade da pseudo-rotulagem (SHAHINFAR; MEEK; FALZON, 2020).

Essa proposta permite a configuração dinâmica do tamanho máximo do *buffer* calculado por meio de um índice que leva em consideração a quantidade de classes presente no fluxo. Ele também oferece três estratégias distintas de remoção para controlar o tamanho máximo, detalhadas na Seção 3.2.1. Isso possibilita uma adaptação flexível às demandas específicas de cada cenário, mas também promove uma otimização contínua do desempenho, mantendo um equilíbrio entre a qualidade da pseudo-rotulagem e a eficiência computacional.

A capacidade de detectar mudança de conceito permite ajustar o comportamento do algoritmo de aprendizado conforme os dados do fluxo evoluem. Quando a detecção de mudança está desligada, o processamento é mais rápido, mas isso pode levar a uma redução na acurácia caso existam alterações nos dados. Ao habilitar essa detecção, é possível monitorar continuamente as alterações nos dados para agilizar o aprendizado e refletir o mais rápido possível as alterações no modelo para manter a acurácia nas predições. O algoritmo ADWIN (BIFET; GAVALDA, 2007) é uma das principais referências da literatura para detecção de mudança de conceito. A abordagem adotada neste trabalho reduz o tamanho da janela deslizante para o valor mínimo configurado assim que uma mudança de conceito é detectada. Dessa forma, o processo de atualização do modelo é intensificado quando uma nova distribuição dos dados é detectada.

O desbalanceamento de classes é um problema comum na classificação de dados, ocorre quando as proporções das classes no conjunto de dados são significativamente diferentes, o que pode levar a um viés no treinamento dos modelos de classificação. Esse desequilíbrio pode ser intrínseco aos dados, como em casos onde uma classe é muito mais frequente do que outras, ou pode ocorrer devido a fatores circunstanciais, como em amostras de dados não representativas. O desbalanceamento de classes pode impactar negativamente o desempenho do modelo de classificação, tornando mais difícil identificar corretamente as classes minoritárias (SHELKE; DESHMUKH; SHANDILYA, 2017). Esse problema é ainda mais evidente em fluxos de dados parcialmente classificado, principalmente porque os dados do fluxo que possuem rótulo podem ter uma classe predominante e, portanto, tornar o treinamento do modelo

ainda mais tendencioso. Para mitigar o problema de desbalanceamento de classes, essa proposta propõe criar super-amostragem dos dados, que será melhor detalhada na Seção 3.2.2.

3.2.1 Controle do Crescimento do *Buffer* de Dados Rotulados

Para fazer o controle dos dados mantidos pelo *buffer* dos dados rotulados, foi introduzido no algoritmo proposto um hiperparâmetro que gerencia a estratégia de inclusão/exclusão dos dados rotulados para garantir que a base de treinamento não cresça descontroladamente e ao mesmo tempo se mantenha atualizada ao longo do tempo. Dada a natureza dinâmica dos fluxos de dados, foram implementadas três estratégias para gerenciar o crescimento do *buffer*:

- FIFO (*First In, First Out*): nessa abordagem os dados mais antigos são removidos conforme novos dados chegam. Isso garante a atualização, priorizando as instâncias mais recentes;
- Proporcional: esta estratégia remove as instâncias de forma proporcional à quantidade de classes de cada categoria dentro do *buffer*. Ou seja, as instâncias são removidas com base na representatividade de cada classe. Por exemplo: Um *buffer* com mil instâncias, segmentadas em três classes: A, B e C, sendo que as classes A e C possuem quatrocentos e cinquenta instâncias cada, e a classe B apenas cem instâncias. Na necessidade de excluir 10% do *buffer* serão excluídas 45 instâncias da classe A e C e apenas 10 da classe B (10% da quantidade de cada classe, dos mais antigos);
- Proporcional com Média: similar à estratégia proporcional, porém leva em consideração a média de instâncias das classes. Se a quantidade de instâncias de determinada uma classe cair abaixo da média (de instâncias por classe), a exclusão dessa é interrompida, evitando a erradicação de uma classe do *buffer*.

Essas estratégias permitem um gerenciamento eficiente do buffer dos dados rotulados, garantindo que o sistema permaneça atualizado e adaptado às mudanças nos fluxos de dados ao longo do tempo.

3.2.2 Tratamento do Desbalanceamento de Classes

O desbalanceamento de classes pode ser ainda mais agravado em cenários de fluxo de dados, onde os dados são recebidos em tempo real e de forma contínua. Nesses casos, a distribuição das classes pode mudar ao longo do tempo, o que pode levar a um desequilíbrio dinâmico entre as classes. O recebimento de um grande

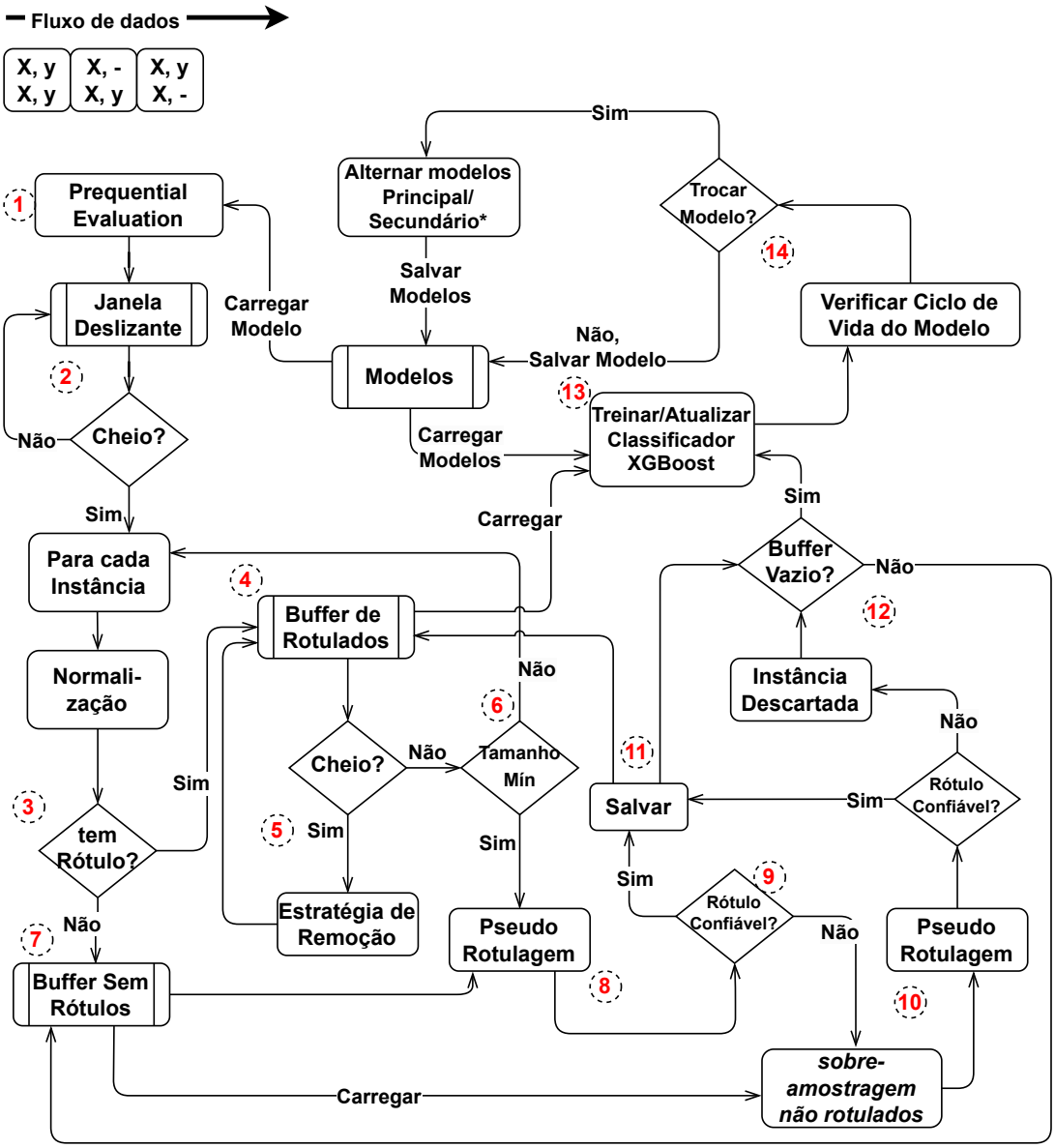
conjunto de dados de dados não rotulados (dentro de um fluxo) pode exacerbar o problema do desbalanceamento de classes. Isso ocorre porque, em muitos casos, os dados não rotulados podem conter uma seção desproporcional de exemplos de uma classe específica, potencialmente distorcendo ainda mais a distribuição das classes (AGUIAR; KRAWCZYK; CANO, 2023).

Essa proposta sugere realizar uma super-amostragem dos dados rotulados quando a confiabilidade da pseudo-rotulagem for baixa, conforme a configuração via hiperparâmetro, sendo re-submetido ao *wrapper* onde todo o processo ocorre novamente de forma recursiva, com todas as regras já existentes como de confiabilidade na pseudo-rotulagem, por exemplo, possibilitando assim uma melhora da acurácia geral da proposta. Ao gerar novas instâncias sintéticas das classes minoritárias o modelo pode aprender de forma equilibrada os padrões presentes nos dados, resultando em melhorias na acurácia. Foram propostos alguns métodos baseados em Lemaître, Nogueira e Aridas (2017) que ampliam as opções disponíveis para lidar com dados desbalanceados e permite ainda estender para outras técnicas.

3.3 FLUXOGRAMA DA SOLUÇÃO

A Figura 7 mostra uma visão geral da solução proposta. Inicialmente, é utilizada a estratégia *Evaluate Prequential* para realizar a predição das instâncias que chegam do fluxo (SKMULTIFLOW, 2020). Este ciclo se repete para cada novo item do fluxo, refletindo o desempenho do modelo em tempo real, Etapa ①. Em seguida, as instâncias são adicionadas em uma janela deslizante, Etapa ②. Quando a janela estiver cheia, as instâncias são normalizadas de acordo com o hiperparâmetro e separadas em rotuladas e não rotuladas, Etapa ③. Para os dados rotulados, Etapa ④, há um controle para gerenciar o tamanho da janela, tanto para seu tamanho máximo, Etapa ⑤, quanto para seu tamanho mínimo, Etapa ⑥, pois um conjunto com poucos dados gera uma pseudo-rotulagem com muitos falso-positivos, enquanto uma janela com muitos dados aumenta a complexidade do processamento. Em seguida, os dados rotulados são utilizados como base de treinamento para a pseudo-rotulagem dos dados não rotulados realizada pelo *wrapper*. Portanto, os dados não rotulados, Etapa ⑦, são pseudo-rotulados pelo classificador *wrapper*, Etapa ⑧, onde sua predição é avaliada para garantir que haja confiança na pseudo-rotulagem, Etapa ⑨. Caso haja baixa confiança na pseudo-rotulagem, um processo de sobre-amostragem (ou super-amostragem - *oversampling*) é aplicado sobre o conjunto de dados rotulados contidos no *buffer*, Etapa ⑩. Após a aplicação da técnica de sobre-amostragem, o conjunto de dados não rotulados que foi pseudo-rotulado com baixa confiança é submetido novamente ao *wrapper* para nova pseudo-rotulagem com o *buffer* de dados super-amostrado e a confiança dessa pseudo-rotulagem é verificado novamente. Utilizando

Figura 7 – Visão geral do processo



Fonte: O autor

o *threshold* como balizador, os dados com confiança adequada na pseudo-rotulagem são concatenados aos dados rotulados, Etapa (11), para prosseguir com o treinamento do classificador base. O processo de pseudo-rotulagem é executado até que o *buffer* de dados não rotulados seja totalmente processado, Etapa (12).

Finalizado o processo de pseudo-rotulagem dos dados recebidos na última janela do fluxo, os modelos XGBoost são atualizados com os dados rotulados que chegam pelo fluxo, incluindo os dados pseudo-rotulados pelo *wrapper*, Etapa (13). A Etapa (14) destina-se a lidar com a limitação de aumento indefinido do tamanho do modelo. Para isso, é utilizada a estratégia de treinamento e atualização incremental

de dois modelos XGBoost, um principal e um secundário. Em um primeiro momento, somente o modelo principal é treinado, a partir de determinado ponto (configurado por meio de um hiperparâmetro), o modelo secundário inicia seu treinamento e ambos são treinados com os dados que chegam pelo fluxo. Quando o modelo principal atinge o final do seu ciclo de vida, esse é substituído pelo secundário, e um novo modelo secundário começa a ser treinado.

O segundo modelo serve para controlar o uso de recursos computacionais e melhoria de performance, pois um modelo muito treinado, além de causar *overfitting* (YING, 2019), também perde eficiência, tornando-se "lento" e com alto consumo de memória. O algoritmo possui um contador que contabiliza a quantidade de vezes que o modelo foi atualizado e, a partir do momento que esse valor ultrapassa o valor definido em um hiperparâmetro específico, o modelo secundário se torna principal, e um novo modelo secundário inicia seu processo de treinamento.

3.4 ALGORITMO

A representação algorítmica do método proposto utiliza um conjunto de variáveis e notações que são listadas na Tabela 2.

Tabela 2 – Lista de variáveis utilizadas no modelo proposto

Notação	**Descrição**
M	Classificador XGBoost Principal
$M2$	Classificador XGBoost Sencundário
W	Janela deslizante
W^i	i-Ésima Instância dentro do *streaming* de dados
W^L	Janela (*Buffer*) de dados rotulados
W^U	Janela (*Buffer*) de dados **não** rotulados
ωt^L	Janela (*Buffer*) de dados super-amostrados rotulados (Temporário)
Y^P	Pseudo Rótulo
$\&$	Endereço de referência
x	Instância
y	Rótulo da Instância
$\Re$	Indicador de Recursividade
LC	Ciclo de vida
HP	Hiperparametro
$HP.sw$	HP.small_window
$HP.tpt$	HP.trees_per_train
$HP.put$	HP.percent_update_tree

O Algoritmo 1 inicia recebendo como entrada as instâncias do fluxo de dados x e y, onde x representa o conjunto de atributos e y é a categoria da classe. Essas instâncias são adicionadas à janela deslizante (linha 4) e, quando a janela estiver cheia (linha 5), o algoritmo incrementa o ciclo de vida (LC, linha 6).

Quando a janela enche, todas as instâncias contidas nela são processadas, iniciando pela separação das instâncias com rótulo em um *buffer* (variável W^L, linha 8) que são utilizadas para o treinamento do *wrapper*, sendo que o tamanho desse *buffer* está relacionado ao número de classes do fluxo de dados sendo processado (computada automaticamente multiplicando a quantidade de classes pelo hiperparâmetro `small_buffer`). Já as instâncias sem rótulo são armazenadas em um outro *buffer* (linha 10, variável W^U), e na sequência removidas da janela deslizante (linha 12).

As instâncias são normalizadas de acordo com o hiperparâmetro definido antes de realizar o processo de pseudo-rotulagem (linha 13, detalhado no Algoritmo 2, Seção 3.4.1) aplica o algoritmo KNN ou 1NN para predizer o possível rótulo das instâncias não rotuladas recebida na última janela deslizante. O algoritmo foi desenvolvido de forma a permitir que seja possível utilizar outros algoritmos não supervisionados, ou mesmo supervisionados, no processo de pseudo-rotulagem. Há ainda a possibilidade de desligar completamente o *wrapper* (hiperparâmetro `no-wrapper`), a fim de executar o algoritmo AFXGB sem a utilização da pseudo-rotulagem de instâncias sem classe.

Na sequência, é verificado o tamanho do *buffer* de dados rotulados (linha 14), e caso seu tamanho tenha excedido o limite definido pelo hiperparâmetro, é chamado o processo que realiza a exclusão de dados, detalhado na Seção 3.2.1. Na linha 17 é chamada a função `atualiza_modelos` descrita na Seção 3.4.2, Algoritmo 3.

A última etapa do processo de treinamento é a verificação de mudança de conceito. Caso o hiperparâmetro `dectect_drift` (linha 18) esteja habilitado, a rotina irá invocar a aplicação do algoritmo ADWIN (BIFET; GAVALDA, 2007) que detectará se aconteceu uma degradação acentuada da acurácia do modelo principal, indicando uma mudança de conceito (linhas 19 e 20). Se uma mudança de conceito for detectada (linha 21), o tamanho da janela deslizante que recebe os dados do fluxo é “reiniciado” (configurado para o valor mínimo, como no início do processo), para que o processo de treinamento seja acelerado, e a mudança seja mais rapidamente incorporada ao modelo (linha 22). Por fim, o modelo atualizado é retornado finalizando o treinamento (linhas 26 e 27).

3.4.1 Algoritmo de Pseudo Rotulagem

O Algoritmo 2 apresenta o processo de pseudo-rotulagem que recebe como entrada as instâncias da janela deslizante que está em processamento, os *buffer's* de dados rotulados, não rotulados, e um controle que identifica se o processo está em estado recursivo (que nesse momento é *false*) ou não.

A função inicia criando um modelo com base nos dados rotulados que acabou

Algoritmo 1 Semi-Supervised Adaptive Fast XGBoost for Multiclass Stream Classification

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

1: **função** TREINAMENTO SSAFXGB
2: $M \leftarrow M2 \leftarrow NULL$
3: $LC \leftarrow 0$
4: Adiciona (x, y) na janela (W)
5: **para cada** $W^i \in W$ **faça** ▷ Para cada instância da Janela W
6: $LC \leftarrow LC + 1$
7: **se** W^i possui rótulo (y) **então**
8: $W^L \longleftarrow$ concat(W^L, W^i) ▷ Adicionar ao buffer de rotulados
9: **senão**
10: $W^U \longleftarrow$ concat(W^U, W^i) ▷ Adicionar ao buffer de Não-rotulados
11: **fim se**
12: $W \leftarrow (W - (W^L + W^U))$ ▷ remover dados da Janela Deslizante
13: $W^L \longleftarrow$ pseudo_rotulagem$(HP, W^i, W^L, W^U, False)$
14: **se** $tamanho(W^L) > HP.sw$ **então**
15: $W^L \longleftarrow$ estratégia_remoção()
16: **fim se**
17: $M, M2 \longleftarrow$ atualiza_modelos$(M, M2, W^L, \&LC, HP)$
18: **se** $HP.detect\_drift$ == $True$ **então**
19: $self\_predicted \leftarrow not(M.predict(x)$ == $y)$
20: $ADWIN.add(self\_predicted)$ ▷ Adiciona a previsão ao ADWIN
21: **se** $ADWIN.drift\_detection$ == $True$ **então**
22: reset_window() ▷ Reset Tamanho da Janela
23: **fim se**
24: **fim se**
25: **fim para**
26: **devolve** M
27: **fim função**

de receber (linha 2), se for o *wrapper* 1NN (linha 3), executa uma predição direta (linha 4) e cria uma matriz de distâncias normalizada para cada instância presente do conjunto de treinamento (linha 5). Em seguida, faz-se o mapeamento da menor distância (linha 6), e se essa for **menor** que o limiar (*threshold*) definido como hiperparâmetro (linha 7), ou seja, há confiança na pseudo-rotulagem é alta, a instância em processamento, juntamente com seu pseudo-rótulo, é adicionada ao *buffer* de dados rotulados (linha 8) e retirada do *buffer* de dados não rotulados (linha 12). Caso não haja confiabilidade na pseudo-rotulagem (linha 10), o processo chama a função `oversample` desse mesmo algoritmo (linha 29), que verifica se o processo já está em recursividade (linha 30). Em caso negativo, esse processo executa a super amostragem (linha 31) com base na técnica que foi configurada via hiperparâmetro ($ADASYN$, ROS ou $SMOTE$), e chama a função `pseudo_rotulagem` novamente (linha 32), com os mesmos parâmetros exceto o *buffer* de dados rotulados, que agora está com super

amostragem, e o controle de recursividade igual a *true*.

Se o *wrapper* configurado via hiperparâmetro for o KNN, o algoritmo executa uma predição de probabilidades (linha 15) retornando-a em formato de matriz. Com base na matriz retornada, é calculada a diferença entre o maior nível de certeza e o segundo maior nível de certeza dentre as probabilidades das possíveis classes (linha 17). Se o resultado da diferença for **maior** que o *threshold* pré-configurado (linha 18), a instância em processamento, juntamente com seu rótulo, é adicionada ao *buffer* de dados rotulados (linha 19). Caso não haja confiabilidade na pseudo-rotulagem (linha 21), o processo chama a função `oversample` desse mesmo algoritmo (linha 29), e o processo se repete. Por fim, o *buffer* de dados rotulados é devolvido (linha 26).

3.4.2 Algoritmo de Atualização dos Modelos XGBoost

O Algoritmo 3 apresenta o processo de atualização dos modelos XGBoost que recebe como entrada os modelos principal e secundário, o *buffer* de dados rotulados e o contador de ciclo de vida do modelo principal.

A cada rodada de treinamento, tanto o modelo principal quanto o modelo secundário, cada um em seu momento específico na linha do tempo, criam novas árvores a partir das instâncias rotuladas recebidas do fluxo (e das instâncias pseudo-rotuladas). A função inicia verificando se o modelo principal é nulo (linha 1), e em caso positivo inicia o treinamento com base no *buffer* de dados rotulados (linha 3). Caso o modelo já esteja em andamento (não nulo), é verificado o se o ciclo de vida do modelo já encerrou (linha 5), e em caso positivo o modelo secundário se torna o modelo principal, o modelo secundário e o contador do ciclo de vida são reinicializados (linhas 6 a 8).

Na sequência, o modelo principal e treinado utilizando os dados do *buffer* de dados rotulados com um determinado número de novas árvores, baseado no hiperparâmetro `trees_per_train` (linha 11). O modelo secundário começa a ser treinado um pouco antes do ciclo de vida do primeiro modelo chegar ao fim, controlado via hiperparâmetro `pre_train` (linhas 12 a 16).

As linhas 19 a 21 tratam da atualização de um percentual de árvores dentro dos modelos principal e secundário, se houver configuração via hiperparâmetro. O processo de troca de modelos (principal pelo secundário) assemelha-se ao início do processo. Um novo modelo secundário será criado com as mesmas condições do início do processo.

Algoritmo 2 Pseudo Rotulagem Via *Wrappers*

Entrada: (Hiperparâmetros, Instância da Janela, *buffer* de rotulados, *buffer* de **não** rotulados, controle de recursividade)

função PSEUDO_ROTULAGEM(HP, W^i, W^L, W^U, $\Re$)
    $w\_wrapper \leftarrow$ fit_classifier(W^L)
    se $HP.wrapper$ == "$1NN$" **então**
        $rotulos \leftarrow w\_wrapper$.predict($W^U$)
        $distancias \leftarrow rotulos$.get_distances($W^U$)
        para cada $(dist, Y^P) \in distancias$ **faça**
            se $dist < HP.threshold$ **então** ▷ Há confiança na predição?
                $W^L \leftarrow$ concat(W^i, Y^P, W^L) ▷ adiciona instância e pseudo-rótulo no buffer de rotulados
            senão
                $W^L \longleftarrow$oversample(HP, W^i, W^L, W^U, $\Re$)
            fim se
            $W^U \leftarrow (W^U - W^i)$ ▷ remove a instância da Janela
        fim para
    senão
        $probabilidades \leftarrow$ w_wrapper.get_proba(W^U)
        para cada $(prob, Y^P) \in probabilidades$ **faça**
            $margem \leftarrow (prob[0] - prob[1])$ ▷ Diferença entre as duas primeiras classes mais prováveis
            se $margem > HP.threshold$ **então**
                $W^L \leftarrow$ concat(W^i, Y^P, W^L) ▷ adiciona instância e pseudo-rótulo no buffer de rotulados
            senão
                $W^L \longleftarrow$oversample(HP, W^i, W^L, W^U, $\Re$)
            fim se
            $W^U \leftarrow (W^U - W^i)$ ▷ remove a instância da Janela
        fim para
    fim se
    devolve W^L
fim função

função OVERSAMPLE(HP, W^i, W^L, W^U, $\Re$)
    se $\Re == False$ **então** ▷ *oversampling*
        $\omega t^L \leftarrow$ OverSamplingConfig(HP).fit_resample(W^L)
        $W^L \longleftarrow$ pseudo_rotulagem($HP, W^i, \omega t^L, W^U, True$)
    fim se
    devolve W^L
fim função

Algoritmo 3 Atualiza e/ou alterna modelo principal e secundário

Entrada: Modelo Principal, Modelo Secundário, *buffer* de rotulados, Ciclo de Vida, Hiper Parâmetros

função ATUALIZA_MODELOS($M, M2, W^L, LC, HP$)
  se M == $NULL$ **então**
    $M \longleftarrow$ treinar_novo_modelo(W^L)
  senão
    se $LC \geq HP.max\_buffer$ **então**
      $M \leftarrow M2$
      $M2 \leftarrow NULL$
      $LC \leftarrow 0$
    fim se
    $M \longleftarrow M$.treinar_novas_arvores($W^L, HP.tpt$)
    se $LC \geq (HP.max\_buffer - HP.pre\_train)$ **então**
      se $M2$ == $NULL$ **então**
        $M2 \longleftarrow$ treinar_novo_modelo(W^L)
      senão
        $M2 \longleftarrow M2$.treinar_novas_arvores($W^L, HP.tpt$)
      fim se
    fim se
    se $HP.update$ == $True$ **então**
      $M1$.atualizar_percent_arvores($W^L, HP.put$)
      $M2$.atualizar_percent_arvores($W^L, HP.put$)
    fim se
  fim se
  devolve $(M, M2)$
fim função

3.5 IMPLEMENTAÇÃO DA TECNOLOGIA

Para o desenvolvimento dessa proposto, foi selecionada a linguagem de programação Python 3 devido à sua ampla disponibilidade de pacotes de aprendizado de máquina e a facilidade de implementação de algoritmos nessa linguagem. Foram utilizadas diversas bibliotecas essenciais para o desenvolvimento, incluindo:

- **xgboost** (XGBOOST, 2023): técnica de aprendizado de máquina usada como base para a implementação do algoritmo proposto. Para o treinamento utilizou-se a função `train`, em que o objetivo foi configurado como `multi:softmax` e métrica de avaliação `mlogloss`. Na etapa de atualização das árvores do modelo também utiliza-se a função `train`, porém alterando os parâmetros de processo para `update` e atualizador para `refresh`. (CHEN; GUESTRIN, 2016b);
- **numpy** (NUMPY, 2021): pacote de funções auxiliares de tratamento de vetores (criar, excluir, concatenar) (OLIPHANT et al., 2006);

- **scikit-multiflow** (MULTIFLOW, 2020; BUITINCK et al., 2013): arquitetura principal (ARCHITECTURE, 2020) da aplicação com Classes básicas “core” como `BaseSKMObject` e `ClassifierMixin`, além de possuir detector de mudança de conceito ADWIN e prover estatísticas de performance via `Evaluate Prequential`.

A implementação do SSAFXGB permite múltiplas combinações de hiperparâmetros detalhado na seção 3.5.1. Para facilitar os testes, todos os parâmetros podem ser facilmente repassados via linha de comando, permitindo ainda customizações. Além disso, todas as bases de dados podem ser facilmente adicionadas no arquivo de configurações `dataset.json`, sendo que os tratamentos para leitura são realizados pelo `dataset.py`, e fica disponível imediatamente para uso via linha de comando, onde as opções de uso são consultadas e validadas.

Essa proposta foi desenvolvida de maneira incremental, passando por várias etapas de aprimoramento, de maneira a estabelecer uma estrutura padrão de implementação como um *framework*, utilizando várias técnicas “prontas” de abordagens a fim de otimizar os resultados. Em cada etapa foram analisados os potenciais de cada técnica com o objetivo de otimizar o modelo final, especialmente em relação à acurácia, boa generalização, velocidade de processamento e capacidade de personalização (deste *framework*, via configuração). Durante o desenvolvimento foram conduzidos testes abrangentes para avaliar o impacto das melhorias implementadas e os efeitos positivos e negativos a cada alteração dos hiperparâmetros.

Dentro do *framework* desenvolvido foi criada uma estrutura modular e extensível para a implementação de *wrappers* de pseudo-rotulagem. Atualmente, estão implementados os algoritmos KNN e 1NN, integrantes do pacote sklearn API (BUITINCK et al., 2013), além da opção de não utilizar nenhum algoritmo (*no_wrapper*). Essa modularização permite a incorporação de qualquer nova técnica de pseudo-rotulagem, desde que respeite a interface de implementação definida. Isso oferece uma grande flexibilidade, permitindo que o framework se adapte e evolua de acordo com as necessidades e avanços na área de aprendizado semi-supervisionado.

A mesma estratégia modular e extensível é aplicada para a implementação de técnicas de super amostragem. Atualmente, estão implementadas as técnicas *Random OverSampling (ROS) (MENARDI; TORELLI, 2014), Synthetic Minority Oversampling Technique (SMOTE) (CHAWLA et al., 2002) e Adaptive Synthetic (ADASYN) (HE et al., 2008)* todas baseadas em Lemaître, Nogueira e Aridas (2017), além da opção de não utilizar nenhuma técnica. Novas técnicas de *oversampling* podem ser facilmente incorporadas ao *framework*, seguindo a mesma interface de implementação. Isso permite que o *framework* seja capaz de lidar com o desbalanceamento de

classes de maneira eficaz, utilizando as técnicas mais adequadas para cada cenário específico.

Para facilitar a execução de testes, foi desenvolvido um “gerador de cenários”, onde são adicionados os parâmetros com suas variações que se pretende executar, e este algoritmo gera um *shell script* com todas as combinações possíveis dos parâmetros e opções repassadas. Esse *script* é gerado com “marcas” de execução (*tags*, ou *markdowns*) para caso a execução venha a ser interrompida o processo possa ser reiniciado do ponto onde parou, ou mesmo para identificar problemas em determinado cenário. E na mesma linha da automação, também foram criados leitores de resultado, para não somente coletar as estatísticas das execuções como também gerar estatísticas das execuções.

3.5.1 Hiperparâmetros

O XGBoost apresenta uma gama de hiper parâmetros, permitindo a configuração para alcançar uma precisão aprimorada do modelo, adaptando-se aos diferentes cenários de uso. A solução proposta nesta pesquisa, SSAFXGB, utiliza alguns dos hiperparâmetros do próprio XGBoost, também herda alguns de Baldo et al. (2023) e acrescenta outros próprios, listados e explicados a seguir:

- Do XGBoost:
  - **Taxa de aprendizado (**$learning\_rate$**)**: o XGBoost usa a técnica de *boosting*, que compreende a criação de árvores iterativamente, sendo que a nova árvore tenta minimizar o erro da árvore anterior. Portanto, a taxa de aprendizagem define quão intensa é a correção aplicada. Ou seja, para valores próximos a 0 os ajustes são pequenos e para próximos a 1 são grandes.
  - **Profundidade máxima (**max_depth**)**: define a profundidade máxima das árvores do modelo. Um valor muito alto implica em um modelo complexo e, possivelmente, superespecializado, além de ocupar mais espaço de memória.
  - **Árvores criadas por rodada de treinamento (**$trees\_per\_train$**)**: equivalente ao hiperparâmetro “`num_boost_round`”, refere-se ao número de árvores novas que serão criadas a cada rodada de treinamento. Este valor reflete a velocidade com que o conjunto irá crescer até atingir o número máximo de árvores do conjunto. Sua definição padrão é 1.
- Herdados de Baldo et al. (2023):
  - **Pré-treinamento (**pre_train**)**: utilizado para controlar o início do treinamento do segundo modelo XGBoost. Combinado com o parâmetro

`max_buffer` menos `pre_train` indica o momento em que o *framework* deve iniciar o treinamento do segundo modelo. Tipo inteiro, aceita valores de 1 até o máximo suportado pelo tipo int.

  - **Máximo de treinamentos (**max_buffer**)**: número máximo de treinamentos do conjunto XGBoost. Um modelo com uma quantidade muito grande de árvores aumenta a complexidade, resultando em perda de eficiência e um maior consumo de memória. Tipo inteiro, aceita valores de 1 até o máximo suportado pelo tipo int.
  - **Tamanho mínimo da janela (**min_window_size**)** e **Tamanho máximo da janela (**max_window_size**)**: tamanhos mínimo e máximo do *buffer* (Janela deslizante) que armazena os dados provenientes do fluxo. O tamanho da janela é dinâmico e é atualizado de acordo com esses valores, crescendo exponencialmente do valor mínimo até o valor máximo. Tipo inteiro, aceita valores de 0 até o máximo suportado pelo tipo int.
  - **Atualização das árvores (**$update$**)**: indica se o modelo deve atualizar um percentual específico de árvores durante o treinamento. É um parâmetro do tipo *booleano* e depende do parâmetro `percent_update_trees`.
  - **Porcentagem de árvores atualizadas (**$percent\_update\_trees$**)**: percentual de árvores que serão atualizadas pelo modelo durante a rodada de treinamento. É necessário que o parâmetro `update` esteja ligado. Tipo *float* que aceita o intervalo de 0,01 até 0,99.
  - **Detecção de mudança de conceito (**$detect\_drift$**)**: valor booleano que define se o método de detecção ativa de mudança de conceito ADWIN será utilizado durante o processo de treinamento ou não.

- Obrigatórios do *framework* SSAFXGB:

  - **dataset.json**: nesse arquivo estão todas as definições da base de dados (*dataset*) a ser utilizada, tais como: quantidade de classes, se for real de onde deve ser lida, e se for sintética quais as características da geração (percentual de ruído, tipo de mudança de conceito: abrupto ou gradual e segregação dos pontos de mudança). Não há definição padrão, sendo obrigatória sua configuração.
  - ***Wrapper* (**$pseudo\_labeling$**)**: indica qual *wrapper* será utilizado. É possível utilizar algum dos pré-definidos (1NN ou KNN) ou ainda "desligar" a utilização de *wrapper* (`no-wrapper`). Há ainda a possibilidade de estender (a partir da classe `wrapper.py`), desenvolvendo novos *wrappers*, pois foi criado com este propósito. Não há definição padrão, sendo obrigatória sua configuração.

  - **Hiperparâmetros do Wrapper (**$wrapp\_hyperpars$**)**: hiperparâmetros do próprio *Wrapper*, devem ser escritos no formato "dicionário" do Python. Não há valor padrão e não é obrigatório.
  - **Margem de incerteza (**$threshold$**)**: utilizado para definir o grau de confiança na pseudo-rotulagem. Para o 1NN é utilizada a distância, que deve ser menor que esse parâmetro. Para os demais é utilizado o método "proba", que é calculado a partir da diferença da primeira maior probabilidade e da segunda maior, e deve ser maior do que esse parâmetro. Tipo *float* que indica um percentual, aceita o intervalo de 0,01 até 0,99.
  - **Tamanho da janela de dados Rotulados (**$small\_buffer$**)**: esse valor não define diretamente o tamanho da janela, mas é um valor de referência, pois a composição do tamanho da janela é realizada pela multiplicação desse parâmetro pelo número de classes da base que está sendo utilizada. Tipo inteiro, aceita valores de 1 até o máximo suportado pelo tipo int.
  - **Estratégia de remoção** do *buffer* dos dados rotulados($proportional\_data$): permite definir qual método utilizado para remover os dados do *buffer* assim que atingirem o máximo, são três opções disponíveis: $FIFO$, $PROP$ e $PROPMEAN$.
  - **Método de super amostragem** (*oversampling*): parâmetro que controla qual o método de super amostragem será utilizando quando houver baixa confiança na pseudo-rotulagem. Quatro opções estão disponíveis: $ADASYN$, ROS, $SMOTE$ ou $None$.
  - **Método de normalização** (*normalize*): parâmetro que define qual o método de normalização será utilizado. Quatro opções estão disponíveis: STD `StandardScaler`, ABS `MaxAbsScaler`, $MINMAX$ `MinMaxScaler` ou $None$ (Nenhum).
  - **Permite reinicialização do *buffer*** quando houver mudança de conceito ($reset\_sb\_ondrift$): valor booleano que define se o *buffer* será reiniciado a cada detecção de mudança de conceito. Valores aceitos `true` ou `false`.

- Opcionais do *framework* SSAFXGB:

  - **Base de Dados (**$dataset$**)**: indica qual base de dados será utilizada, do tipo string, esse é um indicador simples de uma configuração mais ampla especificada no arquivo `dataset.json`.
  - **Percentual de dados não rotulados (**$ratio\_unsampled$**)**: parâmetro exclusivo para simulações e testes. Do tipo *float*, indica um percentual, aceita o intervalo de 0,01 até 0,99.

Por fim, os experimentos e comparações com os algoritmos de classificação multi-classe de fluxo de dados da literatura são descritos no Capítulo 4.

4 RESULTADOS

Este capítulo apresenta a avaliação dos resultados alcançados durante os experimentos realizados com o método proposto em comparação à literatura. Portanto, a Seção 4.1 apresenta os conjuntos de dados utilizados, bem como suas características. A Seção 4.2 descreve a metodologia de avaliação, incluindo a caracterização da sistemática de simulação, a apresentação dos algoritmos utilizados como referência para comparação e as métricas utilizadas para avaliação. Por fim, a Seção 4.3 exibe e discute os resultados alcançados pela solução proposta em comparação à literatura.

4.1 CONJUNTOS DE DADOS

A Tabela 3 apresenta as características dos conjuntos de dados utilizados. Foram utilizados conjuntos de dados reais e sintéticos. Os conjuntos sintéticos possuem os seguintes tipos de mudança: Gradual (G) e Abrupta (A). Nas mudanças graduais ocorrem alterações a cada 100.000 amostras, já nas abruptas as mudanças ocorrem a cada 10.000 amostras.

Tabela 3 – Propriedades das bases de dados.
Mudanças de conceito: Abrupta (A) e Gradual (G).

Dataset	**Instancias**	**Atrib.**	**Classes**	**Real**	**Fonte**
RBF (A)	500000	10	3	não	RBF (2020)
RBF (G)	500000	10	3	não	RBF (2020)
SEA (A)	500000	3	2	não	Street e Kim (2001)
SEA (G)	500000	3	2	não	Street e Kim (2001)
GAS	13910	128	6	sim	Vergara (2012)
KDDCUP99	494021	41	23	sim	Stolfo et al. (1999)
COVERTYPE	581012	54	7	sim	Blackard (1998)
WGSCOPE[1]	3205431	8	7	sim	Henrik et al. (2015)

Fonte: O autor.

Foram utilizados 8 conjuntos de dados, sendo 4 reais e 4 sintéticos. Os conjuntos de dados sintéticos foram gerados a partir das funções `RandomRBFGeneratorDrift` e `SEAGenerator`, pertencentes ao pacote `skmultiflow.data`.

As bases de dados Convertype (ou Forest Cover), RBF (com e sem mudança de conceito), SEA, KDDCUP(99) e GAS (Sensor) foram incluídas para realizar uma comparação direta com o estudo de Vafaie, Viktor e Michalowski (2020). A base de dados SEA incluiu-se variações de drift gradual e abrupto para avaliar o comportamento do modelo frente a diferentes tipos de drift, e a base de dados real WGSCOPE[1] foi adicionada para aumentar a diversidade na avaliação desta proposta.

[1] Watch Gyroscope

Além disso, a Tabela 4 apresenta a razão de prevalência (2.7) dos conjuntos de dados utilizados, destacando os diferentes graus de desbalanceamento entre as classes. Esta diversidade nas bases de dados proporciona uma avaliação mais robusta da capacidade do modelo. Como pode ser visto na Tabela 4, as bases de dados COVTYPE e KDDCUP(99) são significativamente desbalanceados, pois a razão de prevalência é bastante elevada.

Tabela 4 – Razão de Prevalência dos Conjuntos de Dados

Dataset	**Razão de Prevalência**
SEA(A/G)	1.00
RBF(A/G)	1.00
WGSCOPE[1]	1.24
GAS	1.83
COVTYPE	103.13
KDDCUP(99)	140,395.0

Fonte: O autor.

4.2 METODOLOGIA

A avaliação dos algoritmos foi realizada utilizando o método *Evaluate Prequential*, implementada por Montiel et al. (2018). Este método combina teste e treinamento de forma iterativa, onde cada conjunto de exemplo de dados de tamanho configurável é apresentado sequencialmente ao algoritmo para fazer sua predição e posterior atualização do modelo. As métricas de desempenho são registradas e atualizadas continuamente ao longo do processo. Para análise de desempenho de modelos de classificação, é importante utilizar métricas que forneçam uma visão abrangente e precisa da performance do modelo em diferentes aspectos (Seção 2.1.2). Portanto, neste trabalho foram escolhidas as métricas de Acurácia, *Kappa* e F1 Macro.

Para a seleção adequada dos hiperparâmetros, foram realizadas análises de sensibilidade descritas na Seção 4.2.1. Ajustes inadequados nos hiperparâmetros podem levar a uma degradação substancial na precisão e generalização do modelo (BERGSTRA; BENGIO, 2012). Portanto, essa análise permite aumentar a possibilidade de se encontrar uma configuração otimizada dos parâmetros, evitando tanto a superespecialização quanto a adaptação insuficiente do modelo.

Na sequencia, foi realizada uma comparação direta entre a abordagem proposta neste trabalho e o IOE (Seção 4.3.1) de Vafaie, Viktor e Michalowski (2020). Para tal, foram utilizadas as mesmas fontes de dados (exceto *Waveform*, que não foi localizada), utilizando a acurácia (métrica disponibilizada por Vafaie, Viktor e Michalowski (2020)), com 40%, 70% e 90% de dados não rotulados, facilitando uma avaliação comparativa do desempenho entre os algoritmos.

Por fim, conduziu-se uma comparação do método proposto neste trabalho com métodos tradicionais, não supervisionados, considerados referência pela literatura. Esta comparação teve por objetivo avaliar a eficiência do processo de pseudo-rotulagem aqui proposto, envolvendo a análise de três cenários distintos, cada um representando diferentes níveis de disponibilidade de dados rotulados: 10%, 5% e 1% de dados rotulados (90%, 95% e 99% de dados não rotulados, respectivamente, removendo os dados utilizados na análise de sensibilidade dos parâmetros). Para cada cenário, foram comparados a acurácia, o *kappa*, F1-Macro e o tempo de execução da solução proposta em relação às referências mais utilizadas na literatura, apresentados na Seção 4.3.2. A acurácia, *kappa* e F1-Macro refletem a qualidade do modelo, enquanto o tempo de execução representa eficiência computacional da abordagem proposta em lidar com conjuntos de dados de grande escala.

Para avaliar a significância estatística dos resultados obtidos foi empregado o teste de variância não paramétrica de Friedman sobre a acurácia dos algoritmos em cada cenário de experimentação. Após aplicar o teste de Friedman para rejeitar a hipótese nula de que não há diferenças significativas entre os métodos, foi aplicado um teste post-hoc de Nemenyi sobre a classificação (ranking) dos algoritmos testados para identificar quais deles tiveram as melhores médias de classificação no processamento das base de dados utilizadas.

A configuração da máquina utilizada na execução dos experimentos tem as seguintes especificações: Intel® Core™ i7-11700 (8 cores, 16 MB cache, até 4.9 GHz), 32GB (2 de 16 GB) de memória (DDR4, UDIMM, memória non-ECC), armazenamento 512 GB de M.2 Class 40 SSD e sistema operacional Ubuntu 20.04.4 LTS (CLI Server); Linux 5.4.0-113-generic.

4.2.1 Análise de Sensibilidade dos Parâmetros

Buscando maximizar o potencial do algoritmo para criar modelos que forneçam previsões com a melhor qualidade e menor custo computacional possíveis, foi realizada uma análise da sensibilidade dos hiperparâmetros. Alguns hiperparâmetros foram “herdados” da implementação original do XGBoost, exibidos na Tabela 5, e outros foram propostos no trabalho de Baldo et al. (2023), listados na Tabela 6. Baseando-se na eficácia e relevância dos resultados que esses valores demonstraram em seu contexto original, optou-se por manter os mesmos valores já utilizados. Entretanto, para os hiperparâmetros introduzidos no método proposto neste trabalho a análise de sensibilidade é apresentada na Tabela 7. Essa análise da sensibilidade visou garantir que os modelos criados sejam robustos, generalizáveis e eficientes, provendo resultados confiáveis e replicáveis.

Para cada hiperparâmetro, foi realizada uma série de testes com diferentes

Tabela 5 – Hiperparâmetros Herdados do XGBoost

Nome	Possibilidades	Definido
`learning_rate`	[0...1]	0,3
`max_depth`	[0...∞]	6
`num_boost_round`	*Integer*	1

Fonte: XGBOOST (2023).

Tabela 6 – Hiperparâmetros Herdados de Baldo et al. (2023)

Nome	Possibilidades	Definido
`max_buffer`	*Integer*	25
`max_window_size`	*Integer*	1000
`min_window_size`	*Integer*	10
`pre_train`	*Integer*	15
`trees_per_train`	*Integer*	1
`percent_update_trees`	[0...100]	1.0
`detect_drift`	[*true/false*]	*true*

Fonte: O Autor.

valores possíveis, combinados com os valores dos outros hiperparâmetros em 10% das bases de dados (que foram removidos da execução final), caracterizando assim um possível cenário de execução do algoritmo. Os resultados obtidos na execução dos cenários foram submetidos ao teste de Friedman (KO et al., 2016) para validar sua significância estatística. Porém não foi obtido nenhum valor *p-value* inferior a 0.05, ou seja, não há evidências suficientes para rejeitar a hipótese nula de que não há diferença significativa entre os valores dos parâmetros testados. Portanto, os valores utilizados foram os valores que isoladamente apresentaram melhor resultado na acurácia ou em F1 Macro.

O primeiro hiperparâmetro analisado foi o **nível de confiabilidade da pseudo-rotulagem** (*threshold*). Dentre os testes realizados, foram avaliados os valores $0,20$, $0,25$ e $0,30$, ao qual o valor foi definido $0,25$.

Para o próximo hiperparâmetro analisado, **Tamanho mínimo do *buffer*** de dados rotulados (*small buffer*), dentre os testes realizados, foram avaliados os valores 30, 100, 300 e 600. Esse tamanho é o valor que será utilizado como fator de multiplicação pela quantidade de classes, e visando um menor consumo de memória o valor 100 definido para o *preset*.

Para o hiperparâmetro seguinte, **Estratégia de remoção** de dados do *buffer* de rotulados, foram avaliados os tipos $FIFO$, $PROP$ e $PROPMEAN$. Como a estratégia $FIFO$ possui a menor complexidade computacional, esse foi o tipo definido para o *preset*.

Para o hiperparâmetro **estratégia de super-amostragem**, dentre os testes realizados, foram avaliados os tipos $ADASYN$, ROS, $SMOTE$ e $NONE$ (nenhum). O

Tabela 7 – Hiperparâmetros dessa proposta

Nome	Possibilidades	Definido
`threshold`	[0...1]	0,25
`small_buffer`	*Integer*	100
`proportional_data`	[`FIFO, PROP, PROPMEAN`]	FIFO
`oversampling`	[`NONE, ROS, SMOTE, ADASYN`]	ROS
`normalize`	[`NONE, STD, ABS, MINMAX`]	STD
`reset_sb_ondrift`	[*true/false*]	*false*

Fonte: O Autor.

tipo ROS apresentou isoladamente melhores resultados na acurácia motivando manter esse valor no *preset*.

Para o hiperparâmetro **normalização**, dentre os testes realizados, foram avaliados os tipos STD, ABS, $MINMAX$ e $NONE$ (nenhum). O tipo STD apresentou isoladamente melhores resultados na métrica F1 Macro, motivando manter esse tipo no *preset*.

A análise de sensibilidade do último hiperparâmetro é **Reinicializar *buffer*** de rotulados quando houver mudança de conceito. Dentre os testes realizados, foram avaliados os valores $true$ e $false$, porém nas bases de dados avaliadas foi percebido que não esvaziar o *buffer* (valor $false$) apresentou resultados superiores em relação à acurácia, e esse motivo esse foi o valor definido no *preset*.

4.3 RESULTADOS DOS EXPERIMENTOS

Nesta seção são apresentados e discutidos os resultados obtidos durante os testes realizados, confrontando o desempenho da solução proposta neste trabalho com o IOE de Vafaie, Viktor e Michalowski (2020). A avaliação realiza a comparação dos valores da acurácia em três cenários de dados não rotulados nas taxas de 40%, 70% e 90%. Uma segunda análise, são comparados os resultados da acurácia, *kappa*, F1-Macro e tempo de execução para os cenários de 90%, 95% e 99% de dados não rotulados com os algoritmos clássicos da literatura: *Leveraging Bagging*, *Adaptive Random Forest*, *Hoeffding Adaptive Trees* e *Oza Bagging*.

4.3.1 Comparativos com IOE

Os resultados dos experimentos na Tabela 8 mostram a acurácia para as execuções dos algoritmos comparados em experimentos conduzidos com 40%, 70% e 90% de dados não rotulados (mesmo cenário adotado por Vafaie, Viktor e Michalowski (2020)), respectivamente. A melhor acurácia para cada conjunto de dados é destacada em negrito. A notação “N/E” significa que a base de dados não foi encontrada.

De maneira geral, observa-se que o algoritmo SSAFXGB (variação 1NN,

Tabela 8 – Desempenho do IOE *vs* SSAFXGB com as referencias de Vafaie, Viktor e Michalowski (2020)

Dataset	**IOE**			**SSAFXGB**		
	40%	**70%**	**90%**	**40%**	**70%**	**90%**
Forest Cover	0.82	0.79	0.70	**0.88**	**0.84**	**0.78**
RBF	0.84	0.80	0.74	**0.88**	**0.87**	**0.86**
RBF (com *Drift*)	0.61	0.55	0.43	**0.87**	**0.86**	**0.85**
Waveform	0.81	0.79	0.74	N/E	N/E	N/E
SEA	0.64	0.64	0.60	**0.98**	**0.97**	**0.96**
KDDCup99	0.89	0.84	0.85	**0.99**	**0.99**	**0.99**
Gas Sensor	**0.84**	**0.76**	0.69	0.73	0.73	**0.72**
Acurácia média	*0.773*	*0.730*	*0.672*	***0.884***	***0.877***	***0.862***

Fonte: O Autor.

mesmo *wrapper* utilizado pelo IOE) supera o IOE em quase todos os cenários avaliados. Especificamente, o SSAFXGB alcança as melhores performances em cinco das sete bases de dados analisadas (Forest Cover, Random RBF Generator, Random RBF Generator Drift, SEA, e KDDCup99) nas três proporções de avaliação, sendo que a base de dados Waveform não foi localizada e por isso não houve dados para comparação. Apenas na base GAS Sensor, o IOE apresenta melhor desempenho em duas das três proporções de avaliação (40% e 70%), que apresentam a maior quantidade de dados rotulados.

Comparação por Base de dados:

- Covertype (Forest Cover): O SSAFXGB supera o IOE em todas as proporções de avaliação, com uma diferença significativa de aproximadamente 6% a 8% em cada cenário;
- RBF: O SSAFXGB apresenta um desempenho consistentemente superior, com uma diferença média de cerca de 4% a 12% em relação ao IOE;
- RBF (com *Drift*): O desempenho do SSAFXGB é substancialmente melhor, especialmente nas proporções de treino de 70% e 90%, onde a diferença é superior a 30%;
- Waveform: Base de dados não foi encontrada e portanto, não é possível fazer uma comparação direta;
- SEA: O SSAFXGB apresenta um desempenho superior, com uma diferença de cerca de 32% a 37% em relação ao IOE;
- KDDCup(99): O SSAFXGB tem um desempenho muito alto, mantendo uma acurácia de 98,6% nas três proporções de avaliação, enquanto o IOE varia de 84,2% a 89,4%;

- Gas Sensor: Este é a única base de dados onde o IOE apresenta melhores resultados em duas proporções de avaliação. No entanto, com 90% de dados não rotulados, o SSAFXGB marginalmente supera o IOE.

Ao analisar a variação do desempenho com diferentes proporções de dados de treino, fica evidente que o SSAFXGB mantém uma performance mais estável e elevada em comparação ao IOE. O desempenho do IOE tende a decair mais rapidamente com a redução de dados rotulados, comportamento evidenciado nos conjuntos de dados RBF (com *Drift*) e SEA.

Os resultados apresentados sugerem que o SSAFXGB é, em geral, mais robusto e eficaz que o IOE em uma variedade de cenários e bases de dados. A análise das acurácias médias reforça essa conclusão sendo a única exceção para a base de dados GAS, onde o IOE mostra um desempenho competitivo em até 70% de dados rotulados. O SSAFXGB supera o IOE com margens significativas para todas as proporções de dados rotulados, sendo 14,36% para o cenário de 40%, 20,13% em 70% e 28,27% para o cenário de 90%.

4.3.2 Comparação dos índices de qualidade com métodos clássicos

Esta seção apresenta os resultados dos experimentos comparando o desempenho do método proposto em relação aos algoritmos clássicos de classificação supervisionada encontrados na literatura, utilizando três percentuais de dados não rotulados: **90%** (Tabelas 9 e 10), **95%** (Tabelas 11 e 12) e **99%** (Tabelas 13 e 14). Os algoritmos avaliados incluem as duas versões do algoritmo proposto (SSAFXGB-1NN, SSAFXGB-KNN), e os clássicos da literatura (*Leverage Bagging*, *Adaptive Random Forest*, *Hoeffding Adaptive Tree* e *OzaBagging*). As métricas utilizadas para a comparação foram Acurácia, *Kappa*, F1 (Macro) e tempo de processamento em segundos (segs). Cada par de tabelas apresenta os resultados dos algoritmos sob uma mesma condição de dados não rotulados, facilitando a análise de desempenho entre os métodos com e sem pseudo-rotulagem. O melhor valor dentre cada métrica de cada conjunto é destacada em **negrito**. Na parte inferior de cada tabela são apresentados os *Valores Médios* de todos os conjuntos de dados para aquele cenário, destacando o ranqueamento do algoritmo para aquela métrica para todas as bases de dados sobrescrito entre parêntesis. O ranqueamento varia de (1) a (6), onde (1) representa o algoritmo com o melhor desempenho até a classificação (6) sendo o pior desempenho.

Tabela 9 – Desempenho dos algoritmos em diferentes bases, com 0,90% dados não rotulados (Parte 1).

Dataset	SSAFXGB-1NN				SSAFXGB-KNN				LeverageBagging			
	Acc	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**
covtype	**0.78**	**0.64**	**0.56**	282.35	0.74	0.58	0.55	234.62	0.44	0.03	0.41	590.63
gas	**0.72**	**0.66**	**0.61**	15.64	0.70	0.63	0.59	**11.57**	0.38	0.19	0.36	123.51
kddcup	**0.98**	**0.97**	0.33	298.34	0.96	0.93	0.30	255.46	0.61	0.00	**0.61**	875.21
rbf_abrupt	**0.86**	**0.82**	**0.86**	179.30	0.85	0.80	0.85	175.64	0.32	0.08	0.30	412.22
rbf_gradual	**0.85**	**0.81**	**0.85**	181.09	0.84	0.79	0.84	175.48	0.32	0.08	0.30	411.13
sea_abrupt	**0.97**	**0.92**	**0.97**	165.02	0.95	0.89	0.96	146.46	0.89	0.76	0.89	234.08
sea_gradual	**0.96**	**0.92**	**0.97**	166.27	0.95	0.90	0.97	144.76	0.88	0.73	0.88	233.59
watch_gyroscope	0.95	0.94	0.95	1189.28	**0.96**	**0.95**	**0.96**	1050.07	0.13	0.00	0.13	1065.63
Valores Médios	**0.88**$^{(1)}$	**0.84**$^{(1)}$	**0.76**$^{(1)}$	*309.66*$^{(3)}$	*0.87*$^{(2)}$	*0.81*$^{(2)}$	*0.75*$^{(2)}$	*274.26*$^{(2)}$	*0.50*$^{(4)}$	*0.23*$^{(6)}$	*0.49*$^{(4)}$	*493.25*$^{(5)}$

Fonte: O Autor.

Tabela 10 – Desempenho dos algoritmos em diferentes bases, com 0,90% dados não rotulados (Parte 2).

Dataset	AdaptiveRandomForest				HoeffdingAdaptiveTree				OzaBagging			
	Acc	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**
covtype	0.52	0.08	0.53	222.09	**0.35**	**0.05**	0.33	**166.19**	0.54	0.08	0.53	400.89
gas	0.39	0.19	0.37	765.16	0.40	0.18	0.45	873.24	0.56	0.33	0.57	227.84
kddcup	0.13	0.00	0.13	223.64	0.01	0.00	0.01	**42.96**	0.61	0.00	0.61	873.76
rbf_abrupt	0.34	0.11	0.32	396.79	0.31	0.07	0.30	**64.73**	0.34	0.11	0.33	377.58
rbf_gradual	0.33	0.10	0.31	409.08	0.32	0.09	0.31	**63.69**	0.34	0.11	0.33	372.58
sea_abrupt	0.90	0.78	0.89	159.05	0.90	0.77	0.89	**26.27**	0.89	0.74	0.88	193.86
sea_gradual	0.90	0.78	0.89	166.44	0.90	0.77	0.89	**25.67**	0.89	0.74	0.88	191.78
watch_gyroscope	0.16	0.00	0.17	933.54	0.16	0.00	0.16	143.24	0.13	0.00	0.13	1735.90
Valores Médios	*0.46*$^{(5)}$	*0.26*$^{(3)}$	*0.45*$^{(5)}$	*409.47*$^{(4)}$	*0.42*$^{(6)}$	*0.24*$^{(5)}$	*0.42*$^{(6)}$	**175.75**$^{(1)}$	*0.54*$^{(3)}$	*0.26*$^{(3)}$	*0.53*$^{(3)}$	*546.77*$^{(6)}$

Fonte: O Autor.

Tabela 11 – Desempenho dos algoritmos em diferentes bases, com 0,95% dados não rotulados (Parte 1).

Dataset	SSAFXGB-1NN				SSAFXGB-KNN				LeverageBagging			
	Acc	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**
covtype	**0.75**	**0.59**	**0.53**	275.73	0.70	0.51	0.51	240.50	0.44	0.02	0.46	527.31
gas	**0.70**	**0.63**	**0.59**	14.91	0.65	0.57	0.55	**11.57**	0.33	0.11	0.29	127.55
kddcup	**0.99**	**0.98**	**0.37**	295.68	0.92	0.87	0.28	258.35	0.16	0.00	0.16	690.48
rbf_abrupt	**0.86**	**0.81**	**0.86**	180.63	0.84	0.79	0.84	174.19	0.36	0.14	0.35	265.01
rbf_gradual	**0.85**	**0.80**	**0.85**	186.11	0.83	0.78	0.83	177.82	0.36	0.14	0.35	262.30
sea_abrupt	**0.96**	**0.92**	**0.97**	163.35	0.93	0.85	0.95	144.19	0.88	0.73	0.88	140.68
sea_gradual	**0.96**	**0.91**	**0.97**	166.27	0.93	0.84	0.95	144.92	0.88	0.71	0.87	139.27
watch_gyroscope	0.89	0.88	0.89	1228.48	**0.93**	**0.92**	**0.93**	1089.05	0.12	0.00	0.12	1233.03
Valores Médios	**0.86**$^{(1)}$	**0.79**$^{(1)}$	**0.74**$^{(1)}$	*343.39*$^{(3)}$	*0.83*$^{(2)}$	*0.76*$^{(2)}$	*0.72*$^{(2)}$	*294.11*$^{(2)}$	*0.43*$^{(4)}$	*0.23*$^{(4)}$	*0.42*$^{(4)}$	*343.91*$^{(4)}$

Fonte: O Autor.

Tabela 12 – Desempenho dos algoritmos em diferentes bases, com 0,95% dados não rotulados (Parte 2).

Dataset	AdaptiveRandomForest				HoeffdingAdaptiveTree				OzaBagging			
	Acc	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**
covtype	0.48	0.02	0.46	267.16	0.00	0.00	0.00	**140.96**	0.52	0.05	0.51	415.98
gas	0.32	0.09	0.28	838.19	0.25	0.08	0.27	420.13	0.45	0.25	0.44	243.14
kddcup	0.29	0.00	0.29	194.24	0.01	0.00	0.01	**149.97**	0.16	0.00	0.16	1154.89
rbf_abrupt	0.36	0.15	0.34	302.73	0.34	0.11	0.33	**58.51**	0.36	0.14	0.35	243.51
rbf_gradual	0.33	0.10	0.31	314.00	0.34	0.11	0.33	**56.61**	0.36	0.14	0.35	244.75
sea_abrupt	0.91	0.79	0.90	129.04	0.90	0.78	0.90	**23.02**	0.89	0.76	0.89	122.84
sea_gradual	0.91	0.79	0.90	133.69	0.90	0.77	0.89	**22.59**	0.89	0.76	0.89	118.88
watch_gyroscope	0.11	0.00	0.12	802.01	0.14	0.00	0.14	**98.13**	0.12	0.00	0.12	1134.61
Valores Médios	*0.43*$^{(4)}$	*0.14*$^{(6)}$	*0.41*$^{(6)}$	*441.19*$^{(5)}$	*0.43*$^{(4)}$	*0.23*$^{(4)}$	*0.42*$^{(4)}$	**143.19**$^{(1)}$	*0.46*$^{(3)}$	*0.25*$^{(3)}$	*0.45*$^{(3)}$	*444.98*$^{(6)}$

Fonte: O Autor.

Tabela 13 – Desempenho dos algoritmos em diferentes bases, com 0,99% dados não rotulados (Parte 1).

Dataset	SSAFXGB-1NN				SSAFXGB-KNN				LeverageBagging			
	Acc	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**
covtype	**0.67**	**0.46**	**0.44**	298.15	0.61	0.39	0.42	247.15	0.26	0.05	0.32	603.58
gas	**0.61**	**0.53**	**0.51**	**11.89**	0.58	0.50	0.49	13.33	0.33	0.17	0.34	121.41
kddcup	**0.98**	**0.97**	**0.32**	297.14	0.90	0.84	0.27	244.68	0.19	0.00	0.19	711.19
rbf_abrupt	**0.84**	**0.78**	**0.84**	186.13	0.81	0.75	0.81	178.63	0.37	0.15	0.37	199.33
rbf_gradual	**0.84**	**0.78**	**0.84**	186.35	0.81	0.74	0.81	178.81	0.37	0.15	0.37	145.31
sea_abrupt	**0.95**	**0.89**	**0.96**	166.31	0.74	0.41	0.81	150.67	0.88	0.73	0.88	165.57
sea_gradual	**0.95**	**0.89**	**0.96**	166.53	0.73	0.38	0.80	149.51	0.88	0.73	0.88	164.97
watch_gyroscope	**0.75**	**0.71**	**0.74**	1239.7	0.64	0.58	0.64	1206.92	0.16	0.00	0.16	1076.64
Valores Médios	**0.82**$^{(1)}$	**0.75**$^{(1)}$	**0.70**$^{(1)}$	319.02$^{(3)}$	0.73$^{(2)}$	0.57$^{(2)}$	0.63$^{(2)}$	**296.21**$^{(2)}$	0.43$^{(4)}$	0.25$^{(4)}$	0.44$^{(4)}$	398.50$^{(6)}$

Fonte: O Autor.

Tabela 14 – Desempenho dos algoritmos em diferentes bases, com 0,99% dados não rotulados (Parte 2).

Dataset	AdaptiveRandomForest				HoeffdingAdaptiveTree				OzaBagging			
	Acc	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**	**Acc**	**Kappa**	**F1**	**Segs**
covtype	0.16	0.02	0.21	372.26	0.00	0.00	0.00	**242.77**	0.32	0.08	0.38	609.08
gas	0.22	0.06	0.23	808.81	0.27	0.11	0.29	411.19	0.33	0.18	0.34	233.71
kddcup	0.17	0.00	0.17	146.22	0.03	0.00	0.04	**143.74**	0.19	0.00	0.19	913.49
rbf_abrupt	0.37	0.15	0.35	252.09	0.33	0.10	0.32	**58.61**	0.38	0.17	0.38	148.19
rbf_gradual	0.36	0.15	0.35	259.95	0.33	0.10	0.32	**56.78**	0.38	0.17	0.38	144.41
sea_abrupt	0.91	0.80	0.90	107.55	0.91	0.79	0.90	**23.52**	0.90	0.78	0.90	63.60
sea_gradual	0.91	0.79	0.90	115.94	0.91	0.80	0.91	**21.21**	0.91	0.79	0.90	61.89
watch_gyroscope	0.18	0.00	0.19	611.70	0.16	0.00	0.16	**93.90**	0.16	0.00	0.16	953.62
Valores Médios	0.41$^{(5)}$	0.25$^{(4)}$	0.41$^{(5)}$	334.32$^{(4)}$	0.37$^{(6)}$	0.24$^{(6)}$	0.37$^{(6)}$	**131.47**$^{(1)}$	0.45$^{(3)}$	0.27$^{(3)}$	0.45$^{(3)}$	391.00$^{(5)}$

Fonte: O Autor.

A diferença no desempenho entre os algoritmos é esperada, dado o uso de pseudo-rotulagem por SSAFXGB-1NN e SSAFXGB-KNN, que tende a melhorar a qualidade do modelo quando comparado com técnicas que não realizam pseudo-rotulagem. Além disso, os tempos de execução (segs) também variam conforme a complexidade dos algoritmos e a quantidade de dados, com técnicas como *LeverageBagging* e *OzaBagging* apresentando tempos significativamente maiores, o que é consistente com suas arquiteturas mais complexas e o número de árvores envolvidas.

Na análise dos resultados obtidos, observou-se um bom balanceamento entre tempo de execução e qualidade do modelo, além de uma menor tendência de queda nos indicadores avaliados dessa proposta em comparação com as abordagens existentes. Nos diferentes cenários com porcentagens de dados não rotulados, as técnicas SSAFXGB-1NN e SSAFXGB-KNN mostraram-se superiores em termos de acurácia, Kappa e F1 (Macro). Estes resultados são coerentes com a literatura que enfatiza a eficácia da pseudo-rotulagem em melhorar o desempenho de classificadores em ambientes semi-supervisionados (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).

Isso sugere uma maior capacidade de generalização da proposta em condições de dados escassos, o que é muito importante em cenários do mundo real, onde a disponibilidade de dados rotulados é limitada. A capacidade de manter um desempenho estável mesmo em condições de poucos dados rotulados pode ser atribuída à eficácia da abordagem proposta em explorar e utilizar eficientemente informações não rotuladas, conforme evidenciado em estudos como o de (LEARNING, 2006), que ressaltam o potencial de algoritmos semi-supervisionados em melhorar a generalização do modelo com poucos dados rotulados. Esses resultados sugerem que a proposta não apenas oferece vantagens em termos de tempo de execução, mas também demonstra uma robustez superior em cenários de dados limitados, promovendo avanços significativos para aplicações práticas.

Bases de dados como “covtype” e “kddcup” apresentam uma razão de prevalência extremamente alta, indicando um desbalanceamento significativo nas classes. Esse desbalanceamento afeta diretamente o desempenho dos algoritmos, especialmente aqueles que não realizam pseudo-rotulagem, como LeverageBagging e AdaptiveRandomForest, que apresentam baixa acurácia e Kappa nessas bases. Em contraste, SSAFXGB-1NN e SSAFXGB-KNN conseguem manter um bom desempenho, indicando que a pseudo-rotulagem pode ajudar a mitigar os efeitos do desbalanceamento de classes, principalmente quando associadas a técnicas de super amostragem como é o caso do método proposto.

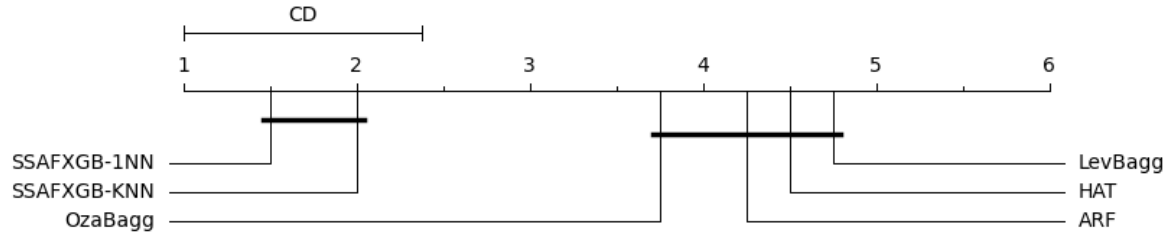
A quantidade de atributos também parece influenciar o desempenho, nas bases “gas” e “covertype” (128 e 54 atributos, respectivamente) mostrando que algoritmos como SSAFXGB-1NN e SSAFXGB-KNN são capazes de lidar melhor com

a alta dimensionalidade. Por outro lado, os tempos de execução mais elevados de SSAFXGB-1NN e SSAFXGB-KNN em bases grandes, como “watch_gyroscope”, indicam que há um *trade-off* entre acurácia e tempo de processamento. A técnica *Leverage Bagging*, apesar de eficiente em alguns casos, apresentou desempenho variado, especialmente em bases desbalanceadas como “covtype” e “watch_gyroscope. Já as técnicas ARF, HAT e OzaBagging, tiveram dificuldades em manter um desempenho consistente, particularmente em cenários com altas porcentagens de dados não rotulados.

Ao se analisar os valores médios das métricas, o melhor desempenho em termos de acurácia *kappa* e F1 foi obtido pelo SSAFXGB-**1NN**. Para confirmar essa evidência, efetuou-se uma avaliação estatística usando o teste de Friedman. Os resultados são apresentados nas Tabelas (9 e 10), (11 e 12) e (13 e 14). O teste resultou em valores-p $2,69 \times 10^{-4}$, $1,73 \times 10^{-4}$ e $7,31 \times 10^{-5}$, respectivamente. Todos esses valores de p são menores que o nível de significância de 0,05, o que rejeita a hipótese nula de que não há diferença significativa entre os grupos. Isso significa que foram encontradas diferenças significativas e, portanto, pode-se aplicar um teste *post-hoc* pareado para comparações múltiplas.

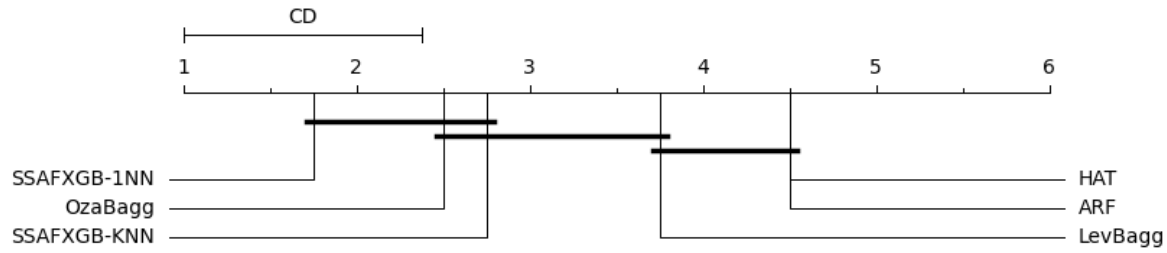
O teste *post-hoc* de Nemenyi foi realizado no *ranking* dos algoritmos para determinar se algum deles apresentou um desempenho significativamente diferente dos demais. Este teste revelou uma diferença crítica (CD) de 2,41. A comparação entre pares dos algoritmos são exibidas nas Figuras 8 (90%), 9 (95%) e 10 (99% dados não rotulados), respectivamente.

Figura 8 – Teste Nemenyi das classificações médias dos algoritmos com 90% de dados não rotulados.



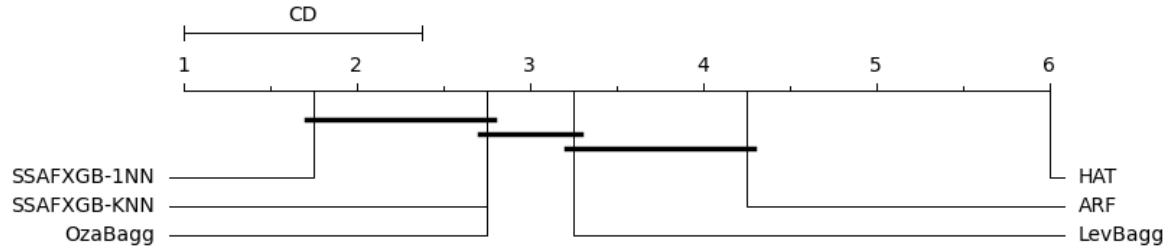
Fonte: O Autor.

Figura 9 – Teste Nemenyi das classificações médias dos algoritmos com 95% de dados não rotulados.



Fonte: O Autor.

Figura 10 – Teste Nemenyi das classificações médias dos algoritmos com 99% de dados não rotulados.

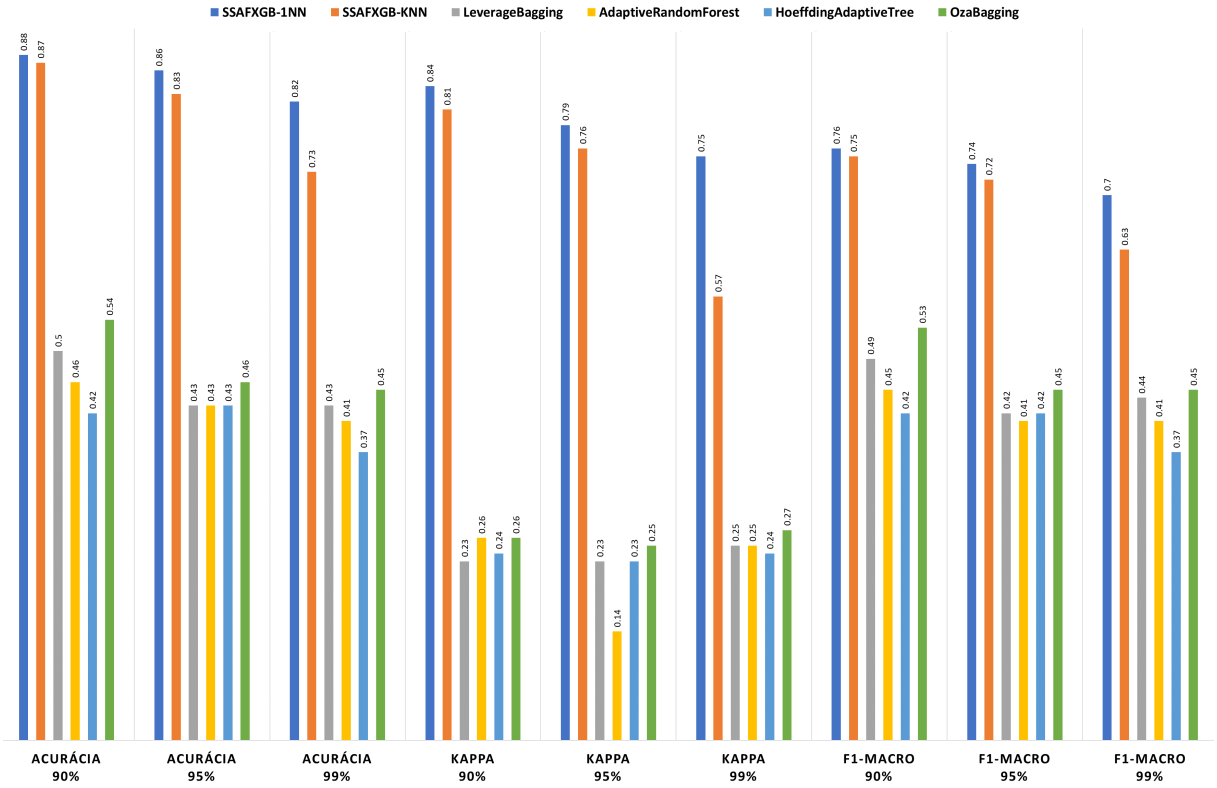


Fonte: O Autor.

A Figura 11 apresenta o gráfico gerado a partir das médias das métricas de desempenho (Acurácia, *Kappa*, F1 Macro) entre todos as bases de dados, e demostram a evolução dessas métricas conforme a porcentagem de dados não rotulados aumenta de 90% para 99%. Observa-se que os algoritmos SSAFXGB-1NN e SSAFXGB-KNN apresentam consistentemente melhores desempenhos em termos de Acurácia, *Kappa* e F1, mesmo com o aumento dos dados não rotulados. Em contraste, os algoritmos *LeverageBagging*, *AdaptiveRandomForest* e *HoeffdingAdaptiveTree* mostram desempenhos significativamente inferiores nas mesmas métricas, enquanto o *OzaBagging* apresenta uma performance intermediária. No que diz respeito ao Tempo de Execução, os algoritmos SSAFXGB tendem a ser mais rápidos do que o *LeverageBagging* e o *OzaBagging*, mas ainda mais lentos que o *HoeffdingAdaptiveTree*, que é consistentemente o mais rápido, embora no *ranking* geral sua melhor colocação tenha sido quarto, em todos os cenários. Esses resultados indicam que a presente proposta apresenta resultados mais eficientes e eficazes em cenários com alta proporção de dados não rotulados, combinando qualidade do modelo na generalização com um tempo de execução razoável.

Conforme a porcentagem de dados não rotulados aumenta de 90% para 99%, os indicadores de qualidade de todos os algoritmos tendem a diminuir. Isso é esperado, pois mais dados não rotulados podem introduzir mais incerteza no modelo. No entanto, em termos de *ranking* médio, que é uma medida de desempenho relativo en-

Figura 11 – Evolução das métricas, médias de todas as bases de dados



Fonte: O autor

tre os algoritmos, SSAFXGB-1NN consistentemente supera os outros algoritmos em todas as três situações. Isso sugere que, mesmo que a média dos indicadores de qualidade possam diminuir com o aumento de dados não rotulados, a posição relativa do SSAFXGB-1NN como o algoritmo de melhor desempenho se mantém, além de prover um tempo de resposta aceitável, considerando sua eficácia nos resultados, demonstrando sua capacidade de lidar com grandes volumes de dados.

4.3.3 Considerações sobre os resultados

Este capítulo apresentou uma análise detalhada dos resultados obtidos a partir da aplicação dos algoritmos propostos neste trabalho em comparação com os encontrados na literatura em diversos conjuntos de dados. A primeira análise foi realizada comparando a presente proposta diretamente com os resultados publicados de Vafaie, Viktor e Michalowski (2020), utilizados os mesmos cenários, com variações de 40%, 70% e 90% de dados não rotulados. A segunda análise foi realizada comparando a presente proposta com os algoritmos clássicos para classificação de fluxo de dados, e para essa comparação, foram criados novos cenários com proporções de dados, variando em 90%, 95% e 99% não rotulados.

Os resultados indicaram que a proposta SSAFXGB (em ambas as variações 1NN e KNN) apresentaram um desempenho superior em comparação com o IOE, com

média de acurácia 21% maior nos três cenários avaliados, sendo 28,3% melhor no cenário de 90% de dados não rotulados. Entretanto, em comparação com os algoritmos clássicos, a presente proposta ficou consistentemente em primeiro lugar no ranqueamento das métricas Acurácia, *Kappa*, e F1 Macro em todos os cenários elaborados. Isso sugere que as técnicas combinadas de pseudo-rotulagem, super-amostragem, balanceamento dos dados do conjunto de pseudo-rotulagem, e demais ajustes implementados nessa pesquisa, melhoram significativamente a acurácia da classificação.

Em relação ao tempo de execução, a proposta apresentou um desempenho satisfatório, posicionando-se em segundo e terceiro lugares no *ranking* (KNN e 1NN, respectivamente) em todos os cenários testados, em comparação com os algoritmos clássicos. Também foi observado que, à medida que a proporção de dados não rotulados aumenta, a qualidade média dos indicadores de todos os algoritmos tende a diminuir. No entanto, o SSAFXGB demonstrou uma menor tendência de queda na qualidade dos indicadores.

Finalmente, a análise revelou que a eficácia dos algoritmos varia dependendo das características específicas dos conjuntos de dados. Por exemplo, conjuntos de dados com mais atributos ou classes tendem a apresentar uma maior queda nos indicadores. No entanto, essa proposta demonstrou maior robustez nessas situações, mantendo uma queda relativamente mais baixa em comparação com os demais. Essas descobertas destacam a importância de considerar as características dos dados ao escolher tanto o algoritmo, quanto os hiperparâmetros adequados para uma tarefa de aprendizado semi-supervisionado.

5 CONCLUSÕES

Este trabalho propôs uma alternativa para a classificação multi-classe semi-supervisionada de fluxos de dados com presença de mudança de conceito, denominada Semi-Supervised Adaptive Fast XGBoost for Multiclass Stream Classification (SSAFXGB). A investigação baseou-se na adaptação do trabalho do (BALDO et al., 2023) com uma estratégia de pseudo-rotulagem desenvolvida por meio de um *wrapper*, que encapsula tanto a técnica de controle de confiança para a pseudo-rotulagem, maximizando o rigor na inclusão de novos dados rotulados ao conjunto de treinamento, além de uma abordagem de super amostragem, para tratar o desbalanceamento das classes presentes no fluxo de dados parcialmente rotulados.

Uma das motivações por trás dessa pesquisa é a escassez de dados rotulados encontrados nos fluxos de dados, onde sua rotulação pode ser proibitivamente cara e demorada de se obter em alguns domínios (WANG; LI; ZHOU, 2019). Além disso, a classificação semi-supervisionada pode ajudar a reduzir a dependência de dados rotulados, permitindo que modelos sejam treinados com um conjunto menor de dados rotulados.

Esta proposta evoluiu por meio de diversas etapas de aprimoramento, passando por múltiplas fases de refinamento. Foram utilizadas técnicas consolidadas de abordagens pré-existentes para otimizar os resultados, entre elas a pseudo-rotulagem encapsulada em um *wrapper* que calcula a margem de incerteza para assegurar a confiança na rotulagem obtida, a super amostragem, para sinteticamente aumentar a quantidade de dados rotulados utilizado na pseudo rotulagem. Todos estes aspectos têm por principal premissa melhorar a qualidade do modelo final gerado.

Ao analisar a acurácia do SSAFXGB em comparação com os métodos similares da literatura, observou-se um desempenho superior em termos de acurácia média principalmente em cenários onde há uma maior escassez de dados rotulados, demonstrando sua robustez. No cenário com 90% de dados não rotulados, o SSAFXGB ficou em primeiro em todos os conjuntos de dados avaliados, com acurácia média quase 20 pontos percentuais acima da média do IOE. Na comparação com os métodos clássicos para classificação supervisionada, a presente proposta ficou em primeiro lugar em todas as métricas avaliadas. Portanto, ela foi a melhor em todos os conjuntos de dados tanto na acurácia, quanto no *kappa* e na F1 Macro, para todos os cenários (90%, 95% e 99% de dados não rotulados). Ainda, o *wrapper* 1NN obteve a primeira colocação e o *wrapper* KNN a segunda, superando em quase 37 pontos percentuais (média de todos os indicadores) o classificador clássico com melhor performance no

cenário de 99% de dados não rotulados.

Apesar da superioridade do SSAFXGB em termos de qualidade do modelo, ele não obteve o melhor desempenho do que diz respeito a eficiência computacional. Os resultados revelam que o tempo de execução do SSAFXGB não foi o mais baixo, perdendo para o HAT, mas superando o *Leverage Bagging*, ARF e *Oza Bagging* (métodos clássicos da literatura), que apresentam os resultados de qualidade mais próximos do método proposto. Entretanto, é importante considerar que o HAT ficou em último lugar no ranqueamento das métricas de qualidade (Acurácia, *Kappa* e F1 Macro). Portanto, mesmo não tendo o melhor desempenho computacinal, o método proposto foi o com segundo melhor desempenho computacional, muito superior aos algoritmos clássicos que tiveram resultados de qualidade semelhantes a ele.

Portanto, como pode ser visto, a abordagem semi-supervisionada via *wrappers* para pseudo-rotulagem combinada com técnicas de super amostragem para controle de desbalanceamento proporcionou bons resultados para lidar com a classificação de fluxos de dados parcialmente rotulados. Isso mostra que é possível processar fluxos onde a quantidade de dados rotulados é escassa, e mesmo assim alcançar resultados satisfatórios relacionados a métricas de qualidade, tais como, Acurácia, Kappa e F1, sem sobrecarregar significativamente o tempo de processamento.

5.1 TRABALHOS PUBLICADOS

Durante o desenvolvimento da dissertação foi publicado o seguinte trabalho:

- Adaptive Fast XGBoost for Multiclass Classification (BALDO et al., 2023) no SBC Open Lib

5.2 TRABALHOS FUTUROS

Após destacar as contribuições e os pontos positivos deste trabalho, é pertinente sugerir melhorias que possam elevar ainda mais a eficácia do modelo proposto. Uma dessas melhorias envolve a necessidade da presença de pelo menos 1% de dados rotulados, e para superar isso, vislumbra-se a exploração de um *wrapper* que funcione totalmente de maneira não supervisionada, como por exemplo o modelo de Misturas Gaussianas (GMM). Esta técnica apresenta potencial devido à sua capacidade de modelar a distribuição de dados complexos e multivariados, facilitando a identificação de agrupamentos naturais ou grupos dentro dos dados. A implementação de um *wrapper* baseado em GMM pode oferecer uma solução para cenários onde não há dados rotulados, permitindo que o sistema identifique estruturas subjacentes nos dados de maneira autônoma.

Com base na importância crítica dos hiper parâmetros na performance dos modelos e nos resultados significativos obtidos por meio da análises de sensibilidade e otimização de hiperparâmetros, sugerem-se explorar metodologias para melhorar o desempenho dos modelos em diferentes domínios de aplicação, analisando todo o conjunto de hiperparâmetros disponíveis, desde os herdados do XGBoost e do Baldo et al. (2023), quanto os criados nesta proposta. Métodos como *Random Search* ou *Bayesian Optimization* podem ser aplicadas para ajustar os modelos de forma mais eficiente e com uma busca mais ampla do espaço de hiperparâmetros, potencialmente descobrindo combinações que oferecem melhor desempenho.

Por fim, outra alternativa é substituir o modelo do classificador principal. Por exemplo, no trabalho de Grando (2023) é proposta a utilização de um único classificador XGBoost que elimina periodicamente uma porcentagem das árvores mais antigas e realiza a substituição por novas. Isso torna o algoritmo potencialmente mais eficiente, reduzindo o risco de obsolescência dos modelos sem a necessidade de gerenciar dois modelos simultaneamente. Essa técnica simplifica o processo ao manter um único modelo que se ajusta dinamicamente, otimizando tanto o tempo de treinamento quanto a resposta do modelo às mudanças no fluxo de dados.

REFERÊNCIAS

AGRAHARI, S.; SINGH, A. K. Concept drift detection in data stream mining : A literature review. **Journal of King Saud University - Computer and Information Sciences**, v. 34, n. 10, Part B, p. 9523–9540, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157821003062>.

AGUIAR, G.; KRAWCZYK, B.; CANO, A. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework. **Machine learning**, Springer, p. 1–79, 2023.

AHMADI, Z.; BEIGY, H. Semi-supervised ensemble learning of data streams in the presence of concept drift. In: **Hybrid Artificial Intelligent Systems**. [S.l.]: Springer, 2012. (Lecture Notes in Computer Science, v. 7209), p. 497–506.

ALI, P. J. M. Investigating the impact of min-max data normalization on the regression performance of k-nearest neighbor with different similarity measurements. **ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY**, 2022.

AMINI, M.-R. et al. Self-training: A survey. **arXiv preprint arXiv:2202.12040**, 2022.

ARCHITECTURE, C. C. **scikit-multiflow Architecture**. 2020. <https://scikit-multiflow.readthedocs.io/en/stable/user-guide/core-concepts.architecture.html>. [Online; acessado 07-Abril-2023].

BALDO, F. et al. Adaptive fast xgboost for multiclass classification. **Journal of Information and Data Management**, v. 14, n. 1, 2023.

BALDO, F. et al. Adaptive fast XGBoost for binary classification. In: SBC. **Anais do XXXVII Simpósio Brasileiro de Bancos de Dados**. [S.l.], 2022. p. 13–25.

BANITALEBI-DEHKORDI, A.; GUJJAR, P.; ZHANG, Y. Auxmix: Semi-supervised learning with unconstrained unlabeled data. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2022. p. 3999–4006.

BARBER, D.; BOTEV, A. Dealing with a large number of classes – likelihood, discrimination or ranking? **arXiv: Machine Learning**, 2016.

BARROS, R. S. M. de; SANTOS, S. G. T. de C. An overview and comprehensive comparison of ensembles for concept drift. **Information Fusion**, v. 52, p. 213–244, 2019. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253518308066>.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. 2, 2012.

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

BIFET, A.; GAVALDA, R. Adaptive learning from evolving data streams. In: SPRINGER. **International Symposium on Intelligent Data Analysis**. [S.l.], 2009. p. 249–260.

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: SPRINGER. **Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I 21**. [S.l.], 2010. p. 135–150.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4.

BLACKARD, J. **Covertype**. 1998. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K5N.

BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: **Proceedings of the eleventh annual conference on Computational learning theory**. [S.l.: s.n.], 1998. p. 92–100.

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

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

BUITINCK, L. et al. API design for machine learning software: experiences from the scikit-learn project. In: **ECML PKDD Workshop: Languages for Data Mining and Machine Learning**. [S.l.: s.n.], 2013. p. 108–122.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. A discussion of semi-supervised learning and transduction. In: **Semi-supervised learning**. [S.l.]: MIT Press, 2006. p. 473–478.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002.

CHEN, M. et al. Comparing the value of labeled and unlabeled data in method-of-moments latent variable estimation. In: PMLR. **International Conference on Artificial Intelligence and Statistics**. [S.l.], 2021. p. 3286–3294.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Disponível em: <https://doi.org/10.1145/2939672.2939785>.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining**. [S.l.: s.n.], 2016. p. 785–794.

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

CHICCO, D. et al. The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. **IEEE Access**, v. 9, p. 78368–78381, 2021.

CMU, C. M. U. **School of Computer Science**. 2023. <https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/>. [Online; acessado 19-Setembro-2023].

DEDDENS, J.; PETERSEN, M. Approaches for estimating prevalence ratios. **Occupational and environmental medicine**, BMJ Publishing Group Ltd, v. 65, n. 7, p. 501–506, 2008.

DENG, H. et al. Ensemble learning for the early prediction of neonatal jaundice with genetic features. **BMC Medical Informatics and Decision Making**, v. 21, n. 1, p. 338, Dec 2021. ISSN 1472-6947. Disponível em: <https://doi.org/10.1186/s12911-021-01701-9>.

DITZLER, G. et al. Learning in nonstationary environments: A survey. **IEEE Computational Intelligence Magazine**, IEEE, v. 10, n. 4, p. 12–25, 2015.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. **Machine learning**, Springer, v. 109, n. 2, p. 373–440, 2020.

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

FATOURECHI, M. et al. Comparison of evaluation metrics in classification applications with imbalanced datasets. In: **2008 Seventh International Conference on Machine Learning and Applications**. [S.l.: s.n.], 2008. p. 777–782.

FERRARIO, A.; HäMMERLI, R. On boosting: Theory and applications. **Machine Learning eJournal**, 2019.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001.

GAMA, J. et al. Learning with drift detection. In: SPRINGER. **Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17**. [S.l.], 2004. p. 286–295.

GAMA, J. et al. A survey on concept drift adaptation. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 46, n. 4, p. 1–37, 2014.

GIRALDO-FORERO, A. F. et al. Evaluation of example-based measures for multi-label classification performance. In: **International Work-Conference on Bioinformatics and Biomedical Engineering**. [s.n.], 2015. Disponível em: <https://api.semanticscholar.org/CorpusID:29652138>.

GOHIYA, H.; LOHIYA, H.; PATIDAR, K. A survey of xgboost system. **Int. J. Adv. Technol. Eng. Res**, v. 8, p. 25–30, 2018.

GOMES, H. M. et al. Adaptive random forests for evolving data stream classification. **Machine Learning**, Springer, v. 106, p. 1469–1495, 2017.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GRANDINI, M.; BAGLI, E.; VISANI, G. Metrics for multi-class classification: an overview. **arXiv preprint arXiv:2008.05756**, 2020.

GRANDO, J. **Adaptação ao XGBOOST para suporte à regressão defluxo de dados não-estacionários**. 2023.

GWEON, H.; YU, H. A nearest neighbor-based active learning method and its application to time series classification. **Pattern Recognition Letters**, Elsevier, v. 146, p. 230–236, 2021.

HABASHY, D. M. et al. Artificial intelligence approaches for studying the pp interactions at high energy using adaptive neuro-fuzzy interface system. IntechOpen, 2023.

HASTIE, T. et al. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. v. 2.

HE, H. et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE. **2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)**. [S.l.], 2008. p. 1322–1328.

HE, H.; GARCIA, E. A. Learning from imbalanced data. **IEEE Transactions on knowledge and data engineering**, Ieee, v. 21, n. 9, p. 1263–1284, 2009.

HENRIK, B. et al. **Heterogeneity Activity Recognition**. 2015. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5689X.

HIGUCHI, Y. et al. Momentum pseudo-labeling: Semi-supervised asr with continuously improving pseudo-labels. **IEEE Journal of Selected Topics in Signal Processing**, IEEE, v. 16, n. 6, p. 1424–1438, 2022.

HOSSEINI, M. J.; GHOLIPOUR, A.; BEIGY, H. An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. **Knowledge and Information Systems**, Springer, v. 46, n. 3, p. 567–597, 2016.

HU, H.; KANTARDZIC, M.; SETHI, T. S. No free lunch theorem for concept drift detection in streaming data classification: A review. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 10, n. 2, p. e1327, 2020.

HUANG, Y.-P.; HOA, V. T. T. General criteria on building decision trees for data classification. In: **Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human**. [S.l.: s.n.], 2009. p. 649–654.

JIANG, J. et al. Tencentboost: A gradient boosting tree system with parameter server. **2017 IEEE 33rd International Conference on Data Engineering (ICDE)**, p. 281–284, 2017.

JUREK, A. et al. A survey of commonly used ensemble-based classification techniques. **The Knowledge Engineering Review**, Cambridge University Press, v. 29, n. 5, p. 551–581, 2014.

KO, S. et al. A survey on visual analysis approaches for financial data. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2016. v. 35, n. 3, p. 599–617.

KOTSIANTIS, S. B. Decision trees: a recent overview. **Artificial Intelligence Review**, v. 39, n. 4, p. 261–283, Apr 2013. ISSN 1573-7462. Disponível em: <https://doi.org/10.1007/s10462-011-9272-4>.

LEARNING, S.-S. Semi-supervised learning. **CSZ2006. html**, v. 5, 2006.

LEMAîTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017. Disponível em: <http://jmlr.org/papers/v18/16-365.html>.

LI, Y.; YIN, J.; CHEN, L. Informative pseudo-labeling for graph neural networks with few labels. **Data Mining and Knowledge Discovery**, v. 37, p. 228–254, 2022.

LIU, J. et al. A semi-supervised ensemble approach for mining data streams. **Journal of Computers**, Academy Publisher, v. 8, n. 11, p. 2873–2880, 2013.

LOSING, V.; HAMMER, B.; WERSING, H. Incremental on-line learning: A review and comparison of state of the art algorithms. **Neurocomputing**, Elsevier, v. 275, p. 1261–1274, 2018.

LU, J. et al. Learning under concept drift: A review. **IEEE Transactions on Knowledge and Data Engineering**, v. 31, n. 12, p. 2346–2363, 2019.

MASUD, M. M. et al. A practical approach to classify evolving data streams: Training with limited amount of labeled data. In: IEEE. **2008 Eighth IEEE International Conference on Data Mining**. [S.l.], 2008. p. 747–754.

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

MENARDI, G.; TORELLI, N. Training and assessing classification rules with imbalanced data. **Data mining and knowledge discovery**, Springer, v. 28, p. 92–122, 2014.

MENDITTO, A.; PATRIARCA, M.; MAGNUSSON, B. Understanding the meaning of accuracy, trueness and precision. **Accreditation and quality assurance**, Springer, v. 12, p. 45–47, 2007.

MITCHELL, T. M. et al. **Machine learning**. [S.l.]: McGraw-hill New York, 2007. v. 1.

MONTIEL, J. et al. Adaptive XGBoost for evolving data streams. In: **2020 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2020. p. 1–8.

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

MULTIFLOW, A. R. scikit. **API documentation for scikit-multiflow**. 2020. <https://scikit-multiflow.readthedocs.io/en/stable/user-guide/core-concepts.architecture.html>. [Online; acessado 07-Abril-2023].

MURPHY, K. P. **Machine learning: a probabilistic perspective**. [S.l.]: MIT press, 2012.

NUMPY, R. **Manual details functions, modules, and objects included in NumPy, release 1.20**. 2021. <https://numpy.org/doc/1.20/reference/index.html>. [Online; acessado 07-Abril-2023].

OLIPHANT, T. E. et al. **A guide to NumPy**. [S.l.]: Trelgol Publishing USA, 2006. v. 1.

OSOJNIK, A.; PANOV, P.; DŽEROSKI, S. Multi-label classification via multi-target regression on data streams. **Machine Learning**, Springer, v. 106, p. 745–770, 2017.

OZA, N. C.; RUSSELL, S. J. Online bagging and boosting. In: PMLR. **International workshop on artificial intelligence and statistics**. [S.l.], 2001. p. 229–236.

QUINONERO-CANDELA, J. et al. Evaluating predictive uncertainty challenge. In: SPRINGER. **Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers**. [S.l.], 2006. p. 1–27.

RASTOGI, R.; SHIM, K. Public: A decision tree classifier that integrates building and pruning. **Data Mining and Knowledge Discovery**, Springer, v. 4, p. 315–344, 2000.

RBF, R. D. **skmultiflow.data.RandomRBFGenerator**. 2020. <https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.data.RandomRBFGenerator.html>. [Online; acessado 07-Abril-2023].

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959.

SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. **SN computer science**, Springer, v. 2, n. 3, p. 160, 2021.

SCHAPIRE, R. E. The strength of weak learnability. **Machine learning**, Springer, v. 5, n. 2, p. 197–227, 1990.

SCHAPIRE, R. E. A brief introduction to boosting. In: CITESEER. **Ijcai**. [S.l.], 1999. v. 99, p. 1401–1406.

SETHI, T. S. et al. An ensemble classification approach for handling spatio-temporal drifts in partially labeled data streams. In: IEEE. **Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)**. [S.l.], 2014. p. 725–732.

SHAHINFAR, S.; MEEK, P.; FALZON, G. “how many images do i need?” understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. **Ecological Informatics**, Elsevier, v. 57, p. 101085, 2020.

SHELKE, M. S.; DESHMUKH, P. R.; SHANDILYA, V. K. A review on imbalanced data handling using undersampling and oversampling technique. **Int. J. Recent Trends Eng. Res**, v. 3, n. 4, p. 444–449, 2017.

SILVA, K. T. et al. Algoritmos de aprendizagem supervisionada com conjuntos de dados desbalanceados para classificação de requisitos não-funcionais. **Anais do 15. Congresso Brasileiro de Inteligência Computacional**, 2021. Disponível em: <https://api.semanticscholar.org/CorpusID:243854485>.

SKMULTIFLOW, A. **Skmultiflow.evaluation.evaluateprequential**. https://github.com/scikit-multiflow/scikit-multiflow/blob/master/CONTRIBUTING.md, 2020. Disponível em: <https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.evaluation.EvaluatePrequential.html>.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information processing & management**, Elsevier, v. 45, n. 4, p. 427–437, 2009.

SOUZA, F. M. de; GRANDO, J.; BALDO, F. Adaptive fast XGBoost for regression. In: XAVIER-JUNIOR, J. C.; RIOS, R. A. (Ed.). **Intelligent Systems**. Cham: Springer International Publishing, 2022. p. 92–106. ISBN 978-3-031-21686-2.

STOLFO, S. et al. **KDD Cup 1999 Data**. 1999. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51C7N.

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**. [S.l.: s.n.], 2001. p. 377–382.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. [S.l.]: MIT press, 2018.

TAKAHASHI, K. et al. Confidence interval for micro-averaged f 1 and macro-averaged f 1 scores. **Applied Intelligence**, Springer, v. 52, n. 5, p. 4961–4972, 2022.

TAUNK, K. et al. A brief review of nearest neighbor algorithm for learning and classification. In: IEEE. **2019 International Conference on Intelligent Computing and Control Systems (ICCS)**. [S.l.], 2019. p. 1255–1260.

THOMPSON, M. L.; MYERS, J.; KRIEBEL, D. Prevalence odds ratio or prevalence ratio in the analysis of cross sectional data: what is to be done? **Occupational and Environmental Medicine**, v. 55, p. 272 – 277, 1998.

TRIGUERO, I.; GARCÍA, S.; HERRERA, F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. **Knowledge and Information systems**, Springer, v. 42, p. 245–284, 2015.

VAFAIE, P.; VIKTOR, H.; MICHALOWSKI, W. Multi-class imbalanced semi-supervised learning from streams through online ensembles. In: **2020 International Conference on Data Mining Workshops (ICDMW)**. [S.l.: s.n.], 2020. p. 867–874.

VERGARA, A. **Gas Sensor Array Drift Dataset**. 2012. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5RP6W.

WAGNER, T. et al. Semi-supervised learning on data streams via temporal label propagation. In: **ICML 2018**. [s.n.], 2018. Disponível em: <https://www.amazon.science/publications/semi-supervised-learning-on-data-streams-via-temporal-label-propagation>.

WANG, Q.-W.; LI, Y.-F.; ZHOU, Z.-H. Partial label learning with unlabeled data. In: **IJCAI**. [S.l.: s.n.], 2019. p. 3755–3761.

WARDHANI, N. W. S. et al. Cross-validation metrics for evaluating classification performance on imbalanced data. In: **2019 International Conference on Computer, Control, Informatics and its Applications (IC3INA)**. [S.l.: s.n.], 2019. p. 14–18.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier Brasil, 2017. v. 2.

WEBB, G. I. et al. Characterizing concept drift. **Data Mining and Knowledge Discovery**, Springer, v. 30, n. 4, p. 964–994, 2016.

WEN, Z. et al. Exploiting gpus for efficient gradient boosting decision tree training. **IEEE Transactions on Parallel and Distributed Systems**, v. 30, p. 2706–2717, 2019.

XGBOOST, L. **Specify the learning task and the corresponding learning objective, version 1.7.5**. 2023. <https://xgboost.readthedocs.io/en/stable/parameter.html>. [Online; acessado 07-Abril-2023].

YAHYAOUI, A. et al. Machine learning based network intrusion detection for data streaming iot applications. In: **2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)**. [S.l.: s.n.], 2021. p. 51–56.

YING, X. An overview of overfitting and its solutions. In: IOP PUBLISHING. **Journal of physics: Conference series**. [S.l.], 2019. v. 1168, p. 022022.

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

ZHU, X. **Semi-supervised learning literature survey, Department of Computer Sciences, University of Wisconsin**. [S.l.], 2008.

ZHU, X. Semi-supervised learning. In: ____. **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 892–897. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_749>.

ZUBAROĞLU, A.; ATALAY, V. Data stream clustering: a review. **Artificial Intelligence Review**, Springer, v. 54, n. 2, p. 1201–1236, 2021.

Assinaturas do documento

Código para verificação: **1YC734AB**

Este documento foi assinado digitalmente pelos seguintes signatários nas datas indicadas:

FABIANO BALDO (CPF: 028.XXX.209-XX) em 12/08/2024 às 15:13:44
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:47:06 e válido até 30/03/2118 - 12:47:06.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwMzQzMTZfMzQzNTlfMjAyNF8xWUM3MzRBQg== ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00034316/2024** e o código **1YC734AB** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.