PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA – PPGIa

SHEILA CRISTIANA DE FREITAS

UMA ABORDAGEM DE APRENDIZAGEM DE MÁQUINA PARA AUTOCONFIGURAÇÃO DE DETECTORES DE MUDANÇA DE CONCEITO EM MINERAÇÃO DE PROCESSOS

Curitiba
2023

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA – PPGIa

SHEILA CRISTIANA DE FREITAS

UMA ABORDAGEM DE APRENDIZAGEM DE MÁQUINA PARA AUTOCONFIGURAÇÃO DE DETECTORES DE MUDANÇA DE CONCEITO EM MINERAÇÃO DE PROCESSOS

Projeto de pesquisa para Tese de Doutorado apresentado ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial à obtenção do título de Doutor em Informática.

Linha de pesquisa: Inteligência Artificial
Orientador: Prof. Dr. Edson Emílio Scalabrin
Coorientador: Prof. Dr. Jean Paul Barddal

Curitiba
2023

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Edilene de Oliveira dos Santos CRB 9 / 1636

F866a
2023

Freitas, Sheila Cristiana de
Uma abordagem de aprendizagem de máquina para autoconfiguração de detectores de mudança de conceito em mineração de processos / Sheila: Cristiana de Freitas ; orientador: Edson Emílio Scalabrin ; coorientador: Jean Paul Barddal. -- 2023
140 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2023.
Bibliografia: f. 127-136

1. Informática. 2. Mineração de processos. 3. Aprendizado do computador. 4. Confiabilidade dos dados. I. Scalabrin, Edson Emilio. II. Barddal, Jean Paul. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título

CDD 20. ed. – 004

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

Curitiba, 15 de maio de 2024.

38-2024

DECLARAÇÃO

Declaro para os devidos fins, que **Sheila Cristiana de Freitas** defendeu a tese de Doutorado intitulada "**UMA ABORDAGEM DE APRENDIZAGEM DE MÁQUINA PARA AUTOCONFIGURAÇÃO DE DETECTORES DE MUDANÇA DE CONCEITO EM MINERAÇÃO DE PROCESSOS"**, na área de concentração Ciência da Computação no dia 05 de dezembro de 2023, no qual foi aprovada.

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

Por ser verdade firmo a presente declaração.

_____
Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus, por me fortalecer e por guiar pessoas e oportunidades que me inspiraram ao longo do meu caminho.

Em especial, expresso minha gratidão ao meu orientador, Professor Edson Scalabrin, cuja orientação não apenas contribuiu para o meu conhecimento acadêmico, mas também para meu desenvolvimento pessoal. Ele compreendeu minhas limitações e acreditou no meu trabalho, sendo sempre um grande exemplo de sabedoria e companheirismo.

Não menos importante, agradeço ao meu coorientador, Professor Jean Barddal, que sempre se mostrou disponível para superar os desafios de cada etapa do trabalho. Seu apoio constante e inspiração, provenientes de sua trajetória como pesquisador, foram fundamentais para o meu desenvolvimento.

Minha sincera gratidão se estende à minha família, com destaque para meu marido, Henrique, e minha filha Julia, pelo apoio incondicional e pela compreensão ao longo desta jornada.

Minha gratidão eterna à minha amiga Denise, por quem tenho total admiração. Sua generosidade e determinação são fontes perenes de inspiração para mim.

Não posso deixar de reconhecer o valioso papel dos professores do PPGIA e de outros programas da PUCPR, que contribuíram significativamente para minha formação acadêmica durante o doutorado.

Meus agradecimentos se estendem aos amigos que fiz durante esse período, com destaque para Luiz, Juliano e Eduardo, cuja amizade enriqueceu minha experiência acadêmica.

Ao Instituto Federal do Paraná, agradeço pela concessão que possibilitou o desenvolvimento integral da minha pesquisa.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), expresso minha gratidão pela concessão da bolsa de custeio de taxa escolar.

Cada um desses agradecimentos reflete a importância fundamental que cada pessoa e instituição teve na minha jornada acadêmica e, por isso, expresso minha profunda gratidão a todos.

RESUMO

A mineração de processos é uma área de pesquisa que visa descobrir modelos de instâncias de processos de negócios para permitir uma análise detalhada do comportamento operacional. Avaliar a qualidade dos modelos é crucial, demandando dados e métricas objetivas. A dinamicidade dos modelos representa um desafio, especialmente na gestão de mudanças nos processos de negócios. Existem várias abordagens para lidar com mudanças de conceito em processos, exigindo um estudo aprofundado. Essa análise se utiliza de técnicas em constante desenvolvimento que requerem conhecimento prévio do analista sobre as ferramentas e técnicas utilizadas, incluindo a escolha do detector de mudanças de processo mais adequada ao tipo de *log* de eventos analisados. Para tanto, visa-se com este trabalho o desenvolvimento de um método autoconfigurável para a detecção de mudanças de conceitos em processos de negócio, com base em aprendizado de máquina. O método utiliza-se do classificador XGBoost, treinado com uma base de atributos gerada em cada análise de detecção de mudança, focando na análise objetiva das dimensões de qualidade dos modelos de processo. Para a autoconfiguração, o método é iniciado pela detecção dos pontos de mudança no *log* de eventos ao utilizar uma ferramenta que permite a autoconfiguração do Detector de Mudança, ou configuração manual. Os pontos de variação detectados geram *sublogs* para a avaliação dos novos modelos de processo. Portanto, quatro métricas de qualidade (*fitness, precisão, generalização e simplicidade*) são calculadas tanto para o modelo geral quanto para cada sublog, registrando as informações de análise para alimentar a base de atributos, e os resultados foram obtidos por meio de testes em bases públicas. Para detecção dos pontos de mudança, foi utilizada a ferramenta IPDD, variando entre dois detectores (ADWIN e HDDM-W) e seus parâmetros, assim como as configurações de detecção. No âmbito do modelo preditivo, ao considerar as restrições inerentes a um conjunto de dados de dimensões reduzidas, obtivemos resultados que podem ser considerados como satisfatórios dentro de um espectro que abrange 20 classes distintas, apresentando uma acurácia de 62%. Com este coeficiente, é refletida a proporção de predições corretas em comparação ao total de tentativas realizadas pelo referido modelo. Ademais, ao proceder os cálculos das métricas de qualidade, nomeadamente a precisão, entre o *log* geral e os *sublogs* específicos, observou-se aumentos altamente significativos. Foi verificado ainda que a precisão para o *log* geral foi de 60%, com um desvio padrão de 5%. Em contrapartida, para os *sublogs*, a precisão alcançou um patamar de 97%, com um desvio padrão de 3%.

Palavras-chave: mineração de processos; mudança de conceito; detector de mudança; seleção de atributos; autoconfiguração.

ABSTRACT

Process mining is an area of research that aims to discover models of business process instances to enable detailed analysis of operational behavior. Assessing the quality of models is crucial, requiring objective data and metrics. The dynamicity of models represents a challenge, especially when managing changes in business processes. There are several approaches to dealing with concept drift in processes, requiring in-depth study. This analysis uses techniques that are constantly developing, requiring the analyst to have prior knowledge of the tools and techniques used, including the choice of the process change detector best suited to the type of event log analyzed. This work aims to develop a self-configurable method for detecting changes in concepts in business processes, based on machine learning. The method uses the XGBoost classifier, trained with a base of attributes generated in each change detection analysis, focusing on the objective analysis of the quality dimensions of the process models. For self-configuration, the method starts by detecting change points in the event log, using a tool that allows self-configuration of the Change Detector or manual configuration. The detected change points generate *sublogs* for the evaluation of new process models. Four quality metrics (fitness, accuracy, generalization and simplicity) are calculated for both the overall model and each sublog, recording analysis information to feed the attribute base. The results were obtained through tests on public bases. To detect change points, the IPDD tool was used, varying between two detectors (ADWIN and HDDM-W) and their parameters, as well as the detection settings. Within the scope of the predictive model, considering the restrictions inherent to a set of reduced dimensions, we obtained results that can be considered satisfactory within a spectrum covering 20 different classes, presenting an accuracy of 62%. This coefficient reflects the proportion of correct predictions compared to the total number of attempts made by that model. Furthermore, when calculating quality metrics, namely precision, between the general log and specific *sublogs*, highly significant increases were observed. It was found that the accuracy for the overall log was 60%, with a standard deviation of 5%. On the other hand, for *sublogs*, precision reached a level of 97%, with a standard deviation of 3%.

Keywords: Process Mining; Concept drift; Change detectors; attribute selection; Autoconfiguration.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

ADWIN	*Adaptive Windowing Algorithm*
BPIC	*Business Process Intelligence Challenge*
BPMN	*Business Process Model and Notation*
DDM	*Drift Detection Method*
DFG	*Directly-follows graph*
DSRM	*Design science research methodology*
KNN	*K-nearest Neighbors Classification*
LCDD	*Detecting business process drifts based on local*
LOO	*Leave-One-Out*
LOG	Registro de eventos significativos em um sistema computacional
MOA	*Massive On-line Analysis*
MXML	*Mining eXtensible Markup Language*
PROM	*Process Mining framework*
TESSERACT	*Time-drifts in event streams using series of evolving rolling averages of completion times*
XES	*eXtensible Event Stream*
XGBOOST	*Extreme Gradient Boostin*g

SUMÁRIO

1 INTRODUÇÃO

As organizações, em sua maioria, sejam elas voltadas para a produção de bens ou serviços, têm a responsabilidade não apenas de gerenciar seus processos e produtos para manter uma vantagem competitiva e assegurar sua sobrevivência no ambiente em que atuam, mas também de estar atentas a todas as etapas que compõem o ciclo de vida de cada processo. Isso é fundamental para evitar a perda de controle sobre ele e otimizar seus recursos. Nesse contexto, é imperativo que todas as atividades sejam submetidas a monitoramento e supervisão, fator que tem levado as empresas a depender cada vez mais de tecnologias de apoio à gestão de processos organizacionais. Seguindo essa premissa, a tecnologia da informação tem a capacidade de armazenar vastas quantidades de informações que, por sua vez, registram os passos e os dados das atividades realizadas de várias maneiras, incluindo recursos como pessoas e mecanismos. Esses registros são mantidos em arquivos sequenciais de dados comumente conhecidos como "*log* de eventos", em que representam a matéria-prima bruta para a área de estudos de mineração de processos.

A mineração de processos é uma área de pesquisa que oferece uma abordagem para a descoberta de modelos de instâncias de processos de negócios. Ela permite uma análise detalhada do comportamento de um processo operacional ao empregar técnicas que combinam os campos da ciência de processos e da ciência de dados. Os dados necessários para essa análise são extraídos de registros de eventos, geralmente denominados “logs de eventos”.

A partir de um *log* de eventos, é possível gerar um modelo que representa os caminhos percorridos por cada etapa do processo. Isso tem um impacto significativo na tomada de decisão e na otimização do processo de negócios sob análise. Além disso, a mineração de processos pode identificar desvios de comportamento inesperados. Pode-se afirmar, portanto, que a mineração de processos desempenha um papel crucial, atuando como uma ponte entre a mineração de dados e o gerenciamento de processos de negócios (1).

Na vida prática, a mineração de processos tem como objetivo a extração de informações valiosas a partir de processos armazenados em bases de dados no formato de logs de eventos. Sua principal contribuição está na descoberta de um modelo de instâncias de processo que reflita a realidade da organização (2).

Já a descoberta desse modelo apresenta diversas características e desafios. Um significativo desafio está relacionado à gestão da mudança nos processos, ou seja, à capacidade de lidar com situações em que o processo se altera durante a análise. Por exemplo, no início do

log de eventos, duas atividades podem ser registradas como ocorrendo simultaneamente; porém, mais tarde, no mesmo registro, essas mesmas atividades podem ser registradas de forma diferente (2). As mudanças nos processos podem ocorrer de forma periódica ou sazonal. Para compreendê-las e lidar eficazmente com elas, foram desenvolvidas técnicas de detecção de mudanças de conceito ou de processo que buscam identificar automaticamente as variações e possibilitar ajustes contínuos no modelo de processo em uso, aprimorando, assim, sua capacidade de se adequar ao comportamento observado nos dados.

Para tanto, as ferramentas de detecção de mudanças de conceito fazem uso de algoritmos denominados "detectores de mudanças", sendo que cada um corresponde a um método específico de detecção. Por exemplo, destacam-se os detectores ADWIN (3) e DDM (4), e outros métodos que serão mencionados ao longo deste documento. A finalidade primordial desses algoritmos é identificar alterações no comportamento atual de um processo de negócios por meio da interpretação dos dados associados aos eventos ocorridos no referido processo. Essas ferramentas frequentemente implementam estratégias distintas, adaptadas a objetivos específicos relacionados ao tipo de mudança de conceito a ser tratado. Isso pode tornar desafiadora a seleção da estratégia apropriada e a configuração dos parâmetros, especialmente quando conduzida por indivíduos não familiarizados com essa área de análise e com as ferramentas correspondentes.

Nos estudos relacionados (5), foram identificadas duas questões de relevância. A primeira delas diz respeito à validação dos pontos de mudança de conceito identificados – dado que é necessário ter conhecimento prévio acerca dos referidos pontos de mudança. Geralmente, a preparação das bases de dados de validação envolve a criação de dados de eventos sintéticos, os quais são gerados com instruções explícitas sobre a localização de cada ponto de mudança. No entanto, em trabalhos que utilizam bases de dados reais, a validação pode contar com a ajuda de um especialista no processo em análise e permitir que esse profissional avalie a detecção do ponto de mudança com base em seu conhecimento do negócio.

A segunda questão de relevância observada está relacionada à própria utilização de ferramentas de detecção de mudanças de conceito – uma tarefa que não é trivial, uma vez que cada ferramenta requer conhecimento prévio dos conjuntos de parâmetros utilizados para a detecção de pontos de mudança.

Com o intuito de contribuir para a tarefa de determinar os dados mais adequados para a parametrização necessária de uma específica ferramenta de detecção de mudanças de conceito, o presente trabalho propõe um método de autoconfiguração baseado em aprendizagem de

máquina e na seleção de características para a parametrização de estratégias/ferramentas de detecção de pontos de mudança. Além disso, como parte das contribuições abordadas em conjunto com a autoconfiguração, a análise da qualidade do modelo descoberto é considerada. Nesse sentido, a avaliação da qualidade deste levará em consideração as quatro dimensões canônicas de qualidade de um modelo de processo (1), a saber: *fitness*, precisão generalização e simplicidade, sendo cada dimensão avaliada de forma quantitativa, por meio de métricas específicas.

Além disso, este trabalho realizará análises da detecção de mudança com foco na qualidade dos modelos encontrados. A partir delas, as fontes de dados para a análise e seleção de características serão compostas por resultados de verificações anteriores bem-sucedidas. Essas fontes de dados incluirão informações do *log* de eventos, do detector utilizado e das métricas de qualidade, e formarão o conjunto de dados necessário para a criação e validação do modelo preditivo.

A abordagem para a previsão de informações utilizada na autoconfiguração é baseada em aprendizado de máquina.

Em termos gerais, esta pesquisa propõe um método capaz de utilizar o aprendizado de máquina de forma eficaz para construir um modelo de previsão ou recomendação. Este é aplicado para sugerir a autoconfiguração do detector de mudança de conceito analisada. Além disso, o trabalho fornece uma análise comparativa entre o modelo global de um processo M_i e cada modelo subsequente (M_{i+1}) descoberto para cada *sublog* do *log* de eventos, compreendido entre os pontos de mudança *i* e *i+1*. $M_{i=0}$ representa o modelo de processo derivado do *log* de eventos completo, considerado em sua totalidade.

Antes de adentrar na descrição das técnicas empregadas, é relevante esclarecer que, ao longo deste estudo, serão consideradas duas perspectivas: a do analista de processos e a do especialista de negócios. O primeiro ator, o analista de processos, representa um indivíduo com proficiência no uso de ferramentas de mineração de processos, capaz de analisar trajetos, avaliar o tempo das atividades, identificar parâmetros necessários, mapear o fluxo de atividades e detectar falhas, entre outras habilidades. Essa pessoa é apta para manusear tais ferramentas e extrair informações geradas a partir dos modelos descobertos e dos resultados das métricas de qualidade.

O segundo ator, o especialista de negócios, é alguém que detém conhecimentos sobre a atividade principal da organização, assim como sobre as operações de produção e/ou serviços.

Essa pessoa possui compreensão da cadeia produtiva da organização, bem como das atividades em cada departamento e de cada colaborador dentro da organização.

1.1 PROBLEMA E MOTIVAÇÃO

As técnicas comuns de mineração de processos, como a descoberta e verificação de modelos, juntamente com o aprimoramento, pressupõem erroneamente a estabilidade dos processos (6). Esse pressuposto, porém, revela-se ingênuo, já que esses frequentemente sofrem alterações não refletidas nos modelos existentes, resultando em análises desatualizadas. Os negócios, sujeitos a mudanças estratégicas, demandam a constante reavaliação dos processos para mitigar incertezas; logo, mudanças substanciais, relacionadas a fatores vitais, humanos ou organizacionais são fatores que geram impacto ao longo do tempo. A gestão dessas mudanças é crucial na saúde, em que os padrões de cuidado evoluem em resposta às necessidades individuais dos pacientes e a interações complexas entre pessoas, processos, tecnologia e estruturas organizacionais em constante evolução (8).

Tomando como exemplo a área da saúde, é inegável que muitos processos nesse setor estão em constante estado de fluxo e evolução. Isso ressalta a importância de detectar mudanças a fim de compreender suas causas e, assim, antecipar-se a futuras alterações ou melhorias nos procedimentos.

Na mineração de processos, existe um campo de pesquisa dedicado ao estudo de mudanças, conhecido como *mudança de conceito* ou *mudança de processo*, que serão tratadas como sinônimas doravante. É relevante destacar que a mudança de conceito é um problema amplamente reconhecido tanto na mineração de dados quanto na aprendizagem de máquina (7). As mudanças nos processos também precisam ser identificadas e detectadas para possibilitar uma análise precisa. Contudo, na área de processos de negócios, as estruturas de dados envolvidas são mais complexas.

Foi observado na literatura, como detalhado na seção *Trabalhos Relacionados*, que, até o momento, a pesquisa tem se concentrado em propor métodos capazes de detectar e caracterizar mudanças. Cada um é, portanto, projetado para atender a uma característica específica de mudança, muitas vezes resultando no desenvolvimento de ferramentas específicas para tipos particulares de dados e mudanças.

Para tanto, as ferramentas de detecção de mudanças de conceitos empregam estratégias implementadas na forma de detectores de mudanças. No entanto, essas ferramentas demandam

um conhecimento avançado para ajustar suas parametrizações e exigem a intervenção de um especialista para sua operação, o que dificulta a utilização por parte de gestores e analistas de processos. Portanto, a necessidade de desenvolver um conhecimento profundo para identificar mudanças e obter uma análise mais precisa é um dos principais desafios a serem superados. Além disso, a validação de um ponto de mudança detectado requer a verificação de sua localização, que pode ser feita por meio da rotulagem em bases de dados sintéticas ou com a intervenção de um especialista no processo de negócios em questão (8, 9).

Apesar dos avanços na pesquisa em mineração de processos, ainda existem diversos desafios na área de mudança de conceito, enquanto muitos dos elencados no manifesto da área (2) continuam demandando atenção até os dias atuais. Claramente, a busca contínua pela qualidade do modelo descoberto é um desafio persistente, e isso pode ser combinado com uma avaliação detalhada de cada ponto de mudança no processo, fazendo uso de cada nova versão do modelo após a detecção de um ponto de mudança.

Nesta pesquisa, acredita-se que, por exemplo, é possível analisar a conformidade ou calcular a qualidade do novo modelo encontrado após a detecção de um ponto de mudança, em busca, como resultado, do modelo que melhor se assemelha à situação real do processo. É importante destacar que as métricas relacionadas às dimensões de qualidade são parte integrante do processo de análise, a saber: *fitness, precisão, simplicidade e generalização.* Essas métricas são detalhadamente descritas em um capítulo específico na fundamentação da mineração de processos.

No entanto, a mineração de processos enfrenta uma carência de ferramentas que permitam lidar com problemas reais relacionados a ambientes nos quais os processos sofrem mudanças frequentes. Isso se torna um obstáculo significativo, uma vez que a detecção e a adaptação de processos sem o auxílio de ferramentas apropriadas se revelam dispendiosas e estão sujeitas ao viés dos especialistas envolvidos em sua construção. É evidente que a detecção de uma mudança representa um evento crucial. O passo subsequente envolve a identificação do novo modelo de processo, e a sua avaliação correta tem um impacto direto na continuidade eficaz, ou não, do fluxo de trabalho de uma organização, influenciando tanto na precisão quanto na confiabilidade do que é produzido. Nesse contexto, destacam-se três aspectos cruciais:

Em primeiro lugar, é essencial desenvolver uma metodologia capaz de recomendar a parametrização automatizada de um detector de mudança de conceito.

Outro ponto relevante consiste no desenvolvimento de um método objetivo para avaliar a qualidade de um modelo de processo de negócios. Isso envolve a comparação da qualidade

do modelo geral do processo de negócios em relação a cada modelo de processo intercorrente, descoberto a partir do *sublog* entre dois pontos de mudança devidamente identificados.

Por fim, é necessário conceber uma metodologia que valide objetivamente o modelo de processo descoberto para cada ponto de mudança detectado, localizado e caracterizado.

A realização deste trabalho de pesquisa, portanto, visa preencher as lacunas mencionadas anteriormente. Esses esforços se justificam uma vez que, durante a revisão da literatura, não foi encontrada uma abordagem que ofereça um método de parametrização automática para ferramentas de detecção de pontos de mudança em processos de negócios, assim como métodos de avaliação e validação objetiva da qualidade do modelo de processo descoberto em um ambiente de mudança. É importante notar que o usuário ou o analista de um processo, pode desempenhar um papel na tarefa de análise, indicando suas escolhas ou preferências em relação às dimensões canônicas de qualidade, a saber: *aptidão/fitness, precisão, generalização e simplicidade.*

1.2 QUESTÕES DE PESQUISA

A mineração de processos tem como objetivo extrair conhecimento valioso dos registros de eventos, comumente disponíveis nos sistemas de informação existentes (2). Considerando que os processos não são estáticos, a mineração de processos se expande para incluir a detecção de mudanças, que pode ser utilizada, por exemplo, para gerar alertas. A escolha do modelo descoberto deve garantir uma qualidade significativamente relevante em relação a cada dimensão de qualidade analisada, ou com a combinações de várias dimensões. Parte-se, então, dessa suposição como um princípio orientador para a análise da qualidade, que desempenhará papel fundamental no refinamento da verificação de melhoria do processo examinado em relação ao modelo descoberto. No que concerne à parametrização automática dos detectores de mudança, a pesquisa concentrou-se na aplicação da aprendizagem de máquina, e visa desenvolver um modelo preditivo com desempenho confiável.

Um dos desafios consiste em avaliar o impacto da utilização de aprendizado de máquina e da seleção de características na recomendação dos parâmetros de configuração de uma ferramenta de detecção de pontos de mudança. A qualidade de um processo de negócios será objetivamente medida por meio de métricas específicas, de acordo com o tipo de análise de modelo que o analista de processos deseja realizar.

Nesse contexto, as questões de pesquisa são as seguintes:

- Q1: Técnicas de aprendizado de máquina podem contribuir por meio de um modelo preditivo para a sugestão dos parâmetros utilizados no detector de mudança de conceito?

- Q2: As métricas de qualidade de processos de negócios, calculadas a partir do *log* geral, quando comparadas com as métricas geradas pelos modelos derivados de *sublogs* encontrados nos pontos de mudança, contribuem, de maneira objetiva, para a avaliação dos modelos de processo gerados a partir de cada ponto de mudança?

Além das questões mencionadas anteriormente, foram estabelecidos diversos objetivos de estudo com o propósito de guiar a estruturação e a realização dos esforços de pesquisa.

1.3 OBJETIVO

Este projeto visa desenvolver um método autoconfigurável para a detecção de mudanças de conceitos em processos de negócio. O método é baseado em aprendizado de máquina, com ênfase na seleção de atributos e na análise objetiva das dimensões de qualidade de cada modelo de processo gerado.

Para atingir esse objetivo, foram definidos os seguintes objetivos específicos:

• Criar uma interface de interação entre o analista de processo e o *framework* de avaliação da qualidade do modelo de processo de negócio descoberto, permitindo que o analista conduza a tarefa de análise de acordo com suas escolhas ou preferências;

• Conceber um método capaz de recomendar a parametrização automática de um detector de mudanças de conceito, utilizando procedimentos de aprendizado de máquina em um processo de negócio;

• Implementar um método para avaliar objetivamente a qualidade de um modelo de processo de negócio, comparando a qualidade do modelo geral com cada subsequente descoberto a partir de *sublogs* entre dois pontos de mudança devidamente identificados;

• Validar o modelo de processo descoberto para cada ponto de mudança detectado, localizado e caracterizado *vis-à-vis* às preferências expressas pelo analista de processo.

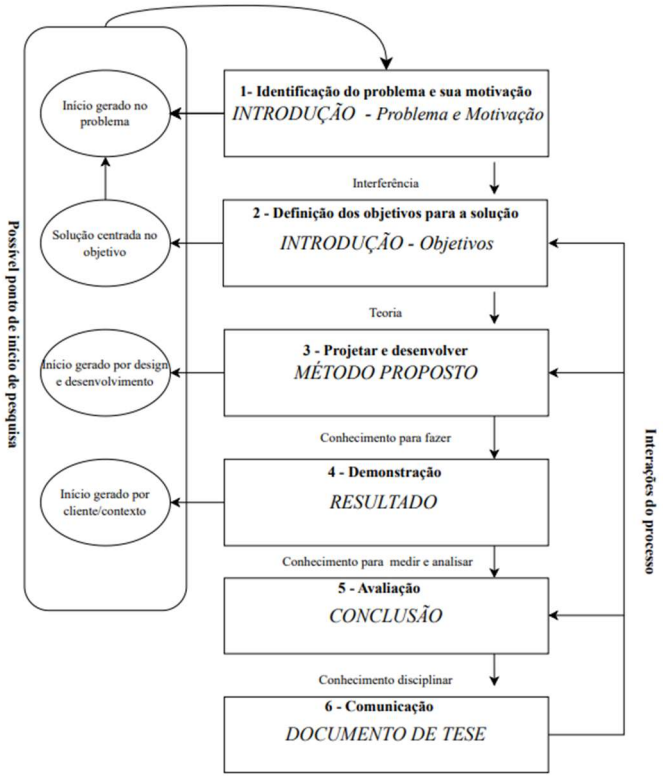
A estruturação do método proposto é apresentada na seção Método Proposto. Já a descrição e formalização das métricas relativas às quatro dimensões de qualidade dos modelos – *aptidão/fitness, precisão, generalização e simplicidade* – são detalhadamente abordadas em uma seção específica no capítulo sobre Dimensões da Qualidade de Modelos de Processo. Além disso, conceitos fundamentais relacionados à proposta de projeto são explorados na seção de Fundamentação e Trabalhos Relacionados.

1.4 MÉTODO DE PESQUISA

Este tópico aborda a estratégia metodológica escolhida para o desenvolvimento do trabalho, que consiste em duas fases principais. A primeira fase é implementada por meio de estudos exploratórios, estes com o objetivo de identificar as principais abordagens para a detecção de mudança de conceito na mineração de processos. Além disso, essa revisão de literatura permitiu a identificação dos principais conceitos e perspectivas relacionados ao tratamento dos problemas abordados nas questões de pesquisa (5). A segunda fase envolveu a implementação do método proposto e a avaliação dos resultados.

Este estudo adota a metodologia proposta por (10), conhecida como *design science research methodology* (DSRM) que orienta o desenvolvimento do trabalho por meio de etapas procedurais – que são sugeridas em uma ordem específica, mas podem ser adaptadas de acordo com as necessidades de cada projeto, vide Figura 1. Essas etapas incluem: Etapa 1 – identificação do problema e sua motivação; Etapa 2 – definição dos objetivos para a solução; Etapa 3 – concepção e desenvolvimento; Etapa 4 – demonstração; Etapa 5 – avaliação; e Etapa 6 – comunicação.

Figura 1 – Etapas do DSRM



Fonte: adaptado de (10) .

1.5 CONTRIBUIÇÕES DA PESQUISA

A principal contribuição deste trabalho consiste no desenvolvimento de um método autoconfigurável para a detecção de mudanças de conceitos em processos de negócio, este baseado em aprendizagem de máquina e com ênfase na seleção de características e na análise objetiva das dimensões de qualidade de cada modelo de processo gerado. Esta contribuição se desdobra em contribuições secundárias: (a) a concepção e implementação de um método que recomenda a parametrização automatizada de um detector de mudança de conceito, utilizando um procedimento de seleção de características em um processo de negócio; (b) o desenvolvimento de uma metodologia para avaliar objetivamente a qualidade de um modelo de processo de negócio, comparando a qualidade do modelo geral com cada subsequente descoberto a partir de *sublogs* entre dois pontos de mudança devidamente identificados; e (c) a criação de um método para validar, de forma objetiva, o modelo de processo descoberto para cada ponto de mudança detectado, localizado e caracterizado em relação às preferências expressas pelo analista de processo.

Além disso, inclui a implementação de uma interface de interação entre o analista de processo e o *framework* de avaliação da qualidade do modelo de processo de negócio descoberto, permitindo ao analista conduzir a análise de acordo com suas escolhas e preferências.

1.6 DEFINIÇÃO DO ESCOPO

Este trabalho propõe a elaboração de um método de apoio à análise da detecção de mudanças de conceito na mineração de processos, com ênfase na utilização da seleção de características para fins de autoconfiguração e aprimoramento da qualidade dos modelos descobertos.

A identificação de mudanças de conceito requer um conhecimento específico, o que torna essa tarefa desafiadora, mesmo para um analista de processos. A proposta, então, concentra-se na definição do fluxo de trabalho de aprendizado de máquina, visando a criação de um modelo preditivo que auxilia na parametrização de ferramentas de detecção de mudanças em processos. O esforço principal é direcionado para a seleção de características para construir o modelo preditivo.

Optou-se por utilizar exclusivamente a ferramenta IPDD para gerar resultados que compõem as bases de características usadas no treinamento do modelo preditivo. Inicialmente, a versão original desta ferramenta utiliza apenas o detector ADWIN. No entanto, ao longo do desenvolvimento do projeto, foram realizadas implementações adicionais que enriqueceram significativamente o método. Essas implementações estão detalhadas na seção *Contribuições da Ferramenta IPDD para o Método proposto – Implementações Específicas.* A princípio, a experimentação do método estava restrita ao detector ADWIN, mas posteriormente inclui-se também o detector HDDM-W (11).

No que se refere ao desenvolvimento do *framework* para validar o método, optou-se por utilizar a linguagem de programação Python[1]. Essa escolha baseou-se na disponibilidade de recursos de código aberto na área de mineração de processos, na seleção de características, no aprendizado de máquina e na análise de dados. Para as tarefas de mineração de processos,

[1] Site oficial: https://www.python.org/

utilizou-se a biblioteca do PM4PY[2], enquanto, para as tarefas de aprendizado de máquina, contou-se com a ferramenta *scikit-learn*[3].

1.7 ORGANIZAÇÃO DO DOCUMENTO

O presente trabalho está estruturado da seguinte maneira: na Seção 2 são apresentados os conceitos fundamentais necessários para o desenvolvimento deste trabalho, abrangendo os seguintes tópicos: mineração de processos, registro de eventos, dimensões da qualidade de modelos de processos e métricas associadas, mudança de conceito e seus tipos, detectores de mudanças de conceitos, pré-processamento dos dados e contribuições, além de aprendizado de máquina e trabalhos relacionados; a Seção 3 detalha o método de pesquisa, descrevendo as etapas que foram implementadas. Na Seção 4, por sua vez, são apresentados os resultados dos testes realizados. Por fim, a Seção 5 encerra com as considerações finais do estudo.

[2] Disponível em: https://pm4py.fit.fraunhofer.de/
[3] Site oficial: https://scikit-learn.org/stable/

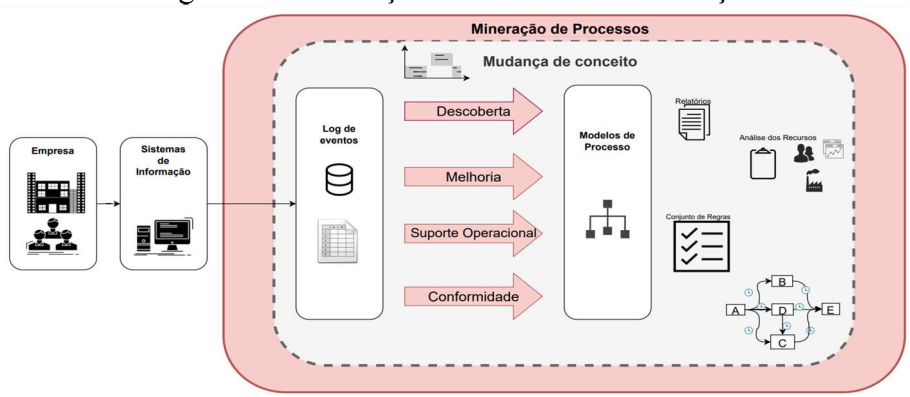
2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

Conforme previamente delineado, o presente trabalho objetiva contribuir para a detecção de mudanças de conceito na mineração de processos. No transcurso deste capítulo, serão expostos os conceitos fundamentais inerentes ao tema de pesquisa, visando proporcionar familiaridade aos leitores desprovidos de conhecimento prévio sobre o assunto. Tal abordagem visa possibilitar a compreensão dos objetivos e contribuições desta pesquisa. Ao término deste, serão discutidos, também, trabalhos relacionados.

2.1 MINERAÇÃO DE PROCESSOS

Os sistemas de informação são indispensáveis para todas as atividades organizacionais do século XXI. Cada ação ou operação gera uma quantidade considerável de dados brutos, na forma de registros de eventos. Esses registros, ou logs de eventos, consistem em rastros que contêm informações valiosas, acessíveis e passíveis de análise por meio de ferramentas de mineração de processos. Em termos gerais, esses eventos são gerados em diversas situações, como ao realizar um pedido de compra de bens ou serviços, ao efetuar um saque em um caixa eletrônico, ao ajustar máquinas em um ambiente de produção ou durante uma consulta médica, incluindo procedimentos e exames relacionados. Além disso, a mineração de processos é uma técnica desafiadora que visa explorar, de maneira significativa, os dados de eventos, com o propósito de oferecer *insights*, identificar gargalos, antecipar problemas, detectar violações de políticas, sugerir medidas corretivas e simplificar os processos. O ponto de partida para tal reside no registro de eventos (2). A Figura 2 ilustra as principais perspectivas da mineração de processos, com o *log* de eventos como ponto de partida.

Figura 2 – Mineração de Processo – Introdução

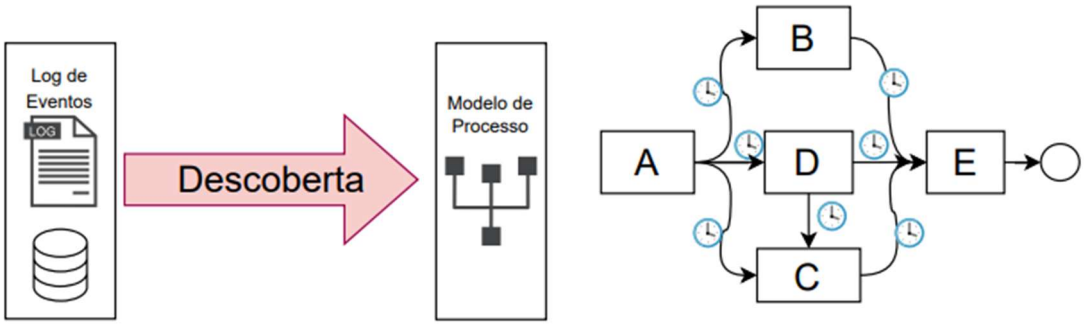


Fonte: adaptado de (1).

As três perspectivas fundamentais da mineração de processos são: descoberta, verificação de conformidade e aprimoramento. Há também uma quarta perspectiva, denominada "suporte operacional", que desempenha um papel complementar, porém igualmente importante.

Descoberta: esta é a fase inicial do processo, e envolve a criação de um modelo de processo a partir de dados brutos, apresentados na forma de um registro de evento. A descoberta é o principal objetivo da mineração de processos, que busca extrair informações relacionadas ao processo, automaticamente revelando um modelo *as-is*, que representa conforme ele ocorre na prática. Uma vez que um modelo *as-is* tenha sido estabelecido, torna-se possível reproduzir os eventos ocorridos para fins de verificação de conformidade e identificação de gargalos, entre outros aspectos (12). A Figura 3 ilustra as etapas da descoberta do processo.

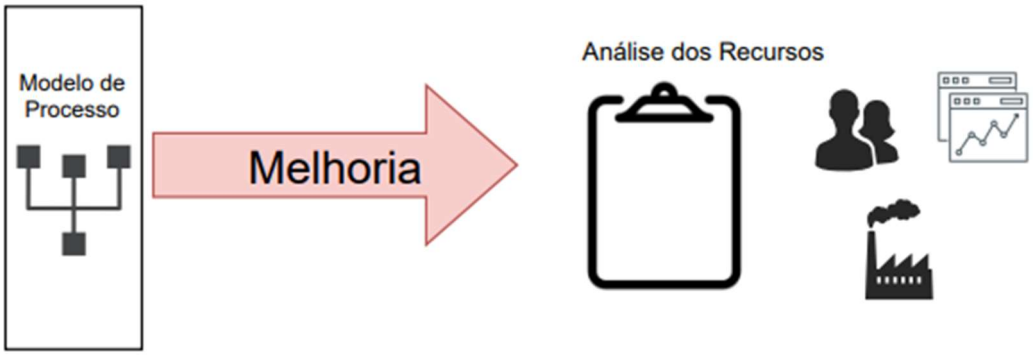
Figura 3 – Descoberta de um modelo de processo *as-is*



Fonte: adaptado de (1).

Melhoria: a perspectiva de melhoria/aprimoramento diz respeito à expansão de um modelo de processo por meio da adição de informações relativas a diferentes aspectos ou da correção de problemas, como gargalos (13). O aprimoramento pode ocorrer mediante a correção do modelo de processo para otimizar seu desempenho, ou pela sua expansão com informações adicionais, introduzindo uma nova dimensão ao processo. De forma análoga à verificação de conformidade, a perspectiva de aprimoramento também depende de um *log* de eventos e de um modelo de processo como entrada. A Figura 4 apresenta as etapas envolvidas no processo de melhoria.

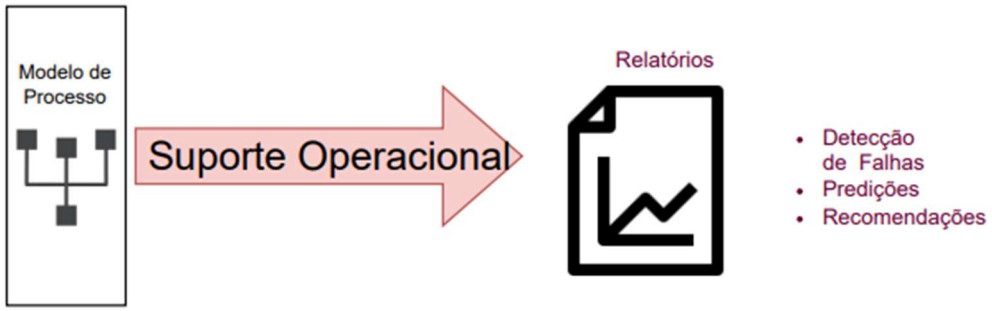
Figura 4 – Melhoria de processo de negócio



Fonte: adaptado (1).

Suporte Operacional: a perspectiva de Suporte Operacional representa uma área de suporte essencial no contexto da mineração de processos. Ela desempenha funções cruciais, como a compreensão dos conceitos-chave, a aquisição de dados de eventos, a consideração de métodos legados para viabilizar a mineração de processos, a utilização de arquiteturas e ferramentas existentes, além da análise de possíveis ontologias e o estudo das lições aprendidas em aplicações anteriores de mineração de processos (14). Muitas atividades realizadas *on-line* podem ser analisadas com o auxílio dessas ferramentas de mineração de processos, mesmo que essa técnica seja aplicável a eventos que já foram concluídos e armazenados em bases de dados transformados em *log* (1). Um exemplo de suporte operacional é evidenciado no estudo apresentado em (15). A Figura 5 apresenta as etapas envolvidas na perspectiva de Suporte Operacional.

Figura 5 – Suporte operacional

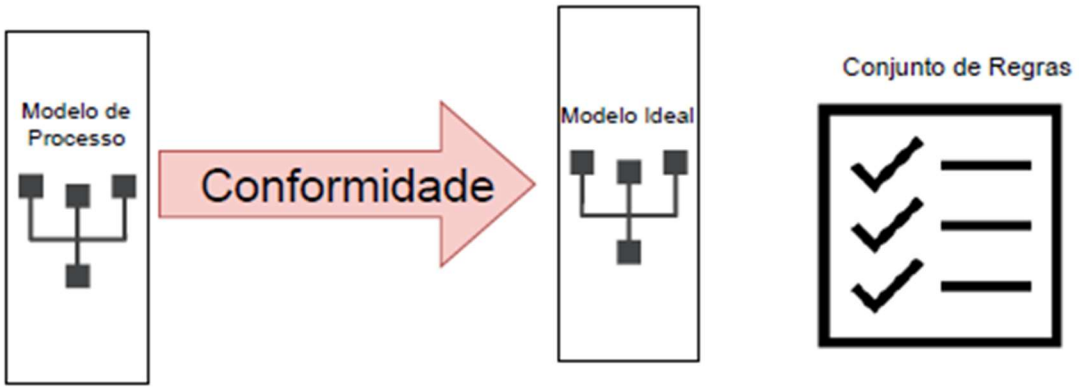


Fonte: adaptado (1).

Em resumo, a implementação das tarefas de mineração de processos tem como objetivo analisar um processo, identificar os gargalos e comparar ou descobrir as variantes, com ênfase na sugestão de melhorias.

Conformidade: a perspectiva de verificação de conformidade é uma área da mineração de processos que oferece técnicas para comparar instâncias de um processo com um modelo seu descoberto ou projetado (1). Ao contrário da descoberta, a verificação de conformidade utiliza um *log* de eventos e um modelo como entrada. O propósito dessas técnicas é avaliar se o comportamento modelado reflete, de forma precisa, o comportamento observado no *log* de eventos (16). A Figura 6 ilustra as etapas relacionadas à verificação de conformidade.

Figura 6 – Conformidade em um modelo de processo



Fonte: adaptado de (1).

O propósito da verificação de conformidade é verificar se o registro de eventos está em conformidade com o modelo e vice-versa, e tem como intuito a identificação de possíveis desvios indesejáveis que possam indicar fraudes, ineficiências ou mudanças no processo. Além disso, a verificação de conformidade pode ser empregada para avaliar o desempenho do

processo analisado, independentemente de o modelo ter sido construído manualmente ou descoberto automaticamente. Em resumo, a verificação de conformidade estabelece a correspondência entre os eventos de um registro de eventos e as atividades de um modelo de processo, realizando comparações entre ambos (1).

Para tanto, este trabalho exemplifica duas abordagens específicas, fundamentadas em *tokens* e em alinhamento, para o cálculo de estatísticas de conformidade.

Token: a abordagem baseada em *token* emprega um modelo no formato de rede de Petri, que parte de um ponto de início e percorre caminhos até alcançar um ponto de término. Durante esse percurso, o cálculo de conformidade identifica *tokens* remanescentes ou ausentes no processo em relação ao modelo. Cada vez que o modelo alcança um ponto sem saída durante a execução, antes de poder concluir a análise, um *token* adicional é gerado para avançar o estado atual para o próximo estágio. Além disso, a abordagem agrega *tokens* que permanecem no modelo após a conclusão do rastreamento do registro de eventos. Ao final, o algoritmo que implementa essa abordagem determina a conformidade do processo com base na soma de todos os *tokens* remanescentes e gerados (17).

A reprodução baseada em *token* é uma técnica importante para a *verificação de conformidade*. Um traço se conforma ao modelo quando, durante sua execução, as transições podem ser acionadas sem a necessidade de inserir *tokens* ausentes. Exemplos que mostram os passos dessa abordagem podem ser encontrados no trabalho desenvolvido em (18).

Alinhamento: a abordagem baseada em alinhamento visa explorar todas as possibilidades de alinhamento entre um traço e um modelo de processo, escolhendo o alinhamento ideal, ou seja, aquele que possui o menor custo. Os alinhamentos de traço permitem expressar desvios no nível do evento, possibilitando a configuração do custo associado a cada atividade omitida ou adicionada. Isso permite a realização de um cálculo de conformidade mais flexível. **Aqui**, o modelo de processo serve como base para o alinhamento de cada traço percorrido pelo caso. Um exemplo prático seria alinhar o traço de um paciente em tratamento de uma determinada patologia, com suas diretrizes específicas.

A seguir, apresenta-se alguns exemplos de alinhamento:

Exemplo 1: verifica-se se o Traço 1, quando submetido ao alinhamento, ajusta-se (*fits*) perfeitamente ao Modelo (Tabela 1).

Tabela 1 – Exemplo 1 de alinhamento

Modelo	A	→	B	→	C	→	E	→	F
Trace 1	A	→	B	→	C	→	E	→	F

Fonte: Autor.

Exemplo 2: no alinhamento do Traço 2, identifica-se um ponto no qual o traço contém uma atividade que não está prevista no Modelo, denominado de *move on log only* (Tabela 2).

Tabela 2 - Exemplo 2 de alinhamento

Modelo	A	→	B	→	C	→	>>	→	E	→	F
Trace 2	A	→	B	→	C	→	D	→	E	→	F

Fonte: Autor.

Exemplo 3: Durante o alinhamento, verifica-se se o modelo contém atividades que não foram executadas no Traço 3. Quando o Modelo requer uma atividade não registrada no traço, essa situação é denominada de *move on model only* (Tabela 3).

Tabela 3 - Exemplo 3 de alinhamento

Modelo	A	→	B	→	C	→	E	→	F
Trace 3	A	→	>>	→	C	→	>>	→	F

Fonte: Autor.

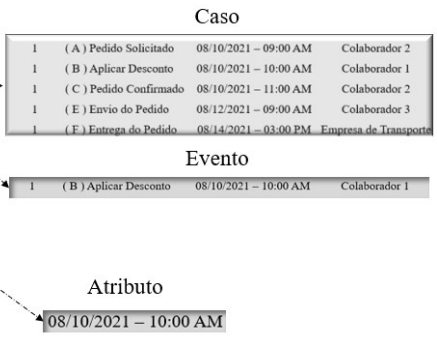
2.2 *LOG* DE EVENTOS

As fontes de dados utilizadas na mineração de processos consistem em logs de eventos. Um *log* é composto por várias instâncias de um processo, também conhecidas como “casos”. Cada caso é constituído por um conjunto ordenado de eventos, em que cada um representa uma instância de uma atividade ou etapa do processo. Além disso, cada evento está associado a um ponto no tempo ou a um intervalo temporal – com uma data de início e fim. Por exemplo, o registro da entrada de um paciente em um hospital e todas as ações subsequentes que ele realiza durante a sua estadia podem ser considerados como um caso. Outros exemplos incluem a compra de um produto em uma loja física ou *on-line*, desde o pedido até a entrega do produto; as etapas de uma máquina em uma linha de montagem, como ligar, programar, executar e desligar; a submissão da declaração de imposto por um cidadão; a identificação de uma falha em um servidor da *web*; entre muitos outros exemplos de atividades compostas por várias etapas, que podem ser atribuídas a uma ou mais pessoas ou sistemas. O trabalho de referência (19) nos fornece *insights* que servem de base para a criação do exemplo.

Assim, na Tabela 4 é exposto o histórico de um processo de venda de produtos *on-line*, em que cada transação de venda representa um caso ou instância do processo. Ademais, cada etapa, como "Pedido Solicitado" ou "Aplicar Desconto", é considerada um evento, ambos pertencentes ao mesmo caso. Em outras palavras, cada evento é definido por sua atividade e seus atributos, sendo alguns destes indispensáveis para a análise do processo, como a data e a hora da execução do *Aplicar Desconto*.

Tabela 4 - Exemplo de log de eventos

Número do Pedido	Atividade	Data e Hora	Recurso
1	(A) Pedido Solicitado	08/10/2021 – 09:00 AM	Colaborador 2
1	(B) Aplicar Desconto	08/10/2021 – 10:00 AM	Colaborador 1
1	(C) Pedido Confirmado	08/10/2021 – 11:00 AM	Colaborador 2
1	(E) Envio do Pedido	08/12/2021 – 09:00 AM	Colaborador 3
1	(F) Entrega do Pedido	08/14/2021 – 03:00 PM	Empresa de Transporte
2	(A) Pedido Solicitado	08/10/2021 – 09:30 AM	Colaborador 2
2	(D) Alterar Pedido	08/10/2021 – 09:40 AM	Colaborador 2
2	(C) Pedido Confirmado	08/10/2021 – 10:30 AM	Colaborador 2
2	(E) Envio do Pedido	08/12/2021 – 09:00 AM	Colaborador 3
2	(F) Entrega do Pedido	08/14/2021 – 11:00 AM	Empresa de Transporte
3	(A) Pedido Solicitado	08/11/2021 – 10:30 AM	Colaborador 2
3	(C) Pedido Confirmado	08/11/2021 – 11:30 AM	Colaborador 2
3	(F) Entrega do Pedido	15/11/2021 – 11:30 AM	Empresa de Transporte
4	(A) Pedido Solicitado	09/11/2021 – 10:30 AM	Colaborador 2
4	(G) Cancelamento Pedido	09/11/2021 – 11:30 AM	Colaborador 2



Fonte: elaborada pela autora.

O *log* de eventos em questão contém três instâncias de um processo de venda, sendo cada uma associada a uma venda específica. O número do pedido é utilizado para identificar cada uma dessas instâncias ou casos do processo. Cada caso tem início com a atividade *Pedido Solicitado*. Além disso, em alguns casos, pode ser aplicado um desconto, como exemplificado no Caso 1. Após o registro do pedido por um colaborador, o processo segue o fluxo de trabalho até a entrega do produto. É importante observar que cada caso possui um traço que registra a sequência de atividades realizadas. No exemplo apresentado na tabela anterior, os traços podem ser descritos conforme ilustrado na Tabela 5.

Tabela 5 - Exemplo de Traços

Case 1	A	→	B	→	C	→	E	→	F
Caso 2	A	→	D	→	C	→	E	→	F
Case 3	A	→	C	→	F				
Case 4	A	→	G						

Fonte: elaborada pela autora.

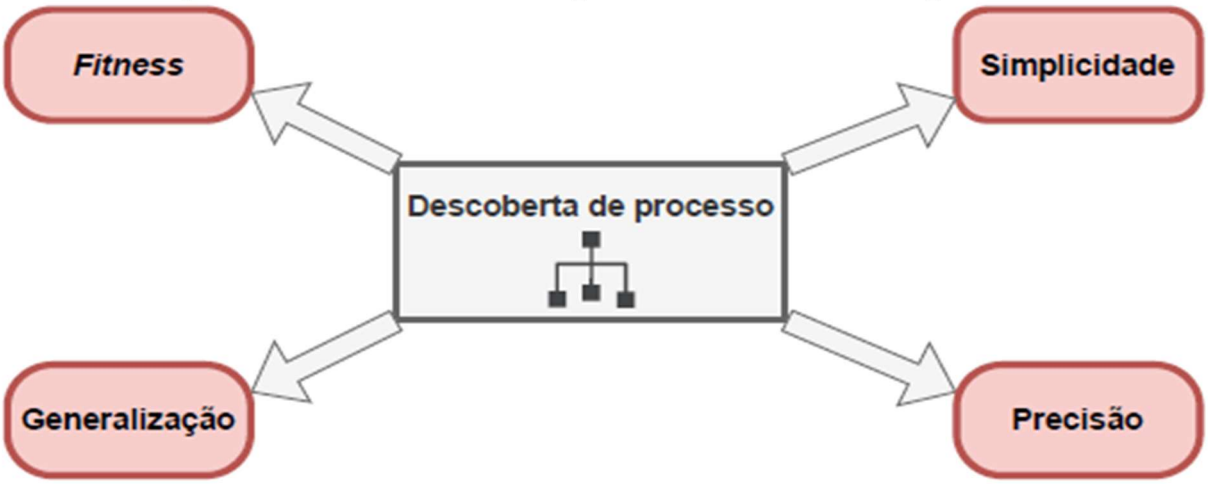
Um traço é definido como uma sequência finita de eventos no qual cada evento ocorre apenas uma vez no log. Por exemplo, o traço associado ao Caso 1 consiste em cinco eventos distintos. No *log* em questão, são registrados quatro casos, cada um com traços únicos para o mesmo processo. Essa coleção de traços serve como fonte de dados para um algoritmo de descoberta do modelo de instâncias do processo *as-is*, e a sua aplicação gera um modelo de processo sem a necessidade de conhecimento prévio. Além disso, é possível realizar uma avaliação objetiva desse modelo com base em suas dimensões de qualidade. A obra (1) – de referência – nos fornece *insights* que servem de suporte para a criação do exemplo.

2.3 DIMENSÕES DA QUALIDADE DE MODELOS DE PROCESSOS

Analogamente à mineração de dados, a análise da qualidade do resultado da mineração é crucial. Para alcançar uma verificação eficaz, requer-se a disponibilidade de dados e métricas de qualidade calculadas de forma objetiva. Nesse contexto, a qualidade de um modelo de processo pode ser medida por meio das quatro dimensões canônicas, a saber: *aptidão/fitness, generalização, precisão e simplicidade.* Enquanto a simplicidade é uma propriedade inerente ao modelo, as outras três dimensões relacionam o modelo ao *log* de eventos (20). É importante destacar, conforme (1), que a qualidade do modelo de um processo, especialmente ao lidar com a verificação de conformidade, não pode ser subjetiva.

A Figura 7 ilustra as quatro dimensões da qualidade do modelo de processo e sugere a existência de certo grau de antagonismo entre elas, como entre precisão e generalização.

Figura 7 - Quatro dimensões da qualidade de um modelo de processo



Fonte: adaptado de (1).

A aplicação combinada ou individual de cada métrica de qualidade de modelo de processo permite a quantificação objetiva do modelo analisado ao considerar a escolha ou preferência do analista de processo quanto ao grau de importância de cada dimensão de qualidade em sua análise.

2.3.1 Métrica *Fitness*

A métrica *fitness* está relacionada ao comportamento presente no *log* de eventos. Ela mede se este*log* está consistente com o modelo de processo, ou seja, verifica se o *log* pode ser reproduzido pelo modelo. *Van der Aalst* (1) explica que "um modelo tem *fitness* perfeito se todos os traços no *log* de eventos forem reproduzidos pelo modelo do início ao fim".

Basicamente, a métrica *fitness* verifica a consistência do *log* de eventos em relação ao modelo de processo escolhido, que pode ter sido descoberto ou definido de forma normativa. Com o valor da métrica, é possível realizar os seguintes diagnósticos:

• Local no *log* de eventos: verifica a existência de traços que não podem ser reproduzidos no modelo;

• Local no modelo de processo: identifica desvios em partes específicas do modelo;

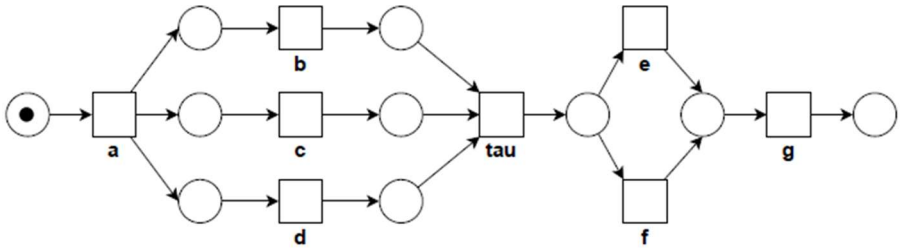
• Global: oferece uma análise abrangente do *log* de eventos em relação ao modelo, produzindo um valor entre 0 e 1. O valor 0 indica que nenhum traço pode ser reproduzido no modelo, enquanto o valor 1 indica que todos os traços podem ser reproduzidos.

Em geral, o cálculo da métrica *fitness* expressa a proporção de traços em um *log* de eventos que um modelo de processo pode reproduzir. É importante ressaltar que existem diversas abordagens para medir *fitness* (17).

2.3.1.1 Cálculo do *fitness* por alinhamento

Uma das maneiras de calcular essa métrica é por meio do alinhamento, que busca alinhar os traços com o modelo de processo. Esse método envolve a simulação de cada passo, verificando os movimentos tanto no traço quanto no modelo. A Figura 8 representa um exemplo de modelo de processo.

Figura 8 – Exemplo de modelo de processo



Fonte: adaptado de (21).

Uma das vantagens desse tipo de cálculo é a possibilidade de configurar o custo para cada atividade omitida ou extra, proporcionando um cálculo de *fitness* mais flexível. Ainda na Figura 8, observa-se estruturas de sequência paralela e de escolha.

Como exemplo, apresenta-se três traços a seguir: Traço 1, Traço 2 e Traço 3 (Tabela 6).

Tabela 6 – *Fitness* por alinhamento

Modelo	**a**	**b**	**c**	**D**	**e**	**>>**	**g**
Traço 1	a	b	c	D	e	>>	g
Traço 2	a	b	c	>>	e	>>	g
Traço 3	a	b	c	D	e	f	g

Fonte: adaptaddo de (21).

Em que:

Traço 1 <a, b, c, d, e, g> se alinha no modelo.

Traço 2 <a, b, c, e, g> não se alinha perfeitamente ao modelo, com desvio na atividade *d*. Quando o modelo precisa de uma atividade não observada no traço, ele é chamado de *movimento no modelo (move on model only)*.

O Traço 3 <a, b, c, d, e, f, g> não se ajusta perfeitamente ao modelo, apresentando um desvio na atividade "f".

Ademais, o Traço 3 também apresenta um desvio, pois inclui uma atividade não esperada no modelo, o que é chamado de movimento no *log* (*move on log only*). O exemplo anterior está publicado em (21), o qual utiliza o cálculo de *fitness* por alinhamento para verificação de conformidade.

A técnica de alinhamento visa, basicamente, equiparar o maior número possível de eventos de um traço às atividades presentes no modelo de processo analisado. Dependendo da necessidade, eventos podem ser ignorados ou atividades podem ser inseridas sem um evento correspondente presente no *log* de eventos. O cálculo da métrica *fitness* considera penalidades tanto para as inserções quanto para as ausências de eventos de um traço. A Fórmula F1 a seguir expressa o cálculo da métrica de qualidade de modelo *fitness* (20):

$$Q_{Fitness} = 1 - \frac{Custo_{ML}}{Custo_{MinMS}} \qquad \text{F1}$$

Onde,

$Custo_{ML}$ é o custo para alinhar o modelo e o *log* de eventos.

$Custo_{MinMS}$ é o custo mínimo para alinhar os traços de um *log* de eventos no modelo sem movimentos síncronos.

Para o cálculo, é necessário normalizar o custo de alinhamento para um valor entre 0 e 1, em que 1 representa o alinhamento perfeito e 0 o pior alinhamento – este ocorre quando não existe nenhum movimento síncrono entre o modelo e o traço analisado.

Como exemplo, considerando o modelo citado anteriormente com traço <a, b, c, d, e, g>, se não houver nenhum movimento síncrono entre o modelo e o traço analisado, tem-se um *movimento no log* com custo 6 e um *movimento no modelo* com custo 6. Portanto, o menor custo possível é a soma dos dois, que nesse caso é 12. Assim, 12 é o custo mínimo para alinhar o traço do evento no modelo. Tomando como exemplo a opção de alinhamento a seguir na Figura 9, e considerando que, para cada movimento (>>), o custo seja =1, tem-se:

Figura 9 – Exemplo de traço

Traço	**a**	**>>**	**b**	**c**	**d**	**e**	**g**
Modelo	a	c	b	>>	d	e	g

Fonte: Autor.

$Custo_{ML}$ = 2

$Custo_{MinMS}$ = 6 + 6

$$Q_{Fitness} = 1 - \frac{2}{12} = 0{,}83$$

O denominador da fórmula F1 representa o custo mínimo quando não há correspondência entre os traços do *log* de eventos e o modelo de processo. Apesar de sua simplicidade no cálculo do custo, este é relevante em muitas situações de verificação de conformidade. Um exemplo prático de aplicação de F1 é fornecido no trabalho que originou o exemplo anteriormente apresentado (21). Contudo, em (20) é mencionado que essa conta não é uma tarefa fácil, uma vez que é demorada, o que entra em conflito com o requisito de medição de eficiência de implementação. Ainda assim, essa métrica é considerada robusta para avaliar a correspondência entre os traços do *log* de eventos e o modelo de processo.

2.3.1.2 Cálculo do *Fitness* por repetição de *token*

Uma segunda maneira para calcular a métrica *fitness* é baseada em *tokens*, e envolve o rastreamento de um *log* de eventos e de um modelo de processo no formato de uma Rede de Petri. A saída desse rastreamento consiste em uma lista de transições habilitadas. Já a teoria detalhada sobre *token* de repetição pode ser encontrada em (1) no Capítulo 8, que trata sobre a verificação de conformidade, mais especificamente na Seção 8.2. No entanto, neste projeto, adotou-se uma versão específica da abordagem de cálculo da métrica *fitness* por *token* de repetição, conforme definida em (1), como ilustrado a seguir.

O modelo de processo gerado por algoritmos de descoberta a partir de um *log* de eventos é frequentemente representado por meio de uma Rede de Petri. Essas redes são usualmente utilizadas como modelos de entrada em simulações de processos, Para tanto, é fundamental notar que devem conter marcações de início e término. A definição de Redes de Petri a seguir baseia-se na literatura clássica (22).

Definição 1: Rede de Petri (22)

Uma rede definida por $PN = (P, T, F, W, M_0, M_F, l)$, que estende uma rede elementar, de modo que:

- (P, T, F) é uma rede, em que P e T são conjuntos finitos disjuntos de lugares e transições, e $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos;
- $W : F \rightarrow \mathbb{N}$ é um multiconjunto de arcos, de modo que a contagem para cada um é uma medida da multiplicidade do arco;
- $M_0 : P \rightarrow \mathbb{N}$ é a marcação inicial;

- $M_F : P \rightarrow \mathbb{N}$ é a marcação final;
- $l : T \rightarrow \sum \cup \{\tau\}$ é uma função de rotulagem que atribui a cada transição $t \in T$ qualquer símbolo de $\sum$ (o conjunto de rótulos) ou a *string* vazia τ.

A Figura 8, anteriormente apresentada, ilustra um modelo de processo na forma de uma *Rede de Petri*. Nesta, são identificadas transições visíveis únicas, uma marcação *inicial*, que reflete o estado inicial de execução do processo, e uma marcação *final*, que indica o estado de término do processo.

Considere a semântica de execução de uma Rede de Petri da seguinte maneira: uma transição t ∈ T é habilitada (ou seja, pode disparar) em uma marcação M se houver *tokens* suficientes em seus locais de entrada para que os consumos sejam possíveis, ou seja, se ∀s ∈ •t: M(s) ≥ W(s, t). Disparar uma transição t ∈ T na marcação M consome W (s, t) *tokens* de cada um de seus lugares de entrada e produz W (t, s) *tokens* em cada um de seus locais de saída.

Na aplicação da reprodução baseada em *token*, um rastreamento é realizado em um *log* de eventos e em um modelo de *Rede de Petri*. A saída dessa operação de repetição é uma lista das transições habilitadas durante o rastreamento, juntamente com um conjunto de valores, em que:

- c é número de *tokens* consumidos durante sua repetição;
- p é número de *tokens* produzidos;
- m é número de *tokens* ausentes;
- r é número de *tokens* restantes;

Durante o processo de repetição de *tokens*, as relações mantidas são: $c \leq p + m$ e $m \leq c$. Ao final desse processo, a relação mantida é $p + m = c + r$. O valor do *fitness* para cada traço é calculado pela Fórmula F2 a seguir.

$$F_{traço} = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right) \qquad \text{F2}$$

A métrica também é calculada para o conjunto de casos de um *log* de eventos L, seja L_i um caso de L, c_i o número de *tokens* consumidos, p_i o número de *tokens* produzidos, M_i o número de tokens ausentes e r_i o número de *tokens* restantes. O valor do *fitness* para o conjunto de casos de um *log* de eventos é calculado pela Fórmula F3.

$$F_{Log} = \frac{1}{2}\left(1 - \frac{\sum_{L_i \in L} m_i}{\sum_{L_i \in L} c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{L_i \in L} r_i}{\sum_{L_i \in L} p_i}\right) \qquad \text{F3}$$

Pode-se afirmar, portanto, que a métrica *fitness* é de suma importância no contexto da mineração de processos.

2.3.2 Métrica de Precisão

A métrica *Precisão* refere-se ao grau mais restrito de adesão do processo em comparação com o registro de eventos. Ela avalia se o modelo de processo em análise apresenta uma generalização excessiva do comportamento observado no registro de eventos, caracterizando um modelo de baixa precisão como *underfitting*. Por outro lado, quando um modelo não consegue generalizar adequadamente o comportamento observado, isso é denominado *overfitting* (23). Portanto, a dimensão de qualidade denominada *Precisão* presume a existência de uma extensão do comportamento permitido pelo modelo de processo que não está refletida no registro de eventos devido à ampla variedade de possibilidades inerentes a ele durante iterações cíclicas. Logo, o resultado do cálculo proporciona uma estimativa da precisão (20).

Assim como a métrica *Fitness*, existem diversas formas para o cálculo da métrica *Precisão* – a seguir, será descrita uma delas. É importante salientar que, de modo geral, o cálculo realizado para o alinhamento é considerado mais preciso, embora o cálculo por *token* de repetição seja mais eficiente.

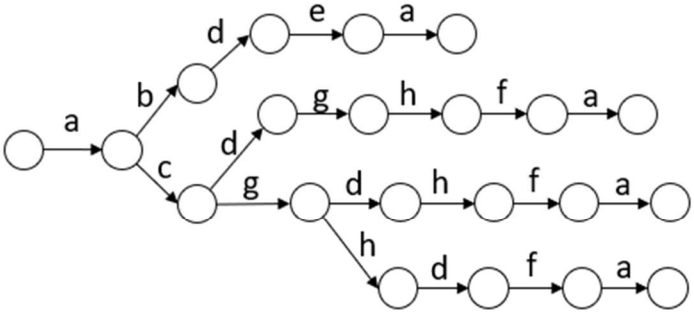
A abordagem para o cálculo da precisão por alinhamento requer a utilização de um modelo – neste caso, uma *Rede de Petri* – e um *log* de eventos (24). O método de cálculo é denominado *ETConformance* (24), e segue os seguintes passos:

Passo 1: mapear o comportamento do *log* de eventos para obter o conjunto de autômatos possíveis, levando em conta cada traço presente no *log* de eventos (Figura 10).

Um autômato constitui uma representação formal de um processo, ou máquina de estados, capaz de transitar entre diferentes estados. Neste contexto, eles são gerados a partir de cada variante do processo, conforme exemplificado a seguir.

Figura 10 – Mapeamento de traços para autômatos

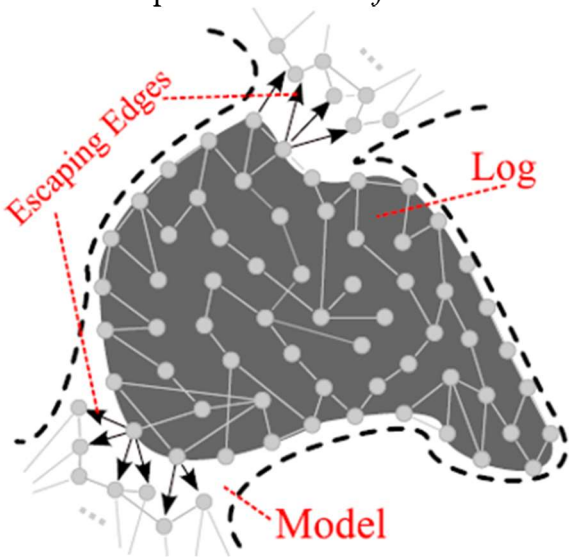
Log
A B D E A
A C D G H F A
A C G D H F A
A C G H D F A



Fonte: adaptado de (24).

Passo 2: identificar as bordas de escape, denominadas E_E (*Escaping Edges* em inglês), que representam situações em que o modelo permite mais comportamentos do que os observados no *log* de eventos a partir de seus traços/casos, resultando em uma diminuição da *Precisão*. A Figura 11 ilustra, de forma gráfica, o conceito central das bordas de escape.

Figura 11 – Mapeamento *ETConformance*



Fonte: extraída de (24).

Definição de bordas de *escape* (24): considere um estado do autômato TS = (S, T, A, s_0) do *log* de eventos EL – em que S é um conjunto de estados, T é um conjunto de transições, A é um conjunto de transições rotuladas e s_0 é inicial – e PN = (P, T, W, M_0) como um modelo de processo no formato de uma *Rede de Petri*, σ é um traço. A borda de *escape* E_E é definida pela

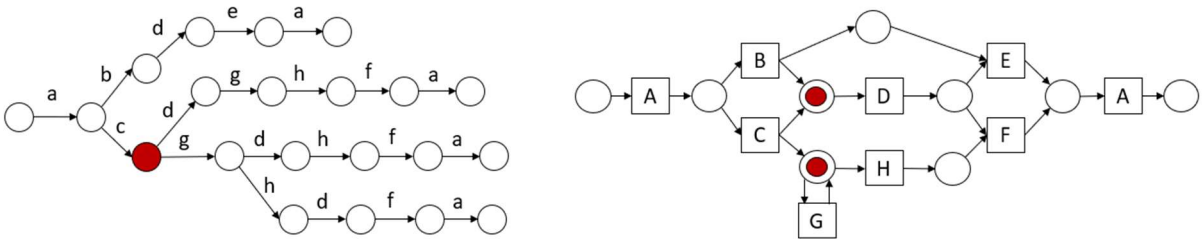
Fórmula F4, e a precisão $etcp$ (EL, PN) leva em consideração as bordas de *escape* definidas por ela.

$$E_E(s) = A_T(s) - R_T(s) \qquad \text{F4}$$

Em que, A_T é o conjunto de transições permitidas, R_T é o conjunto de transições refletidas e s é o estado.

Exemplo de cálculo da borda de *escape*: cada estado/evento do *log* é mapeado no modelo PN (cf. parte direita da Figura 12) e assume o valor da métrica *Fitness* =1, cf. Figura 12.

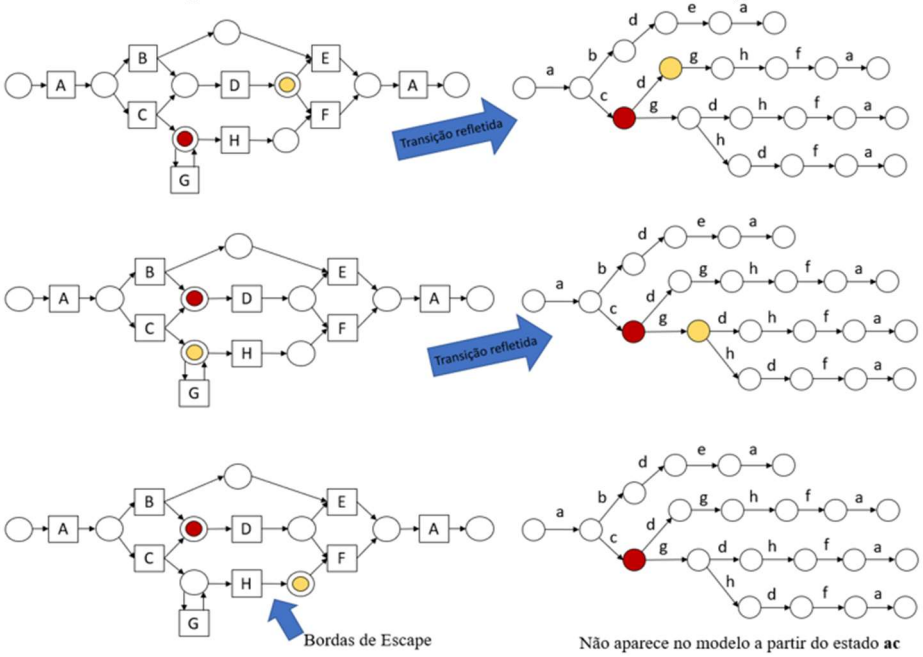
Figura 12 – Mapeamento no modelo



Fonte: adaptado de (24).

A partir daí, são verificadas as próximas transições possíveis no modelo e identificadas as bordas de *escape*, conforme Figura 13.

Figura 13 – Identificação das Bordas de Escape



Fonte: adaptado de (24).

Sendo $E_E(s)$ a quantidade de bordas de *escape* por estado, A_T o número de transições permitidas no modelo e R_T o número de transições refletidas no autômato, em que é possível aplicar à Fórmula F4, mostrada anteriormente. No exemplo: $E_E(ac) = 3 - 2 = 1$.

Após a obtenção dos valores de bordas de *escape,* o cálculo da métrica de *Precisão* pode ser aplicado utilizando a Fórmula F5.

$$etcp(\mathrm{EL}, \mathrm{PN}) = 1 - \frac{\sum_{i=1}^{|\mathrm{EL}|} \sum_{j=1}^{|\sigma_i|+1} | E_E(s_j^i) |}{\sum_{i=1}^{|\mathrm{EL}|} \sum_{j=1}^{|\sigma_i|+1} | A_T(s_j^i) |} \qquad \text{F5}$$

A definição de autônomo *prefixado*, dada por (24), envolve: um *log* de eventos EL e um autômato derivado TS = $(S, T, A. s_0)$, em que S é um conjunto de estados de EL, $\sigma_i = t_1 \ldots t_{|\sigma i|} \in$ EL é um traço, e $S_j^i \in S$ é um estado em TS correspondente ao prefixo $t_1 \ldots t_{j-1}$, para $0 \leq j \leq |\sigma_i| + 1$. Na Figura 10, apresentada anteriormente, cada traço é mapeado para a criação do autômato prefixado, exemplificando, assim, o processo de transformação.

Para tanto, a abordagem *ETConformance* baseia-se no princípio de *token replay*, em que um conjunto de transições habilitadas no modelo de processo é comparado com o conjunto de atividades que seguem o prefixo – este composto por partes do caminho percorrido no traço. Por exemplo, "a, b, d" constitui um prefixo do primeiro traço apresentado na Figura 10.

Quanto mais distintos forem esses conjuntos, menor será o valor da P*recisão*. Por outro lado, quanto mais semelhantes forem, maior será a *precisão*.

Existem outras formas para calcular a métrica de *precisão*, e a principal diferença reside no método de repetição adotado. É relevante, então, observar que nossa preferência pelo *replay* baseado em *token* é devido à sua eficiência, embora ele se baseie em heurísticas. Em contrapartida, é importante notar que o cálculo por alinhamento é exato e aplicável a qualquer tipo de rede, mas é ineficiente em termos de custo computacional e de tempo.

2.3.3 Métrica de Generalização

A terceira dimensão a ser analisada é a *generalização*, que está relacionada à capacidade do modelo de processo em generalizar o comportamento observado. Essa dimensão determina em que medida o modelo não está restrito ao comportamento da amostra e se é capaz de descrever adequadamente o sistema real. Além disso, a generalização pode ser capaz de representar o comportamento não observado anteriormente no sistema (20).

No estudo descrito em (20), a medida de generalização foi baseada na frequência de uso das partes do modelo de processo durante a reprodução do *log* de eventos, considerando a estrutura – em árvore – na representação do modelo. A notação em formato de árvore de processo pode ser traduzida para outras modelagens de processo, como BPMN, *Redes de Petri* e outras. A escolha dessa notação, contudo, é motivada pela sua facilidade de interpretação e pela viabilidade de tradução para a notação de *Redes de Petri*.

Ademais, baseou-se no resultado da métrica *Fitness* para verificar a frequência de visitação de um nó, destacando, assim, se o seu comportamento está correto ou incorreto. Se determinadas seções da árvore são raramente visitadas, isso indica uma baixa capacidade de generalização. Portanto, é importante notar que, se todas as partes do modelo de processo são frequentemente percorridas, o modelo provavelmente é genérico.

A Fórmula F6, apresentada a seguir, define o cálculo da métrica de qualidade de *generalização* para um modelo de processo.

$$Q_{Generalização} = 1 - \frac{\sum_{nós} \left(\sqrt{\#execuções}\right)^{-1}}{\#nós\ na\ árvore} \qquad \text{F6}$$

O inverso (expoente -1) da raiz quadrada é aplicado ao número de execuções para normalizar os valores no intervalo (0, 1). É relevante observar que o efeito de aumentar de 1

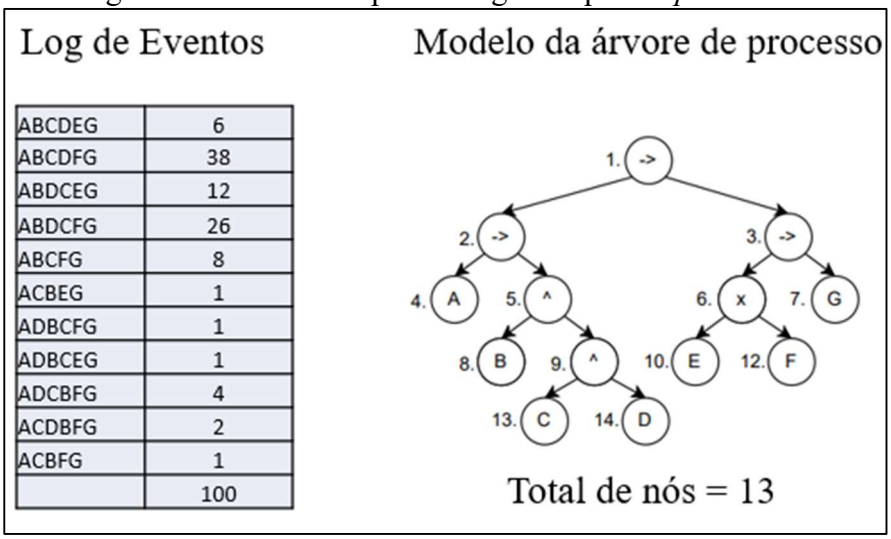
execução para 10 é considerado mais significativo do que aumentar de 10 execuções para 100 (20).

Outrossim, o número de nós de cada árvore é somado e, em seguida, dividido pelo número total de nós nas execuções do conjunto de árvores, resultando em um valor médio. Embora a métrica de generalização seja complexa de se expressar, ela é importante como uma dimensão de qualidade para modelos de processos de negócios. Um modelo altamente genérico é pouco preciso, enquanto um altamente específico é menos flexível em relação à reprodução e à aceitação de comportamentos diversos.

Exemplo 1: Cálculo da métrica de generalização (cf. (20)).

Considere o modelo de *log* de eventos a seguir e a árvore do processo associada, gerada pelo *Alpha Miner,* ilustrada na Figura 14.

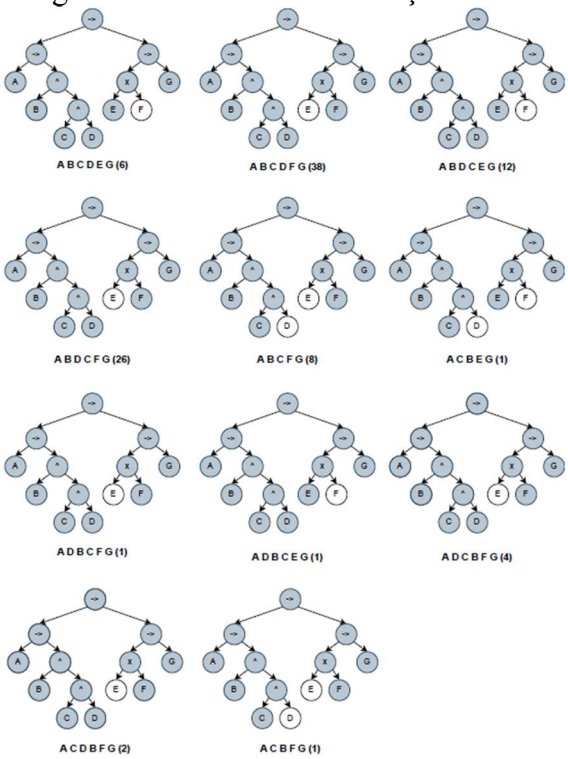
Figura 14 – Árvore de processo gerada pelo *Alpha Miner*



Fonte: adaptado de (20).

A Figura 15 apresenta as árvores de execuções do *log* de eventos.

Figura 15 – Árvores de execuções



Fonte: adaptado de (20).

A Figura 16, por sua vez, detalha o cálculo das execuções e, ao lado, a aplicação da fórmula e o resultado.

Figura 16 – Cálculo da métrica de generalização baseada no número de nós e execuções

Nodes	*# executions*	$(\sqrt{\#\ executions})^{-1}$
1 (->)	100	0.1
2 (->)	100	0.1
3 (->)	100	0.1
4 (A)	100	0.1
5 (^)	100	0.1
6 (x)	100	0.1
7 (G)	100	0.1
8 (B)	100	0.1
9 (^)	100	0.1
10 (E)	20	0.223606798
11 (F)	80	0.111803399
12 (C)	100	0.1
13 (D)	90	0.105409255
	∑	1.440819452
	#nodes in tree	13

$$Qg = 1 - \frac{\sum nodes\ (\#executions)^{-1}}{\#nodes\ in\ tree}$$

$$Qg = 1 - \frac{(1.440819452\ )}{13}$$

Resultado= **0.889**

Fonte: adaptado de (20).

A próxima dimensão de interesse de um processo é a simplicidade.

2.3.4 Métrica de Simplicidade

A *simplicidade* representa a quarta dimensão da qualidade de um modelo. A métrica associada a ela visa quantificar o grau de *simplicidade* do modelo, que pode ser avaliado com base em suas propriedades estruturais e comportamentais com o objetivo de determinar o quão facilmente compreensível é o modelo de processo para os seres humanos. Em termos de escolhas de modelos para descrever um processo, optar pelo mais simples é geralmente aconselhável, especialmente no que se refere à facilidade de compreensão. Em algumas situações, a simplificação de um modelo complexo pode ser alcançada alterando seu comportamento.

A métrica de *simplicidade* compara o tamanho da árvore do modelo com o número de atividades no *log* de eventos, e considera as suas características. Essa métrica é frequentemente utilizada para refletir as preferências do usuário, e é importante notar que seu cálculo influencia diretamente outras métricas (20).

Assim como nas dimensões mencionadas anteriormente, existem diversas abordagens para medir o grau de simplicidade de um modelo de processo.

Portanto, a escolha do modelo mais simples é a abordagem preferencial, como expresso na *Navalha de Occam*, que estabelece que "não se deve aumentar, além do necessário, o número de entidades necessárias para explicar um conceito" (20).

A estratégia de implementação da métrica de simplicidade é fundamentada na estrutura de uma *Rede de Petri*, em que o principal critério é o inverso do grau dos arcos da rede ou do grafo que descreve o modelo de processo (25). O resultado desse cálculo é um valor no intervalo entre 0 e 1, que corresponde à média dos graus de entrada e saída (26). Portanto, o cálculo da métrica de simplicidade é realizado por meio da comparação dos graus médios ponderados dos arcos/arestas do modelo descoberto em relação ao original. Quanto maior for a diferença, mais complexo será o modelo descoberto.

A Fórmula F7 a seguir define o procedimento de cálculo da dimensão da simplicidade.

$$Q_{Simplicidade} = \frac{1}{1 + \max\{0, S'_{M_o} - S'_{M_m}\}} \qquad \text{F7}$$

Em que,

S'_M corresponde ao grau médio de arco ponderado do modelo.

M_o é modelo de processo original, ou modelo conhecido.

M_m é modelo de processo descoberto ou minerado.

A avaliação da qualidade de um modelo de processo e a busca por um modelo mais simples representam um desafio constante (27). O melhor modelo é aquele que é mais simples e, ao mesmo tempo, capaz de explicar o comportamento registrado no *log* de eventos (25).

2.4 CRITÉRIOS DE ANÁLISE DAS QUATRO DIMENSÕES DE QUALIDADE

Quando se realiza a análise de um processo, considerando as dimensões da qualidade, uma tarefa desafiadora surge: como combinar ou selecionar as dimensões de qualidade mais apropriadas para a análise desejada? Diversos problemas podem surgir nesse contexto: os logs de eventos frequentemente não são completos em relação às possibilidades de traços, ou seja, não abrangem todos os comportamentos possíveis. Além disso, nos modelos, é possível identificar um grande número de traços diferentes, especialmente na presença de *loops*. Adicionalmente, existe a possibilidade de que traços ocorram com frequências diversas, ou seja, alguns podem ocorrer com baixa frequência, enquanto outros aparecem repetidamente e em grande número. Por fim, não se pode garantir que todos os traços possíveis estejam presentes no *log* de eventos (2).

As quatro dimensões de qualidade são, em certa medida, concorrentes entre si, o que torna o equilíbrio entre *fitness, simplicidade, precisão e generalização* um DESAFIO. Isso dificulta a avaliação da qualidade de um processo como alta, média ou baixa. No caso de um modelo de processo de negócios, portanto, é essencial compreender as características de cada dimensão para avaliar a qualidade do modelo, de acordo com as metas estabelecidas.

Para tanto, é necessário combinar as dimensões com base nas preferências e nos objetivos específicos. Entretanto, faz sentido considerar *precisão, generalização e simplicidade* somente se o *fitness* for aceitável (20). Nesse contexto, a primeira verificação a ser feita é se o modelo descoberto corresponde ao processo real. Os resultados das métricas podem fornecer um perfil do modelo e oferecer informações que permitem a um especialista de domínio fazer escolhas informadas.

A primeira dimensão, então, a ser considerada é a métrica *fitness*, que avalia o quão bem representado é o comportamento presente no *log* de eventos pelo modelo de processo. Por isso, é importante reconhecer, desde o início, que é desafiador obter um modelo perfeito ou

determinar qual o melhor; no entanto, é válido observar que, quando as métricas são combinadas, elas tendem a fornecer indicativos de modelos que atendem às necessidades de qualidade de um processo.

Ainda no contexto da métrica *fitness*, existem três tipos de análises que podem auxiliar na tomada de decisão sobre a preferência por um determinado modelo de processo e no entendimento da justificativa por trás de cada métrica de qualidade de modelo. Assim é importante salientar que a escolha do tipo de análise é uma decisão do analista de processo, uma vez que a sua configuração depende dos objetivos pretendidos.

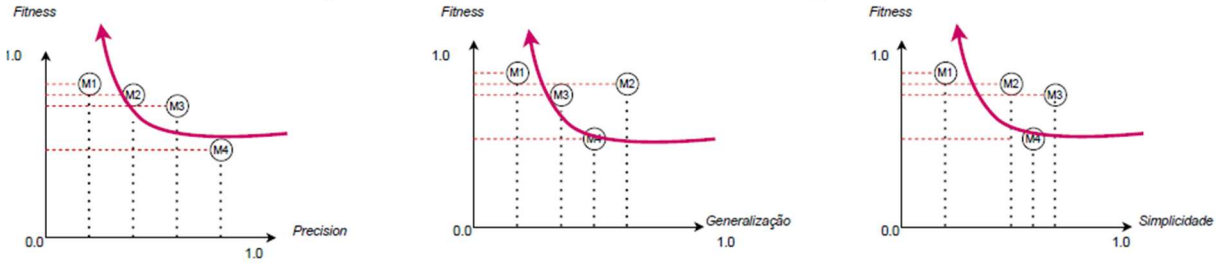
Tipos de análises: Para avaliar a qualidade de um modelo de processo, é necessário estabelecer as dimensões de qualidade e suas respectivas métricas a fim de atingir um objetivo específico de análise. Neste estudo, foi proposta a realização de três análises relacionadas à qualidade do modelo:

1. Análise *FxP*, que estabelece uma relação entre as métricas *Fitness e Precisão*;
2. Análise *FxS,* que estabelece uma relação entre as métricas *Fitness e Simplicidade*;
3. Análise *FxG,* que estabelece uma relação entre as métricas *Fitness e Generalização.*

Cada tipo de análise proposta é uma sugestão de representação gráfica, uma vez que não existe uma regra definida para a comparação dos valores das métricas. A ideia subjacente é que, ao identificar um ponto de inflexão, o analista possa visualmente comparar tais valores.

A Figura 17 exemplifica as análises *FxP, FxG* e *FxS* por meio de um gráfico 2D, que ilustra a aplicação das métricas de qualidade em quatro modelos de processo: M1, M2, M3 e M4. Essa representação visual permite observar a contribuição de cada métrica em uma análise de *ganho/perda, perda/ganho* ou *perda/perda*. O objetivo principal é disponibilizar ferramentas que facilitem a realização de uma análise mutuamente benéfica (*ganha/ganha*).

Figura 17 – Três análises – Métricas de qualidade



FxP (1) FxG (2) FxS (3)

Fonte: elaborada pela autora.

Em uma abordagem inicial, a escolha entre as análises *FxP, FxG e FxS* pode ser realizada mediante uma decisão direta do analista. Alternativamente, é possível oferece-lo a oportunidade de expressar suas preferências em um contexto de comparação *par-a-par*. Posteriormente, por meio de um método de cálculo de preferência, determinar qual delas reflete eficazmente as inclinações do analista em relação às dimensões de qualidade do modelo.

No âmbito deste projeto, a principal motivação para a aplicação da análise de qualidade é a busca objetiva pela avaliação de qualidade do modelo de processo identificado nas proximidades de um ponto de mudança. Em outras palavras, o objetivo é responder, de forma objetiva, à seguinte pergunta: qual é a qualidade do modelo de processo descoberto a partir do *sublog* de eventos formado pelo ponto de mudança e sua vizinhança recente? A extensão dessa vizinhança é representada por meio de uma janela deslizante de tamanho variável, a qual é determinada pelo método de detecção de pontos de mudança de conceito/processo.

2.5 MUDANÇA DE CONCEITO

A noção de mudança de conceito ou de processo é um conceito fundamental amplamente explorado na subárea da inteligência artificial, conhecida como “aprendizado de máquina”. A pesquisa e a aplicação de detecção de mudanças de conceito na aprendizagem de máquina abrangem tanto estratégias supervisionadas quanto não supervisionadas. Neste contexto, o interesse reside em adaptar as técnicas e concepções de detecção de mudanças para processos operacionais de negócios.

É crucial ressaltar que, em muitos casos, a motivação subjacente a uma mudança não é previsível antecipadamente. Diversos eventos têm o potencial de desencadear alterações em um processo, seja uma única modificação ou múltiplas mudanças simultâneas. Essas transformações podem ocorrer em resposta a diferentes estímulos, tais como ajustes sazonais –

para se adequarem a circunstâncias recorrentes –, mudanças na política organizacional, adaptações necessárias para atender às flutuações na oferta e demanda, ou modificações essenciais no atendimento hospitalar diante de situações críticas, como pandemias ou desastres. Além disso, tais mudanças podem ser desencadeadas por circunstâncias criminosas, como desvios ou fraudes (28).

Já a abordagem proposta no estudo (29) enfatiza, em sua investigação, a significativa consideração que deve ser concedida quando o escopo da análise envolve a aplicação de técnicas de aprendizado de máquina, especialmente em ambientes suscetíveis a mudanças. Destaca-se a relevância de reconhecer que, caso as alterações nos dados coletados não sejam adequadamente contempladas, os resultados decorrentes das análises subsequentes podem ser comprometidos. Em virtude da atual importância desse cenário, observa-se, então, um crescente interesse e empenho de pesquisa no tratamento de dados em contextos abertos e dinâmicos.

Inicialmente, essa atenção concentrou-se na área de aplicação do aprendizado de máquina, com foco no tema da detecção de *mudanças de conceito.* Mais recentemente, a atenção se estendeu para a área de mineração de processos, com ênfase na detecção de *mudanças de processos*.

É relevante observar que a distinção entre *mudança de conceito* e *mudança de processo* é mais uma questão de complexidade prática relacionada à detecção, caracterização e localização de um ponto de mudança em um processo do que uma abordagem conceitualmente distinta ou inovadora nessas áreas de aplicação.

Entretanto, a detecção de uma mudança implica que, ao longo do tempo, as propriedades estatísticas relacionadas ao processo tenham mudado, tornando necessária a análise e adaptação a esse cenário (30). O problema da mudança de processo se manifesta em diversas situações do mundo real. Recentemente, grande parte dos processos de negócios nas organizações foi obrigada a se adaptar devido à pandemia, afetando, por exemplo, os procedimentos de companhias aéreas, comércio eletrônico, escolas, entre outros. Certamente, o período da pandemia representa o cenário mais relevante para a pesquisa e investigação sobre o tema.

Na mineração de processos, quando se identifica a descoberta de um modelo de processo dentro de um determinado intervalo de tempo, presume-se que este é estacionário, isto é, que permanece inalterado do início ao fim. Entretanto, essa não é sempre a situação, uma vez que o modelo *pode estar passando por mudanças*. Nesse contexto, a abordagem de detecção de variações auxilia na compreensão das possíveis causas e consequências dessas alterações. É importante destacar que esta pesquisa se baseia em estudos anteriores que abordam

especificamente aspectos relacionados ao entendimento da área de aplicação, os quais serão brevemente resumidos.

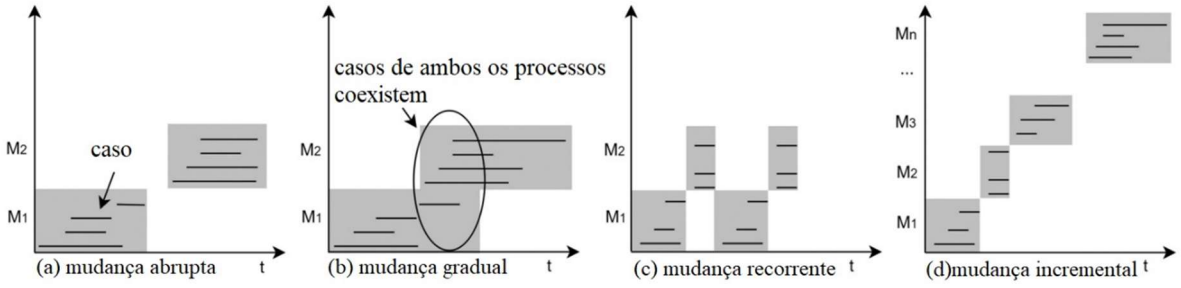
2.6 TIPOS DE MUDANÇA DE CONCEITO

Em um modelo de processo do mundo real, ao observar sua execução cotidiana, é possível identificar que as mudanças que ocorrem podem variar em natureza. Portanto, a compreensão delas requer a consideração de uma classificação apropriada.

As mudanças podem ser classificadas como momentâneas ou permanentes, a depender do período em que a mudança permanece ativa. Uma variação momentânea é de curta duração e afeta apenas um pequeno número de casos, enquanto uma permanente é persistente e permanece ativa por um longo período (31). Vale ressaltar que uma mudança momentânea pode ser compreendida como um *outlier* (ou anomalia), representando um comportamento atípico.

Geralmente, técnicas de descoberta de modelos de instâncias de processos filtram *outliers*. As abordagens para tratar mudanças em processos, contudo, tendem a se concentrar nas mudanças permanentes. A literatura identifica quatro tipos de mudanças (28) (32): *abruptas, graduais, recorrentes e incrementais* (Figura 18).

Figura 18 – Tipos de mudanças de conceito ou de processo



Fonte: adaptado de (32).

Tipos de Mudanças: na Figura 18, são apresentados os seguintes tipos de mudanças: (a) mudança abrupta, (b) mudança gradual, (c) mudança recorrente e (d) mudança incremental. O eixo *x* representa a dimensão temporal, enquanto o eixo *y* indica diferentes modelos de processos. Cada linha dentro de cada retângulo sombreado representa um caso ou uma instância do processo correspondente.

Mudança Abrupta: uma mudança abrupta ocorre quando uma transição repentina leva a uma nova versão do modelo de processo, que passa a tratar todos os casos em andamento.

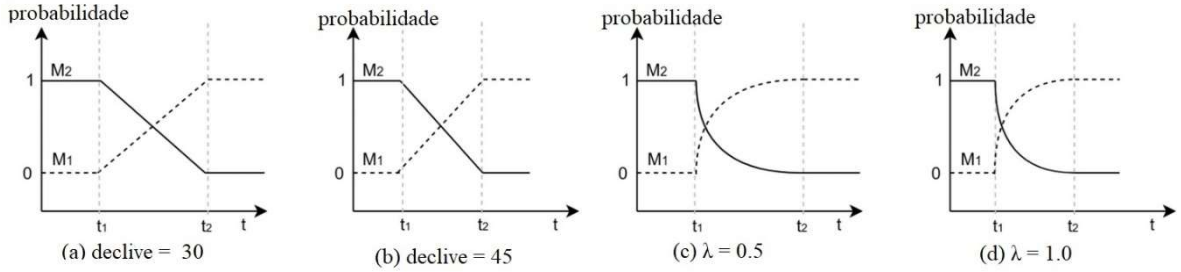
Isso implica uma substituição completa do processo atual (M_1) por um novo (M_2), como ilustrado na Figura 18 (a).

Esse tipo de variação pode ocorrer em situações de emergência ou quando novas diretrizes precisam ser seguidas. Ademais, durante uma alteração abrupta, é necessário redirecionar os casos em andamento para o novo processo, especialmente em cenários do mundo real; este também é referido como "mudança intra-traço", conforme mencionado por (33). Nesse contexto, um caso pode conter eventos relacionados a diferentes versões do processo. Diversas abordagens lidam com mudanças abruptas, mas é importante observar que os registros de eventos sintéticos usados para validação nem sempre incluem mudanças intratraço.

Uma revisão detalhada dos tipos de mudanças encontra-se em (5). Utilizaremos a mesma revisão para um breve resumo dos tipos de modificações.

Mudança gradual: ocorre quando o processo atual (M_1) é gradualmente substituído por um novo processo, representado por M_2. Durante esse período de transição, ambos os processos coexistem por tempo determinado. Após esse intervalo de tempo, apenas instâncias do processo M_2 permanecerão. É viável modelar mudanças graduais de diversas formas, empregando funções distintas para descrever o crescimento ou declínio ao longo do tempo, como ilustrado na Figura 19.

Figura 19 – Variantes de mudança



Fonte: adaptado de (5).

Em que (a) e (b) são variantes de mudança linear gradual – com uma inclinação que define a taxa de mudança – e (c) e (d) mostram variantes de mudança gradual exponencial, caracterizadas pela função $e^{-\lambda t}$ para M_1.

As Figura 19 (a) e (b) demonstram mudanças graduais, caracterizadas por uma variação linear entre M_1 e M_2; ou seja, os casos de ambos os processos decrescem e aumentam continuamente. Já a inclinação define o grau de diminuição/aumento, e após um tempo, todos os casos terão origem em M_1.

Mudança recorrente: ocorre quando um processo M_1 é substituído pelo processo M_2, porém, posteriormente, M_1 ressurge. Essas transições de modelos em mudanças recorrentes podem ocorrer de forma abrupta ou gradual. É relevante observar que, em situações em que um processo modifica sua dinâmica de mudança com base em influências sazonais, esta é classificada como recorrente.

Mudança incremental: refere-se a uma situação em que um processo M_1 é substituído por um processo M_{1n} por meio de pequenas mudanças sucessivas e incrementais. É importante notar que as mudanças abruptas e graduais podem ser consideradas como padrões fundamentais de mudança, enquanto as recorrentes e incrementais podem incorporar características de mudanças graduais, abruptas ou de ambas.

2.7 MUDANÇA DE CONCEITO EM MINERAÇÃO DE PROCESSOS

Quando um modelo de processo é descoberto em determinado período de tempo, assume-se que ele permanece o mesmo desde a data de início até a data final. Entretanto, essa premissa pode não ser inteiramente verdadeira ao longo do tempo, pois o modelo pode estar sujeito a mudanças. Portanto, a abordagem de mudança de processo desempenha papel fundamental na compreensão das suas possíveis causas e consequências.

Além disso, a mudança de processo representa um componente crucial em ambientes que necessitam detectar e avaliar mudanças para orientar eficazmente o objeto conduzido pelo processo de negócio em questão. Para sua análise, é essencial dispor de ferramentas robustas que ajudem a manter a vantagem competitiva da organização em um cenário dinâmico, visando a otimização dos processos para redução de custos, melhoria de desempenho e compreensão mais profunda.

Em relação a (31), cada perspectiva de mudança também deve ser avaliada por meio de ferramentas robustas. Nesse contexto, encontram-se três desafios na abordagem de mudança de processo, a saber: detecção de mudanças no processo, localização e caracterização dos pontos de mudança no processo, e descoberta da evolução do processo.

Com base no estudo (5), que fundamentou a revisão da literatura, foram identificados obstáculos e constatações, em que:

O primeiro desafio, que é de suma importância, envolve a *detecção do ponto de mudança no processo,* ou seja, a capacidade de identificar quando ocorre uma alteração no processo. O ponto de mudança detectado pode ser identificado por meio de um código de

identificação ou pelo tempo decorrido desde a ocorrência, utilizando a data e a hora do evento como referência. No caso de ocorrência de uma ou mais mudanças, é necessário identificar os períodos de tempo entre elas. Nesse contexto, o desafio engloba não apenas a detecção, mas também a localização da mudança e a caracterização desse ponto. Em outras palavras, é preciso compreender a natureza da mudança e identificar a região onde ela ocorreu.

A caracterização está relacionada à natureza da mudança, que pode ocorrer em várias perspectivas, como no fluxo de controle, dados, recursos, de forma súbita ou gradual, entre outras. É importante observar que há escassez de ferramentas e métodos que incorporem a caracterização em suas estratégias, principalmente devido à dificuldade em detectar diferentes tipos de mudança.

Outro desafio relaciona-se ao respeito à descoberta do processo e à compreensão de sua evolução, que é realizada após a localização da mudança e a caracterização de sua natureza. Nesse contexto, pressupõe-se que são necessárias técnicas que explorem e relacionem as descobertas identificadas e compreendam as mudanças ao longo do tempo.

Para realizar a tarefa de detecção de pontos de mudança, existem diversas estratégias que auxiliam o analista de processo a superar esse desafio; essas são conhecidas como “detectores de mudança”.

2.8 DETECTORES DE MUDANÇA

A detecção de mudança de processo desempenha um papel crucial na análise de processos, uma vez que sua identificação permite gerar um alerta para adaptar imediatamente o modelo de processo à nova situação. Essa capacidade de detecção é altamente valiosa, visto que ela proporciona a oportunidade para uma análise adaptativa da situação presente. Por exemplo, em cenários que envolvem modelos preditivos monitorados, o detector emite um alarme de mudança e, em resposta, avalia-se se o modelo de processo deve ser atualizado ou substituído por um novo.

Esse mecanismo de percepção e reação, iniciado pelo detector de mudança, é frequentemente utilizado em conjunto com diversas abordagens para o tratamento de mudanças de conceito – usualmente associadas a métodos de classificação (e.g., árvores de decisão) –, e a avaliação de desempenho é conduzida com base na precisão da classificação (3). Cada detector desempenha uma função fundamental e desafiadora, visando encontrar mudanças reais e verdadeiras ao mesmo tempo que procura evitar alarmes falsos (34).

Diante do presente contexto, a seleção de um algoritmo detector e a meticulosa definição de seus parâmetros são determinadas pelo escopo da análise desejada. Tal escolha não se revela uma tarefa trivial, especialmente no que tange à seleção dos parâmetros apropriados – mesmo para aqueles que possuem conhecimento prévio ou experiência na área. Proporcionar apoio para essa escolha ou tomada de decisão é um dos objetivos desta pesquisa.

Assim, existem dois tipos de análises possíveis: as técnicas de mineração de processos podem ser executadas nos modos *on-line* ou *off-line*. Em uma análise *off-line*, os dados são coletados na forma de registros de eventos, que representam dados históricos de um determinado período de tempo a serem posteriormente analisados. Segundo (1), a maioria das técnicas de mineração de processos é projetada para essa análise*off-line*, e o problema de mudança de processo também é explorado nesse contexto. As possíveis aplicações da análise de mudança de processo *off-line* incluem:

- dividir o *log* de eventos em partes menores com o objetivo de facilitar a compreensão do processo, evitando a criação de um modelo de processo excessivamente complexo e impreciso;
- indicar uma mudança desconhecida e provavelmente inesperada no processo, melhorando a sua análise; e
- utilizar o resultado da análise para redesenhar/melhorar o processo analisado.

Em uma configuração *off-line*, o tempo para a detecção de mudança não é uma consideração crítica. Nesse contexto, a abordagem empregada pode analisar os dados após a sua ocorrência, e o *log* de eventos pode ser filtrado para reter apenas os traços completos.

Por outro lado, uma análise *on-line* – também conhecida como "suporte operacional" (1) – do processo proporciona uma maneira de influenciar e reagir em cada caso à medida que ele ocorre, acessando os eventos enquanto eles são gerados em tempo real – geralmente na forma de um fluxo contínuo de eventos, que é uma sequência infinita daqueles gerados ao longo do tempo (35).

Uma configuração *on-line* geralmente implica em algumas suposições herdadas do domínio da mineração de dados, conforme apresentado por (3): (a) Os dados devem possuir um número fixo e pequeno de atributos, e os algoritmos de detecção devem ser capazes de processar dados ilimitados sem ultrapassar as restrições de memória; (b) Os algoritmos devem considerar uma quantidade finita de memória disponível, normalmente muito menor do que os dados analisados, em um tempo razoável; (c) Há um limite superior de tempo permitido para processar

um evento – por exemplo, geralmente os algoritmos operam com uma única passagem de dados, e o conceito de fluxo pode ser entendido como estacionário ou em evolução.

É necessário abordar o problema da mudança de processo de forma *on-line* quando é imperativo detectar, em tempo real ou aproximado, a presença de uma alteração ou ocorrência de mudança. A análise *on-line* é apropriada quando a organização busca reagir imediatamente a uma mudança ou logo após sua detecção, utilizando, por exemplo, sistemas de alarme instantâneos. O termo *tempo real* é ocasionalmente utilizado sem uma definição formal, sendo sinônimo de processamento rápido de informações. Em outras palavras, um sistema em tempo real deve ser capaz de reagir dentro de restrições precisas de tempo a cada evento no ambiente, conforme definido por (36). No contexto da detecção de mudanças no processo, *tempo real* significa identificar cada mudança no processo o mais rapidamente possível, assegurando, simultaneamente, a relevância dessa identificação.

Assim, pode-se identificar duas principais limitações no contexto da detecção de mudança *on-line:* corretude das detecções e tempo. Essas limitações são, muitas vezes, conflitantes, uma vez que a busca por maior precisão implica na necessidade de processar uma maior quantidade de dados – o que, por sua vez, acarreta no aumento do tempo de processamento. Consequentemente, métodos de detecção *on-line* precisam enfrentar ambos os desafios a fim de alcançar um equilíbrio satisfatório entre eles. Em geral, toda abordagem para a detecção de mudanças em processos *on-line* estabelece um período durante o qual os eventos são coletados para a construção de um modelo de referência, em que este pode ser derivado a partir dos dados dos eventos, servindo como uma base para a detecção de mudanças ou para a identificação de gargalos (1).

De acordo com a pesquisa mencionada no trabalho (3), o período de tempo pode ser configurado em unidades temporais, em termos do número de eventos ou de casos. Após a definição do modelo de referência, o algoritmo de detecção de mudança passa a processar os novos eventos à luz desse novo modelo. Para evitar a necessidade de armazenar todos os eventos do fluxo, é essencial estabelecer um método de esquecimento. Isso pode ser feito por meio de uma estratégia de janelamento – que considera apenas os eventos mais recentes na análise – ou por meio da aplicação de um método adicional, como a introdução de um fator de envelhecimento.

Ademais, existem abordagens para a detecção de mudança de processo *on-line* que utilizam fluxo de traços, como os trabalhos (30) e (9). Nesse contexto, considera-se estritamente *on-line* apenas aquela que emprega um fluxo de eventos, uma vez que o uso de um fluxo de

traços requer a espera pela conclusão dos traços antes de sua inclusão no fluxo. Em outras palavras, cenários que não se enquadram nesse critério não são adequados para configuração *on-line*. Para obter informações adicionais sobre os diferentes tipos de análise, recomenda-se a consulta de (5).

2.8.1 Métodos de Detecção

A tarefa de detecção de mudança, presente em alguns *frameworks,* envolve a aplicação de um ou mais métodos de detecção de mudança.

Na literatura, encontram-se diversos métodos de detecção, tais como: ADWIN, DDM (*Drift Detection Method*) (37), EDDM (*Early Drift Detection Method*) (38), CUSUM, PAGE-HINKLEY (39), IPDD e HDDM-W.

Nesta seção, são apresentados os dois métodos que foram utilizados neste trabalho, que são ADWIN e HDDM_W.

2.8.2 Método de Detecção de Mudança – ADWIN (*Adaptive Sliding Window*)

O ADWIN é um detector de mudança que mantém uma janela de comprimento variável com os itens mais recentes observados. Essa janela tem a propriedade de ter o comprimento máximo estatisticamente consistente com a hipótese de que "não houve mudança no valor médio dentro da janela". O método de esquecimento é acionado quando o seu valor médio diverge do valor médio do restante dos itens (3).

A confiabilidade do método de detecção de mudança é assumida sempre que a janela diminui. Além disso, a média calculada sobre a janela existente pode ser considerada uma estimativa confiável da média atual do fluxo. Isso resulta em um bom desempenho, com limites bem definidos nas taxas de falsos positivos e falsos negativos (40).

Aém disso, e importante destacar que o método ADWIN requer apenas um parâmetro, que é expresso na forma de um limite de confiança δ (delta), com um valor padrão de 0,002 (ver Tabela 7).

Tabela 7 – Algoritmo ADWIN

Algoritmo: *ADWIN: Adaptive Windowing Algorithm*	
1	*Initialize Window* W
2	*for each* t > 0

3 *do* W ← W ∪ {x_t} (i.e., *add* x_t *to the head of* W)
4 *repeat drop elements from the tail of* W
5 *until* |μ̂ W_0 − μ̂W_1 | ≥ ϵ_{cut} *holds*
6 *for every split of* W *into* W = W_0 · W_1
7 *output* μ̂W

Para cada passo t:

Fonte: elaborada pela autora.

- **Limite de taxa de falso positivo**: se μ_t permanecer constante dentro de W, a probabilidade de que ADWIN reduza a janela nesta etapa é, no máximo, δ (valor de significância);
- **Limite de taxa falso negativo:** suponha que, para alguma partição de W em duas partes W_0W_1 (em que W_1 contém itens mais recentes), tem-se $|\mu W_0 - \mu W_1| > 2\epsilon_{cut}$. Então, com probabilidade 1 – δ ADWIN, reduz-se W para W_1 ou menos. a mudança é identificada na linha 5 do algoritmo.

O ADWIN ainda é capaz de identificar sequências de dados que sofrem variações ao longo do tempo, informando o ponto em que a mudança ocorre. Ele utiliza uma janela deslizante de tamanho variável, que é calculada com base nas taxas de mudança observadas nos dados contidos na janela. Este algoritmo é caracterizado por fornecer uma garantia sólida de desempenho, assegurando limites bem definidos para as taxas de falsos positivos e falsos negativos (40).

2.8.3 Método de Detecção de Mudança – HDDM – HDDM-W (Drift Detection Method based on the Hoeffding's inequality)

O HDDM é um método de detecção de mudança que se destaca pela incorporação da técnica *Hoeffding's Bounds*, cuja origem reside nas desigualdades de *Hoeffding*. Estas são empregadas a fim de estabelecer limites superiores para a probabilidade de erro em estimativas baseadas em um número limitado de amostras (41). As desigualdades de *Hoeffding* possibilitam também a construção de intervalos de confiança para a média de uma variável aleatória a partir de um conjunto reduzido de observações (42).

Os autores em (42) desenvolveram dois algoritmos de detecção de mudança que se distinguem pelos tipos de testes empregados:

HDDM-A (*Hoeffding's Bounds with Moving Average*): Utiliza o chamado *A-Test*, um método de detecção de mudança baseado nos limites de Hoeffding com o teste da média móvel.

O HDDM-A emprega a média como estimador; ele recebe como entrada um fluxo de valores reais e emite um estado estimado para o fluxo.

HDDM-W (*Hoeffding's Bounds with Moving Weighted Average*): Utiliza o chamado *W-Test* e apresenta um teste estatístico mais geral para médias móveis ponderadas. Neste caso, os valores mais recentes são ponderados com mais relevância do que os mais antigos, pressupondo uma maior probabilidade de ocorrência.

Ambos os métodos são capazes de calcular intervalos de confiança e limites em cenários de fluxo de dados, nos quais a distribuição das informações pode variar ao longo do tempo. No entanto, o detector HDDM-W distingue-se pelo uso do cálculo de médias ponderadas, o que resulta em uma atribuição de maior peso aos valores mais recentes, com a suposição de maior probabilidade de ocorrência. Os pesos associados aos dados mais antigos podem diminuir gradualmente à medida que novos dados são incorporados à média (ver Tabela 8). Algoritmo conforme trabalho (11).

Tabela 8 – Algoritmo HDDM

Algoritmo: HDDM: Drift Detection Method based on the Hoeffding's inequality.

Requer: $x_{1,} x_{2,}$ ...: fluxo de valores reais, onde: $\forall_i, x_i \in (a,b)$
Requer: $\alpha_W \in (0,1)$: confiança para o nível de alerta
Requer: $\alpha_D \in (0,1)$: confiança para o nível de mudança
Garante: ESTADO ∈ {ESTÁVEL, ALERTA, MUDANÇA}
  /* Declaração de variáveis, em *n* valores reais recebidos: */
  $\hat{X}_{cut}$: estatística calculada a partir de $x_{1,} x_2,\ldots, x_{cut}$
  $\hat{Y}_{n-cut}$: estatística calculada a partir de $x_{cut+1},\ldots, x_n$
  $\hat{Z}_n$: estatística calculada a partir de $x_{1,} x_2,\ldots, x_n$
  $\varepsilon_{\hat{X}_{cut}}, \varepsilon_{\hat{Y}_{n-cut}}$ e $\varepsilon_{\hat{Z}_n}$: limites de erro alinhados com a estatística empregada
  inicio() /* reinicialização/reset de variáveis */
  para todos os x_i que chegam na sequência $x_{1,} x_2,\ldots, x_i,\ldots$ **faça**
    /* Atualizar estatísticas e intervalos de confiança */
    atualizar $\hat{Y}_{i-cut}$, $\hat{Z}_i$, , $\varepsilon_{\hat{Y}_{i-cut}}$, **e** $\varepsilon_{\hat{Z}_i}$ com um novo valor real x_i
    /*atualizar o ponto de corte*/
    se $\hat{Z}_i + \varepsilon_{\hat{Z}_i} \leq \hat{X}_i + \varepsilon_{\hat{X}_i}$ **então**
      $\hat{X}_{cut} = \hat{Z}_i$ **e** $\varepsilon_{\hat{X}_{cut}} = \varepsilon_{\hat{Z}_i}$
      Reiniciar $\hat{Y}_{i-cut}$ e $\varepsilon_{\hat{Y}_{i-cut}}$
    fim se
    /* Determine o estado atual do fluxo de dados */
    se $H_0 : E(\hat{X}_{cut}) \geq \mathrm{E}(\hat{Y}_{i-cut})$ é rejeitado com tamanho α_D **então**
      ESTADO ← MUDANÇA
      inicio()
    se não
      se $H_0 : E(\hat{X}_{cut}) \geq \mathrm{E}(\hat{Y}_{i-cut})$ é rejeitado com tamanho α_W **então**
        ESTADO ← ALERTA
      se não
        ESTADO ← ESTÁVEL
      fim se
    fim se
  fim para
fim()

Fonte: adaptado de (11).

A etapa inicial do algoritmo consiste na estimativa de um ponto de corte relevante denominado *cut*, seguida pela execução do teste *A-Test* ou do teste *W-Test* nas amostras. No contexto deste estudo, empregou-se o algoritmo que realiza o teste *W-Test*, denominado HDDM-W.

Importa esclarecer que há diversos detectores mencionados na literatura, e que apenas alguns foram citados com o intuito de proporcionar uma compreensão básica de suas operações. Ressalta-se a finalidade deste trabalho, que visa auxiliar o analista de processos, cujo objetivo é desenvolver um método de autoconfiguração de detectores de mudanças no processo. Para realizar o treinamento dessa metodologia de autoconfiguração, ele se baseará nos atributos extraídos de análises já efetuadas, que compõem a fonte de dados e, a partir deles, o sistema aprenderá com o perfil de cada registro de eventos e configurações já implementadas.

2.9 MÉTODO DE DETECÇÃO – IPDD

IPDD (*Interactive Process Drift Detection Framework*) apresenta uma proposta diferenciada no que se refere à detecção de mudanças em modelos de processos. Esta visa minimizar o impacto dos parâmetros na detecção e é acompanhada por uma interface que permite a parametrização de acordo com as perspectivas individuais do usuário e sua visão do negócio, proporcionando uma abordagem mais simplificada. Além disso, a interface oferece um acompanhamento visual das mudanças identificadas (42).

Nesta seção, contudo, apresentaremos uma descrição concisa sobre a abordagem, destacando os aspectos mais relevantes que delineiam a ferramenta e as contribuições que ela traz para compor o método proposto.

O IPDD adota uma abordagem que se baseia no uso de uma linha do tempo para monitorar a evolução do modelo de processo em análise. A premissa é que, enquanto o modelo de processo estiver estável, as métricas de qualidade calculadas a partir dos traços do *log* de eventos também se manterão estáveis. Essas métricas (*precisão* e *fitness*) são informadas a um detector de mudança, e o mecanismo de detecção mudança associado ao IPDD, reporta uma mudança caso alguma delas se apresente uma mudança significativa. Após a detecção de uma mudança, um novo modelo de processo é gerado para monitorar novamente as mudanças a partir das métricas. Já os modelos descobertos são apresentados ao longo do tempo, permitindo, assim, o rastreamento das alterações ocorridas no processo. Cada um é representado por um DFG (*Directly Follows Graph*), e entre os modelos adjacentes, são ainda realizados cálculos de

similaridade entre os grafos (nós e arestas), indicando ao usuário quando ocorreu uma mudança na estrutura do processo.

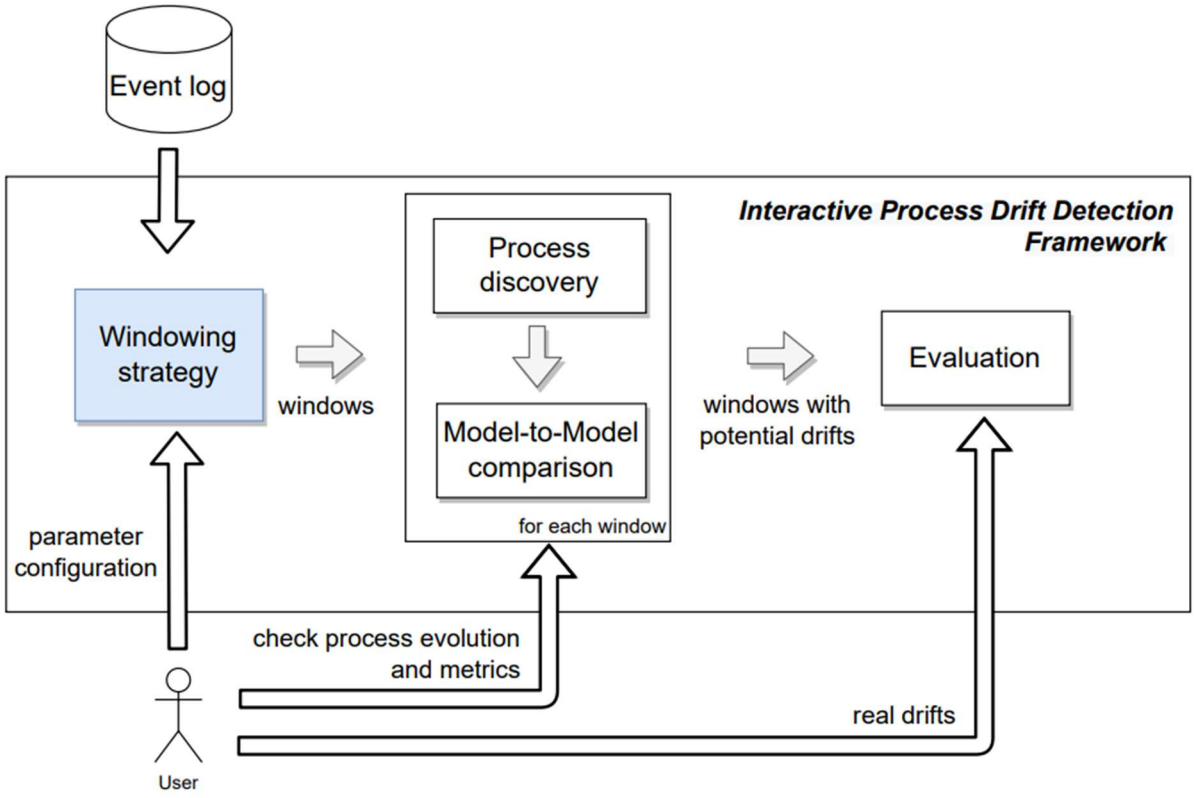
Um aspecto distintivo deste *framework* é sua interface com o usuário, que viabiliza a visualização da evolução dos modelos de processo. Essa funcionalidade permite uma interpretação visual das mudanças no processo, fornecendo informações sobre as discrepâncias em relação ao modelo anterior para cada modelo de processo identificado após a detecção de uma mudança. Tal abordagem contribui significativamente para uma análise mais acessível e compreensível do processo em questão.

Ademais, diferentes implementações estão disponíveis para o IPDD *framework*: Janelamento fixo para *drifts* na perspectiva *control-flow,* janelamento adaptativo para *drifts* na perspectiva de tempo ou dados, e janelamento adaptativo para *drifts* na perspectiva *control-flow*. Neste trabalho, utilizou-se a abordagem de janelamento adaptativo para *drifts* da perspectiva *control-flow*.

O IPDD buscou minimizar a dificuldade de definir os parâmetros associados à configuração do método de detecção de mudanças de processo ao utilizar poucos parâmetros e sua interface simplificada. É importante ressaltar que as ferramentas de detecção propostas e disponíveis frequentemente requerem uma cuidadosa sintonia desses parâmetros para uma configuração eficaz do processo de detecção de mudanças.

A Figura 20 apresenta a estrutura utilizada para a detecção de mudanças no IPDD. No caso da abordagem adaptativa para detenção de mudança na perspectiva *control-flow*, foram implementadas estratégias para o janelamento do *log* de eventos *(Windowing strategy)*. Durante esse processo de janelamento adaptativo, as métricas de qualidade – incluindo *fitness* e *precisão* – são calculadas e comunicadas ao detector de mudanças ADWIN (42). Essas são relatadas pelo detector e utilizadas para determinar as janelas (*sublogs*) que serão repassadas às etapas subsequentes do IPDD.

Figura 20 – Interativa de detecção de mudança da ferramenta IPDD



Fonte: publicado em (41).

O método IPDD é empregado durante a fase de identificação de mudanças de processo e na subsequente geração de um novo modelo de processo após a detecção de cada ponto de mudança. Como evidenciado nesta seção, a versão original do IPDD utiliza exclusivamente o detector ADWIN, e ele opera com duas abordagens distintas: a *Trace-by-Trace* e a *Windowing*, as quais são descritas a seguir:

- *Trace-by-Trace:* nesta abordagem, a detecção da estratégia de janelamento envolve a análise das métricas *fitness* e *precisão* à medida que cada traço do *log* de eventos é processado. Na fase de configuração, esta deriva do primeiro modelo de processo e incorpora os valores iniciais de *fitness* e *precisão* no detector. Subsequentemente, durante a fase de detecção, a abordagem analisa o traço mais recente lido pelo *log* de eventos e calcula as métricas de *fitness* e *precisão* utilizando tanto o novo traço quanto o modelo previamente descoberto na fase de configuração.

Essa abordagem requer duas entradas essenciais: o *log* de eventos e o número de traços iniciais, frequentemente referido como o tamanho da janela (w). Após a detecção de uma mudança, o IPDD utiliza os próximos traços (w) para derivar o modelo do processo; além disso,

o modelo anterior e o ponto de mudança são registrados e salvos. Após a análise completa do *log* de eventos, a abordagem de janelas adaptativas relata todas as janelas salvas como entrada para a etapa subsequente do IPDD. Aqui, a descoberta do processo e a comparação entre os modelos são realizadas. O processo de janelamento aplicado à abordagem *Trace-by-Trace* é delineado na Figura 21 e apresentado detalhadamente em (43).

- *Windowing*: essa abordagem se difere da abordagem *Trace-by-Trace* no que diz respeito ao cálculo da métrica de *precisão*. A avaliação dessa *precisão* tem como objetivo estimar quanta variação de comportamento é permitida pelo modelo mas não fica registrada no *log* de eventos. Entretanto, calcular a precisão com base apenas em um traço (o mais recente) não fornece informações precisas, uma vez que um único traço não representa adequadamente a complexidade do *log* de eventos. Essa limitação motiva a adoção de uma abordagem de *janela deslizante* para o cálculo das métricas de *precisão*.

Nessa abordagem, a fase de configuração preserva a janela inicial. Após a leitura de cada traço durante a fase de detecção, ela é atualizada ao incorporar o novo traço e eliminar o mais antigo, mantendo, assim, uma janela de tamanho constante.

Calcula-se a métrica de *precisão* utilizando a janela e o modelo de processo vigente – dado que a *precisão* é sensível à detecção de comportamento *esquecido*, ou seja, a elementos do modelo que não são mais observados nos traços, o ponto de mudança relatado corresponde ao início da janela atual. Ademais, a diminuição na métrica de *precisão* somente se inicia após a janela atual conter, exclusivamente, traços subsequentes ao ponto de variação. Uma visão abrangente do processo adotado na abordagem *windowing* é ilustrada na Figura 22 e pormenorizada em (43).

2.10 PRÉ-PROCESSAMENTO DOS DADOS E CONTRIBUIÇÕES

Na presente seção, serão detalhadas as etapas necessárias de preparação e implementação de ajustes cruciais para o desenvolvimento e a validação do método.

2.10.1 Contribuições da Ferramenta IPDD para o Método proposto ─ Implementações Específicas

Como mencionado anteriormente neste estudo, o primeiro e mais premente desafio repousa na capacidade de identificar os pontos de mudança em processos, ou seja, na habilidade

de reconhecer a ocorrência de alterações nos processos em questão. Essa dificuldade emerge da necessidade de ajustar os parâmetros de detecção de forma apropriada. Dado que existem numerosos métodos disponíveis para a detecção de mudanças, fica evidente, com base nas conclusões desta investigação, que não é adequado selecionar um método de maneira arbitrária. Em vez disso, é imperativo dispor de uma ferramenta que permita ao especialista escolher o detector e ajustar seus parâmetros e abordagens, levando em consideração a complexidade das variáveis envolvidas para atingir os objetivos almejados.

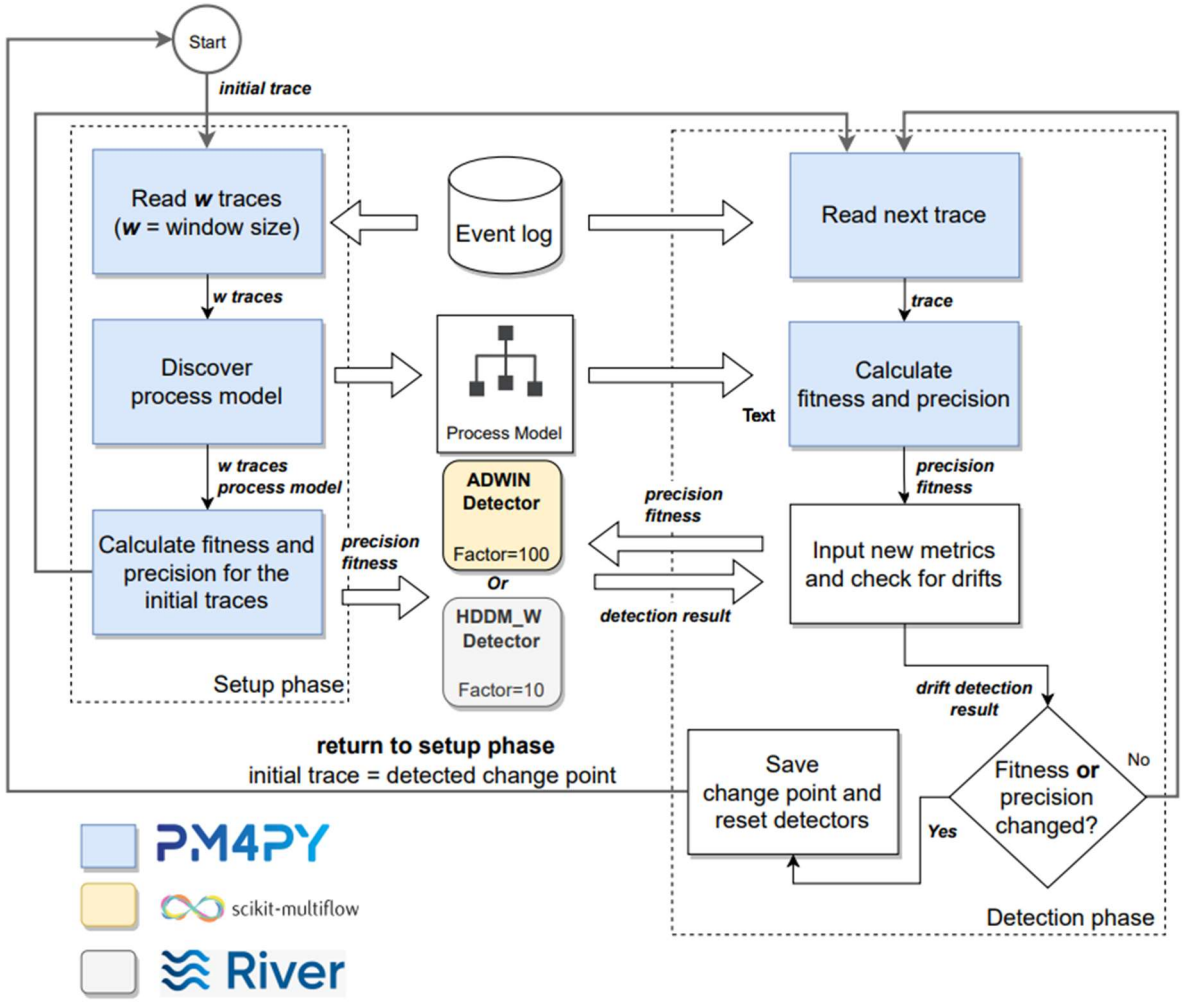
Conforme foram identificadas as premissas essenciais para a implementação do método proposto, várias ajustagens tiveram de ser efetuadas na ferramenta de detecção de mudanças – neste caso denominada IPDD. Com o intuito de alcançar esse objetivo, as necessidades são identificadas, os requisitos são levantados, e ocorre a colaboração da equipe de desenvolvimento da ferramenta para efetuar ajustes nos módulos pertinentes. Foram abordadas as adaptações efetuadas na ferramenta IPDD a fim de possibilitar a geração adequada e confiável dos dados – é relevante destacar que tais adaptações se concentram na aplicação de métodos para a detecção de mudanças dentro de abordagens voltadas para a mineração de processos.

Outrossim, será destacada a notável contribuição dessas modificações tanto para este estudo quanto para investigações subsequentes na área. Para efetuar essas adaptações, o projeto contou com a colaboração da autora da ferramenta IPDD. As adaptações realizadas compreenderam o seguinte conjunto de ações:

1) Incorporação de um novo algoritmo para a detecção de mudanças de conceito: na versão original do IPDD, a implementação do detector ADWIN estava presente. A contribuição nesta fase consistiu na introdução de um novo detector, neste caso, o HDDM_W. Este aprimoramento é ilustrado nas Figura 21 e Figura 22. A inclusão do detector HDDM-W foi realizada em conformidade com as duas abordagens propostas pela ferramenta, nomeadas *Trace-by-Trace* e *Windowing*.

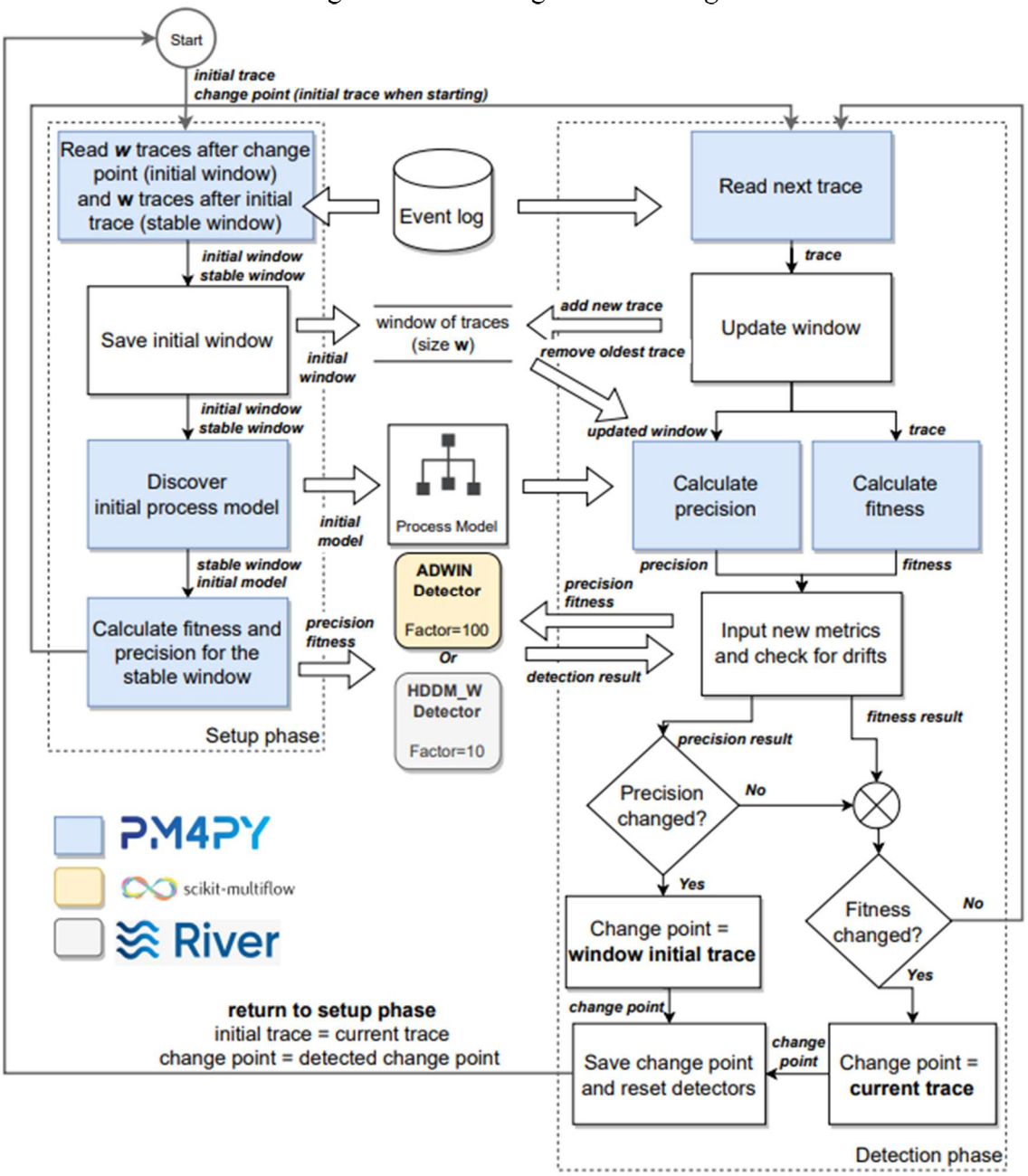
Importa esclarecer que a inclusão do HDDM_W na ferramenta foi uma opção, visto que se trata de um detector mais recente. Além disso, os seus desenvolvedores realizaram uma comparação com outros detectores, e os resultados podem ser encontrados no trabalho de referência (47);

Figura 21 – Abordagem *Trace-by-Trace*



Fonte: adaptado de (43).

Figura 22 – Abordagem *Windowing*



Fonte: adaptado de (43).

2) Fator de Ajuste: esta observação diz respeito à escala de valores, visto que ambos os detectores utilizados são sensíveis à escala de valores do fator de ajuste utilizado. Originalmente, as métricas *fitness* e *precisão* fornecem um valor entre 0 e 1. Porém, no caso do ADWIN, foi verificado que, para as mudanças nas métricas serem adequadamente identificadas, era necessário utilizar valores entre 1 e 100 (fator de ajuste 100). Já nos experimentos com o detector HDDM_W, percebeu-se que era necessário alterar esse fator de ajuste para 10. Esta constatação assume extrema relevância na detecção de mudanças em

mineração de processos, pois a configuração inadequada resultará em resultados insatisfatórios, ou seja, as mudanças não serão identificadas de maneira precisa. Uma contribuição significativa nesta situação é que, caso sejam incorporados novos detectores, o valor do fator de ajuste deve ser rigorosamente testado. Esta informação foi incorporada à fonte de atributos do detector, a qual será utilizada pelo classificador. Dessa forma, a ferramenta passa a considerar essa variável como parametrizável;

3) Geração de *Sublogs* e Submodelos: um dos objetivos fundamentais deste estudo consiste na avaliação da qualidade dos modelos derivados a partir de cada *sublog* identificado entre dois pontos de mudança em um processo. Para alcançar tal propósito, foi implementada a funcionalidade na geração de *sublogs* e submodelos a partir de cada ponto de mudança localizado. A geração de cada submodelo de processo se baseou no número de traços na janela (75 traços) a partir do ponto de transição detectado, e cada *sublog* foi composto por todos os traços existentes até a identificação de um novo ponto de mudança. Esses *sublogs* e submodelos formam a base para a avaliação da qualidade dos novos modelos em relação às mudanças identificadas.

Exemplo: utilizando um *log* de eventos denominado 'cb5k.xes', com as seguintes configurações: Detector de mudança ADWIN, emprega-se a abordagem *Trace-by-Trace*, um valor de delta de 0,002 e uma janela de tamanho de 75 traços. A geração de cada submodelo de processo seguiu o número de traços na janela (75 traços), a partir do ponto de detecção de mudança identificado, e cada *sublog* foi composto por todos os traços a partir desse ponto até a identificação de um novo ponto de mudança. No referido exemplo, constante na Tabela 9, considerando as condições estabelecidas pela parametrização informada, foi identificado um ponto de mudança (510).

Tabela 9 - Exemplo de sublogs e submodelos

Pontos Localizados	***Sublogs***	**Submodelos**
Ponto 1	sublog1_0_74.xes	model1_0-74.pnml
Ponto 2	sublog2_511_585.xes	model2_511-585.pnml

Fonte: elaborada pela autora.

Os critérios para a criação de *sublogs* e submodelos foram definidos da seguinte maneira:

1. *Sublogs*: Correspondem aos traços entre dois pontos de detecção consecutivos; e

2. Submodelos: O primeiro submodelo é descoberto a partir do início do *log* de eventos até a quantidade de traços definida pela janela (75), e os próximos submodelos são descobertos

a partir do ponto de detecção inicial (0-74 traços; 511-585 traços). Adicionalmente, os arquivos com a extensão *.pnml* (*Petri Net Markup Language*) representam o modelo de processo correspondente a cada ponto de mudança identificado de forma apropriada.

Observa-se, neste ponto do trabalho, que as investigações na área de mudança de conceito transcendem a mera utilização de um algoritmo detector. O que se destaca é a necessidade de uma preparação minuciosa, a implementação cuidadosa e testes rigorosos de um ambiente que possa conduzir todas as etapas de maneira detalhada. Isso deve ser feito levando em consideração as particularidades inerentes à área de mineração de processos.

2.10.2 Caracterização e Extração de Elementos para o Processo de Aprendizagem de Máquina

A preparação das informações do arquivo de *log* de eventos e os cálculos das métricas correspondentes são executados com base em configurações específicas do detector de mudanças. Cada configuração aplicada a um *log* de eventos gera um registro na base de atributos. Esta constituirá a matéria-prima para o processo de aprendizagem de máquina, sendo empregada pelo algoritmo classificador no módulo de autoconfiguração. Já o vetor de atributos é composto por elementos extraídos do *log* de eventos e de suas análises. A seguir, eles serão descritos de forma detalhada.

Informação do *log*: nome do arquivo, total de eventos, total de casos, total de atividades, média dos tempos de duração dos casos, mediana dos tempos de duração dos casos, número de variantes do processo, média das frequências das atividades, média das frequências dos recursos.

Informações do processo de detecção de mudança, detectores e seus parâmetros: tipo de abordagem (*Trace-by-Trace*; *windowing*), perspectiva de mudança, detector (ADWIN; HDDM_W), tamanho da janela, valor do fator, valor do parâmetro delta, valor do parâmetro *drift confidence*, valor do parâmetro *warning confidence*, valor do parâmetro lambda, valor do parâmetro *two sided test*.

Informações das métricas de qualidade do *log* e do modelo completo: valor da métrica *fitness* do *log* completo, valor da métrica precisão do *log* completo, valor da métrica generalização do *log* completo, e valor da métrica simplicidade do *log* completo.

Informações das métricas de qualidade dos *sublogs* e submodelos: são gerados os valores a partir dos resultados dos *sublogs* e submodelos, incluindo média, mediana, desvio

padrão, além de valor mínimo e valor máximo para cada uma das métricas, tais como *fitness, precisão, generalização* e *simplicidade.*

Informações de estatísticas de tempo: realizou-se a normalização dos períodos de atividade por traço, como explicado no próximo item da seção, intitulada “Normalização dos Dados de Tempo”. As informações relativas aos períodos de atividade por traço compreendem quantidade de atividades, média de tempo, mediana de tempo, desvio padrão, valor máximo e valor mínimo de tempo.

2.10.3 Normalização dos Dados de Tempo

Na área de aprendizado de máquina e em disciplinas relacionadas, como a mineração de processos e a mineração de dados, são empregadas técnicas de pré-processamento com o objetivo de adequar os dados e mitigar os efeitos das variações de escala. Dentre essas técnicas, destacam-se a normalização e a padronização dos dados. Estas são duas abordagens distintas para o pré-processamento de informações amplamente utilizadas em contextos de aprendizado de máquina e análise de dados, com o propósito de preparar conjuntos de dados antes de aplicar algoritmos.

Quando as informações não são normalizadas e apresentam escalas ou valores substancialmente discrepantes, alguns atributos podem exercer uma influência desproporcional sobre os resultados, capaz de gera um viés em direção a tais atributos.

Um exemplo é o método de normalização mínimo-máximo, amplamente reconhecido e utilizado. Essa técnica, conhecida por sua simplicidade, envolve a reconfiguração dos valores dos atributos para garantir que estejam dentro do intervalo (0, 1) ou (-1, 1) (44). A fórmula geral para a normalização mínimo-máximo no intervalo (0, 1) é representada em F8.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)'} \qquad \text{F8}$$

No contexto de preparação de dados, uma das perspectivas essenciais na análise de processos é a determinação do tempo médio de execução das atividades. Ferramentas de mineração oferecem informações temporais relacionadas às atividades, em que o intervalo entre elas é comumente referido como "tempo de espera" ou "tempo de execução". Tais medidas desempenham papel crucial em várias aplicações e servem como indicadores de desempenho.

Entretanto, o modelo proposto construiu sua base de atributos a partir de diversos logs de eventos, os quais podem conter atividades que variam de durações extremamente curtas, medidas em segundos ou milésimos de segundo, até atividades de longa duração, registradas em horas ou dias. Em alguns casos, dentro de um único registro de eventos, podem coexistir atividades com durações de segundos e outras com durações de dias. Para que o modelo possa armazenar os resultados estatísticos relativos aos tempos das atividades, os quais serão utilizados pelo classificador de aprendizado de máquina, torna-se necessário calcular os valores estatísticos das atividades por traço.

O algoritmo desenvolvido neste contexto realiza a segmentação dos traços de maneira individual, e constrói um modelo para cada traço, calculando métricas estatísticas relacionadas aos tempos das atividades em relação ao tempo total do traço. As métricas calculadas compreendem a média aritmética, a mediana, o valor máximo e o valor mínimo dos tempos das atividades. Posteriormente, são calculadas métricas estatísticas dessas medidas, ou seja, a média das médias, a mediana das medianas, o máximo dos máximos e o mínimo dos mínimos. Esse conjunto resultante, designado por este trabalho como "métrica estatística das métricas estatísticas", constituirá a fonte de atributo dos registros de eventos em relação ao tempo.

A mencionada normalização refere-se ao ajuste dos tempos de cada atividade em relação ao tempo total do traço, assegurando, desse modo, uma representação uniforme e comparável dos dados temporais, conforme demonstrado no algoritmo (ver Tabela 10):

Tabela 10 – Tempo de atividade por traço

Algoritmo: Cálculo das métricas de tempo de atividade por traço

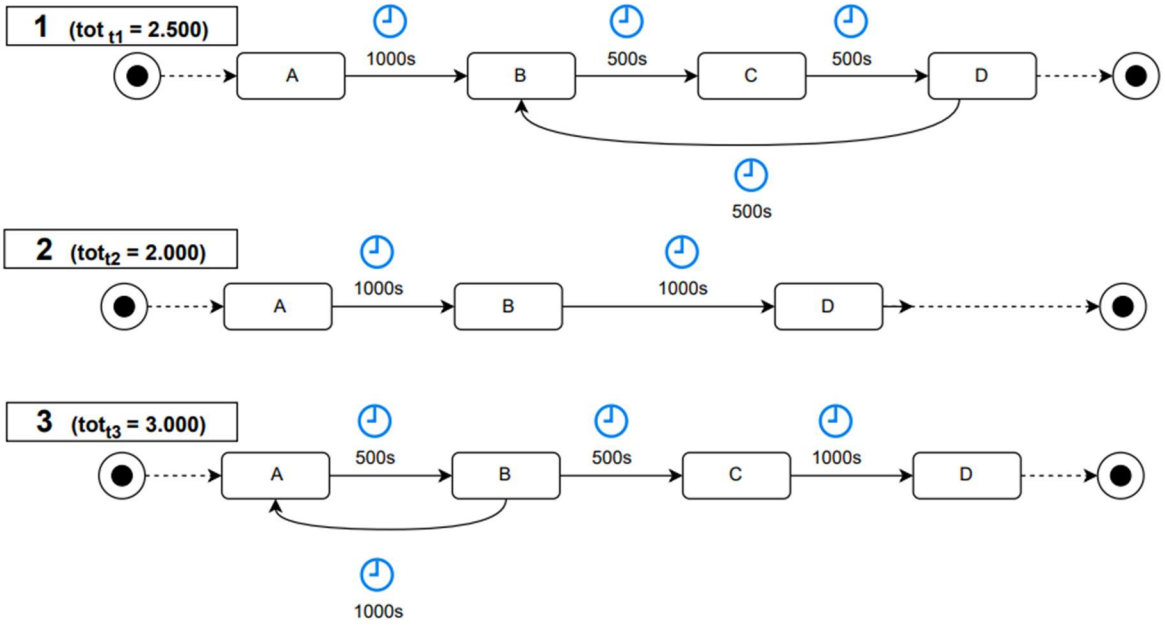
```
1   Requer: xt1, xt2, ...: fluxo de traços do log,  ∀ xti ∈ log
2   /* Declaração de variáveis */
3   tai: tempo da atividade no traço
4   totxti: tempo total do traço
5   totracei : armazena as divisões de tempo de cada atividade pelo tempo total do traço (gera um
6   registro por traço)
7   mti: total das métricas calculadas por tempo de atividade no traço (média, mediana, mínimo e
8   máximo)
9   mtg: total das métricas das métricas de todos dos traços do log em mti (média, mediana, mínimo e
10  máximo)
11  inicio()
12    para todos os xti  do log  xt1, xt2,..., xti,... faça
13      /* divide o total de tempo da atividade pelo total de tempo do traço e armazena */
14      para todas as tai do traço  ta1, tai,..., tai faça
15        totracei ← tai dividir por totxti
16      fim para
17      /* executa função de cálculo das métricas – para o total todas as atividades do traço */
18      mti  (totracei)
19    fim para
20    /* executa função de cálculo das métricas – para o total de todos do os traços do log */
21    mtg  (mti)
22  fim()
```

Fonte: elaborada pela autora.

A seguir, será apresentado um exemplo simples referente ao tema abordado. Tomou-se como base um arquivo de registro de eventos denominado “log1”, o qual é composto por três traços distintos (Traço 1, Traço 2 e Traço 3), cada um englobando quatro categorias distintas de atividades, identificadas como A, B, C e D.

O Traço 1 abrangeu um intervalo temporal de 2.500 segundos, desde o início até o fim do registro. O Traço 2, por sua vez, demandou um total de 2.000 segundos, enquanto o Traço 3 consumiu 3.000 segundos. É relevante observar que cada traço foi segmentado de maneira individual, sendo registrado não apenas o tempo total, mas também a duração entre cada atividade, como ilustrado no exemplo a seguir, na Figura 23.

Figura 23 – Exemplo de tempo de atividade por traço



Fonte: elaborada pela autora.

Exemplo de cálculo, tomando como referência o *log1*, conforme ilustrado na Figura 23.

Passo 1) Cálculo do tempo total e do tempo dedicado a cada atividade em cada traço (ver Tabela 11).

Tabela 11 – Exemplo de tempo de atividade

Caso/traço	tempo Total	'A', 'B'	'B', 'C'	'C', 'D'	'D', 'B'	'B', 'D'	'B', 'A'
1	2.500	1.000	500	500	500		
2	2.000	1.000				1.000	
3	3.000	500	500	1.000			1.000

Fonte: elaborada pela autora.

Passo 2) Normalização do tempo de cada atividade pelo tempo total do respectivo caso (ver Tabela 12).

Tabela 12 – Tempo de atividade pelo tempo total

Caso/traço	tempo Total	'A', 'B'	'B', 'C'	'C', 'D'	'D', 'B'	'B', 'D'	'B', 'A'
1	2.500	0,40	0,20	0,20	0,20		
2	2.000	0,50				0,50	
3	3.000	0,17	0,17	0,33			0,33

Fonte: elaborada pela autora.

Passo 3) Cálculo das métricas estatísticas para cada caso e posterior obtenção de uma métrica estatística global das métricas estatísticas (ver Tabela 13).

Tabela 13 – Estatística por caso

Caso/traço	Média	Mediana	Desvio Padrão	Máximo	Mínimo
1	0,25	0,20	0,10	0,40	0,20
2	0,50	0,50	0,00	0,50	0,50
3	0,25	0,25	0,10	0,33	0,17
Métrica estatística das métricas estatísticas.	Média das Médias	Mediana das Medianas	Desvio Padrão dos Desvios Padrões	Máximo dos máximos	Mínimo dos mínimos
Fonte de atributos	0,33	0,25	0,06	0,50	0,17

Fonte: elaborada pela autora.

Os valores métricos, provenientes das métricas estatísticas, compõem uma parcela fundamental da fonte de atributos utilizada pelo classificador. Nesse sentido, os intervalos de tempo foram submetidos a um processo de normalização em relação às atividades associadas a cada traço. Os resultados obtidos foram, então, expressos em termos percentuais, viabilizando, dessa maneira, uma comparação mais significativa e contextualizada entre os diferentes conjuntos de dados.

2.11 APRENDIZAGEM DE MÁQUINA

Este estudo utiliza-se de ferramentas de aprendizado de máquina para sugerir a autoconfiguração na detecção. Em síntese, o aprendizado de máquina é uma disciplina da inteligência artificial que capacita a máquina a resolver problemas com base nos dados fornecidos. Aqui, estes incluem atributos extraídos dos registros de eventos, detalhados na Seção 3 (Passo 2).

De acordo com (45), a aprendizagem de máquina é definida como a “disciplina de estudo que concede aos computadores a capacidade de aprender sem serem explicitamente programados”. É relevante ressaltar que sistemas de aprendizagem confiáveis desempenham papel crucial em diversas áreas de aplicação, uma vez que muitas tarefas não podem ser resolvidas por meio de técnicas de programação convencionais. Nesse contexto, os algoritmos de aprendizagem de máquina desempenham um papel fundamental nesse tipo de resolução de problemas à medida que são capazes de “aprender” padrões das classes envolvidas no problema a partir de exemplos reais obtidos do ambiente (46).

Os tipos de aprendizado de máquina são geralmente categorizados como supervisionado e não supervisionado. No supervisionado, um conjunto de exemplos de treinamento é fornecido ao algoritmo de aprendizado, também chamado de indutor, no qual o rótulo da classe associada

a cada exemplo é conhecido (47). As principais tarefas associadas a esse paradigma incluem classificação e regressão, em que a classificação lida com os resultados categóricos.

Por outro lado, no aprendizado não supervisionado (48), o algoritmo analisa os exemplos fornecidos e tenta determinar se eles podem ser agrupados de alguma forma, formando agrupamentos. Nesse contexto, os dados não possuem uma saída ou resposta previamente definida. Adicionalmente, vale mencionar que existem outros tipos de aprendizado de máquina, os quais o leitor pode explorar em (49, 47, 50).

Nesta seção, são apresentados alguns conceitos gerais, comumente empregados na área de aprendizado de máquina, bem como as escolhas metodológicas realizadas neste trabalho.

Classificadores: na etapa de autoconfiguração, descrita no Capítulo 4, busca-se construir um método para extrair/selecionar um conjunto de atributos de logs de eventos diversificados e apropriar-se de ferramentas de aprendizagem de máquina para treinar um modelo de previsão robusto. Este último deve ser capaz de predizer a configuração mais relevante a ser utilizada face à situação presente de detecção de mudança de processo.

A classificação, em aprendizagem de máquina, é o processo que intenta identificar características e rotular as saídas destas dentro de um contexto. Em um nível mais detalhado, um classificador é uma função que opera – sobre valores de diversas características (variáveis independentes, ou de previsão, em regressão) – em um exemplo dado (o conjunto de valores de variáveis independentes) para predizer a classe à qual ele pertence (a variável dependente) (50). Há diversos algoritmos de classificação, entre os quais se destacam *Decision Trees, K-Nearest Neighbor, Random Forest, Multi Layer Perceptron, Support Vector Machines, Gradient Boosting,* entre outros (alguns dos quais podem ser baseados em regras). Para uma explanação mais abrangente sobre eles, remetemos o leitor a (51). A título ilustrativo, apresentaremos, de forma sucinta, o funcionamento do KNN (*K-Nearest Neighbor*).

O KNN implementa uma estratégia simples. A ideia é buscar um novo valor baseado na semelhança dos pontos do conjunto de treinamento. Em outras palavras, este é baseado na similaridade entre instâncias. O algoritmo original utiliza a distância euclidiana para calcular a distância entre o novo ponto e cada ponto do conjunto de treinamento (52). De forma sucinta, o seu funcionamento pode ser ilustrado como segue: dado um vetor de consulta x_0 e um conjunto de N instâncias rotuladas $\{x_i, y_i\}_1^N$, a tarefa do classificador é predizer o rótulo de x_0 nas classes P predefinidas. O algoritmo de classificação tenta encontrar os K vizinhos, mais próximos de x_0, e usa um voto majoritário para determinar o rótulo da classe de x_0, sem

conhecimento prévio. Um ponto importante desse algoritmo é a escolha do valor de K e da métrica de distância/similaridade.

Cada algoritmo classificador apresenta um método distinto para calcular e atribuir rótulos às suas respectivas classes-alvo. A seleção do classificador apropriado constitui uma tarefa que demanda rigorosa experimentação. Nesse contexto, foram conduzidos testes com diversos algoritmos classificadores, incluindo Regressão Logística, Árvore de Decisão, K Vizinhos Mais Próximos (*K-Nearest Neighbor*), Perceptron de Múltiplas Camadas (*Multi Layer Perceptron*), Floresta Aleatória (*Random Forest*), *Gradient Boosting* e *Light Gradient Boosting Machine* (LGBM) (53) .

É relevante ressaltar que a magnitude da base de dados utilizada para o treinamento é relativamente limitada, conferindo, assim, uma série de alternativas na seleção do algoritmo classificador. Assim essa etapa do trabalho é, por natureza, adaptável, permitindo certa flexibilidade na abordagem metodológica. No âmbito deste estudo, optou-se por empregar o algoritmo de XGBoost *(Extreme Gradient Boosting)* como classificador de escolha.

Conforme mencionado anteriormente, foram conduzidos testes utilizando diversos classificadores. Contudo, para alcançar os resultados mais promissores, estes foram refinados empregando os classificadores *Gradient Boosting* e XGBoost. O XGBoost (*Extreme Gradient Boosting*) e o *Gradient Boosting* são duas técnicas de aprendizado de máquina amplamente empregadas em contextos de classificação e regressão. Ambos são métodos de *ensemble* que amalgamam diversos modelos considerados frágeis, e resultam na formação de um mais robusto.

No contexto desta pesquisa, decidiu-se por utilizar o algoritmo XGBoost (*Extreme Gradient Boosting*) como classificador. É relevante destacar que este método demonstra adaptabilidade, resultando na exploração de testes envolvendo múltiplos classificadores e seus respectivos parâmetros. Desta forma, proporciona certa flexibilidade na escolha do classificador a ser empregado.

Avaliação: além de definir o tipo de aprendizado a ser utilizado, é fundamental avaliar o desempenho dos modelos supervisionados. Para tanto, existem diversas abordagens, tais como *Holdout*, Validação Cruzada, Redistribuição, entre outras.

Como exemplo, há a Validação Cruzada, a qual se mostra recomendada tanto para abordagens exploratórias como confirmatórias. No âmbito desta técnica, é essencial que as amostras apresentem tamanhos suficientemente grandes para permitir a alocação aleatória dos participantes em, no mínimo, dois grupos. Um desses é empregado como amostra de

treinamento, enquanto o(s) grupo(s) restante(s) serve(m) como amostra(s) de validação cruzada. A aleatoriedade na atribuição dos participantes é de suma importância para garantir que os grupos não apresentem diferenças significativas em características, que possam afetar as estruturas fatoriais observadas (54).

2.11.1 Classificador Gradient *Boosting*

O *Gradient Boosting* é um algoritmo de aumento de gradiente, cujos pormenores técnicos podem ser encontrados na literatura (55). O algoritmo em questão utiliza árvores de decisão como estimadores base por padrão, embora também ofereça a flexibilidade de empregar outros tipos de estimadores. É pertinente ressaltar que o *boosting* representa uma técnica no campo de aprendizado de máquina que se dedica à combinação de múltiplos modelos de aprendizado, denominados estimadores ou classificadores fracos, com o propósito de construir um modelo robusto e preciso. A finalidade primordial subjacente ao processo de *boosting* é aprimorar o desempenho preditivo em comparação com um único modelo. Menciona-se a biblioteca *scikit-learn*[4] em Python como exemplo.

Ademais, esta técnica é um método de *ensemble* que combina múltiplos modelos de árvore de decisão, sendo um procedimento iterativo que opera da seguinte forma: inicialmente, parte-se de um modelo de árvore de decisão simples; posteriormente, calculam-se os erros residuais, que representam a diferença entre os valores observados e os valores previstos pelo modelo corrente. Em seguida, é ajustado um novo modelo de árvore de decisão com o objetivo de prever os erros residuais. As previsões obtidas são, então, adicionadas ao modelo existente. Este processo é repetido em ciclos até que se atinja um critério de parada predefinido.

2.11.2 Classificador XGBoost – *Extreme Gradient Boosting*

O XGBoost destaca-se por uma série de características, tais como a aplicação de técnicas de regularização com intuito de mitigar o *overfitting*, a habilidade de lidar diretamente com valores ausentes nos dados, além de suportar treinamento paralelo – característica que o torna particularmente eficiente em termos de tempo de execução, em comparação a muitas outras implementações do *Gradient Boosting*. Também incorpora um mecanismo de poda de árvore

[4] Disponível em: https://scikit-learn.org/stable/.

para evitar árvores excessivamente profundas. As informações técnicas apresentadas neste documento foram obtidas a partir da documentação oficial da ferramenta XGBoost[5].

Pode-se afirmar que o XGBoost representa uma evolução do método *Gradient Boosting*, e oferece aprimoramentos significativos em termos de desempenho e funcionalidades. No entanto, como mencionado anteriormente neste estudo, a seleção entre diferentes classificadores depende das necessidades específicas do problema, incluindo considerações de eficiência e regularização do modelo. Para tanto, a decisão final por um classificador só pode ser tomada após a realização de testes rigorosos.

Na pesquisa em questão, para o treinamento dos modelos de aprendizado de máquina, foi utilizada a biblioteca XGBoost (versão 2.0.0). A primeira versão foi originalmente proposta por (56) em 2016, e tem sido amplamente empregada na resolução de aplicações em diversas áreas, como ilustrado no trabalho de referência (57) – que aborda o campo da bioestatística e utiliza dados de pacientes. Além disso, (58) demonstra que pesquisadores estão investindo esforços no desenvolvimento de novas ferramentas com o intuito de aprimorar e expandir as capacidades do XGBoost, tornando essa tecnologia cada vez mais atrativa para a abordagem de questões que envolvem conjuntos de dados do mundo real. Essas questões frequentemente exigem a aplicação de classificadores capazes de abordar desafios como conjuntos de dados desbalanceados.

Vale ressaltar que, além dos artigos acadêmicos, é possível observar a aplicação prática como exemplo concreto na utilização da plataforma *Kaggle*[6] – que é uma instituição reconhecida por sediar competições de ciência de dados, nas quais os participantes se engajam na resolução de problemas complexos por meio do emprego de técnicas de aprendizado de máquina e análise de dados. Muitas das soluções apresentadas nessas disputas fazem uso do classificador XGBoost.

Ademais, seu algoritmo é fundamentado em árvores de decisão, no entanto, como os próprios autores mencionam em sua versão original, ele representa uma evolução em relação às árvores de decisão tradicionais no contexto do *Gradient Boosting*. A função original do algoritmo XGBoost (69) é definida como se segue, conforme descrito no trabalho (72):

Outrossim, seu algoritmo de base produz um modelo robusto por meio da combinação de modelos fracos, enquanto o algoritmo XGBoost visa otimizar a função objetivo de custo,

[5] Disponível em: https://xgboost.readthedocs.io/en/stable/
[6] Disponível em: https://www.kaggle.com/

que é composta por uma função de perda (*d*) e um termo de regularização (*β*), definida pela Fórmula F9.

$$\Omega\,\theta = \underbrace{\sum_{i=1}^{n} d\left(Y_i, \hat{Y}_i\right)}_{\text{Perda}} + \underbrace{\sum_{k=1}^{K} \beta(f_k)}_{\text{Regularização}} \qquad \text{F9}$$

Na função apresentada acima, $\hat{Y}_i$ representa o valor preditivo, *n* corresponde ao número de instâncias no conjunto de treinamento, *K* indica o número de árvores a serem geradas e f_k denota uma árvore pertencente ao conjunto (*ensemble*). O Processo de Otimização XGBoost é caracterizado por duas partes distintas: treinamento e regularização.

Treinamento: a fase de treinamento tem como objetivo identificar os parâmetros ideais do sistema. Dado que é desafiador determinar todos os parâmetros simultaneamente, essa etapa é realizada de forma sequencial; com base no conhecimento adquirido, novas árvores são acrescentadas ao modelo, uma de cada vez (59).

Regularização: a regularização engloba um conjunto de técnicas empregadas para evitar o *overfitting* durante o treinamento do modelo. Este ocorre quando o modelo se ajusta muito bem aos dados de treinamento, incluindo o ruído nas informações, mas falha em generalizar bem para novos dados não observados. A regularização é crucial para garantir que o modelo seja capaz de fazer previsões precisas em dados que não foram vistos durante o treinamento. Salienta-se que algumas das técnicas comuns de regularização utilizadas no *XGBoost* estão detalhadas no trabalho (59).

2.11.3 Identificação de hiperparâmetros ideais – *Grid Search*

A fim de garantir a adequação dos conjuntos de hiperparâmetros utilizados no classificador XGBoost, pode-se empregar uma técnica denominada *Grid Search*. Essa abordagem viabiliza a identificação dos melhores conjuntos de hiperparâmetros para um algoritmo de aprendizado de máquina, como o XGBoost, que é notório por sua ampla gama de hiperparâmetros influenciando o seu desempenho, incluindo a taxa de aprendizado, o número de árvores e a sua profundidade máxima, entre outros. De maneira concisa, o *Grid Search* opera da seguinte maneira: para cada hiperparâmetro escolhido, é imperativo especificar um conjunto

de valores a serem avaliados. A partir disso, ele procede para a criação de todas as combinações possíveis dos valores dos hiperparâmetros previamente definidos. É crucial notar que, ao empregar a referida técnica em cada combinação de hiperparâmetros, o modelo é treinado com um conjunto de dados de treinamento e avaliado por meio de validação cruzada a partir do uso de métricas de desempenho. Ao término de todas as iterações, identifica-se o conjunto de hiperparâmetros que produziu o desempenho mais otimizado.

Além disso, a ferramenta possui atributos para armazenar os resultados, tais como os melhores parâmetros (best_params_) e a melhor precisão para validação cruzada (best_score_)(60).

Para uma descrição mais detalhada dessa técnica e seus referidos atributos, é possível consultar a documentação correspondente na página da biblioteca *scikit-learn*[7].

2.11.4 Validação Cruzada *Leave-One-Out*

A técnica denominada *Leave-One-Out* (também conhecida como *LeaveOneOut*, LOO ou validação cruzada de uma única observação) representa uma abordagem particular de validação cruzada, na qual o número de dobras é igual ao número de amostras no conjunto de dados. De acordo com (61), o LOO é um estimador quase não viesado do erro, uma vez que a amostra de treinamento abrange quase todo esse grupo. Nesse contexto, essa técnica pode ser indicada para conjuntos de dados pequenos, como é o caso do trabalho em questão, porém, pode se tornar computacionalmente intensiva em conjuntos grandes. A principal vantagem da validação cruzada *Leave-One-Out* reside no fato de que ela utiliza todos os dados disponíveis para o treinamento.

A explanação dessa técnica e seus atributos correlatos estão disponíveis na página da biblioteca *scikit-learn*[8].

2.11.5 Métricas de Avaliação na Análise de Desempenho do Classificador

A análise de desempenho é essencial em qualquer projeto ou sistema, independentemente de sua natureza, seja este um modelo de aprendizado de máquina ou de

---

[7] Disponível em: https://scikit-learn.org/stable/modules/grid_search.html

[8] Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html. Acesso em: 30 de Out de 2023.

qualquer outra índole. Para aferir de maneira precisa o grau de eficácia operacional, faz-se imperativo recorrer a métricas de avaliação sólidas. Neste contexto, quatro métricas de grande relevância se destacam, a saber: *acurácia*, *precisão*, *recall* e *F1-score*. (62).

Ao considerar os valores extraídos de uma matriz de confusão, é possível identificar as seguintes categorias: verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN).

Acurácia: a acurácia configura-se como a métrica de escolha para a avaliação do desempenho de um dado modelo ou sistema. Sua definição repousa na proximidade entre o número de previsões correta e experimentalmente verificadas e o número global de exemplos submetidos à análise. A acurácia assume relevância notável em situações em que todas as classes de resultados detêm a mesma importância, como ocorre em problemas de classificação – definida pela Fórmula F10.

$$\frac{VP + VN}{VP + VN + FP + FN} \qquad \text{F10}$$

Precisão: a precisão concentra-se na proporção de previsões positivas corretas em relação ao seu total. A precisão é um indicador que avalia o quão eficaz o modelo é na minimização de falsos positivos – definida pela Fórmula F11, a seguir.

$$\frac{VP}{VP + FP} \qquad \text{F11}$$

Revocação: a revocação mede a proporção de previsões positivas corretas em relação ao total de casos reais positivos – definida pela Fórmula F12, a seguir.

$$\frac{VP}{VP + FN} \qquad \text{F12}$$

F1-*Score*: o F1-*score* é uma métrica que combina precisão e revocação em uma única pontuação. É uma média harmônica entre elas – definida pela Fórmula F13, a seguir.

$$2 * \frac{Revocação * \text{Precisão}}{Precisão + Revocação} \qquad \text{F13}$$

Média Macro: é a média não ponderada das métricas para cada classe. Ela calcula as métricas e, em seguida, tira a média aritmética – definida pela Fórmula F14, a seguir.

$$Média\ Macro = \frac{Soma\ das\ Precisões\ de\ Todas\ as\ Classes}{Número\ de\ Classes} \quad \text{F14}$$

Média Ponderada: a média ponderada das métricas para cada classe. Ela calcula as métricas e, em seguida, tira a média aritmética – definida pela Fórmula F15, a seguir.

$$Média\ Macro = \frac{Soma\ das\ Precisões\ Ponderadas\ de\ Todas\ as\ Classes}{Número\ Total\ de\ Amostras} \quad \text{F15}$$

A escolha das métricas de avaliação depende do contexto e dos objetivos específicos do projeto.

2.12 TRABALHOS RELACIONADOS

O presente trabalho concentra-se primariamente na evolução do conceito de mudança, e tem como objetivo compreender e extrair os avanços alcançados pelas ferramentas e pelos métodos já existentes nessa área. Portanto, realizou-se uma investigação aprofundada nesta esfera específica. Esta seção foi dividida em duas partes, sendo a primeira dedicada à revisão da literatura sobre o conceito de mudança na mineração de processos, seguida por uma breve análise das técnicas complementares a este estudo. A busca bibliográfica foi conduzida utilizando os termos de pesquisa "mineração de processos", "mudança de conceito" e "aprendizado de máquina".

2.12.1 Mudança de Conceito em Mineração de Processos

A literatura sobre o tema mudança de conceito se divide em várias abordagens, com trabalhos focados em determinar as perspectivas a serem analisadas, tais como controle de fluxo, tempo e recursos, ou na apresentação de um *framework* mais aderente a diferentes tipos de mudanças, como *abruptas, graduais, recorrentes e incrementais*, atendendo a mudanças *on-line* ou *off-line*. Assim, a revisão da literatura fornece diversas situações e características que nos permitem avançar em direção ao objetivo por este proposto.

O ponto de partida para essa trajetória foi a revisão abrangente sobre *Concept Drift in Process Mining*, conforme discutido em detalhes no artigo (5). Essa análise abordou um total de quarenta e cinco artigos considerados pertinentes. No âmbito deste contexto, destacaremos uma seleção das abordagens relacionadas à mudança de conceito.

Dada a meta de desenvolver um modelo preditivo para facilitar a autoconfiguração de métodos de detecção de mudanças em processos, o primeiro passo envolverá a criação da base de dados de treinamento e teste para o referido modelo. Em outras palavras, a conclusão desta etapa resultará na geração de bases de dados que contenham atributos extraídos de registros de eventos relacionados a processos de negócios realistas.

Ademais, os trabalhos selecionados para análise destacam-se especialmente pela base de dados que se pretende utilizar na validação de nossa proposta. Ao final desta seção, forneceremos um resumo dos dados provenientes da literatura, apresentados de forma tabelar. Esses conjuntos de informações desempenharão um papel crucial na fase de criação da base de teste/avaliação, bem como na análise das características pertinentes ao método de autoconfiguração em desenvolvimento.

Dessa forma, alguns trabalhos desempenharam um papel significativo ao fornecer subsídios relacionados à aplicação e validação das propostas. O estudo completo (5) compila informações acerca das abordagens adotadas pelos autores na detecção de mudança de conceito em processos, as categorias de ferramentas empregadas e os métodos utilizados para validar seus procedimentos. A seguir, apresentaremos, de maneira detalhada, algumas propostas de destaque na análise de mudança de conceito.

No trabalho intitulado *"A multi-level approach for identifying process change in cancer pathways"* (8), há uma abordagem estruturada para analisar a mudança de processos ao longo do tempo no domínio da saúde. Esta concentra-se nas perspectivas relacionadas ao modelo de processo e à detecção de variáveis no nível de traço, com ênfase nas atividades do processo. O objetivo primordial é detectar um ponto de mudança no tempo e, subsequentemente, localizar, caracterizar e explicá-la. O método, implementado, com base no método PM2 (63), é manual e estruturado para a análise de mudança de processos (63).

Outrossim, o método (8) direciona sua análise em três níveis distintos: o modelo de processo, o traço e as atividades. No que tange ao primeiro, é elaborado um gráfico que evidencia métricas de *adequação/fitness, precisão* e *generalização* ao longo do tempo, empregando o método de *replay*. Esse gráfico possibilita identificar possíveis mudanças, embora não forneça a localização precisa delas. As métricas são calculadas utilizando o modelo de processo descoberto pelo método iterativo *iDHM (Data-Aware Heuristics Miner)* (64). Já a base de dados utilizada para validação engloba informações de diagnóstico de pacientes diagnosticados com câncer de endométrio durante o período de 2003 a 2017, provenientes do *Leeds Cancer Center*.

No trabalho (28), destaca-se a importância do papel do especialista no auxílio da validação de cada mudança detectada. Embora a dependência desse profissional seja um aspecto a ser considerado, a discussão apresentou diversos *insights* relacionados ao diagnóstico de câncer endometrial em diferentes níveis. Isso abrange as variantes observadas pela especialidade clínica no nível de traço, bem como as mudanças nas atividades do processo ao longo do tempo. Vale ressaltar ainda que a detecção de mudança de processo foi aplicada em um contexto real.

Em outro estudo na área de saúde, conforme discutido no artigo (65) intitulado *"A Framework for Human-in-the-loop Monitoring of Concept-drift Detection in Event Log Stream"*, destaca-se a importância de compreender o processo com base em um histórico de eventos. Essa abordagem considera o desafio de detectar mudanças de processo – em que a dinâmica de longo e curto prazo devem ser analisadas simultaneamente –, e esta ocorre em um fluxo de eventos, no qual o fluxo de dados está relacionado à execução de um processo de negócios baseado na Web. O registro de eventos passa por um pré-processamento, que envolve técnicas de agrupamento e objetiva avaliar os traços e identificar possíveis irregularidades. As métricas de análise, por sua vez, baseiam-se em histogramas de traços e *timestamps*. O algoritmo de edição de distância (66) é empregado para comparar duas *strings* e quantificar sua dissimilaridade.

Foram definidas três características que representam o traço: edição de distância ponderada, distância ponderada no tempo e tempo (*timestamp* do último evento do traço). Estas são calculadas com uso de histogramas de traço e tempo, e representam o comportamento atual em termos de frequência de atividades (histograma de rastreamento) e diferenças de tempo (histograma de tempo). Ambas as características indicam a diferença entre as informações do novo traço e do histograma em relação à frequência das atividades ou diferença de horário.

Após a identificação de eventos discrepantes entre as duas *séries* (histograma e traço), procede-se o cálculo de um valor ponderado baseado nas ocorrências do histograma. A soma das distâncias ponderadas culmina no valor final da comparação do traço com o histograma, sendo esse processo denominado Distância Ponderada de Edição (EWD).

Quando uma nova amostra se encontra fora do limite de qualquer *cluster* existente, ela é identificada como uma anomalia, e sua densidade é monitorada. Essa abordagem pode ser conceptualizada como um detector de mudanças implícitas, uma vez que se fundamenta nos valores das características dos casos não rotulados.

Em relação à avaliação do método, esta foi conduzida com base em um *log* de eventos relacionados ao faturamento de serviços médicos. Este registro abrange 451.359 eventos, compreendendo mais de 100.000 traços ao longo de um período de três anos. Durante a análise, observaram-se diversas mudanças nos processos ao longo do tempo, e foram identificados vários casos anômalos na forma de *outliers.*

No estudo intitulado (67)*"A Framework for Explainable Concept Drift Detection in Process Mining"* (67), foi desenvolvido um *framework* com ênfase na explicação de cada mudança de processo, bom como na geração de *insights* sobre as relações de causa e efeito subjacentes a variações significativas. A implementação prática deste trabalho envolveu a aplicação de séries temporais e o uso do detector PELT (68). No que concerne à abordagem da causalidade, recorreu-se ao conceito de *Granger-causality*, o qual avalia a probabilidade de correlação entre duas séries temporais em um intervalo de tempo específico, sendo, portanto, interpretado como um tipo de causalidade preditiva.

Já a avaliação da proposta em questão foi conduzida mediante a utilização de bases de dados sintéticas, além de uma base real, proveniente do *Business Process Intelligence Challenge (BPI)*, de 2017.

A base de dados em questão refere-se aos processos de solicitação de empréstimo em um sistema *on-line*. O funcionamento geral deste sistema implica na submissão de um pedido de empréstimo – por parte do cliente – à instituição financeira e receber uma oferta como resposta. A abordagem implementada integra a detecção de mudança de processo e a análise de causa-efeito, visando estabelecer um *framework* para a detecção de mudança de processo explicável. Embora constitua uma iniciativa experimental, detém o potencial de proporcionar explicações para variações específicas nos processos.

No estudo (69) intitulado "*BPIC'2018: Mining Concept Drift in Performance Spectra of Processes*," (69) apresentou-se um desafio de detecção de mudança em um processo vinculado ao espectro de desempenho. A base de dados aborda o processamento de pedidos de pagamento direto provenientes da União Europeia e destinados aos agricultores alemães do Fundo Europeu de Garantia Agrícola. A tarefa proposta visa responder a várias indagações por meio da análise dos dados de eventos, considerando as instâncias do processo de pedido de pagamento entre os anos de 2015 e 2017.

Embora o número de solicitações de pagamento tenha experimentado mudanças mínimas ao longo do tempo, o processo em si mantém uma notável constância. Contudo, é importante salientar que este pode ser suscetível a alterações decorrentes de mudanças

regulatórias ou à implementação de novas técnicas. O cerne da questão abordada reside na caracterização dessas diferenças, delineando-as, de maneira específica, diante de uma instância particular de mudança no processo.

Além disso, a análise da mudança de processo foi conduzida mediante a observação da variação nos padrões comportamentais por segmento, os quais descrevem mudanças no desempenho e na sincronização comportamental de diferentes instâncias. A metodologia empregada para tal análise foi facilitada por uma ferramenta concebida na forma de um *plugin* para o ProM (70).

No (6) intitulado *"LCDD: Detecting business process drifts based on local completeness"* (7), foi proposto um método de detecção de mudanças de processos chamado *LCDD (Local Complete-based Drift Detection).* Essa metodologia, centrada em cenários *off-line,* propõe uma estratégia que se utiliza de uma janela adaptativa para a detecção eficaz de mudanças no comportamento do processo.

A avaliação do método foi conduzida mediante a análise de diversos registros de eventos sintéticos, além da aplicação prática em um conjunto de dados reais provenientes do BPI (BPI2011). Esse último, composto por dados de eventos de um hospital universitário holandês, totalizou 150.291 eventos e 1.143 traços.

A metodologia emprega a propriedade de completude local (LC) de um *log* de eventos, conforme definido em (71), como um mecanismo para a detecção de mudanças no processo. A concepção subjacente a ela repousa na assertiva de que é viável afirmar, com um nível de confiança K predefinido, que um registro de eventos atende à completude local em um intervalo de comprimento limitado.

Com base na premissa supracitada, a detecção de mudança pelo LCDD estabelece um comprimento mínimo *(ML),* ou seja, um número mínimo de traços no *log L,* visando assegurar a completude local LC a um nível de confiança *K*. Dessa forma, a abordagem de detecção inicia-se ao obter as relações de sucessão direta *(DS)* em uma janela completa *(CW),* cujo valor é definido com base no *ML*. Em seguida, lê-se os traços da próxima janela de detecção (DW), obtém-se os *DSs* e identifica-se duas características: *DSs* novas e *DSs* desaparecidas. Uma *DS* nova ou uma *DS* desaparecida, após *CW* na *DW,* indica uma mudança. Tanto a *CW* quanto a *DW* possuem o mesmo efeito que a aplicação de janelas deslizantes. Já o LCDD determina o traço com a nova *DS* como um ponto de mudança.: se uma *DS* preexistente desaparece no *DW*, o LCDD considera o traço inicial do *DW* como um ponto de mudança candidato, assim, o LCDD integra o período estável (*sp)* para definir o ponto de mudança (*CP*) exato.

O valor do período estável (*sp*) indica o número de traços a ser considerado neste momento (configurado por parametrização). Após a leitura dos traços do período estável *sp*, se não houver mais sucessões diretas (*DSs)* desaparecidas, o início do período estável é declarado como um ponto de mudança. O LCDD também implementa uma janela adaptativa baseada no aumento do tamanho da janela completa (*CW*), ajustando o valor do parâmetro *minWin* a cada execução. Além disso, avaliou-se a precisão do LCDD usando a equação de comprimento mínimo (*ML)* para definir o parâmetro de janela completa (*CW*). A precisão das janelas fixas foi obtida usando logs de eventos sintéticos, enquanto a avaliação da abordagem adaptativa mostrou que a precisão do LCDD é sensível ao parâmetro *minWin*. No entanto, o ajuste desse parâmetro pode resultar em uma maior precisão quando se compara aos parâmetros padrões aplicados às abordagens aqui colocadas em prática.

No estudo apresentado por meio da referência (72), *"Looking into the TESSERACT: Time-drifts in event streams using series of evolving rolling averages of completion times"*, aplicou-se um método de detecção de tendência chamado TESSERACT. Este método, adaptado da área de mineração de texto (73), foi direcionado à detecção de mudanças temporais em fluxos de eventos. O seu procedimento compreende três fases distintas: (i) a observação contínua do fluxo de eventos, visando a coleta dos tempos de conclusão de cada atividade; (ii) o cálculo de uma pontuação de significância fundamentada em cada tempo calculado, associado a uma função indicadora, em que esta desempenha a função de filtrar uma proporção de ruídos presentes nos dados; e (iii) a apresentação visual dos resultados obtidos.

Já a pontuação de significância quantifica a distância entre uma nova observação (intervalo entre atividades) e o valor médio. Contudo, devido à natureza não estacionária do ambiente de fluxo, o escore proposto apresenta médias e variâncias que decrescem exponencialmente. Posteriormente, ocorre a suavização adicional (segundo fator de decaimento) com o intuito de obter um indicador estável. Ambos os fatores de decaimento são definidos pelo usuário, exercendo impacto na detecção de mudanças. Referente à implementação do TESSERACT, este ocorre como um *plugin* ProM (74), e sua validação foi conduzida a partir do uso de dados sintéticos e reais. O TESSERACT possibilita detectar mudanças repentinas, incrementais e recorrentes, sendo relevante observar que tais detecções são efetuadas exclusivamente por meio de inspeção visual.

Não obstante, é importante observar que a precisão do método não foi avaliada usando uma métrica específica, como o *F-score*. A avaliação da detecção de mudança foi realizada em um conjunto de dados sintéticos, utilizando o cálculo do atraso de detecção. Este último é

definido pelo número de eventos que transcorre entre o início da mudança e o evento que aciona a detecção.

Ademais, a avaliação foi conduzida em duas bases de dados reais e distintas. A primeira, proveniente do BPI (75), aborda questões no âmbito financeiro e compreende aproximadamente 32.000 pedidos de empréstimo e seus respectivos desdobramentos. A segunda, usada para validação, está vinculada ao processo de gestão de multas de tráfego rodoviário (76).

No âmbito desta pesquisa, foram analisados trinta e seis estudos relacionados com a finalidade de mapear os diversos tipos de bases de dados de eventos empregados nas diferentes pesquisas. Essas informações assumirão relevância crucial durante a fase de validação do método proposto. A Tabela 14 apresenta uma síntese dos estudos analisados até o presente momento, categorizando-os com base na abordagem adotada e no tipo de base de dados utilizado para fins de validação.

Tabela 14 – Classificação por tipo de base utilizada para validação e por abordagem

Seq.	Artigo	Base Sintética	Base Real	Acesso Público Disponível	Abordagem	***Software***
1	(8)		✓		Análise visual	(*) Método manual
2	(65)	✓	✓	✓	Agrupamento de traços	CDESF
3	(67)	✓	✓	✓	Outras abordagens (**)	*Framework* genérico (CD Conceito Explicável)
4	(69)		✓	✓	Outras abordagens (**)	*Plugin* ProM - pacote "*Performance Spectrum*"
5	(6)	✓	✓	✓	Outras abordagens (**)	LCDD
6	(72)	✓	✓	✓	Detecção de tendências	ProM (TESSERACT)
7	(77)	✓			Agrupamento de traços	Não identificado
8	(28)	✓	✓	✓	Teste de hipótese estatística	ProM (*Concept Drift*)
9	(32)	✓	✓	✓	Teste de hipótese estatística	ProM
10	(78)	✓			Análise visual	ProM
11	(79)	✓			Detecção de Mudança	Não identificado
12	(80)	✓	✓	✓	Detecção de Mudança	StrProMCDD
13	(81)	✓	✓	✓	Análise visual	ProM
14	(82)	✓	✓	✓	Detecção de Mudança	Não identificado
15	(83)	✓			Análise visual	(*) Método manual
16	(84)		✓	✓	Outras abordagens (**)	Não identificado
17	(85)	✓			Agrupamento de traços	Não identificado
18	(9)	✓	✓	✓	Teste de hipótese estatística	Apromore
19	(86)	✓	✓	✓	Teste de hipótese estatística	Apromore
20	(87)	✓		✓	Teste de hipótese estatística	Não identificado
21	(88)	✓	✓	✓	Teste de hipótese estatística	ProM
22	(89)	✓	✓	✓	Agrupamento de traços	CDESF
23	(90)	✓	✓	✓	Teste de hipótese estatística	Apromore
24	(91)	✓	✓	✓	Teste de hipótese estatística	Apromore

25	(33)	✓	✓	✓	Teste de hipótese estatística	Apromore
26	(92)		✓	✓	Teste de hipótese estatística	EDBN
27	(93)	✓		✓	Teste de hipótese estatística	Experimental ProM *plugin*
28	(94)	✓			Outras abordagens (**)	Não identificado
29	(95)	✓			Outras abordagens (**)	Não identificado
30	(96)	✓	✓	✓	Agrupamento de traços	CDESF
31	(97)	✓	✓	✓	Detecção de ponto de mudança de processo	VDD
32	(98)	✓	✓	✓	Detecção de ponto de mudança de processo	VDD
33	(99)	✓	✓	✓	Detecção de ponto de mudança de processo	VDD
34	(100)	✓	✓	✓	Detecção de ponto de mudança de processo	VDD
35	(101)	✓	✓		Agrupamento de traços	DOA
36	(102)	✓			Outras abordagens (**)	TPCDD

(*) Método manual: trata-se de uma estrutura de análise que utiliza a comparação entre os modelos de processo com resultados, apresentados em formato de gráfico. A partir da análise do gráfico, pode-se identificar possíveis desvios sem a localização exata.
(**) Na revisão de literatura, os trabalhos foram agrupados nas seguintes categorias de detecção de mudança de processo: Teste de hipótese estatística, agrupamento de traços, detecção de ponto de mudança de processo, análise visual, detecção de mudança e detecção de tendências. Alguns autores propõem diferentes abordagens para lidar com as mudanças de processos que não se enquadram em nenhuma das categorias anteriores. Essas abordagens foram definidas como “outras abordagens” e foram detalhadas em (5).

2.12.2 Aprendizagem de Máquina na Mineração de Processos e Mudança de Conceito

Na mesma seção dedicada aos estudos relacionados, é relevante destacar que, além das pesquisas pertinentes à mudança de conceito, este trabalho também se empenhou na identificação de pesquisas que empregassem técnicas de aprendizado de máquina nas áreas conexas de mudança de conceito e mineração de processos. No entanto, é fundamental salientar

que não foi encontrado na literatura um método análogo. Assim, foram incorporadas referências a algumas pesquisas que desempenharam papel significativo na elaboração da seção relacionada ao processo de autoconfiguração.

Em (103), destaca-se a sinergia entre a mineração de processos, a aprendizagem de máquina e a inteligência artificial a fim de alcançar resultados de forma inteligente e totalmente automatizada. O estudo apresenta uma ferramenta denominada Celonis[9], a qual faz uso de recursos de aprendizado de máquina e inteligência artificial para identificar, automaticamente, pontos fracos nos processos de negócios, auxiliando na tomada de decisões e sugerindo formas de aprimorar as ineficiências dos processos. Tal abordagem é demonstrada por meio de quatro elementos abrangentes: conformidade, aprendizado de máquina, aspectos sociais e consultoria.

No âmbito do trabalho (104), aborda-se a aplicação da aprendizagem de máquina em conjunto com ferramentas de clusterização para a detecção de mudanças de conceito. Esta abordagem incorpora algoritmos detectores previamente mencionados no referido estudo, a saber: ADWIN, DDM e EDDM. Ademais, este estudo constitui uma revisão de literatura cujo propósito primordial é a identificação de tendências de pesquisa e a indicação de direções futuras para estudos relacionados ao tema. Trata-se, portanto, de uma análise do emprego de técnicas de aprendizado de máquina em fluxos de dados, notadamente em cenários que envolvem mudanças de conceito.

A constatação destacada pelos autores na literatura sobre o referido tema reforça a observação de que a maioria das técnicas existentes demonstra ineficácia na adaptação e atualizações de dados. Este cenário realça a imperatividade de uma abordagem cuidadosa e da combinação de tecnologias complementares. Ao longo da investigação, são evidenciados trabalhos de relevância na literatura de aprendizado de máquina e que empregam técnicas de clusterização para representar mudanças de conceito. Ao final, a conclusão do estudo ressalta que tais trabalhos evidenciam a aplicabilidade das técnicas apresentadas na solução de problemas similares, sinalizando a viabilidade de sua implementação em contextos diversos.

No âmbito da pesquisa em questão, destaca-se outro trabalho relevante, o estudo (105), que focaliza a avaliação da qualidade de modelos sob múltiplas abordagens. Para além da utilização de métricas convencionais de qualidade – como *fitness*, *precisão, generalização* e *simplicidade* –, o referido estudo propõe uma segunda abordagem, denominada "*Machine Learning Approach*", traduzida como *Abordagem de Aprendizado de Máquina*. Esta preconiza

[9] Disponível em: https://www.celonis.com/

a aplicação da configuração *k-fold* para a validação cruzada na avaliação de algoritmos de descoberta de processos.

Já o propósito primordial deste estudo é viabilizar a aplicação do método proposto na comparação do desempenho de diferentes algoritmos de aprendizado, e tem a finalidade de investigar a eficácia dessa estrutura de validação na avaliação de algoritmos de descoberta de processos. O conceito subjacente é que o comportamento do processo pode ser interpretado como um elemento passível de aprendizado e captura pelos modelos de descoberta de processos, que, por sua vez, representam a descrição do conhecimento adquirido.

Para tanto, como parte integrante dessa pesquisa, desenvolveu-se um *plugin* denominado "*Control Flow Benchmark"* no *framework ProM*[10]. Este disponibiliza testes fundamentados na abordagem proposta.

Por fim, apesar da existência de pesquisas na área de mudança de conceito em mineração de processos, ainda não se vislumbra uma abordagem única para sua resolução. Ao invés disso, a necessidade de empregar combinações de tecnologias para superar esse desafio torna-se evidente. Nesse contexto, é perceptível que a mudança de conceito, a mineração de processos e a aprendizagem de máquina desempenharão papéis colaborativos na elaboração de soluções em trabalhos futuros. Dentro dessa premissa, o presente estudo apresenta algumas contribuições que indiretamente influenciaram a formulação do método proposto.

[10] Disponível em: https://promtools.org/

3 MÉTODO PROPOSTO

Esta seção tem como objetivo descrever o método, detalhando suas etapas de construção. Conforme observado na literatura ao longo de uma década e meia, o gerenciamento inteligente de processos de negócios tornou-se uma disciplina importante, dotada de princípios, metodologias e ferramentas voltados para a sua melhoria (106). Portanto, o presente trabalho fundamenta-se em métodos de detecção previamente testados e desenvolvidos, visando explorar a análise dos modelos de processos em diferentes momentos de mudança.

Para compreender precisamente cada etapa, é fundamental ressaltar o propósito de estabelecer um método que sustente a análise de detecção de mudança, com foco na autoconfiguração do método de detecção de mudança de processo e na avaliação da qualidade do modelo identificado entre duas mudanças de processo subjacentes. Nesse contexto, foram delineados dois objetivos específicos: o primeiro é *a seleção de atributos relevantes nos logs de eventos*, e o segundo é *avaliar a qualidade do modelo identificado em cada sublog entre dois pontos de mudança em um processo.*

É importante destacar que, caso seja detectado apenas um ponto de mudança ou se a detecção atual corresponder ao primeiro ponto de mudança dentre vários, a análise não ocorrerá entre dois pontos de mudança; mas se concentrará no período que vai do início até o primeiro ponto de mudança do processo. Além disso, se o detector estiver configurado com uma janela deslizante, a investigação pode ser conduzida com base no modelo de processo obtido a partir do *log* de eventos que corresponde aos limites dessa janela. Entretanto, a definição dos limites de início e fim do *sublog* de eventos a ser considerado para a descoberta do modelo de processo para análise é um desafio, uma vez que cada detector adota sua própria estratégia de detecção de mudanças e a localização do ponto de mudança pode variar entre diferentes estratégias ou detectores.

A seguir, será apresentada, de forma detalhada, cada passo seguido na implementação do método proposto juntamente com os testes e os resultados esperados, a fim de atingir os objetivos estabelecidos.

Descrição do projeto: a Figura 24 apresenta uma visão geral do método proposto, dividindo as fases em passos, com o propósito de facilitar a compreensão de cada uma.

O método inicia-se com a obtenção do *log* de eventos. Em paralelo, dois caminhos se desdobram: (a ou b) a configuração de um detector de mudança de processo e (k) a descoberta do modelo de processo a partir do *log* de eventos em questão. O caminho (b) inicia-se com a

configuração manual do detector de mudança, já que não há um modelo treinado para prever a sua autoconfiguração. O fluxo de trabalho segue nessa sequência: (e) detecção de ponto de mudança; (f) descoberta dos modelos de processos a partir dos *sublogs*; (g) extração de valores para as métricas de qualidade, por *sublog* gerado, a partir do detector de mudança; (h) definição das preferências de análise em relação às dimensões de qualidade; e, finalmente, a execução da análise.

No caminho (l), prossegue-se com a extração de valores para as métricas de qualidade sobre o modelo de processo geral. Em seguida, ocorre a (m) definição das preferências de análise em relação às dimensões de qualidade e a realização da (n) análise propriamente dita.

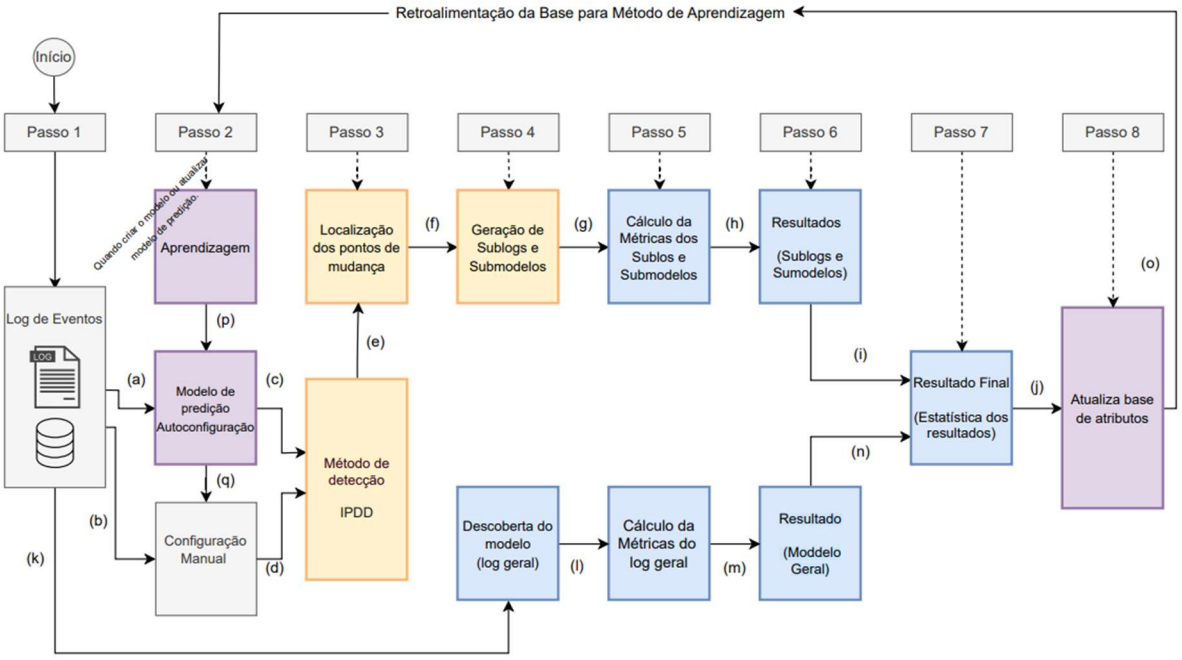
A partir desse ponto, os caminhos (a) e (k) são sincronizados com o objetivo de responder à seguinte pergunta: os modelos de processo descobertos a partir de cada ponto de mudança detectado são objetivamente melhores que o modelo geral?

A resposta a esse questionamento (i e n), bem como os demais valores das métricas extraídas durante o fluxo de trabalho em questão, são registrados na base de dados de atributos (j). Em seguida, continua-se executando as etapas de seleção de atributos e geração do modelo de precisão, que está subjacente à etapa de autoconfiguração do detector de mudança de processo (o).

Neste ponto, é importante ressaltar que foi enfatizado o fluxo de trabalho relacionado à seleção dos atributos. Em outras palavras, a geração do modelo de previsão que apoia o método de autoconfiguração do detector de mudança de processo está integrada à atividade "Seleção de atributos".

A Figura 24 apresenta um ciclo fechado de atividades, que começa com a geração (p), extração e seleção de atributos, segue com o treinamento ou aprimoramento do (q) modelo de precisão e termina com a aplicação do modelo. Essa última atividade tem o propósito de sugerir uma configuração ou realizar a autoconfiguração de cada detector de mudança no ambiente de experimentação (c).

Figura 24 – Visão geral do fluxo de aprendizagem para a construção do método de autoconfiguração de detector de mudança em processo de negócio



Fonte: elaborada pela autora.

3.1 DETALHAMENTO DOS PASSOS DO MÉTODO

Para uma compreensão mais clara, os passos do método serão detalhados a seguir.

Passo 1: *Log* de Eventos.

O *log* de eventos constitui a entrada do fluxo de trabalho e é codificado no formato XES[11] (*eXtensible Event Stream*). O procedimento deste passo segue a seguinte sequência: carregamento do *log* de eventos a partir de uma base de dados e extração de um vetor de atributos. Se um modelo de previsão já estiver estabelecido, o fluxo de trabalho (a) continua com a aplicação deste para compor a autoconfiguração do detector de mudança. No caso de não haver um modelo disponível para a, o fluxo (b) é seguido para a configuração manual.

É importante destacar que o fluxo de trabalho em questão não envolve tarefas específicas de pré-processamento do *log* de eventos de entrada. Pressupõe-se que o *log* é bem-formado e contém as informações necessárias para a análise, conforme os requisitos delineados na Seção 2.2.

Passo 2: autoconfiguração do Detector de Mudança ou configuração manual.

[11] Informações sobre o padrão XES podem ser consultadas em: www.xes-standard.org.

Neste estágio, dois fluxos de trabalho são executados: o fluxo de aprendizagem do modelo de previsão e o fluxo de aplicação desse modelo de previsão. O primeiro fluxo utiliza uma base de atributos do *log* como entrada e realiza, sequencialmente, o treinamento ou aprimoramento do modelo de previsão. O segundo fluxo utiliza o resultado da classificação, que é empregado na autoconfiguração do detector de mudança de processo. O objetivo é que o detector, com a configuração resultante, apresente o melhor desempenho possível em relação ao processo sob análise – representado pelo *log* de eventos fornecido.

A principal fonte de dados para a seleção de atributos é a base de dados, a qual é gerada e atualizada ao término do fluxo de trabalho, representado na Figura 24, retornando ao Passo 2. Cada entrada nessa base primária armazena uma série de medidas extraídas do *log* de eventos, conforme detalhado na Seção 2.10.2 (Caracterização e Extração de Elementos para o Processo de Aprendizagem de Máquina), bem como informações sobre o processo de detecção de mudança, detalhes sobre os detectores e seus parâmetros, informações sobre as métricas de qualidade do *log* completo, além de informações sobre as métricas de qualidade dos *sublogs* e submodelos.

A cada utilização do método, o analista tem a possibilidade de decidir se deseja ou não intervir no processo de configuração do detector de mudança. Essa intervenção é registrada como uma entrada na base de dados.

É relevante observar que o método proposto permite a intervenção do analista na escolha do detector de mudança e em seus parâmetros de configuração, mesmo reconhecendo que um dos objetivos do trabalho em questão é auxiliá-lo na parametrização do detector de mudança de processo, sem a necessidade de conhecimento técnico aprofundado quanto à utilização do método proposto.

Autoconfiguração do Detector de Mudanças: pressupõe-se que a autoconfiguração de um detector de mudança de processo é uma tarefa não trivial, embora seja altamente benéfica. Observa-se que o domínio de conhecimento no qual se opera pode ter uma influência direta, especialmente na configuração do detector de mudança. Além disso, a parametrização da estratégia de detecção em um processo de comercialização de bens e serviços não será a mesma que em um processo de fabricação. Essas diferenças podem ser explicadas considerando os valores das métricas ou dos atributos do processo em questão, como a análise dos tempos das atividades, que podem ser substancialmente distintos.

Na seção específica sobre Aprendizado de Máquina, foram conduzidos testes com alguns classificadores. No entanto, este trabalho optou pelo uso do classificador XGBoost, cujas

configurações, validações e resultados obtidos estão documentados na seção de Resultados, conforme detalhado na Seção 4.4 (Autoconfiguração).

É relevante destacar que esse módulo é flexível e permite a utilização de outros classificadores, caso seja desejado.

Já a Figura 25 apresenta um cenário que mapeia três fontes de dados para a extração de atributos, em que esses incorporam informações provenientes do *log* de eventos, do método de detecção e das métricas das dimensões de qualidade. A extração de dados de cada uma dessas três fontes gera uma linha em uma tabela, representando a integração, sincronização e o armazenamento dos atributos.

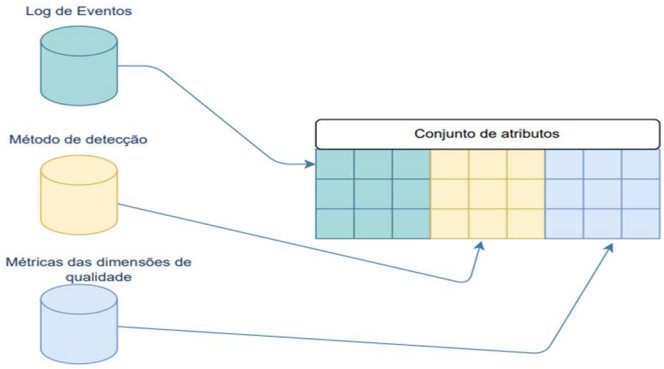
De maneira concisa, a seguir é apresentada a relação dos atributos de acordo com cada fonte de dados:

***Log* de eventos**: nome do arquivo, total de eventos, total de casos, total de atividades, média dos tempos de duração dos casos, mediana dos tempos de duração dos casos, número de variantes do processo, média das frequências das atividades, média das frequências dos recursos.

Método de detecção: tipo de abordagem (*Trace-by-Trace; windowing*), perspectiva de mudança, detector (ADWIN; HDDM_W), tamanho da janela, valor do fator, valor do parâmetro delta, valor do parâmetro *drift confidence*, valor do parâmetro *warning confidence*, valor do parâmetro lambda, valor do parâmetro *two sided test*.

Métrica de qualidade: valor da métrica *fitness* do *log* completo, valor da métrica precisão do *log* completo, valor da métrica generalização do *log* completo, valor da métrica simplicidade do *log* completo e as métricas geradas a partir dos *sublogs* e submodelos, como média, mediana, desvio padrão, valor mínimo e valor máximo para cada uma das métricas, tais como *fitness*, *precisão*, *generalização* e *simplicidade.*

Figura 25 – Fontes de dados para a extração e seleção de atributos



Fonte: elaborada pela autora.

É importante observar que, ao utilizar detectores de mudança, é imperativo compreender os requisitos para sua aplicação. Por exemplo, no caso específico do detector ADWIN, é necessário fornecer o valor do parâmetro delta. Já o desempenho do método, em um determinado contexto, pode ser aprimorado ou prejudicado com base na afinação dos parâmetros. Logo, esse ajuste de parâmetros não é uma tarefa simples e pode demandar considerável tempo e esforço para ser implementado.

Passo 3: seleção de *sublog* a partir do ponto de mudança detectado.

A descoberta do ponto de mudança envolve a utilização da ferramenta IPDD para identificar o ponto de transição em um registro de eventos. É pertinente observar que, embora outras ferramentas semelhantes possam ser empregadas, exclusivamente a IPDD foi utilizada no estágio de geração dos resultados.

A detecção de um ponto de mudança representa um marco significativo, pois, a partir dela, derivam duas operações igualmente relevantes: a localização e a caracterização de cada ponto de mudança. A localização do ponto de mudança não é uma tarefa simples, uma vez que nem todas as abordagens de detecção de mudança disponíveis na literatura oferecem uma estratégia clara para sua determinação. No entanto, a localização é crucial, pois ela permite a caracterização/explicação do ponto de mudança e, com base nisso, a definição dos limites que delimitam o início e o fim do *sublog* de eventos – que serve como base para a descoberta do um novo modelo de processo.

Nesta etapa, o registro geral é subdividido em *sublogs*.

Passo 4: descoberta do novo modelo de processo.

A descoberta do novo modelo de processo representa a sua versão mais atualizada. Esta reflete o momento da análise, pois o *log* de eventos, a partir do qual o novo modelo de processo

é descoberto, é delimitado pelos limites de início e fim do ponto de mudança em questão. A partir da obtenção do novo modelo de processo, é possível a realização de uma avaliação, que pode ser subjetiva – feita por um indivíduo – e/ou objetiva – realizada por meio de máquina. A avaliação subjetiva é essencialmente visual, enquanto a avaliação objetiva baseia-se fortemente na aplicação de métricas para determinar e interpretar a qualidade do modelo de processo descoberto. Vale ressaltar que o foco principal deste projeto, em relação à avaliação da qualidade do modelo de processo, está restrito à perspectiva objetiva e limitado às dimensões canônicas de qualidade de processo.

Outrossim, o fluxo de trabalho relacionado à descoberta e à avaliação objetiva é aplicado tanto a cada *sublog* de eventos entre dois pontos de mudança de processo quanto ao *log* de eventos geral, ou *baseline.* A expectativa é que o modelo de processo descoberto a partir de cada *sublog* apresente uma qualidade superior ao modelo de processo descoberto a partir do *log* de eventos geral.

Passo 5: cálculo das métricas de qualidade por modelo de processo.

A partir dos *sublogs* e submodelos identificados no Passo 4, são calculadas as métricas de qualidade tanto para o modelo geral quanto para cada sublog. Com o intuito de facilitar sua compreensão, apresenta-se um exemplo de cálculo das métricas derivadas de cada *sublog* e seu modelo de processo correspondente. Os valores utilizados para essa demonstração foram extraídos do *log* denominado "Cb5k.xes", cujas configurações incluem o Detector Adwin, uma janela de 75, delta de 0,3 e a abordagem *Trace-by-Trace* (Tabela 17).

A obtenção de um valor para cada métrica de qualidade em relação a cada modelo de processo possibilita sua avaliação e interpretação da qualidade de maneira objetiva. Conforme mencionado anteriormente, quatro métricas de qualidade são consideradas: *aptidão/fitness, precisão, generalização* e *simplicidade.*

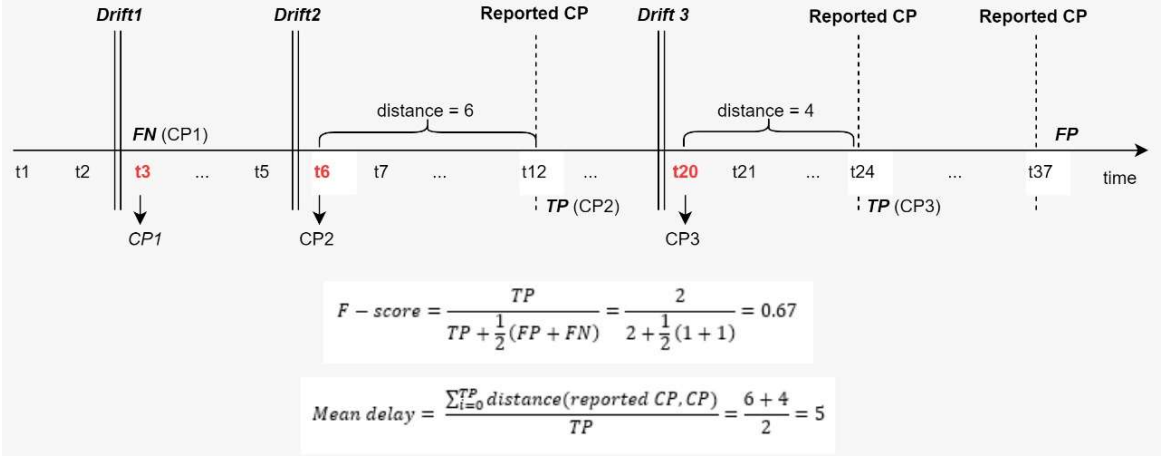
Além da interpretação de cada métrica de qualidade de um modelo de processo, destaca-se que, na validação dos pontos de mudança, o IPDD utiliza o cálculo da métrica do *F-score*.

Conforme (43), o *F-score* é a média harmônica entre *precisão* e revogação, que se baseiam nos valores de verdadeiros positivos (TP), falsos positivos (FP) e falsos negativos (FN). Um TP indica que há um desvio relacionado a um desvio *de fato* (considera-se apenas o primeiro desvio relatado após um desvio real); um FP é um desvio relatado não relacionado a um desvio *de fato*, e um FN é um desvio *de fato* não detectado. Considera-se $(rd, rd + et)$ o intervalo para um TP, em que rd é o traço onde a mudança *de fato* foi informada, e *et* é o número de traços até a próxima mudança conhecida. Se houver apenas uma mudança no *log* de

eventos, um TP será contado se os métodos relatarem um ponto de mudança após o ponto de desvio real, ou *de fato*. Já o *F-score* relata um valor entre (0, 1), medindo a precisão da detecção de desvio. No entanto, ele não mede se o ponto de mudança relatado está "próximo" do ponto de mudança real. O atraso médio complementa a análise do *F-score*, medindo a distância entre os pontos de mudança relatados e os pontos de mudança reais para os desvios relatados, considerando um TP

A Figura 26 mostra um exemplo de aplicação das duas métricas supracitadas. No eixo *x*, estão os traços ordenados por *timestamp (t1, t2, ..., tn)*. O *log* de eventos contém três pontos de mudança: *t3, t6* e *t20*. Enquanto isso, o método de detecção relata três mudanças, a saber: *t12, t24 e t37*. O *F-score* contará um TP somente se uma mudança de processo for relatada após uma mudança real e antes da mudança conhecida a seguir. No exemplo, apenas *t12* e *t24* são assumidos como TP. A distância entre a mudança relatada e a real é o número de traços entre elas.

Figura 26 – Exemplo de aplicação das métricas de avaliação *F-score* e *Mean delay*



Fonte: extraído de (43).

Passo 6: definição de preferências de análise da qualidade.

A definição das preferências de análise da qualidade permite concentrar esforços nas dimensões de qualidade mais relevantes. As sugestões de análise, com base nessas dimensões, incluem as seguintes opções: *FxP* (*fitness* vs. precisão), *FxS* (*fitness* vs. simplicidade) e *FxG* (*fitness* vs. generalização). Inicialmente, o analista faz uma escolha direta entre um dos seguintes planos: *FxP, FxS* ou *FxG*. Em um estágio posterior, a indicação de análise é feita de forma indireta, em que o analista estabelece um conjunto de preferências, e a decisão final sobre o plano de análise é determinada por meio de um método formal, como o AHP (107). De maneira pragmática, parte dos resultados estão estruturados no formato tabelar. A tabela conterá

os valores das quatro métricas de qualidade do modelo de processo para cada um, incluindo o modelo geral, ou *baseline.* Além disso, para cada plano de análise, como *FxP, FxS e FxG*, será gerado um gráfico 2D correspondente.

A validação da qualidade opera sobre dois conjuntos de valores. O primeiro é determinado pelo valor das estatísticas. Neste exemplo, utilizou-se a mediana de cada métrica de qualidade para cada modelo de processo gerado entre dois pontos de mudança. Neste caso, é importante observar que cada detector de mudança pode indicar uma quantidade variável de pontos de mudança de processo para o mesmo *log* geral de eventos.

A partir do exemplo com o *log* Cb5k.xes, serão gerados resultados utilizando a mediana das métricas, conforme detalhado na Tabela 15 e no Gráfico 1.

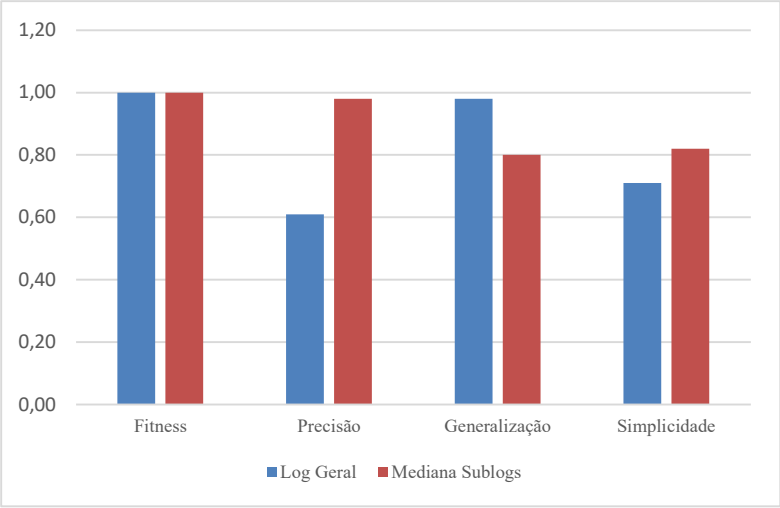
Nota: O cálculo das métricas, a partir de cada *sublog* e seu modelo de processo correspondente, é exemplificado com os valores extraídos do *log* Cb5k.xes, que possui as configurações: Detector Adwin, janela 75, delta 0,3 e abordagem *Trace-by-Trace*.

Tabela 15 – Mediana das Métricas de qualidade calculadas sobre o *Log* Geral *versus* seus *Sublogs* (Cb5k.xes)

***Log* Cb5K.XES**	***Fitness***	**Precisão**	**Generalização**	**Simplicidade**
***Log* Geral**	1,00	0,61	0,98	0,71
Mediana *Sublogs*	1,00	0,98	0,80	0,82

Fonte: elaborada pela autora.

Gráfico 1 – Valores das métricas de qualidade para o *Log* Geral e os seus *Sublogs* (Cb5k.xes)



Fonte: elaborada pela autora.

A validação da qualidade opera sobre dois conjuntos de valores. O primeiro é determinado pelo valor das estatísticas. Neste exemplo, foi utilizada a mediana de cada métrica

de qualidade para cada modelo de processo gerado entre dois pontos de mudança. É importante observar que cada detector de mudança pode indicar uma quantidade variável de pontos de mudança de processo para o mesmo *log* geral de eventos.

Passo 7: apresentação final do resultado da análise.

Neste passo, o resultado final será apresentado de acordo com a escolha do tipo de análise feita pelo especialista, que pode optar por uma ou mais, a saber: *FxP, FxS, FxG*.

A apresentação final de cada uma será limitada à exibição da tabela de resultados estruturada na etapa anterior, que contém os valores das quatro métricas de qualidade do modelo de processo para cada um, incluindo o modelo geral. Além disso, em função das preferências de análise informadas pelo analista, será apresentado o plano de análise correspondente, ou seja, *FxP, FxS ou FxG.*

Nessa fase, o analista dispõe dos resultados de cada métrica de qualidade, calculada a partir do *Log* Geral e de cada sublog, e dos valores das estatísticas, conforme evidenciado nos Passos 5 e 6.

A base de atributos fornece informações abrangentes sobre as métricas, proporcionando ao analista uma ampla fonte de informações a serem utilizadas em suas análises. Ele inclui: **informações das métricas de qualidade do *log* completo e modelo completo** (valor da métrica *fitness* do *log* completo, valor da métrica *precisão* do *log* completo, valor da métrica *generalização* do *log* completo, valor da métrica *simplicidade* do *log* completo) e também **informações das métricas de qualidade dos *sublogs* e submodelos (**média, mediana, desvio padrão, valor mínimo e valor máximo para cada uma das métricas, como *fitness, precisão, generalização e simplicidade*).

Isso permite uma análise detalhada das métricas em diferentes segmentos do *log* de eventos e para submodelos gerados em cada ponto de mudança descoberto.

Passo 8: atualização da base de atributos.

Cada análise poderá atualizar ou enriquecer a base de atributos com o perfil de análise. Em relação ao armazenamento destes, cabe ao especialista decidir quando um resultado é significativo suficientemente para apoiar análises futuras de uma mudança no processo. No entanto, o método foi construído de forma a utilizar apenas os melhores resultados no pré-processamento da base de treinamento do classificador.

Os resultados apresentados neste trabalho, portanto, utilizaram como base o melhor resultado obtido na combinação *FxP,* que estabelece uma relação entre as métricas de qualidade *fitness* e *precisão*. Isso foi feito para assegurar resultas confiáveis ao longo dos testes. No

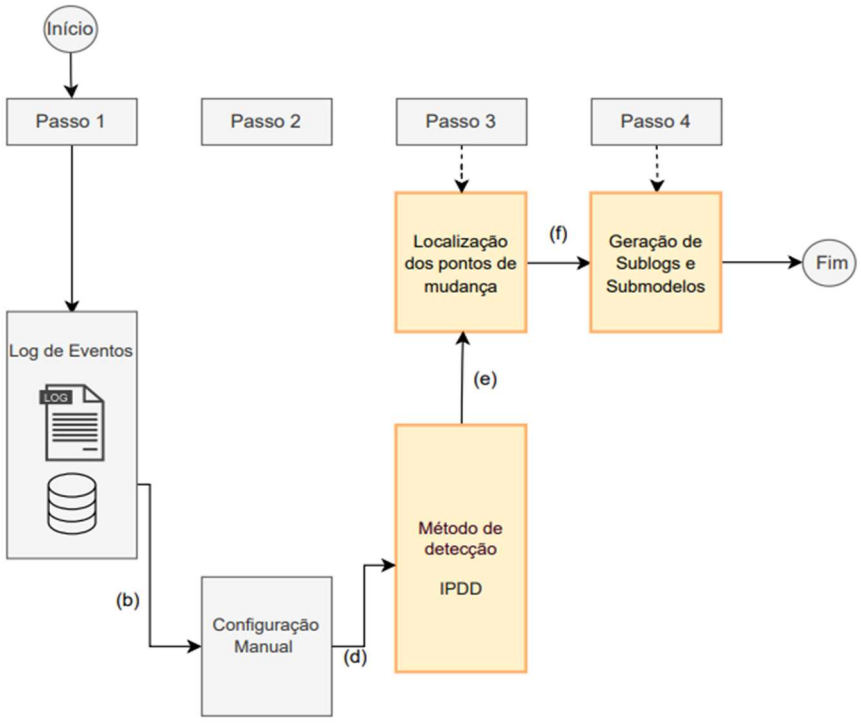
entanto, é importante ressaltar que outra opção de combinação pode ser realizada, como descrito na Seção 2.4 – que mostra os diferentes arranjos dos critérios de análise, envolvendo das quatro dimensões de qualidade.

Os detalhes completos do Passo 8 estão disponíveis nos resultados apresentados na Seção 4, no Item 4.4 – Autoconfiguração.

3.2 ETAPAS DO MÉTODO

O método foi desenvolvido em etapas. A primeira consistiu na montagem de um ambiente de experimentação capaz de gerar logs de eventos, nos quais os pontos de mudança foram devidamente detectados. Essa etapa envolveu o desenvolvimento dos Passos de 1 a 4, conforme detalhado abaixo na Figura 27.

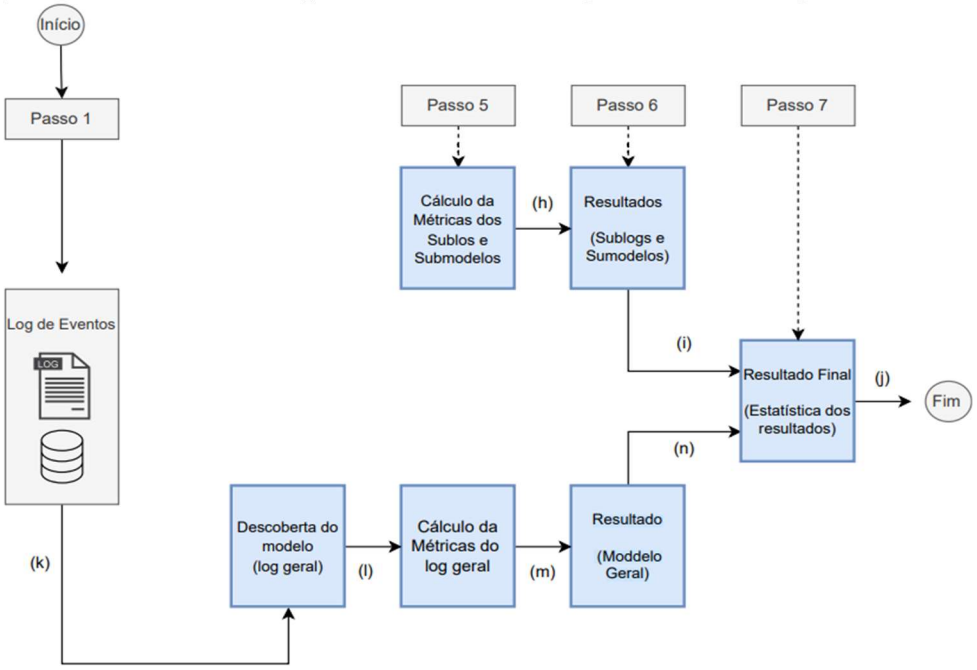
Figura 27 – Ambiente de detecção de pontos de mudança e geração de *sublogs* e submodelos



Fonte: elaborada pela autora.

Posteriormente, os esforços concentraram-se na detecção dos pontos de mudança e na geração dos *sublogs*, abrangendo os Passos 5 a 7, conforme ilustrado na Figura 28.

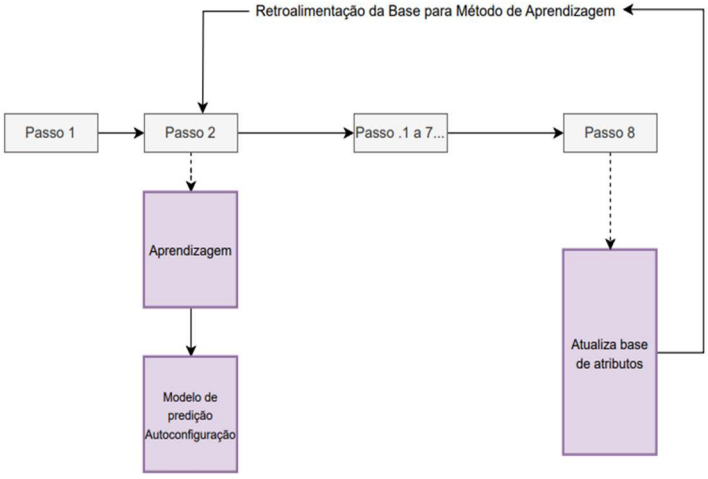
Figura 28 – Detecção dos pontos de mudança e geração de *sublogs* e submodelos



Fonte: elaborada pela autora.

Para alcançar o objetivo de autoconfiguração, os próximos passos incluem a seleção dos atributos relevantes e, em seguida, treinamento, teste e validação de um modelo de predição para configurar os parâmetros, conforme indicado na

Figura 29 – Validação do modelo de predição



Fonte: elaborada pela autora.

No próximo capítulo, serão apresentados os resultados obtidos.

4 RESULTADOS

A motivação desta pesquisa, como já mencionado anteriormente, está relacionada à criação de um método capaz de prever a configuração de um detector de mudança de processo em um contexto específico. Como parte desse esforço, o primeiro passo envolveu a criação de um ambiente de experimentação que pudesse gerar logs de eventos com pontos de mudança devidamente rotulados.

A partir da obtenção dos pontos de variação e seus respectivos *sublogs*, foi implementado um fluxo de trabalho para calcular as métricas de qualidade de cada modelo e submodelo do processo. Esta ação envolveu a extração de dados de cada uma das fontes, representando a integração, sincronização e o armazenamento dos atributos. Adicionalmente, foi realizada a criação do ambiente de treinamento da base de atributos, culminando na execução de treinamento, teste e validação do modelo de previsão da configuração do detector de mudança de processo. Nesta seção, serão apresentados os resultados obtidos a fim de validar o método proposto.

De forma resumida, o fluxo de trabalho para o controle básico do ambiente de experimentação inclui as seguintes etapas:

a) Obtenção de um *log* de eventos L com pontos de mudança rotulados R;

A utilização da ferramenta IPDD foi fundamental para tal. Nessa etapa, foram identificadas as necessidades específicas de implementação que possibilitariam a criação de um ambiente capaz não apenas de localizar os pontos de mudança, mas também de gerar *sublogs* e submodelos. A partir desse ponto, foram conduzidos testes para descobrir informações adicionais que influenciavam na sensibilidade dos ajustes de outros atributos, como o tamanho da janela e o fator de ajuste de valores. Além disso, foi implementada a inclusão de mais um algoritmo detector.

b) Configuração *cf* do método de detecção de mudança de processo *md*;

Para cada *log* de eventos, foram geradas combinações de parâmetros de configurações que pudessem apresentar variações nos resultados de detecção.

c) Detecção de cada ponto de mudança cp_i em L;

Todas as detecções realizadas com as distintas variações de parâmetros geraram os respectivos resultados de *sublogs* e submodelos, os quais foram armazenados em pastas identificadas com as parametrizações utilizadas.

d) Para cada cp_i realizou-se a:

- localização s_i de cada ponto de mudança cp_i em L;
- obtenção, em função de s_i, o *sublog* de traços g_i de L para a descoberta do novo processo m_i;
- mensuração do modelo m_i por meio do cálculo das métricas de qualidade *fitness* (vf_i), *generalização* (vg_i), *simplicidade* (vs_i) e *precisão* (vp_i).

Nesta etapa do trabalho, foi realizada a implementação do módulo responsável pelo cálculo das métricas de qualidade para o modelo do *log* geral, assim como para os submodelos dos *sublogs* gerados em cada ponto de mudança.

e) Obtenção dos valores estatísticos da métricas de qualidade – *fitness* ($\overline{vf}$), *generalização* ($\overline{vg}$), *simplicidade* ($\overline{vs}$) e *precisão* ($\overline{vg}$);

Para cada conjunto de *sublogs* e submodelos, foram calculadas as estatísticas (média, mediana, máximo e mínimo) das métricas de qualidade a fim de realizar comparações com as métricas de qualidade do *log* geral.

f) Extração do conjunto de atributos F a partir das seguintes fontes: *log* de eventos L, métricas de qualidade *fitness* ($\overline{vf}$), *generalização* ($\overline{vg}$), *simplicidade* ($\overline{vs}$) e *precisão* ($\overline{vg}$), configuração *cf*, e método de detecção de mudança *md*;

g) Atualização do conjunto do treinamento *DSet* com a inserção de *RF* nele;

Para cada resultado gerado a partir de uma nova configuração, a base de dados foi atualizada com o registro de atributos que continham todas as informações relativas ao *log* e seus *sublogs*, bem como suas parametrizações e valores obtidos.

h) Filtragem de F dos melhores resultados na base de atributos $RF \subseteq F$;

Considerando que várias configurações foram testadas para cada log, foi aplicado um filtro para incluir apenas os melhores resultados – de acordo com as preferências estabelecidas na geração da base de dados utilizada no treinamento pelo classificador.

i) Geração do *DSet_b* a partir dos melhores resultados em *DSet*;

São considerados, para o treinamento do modelo preditivo, apenas os melhores resultados – de acordo com as preferências de análise *FxP, FxG* e *FxS.* Para os apresentados, foi adotada a preferência pelo registro com o maior valor de métricas de *Fitness* + *Precisão* (Análise *FxP*).

j) Realização de treinamento, teste e validação do modelo de previsão.

Com a utilização do classificador *XGBoost*, foi possível obter as configurações mais adequadas, alcançando o objetivo de autoconfiguração do método proposto.

Seguindo o procedimento delineado anteriormente, foram conduzidos ensaios visando a construção e validação do método proposto – mais precisamente do procedimento de autoconfiguração para a detecção de mudanças de processo. Vale ressaltar, conforme mencionado previamente, que o ponto de partida para a implementação do fluxo de trabalho, que envolve a seleção de atributos e o treinamento do modelo de previsão de configuração, reside na obtenção de bases de dados ou logs de eventos.

4.1 BASE DE DADOS E SUAS ESPECIFICIDADES

A condução dos ensaios foi realizado sobre uma base de dados pública, disponível[12] no formato mxml – doravante, denominada de "Conjunto de Dados 1", ou simplesmente *DSet01*. Esta foi criada especificamente para testar a descoberta de mudanças de processos (9). A Tabela 16 ilustra a especificidade sobre a qual a referida base de dados foi criada, da mesma forma que inclui o comportamento de seus eventos e diversos padrões de mudança de processo. Deve-se notar que, a escrita original, em inglês, da Tabela 16 em questão, foi mantida na expectativa de manter-se inalterado o enunciado original.

[12] Disponível em: https://data.4tu.nl/articles/dataset/Business_Process_Drift/12712436

Tabela 16 – Padrões de mudanças

Code	*Change pattern*	*Category*	*Description*
cb	*Make fragment skippable/non-skippable*	O	*"Prepare acceptance pack" and "Check if home insurance quote is requested" turned skippable.*
cd	*Synchronize two fragments*	R	*"Assess loan risk" can only be performed after both "Appraise property" and "Check credit history".*
cf	*Make two fragments conditional/sequential*	R	*"Send acceptance pack" and "Send home insurance quote" from conditional to sequential.*
Cm	*Move fragment into/out of conditional branch*	I	*"Prepare acceptance pack" moved into conditional branch containing "Send acceptance pack" and "Send home insurance quote".*
cp	*Duplicate fragment*	I	*Fragment "Check credit history" and "Assess loan risk" duplicated after "Verify repayment agreement".*
IOR	I: re O: cb R: cd	IOR	*I: "Added activity" included.* *O: "Reject application" turned skippable.* *R: "Added activity" e "Reject application" synchronized.*
IRO	I: re R: cd O: lp	IRO	*I: Fragment containing "Added activity", "Verify repayment agreement", "Prepare acceptance pack" included.* *R: "Added activity" synchronized with "Verify repayment agreement", "Prepare acceptance pack".* *O: Fragment containing "Approve application", "Verify repayment agreement", "Prepare acceptance pack", "Added activity" turned loopable.*
lp	*Make fragment loopable/non-loopable*	O	*"Assess eligibility" turned loopable.*
OIR	O: lp I: re R: sw	OIR	*O: Fragment "Check credit history", "Assess loan risk", "Appraise property" turned loopable.* *I: "Added activity" included.* *R: "Check credit history" and "Assess loan risk" swapped.*
ORI	O: lp R: cf I: re	ORI	*O: Fragment "Check if home insurance quote is requested", "Send home insurance quote", "Send acceptance pack", "Verify repayment agreement" turned loopable.* *R: Change "Send home insurance quote", "Send acceptance pack" from conditional to sequential.* *I: "Added activity" included.*
pl	*Make two fragments parallel/sequential*	R	*Fragment "Appraise property", "Check credit history", "Assess loan risk" from parallel to sequential.*
pm	*Move fragment into/out of parallel branch*	I	*"Prepare acceptance pack" moved into a parallel branch with "Send home insurance quote".*
re	*Add/remove fragment*	I	*"Assess eligibility" removed.*

(fim)

RI	R: cf I: cp	RI	*R: Fragment "Prepare acceptance pack", "Check if home insurance quote is requested" from sequential to conditional after "Send home insurance quote".* *I: Fragment "Prepare acceptance pack", "Check if home insurance quote is requested" duplicated after "Assess eligibility".*
ROI	R: pl O: lp I: rp	ROI	*R: Fragment "Check credit history", "Assess loan risk" and "Appraise property" from parallel to sequential ("Appraise property" first").* *O: Fragment "Appraise property", Check credit history" turned loopable.* *I: "Check credit history" replaced by "Replaced activity".*
rp	*Substitute fragment*	I	*"Verify repayment agreement" replaced by "Replaced activity".*
sw	*Swap two fragments*	R	*Swap fragments "Prepare acceptance pack" followed by "Check if home insurance is requested" with "Verify repayment agreement".*

Fonte: adaptado de (9).

São onze padrões simples e seis complexos – derivados dos simples. Os padrões simples são*: cb, cd, cf, cm, cp, lp, pl, pm, re, rp* e *ws*, enquanto os complexos são: *IOR, IRO, OIR, ORI, RI e ROI*. Deve-se notar que se manteve a mesma classificação proposta originalmente em (9) para os padrões simples: I – inserção, R – *ressequencialização* e O – *opcionalização*, mas alterou-se a categorização do padrão *sw* (*swap two fragments*) para R – e tal alteração também foi repercutida no padrão OIR. Ademais, foram realizadas as seguintes alterações (43), a saber: nome do padrão RIO para RI, por não haver *opcionalização* incluída no modelo alterado – definido em *DSet01*. A recategorização do padrão *sw (swap two fragments)* foi feita para *R* por se ajustar melhor à descrição.

Para cada um dos 17 padrões de mudança de processo, o conjunto de dados *DSet01* encerra quatro logs de eventos, com os respectivos tamanhos: 2.500, 5.000, 7.500 e 10.000 traços ou instâncias completas. Cada *log* de eventos contém nove pontos de mudanças injetados a uma distância fixa equivalente a 10% do tamanho de cada log, respectivamente, a saber: 250, 500, 750 e 1.000 traços.

Em resumo, a estratégia de criação de um *log* de eventos com mudanças de processo pode ser simples e bem-definida. Por exemplo, considere M_1 e M_2 dois modelos de processos com padrões de mudança diferentes (Tabela 16): a) gera-se, por meio de simulação computacional, o *log* de eventos B_1 a partir do modelo M_1, e o *log* de eventos D_1 a partir do modelo M_2, em que B_1 é aquele *log* que contém registros a partir do modelo de referência base M_1, enquanto D_1 é o *log que* contém registros gerados a partir do modelo alterado – que representa o modelo de processo após uma mudança; b) cria-se um terceiro *log* de eventos

DSet01, juntando-se B_1 e D_1 de forma alternada k vezes, em que k é o número de mudanças presente em *DSet01* no final do processo de criação. No intervalo de tempo em que as alternâncias entre B_1 e D_1 (e vice-versa) ocorrem, são gerados e incluídos eventos seguindo uma distribuição de probabilidades (e.g. exponencial). Obviamente, essa dinâmica pode ser mais elaborada do que como foi aqui descrita.

Também foram realizados testes em um segundo conjunto de dados – *DSet02* –, usando os mesmos padrões de mudança definidos no *DSet01* (Tabela 16) e a mesma proposta de mudança de um modelo base para um modelo alterado. O conjunto de dados 2 contém três tamanhos de log: 3.000, 4.500 e 8.000 traços.

Além disso, foi incluído ainda uma base de dados real – que representa um processo de gerenciamento de ingressos de uma empresa italiana, que se encontra disponível em domínio público[13]. O *log* de eventos contém 4.580 casos e 14 atividades de janeiro de 2010 a janeiro de 2014. As bases selecionadas para serem utilizadas neste trabalho foram escolhidas por se tratar de base já validadas em (43) outros trabalhos relativos à mudança de conceito, e por serem domínio público.

4.2 DETECÇÃO DE MUDANÇA DE PROCESSO

A detecção de mudança de processo foi realizada utilizando a ferramenta IPDD. Para todos os logs que fizeram parte dos testes, foram geradas 30 combinações distintas, variando os seguintes atributos: detector (Adwin e HDDM-W), parâmetros dos detectores (Delta, *Drift Confidence, Warning Confidence, Lambda, Two Sided Test*), tamanho de janela, abordagem (Trace-by-*Trace e Windowing*) e fator. Cada variação única foi transformada em uma classe distinta, como detalhado na Tabela 20 – apresentada nos resultados (Item 4.3).Utilizando essa estratégia, pôde-se assegurar que o método suporta a incorporação de múltiplos detectores e seus respectivos parâmetros, conforme necessário para aplicações em trabalhos futuros. A ferramenta foi empregada para identificação de pontos de mudança em cada variação, resultando na geração de submodelos e *sublogs* correspondentes.

A partir dos *sublogs* e submodelos gerados, calcularam-se os valores das métricas para cada combinação em cada log. Cada uma resultou em um registro na base de atributos, que foi

[13] Disponível em: https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb.

utilizada na aprendizagem de máquina para que o classificador pudesse sugerir a autoconfiguração.

Todas as informações foram geradas em um *framework* desenvolvido especificamente para o cálculo das métricas e demais informações. Vale destacar que, a cada análise, todos os dados são gerados com base nos resultados da parametrização informada, incluindo informações como o nome do arquivo, o total de eventos, o total de casos, o total de atividades, a média dos tempos de duração dos casos, a mediana dos tempos de duração deles, o número de variantes do processo, a média das frequências das atividades e a média das frequências dos recursos.

Além disso, as informações do processo de detecção de mudança, dos detectores e de seus parâmetros incluem: tipo de abordagem (*Trace-by-Trace; windowing*), perspectiva de mudança, detector (ADWIN; HDDM_W), tamanho da janela, valor do fator, valor do parâmetro delta, valor do parâmetro *drift confidence*, valor do parâmetro *warning confidence*, valor do parâmetro lambda, valor do parâmetro *two sided test*.

As informações relacionadas às métricas de qualidade do *log* e do modelo completo compreendem: valor da métrica *fitness* do *log* completo, valor da métrica precisão do *log* completo, valor da métrica generalização do *log* completo, valor da métrica simplicidade do *log* completo.

Quanto às métricas de qualidade dos *sublogs* e submodelos, são gerados valores a partir dos resultados dos *sublogs* e submodelos, os quais incluem métricas como *fitness, precisão, generalização e simplicidade.* Esses valores são calculados em termos de média, mediana, desvio padrão, valor mínimo e valor máximo para cada uma dessas métricas.

O experimento foi conduzido mediante o emprego de registros de eventos, conforme delineado no tópico anterior (4.1 – Base de dados e suas especificidades). Cada análise, conduzida sob uma parametrização única, resultou na geração de um conjunto de atributos que serviu como entrada para o classificador. Este conjunto foi designado como *DSet*, abrangendo os experimentos de todas as variações originalmente testadas. Esta base de dados é composta por 2.805 registros, numerados de 0 a 2.804, cada um contendo um total de 55 colunas de dados.

A Tabela 17 ilustra os cálculos das métricas de qualidade, incluindo *fitness, precisão, simplicidade e generalização*, para o modelo de processo descoberto a partir do *log* de eventos geral, em que*log* este é identificado na primeira coluna da tabela como GR_0. Além disso, são apresentados os cálculos das mesmas métricas para cada submodelo descoberto a partir dos *sublogs* criados após a detecção de pontos de mudança. Cada submodelo é identificado na

primeira coluna da tabela, sendo o primeiro chamado de CP_1, o segundo de CP_2, e assim por diante. As colunas *início* e *fim* denotam os pontos de início e fim do log.

A Tabela 17 exemplifica os cálculos das métricas de qualidade utilizando o *log* Cb5k.xes como exemplo, com as seguintes configurações: detector ADWIN, janela de tamanho 75, delta de 0,3 e abordagem *Trace-by-Trace*.

Tabela 17 – Exemplo de cálculo das métricas de qualidade do modelo de processo descoberto geral (GR) e cada modelo após detecção de ponto de mudança (CP). *Log* de eventos Cb5k.xes

Modelo Processo	***Log* de eventos**		***Fitness***	**Precisão**	**Generalização**	**Simplicidade**
	Início	***Fim***				
GR_0	1	5000	1	0,61	0,98	0,71
CP_1	0	74	1	0,98	0,83	0,83
CP_2	511	585	1	0,98	0,76	0,80
CP_3	1343	1417	1	0,99	0,82	0,83
CP_4	1535	1609	1	0,97	0,80	0,80
CP_5	4511	4585	1	0,98	0,82	0,80

Fonte: elaborada pela autora.

Para permitir uma análise comparativa posterior entre os resultados obtidos no modelo geral e aqueles encontrados nos *sublogs*, realizou-se uma comparação entre os valores calculados para o modelo geral e os valores medianos dos modelos dos *sublogs* de eventos, conforme ilustrado na Tabela 18. Esses *sublogs*são delimitados entre dois pontos de mudança.

Tabela 18 – Mediana (M) e desvio padrão (D) das métricas de qualidade de cada processo descoberto para cada *sublog* criado após detecção de um ponto de mudança (CP) e respectivas métricas para o modelo geral (GR)

Modelo	***Log* de eventos**						
Processo	***Início***	***Fim***		***Fitness***	**Precisão**	**Generalização**	**Simplicidade**
GR_0 (cb5)	1	5000		1.00	0.61	0.98	0.71
CP_{1-10} (cb5)	1	4999	M	1.00	0.98	0.80	0.82
			D	±0.00	±0.01	±0.03	±0.02

Fonte: elaborada pela autora.

Incorporou-se, adicionalmente, um exemplo que abarca integralmente as informações e os cálculos das métricas realizados para cada registro. Neste caso, manteve-se a utilização do *log* Cb5k, conforme evidenciado na Tabela 19.

Tabela 19 – Exemplo de Atributos – *DSet*

Campo	Valor
Nome do arquivo (log)	cb5kxes
Total de eventos (nr_eventos)	100836
Total de casos (nr_casos)	5000
Total de atividades (nr_atividades)	19
Média dos tempos de duração dos casos (media_tempo_casos)	56135.72
Mediana dos tempos de duração dos casos (mediana_tempo_casos)	18162.95
Número de variantes do processo (nr_variantes)	114
Média das frequências das atividades (media_freq_ativ)	5307.16
Média das frequências dos recursos (mediana_freq_rec)	3726.00
Tipo de abordagem: trace-by-trace; windowing (tipo_abordagem)	*trace-by-trace*
perspectiva de mudança: Control_Flow (perspectiva_mudanca)	*Control_Flow*
Detector: ADWIN; HDDM_W (detector)	*Adwin*
Valor do parâmetro delta (delta)	0.3
Fitness – *Log* Geral (fitness_log_completo)	1.00
Precisão – *Log* Geral (precisao_log_compelto)	0.61
Generalização – *Log* Geral (generalizacao_log_completo)	0.98
Simplicidade – *Log* Geral (simplicidade_log_completo)	0.71
Número de atividades no traço – *Log* Geral (log_completo_count_ativ_trace)	5000
Mediana de tempo de atividades por traço – *Log* geral (log_completo_mean_ativ_trace)	0.134
Desvio padrão do tempo de atividades por traço (log_completo_std_ativ_trace)	0.097
Valor mínimo de tempo de atividade por traço (log_completo_min_ativ_trace)	0.0
25% tempo de atividade por traço (log_completo_25%_ativ_trace)	0.008
Mediana de tempo de atividade por traço (log_completo_50%_ativ_trace)	0.054
75% tempo de atividade por traço (log_completo_75%_ativ_trace)	0.186
Valor máximo de tempo de atividade por traço (log_completo_max_ativ_trace)	0.996
Fitness – Mediana dos *sublogs* (mediana_sublog_fitness)	1.00
Precisão – Mediana dos *sublogs* (mediana_sublog_precisao)	0.98
Generalização – Mediana dos *sublogs* (mediana_sublog_generalizacao)	0.82
Simplicidade – Mediana dos *sublogs* (mediana_sublog_simplicidade)	0.80
Fitness – Média dos *sublogs* (media_sublog_fitness)	1.00
Precisão – Média dos *sublogs* (media_sublog_precisao)	0.98
Generalização – Média dos *sublogs* (media_sublog_generalizacao)	0.81
Simplicidade - Média dos *sublogs* (media_sublog_simplicidade)	0.81
Fitness – Mínimo dos *sublogs* (min_sublog_fitness)	1.00
Precisão – Mínimo dos *sublogs* (min_sublog_precisao)	0.97
Generalização – Mínimo dos *sublogs* (min_sublog_generalizacao)	0.76
Simplicidade – Mínimo dos *sublogs* (min_sublog_simplicidade)	0.80
Fitness – Máximo dos *sublogs* (max_sublog_fitness)	1.00
Precisão – Máximo dos *sublogs* (max_sublog_precisao)	0.99
Generalização – Máximo dos *sublogs* (max_sublog_generalizacao)	0.83
Simplicidade – Máximo dos *sublogs* max_sublog_simplicidade	0.83
Fitness – Desvio padrão dos *sublogs* (std_sublog_fitness)	0.00

Precisão – Desvio padrão dos *sublogs* (std_sublog_precisao)	0.01
Generalização – Desvio padrão dos *sublogs* (std_sublog_generalizacao)	0.03
Simplicidade – Desvio padrão dos *sublogs* (std_sublog_simplicidade)	0.02
Tamanho da Janela (tamanho_janela)	75
Fator de ajuste - escala de valores - (fator)	100
Valor do parâmetro *drift confidence (*drift_confidence*)*	0
Valor do parâmetro *warning confidence (*warning_confidence)	0
Valor do parâmetro lambda (lambda_val)	0
Valor do parâmetro *two sided test* (two_sided_test)	*True*

Fonte: elaborada pela autora.

A cada análise realizada, gerou-se um novo registro na base de atributos de *log* de eventos. O *DSet* é um arquivo *.csv* com nome *base_caracteristicasT.csv*, o qual contém todos os atributos descritos na Tabela 19, de exemplo.

O dicionário de dados da base de atributos encontra-se no Anexo A.

4.3 GERAÇÃO DA BASE DE ATRIBUTOS

Foi criado o conjunto de dados *DSet*, gerado a partir da inclusão dos registros de atributos dos logs. Para isso, foram executadas várias configurações com intuito de abranger a diversidade de possíveis combinações. Cada configuração foi aplicada consistentemente a todos os logs, resultando em um registro no *DSet* destinado ao treinamento. Ademais, cada conjunto específico de parâmetros gerou uma *classe*, que serve como a variável-alvo a ser prevista. Considerando os experimentos realizados com essas diversas combinações, obtivemos um total de 30 classes, conforme apresentado na Tabela 20:

Tabela 20 – Classes

Classe	Detector	Tamanho janela	Abordagem	Fator	Delta	*Drift confidence*	*Warning confidence*	Lambda	*Two sided test*
0	adwin	25	*trace-by-trace*	100	0.002	0	0	0	True
1	adwin	25	*trace-by-trace*	100	0.05	0	0	0	True
2	adwin	25	*trace-by-trace*	100	0.1	0	0	0	True
3	adwin	25	*trace-by-trace*	100	0.3	0	0	0	True
4	adwin	75	*trace-by-trace*	100	0.002	0	0	0	True
5	adwin	75	*trace-by-trace*	100	0.05	0	0	0	True
6	adwin	75	*trace-by-trace*	100	0.1	0	0	0	True
7	adwin	75	*trace-by-trace*	100	0.3	0	0	0	True
8	adwin	100	*trace-by-trace*	100	0.002	0	0	0	True
9	adwin	100	*trace-by-trace*	100	0.05	0	0	0	True
10	adwin	100	*trace-by-trace*	100	0.1	0	0	0	True
11	adwin	100	*trace-by-trace*	100	0.3	0	0	0	True
12	hddm_w	25	*trace-by-trace*	10	0	0.001	0.005	0.05	True
13	hddm_w	75	*trace-by-trace*	10	0	0.001	0.005	0.05	True
14	hddm_w	100	*trace-by-trace*	10	0	0.001	0.005	0.05	True
15	adwin	25	*windowing*	100	0.002	0	0	0	True
16	adwin	25	*windowing*	100	0.05	0	0	0	True
17	adwin	25	*windowing*	100	0.1	0	0	0	True
18	adwin	25	*windowing*	100	0.3	0	0	0	True
19	adwin	75	*windowing*	100	0.002	0	0	0	True
20	adwin	75	*windowing*	100	0.05	0	0	0	True
21	adwin	75	*windowing*	100	0.1	0	0	0	True
22	adwin	75	*windowing*	100	0.3	0	0	0	True
23	adwin	100	*windowing*	100	0.002	0	0	0	True
24	adwin	100	*windowing*	100	0.05	0	0	0	True
25	adwin	100	*windowing*	100	0.1	0	0	0	True
26	adwin	100	*windowing*	100	0.3	0	0	0	True
27	hddm_w	25	*windowing*	10	0	0.001	0.005	0.05	True
28	hddm_w	75	*windowing*	10	0	0.001	0.005	0.05	True
29	hddm_w	100	*windowing*	10	0	0.001	0.005	0.05	True

Fonte: elaborada pela autora.

As classes geradas foram associadas ao conjunto de dados *(DSet),* resultando na criação de um novo atributo – correspondente à variável-alvo, que será utilizada para o treinamento do algoritmo classificador. Utilizando esta abordagem, permite-se a inserção de diversos detectores distintos, cada um com seus respectivos parâmetros.

Outrossim, uma pré-seleção dos atributos foi realizada a partir de uma análise de correlação entre variáveis com o auxílio da ferramenta *ProfileReport*[14] (versão 4.6.1).

[14] Disponível em: https://pypi.org/project/pandas-profiling/.

O conjunto de dados *(DSet)* que contém os experimentos de todas as variações testadas originalmente possui 2.805 registros. No entanto, para analisar apenas aqueles com as melhores configurações, foram selecionados apenas um registro de cada log, utilizando o seguinte critério de qualidade: o registro com o maior valor de métricas de *Fitness* + *Precisão* (*Análise FxP*), o que resultou na criação de um novo conjunto de dados, denominado *DSet_b.*

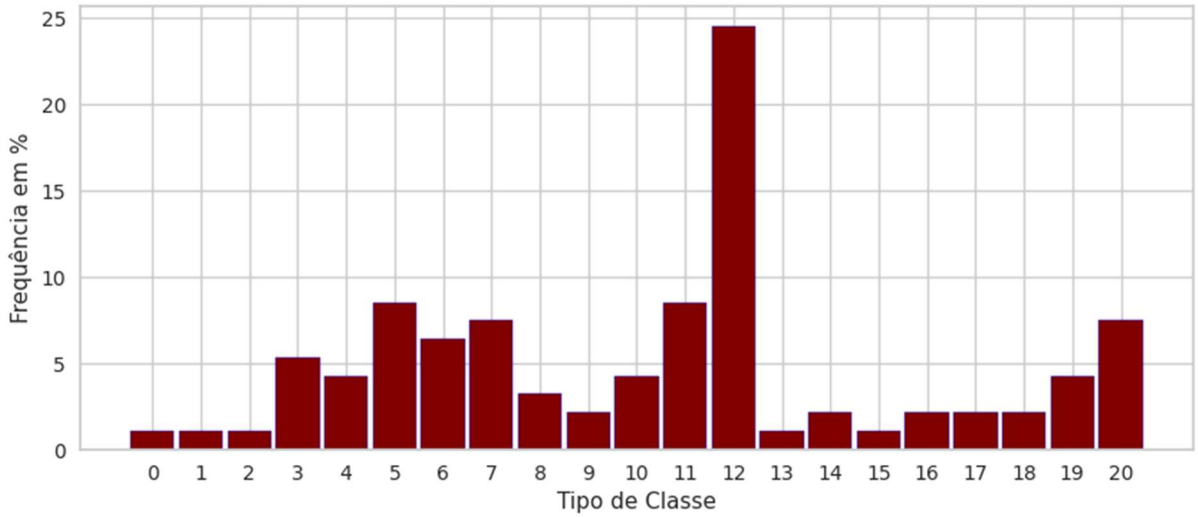
Ao efetuar a seleção dos registros mais destacados no conjunto de *DSet_b*, foram retidas, exclusivamente, as classes que se alinham com as parametrizações associadas aos resultados mais favoráveis, totalizando 21 classes – essas representadas numericamente de 0 a 20, conforme apresentado na Tabela 21 e no Gráfico 2.

Tabela 21 – Distribuição de Classes e Suas Frequências no *DSet_b* com Base nos Melhores Resultados

Melhores Classes *DSet_b*	**Freq.**	**Detector**	**Tamanho janela**	**Abordagem**	**Fator**	**Delta**	***Drift confidence***	***Warning confidence***	**Lambda**	***Two sided test***
0	1	adwin	25	*trace-by-trace*	100	0.002	0	0	0	True
1	1	adwin	25	*trace-by-trace*	100	0.05	0	0	0	True
2	1	adwin	25	*trace-by-trace*	100	0.3	0	0	0	True
3	5	adwin	75	*trace-by-trace*	100	0.002	0	0	0	True
4	4	adwin	75	*trace-by-trace*	100	0.05	0	0	0	True
5	8	adwin	75	*trace-by-trace*	100	0.1	0	0	0	True
6	6	adwin	75	*trace-by-trace*	100	0.3	0	0	0	True
7	7	adwin	100	*trace-by-trace*	100	0.002	0	0	0	True
8	3	adwin	100	*trace-by-trace*	100	0.05	0	0	0	True
9	2	adwin	100	*trace-by-trace*	100	0.1	0	0	0	True
10	4	adwin	100	*trace-by-trace*	100	0.3	0	0	0	True
11	8	hddm_w	75	*trace-by-trace*	10	0	0.001	0.005	0.05	True
12	23	hddm_w	100	*trace-by-trace*	10	0	0.001	0.005	0.05	True
13	1	adwin	25	*windowing*	100	0.002	0	0	0	True
14	2	adwin	75	*windowing*	100	0.002	0	0	0	True
15	1	adwin	100	*windowing*	100	0.002	0	0	0	True
16	2	adwin	100	*windowing*	100	0.05	0	0	0	True
17	2	adwin	100	*windowing*	100	0.3	0	0	0	True
18	2	hddm_w	25	*windowing*	10	0	0.001	0.005	0.05	True
19	4	hddm_w	75	*windowing*	10	0	0.001	0.005	0.05	True
20	7	hddm_w	100	*windowing*	10	0	0.001	0.005	0.05	True

Fonte: elaborada pela autora.

Gráfico 2 – Distribuição de Frequências das Classes no *DSet_b* com Base nos Melhores Resultados



Fonte: elaborada pela autora.

4.4 AUTOCONFIGURAÇÃO

Conforme já mencionado neste trabalho, a autoconfiguração de um detector de mudança de processo é uma tarefa desafiadora que requer uma abordagem rigorosa. A partir de uma busca exaustiva de parâmetros e testes em massa de combinações de configurações, juntamente com a seleção criteriosa de informações relevantes, esta etapa do estudo apresenta os resultados obtidos e os esforços dedicados ao treinamento da base de dados para realizar predições das configurações apropriadas. Neste contexto, apresenta-se as resultas alcançadas com o treinamento na base de dados denominada *DSet_b*, que representa nossa base de teste.

A decisão de gerar a base de dados com o maior número possível de combinações, seguida da filtragem do conjunto de informações para incluir apenas as melhores parametrizações, resultou em um conjunto de treinamento composto por 94 registros no caso do *DSet_b*. É relevante destacar que se trata de uma base de dados de tamanho reduzido, o que representa um desafio para a tarefa de treinamento devido à limitação de informações disponíveis. Isso implica na necessidade de utilizar um classificador capaz de lidar com essa característica específica.

Nesta seção, portanto, serão apresentados os resultados da sequência dos protocolos experimentais, destacando o desempenho dos classificadores *Gradient Boosting* e XGBoost, além de como foram executados os protocolos para otimização de hiperparâmetros e validação.

4.4.1 Desempenho e Comparação dos Classificadores Utilizados

Como mencionado anteriormente neste trabalho, foram conduzidos testes com diversos classificadores em busca do mais adequado. No entanto, neste momento, apresentaremos apenas os resultados iniciais obtidos pelos dois classificadores de melhor desempenho.

Como protocolo experimental, foram selecionados dois classificadores para avaliação, nomeadamente *GradientBoosting* e *XGBoost*. Ambos foram configurados com seus parâmetros *defaults,* e as resultas indicam que o classificador XGBoost alcançou o melhor desempenho, conforme observado na Tabela 22.

Para avaliar a performance, foram empregadas as métricas de avaliação dos resultados do classificador. Ressalta-se que, neste contexto, faz-se menção específica às métricas delineadas no Item 2.11.5, intitulado "Métricas de Avaliação na Análise de Desempenho do Classificador".

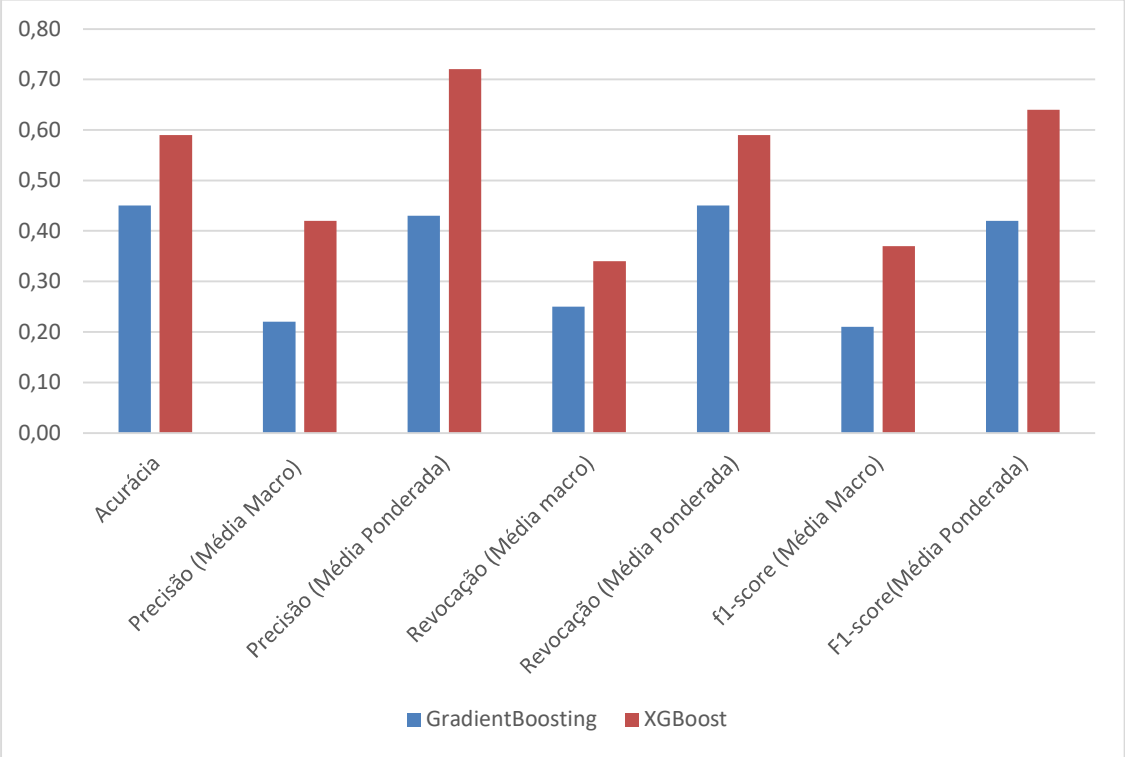
Tabela 22 – Comparação de Desempenho entre *Gradient Boosting* e XGBoost

Classificador	*Acurácia*	Precisão (Média Macro)	Precisão (Média Ponderada)	Revocação (Média Macro)	Revocação (Média Ponderada)	***f1-score*** (Média Macro)	***f1-score*** (Média Ponderada)
Gradient Boosting	0.45	0.22	0.43	0.25	0.45	0.21	0.42
XGBoost	0.59	0.42	0.72	0.34	0.59	0.37	0.64

Fonte: elaborada pela autora.

No intuito de aprimorar a visualização do desempenho do classificador *XGBoost*, enfatiza-se a sua representação gráfica no Gráfico 3, a fim de que esteja claro que todas as métricas analisadas neste contexto foram melhores na utilização do classificador XGBoost.

Gráfico 3 – Avaliação Comparativa de Desempenho entre *Gradient Boosting* e XGBoost com Parametrização Padrão



Fonte: elaborada pela autora.

Com base nos resultados obtidos, decidiu-se por focar na utilização do classificador *XGBoost* e proceder com as etapas de treinamento, ajustando os hiperparâmetros conforme necessário e aplicando métodos de avaliação.

4.4.2 Otimização dos Hiperparâmetros – *Grid Search*

Para otimizar os conjuntos de hiperparâmetros utilizados no classificador *XGBoost,* empregou-se a técnica denominada *Grid Search*, que realiza uma busca exaustiva pela melhor configuração dos principais parâmetros. Estes foram variados em sua totalidade para o classificador e incluíram 'max_depth' (2, 10, 1), 'n_estimators' (60, 220, 40), 'learning_rate' (0.1, 0.01, 0.05) e 'gamma' (0.5, 1, 1.5). Os demais mantivemos seus valores padrão.

Já os parâmetros fornecidos à função *Grid Search* foram os seguintes: 'estimator' (classificador base), 'param_grid' (os parâmetros mencionados anteriormente), 'scoring' ('roc_auc'), 'n_jobs' (10), 'cv' (10) e 'verbose' (True). Os melhores hiperparâmetros identificados foram: 'gamma' = 0.5, 'learning_rate' = 0.1, 'max_depth' = 2 e 'n_estimators' = 60.

É relevante destacar que a função *Grid Search* integra validação cruzada em sua estrutura.

Após a obtenção dos parâmetros ótimos, conduziu-se uma comparação entre os resultados alcançados com o classificador *XGBoost* e aqueles utilizando os hiperparâmetros com os valores padrão, conforme apresentado na Tabela 23.

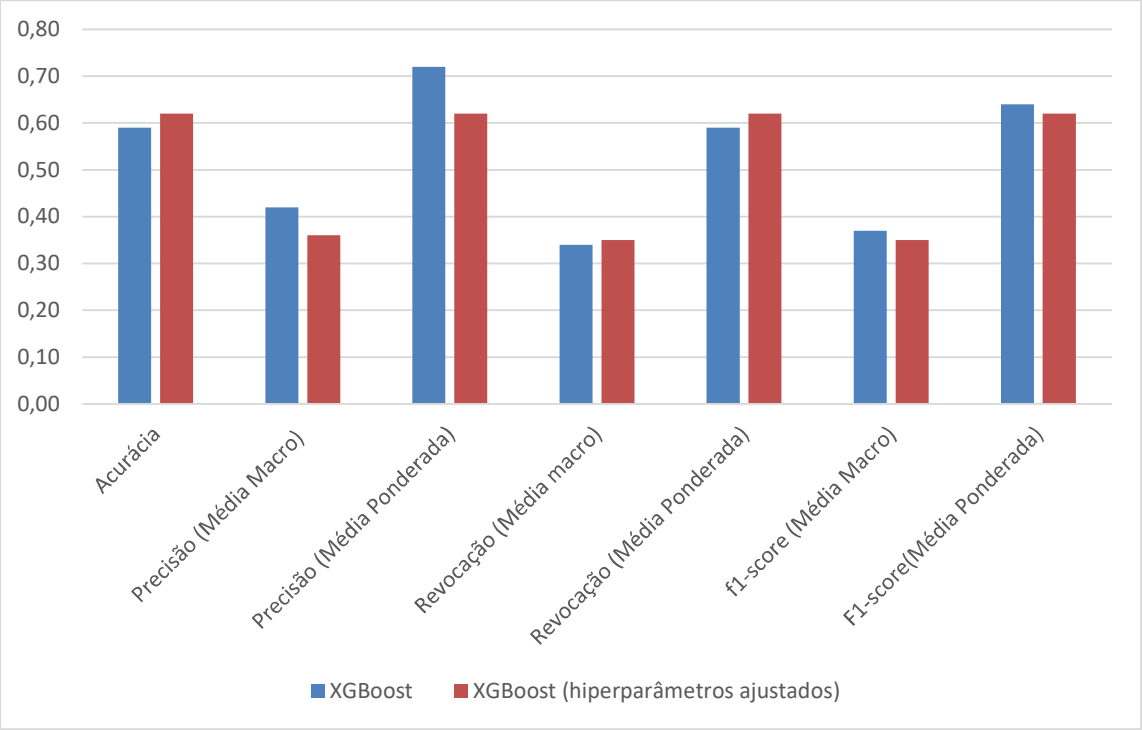
Tabela 23 – Análise comparativa dos resultados do algoritmo XGBoost antes e após o ajuste de hiperparâmetros

Classificador	*Acurácia*	Precisão (Média Macro)	Precisão (Média Ponderada)	Revocação (Média Macro)	Revocação (Média Ponderada)	***f1-score*** (Média Macro)	***f1-score*** (Média Ponderada)
XGBoost (Antes)	0.59	0.42	0.72	0.34	0.59	0.37	0.64
XGBoost (Após)	0.62	0.36	0.62	0.35	0.62	0.35	0.62

Fonte: elaborada pela autora.

Com o intuito de aprimorar a visualização dos resultados do algoritmo XGBoost antes e após o ajuste de hiperparâmetros, enfatiza-se a sua representação gráfica a seguir (Gráfico 4).

Gráfico 4 – Resultados do algoritmo XGBoost antes e após o ajuste de hiperparâmetros



Fonte: elaborada pela autora.

Os resultados apresentados acima têm a finalidade de validar a aplicação da aprendizagem de máquina no módulo de autoconfiguração do método proposto.

4.4.3 Validação *Leave-One-Out* (LOO): Avaliando o Desempenho do Modelo

Na fase de validação, ao término de todas as iterações do *Leave-One-Out (LOO),* a acurácia do modelo é calculada por meio da função *accuracy_score*. Esta compara as classes reais com as classes previstas, resultando na proporção de predições corretas em relação ao seu total. O LOO foi aplicado de duas maneiras: primeiro, utilizando os valores padrão do classificador *XGBoost*; segundo, empregando os melhores hiperparâmetros obtidos por meio da função *Grid Search.* Com a parametrização padrão, a acurácia calculada foi de 0,57, enquanto que, com os hiperparâmetros otimizados, a acurácia calculada foi de 0,55.

Portanto, com base nesses resultados, pode-se concluir que o método proposto demonstrou eficácia ao utilizar o classificador XGBoost, atendendo satisfatoriamente às expectativas estabelecidas.

4.5 DISCUSSÃO

A discussão do presente estudo segue a ordem das questões de pesquisa previamente apresentadas. Tem-se a primeira indagação (Q1): “Técnicas de aprendizado de máquina podem contribuir, por meio de um modelo preditivo, para a sugestão dos parâmetros utilizados no detector de mudança de conceito?”

Neste contexto, é possível afirmar que a autoconfiguração do detector de mudança apresenta o potencial necessário para auxiliar o analista de processo na aplicação de ferramentas de detecção de mudanças.

Mediante a utilização de técnicas de aprendizado de máquina, especificamente com o emprego do classificador XGBoost, foi alcançado uma acurácia de 0.62, mesmo diante da limitação de uma base de dados de pequena escala. A acurácia, que representa a proporção de predições corretas em relação ao total realizadas pelo modelo, revelou-se satisfatória nas condições testadas. Cabe ressaltar que este resultado, considerando a reduzida quantidade de registros, é percebido como positivo para este estudo, e há perspectivas de aprimoramento com o aumento do número de atributos na referida base.

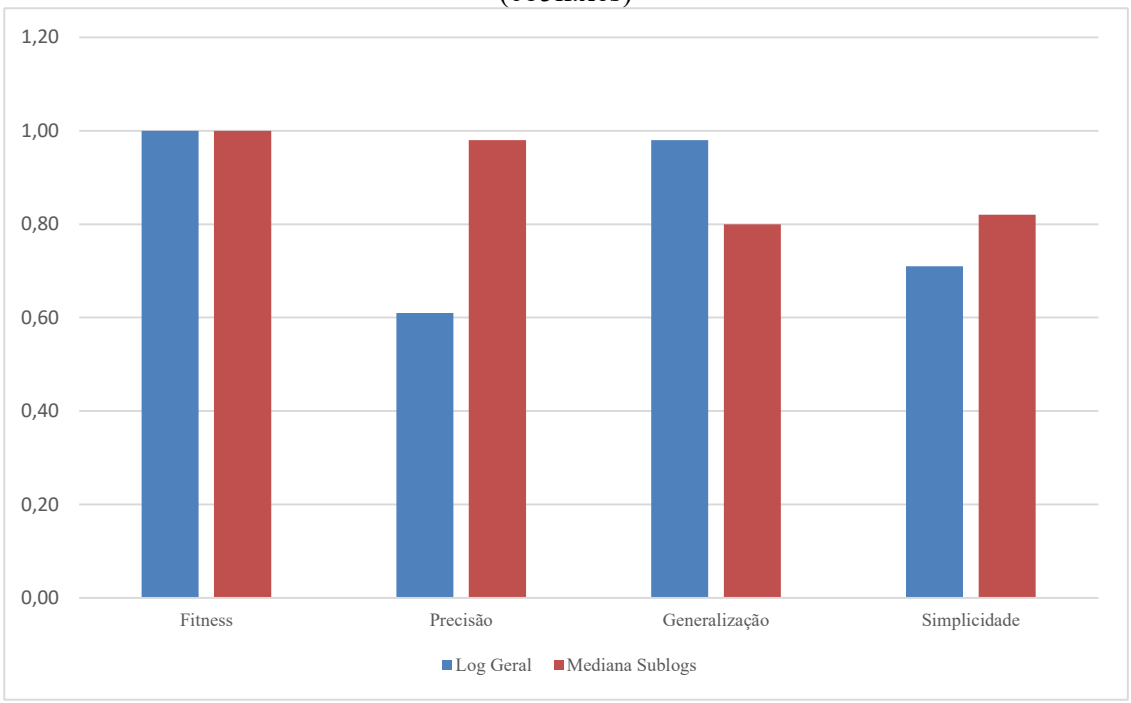
Ademais, é igualmente relevante salientar que a estratégia de criar classes para parametrização distinta demonstrou ser vantajosa. Tal abordagem possibilita a inserção de detectores de mudanças e suas configurações de forma flexível, adaptando-se conforme necessário para otimizar o desempenho do sistema

Na revisão da literatura, não foi identificado nenhum método semelhante ao proposto, portanto, as avaliações são realizadas com base em resultados inéditos.

No que concerne à questão de pesquisa Q2: “As métricas de qualidade dos processos de negócios, derivadas do *log* geral, quando comparadas às métricas geradas pelos modelos derivados de *sublogs* encontrados nos pontos de mudança, contribuem de forma objetiva para a avaliação dos modelos de processo gerados a partir de cada ponto de mudança?”

Ao analisar as resultas na extração das métricas do *log* geral e confrontá-las com os resultados dos *sublogs*, é notada a existência de diferenças e os benefícios ao combinar as métricas para análises. Este aspecto é observado no exemplo destacado Tabela 17 na Tabela 18, os quais deram origem à ilustração no Gráfico 5.

Gráfico 5 – Comparação das Métricas de Qualidade de Processo entre o *Log* Geral e *Sublogs* (cb5k.xes)



Fonte: elaborada pela autora.

Com essa constatação, é possível contribuir para a análise dos modelos de processo que passaram por mudanças, proporcionando uma visão mais precisa da realidade durante suas etapas de transformação. Dessa forma, o analista terá, à sua disposição, modelos que melhor representam a execução, permitindo uma interação mais alinhada às suas escolhas. Isso viabiliza, também, uma avaliação objetiva das mudanças identificadas pelo detector.

Adicionalmente, uma análise conduzida com base nas métricas de qualidade comparou os resultados médios dos melhores registros, que constituem todo o conjunto de dados *DSet_b*. Essa comparação foi realizada calculando a média e o respectivo desvio padrão, confrontando os resultados do log geral com os resultados dos sublogs, conforme Tabela 24.

Tabela 24 – Análise Comparativa das Métricas entre o *Log* Geral e *Sublogs* no Conjunto de Dados *DSet_b*

		Log Geral	*Sublogs*
Fitness	Media:	1.00	1.00
	±	0.00	0.00
Precisão	Media:	0.61	0.95
	±	0.06	0.07
Generalização	Media:	0.98	0.80
	±	0.01	0.08
Simplicidade	Media:	0.70	0.82
	±	0.70	0.03

Fonte: elaborada pela autora.

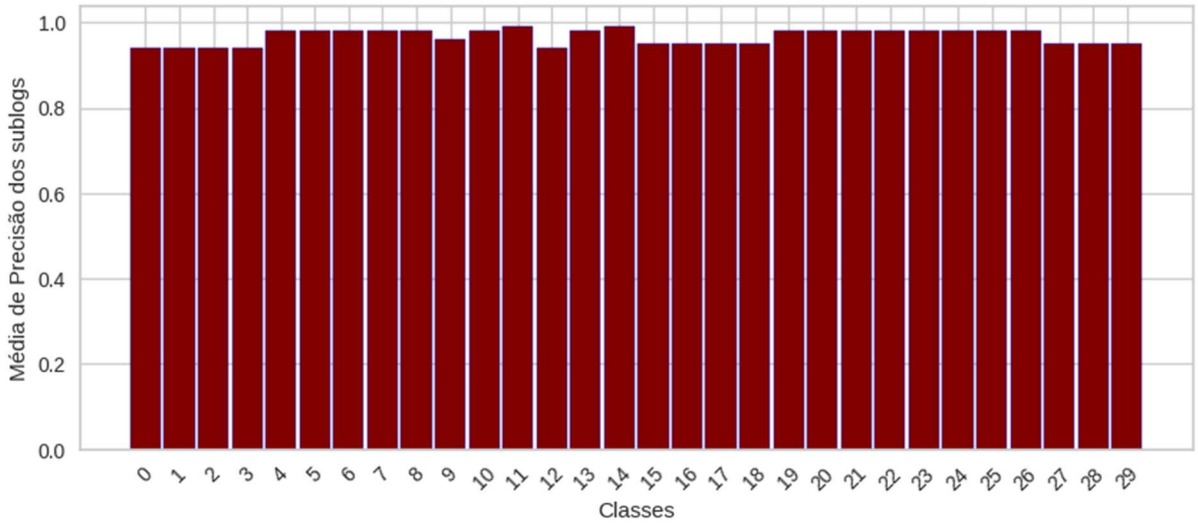
As métricas de *Fitness* para o *Log* Geral e *Sublogs* revelam uma média ótima (1.0) e um desvio padrão mínimo (0.0), indicando consistência e estabilidade no desempenho do modelo. Já as métricas de *Precisão* mostram uma média inferior para o *Log* Geral (0.61) em comparação com os *Sublogs* (0.95), com desvio padrão indicando maior variabilidade nos *Sublogs*. Quanto à *Generalização*, ambos os conjuntos apresentam médias significativas (0.98 para o *Log* Geral e 0.80 para os *Sublogs*), mas com um desvio padrão menor para o *Log* Geral, sugerindo maior consistência nos resultados. As métricas de *Simplicidade* indicam uma média superior para os *Sublogs* (0.82) em comparação ao *Log* Geral (0.70), com desvio padrão significativamente pequenos para ambas as médias. Em síntese, as métricas indicam um desempenho global robusto, destacando a consistência e eficácia do modelo em *Fitness, Precisão, Generalização* e *Simplicidade*.

Outrossim, é relevante ressaltar que foi constatado que configurações distintas no método de detecção podem resultar em valores diferentes para as métricas de qualidade. Essa observação foi verificada ao aplicar 30 classes distintas de parametrizações em um *log* e analisar individualmente cada métrica. Por exemplo, percebe-se uma variação nos valores das métricas ao utilizar configurações diferentes.

A título ilustrativo, destacam-se as médias da métrica de precisão – geradas a partir dos *sublogs* e submodelos –, extraídas dos resultados de todas as classes geradas a partir do *log* de eventos (cb5k.xes). Nota-se variações entre as configurações escolhidas, como ilustrado no Gráfico 6.

Nota: As classes correspondem às configurações informadas na Tabela 20.

Gráfico 6 – Valores médios de precisão em todas as classes a partir do *log* cb5k.xes



Fonte: elaborada pela autora.

Outro achado relevante é observado ao aplicar preferências nos resultados das métricas – no caso, considerando *Fitness* e *Precisão*, resultando na redução do número de classes para 20. Essa seleção consiste apenas nos resultantes mais destacados, a serem utilizados no modelo de previsão.

Ademais, ao filtrar os melhores resultados, foi possível identificar quais classes têm maior frequência na base de dados, conforme evidenciado na Tabela 21 e no Gráfico 2.

Nesse contexto, a classe 12 apresenta a maior frequência entre os resultados, conforme Tabela 25.

Tabela 25 – Configuração da Classe com maior Frequência para o conjunto *DSet_b*

Classe	Melhor	12
Atributos	Freq.	23
	Detector	hddm_w
	Tamanho de janela	100
	Abordagem	*trace-by-trace*
	Fator	10
	Delta	0
	Drift	0.001
	Warning	0.005
	Lambda	0.05
	Two	True

Fonte: elaborada pela autora.

Dado que não foi encontrado um método semelhante ao proposto, foram realizadas análises parciais das etapas, as quais podem ser consideradas satisfatórias, contribuindo para a autoconfiguração de detectores de mudança de conceito em mineração de processos, assim como para uma análise objetiva da qualidade dos modelos de processos descobertos a partir dos *sublogs* entre dois pontos de mudança devidamente identificados.

5 CONCLUSÃO

Nesta seção, são abordadas questões relevantes relacionadas à importância da pesquisa, suas contribuições, limitações e perspectivas para trabalhos futuros.

A mineração de processos desempenha um papel de significativa relevância na tomada de decisões, uma vez que ostenta a capacidade de gerar modelos de processos que representam os caminhos percorridos em cada fase do processo de negócios. Para tanto, o emprego de mapas de processos assume uma importância primordial para a visualização dos fluxos de execução das atividades.

Dessa forma, é relevante ressaltar que, na prática, é fundamental extrair informações pertinentes no contexto do processo identificado. Em outras palavras, a descoberta de um modelo de processo engloba uma série de características e desafios, sendo a gestão da mudança do processo ou mudança de conceito um dos mais notáveis. A gestão da mudança do processo refere-se à capacidade de lidar com situações em que este passa por modificações durante a análise. Portanto, é essential garantir a análise de modelos de alta qualidade por meio da utilização de métricas validadoras.

Durante a condução deste estudo, foi evidenciado que a identificação de tais alterações implica em um substancial esforço e requer um profundo conhecimento especializado. Além disso, a autoconfiguração de métodos para a detecção de mudanças pode fornecer um valioso apoio aos analistas no momento de eleger os parâmetros ideais. Adicionalmente, notou-se que a aplicação de técnicas de aprendizado de máquina se revela uma abordagem eficaz para abordar essa tarefa.

5.1 CONTRIBUIÇÕES

Em relação ao levantamento teórico realizado, foram identificadas lacunas representativas por esta pesquisa, as quais se manifestam como desafios substanciais. A partir dessa análise, emergiram as contribuições deste trabalho – originadas das motivações iniciais.

No que diz respeito ao desenvolvimento de um método capaz de recomendar a parametrização de um detector de mudança de conceito de forma automatizada, os resultados obtidos foram positivos e satisfatórios. Esses foram alcançados mediante um levantamento das informações essenciais, necessárias para a criação do modelo de previsão. Não obstante, é importante salientar que o foco de grandes esforços concentrou-se na geração da base de

atributos, que é o agrupamento de todas as informações conseguidas durante o processo e que foram utilizadas para o treinamento preditivo da máquina.

Concomitantemente, como discutido ao longo do documento, a aplicação da aprendizagem de máquina com a utilização do classificador *XGboos*t mostrou-se viável, mesmo quando se dispõe de conjuntos de dados reduzidos. O resultado final, portanto, evidencia a validade do método proposto, que se revela capaz de recomendar a parametrização para um detector de mudança de conceito. Já a aquisição de uma base de atributos é considerada uma contribuição adicional deste trabalho, visto que ela não apenas é utlizada para treinamento e previsão subsequentes, mas também pode ser explorada em várias análises, ao utilizar ferramentas de aprendizado de máquina e análise de dados.

No que se refere à avaliação dos modelos por meio das métricas de qualidade, confrontando o *log* geral com os resultados dos *sublogs*, torna-se evidente a presença de diferenças e os benefícios decorrentes da combinação dessas métricas para análise. A importância significativa das métricas reside na capacidade de automatizar a avaliação de um modelo de processo quanto ao *fitness, simplicidade, generalidade e precisão.*

No que se refere à concepção de um método para validar objetivamente o modelo de processo descoberto em cada ponto de mudança detectado, identificado e caracterizado – com base nas preferências do analista de processo, expressas na forma de métricas –, torna-se viável a obtenção de métricas específicas para cada modelo gerado durante esses pontos de mudança. Adicionalmente, é possível realizar análises em diversos modelos, conferindo ao analista a liberdade de escolher a métrica de qualidade mais alinhada à sua análise. A partir dessa escolha, os resultados mais destacados, que compõem a base de treinamento *DSet_b*, serão gerados. Em suma, o analista pode selecionar as configurações ideais com base na métrica de qualidade mais apropriada para sua análise.

A Tabela 26 apresenta as métricas do *log* geral e de cada *sublog* devidamente identificado – valores extraídos do *log* cb5k.xes, conforme detalhado no Passo 7 do Item 3 (Método Proposto).

Tabela 26 – Valores das métricas de qualidade do *Log* Geral e de *Sublogs*

Modelo Processo	*Log* de eventos					
	Início	***Fim***	***Fitness***	**Precisão**	**Generalização**	**Simplicidade**
GR_0	1	5000	1	0,61	0,98	0,71
CP_1	0	74	1	0,98	0,83	0,83
CP_2	511	585	1	0,98	0,76	0,80
CP_3	1343	1417	1	0,99	0,82	0,83
CP_4	1535	1609	1	0,97	0,80	0,80
CP_5	4511	4585	1	0,98	0,82	0,80

Fonte: elaborada pela autora.

Com base nos resultados obtidos na análise das métricas em relação a cada ponto de mudança, é factível afirmar que as métricas de qualidade de processos de negócios, calculadas a partir do *log* geral e contrastadas com as métricas geradas pelos modelos derivados dos *sublogs* encontrados nos pontos de mudança, desempenham um papel objetivo na validação da detecção de um ponto de mudança.

Conforme mencionado previamente na Seção 4.3, a execução do processo de geração da base de atributos, englobando a extração e seleção das informações do *log* e o cálculo das métricas de qualidade, revelou não ser uma atividade simples. Contudo, por meio da assistência e adoção da ferramenta IPDD, que proporcionou implementações de ajustes significativos, logrou-se êxito em abordar a eficiência à questão de pesquisa 2 (Q2) e seus objetivos específicos.

O método, por fim, viabiliza a condução de várias análises disponíveis, tais como a avaliação da conformidade ou o cálculo da qualidade do modelo recém-descoberto após a detecção de um ponto de mudança, e objetiva identificar aquele que mais se assemelha à situação real do processo. Cabe ressaltar que as métricas vinculadas às dimensões de qualidade – a saber, *fitness, precisão, simplicidade e generalização* – constituem elementos indispensáveis no processo de análise.

Além das contribuições originalmente propostas, foram identificadas outras durante a pesquisa, tais como:

- A geração da normalização dos dados de tempo de atividade por traço, cujos detalhes estão descritos no Item 2.10.3;

- O desenvolvimento de um conjunto de dados *(DSet)* contendo os atributos detalhados dos registros de eventos, derivado das detecções de mudanças. Este pode ser empregado em trabalhos futuros que demandem uma visão abrangente de cada registro de eventos;

- Implementações na ferramenta IPDD decorrentes das necessidades identificadas durante esta pesquisa, com detalhes apresentados no Item 2.10.1.

5.2 LIMITAÇÕES E TRABALHOS FUTUROS

Algumas limitações e sugestões para implementações futuras foram identificadas, tais como:

- A geração da base de atributos revelou-se uma atividade que demandou considerável tempo, em termos de processamento. Apesar dos resultados positivos obtidos no treinamento do classificador, é perceptível que prosseguir com o desenvolvimento e gerar mais registros pode contribuir para a obtenção de resultados ainda mais aprimorados, como aumento da acurácia;
- Foi observado que a expansão do número de detectores de mudança no âmbito da mineração de processos demanda cuidados detalhados e avaliações que ultrapassam os parâmetros sugeridos pela ferramenta. Ademais, identificou-se a necessidade de realizar testes em diversas escalas de valores, contemplando o *Fator de Ajuste* como um parâmetro a ser analisado na implementação de um novo detector, conforme especificado no Item 2.10.1 (Contribuições da Ferramenta IPDD para o método).

Visando aprimorar a metodologia e considerando as limitações destacadas nesta seção, torna-se viável a identificação dos seguintes trabalhos a serem realizados no futuro:

- Explorar a possibilidade de transformar o método em um *framework* com a integração de outros métodos de detecção;
- Desenvolver uma interface específica de interação entre o analista de processo e o *framework* de avaliação da qualidade do modelo descoberto. Esta permitirá que o profissional conduza a análise de acordo com suas escolhas e preferências, de maneira gráfica e intuitiva.

REFERÊNCIAS

1. VAN DER AALST, W. Process mining: Data science in action. Springer: Berlin, Heidelberg, 2016. DOI: 10.1007/978-3-662-49851-4.

2. DANIEL, F.; DUSTDAR, S.; BARKAOUI, K. Process Mining Manifesto. **IEEE - BPM 2011 Workshops**, [*s. l.*], v. 99, p. 169–194, 2011.

3. BIFET, A. et al. CD-MOA: Change detection framework for massive online analysis. *In*: **Lecture Notes in Computer Science** [including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics], [*s. l.*], v. 8207 LNCS, p. 92–103, 2013. DOI: 10.1007/978-3-642-41398-8_9.

4. GAMA, J. *et al.* Some studies in machine learning using the game of checkers. **Data Min Knowl Discov**, [*s. l.*], v. 23, n. 1, p. 128–168, 2011. DOI: 10.1007/s10618-010-0201-y.

5. MARIA, D. *et al*. A Survey on Concept Drift. **Process Mining**, [*s. l.*], v. 1, n. 1, p. 1–36, 2018.

6. LIN, L. *et al.* LCDD: Detecting Business Process Drifts Based on Local Completeness. **IEEE Trans Serv Comput**, [*s. l.*], v. 14, n. 8, pp. 1–1, 2020. DOI: 10.1109/TSC.2020.3032787.

7. DITZLER, G. *et al.* Learning in Nonstationary Environments: A Survey. **IEEE Comput Intell Mag**, [*s. l.*], v. 10, n. 4, p. 12–25, 2015. DOI: 10.1109/MCI.2015.2471196.

8. KURNIATI, A. P. *et al.* A multi-level approach for identifying process change in cancer pathways. *In*: **International Conference on Business Process Management**. Cham: Springer International Publishing, 2019. p. 595-607.

9. MAARADJI, A. *et al.* Fast and accurate business process drift detection. *In*: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, [*s. l.*], v. 9253, p. 406–422, 2015. DOI: 10.1007/978-3-319-23063-4_27.

10. PEFFERS, K. *et al.* A design science research methodology for information systems research. **Journal of Management Information Systems**, [*s. l.*], v. 24, n. 3, p. 45–77, 2007. DOI: 10.2753/MIS0742-1222240302.

11. FRÍAS-BLANCO, I. et al. Online and non-parametric drift detection methods based on Hoeffding's bounds. **IEEE Trans Knowl Data Eng**, [*s. l.*], v. 27, no. 3, pp. 810–823, Mar. 2015. doi: 10.1109/TKDE.2014.2345382.

12. LEONI, M. de *et al*. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. **Inf Syst**, [*s. l.*], v. 56, p. 235–257, 2016. DOI: 10.1016/j.is.2015.07.003.

13. MEDEIROS, A. K. A. de *et al.* Process mining based on clustering: A quest for precision. Lecture Notes. **Computer Science**, v. 4928 LNCS, p. 17–29, 2008. DOI: 10.1007/978-3-540-78238-4_4.

14. GARCIA, C. D. S. *et al.* Process mining techniques and applications – A systematic mapping study. **Expert Syst Appl**, [*s. l.*], v. 133, p. 260–295, 2019. DOI: 10.1016/j.eswa.2019.05.003.

15. FERRONATO, J. J. *et al.* PM4SOS: low-effort resource allocation optimization in a dynamic environment. IEEE International Conference on Systems, Man and Cybernetics, 2022, [*s. l.*]. **Conference Proceedings**. [*s. l.*]: Institute of Electrical and Electronics Engineers Inc, 2022. p. 1742–1747. DOI: 10.1109/SMC53654.2022.9945393.

16. MUNOZ-GAMA, J. **Conformance checking and diagnosis in process mining: comparing observed and modeled processes**. 2014. Tese (Doutorado) – Polytechnic University of Catalonia, Spain, 2014.

17. DUNZER, S. *et al.* Conformance checking: a state-of-the-art literature review. *In:* PROCEEDINGS OF THE 11TH INTERNATIONAL CONFERENCE ON SUBJECT-ORIENTED BUSINESS PROCESS MANAGEMENT. 2019, [*s. l.*]. **Proceedings** […]. [*s. l.*: s. n.]. p. 1-10, , 2019. DOI: 10.4108/eai.16-9-2019.2290226.

18. (ROZINAT, A.; VAN DER AALST, W. M. P. Conformance checking of processes based on monitoring real behavior. **Information Systems**, [*s. l.*], v. 33, n. 1, p. 64–95, 2008. DOI: 10.1016/j.is.2007.07.001.

19. VAN DER AALST, W.; WEIJTERS, T.; MARUSTER, L. Workflow Mining: Discovering Process Models from Event Logs. **IEEE Transactions on Knowledge and Data Engineering**, [*s. l.*], v. 16, n. 9, p. 1128–1142, 2004.

20. BUIJS, J. C. A. M.; VAN DONGEN, B. F.; VAN DER AALST, W. M. P. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. **International Journal of Cooperative Information Systems**, [*s. l.*], v. 23, n. 1, 2014. DOI: 10.1142/S0218843014400012.

21. SATO, D. M. V.; DALLAGASSA, M. R. Conformance checking with different levels of granularity: a case study on bariatric surgery. In: INTERNATIONAL CONGRESS ON IMAGE AND SIGNAL PROCESSING, 13, 2020, [*s. l.*]. **Proceedings** […]. [s. l.]: BioMedical Engineering and Informatics, 2020. p. 820-826.

22. MURATA, Tadao. Petri Nets: Properties, Analysis and Applications. **IEEE Proceedings of the IEEE**, [*s. l.*], v. 77, n. 4, p. 541–580, 1989.

23. LEONI, M. de; MAGGI, F. M.; VAN DER AALST, W. M. P. An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. **Information Systems**, [*s. l.*], v. 47, p. 258–277, 2015. DOI: 10.1016/j.is.2013.12.005.

24. MUÑOZ-GAMA, J.; CARMONA, J. A fresh look at precision in process conformance. **Lecture Notes in Computer Science**, [*s. l.*], v. 6336 LNCS, n. September 2010, p. 211–226, 2010. DOI: 10.1007/978-3-642-15618-2_16.

25. BLUM, F. R. Metrics in process discovery. **Technical Report TR/DCC**, [*s. l.*], n. 2015–6, p. 1–21, 2015.

26. SÁNCHEZ-GONZÁLEZ, L. *et al.* Prediction of business process model quality based on structural metrics. **Lecture Notes in Computer Science**, [*s. l.*], v. 6412 LNCS, p. 458–463, 2010. DOI: 10.1007/978-3-642-16373-9_35.

27. MACINTOSH, A.; WHYTE, A. Towards an evaluation framework for Process Mining Algorithms. **Transforming Government**: People, Process and Policy, [*s. l.*], v. 2, n. 1, p. 16–30, 2008. DOI: 10.1108/17506160810862928.

28. BOSE, R. P. J. C. *et al.* Handling concept drift in process mining. **Lecture Notes in Computer Science**, Berlin, p. 391–405, 2011. DOI: 10.1007/978-3-642-21640-4_30.

29. LU, J. *et al.* Learning under Concept Drift: A Review. **IEEE Transactions on Knowledge and Data Engineering**, [*s. l.*], v. 31, n. 12, p. 2346–2363, dec. 2019. DOI: 10.1109/TKDE.2018.2876857.

30. LU, J. *et al*. Learning under Concept Drift: A Review. **IEEE Transactions on Knowledge and Data Engineering**, [*s. l.*], v. 31, n. 12, p. 2346–2363, Dec. 2019. DOI: 10.1109/TKDE.2018.2876857.

31. SCHONENBERG, H. *et al.* Process flexibility: A survey of contemporary approaches. *In*: INTERNATIONAL WORKSHOP ON COOPERATION AND INTEROPERABILITY, ARCHITECTURE AND ONTOLOGY WORKSHOP ON ENTERPRISE AND ORGANIZATIONAL MODELING AND SIMULATION CIAO! 2008, EOMAS 2008, Montpellier. **Proceedings** […]. Montpellier: Springer, 2008. p. 16–30. DOI: 10.1007/978-3-540-68644-6_2.

32. BOSE, R. P. J. C. *et al*. Dealing with concept drifts in process mining. **IEEE Transactions on Neural Networks and Learning Systems**, [*s. l.*], v. 25, n. 1, p. 154–171, 2014. doi: 10.1109/TNNLS.2013.2278313.

33. OSTOVAR, A. *et al.* Detecting drift from event streams of unpredictable business processes. **Lecture Notes in Computer Science**, [*s. l.*], v. 9974 LNCS, p. 330–346, 2016. DOI: 10.1007/978-3-319-46397-1_26.

34. GUSTAFSSON, F. Adaptive Filtering and Change Detection. New York: Wiley, 2001. DOI: 10.1002/0470841613.

35. VAN ZELST, S. J.; VAN DONGEN, B. F.; VAN DER AALST, W. M. P. Event stream-based process discovery using abstract representations. **Knowledge and Information Systems**, [*s. l.*], v. 54, n. 2, p. 407–435, 2018. DOI: 10.1007/s10115-017-1060-2.

36. BUTTAZZO, G. C. **Hard Real-Time Computing Systems**. 3. ed. New York: Springer US, 2011. DOI: 10.1007/978-1-4614-0676-1.

37. GAMA, J. *et al.* Learning with Drift Detection. *In*: ADVANCES IN ARTIFICIAL INTELLIGENCE–SBIA 2004: BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 17, 2004, São Luís, Maranhão, Brasil, sept. 29-ocotber 1. **Proceedings 17**. Berlin, Heidelberg: Springer, 2004. p. 286-295.

38. BIFET, A. *et al.* Early Drift Detection Method. **Fourth international workshop on knowledge discovery from data streams**, [*s. l.*], p. 77-86, 2006.

39. PAGE, A. E. S. Continuous Inspection Schemes. **Biometrika**, [*s. l.*], v. 41, n. 1, p. 100–115, 1954.

40. BIFET, A.; GAVALDÀ, R. Learning from time-changing data with adaptive windowing. In: SIAM INTERNATIONAL CONFERENCE ON DATA MINING, 7, 2007, [*s. l.*]. Proceedings […]. [*s. l.*: *s. n.*], 2007. p. 443–448. DOI: 10.1137/1.9781611972771.42.

41. HOEFFDING, W. Probability inequalities for sums of bounded random variables. **Journal of the American Statistical Association**, [*s. l.*], v. 58, n. 301, p. 13–30, 1963.

42. SATO, D. M. V. et al. Interactive Process Drift Detection: A Framework for Visual Analysis of Process Drifts. **CEUR Workshop Proceedings**, [*s. l.*], p. 41–42, 2021.

43. SATO, D. **Concept Drift Detection in Process Models**. 2022. Thesis (Graduate) – Pontifícia Universidade Católica do Paraná, Paraná, 2022.

44. BORKIN, D. *et al*. Impact of Data Normalization on Classification Model Accuracy. **Research Papers Faculty of Materials Science and Technology Slovak University of Technology**, [*s. l.*], v. 27, n. 45, p. 79–84, Sep. 2019. DOI: 10.2478/rput-2019-0029.

45. SAMUEL, A. L. Some studies in machine learning using the game of checkers. II-Recent progress. **Annual Review in Automatic Programming**, [*s. l.*], v. 6, n. PART 1, p. 1–36, 1959. DOI: 10.1016/0066-4138(69)90004-4.

46. HENKE, M. *et al.* Aprendizagem de Máquina para Segurança em Redes de Computadores: Métodos e Aplicações. In: MINICURSOS DO XI SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 11, 2011, [s. l.]. **Anais** [...]. [*s. l.: s. n.*], 2011. p. 53–103. DOI: 10.5753/sbc.9559.1.2.

47. MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre Aprendizado de Máquina. In: **Sistemas inteligentes**: fundamentos e aplicações. [*s. l.: s. n.*], 2003. p. 89–114.

48. HAUPT, M. Research Institute for Advanced Computer Science. 2005.

49. BARANAUSKAS, J. A.; MONARD, M. M. Reviewing some Machine Learning Concepts and Methods. São Carlos (SP): Relatório Técnico do Instituto de Ciências Matemáticas e de Computação (ICMC), 2002. p. 60.

50. PEREIRA, F.; MITCHELL, T.; BOTVINICK, M. Machine learning classifiers and fMRI: a tutorial overview. Neuroimage, v. 45, n. 1 Suppl, p. S199–S209, 2009. doi: 10.1016/j.neuroimage.2008.11.007.

51. NOI, P. T.; KAPPAS, M. Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery. **Sensors (Basel)**, [*s. l.*], v. 18, n. 1, 2017. DOI: 10.3390/S18010018.

52. YANG, S. *et al.* IKNN: Informative K-nearest neighbor pattern classification. Lecture Notes. **Computer Science**, [*s. l.*], v. 4702 LNAI, p. 248–264, 2007. DOI: 10.1007/978-3-540-74976-9_25.

53. ALPAYDIN, Ethem. **Introduction to Machine Learning**: The MIT Press. Massachusetts: MIT, 2014.

54. FLOYD, F. J.; WIDAMAN, K. F. Factor Analysis in the Development and Refinement of Clinical Assessment Instruments. **Psychol Assess**, [s. l.], v. 7, n. 3, p. 286–299, 1995. DOI: 10.1037/1040-3590.7.3.286.

55. FRIEDMAN, J. H. **Greedy Function Approximation**: A Gradient Boosting Machine. 2001. **Annals of statistics**, [*s. l.*], p. 1189-1232, 2001.

56. CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. *In*: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, aug., 2016, [*s. l.*]. **Proceedings** […]. [s. l.]: Association for Computing Machinery, 2016. p. 785–794. DOI: 10.1145/2939672.2939785.

57. WANG, C. *et al.* Robust Propensity Score Computation Method based on Machine Learning with Label-corrupted. **arXiv preprint arXiv:1801.03132**, jan. 2018. Disponível em: http://arxiv.org/abs/1801.03132. Acesso em: 15 maio 2024.

58. WANG, C.; DENG, C.; WANG, S. Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. **Pattern Recognit Lett**, [*s. l.*], v. 136, p. 190–197, Aug. 2020. DOI: 10.1016/J.PATREC.2020.05.035.

59. IEEE COMMUNICATIONS SOCIETY; INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING; INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **On using eXtreme Gradient Boosting (XGBoost) Machine Learning algorithm for Home NetworkTraffic Classification**. Manchester, IEEE *Xplore* , 2019.

60. SHUAI, Yong; ZHENG, Yujie; HUANG, Hao. Hybrid Software Obsolescence Evaluation Model Based on PCA-SVM-GridSearchCV. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND SERVICE SCIENCE

(ICSESS), 9, 2018, [*s. l.*]. **Proceedings** […]. [*s. l.*]: ICSESS, 2018. p. 449-453. DOI: 10.1109/ICSESS.2018.8663753.

61. BORRA, S.; DI CIACCIO, A. Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods. **Comput Stat Data Anal**, [*s. l.*], v. 54, n. 12, p. 2976–2989, Dec. 2010. DOI: 10.1016/J.CSDA.2010.03.004.

62. BRZEZINSKI, D. *et al.* On the dynamics of classification measures for imbalanced and streaming data. **IEEE Trans Neural Netw Learn Syst**, [*s.* l.], v. 31, n. 8, p. 2868–2878, aug. 2020. DOI: 10.1109/TNNLS.2019.2899061.

63. ECK, M. *et al.* PM2: A Process Mining Project Methodology. *In*: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, 2015, Stockholm, Sweden. **Proceedings** […]. Sweden: Springer, 2015. p. 297–313. DOI: 10.1007/978-3-319-19069-3_19.

64. MANNHARDT, F.; DE LEONI, M.; REIJERS, H. A. Heuristic mining revamped: An interactive, data-Aware, and conformance-Aware miner. **CEUR Workshop Proc**, [*s. l.*], v. 1920, 2017.

65. BARBON JUNIOR, S. *et al*. A Framework for Human-in-the-loop Monitoring of Concept-drift Detection in Event Log Stream. In: THE WEB CONFERENCE 2018 - COMPANION OF THE WORLD WIDE WEB CONFERENCE, WWW 2018, [*s. l.*]. **Proceedings** […]. [*s. l.: s. n.*], 2018. v. 2, p. 319–326 DOI: 10.1145/3184558.3186343.

66. WAGNER, R. A.; FISCHER, M. J. The String-to-String Correction Problem. **Journal of the ACM (JACM)**, [*s. l.*], v. 21, n. 1, p. 168–173, 1974. DOI: 10.1145/321796.321811.

67. ADAMS, J. N. *et al.* A Framework for Explainable Concept Drift Detection in Process Mining. **Lecture Notes in Computer Science**, [*s. l.*], v. 12875 LNCS, p. 400–416, 2021. DOI: 10.1007/978-3-030-85469-0_25.

68. WAMBUI, G. D. The Power of the Pruned Exact Linear Time(PELT) Test in Multiple Changepoint Detection. **American Journal of Theoretical and Applied Statistics**, [*s. l.*], v. 4, n. 6, p. 581, 2015. DOI: 10.11648/J.AJTAS.20150406.30.

69. DENISOV, V.; ELENA, B.; D, F. BPIC'2018: Mining Concept Drift in Performance Spectra of Processes. *In*: INTERNATIONAL WORKSHOP ON BUSINESS PROCESS INTELLIGENCE 2018, 2018, [*s. l.*]. **Proceedings** […]. [*s. l.: s. n.*], 2018. DOI: 10.4121/UUID.

70. DENISOV, V.; FAHLAND, D.; VAN DER AALST, W. M. P. Unbiased, fine-grained description of processes performance from event data. **Lecture Notes in Computer Science**, [*s. l.*], v. 11080 LNCS, p. 139–157, 2018. DOI: 10.1007/978-3-319-98648-7_9.

71. YANG, H.; WEN, L.; WANG, J. An Approach to Evaluate the Local Completeness of an Event Log. In: IEEE INTERNATIONAL CONFERENCE ON DATA MINING,

12, 2012, Brussels, Belgium. **Proceedings** […]. Belgium: IEEE, 2012. p. 1164–1169. DOI: 10.1109/ICDM.2012.66.

72. RICHTER, F.; SEIDL, T. Looking into the TESSERACT: Time-drifts in event streams using series of evolving rolling averages of completion times. **Inf Syst**, [*s. l.*], v. 84, p. 265–282, 2019. DOI: 10.1016/J.IS.2018.11.003.

73. SCHUBERT, E.; WEILER, M.; KRIEGEL, H.-P. SigniTrend: Scalable Detection of Emerging Topics in Textual Streams by Hashed Significance Thresholds. *In*: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in KDD '14, 20 New York. **Proceedings** […]. New York, USA: Association for Computing Machinery, 2014.

74. AALST, W. M. P. van der et al. Prom: The process mining toolkit. *In*: BPM 2009 Demonstration Track, Ulm, 2019, Germany. **Proceedings** […]. Germany: CEUR-WS.org, 2009. p. 1–4.

75. DONGEN, B. F. van. **Bpi challenge 2017 - offer log**. [*s. l.*]: Bpi challenge 2017.

76. LEONI, F.; MASSIMILIANO, M. de. Road Traffic Fine Management Process. Holanda: Eindhoven University of Technology, 2015. Available at: http://dx.doi.org/%0A10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5. Accessed: Mar. 11, 2022.

77. ACCORSI, R.; STOCKER, T. Discovering workflow changes with time-based trace clustering. Lecture Notes. **Business Information Processing**, [*s. l.*], v. 116 LNBIP, p. 154–168, 2012. DOI: 10.1007/978-3-642-34044-4_9.

78. BROCKHOFF, T.; UYSAL, M. S.; AALST, W. M. P. van der. Time-aware Concept Drift Detection Using the Earth Mover's Distance. In: 2nd INTERNATIONAL CONFERENCE ON PROCESS MINING (ICPM), 2, 2020, Padua, Italy. **Proceedings** […]. Italy: IEEE, Oct. 2020. p. 33–40. DOI: 10.1109/ICPM49681.2020.00016.

79. CARMONA, J.; GAVALDÀ, R. Online Techniques for Dealing with Concept Drift. **Process Mining**, Berlin, Heidelberg, p. 90–102, 2012. DOI: 10.1007/978-3-642-34156-4_10.

80. HASSANI, M. Concept drift detection of event streams using an adaptive window. In: EUROPEAN COUNCIL FOR MODELLING AND SIMULATION, ECMS, 2019, Napoli, Italy. **Proceedings** […]. Italy: European Council for Modelling and Simulation, 2019. p. 230–239. DOI: 10.7148/2019-0230.

81. HOMPES, B. *et al*. Detecting Changes in Process Behavior Using Comparative Case Clustering. *In*: DATA-DRIVEN PROCESS DISCOVERY AND ANALYSIS (SIMPDA 2015), Vienna, Austria. **Proceedings** […]. Austria: Springer, 2017. p. 0–22. DOI: 10.1007/978-3-319-53435-0.

82. IMPEDOVO, A. *et al.* Simultaneous Process Drift Detection and Characterization with Pattern-Based Change Detectors. *In*: INTERNATIONAL CONFERENCE ON

DISCOVERY SCIENCE (DS2020), 2020, Thessaloniki, Greece. **Proceedings** […]. Greece: Springer, 2020. p. 451–467. DOI: 10.1007/978-3-030-61527-7_30.

83. KURNIATI, A. P. *et al.* Using a multi-level process comparison for process change analysis in cancer pathways. **Int J Environ Res Public Health**, [*s. l.*], v. 17, n. 19, p. 1–16, 2020. DOI: 10.3390/ijerph17197210.

84. LIU, N.; HUANG, J.; CUI, L. A Framework for Online Process Concept Drift Detection from Event Streams. In: 2018 IEEE International Conference on Services Computing (SCC), 2018, San Francisco, CA, USA. **Proceedings** […]. San Francisco: IEEE, 2018. p. 105–112. DOI: 10.1109/SCC.2018.00021.

85. LUENGO, D.; SEPÚLVEDA, M. Applying Clustering in Process Mining to Find Different Versions of a Business Process that Changes over Time. *In*: DANIEL, F.; BARKAOUI, K.; DUSTDAR, S. (eds.), **Business Process Management Workshops.** Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 153–158. DOI: 10.1007/978-3-642-28108-2_15.

86. MAARADJI, A.; DUMAS, M.; ROSA, M. L.; OSTOVAR, A. Detecting Sudden and Gradual Drifts in Business Processes from Execution Traces**. IEEE Trans Knowl Data Eng**, [*s. l.*], v. 29, n. 10, p. 2140–2154, 2017. DOI: 10.1109/TKDE.2017.2720601.

87. MANOJ KUMAR, M. V.; THOMAS, L.; ANNAPPA, B. Capturing the sudden concept drift in process mining. **CEUR Workshop Proc**, [*s. l.*], v. 1371, n. January, p. 132–143, 2015.

88. MARTJUSHEV, J.; BOSE, R. P. J. C.; VAN DER AALST, W. M. P. Change Point Detection and Dealing with Gradual and Multi-Order Dynamics in Process Mining. *In*: INTERNATIONAL CONFERENCE ON BUSINESS INFORMATICS RESEARCH, 2015, [*s. l.*]. **Proceedings** […]. [s. l.: s. n.], 2015. p. 1–15. DOI: 10.1007/978-3-319-21915-8_11.

89. MORA, D. *et al.* The CDESF toolkit: An introduction. *In*: ICPM DOCTORAL CONSORTIUM AND TOOL DEMONSTRATION TRACK 2020, 2020, Padua, Italy. **Proceedings** […]. Italy: CEUR-WS.org, 2020. p. 47–50.

90. OSTOVAR, A.; LEEMANS, S. J. J.; LA ROSA, M. Robust Drift Characterization from Event Streams of Business Processes. **ACM Trans Knowl Discov Data**, [*s. l.*], v. 14, n. 3, p. 1–57, may 2020. DOI: 10.1145/3375398.

91. OSTOVAR, A. *et al.* Characterizing Drift from Event Streams of Business Processes. *In*: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING CAISE 2017: Advanced Information Systems Engineering, Essen, Germany, 2017. **Proceedings** […]. Germany: Springer, 2017. p. 210–228. DOI: 10.1007/978-3-319-59536-8_14.

92. PAUWELS, S.; CALDERS, T. An anomaly detection technique for business processes based on extended dynamic Bayesian networks. **Proceedings of the ACM**

Symposium on Applied Computing, [*s. l.*], Part F1477, no. april, p. 494–501, 2019. DOI: 10.1145/3297280.3297326.

93. SEELIGER, A.; NOLLE, T.; MÜHLHÄUSER, M. Detecting concept drift in processes using graph metrics on process graphs. *In*: CONFERENCE ON SUBJECT-ORIENTED BUSINESS PROCESS MANAGEMENT, S-BPM ONE '17, 9, 2017. **Proceedings** […]. Darmstadt: [*s. n*.], 2017. doi: 10.1145/3040565.3040566.

94. STERTZ, F.; RINDERLE-MA, S. Process Histories - Detecting and Representing Concept Drifts Based on Event Streams. *In*: On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, 2018, Valletta, Malta. **Proceedings** […]. Valletta, Malta: Springer, 2018. p. 318–335. DOI: 10.1007/978-3-030-02610-3_18.

95. STERTZ, F.; RINDERLE-MA, S. Detecting and Identifying Data Drifts in Process Event Streams Based on Process Histories. *In*: INFORMATION SYSTEMS ENGINEERING IN RESPONSIBLE INFORMATION SYSTEMS. CAiSE 2019, 2019, Rome, Italy. **Proceedings** […]. Italy: Springer International Publishing, 2019. p. 133–144. DOI: 10.1007/978-3-030-21297-1.

96. TAVARES, G. M. *et al*. Overlapping analytic stages in online process mining. In: IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING (SCC 2019), 2019, Milan, Italy. **Proceedings** […]. Italy: IEEE, 2019. p. 167–175. DOI: 10.1109/SCC.2019.00037.

97. YESHCHENKO, A. *et al.* Comprehensive Process Drift Analysis with the Visual Drift Detection Tool. **CEUR Workshop Proceedings**, Salvador, Brazil, CEUR-WS, p. 108-112, 2019..

98. YESHCHENKO, A. *et al*., Comprehensive Process Drift Detection with Visual Analytics. In: CONCEPTUAL MODELING. ER 2019, 2019, Salvador. **Lecture Notes in Computer Science**. Salvador, Brazil: Springer, 2019. p. 119–135. DOI: https://doi.org/10.1007/978-3-030-33223-5_11.

99. YESHCHENKO, A. *et al.* Visual Drift Detection for Sequence Data Analysis of Business Processes. **IEEE Trans Vis Comput Graph**, [*s. l.*], p. 1–1, 2021. DOI: 10.1109/TVCG.2021.3050071.

100. YESHCHENKO, A. *et al*. VDD: A Visual Drift Detection System for Process Mining. In: ICPM 2020 DOCTORAL CONSORTIUM AND TOOL DEMONSTRATION TRACK, 2020, Padua, Italy. **Proceedings** […]. Italy: CEUR-WS.org, 2020. p. 31–34. DOI: 10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

101. ZELLNER, L. *et al*. Concept Drift Detection on Streaming Data with Dynamic Outlier Aggregation. In: PROCESS MINING WORKSHOPS - INTERNATIONAL WORKSHOP ON STREAMING ANALYTICS FOR PROCESS MINING (SA4PM'20), 1, 2020, Padua, Italy. **Proceedings** […]. Italy: Springer, 2020. p. 189–192.

102. ZHENG, C. *et al.* Detecting Process Concept Drifts from Event Logs. *In*: ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS. OTM 2017, 2017, Cham. **Proceedings** […]. Cham: Springer International Publishing, 2017. p. 524–542. DOI: 10.1007/978-3-319-69462-7.

103. GEYER-KLINGEBERG, J. The Proactive Insights Engine: Process Mining meets Machine Learning and Artificial Intelligence. Barcelona, Spain: 15 th International Conference on Business Process Management, 2017. Disponível em: https://www.researchgate.net/publication/319550867. Acesso em: 15 maio 2024.

104. SUÁREZ-CETRULO, A. L.; QUINTANA, D.; CERVANTES, A. A survey on machine learning for recurring concept drifting data streams. **Expert Systems with Applications**, [*s. l.*], v. 213, [s. l.], Mar. 01, 2023. doi: 10.1016/j.eswa.2022.118934.

ANEXO A – DICIONÁRIO DE DADOS DA BASE DE ATRIBUTOS

Nome Variável/coluna	Descrição	Tipo de Variável	Valores Permitidos	Anotações
log	Nome do arquivo de *log.*	Caracteres	Alfanuméricos.	Não usar caracteres especiais.
nr_eventos	Total de eventos	Numérico	Não informar números negativo.	Não permitir campo vazio.
nr_casos	Total de casos	Numérico	Não informar números negativo.	Não permitir campo vazio.
nr_atividades	Total de atividades	Numérico	Não informar números negativo.	Não permitir campo vazio.
media_tempo_casos	Média dos tempos de duração dos casos	Numérico	Não informar números negativo.	Não permitir campo vazio.
mediana_tempo_casos	Mediana dos tempos de duração dos casos	Numérico	Não informar números negativo.	Não permitir campo vazio.
nr_variantes	Número de variantes do processo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
media_freq_ativ	Média das frequências das atividades	Numérico	Não informar números negativo	Não permitir campo vazio.
mediana_freq_rec	Média das frequências dos recursos.	Numérico	Não informar números negativo.	Não permitir campo vazio.
tipo_abordagem	Tipo de abordagem	Caracteres	<trace by trace> <windowing>	Não permitir campo vazio.
perspectiva_mudanca	perspectiva de mudança	Caracteres	<Control_Flow>	Não permitir campo vazio.
Detector	Nome do detector	Caracteres	<ADWIN> <HDDM_W>	Não permitir campo vazio.
nome_parametro	Valor do parâmetro	Caracteres	<delta>	-
Delta	Valor do Parâmetro Delta	Numérico	0.0 a 9.9	Permitir apenas valores no intervalor informado.
fitness_log_compelto	Valor da métrica “Fitness” para o *log* completo.	Numérico	0.0 a 1.0	Permitir apenas valores no intervalor informado.
precisao_log_compelto	Valor da métrica “Precisão” para o *log* completo.	Numérico	0.0 a 1.0	Permitir apenas valores no intervalor informado
generalizacao_log_completo	Valor da métrica “Generalização” para o *log* completo.	Numérico	0.0 a 1.0	Permitir apenas valores no intervalor informado
simplicidade_log_completo	Valor da métrica “Simplicidade” para o *log* completo.	Numérico	0.0 a 1.0	Permitir apenas valores no

				intervalor informado
f1score_IPDD	Valor da Métrica F1Score gerada pelo IPDD quando utiliza logs sintéticos com marcações de mudança de conceito.	Numérico	Não informar números negativos.	Campo só existe para dados sintéticos.
log_completo_count_ativ_trace	Número de atividades no traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_mean_ativ_trace	Mediana de tempo de atividades por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_std_ativ_trace	Desvio padrão do tempo de atividades por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_min_ativ_trace	Valor mínimo de tempo de atividade por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_25%_ativ_trace	25% dos dados estão abaixo do primeiro quartil do tempo de atividade por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_50%_ativ_trace	Mediana de tempo de atividade por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_75%_ativ_trace	75% dos dados estão abaixo do terceiro quartil do tempo de atividade por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
log_completo_max_ativ_trace	Valor máximo de tempo de atividade por traço para o *log* completo.	Numérico	Não informar números negativo.	Não permitir campo vazio.
mediana_sublog_fitness	Mediana do total das métrica fitness para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
mediana_sublog_precisao	Mediana do total das métrica precisão o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo	Não permitir campo vazio.
mediana_sublog_generalizacao	Mediana do total das métrica generalização para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
mediana_sublog_simplicidade	Mediana do total da métrica simplicidade para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
media_sublog_fitness	Média do total da métrica fitness para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
media_sublog_precisao	Média do total da métrica precisão para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
media_sublog_generalizacao	Média do total da métrica generalização para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.

media_sublog_simplicidade	Média do total da métrica simplicidade para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
min_sublog_fitness	Valor mínimo do total da métrica fitness para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
min_sublog_precisao	Valor mínimo do total da métrica precisão para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
min_sublog_generalizacao	Valor mínimo do total da métrica generalização para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
min_sublog_simplicidade	Valor mínimo do total da métrica simplicidade para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
max_sublog_fitness	Valor máximo do total da métrica fitness para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
max_sublog_precisao	Valor máximo do total da métrica precisão para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
max_sublog_generalizacao	Valor máximo do total da métrica generalização para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo	Não permitir campo vazio.
max_sublog_simplicidade	Valor máximo do total da métrica simplicidade para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
std_sublog_fitness	Valor do desvio padrão calculado para o total da métrica fitness para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
std_sublog_precisao	Valor do desvio padrão calculado para o total da métrica precisão para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
std_sublog_generalizacao	Valor do desvio padrão calculado para o total da métrica generalização para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
std_sublog_simplicidade	Valor do desvio padrão calculado para o total da métrica simplicidade para o todos os *sublogs* gerados em cada ponto de mudança localizado.	Numérico	Não informar números negativo.	Não permitir campo vazio.
tamanho_janela	Parâmetro utilizado pela ferramenta de detecção que se refere ao número de traços iniciais, frequentemente referido como o tamanho da janela.	Numérico	Não informar números negativo.	Não permitir campo vazio.

fator	O fator de ajuste refere-se a escala de valores.	Numérico	<10> <100>	Não informar números negativo.
drift_confidence	Parâmetro interno utilizado pelo algoritmo detector HDDM_W - refere-se a confiança da mudança. Valor padrão sugerido: 0.001.	Numérico	Não informar números negativo	Não permitir campo vazio.
warning_confidence	Parâmetro interno utilizado pelo algoritmo detector HDDM_W - refere-se aviso de confiança da mudança. Valor padrão sugerido: 0.005.	Numérico	Não informar números negativo	Não permitir campo vazio.
lambda_val	Parâmetro interno utilizado pelo algoritmo detector HDDM_W - refere-se ao peso dado aos dados recentes. Valores menores significam menos peso dado aos dados recentes. Valor padrão sugerido: 0.05	Numérico	Não informar números negativo	Não permitir campo vazio.
two_sided_test	Parâmetro interno utilizado pelo algoritmo detector HDDM_W - Se “True”, monitorará incrementos e decrementos de erros (frente e verso). Por padrão, monitorará apenas incrementos (unilaterais). Valor padrão sugerido: “False”.	Lógico	<True> < False>	-