Seleção adaptativa de hiperparâmetros com aprendizado por reforço para modelos de detecção de anomalias em séries temporais multivariadas

Rodrigo Marcel Araujo Oliveira
Escola Politécnica
Universidade Federal da Bahia
Salvador, Bahia
email:rodrigomarcel@ufba.br

Ângelo Márcio Oliveira Sant'Anna
Escola Politécnica
Universidade Federal da Bahia
Salvador, Bahia
email: angelo.santanna@ufba.br

Paulo Henrique Ferreira da Silva
Instituto de Matemática e Estatística
Universidade Federal da Bahia
Salvador, Bahia
email: paulohenri@ufba.br

***Resumo*—A seleção de hiperparâmetros em modelos de aprendizado de máquina é um fator crítico para assegurar a eficácia de sua performance. No contexto de modelos não supervisionados, esse desafio se torna ainda mais complexo, especialmente na tarefa de detecção de anomalias, dado que o processo de treinamento ocorre sem orientação explícita de classes. Este trabalho propõe uma abordagem inteligente para a seleção automática de hiperparâmetros de modelos não supervisionados de detecção de anomalias em séries temporais multivariadas, utilizando o algoritmo *Soft Actor-Critic*, baseado em aprendizado por reforço. Foram estudados os modelos: *Isolation Forest*, *Principal Component Analysis*, *Local Outlier Factor*, *Lightweight on-line detector of anomalies*. O estudo de caso explora dados de séries temporais oriundos do monitoramento de poços de produção de petróleo. Os resultados obtidos são promissores e evidenciam a robustez da metodologia proposta em potencializar a performance de modelos para a tarefa de detecção de anomalias.**

Palavras-chave—aprendizado não supervisionado, aprendizado por reforço, detecção de anomalias, otimização de hiperparâmetros, séries temporais.

I. Introdução

A seleção de hiperparâmetros é um dos pilares para garantir um bom desempenho de algoritmos de aprendizado de máquina. A escolha inadequada desses parâmetros pode comprometer a eficácia de modelos sofisticados, enquanto uma configuração otimizada pode melhorar a performance e a capacidade de generalização desses modelos. No contexto da Indústria 4.0, com a evolução das arquiteturas de modelos e o crescimento dos dados disponíveis, a tarefa de ajuste de hiperparâmetros se tornou não apenas mais crítica, mas também mais custosa em termos de tempo e recursos computacionais. Para isso, soluções automatizadas precisam ser robustas, escaláveis e eficientes computacionalmente, preservando um bom desempenho na tarefa de aprendizado [1].

Tradicionalmente, o ajuste de hiperparâmetros tem sido realizado por meio de abordagens como a busca em grade, aleatória ou mesmo por tentativa e erro orientada por especialistas. Apesar de simples, essas estratégias raramente são eficientes em domínios de alta dimensionalidade, uma vez que demandam múltiplas execuções do modelo para encontrar boas configurações. A complexidade não reside apenas na dimensionalidade, mas principalmente no alto custo de avaliação de cada configuração. Como alternativa, métodos mais recentes e eficazes, como o *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [2] e *Tree-structured Parzen Estimator* (TPE) [3], têm ganho destaque na tarefa de seleção de hiperparâmetros. O CMA-ES é um algoritmo evolucionário baseado em amostragem adaptativa de distribuições multivariadas, que atualiza iterativamente a média, a matriz de covariância e o passo de mutação com base no desempenho das soluções candidatas. O TPE é um método Bayesiano que modela a função objetivo como duas distribuições probabilísticas separadas: uma para os valores de hiperparâmetros associados a perdas de valores baixos e outra para as demais. A otimização ocorre ao maximizar a razão entre essas distribuições, priorizando regiões promissoras do espaço de busca.

A otimização de hiperparâmetros também pode ser compreendida como um processo de natureza dinâmica e sequencial [4]. O objetivo é otimizar uma métrica de desempenho, calculada sobre um conjunto de teste. Essa visão permite reestruturar o problema como um processo de decisão, no qual o histórico de iterações com o modelo pode ser usado para guiar a exploração do espaço de parâmetros em busca de soluções cada vez melhores. Nessa perspectiva, o ajuste de hiperparâmetros pode ser modelado como um ambiente de aprendizado por reforço, do inglês *Reinforcement Learning* (RL), no qual um agente é encarregado de aprender políticas para selecionar configurações promissoras [5]. Assim, o estado é composto por um histórico de avaliações, enquanto as ações consistem em modificações nos hiperparâmetros. A função de recompensa reflete diretamente o resultado obtido, como uma melhoria de uma métrica, guiando o aprendizado da política ao longo de múltiplos episódios, acelerando a convergência para um ótimo global.

Este artigo propõe uma arquitetura baseada em RL utilizando o algoritmo *Soft Actor-Critic* (SAC) para a seleção automática de hiperparâmetros, especificamente para modelos não supervisionados de detecção de anomalias em séries temporais multivariadas. A abordagem considera uma topologia de modelo fixa, com a formulação do problema como um

processo de decisão sequencial, no qual são definidos estados informativos, um conjunto de ações e uma função de transição que permite ao agente explorar a superfície de desempenho do modelo de forma estratégica e eficiente. O objetivo é desenvolver um sistema inteligente que reduza o custo de experimentação e potencialize os resultados de algoritmos de aprendizado de máquina, mesmo em cenários desafiadores como a detecção de anomalias sem supervisão.

Na Seção II são discutidos trabalhos relacionados com a pesquisa. Na Seção III são apresentados os referenciais teóricos utilizados no artigo, abrangendo o algoritmo SAC baseado em RL, abordagem para otimização de hiperparâmetros, modelos não supervisionados para detecção de anomalias tais como *Isolation Forest* (iForest), *Principal Component Analysis* (PCA), *Local Outlier Factor* (LOF), *Lightweight on-line detector of anomalies* (LODA), além das métricas de avaliação dos modelos. A Seção IV trata do estudo de caso, com dados de alta dimensionalidade no contexto de séries temporais. Os resultados do processo de seleção de hiperparâmetros com aprendizado por reforço para os modelos de detecção de anomalias estão expostos na Seção V. Na Seção VI são delineadas as conclusões do estudo, bem como as possibilidades para pesquisas futuras.

II. Revisão da literatura

[4] propuseram uma primeira abordagem de ajuste de hiperparâmetros como um problema de RL, em que um agente aprende a explorar o espaço de configurações com base em experiências anteriores, maximizando recompensas futuras. Os autores introduzem o *Hyp*-RL, uma política baseada em *Q-learning* capaz de lidar com espaços de alta dimensão. A representação do estado combina características estáticas dos dados com uma sequência dinâmica de configurações e recompensas, modelada como uma série temporal multivariada. Para capturar dependências de longo prazo, são utilizadas células de *Long Short-Term Memory* (LSTM). As avaliações em 50 conjuntos de dados da *University of California, Irvine* (UCI) mostram que o método é escalável e supera abordagens tradicionais.

Em [6] e [7], os autores propuseram uma abordagem para acelerar o processo de busca de hiperparâmetros por RL. Para isso, foi utilizado um modelo preditivo auxiliar para estimar a performance do algoritmo avaliado, evitando treinamentos custosos. O controle é realizado dinamicamente ao limitar a distância entre políticas antes e depois do uso do modelo. Essa técnica permite ajustar adaptativamente o horizonte de uso do modelo, equilibrando eficiência e precisão. Os testes com *eXtreme Gradient Boosting* (XGBoost) e redes neurais convolucionais em 101 tarefas reais mostraram que o método proposto alcança a maior acurácia em 86,1% dos casos, superando outros métodos do estado da arte em precisão e velocidade.

O estudo desenvolvido em [8] propôs um novo método que combina o algoritmo SAC com regularização hierárquica de mistura para otimizar hiperparâmetros. Os autores propuseram um modelo LSTM que atua como ambiente, recebendo ações de uma rede *actor Multilayer Perceptron* (MLP) e retornando o próximo estado, enquanto uma rede *critic* calcula a recompensa esperada, visando maximizá-la com base no desempenho. Para acelerar a convergência, foi empregada a regularização hierárquica de mistura, melhorando a eficiência na obtenção dos melhores hiperparâmetros.

Uma abordagem de detecção ativa de anomalias foi proposta por [9]; este artigo utiliza RL profundo para otimizar explicitamente a descoberta de anomalias ao longo do tempo, diferentemente de métodos tradicionais que apenas reordenam instâncias com base na resposta de humano imediato. A política é treinada com o algoritmo *Proximal Policy Optimization* (PPO), utilizando meta-características extraídas dos fluxos de dados. Essa abordagem consegue selecionar de forma mais estratégica quais instâncias devem ser apresentadas a um analista humano em cada iteração, reduzindo falsos positivos e descobrindo mais anomalias dentro de um orçamento de tempo. O artigo [10] apresenta abordagens para detecção de anomalias em séries temporais baseadas em políticas de RL, voltado para dados em fluxo como os gerados pela Internet das Coisas (do inglês *Internet of Things*, IoT). Em [11], os autores propõem uma abordagem dinâmica para detecção de anomalias em veículos conectados e autônomos, com o objetivo de garantir uma operação segura e robusta em tempo real. A solução combina um classificador de anomalias baseado em redes neurais com o algoritmo de RL *Asynchronous Advantage Actor-Critic* (A3C).

Neste trabalho é apresentada uma abordagem inédita para a otimização de hiperparâmetros usando o algoritmo SAC para modelos não supervisionados no contexto de detecção de anomalias. Embora estudos anteriores tenham abordado o tema, este estudo é o primeiro a lidar com dados de séries temporais e a propor um protótipo de validação cruzada temporal como ambiente de RL.

III. Metodologia

Nesta seção são discutidos os referenciais teóricos das técnicas utilizadas para o desenvolvimento deste artigo. Os métodos foram implementados em uma máquina equipada com processador *Intel Core i7-7500U* (7ª geração, 2,70 GHz), 32 GB de memória RAM e placa de vídeo *NVIDIA GeForce* com 2 GB de memória dedicada. A linguagem de programação adotada foi Python versão 3.9, com auxílio de ferramentas de *software* livre como Jupyter Notebook. A biblioteca `stable-baselines3` foi utilizada como base para a implementação da proposta do ambiente de RL, `scikit-learn` e `pyod` para os modelos de detecção de anomalias. Os métodos TPE e CMA-ES foram baseados no `optuna`.

A. Aprendizado por reforço

O RL é um tipo de aprendizado de máquina em que um agente inteligente aprende a tomar decisões através de tentativa e erro, interagindo com um ambiente [12]. O agente recebe recompensas ou punições conforme suas ações, buscando maximizar o retorno acumulado ao longo do tempo.

Geralmente, os problemas de RL são modelados por Processos de Decisão de Markov (MDP, do inglês *Markov Decision Process*), definidos pelo conjunto $\mathcal{X} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, em que: $\mathcal{S}$ é o espaço de estados possíveis no ambiente; $\mathcal{A}$ é o espaço de ações que o agente pode tomar; $P(s'|s,a)$ representa a função de transição de estados, que define a probabilidade de ir para o estado s' dado que o agente está no estado s e executa a ação a; $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ é a função de recompensa, denotada por $r(s,a)$, que retorna um escalar depois de executar a ação a no estado s; $\gamma \in (0,1)$ é o fator de desconto, que controla a importância de recompensas futuras em relação às imediatas.

Um dos componentes centrais do RL é a função valor de ação, denotada por $Q(s,a)$, cuja definição é apresentada na Equação (1). Essa função representa o valor esperado do retorno total descontado quando o agente começa no estado s, executa a ação a e, em seguida, segue uma política π.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t,a_t)\right) \Big| s_0 = s, a_0 = a\right]. \quad (1)$$

O SAC é um algoritmo de RL que combina elementos de políticas estocásticas com maximização da entropia, visando melhorar a exploração e a robustez [13]. O objetivo não é apenas maximizar a recompensa acumulada, mas também a entropia da política. Isso significa que o agente é incentivado a explorar diferentes ações enquanto ainda aprende a tarefa principal. A maximização da entropia ajuda a evitar convergência prematura para subótimas e melhora a adaptabilidade em ambientes complexos. O SAC demonstra desempenho eficaz em ambientes onde a exploração é crítica.

O SAC permite balancear *exploration* e *exploitation*, diferente das abordagens tradicionais como *Deep Q-Network* (DQN) e PPO. O *exploration* refere-se à busca por novas ações ou caminhos que o agente pode seguir para descobrir novas informações e melhorar seu entendimento do ambiente [14]. O *exploitation* está relacionado à utilização do conhecimento atual do agente para maximizar as recompensas, escolhendo as ações que já se sabe que são benéficas. O objetivo é encontrar uma política $\pi(a|s)$ que maximize o retorno esperado ajustado pela entropia, conforme a Equação (2):

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t} \gamma^t \Big(r(s_t,a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\Big)\right], \quad (2)$$

em que π é a política estocástica do agente, $\gamma \in (0,1)$ é o fator de desconto, $r(s_t,a_t)$ é a recompensa imediata no estado s_t após a ação a_t, $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_{a\sim\pi(\cdot|s)}[\log \pi(a|s_t)]$ é a entropia da política, $\alpha > 0$ é um coeficiente de temperatura que regula o *trade-off* entre *exploration* e *exploitation*.

O SAC é um método *off-policy*, o que permite o reuso eficiente de dados de experiências anteriores armazenados em um *replay buffer*. O método utiliza duas funções críticas: a função valor, $V(s)$, que corresponde ao valor esperado de um estado considerando a política atual e a entropia; a função Q, denotada por $Q(s,a)$, que consiste no valor esperado de um par estado-ação, incluindo a recompensa futura e a entropia. A relação entre V e Q está representada pela Equação (3). A função Q é atualizada minimizando o erro de Bellman, conforme a Equação (4), em que $\mathcal{D}$ é um *replay buffer* de experiências armazenadas.

$$V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q(s,a) - \alpha \log \pi(a|s)\right]. \quad (3)$$

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\Big(Q(s,a) - \big(r + \gamma(V(s'))\big)\Big)^2\right]. \quad (4)$$

O SAC emprega uma estrutura chamada *actor-critic*, para reduzir o *superestimation bias*; além disso, utiliza duas *Q-networks* independentes ($Q_{\theta_1}, Q_{\theta_2}$), que são treinadas simultaneamente. O valor alvo é calculado usando o mínimo entre elas, conforme a Equação (5), em que $\tilde{a}' \sim \pi(\cdot|s')$.

$$Q_{\text{target}}(s,a) = r + \gamma\Big(\min_{i=1,2} Q_{\theta_i}(s',\tilde{a}') - \alpha \log \pi(\tilde{a}'|s')\Big). \quad (5)$$

A política π_ϕ é atualizada para maximizar a Equação (6). Como a política é estocástica e, geralmente, uma Gaussiana multivariada, a reparametrização (*reparameterization trick*) é usada para viabilizar o gradiente, conforme a Equação (7). A política π_ϕ no SAC é atualizada implicitamente para minimizar a divergência de Kullback-Leibler (KL) entre a política atual e uma distribuição-alvo exponencial derivada da função Q.

$$\mathcal{L}_\pi = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\phi}\left[\alpha \log \pi_\phi(a|s) - Q_\theta(s,a)\right], \quad (6)$$

$$a = f_\phi(s,\epsilon), \quad \epsilon \sim \mathcal{N}(0,1). \quad (7)$$

O ajuste do coeficiente de entropia (α) é adaptado dinamicamente para manter um nível desejado de entropia $\bar{\mathcal{H}}$, conforme a Equação (8), em que λ é uma taxa de aprendizado.

$$\alpha \leftarrow \alpha - \lambda\left(\mathbb{E}_{a\sim\pi}[-\log \pi(a|s)] - \bar{\mathcal{H}}\right). \quad (8)$$

B. Otimização de hiperparâmetros baseada em MDP

A otimização de hiperparâmetros pode ser modelada como um MDP. Considere Ω o espaço de todas as possíveis configurações de hiperparâmetros, e $f : \Omega \to \mathcal{R}$ uma função de resposta associada ao desempenho do modelo, em que $f(\cdot)$ pode representar qualquer métrica de avaliação, como *accuracy*, *F_1-score* ou uma função de perda sobre o conjunto de validação. A função de resposta pode ser escrita conforme a Equação (9), em que M representa um modelo parametrizado por uma configuração de hiperparâmetros $\boldsymbol{\omega} \in \Omega$, o qual é treinado sobre um conjunto de treino $\mathcal{D}_{\text{train}}$ e validado sobre $\mathcal{D}_{\text{valid}}$, com $\mathcal{L}_M$ representando a função de perda ou métrica avaliada.

$$f(\boldsymbol{\omega}) = \mathcal{L}_M(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}} \mid \boldsymbol{\omega}). \quad (9)$$

O algoritmo SAC segue uma abordagem baseada em entropia máxima para RL profundo e contínuo, que busca aprender uma política estocástica que maximiza a recompensa esperada ao mesmo tempo em que promove a diversidade de ações, encorajando a exploração do espaço de ações. Isso é especialmente útil na otimização de hiperparâmetros, em que a exploração de diferentes combinações pode levar a melhores resultados. Nesse caso, o espaço de ações pode ser definido como o próprio espaço de hiperparâmetros, $\mathcal{A} = \Omega$. O estado

do ambiente, por sua vez, é composto por um histórico das avaliações já realizadas, conforme a Equação (10), com $n \in \mathbb{N}$.

$$\mathcal{S} = \{[(\boldsymbol{\omega}_1, r_1), \ldots, (\boldsymbol{\omega}_n, r_n)] \mid \ \boldsymbol{\omega}_i \in \Omega,\ r_i \in \mathcal{R}\}\,. \quad (10)$$

A recompensa do agente pode ser escrita por uma função que dependa do valor de desempenho observado após treinar o modelo com determinada configuração, ou seja, $\mathcal{R}(\cdot) = g(f(\cdot))$, em que $g(\cdot)$ pode representar uma transformação da métrica de desempenho de $f(\cdot)$. A cada passo t, o agente propõe uma nova configuração de hiperparâmetros $\boldsymbol{\omega}_t \sim \pi_\phi(\cdot|\boldsymbol{s}_t)$, em que π_ϕ é a política estocástica parametrizada. Após a avaliação do modelo com $\boldsymbol{\omega}_t$, o ambiente retorna uma recompensa r_t e atualiza seu estado com a nova tupla $(\boldsymbol{\omega}_t, r_t)$, conforme a Equação (11); este processo é repetido até que uma condição de parada seja atingida. Os critérios de parada consistem em considerar um limite máximo de iterações, ou quando o agente repete uma configuração de hiperparâmetros duas vezes consecutivas, forçando a política a explorar novas regiões do espaço. O estado inicial é definido por $\boldsymbol{s}_0 = \emptyset$.

$$\boldsymbol{s}_{t+1} = \boldsymbol{s}_t \cup (\boldsymbol{\omega}_t, r_t). \quad (11)$$

O processo proposto de validação cruzada temporal, detalhado no Algoritmo 1, quantifica a recompensa atribuída a cada configuração específica. Esse algoritmo avalia um modelo Φ com hiperparâmetros $\boldsymbol{\omega}$, usando K partições dos dados, respeitando a ordenação temporal. Para cada partição, o modelo é treinado sem rótulos de eventos anômalos, aplicado sobre os dados de teste com rótulos de ambas as classes e avaliado com uma métrica $\mathcal{G}$. Esse resultado é acumulado para obter a média final e o respectivo desvio-padrão. O valor retornado pelo algoritmo é a diferença entre esses valores.

O processo completo de otimização de hiperparâmetros utilizando o SAC está descrito no Algoritmo 2. A cada episódio, o agente interage com o ambiente propondo novas configurações de hiperparâmetros, avaliando o desempenho do modelo e atualizando sua política por meio do gradiente das funções de valor, de política e da entropia. Os parâmetros de entrada utilizados são: o conjunto de dados $\mathcal{D}$; o espaço de ações $\mathcal{A}$, correspondente ao domínio das configurações de hiperparâmetros a serem exploradas; o número de partições K adotado no processo de validação cruzada temporal; a métrica de avaliação $\mathcal{M}$, responsável por quantificar o desempenho do modelo; o valor de N_e que corresponde ao número de episódios de interação com o ambiente; o T, que representa o número máximo de passos por episódio. Além disso, o parâmetro γ corresponde ao fator de desconto aplicado às recompensas futuras, e α controla a importância do termo de entropia na função objetivo, incentivando políticas estocásticas. A constante τ é utilizada na atualização suave da rede de valor alvo. O modelo de detecção de anomalias a ser ajustado é representado por Φ, e $\Omega' \subseteq \Omega$ indica o subespaço de busca de possíveis intervalos de configurações dos hiperparâmetros. Por fim, o parâmetro λ corresponde à taxa de aprendizado, controlando a velocidade com que o coeficiente de entropia α é ajustado para alcançar o nível desejado de entropia $\bar{\mathcal{H}}$, promovendo o equilíbrio entre exploração e convergência.

Algoritmo 1 Função de avaliação de desempenho.

Função AVALIAR(Φ, $\boldsymbol{\omega}$, $\mathcal{D}$, K, $\mathcal{E}$)
  $\mathcal{V} \leftarrow \{\}$
  Para $(\mathcal{D}_k^{\text{treino}}, \mathcal{D}_k^{\text{teste}}) \in \text{TimeSeriesSplit}(\mathcal{D}, K)$ **faça**
    $\overline{\mathcal{D}}_k^{\text{treino}} \leftarrow \mathcal{D}_k^{\text{treino}} \setminus \mathcal{D}_{\text{anômalos}}^{\text{treino}}$
    $\Psi_k \leftarrow \Phi(\overline{\mathcal{D}}_k^{\text{treino}}; \boldsymbol{\omega})$
    $\hat{\mathbf{y}}_k \leftarrow \Psi_k(\mathcal{D}_k^{\text{teste}})$
    $g_k \leftarrow \mathcal{G}(\hat{\mathbf{y}}_k)$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{g_k\}$
  fim Para
  $\bar{g} \leftarrow \frac{1}{K}\sum_{k=1}^{K} g_k$
  $std \leftarrow \sqrt{\frac{1}{K}\sum_{k=1}^{K}(g_k - \bar{g})^2}$
  $e \leftarrow \bar{g} - std$
  Retorne e
fim Função

Algoritmo 2 Otimização de hiperparâmetros com SAC.

Entrada: $\mathcal{D}$, $\mathcal{A}$, K, $\mathcal{M}$, γ, α, τ, Φ, Ω', λ, N_e, T
Inicializar redes críticas Q_{θ_1}, Q_{θ_2}, valor V_ψ, alvo $V_{\bar{\psi}}$, política π_ϕ
Buffer de replay $\mathcal{B} \leftarrow \emptyset$
Para $e = 1$ até N_e **faça**
  nicializar $s_0 \leftarrow \emptyset$
  Para $t = 1$ até T **faça**
    Amostrar $\boldsymbol{\omega}_t \sim \pi_\phi(\cdot|s_t)$
    $\bar{r}_t \leftarrow$ AVALIAR$(\Phi, \boldsymbol{\omega}_t, \mathcal{D}, K, \mathcal{M})$
    $r_t \leftarrow \mathcal{R}(\bar{r}_t)$
    $s_{t+1} \leftarrow f(s_t, \boldsymbol{\omega}_t, r_t)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, \boldsymbol{\omega}_t, r_t, s_{t+1})\}$
    Se $|\mathcal{B}| \geq$ tamanho_lote **então**
      $\mathcal{B}_{\text{mini}} \sim$ Amostrar$(\mathcal{B})$
      Para todo $(s, \boldsymbol{\omega}, r, s') \in \mathcal{B}_{\text{mini}}$ **faça**
        $\tilde{\boldsymbol{\omega}}' \sim \pi_\phi(\cdot|s')$
        Atualizar Q_{θ_i}, V_ψ, π_ϕ, α, $V_{\bar{\psi}}$
      fim Para
    fim Se
    $s_t \leftarrow s_{t+1}$
  fim Para
fim Para
Retorne $\boldsymbol{\omega}^* \leftarrow \arg\max_{\boldsymbol{\omega}}$ AVALIAR$(\Phi, \boldsymbol{\omega}, \mathcal{D}, K, \mathcal{M})$

Nesta formulação, as funções de valor de ação Q_{θ_1} e Q_{θ_2}, responsáveis por estimar o retorno esperado de uma determinada configuração de hiperparâmetros, são aproximadas por redes neurais do tipo MLP. Para cada rede Q_{θ_i}, recebe como entrada o vetor de estado $\boldsymbol{s}$ e a ação contínua $\boldsymbol{\omega} \in \mathcal{A}$, representando uma instância de configuração de hiperparâmetros. O resultado é uma estimativa escalar do valor esperado da recompensa. A escolha por MLP se deve à sua capacidade de aproximação e boa performance empírica em ambientes com espaços de ação contínuos de baixa a média dimensionalidade. Essas redes são treinadas para minimizar o erro de Bellman suavizado, utilizando a perda computada com base na política estocástica atual π_ϕ, também parametrizada

por uma rede MLP com saída Gaussiana reparametrizada. As atualizações dos parâmetros θ_i das redes críticas são realizadas via descida do gradiente estocástico, a partir de amostras de *replay buffer*. Esse processo permite que o agente aprenda de forma eficiente, utilizando dados fora da política atual, característica que distingue o SAC como um método *off-policy* robusto.

Essa abordagem fornece um mecanismo robusto para a otimização de hiperparâmetros em cenários complexos e não triviais, como detecção de anomalias em séries temporais, em que o espaço de busca é altamente não linear, ruidoso e muitas vezes com restrições de avaliação computacional. O método proposto foi denominado HBSAC-ADT (*Hypertuning-based Soft Actor-Critic for Anomaly Detection in Time Series*). Para avaliação comparativa da abordagem proposta, os métodos TPE e CMA-ES serviram como base, mantendo a estrutura de validação cruzada temporal.

C. Modelos de detecção de anomalias

Os modelos de aprendizado de máquina para detecção de anomalias dentro do escopo desta pesquisa são apresentados nas subseções a seguir. Esses modelos fornecem a fundamentação necessária para a avaliação do método proposto de otimização de hiperparâmetros como um MDP.

1) Isolation Forest: O iForest, proposto por [15], é um modelo não supervisionado para detecção de anomalias baseado em árvores que visa isolar amostras através de particionamentos recursivos. Para cada ponto inédito, o algoritmo gera uma pontuação, representada pela Equação (12), em que $E(h(x))$ é o valor médio das profundidades que um ponto inédito atinge em todas as árvores da floresta. O fator de normalização $c(k)$ (Equação (13)) representa a profundidade média mal-sucedida em uma busca na árvore binária, com k representando o número de pontos usados na construção da árvore; e a função $H(i)$ (Equação (14)) o número harmônico estimado.

$$S(x,k) = 2^{-\frac{E(h(x))}{c(x)}}, \tag{12}$$

$$c(x) = 2H(k-1) - \left(\frac{2(k-1)}{k}\right), \tag{13}$$

$$H(i) = \ln i + 0{,}5772156649. \tag{14}$$

2) Principal Component Analysis: O PCA consiste na redução de dimensionalidade linear usando decomposição de valores singulares dos dados para projetá-los em um espaço dimensional inferior. A abordagem usando PCA para detecção de anomalias foi proposta por [16]. O método consiste em calcular uma matriz de covariância dos dados que é decomposta por vetores ortogonais, denominados autovetores, associados a autovalores. A maior parte da variação nos dados é capturada por autovetores com altos autovalores. Portanto, valores discrepantes são caracterizados no hiperplano por autovetores com pequenos autovalores. O algoritmo gera uma pontuação baseada na soma da distância projetada de uma amostra em todos os autovetores.

3) Local Outlier Factor: O algoritmo LOF, proposto por [17], é um método de detecção de anomalias que consiste em calcular o desvio de densidade local de um determinado ponto em relação aos pontos vizinhos. A estimativa da densidade local é obtida calculando as distâncias entre os k-vizinhos mais próximos. O método compara quais pontos têm densidade menor que seus vizinhos; essas distâncias são utilizadas para calcular a distância de alcance, definida como o máximo da distância entre dois pontos e a k-distância em relação ao ponto de referência. A distância de alcance determina o quanto os pontos estão próximos. A pontuação do LOF é obtida pela razão entre a média das densidades de alcance local, o número k de vizinhos e a densidade de alcance local do ponto.

4) Lightweight on-line detector of anomalies: O algoritmo LODA, proposto por [18], consiste em uma coleção de k histogramas unidimensionais $\{h_i\}_{i=1}^{k}$, cada um aproximando a densidade de probabilidade dos dados de entrada projetados em um único vetor de projeção $\{\boldsymbol{w}_i \in \mathbb{R}^d\}_{i=1}^{k}$. A pontuação do LODA é uma média do logaritmo de probabilidades estimadas em vetores de projeção individuais, descrita pela Equação (15), em que $\hat{p}_i$ denota a probabilidade estimada pelo i-ésimo histograma e $\boldsymbol{w}_i$ denota o vetor da projeção correspondente.

$$f(\boldsymbol{x}) = -\log\Big(\prod_{i=1}^{k} \hat{p}_i(\boldsymbol{x}^\top \boldsymbol{w}_i)\Big)^{\frac{1}{k}}. \tag{15}$$

D. Métricas de desempenho

A principal métrica adotada para a avaliação do desempenho dos modelos de detecção de anomalias é a área sob a curva característica de operação do receptor (AUC-ROC), que representa uma forma eficiente de avaliar a taxa de falsos positivos e a taxa de verdadeiros positivos. Essa função foi utilizada como recompensa para o sistema HBSAC-ADT. O cálculo final da recompensa considera a diferença entre a média e o desvio-padrão obtidos no processo de validação cruzada, multiplicado por um fator de 100; isso permite capturar diferenças significativas entre os valores, corroborando para o aprendizado do sistema e evitando soluções associadas a mínimos locais. Além disso, as métricas derivadas da matriz de confusão também são importantes para analisar a efetividade dos modelos para a tarefa de detecção de anomalias. *Accuracy* corresponde à proporção de predições corretas do modelo. *Precision* é definida como o número de verdadeiros positivos sobre o número de verdadeiros positivos somado ao número de falsos positivos. *Recall* é definido como o número de verdadeiros positivos sobre o número de verdadeiros positivos somado ao número de falsos negativos. *Specificity* corresponde ao número de verdadeiros negativos sobre o número de verdadeiros negativos somado ao número de falsos positivos. O F_1-*score* é a média harmônica de *precision* e *recall* [1].

IV. Estudo de Caso

No setor industrial de petróleo e gás, a segurança operacional depende diretamente do monitoramento contínuo dos processos, impulsionado por fatores econômicos, ambientais e regulatórios. A produção de petróleo e gás exige estruturas

complexas; nesse cenário, um gerenciamento eficaz é essencial para processar o grande volume de dados gerados pelos diversos componentes de um poço. Essas estruturas abrangem sensores, sistemas elétricos, mecânicos e hidráulicos, integrados em uma rede operacional [19].

A Petrobras, a maior empresa de petróleo e gás do Brasil, disponibilizou o conjunto de dados de séries temporais multivariadas com informações de variáveis de sensores utilizados no monitoramento de processos de poços marítimos surgentes de petróleo. O *dataset* 3W [20] é composto por múltiplos sensores de pressão e temperatura. O conjunto de dados possui três categorias de dados: reais; simulados; desenhados. Este trabalho se restringe a dados reais. Os dados históricos possuem três tipos de períodos: normalidade; transientes; anomalia. Para o desenvolvimento dos modelos, foram considerados dados transientes como anômalos, com a finalidade de desenvolver modelos que sirvam de prognóstico para detecção de anomalias e não somente de diagnóstico. Nesse cenário, é possível intervir no processo de falha. A amostra de dados reais contém 14.516.197 observações, com 9.439.612 observações com padrões de normalidade e 5.076.585 observações anômalas.

A motivação dos dados consiste em monitorar informações para detectar anomalias em poços de petróleo marítimos operados por elevação natural. Os sensores são: pressão do fluido do *Permanent Downhole Gauge* (P-PDG); pressão e temperatura do fluido no *Temperature and Pressure Transducer* (P-TPT e T-TPT); pressão do fluido à válvula *Choke* de Produção (P-MON-CKP); temperatura jusante à válvula *Choke* de Produção (T-JUS-CKP); pressão variável a montante do gás *lift choke* (P-JUS-CKGL); variável de temperatura a montante do gás *lift choke* (T-JUS-CKGL); vazão de gás *lift* (QGL). As unidades de medida para pressão e temperatura são, respectivamente: Pascal (Pa) e graus Celsius (°C). As anomalias são classificadas por especialistas da área de elevação e escoamento de petróleo, conforme descrito por [20].

V. Resultados e Discussão

A. Pré-processamento de dados

Os dados das séries temporais foram transformados em estatísticas das variáveis; foram utilizados a média, valores mínimo, máximo e desvio-padrão, com base no método de janelamento de tempo para intervalos de 5 minutos, para cada poço de petróleo. Essa abordagem permite resumir e interpretar grandes volumes de dados ao longo do tempo, facilitando a identificação de padrões e tendências. Nesse contexto, o intervalo é categorizado como anômalo se houver pelo menos um evento. Os dados foram divididos em dois conjuntos de amostras, levando em consideração a ordem temporal dos dados de cada poço de petróleo, com 80% alocados para desenvolvimento e 20% para validação, representando 32.938 e 8.241 amostras, respectivamente. Valores nulos foram descartados; com isso, a variável T-JUS-CKGL foi removida da análise. A padronização dos dados foi adotada utilizando a distribuição $\mathcal{N}(0,1)$, devido às diferenças nas magnitudes das variáveis preditoras. O método consiste em escalonar as variáveis de forma que a média seja zero e o desvio-padrão seja igual a um. Essa abordagem é utilizada durante o processo de treinamento dos modelos. Após a seleção do modelo com melhor desempenho, os parâmetros μ e σ são armazenados para a transformação dos dados de teste.

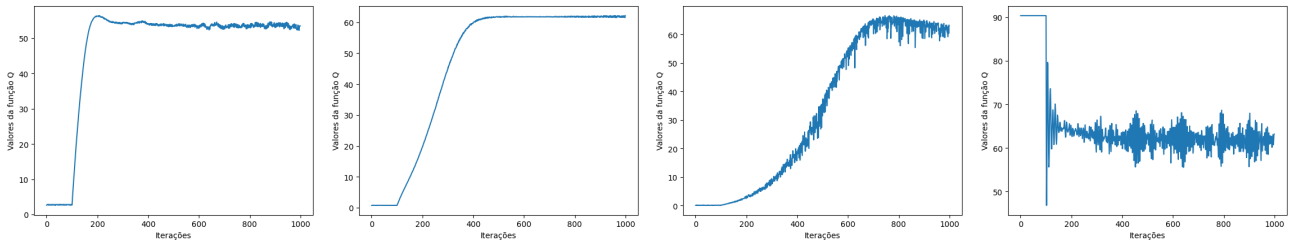
B. Desempenho dos modelos

Para o treinamento dos modelos foram consideradas 3 divisões das séries temporais utilizadas para validação cruzada; as porcentagens de anomalias nos subconjuntos de teste observadas para esse processo foram, respectivamente, 10,5%, 10,2% e 27,8%. A terceira divisão apresenta uma taxa de anomalias mais alta devido à falha do processo ocorrer principalmente no final da série temporal. As configurações do HBSAC-ADT foram: o *buffer* com tamanho 1.000; tamanho do *batch* de 128; fator de desconto de 0,99; a taxa de atualização do *target network* de 0,005; a taxa de aprendizado de 0,0001; o coeficiente de entropia *"auto"* para adaptar automaticamente para equilibrar os fatores *exploration* e *exploitation*. Os parâmetros do MLP utilizam duas camadas ocultas com 256 neurônios cada, ativadas por *Rectified Linear Unit* (ReLU), proporcionando uma capacidade expressiva e suficiente para aprender políticas complexas. Para avaliação dos métodos HBSAC-ADT, CMA-ES e TPE foram consideradas 1.000 iterações.

Os espaços de busca para os hiperparâmetros do modelo iForest foram: $(5, 256)$ para quantidade de árvores, $(0{,}3, 1{,}0)$ para máximo de amostras, $(0{,}3, 1{,}0)$ para máximo de variáveis. Para o LOF, o número de vizinhos foi de $(3, 25)$. O PCA considerou o percentual do número de componentes variando de $(0{,}5, 1{,}0)$. Para o LODA, o intervalo do número de *bins* foi $(50, 5.000)$ e o número de cortes foi de $(10, 1.000)$. O intervalo para o fator de contaminação dos modelos de detecção de anomalias foi definido por $(0{,}0001, 0{,}005)$. Para os demais parâmetros dos modelos que não foram mencionados, foram definidos como os valores padrões.

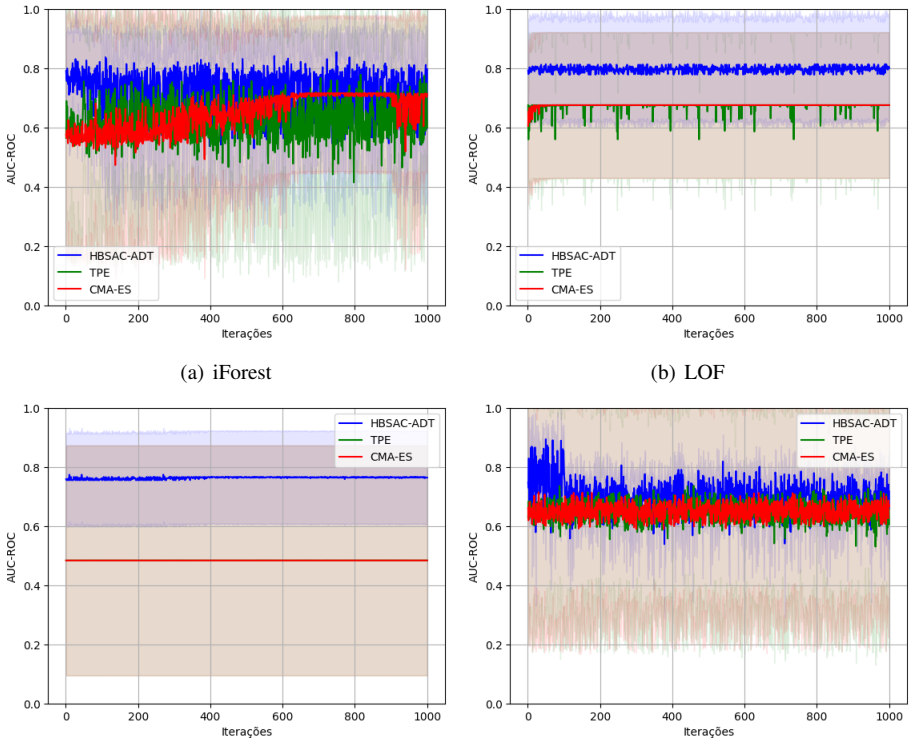
A Figura 1 representa os valores da função Q associada ao método HBSAC-ADT em relação à quantidade de iterações para cada modelo de detecção de anomalias; nota-se que os valores convergem para valores estáveis antes de chegar no limite de iterações. Os resultados do desempenho dos modelos para cada processo de otimização de hiperparâmetros estão representados na Figura 2. Os melhores hiperparâmetros foram obtidos com o método HBSAC-ADT. Para todos os cenários, o método HBSAC-ADT resultou em valores de AUC-ROC superiores comparados aos métodos tradicionais CMA-ES e TPE. Isso também pode ser visto conforme a Tabela I, com destaque para os modelos iForest e LOF, que obtiveram os melhores desempenhos. O tempo de processamento, em minutos, do modelo iForest para os métodos HBSAC-ADT, CMA-ES e TPE, foi, respectivamente: 53, 11, 14. Para o modelo LOF: 109, 142, 131. Para o PCA: 18, 8, 3. Por fim, para o LODA: 144, 61, 27.

O fator de contaminação corresponde ao percentual de eventos anômalos; os valores obtidos, considerando o método HBSAC-ADT, para os modelos iForest, PCA, LODA e LOF



(a) iForest (b) LOF (c) PCA (d) LODA

Figura 1. Valores da função Q para cada modelo em função dos episódios.



(a) iForest (b) LOF

(c) PCA (d) LODA

Figura 2. Valores de AUC-ROC para cada modelo e método de otimização em função da quantidade de iterações.

foram, respectivamente: 0,057%, 0,495%, 0,283%, 0,493%. A configuração de melhor desempenho para o modelo iForest foi: 79 árvores na floresta; 100% para o número máximo de variáveis em cada árvore; e 31% para o número de amostras extraídas para inferência do modelo. Para o LOF, o número de vizinhos foi 3. Para o PCA, 36% das componentes principais foram consideradas na composição do modelo. Os parâmetros para o LODA foram: número de *bins* do histograma definido como 1472 e número de cortes aleatórios definido como 433.

A Tabela II apresenta os valores das métricas de desempenho para o conjunto de validação. Para o iForest, o método HBSAC-ADT permitiu alcançar o melhor desempenho comparado aos demais, sendo o maior valor de AUC-ROC observado. Para o PCA, os resultados foram idênticos; isso é justificado pelo fato de o modelo ser determinístico e linear, pois é baseado em projeção. Para os modelos LOF e LODA, o método HBSAC-ADT apresentou resultados semelhantes e, para alguns casos, superiores.

VI. Conclusão

A otimização de hiperparâmetros desempenha um papel crucial na obtenção de uma boa capacidade de generalização por parte dos algoritmos de detecção de anomalias. A busca por métodos mais robustos e eficazes tem despertado interesse em diversos setores que utilizam essas técnicas como suporte à tomada de decisão. Nesse contexto, o método proposto, HBSAC-ADT, revelou-se uma abordagem robusta e competitiva em relação aos métodos tradicionais amplamente empregados na literatura para a otimização de hiperparâmetros. A proposta introduz um novo paradigma de seleção adaptativa de hiperparâmetros, com destaque para sua aplicabilidade em cenários de detecção de anomalias em séries temporais multivariadas, especialmente com o uso de métodos não supervisionados.

Tabela I
MÉDIA E DESVIO-PADRÃO DO AUC-ROC NA VALIDAÇÃO CRUZADA.

Modelo	HBSAC-ADT	CMA-ES	TPE
iForest	**0,855** (0,118)	0,700 (0,235)	0,756 (0,225)
LOF	**0,814** (0,182)	0,676 (0,246)	0,676 (0,246)
PCA	**0,775** (0,159)	0,484 (0,389)	0,484 (0,389)
LODA	**0,891** (0,062)	0,707 (0,295)	0,728 (0,268)

Tabela II
MÉTRICAS DE AVALIAÇÃO PARA O CONJUNTO DE VALIDAÇÃO.

Modelo	Método	*Accuracy*	*Precision*	*Recall*	*Specificity*	F_1-*score*	AUC-ROC
iForest	HBSAC-ADT	**0,971**	**0,916**	**0.986**	**0,950**	**0,965**	**0,976**
	CMA-ES	0,895	0,733	0,984	0,861	0,840	0,922
	TPE	0,845	0,659	0,925	0,814	0,770	0,870
LOF	HBSAC-ADT	0,975	0,919	**1,000**	0,965	0,958	0,982
	CMA-ES	0,952	0,976	0,849	**0,992**	0,908	0,921
	TPE	**0,992**	**0,979**	0,992	**0,992**	**0,985**	**0,992**
PCA	HBSAC-ADT	**0,958**	**0,976**	**0,872**	**0,992**	**0,921**	**0,932**
	CMA-ES	**0,958**	**0,976**	**0,872**	**0,992**	**0,921**	**0,932**
	TPE	**0,958**	**0,976**	**0,872**	**0,992**	**0,921**	**0,932**
LODA	HBSAC-ADT	0,945	0,874	**0,941**	0,947	**0,960**	**0,944**
	CMA-ES	0,944	**0,998**	0,831	**0,999**	0,907	0,915
	TPE	**0,960**	0,989	0,890	0,995	0,937	0,943

Agradecimentos

Este estudo foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Brasil - Código de Fomento 001.

REFERÊNCIAS

[1] R. M. A. Oliveira, M. O. Sant'Anna, and P. H. F. da Silva, "Explainable machine learning models for defects detection in industrial processes," *Computers & Industrial Engineering*, p. 110214, 5 2024.

[2] N. Hansen and S. Kern, "Evaluating the CMA Evolution Strategy on Multimodal Test Functions," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3242, pp. 282–291, 2004.

[3] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems," *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 533–541, 6 2020.

[4] H. S. Jomaa, J. Grabocka, and L. Schmidt-Thieme, "Hyp-RL : Hyperparameter Optimization by Reinforcement Learning," *arXiv*, 6 2019.

[5] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, F. Hutter, and R. Calandra, "On the Importance of Hyperparameter Optimization for Model-based Reinforcement Learning," *Proceedings of Machine Learning Research*, vol. 130, pp. 4015–4023, 2 2021.

[6] J. Wu, S. P. Chen, and X. Y. Liu, "Efficient hyperparameter optimization through model-based reinforcement learning," *Neurocomputing*, vol. 409, pp. 381–393, 10 2020.

[7] Z. Wang, Y. Wang, H. Xu, and Y. Wang, "Effective Anomaly Detection Based on Reinforcement Learning in Network Traffic Data," *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, vol. 2021-December, pp. 299–306, 2021.

[8] C. Liu and Y. Zhang, "Hyper-parameter optimization based on soft actor critic and hierarchical mixture regularization," *arXiv*, 12 2021.

[9] D. Zha, K. H. Lai, M. Wan, and X. Hu, "Meta-AAD: Active anomaly detection with deep reinforcement learning," *Proceedings - IEEE International Conference on Data Mining, ICDM*, vol. 2020-November, pp. 771–780, 11 2020.

[10] M. Yu and S. Sun, "Policy-based reinforcement learning for time series anomaly detection," *Engineering Applications of Artificial Intelligence*, vol. 95, p. 103919, 10 2020.

[11] J. Watts, F. Van Wyk, S. Rezaei, Y. Wang, N. Masoud, and A. Khojandi, "A Dynamic Deep Reinforcement Learning-Bayesian Framework for Anomaly Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 22884–22894, 12 2022.

[12] K. Arshad, R. F. Ali, A. Muneer, I. A. Aziz, S. Naseer, N. S. Khan, and S. M. Taib, "Deep Reinforcement Learning for Anomaly Detection: A Systematic Review," *IEEE Access*, vol. 10, pp. 124017–124035, 2022.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *35th International Conference on Machine Learning, ICML 2018*, vol. 5, pp. 2976–2989, 1 2018.

[14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft Actor-Critic Algorithms and Applications," *arXiv*, 12 2018.

[15] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 413–422, 2008.

[16] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A Novel Anomaly Detection Scheme Based on Principal Component Classifier," in *IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03)*, vol. 9, pp. 1–9, 2003.

[17] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *SIGMOD 2000 - Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 93–104, 2000.

[18] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, pp. 275–304, 2 2016.

[19] A. P. F. Machado, R. E. V. Vargas, P. M. Ciarelli, and C. J. Munaro, "Improving performance of one-class classifiers applied to anomaly detection in oil wells," *Journal of Petroleum Science and Engineering*, vol. 218, p. 110983, 11 2022.

[20] R. E. V. Vargas, C. J. Munaro, P. M. Ciarelli, A. G. Medeiros, B. G. d. Amaral, D. C. Barrionuevo, J. C. D. d. Araújo, J. L. Ribeiro, and L. P. Magalhães, "A realistic and public dataset with rare undesirable real events in oil wells," *Journal of Petroleum Science and Engineering*, vol. 181, p. 106223, 10 2019.