Sistema de Decisão Autônoma de Detecção para Bloqueio de Fraudes Financeiras

Gustavo Anthony R. Santana[1], **José Breno K. B. Fontes**[1], **Ramon dos Santos Oliveira**[1], **Stephanie Kamarry A. de Sousa**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Sergipe (IFS)
Lagarto – SE – Brazil

contatogustavostna@gmail.com
ramonoliveira1604@gmail.com
brbrenokayki@gmail.com
stephanie.sousa@ifs.edu.br

Abstract. *Real-time anti-fraud systems need to reconcile analytical precision and low latency, given the accelerated expansion of instant payments. This paper presents a software architecture based on XGBoost and the FastAPI framework, optimized by consuming contiguous memory matrices with NumPy. The engine was validated using the Paysim and IEEE-CIS datasets, achieving an average PR-AUC across models of 0.7348 and a precision of 0.4568 through Cost-Sensitive Learning. The architecture was validated under estresse tests and proved capable of operating at a peak of 406.75 RPS with zero application errors and an average latency across models operating at 405 ms.*

Resumo. *Os sistemas de antifraude em tempo real necessitam conciliar a precisão analítica e a baixa latência, dado a expansão acelerada dos pagamentos instantâneos. Este artigo apresenta uma arquitetura de software baseada em XGBoost e no framework FastAPI, otimizada pelo consumo de matrizes em memória contígua com NumPy. O motor foi validado com os datasets Paysim e IEEE-CIS, e alcança uma média entre modelos de PR-AUC de 0.7348 e precisão de 0.4568 através de Cost-Sensitive Learning. A arquitetura foi validada sob testes de estresse e mostrou-se capaz de operar um pico de 406,75 RPS sem erros na aplicação e latência média entre modelos operante à 405 ms.*

1. Introdução

A revolução digital permite a adoção de métodos sofisticados de transações financeiras para pagamentos instantâneos. Contudo, este crescimento trouxe, também, métodos sofisticados de fraudes financeiras, as quais aumentam significativamente o arsenal de criminosos desta categoria. Os métodos tradicionais de detecção de fraudes financeiras atuam através de um conjunto de regras pré-definidas e reconhecimento de padrões históricos [Odufisan et al. 2025]. Atualmente, isto significa utilizar inferência humana sujeita a erros, somada à comparação de padrões defasados; contra a possibilidade dos criminosos alterarem o *modus operandi* em apenas alguns segundos, dado o acesso à tecnologia de pagamentos instantâneos.

As abordagens dos sistemas de prevenção a estas fraudes sofrem com três problemas clássicos: 1) desbalanceamento extremo dos dados de treino; 2) mudança dos

padrões de fraudes; 3) seleção de métricas de forma equivocada [Hajek et al. 2023]. Clientes reais frequentemente possuem sua experiência interrompida devido a bloqueios de transações legítimas. Logo, o desbalanceamento de dados permite a existência de falsos positivos em uma escala indesejada, o que corrobora para o dilema de "Fricção vs Perda"[Hajek et al. 2023], fator principal atrelado a taxa de cancelamento de produtos desta natureza. Ao bloquear uma transação legítima, a empresa não só impede os lucros da transação, como também gera uma fricção indesejada pelo usuário final, o qual, dado a frustração, abandona o produto inicial e derrete a taxa de *lifetime value* (LTV).

A adoção de Inteligência Artificial (IA) para auxiliar no reconhecimento de padrões é o ponto de discussão mais acalorado do estado da arte atual, visto que o reconhecimento do fluxo contínuo de dados tenha, de fato, sido potencializado através das redes neurais [Immadisetty 2025]. Entretanto, o uso de IA visando uma solução em tempo real apresenta um desafio de latência emergente. Modelos pesados sofrem o problema de *TCP Backlog*, isto é, com o volume exacerbado de tráfego, a taxa de chegada das requisições excede o tempo de resposta do modelo, o que leva a *timeouts* inesperados. Sob esta ótica, a arquitetura em nuvem torna-se imprescindível a fim de mitigar aspectos de escalabilidade e flexibilidade de armazenamento de dados [Sekar 2023]

Mediante ao exposto, o presente trabalho aborda uma arquitetura de engenharia de *software* para minimizar gargalos I/O em inferências preditivas. Os dados são processados diretamente em matrizes de memória contígua, ou seja, são armazenados lado a lado na memória física do computador, isso garante que a arquitetura implementada usufrua de forma mais adequada das vantagens de localidade espacial dos dados. Computacionalmente, esta mesma organização espacial permite a *CPU* utilizar instruções *SIMD (Single Instruction, Multiple Data)*, as quais processam cálculos do modelo de IA em um ciclo único de *clock*.

Além da otimização apresentada, as contribuições arquiteturais desta pesquisa compõem o próprio conceito do microsserviço. O sistema está construído com base em um paradigma totalmente assíncrono, com a finalidade de suportar alta concorrência sem bloqueio de *threads*. Outrossim, o projeto tende a utilizar um *Expected Value Framework*, cruzando o risco preditivo com a matriz de custo da operação. Por fim, toda a proposta foi validada empiricamente por meio de testes de estresse, com foco em mapear a latência e a estabilidade de requisições por segundo (RPS) sob ataques simulados

Portanto, com este trabalho propomos uma arquitetura de *software* e inferência em tempo real com foco em construir um ecossistema integrado ao unir uma API assíncrona de baixa latência e capaz de comparar os efeitos da utilização de artefatos preditivos distintos. Destarte, visa oferecer um motor de decisão escalável e com custo computacional sustentável para *gateways* de pagamentos.

2. Trabalhos Relacionados

No contexto atual, a literatura enfrenta um significativo problema frente ao treinamento de modelos de predição de fraudes devido à escassez de dados públicos, provocada pela necessidade de proteção de dados, o que frequentemente resulta no uso de dados viciados e desbalanceados. Neste ponto de vista, a introdução de um simulador baseado em transações financeiras *mobiles* advindas de logs, o PaySim, que gera dados sintéticos visa mitigar a ausência de contexto real [Lopez-Rojas et al. 2016]. Ade-

mais, [Jesus et al. 2022] propuseram a suíte Bank Account Fraud (BAF), um *dataset* que compõe outros *datasets* e injeta tipos distintos de viés e simula desbalanceamento extremo, como método de avaliação para a acurácia das predições, tanto quanto a métodos de aprendizado de máquina que encaram desafios em contexto real.

Como citado, o *deep learning* e o aprendizado de máquina demonstram, níveis de acurácia satisfatórios. O uso de Redes Neurais Convolucionais, as *CNNs*, demonstra uma precisão de até 99,9% para classificação exclusiva de transações de cartões de crédito, embora necessitem de uma capacidade superior de processamento, aumentando assim o custo computacional e consequentemente, de implementação [Alarfaj et al. 2022]. Simultaneamente, a viabilidade de modelos clássicos de Floresta Aleatória, em operação de dados sintéticos, provou-se ao lidar de maneira agradável com distribuições desbalanceadas e um recall de 99,47% na detecção preditiva de atividades maliciosas [Teles 2025]. Portanto, a literatura evidencia a resolução de problemas comuns na predição financeira utilizando inteligência artificial.

Entretanto, mesmo com a alta eficácia preditiva comprovada, há uma lacuna que permeia a literatura e o desenvolvimento dos sistemas em contexto real, isto é, em ambiente de fraudes reais, os criminosos mudam de táticas rapidamente, o que exige a implementação de *pipelines* de retreinamento contínuo e aprendizado em produção. Somente com o uso dessa adaptação dinâmica será possível mitigar o *Concept Drift*, fenômeno responsável pela degradação dos modelos preditivos ao longo do tempo, advindo da alteração constante das práticas criminosas [Pelosi et al. 2025].

Além do mais, é preciso analisar os ruídos de custo computacional e latência de I/O que são percebidos apenas em produção. O volume extremo de requisições simultâneas e a exigência de processamento rápido, ou seja, com latência mínima, afetam negativamente a eficiência do sistema e a percepção do usuário frente ao mesmo [Immadisetty 2025]. Urge, portanto, uma avaliação advinda de testes rigorosos do tempo de resposta prático.

3. Metodologia

Com o intuito de estabelecer uma base sólida de testes acerca da latência, a base de dados utilizada neste trabalho é o PaySim. Trata-se de um simulador estatístico multiagente feito para registros sintéticos de transações financeiras, baseado no comportamento real de logs africanos [Lopez-Rojas et al. 2016]. Ao fornecer milhões de registros e a classificação exata entre transações legítimas e fraudelentas, o *dataset* resolve a escassez de dados e ainda fornece uma linha controlada de testes e validação da solução. Através do *dataset* citado, foi realizado a validação do limite de *throughput* da API e o consumo de memória sob estresse de requisições simultâneas.

Entretanto, esta, dentre outras, simulações em ambientes laboratoriais raramente refletem o contexto real de ambientes em produção. Como método de comprovação de generalização, e visando uma maior eficácia e confiabilidade do modelo, integrou-se o *dataset* especializado da IEEE-CIS Fraud Detection. A tabela 1 ilustra informações relevantes acerca dos *datasets*, liberando milhões de transações baseadas em e-commerce e com caracterização pelo alto ruído, fator este que é principal divisor entre ambientes de teste e produção. A literatura analisada demonstra que o XGBoost, somado às bases de dados escolhidas, torna-se capaz de prever cenários de fraudes financeiras. No contexto

do presente estudo, isto serve para provar que a solução se adequa ao contexto real de forma nativa, sem rotinas de imputação de dados.

Tabela 1. Características dos Datasets Financeiros Utilizados na Validação.

Dataset	Volume de Linhas	Colunas	Classe de Fraude (%)
PaySim	6.362.620	13	0,1291%
IEEE-CIS	590.540	394	3,49%

Foi necessário descartar métricas generalistas como meio de avaliar se o motor alcançaria a performance estimada. No âmbito de transações financeiras instantâneas as atividades fraudulentas representam menos de 0,1% do tráfego total. Devido a esta assimetria, a otimização sob a curva *ROC*, *ROC-AUC*, permite uma taxa elevada de Falsos Positivos, quer dizer, um motor baseado unicamente nessa otimização pode bloquear indevidamente milhares de clientes legítimos devido ao volume bruto de processamento diário. Logo, o modelo foi avaliado apenas sob a curva de Precisão e Revocação, PR-AUC, focando no alinhamento do negócio, isto é, maximizar a certeza ao disparar o bloqueio apenas para os fraudadores, reduzindo a fricção indesejada de análise humana.

Finalmente, o treino obedeceu a um rigor metodológico voltado para a eliminação do vazamento de dados no mercado. A aplicação de validação cruzada, o método *K-Fold*, o qual é baseado em particionamento aleatório é inválido, neste contexto específico, pois eleva as chances do modelo buscar aprender em padrões futuros, que ainda não ocorreram, para prever eventos do passado. Para aproximar ao comportamento dos fraudadores, a validação experimental adotou a divisão *Out-of-Time*. Consoante o explicitado por arquiteturas para dados tabulares dinâmicos em cenários reais aplicados a contas bancárias, o motor de predição foi treinado com dados históricos fixos e testado com um futuro inédito, sendo realizado por meio da segregação do *dataset* original utilizado.

4. Arquitetura Proposta

O cerne do motor analítico foi construido sobre o *eXtreme Gradient Boosting*, escolhido para lidar com o desbalanceamento extremo de classes sem recorrer a técnicas de construção de amostragens sintéticas. Tradicionalmente, são perceptíveis métodos de *oversampling*, abordagens que geram, artificialmente, instâncias com ruído geométrico e um *overfitting* em espaços de alta dimensionalidade além da sobrecarga do *pipeline* de dados. Em contraponto, foi preferível manipular o hiperparâmetro do eXtreme Gradient, *scale pos weight*, como abordagem de implementação de Aprendizado Sensível ao Custo, isto é, *Cost-Sensitive Learning*.

Assim sendo, o cálculo do gradiente, durante o treinamento, penaliza o algoritimo de forma, matematicamente, muito mais severa ao classificar erroneamente uma fraude. Adicionalmente, para a base de dados IEEE-CIS, adotou-se o método de construção de árvores baseado em histogramas, a fim de reduzir a complexidade computacional e permitir o treinamento nativo mesmo com variáveis ausentes. O uso de *frameworks* baseados no *XGBoost* preserva a distribuição e balanceamento dos dados originalmente dispostos no *dataset*, como também maximiza os ganhos operacionais e as economias de custo em sistemas de pagamentos *mobile* [Hajek et al. 2023].

Além das motivações pautadas frente à escolha do *eXtreme Gradient Boosting*, outro fator corroborativo para a decisão é a possibilidade do particionamento por aumento de gradiente com o algoritmo baseado em histogramas, ou seja, a alteração utilizada especificamente para o *dataset* da IEEE-CIS. Esta permite o mapeamento de limites de decisão não linear complexos e a identificação de iterações não óbvias. Somado a isto, o algoritmo opera sobre matrizes de memória contíguas, o que reforça ainda mais a escolha dado a métrica central de latência, adotada durante toda pesquisa.

Outrossim, em ambientes de microsserviços devem-se analisar *edges case* de *timeouts* e dados que não completaram o trajeto em virtude de uma requisição, ou seja, em produção é comum valores nulos. A partir desta ótica, a grande maioria dos modelos exige rotinas bem definidas de imputação de dados para que alcance o funcionamento correto. O problema resultante da abordagem, é a necessidade de processamento extra, que inviabiliza o objetivo de cumprimento dos *SLAs* em milissegundos. O contorno adotado foi a inversão do núcleo do algoritimo já existente do *XGBoost*. Durante o treino, a estrutura de árvore aprende ativamente uma rota padrão para lidar com valores ausentes; assim, as transações cujo envelope de informações se enquadrem no contexto, são direcionadas por este fluxo sem gerar exceções.

Além do mais, como em sistemas de produção com alto tráfego a latência é ditada pelo tamanho da requisição HTTP processada e pelo tempo de serialização, adotou-se o mecanismo de payload enxuto apenas para o *dataset* IEEE-CIS, dado sua extensão de colunas e o roteamento de modelos apenas na abordagem de envio de *payloads* segregados para avaliar a resiliência dos artefatos. Desse modo, o roteamento processa os vetores reduzidos, selecionados através Feature Importance, em vez de processar todo o volume bruto original. Essa abordagem aceita um abrandamento marginal na precisão em troca de manter os SLAs de latência.

A abordagem assíncrona entre a API e o motor preditivo firma um pipeline de inferência, no qual o tempo de resposta é isolado de rotinas de pré-processamento. Entretanto, em ambientes de produção a resposta de um modelo de *Machine Learning*, advinda de uma probabilidade estatística, faz-se insuficiente para guiar as ações da transação. Consequentemente, torna-se imperativa a tradução do vetor probabilístico em uma política de governança que analise o risco computacional frente ao impacto financeiro, por meio de uma camada intermediária na arquitetura.

Mesmo com toda arquitetura demonstrando ser promissora, ainda faz-se necessário traduzir o valor técnico em valor de negócio, a fim de evitar o bloqueio de transações baseadas unicamente em uma hipótese de classificação binária superior a 50%, em outras palavras, uma probabilidade de 50% de riscos para bloqueio de Falsos Positivos. Como meio de correção, o trabalho implementa um *Expected Value Framework*, apresentado na figura 1. Logo, a *API* cruza um score de risco gerado pelo próprio modelo, com uma matriz dinâmica de custo. Com a visão sobre tal implementação, uma transação de baixo valor monetário é aprovado pelo sistema, como forma de reduzir a probabilidade de cancelamento do cliente, dado que o risco de evasão, sob este contexto, supera o valor financeiro sob ameaça. Em contrapartida, tal probabilidade de risco recaindo sobre uma transação de alto valor aciona o bloqueio imediato. Portanto, a dinâmica visa equilibrar a predição agressiva do sistema, frente à experiência do usuário durante uso.

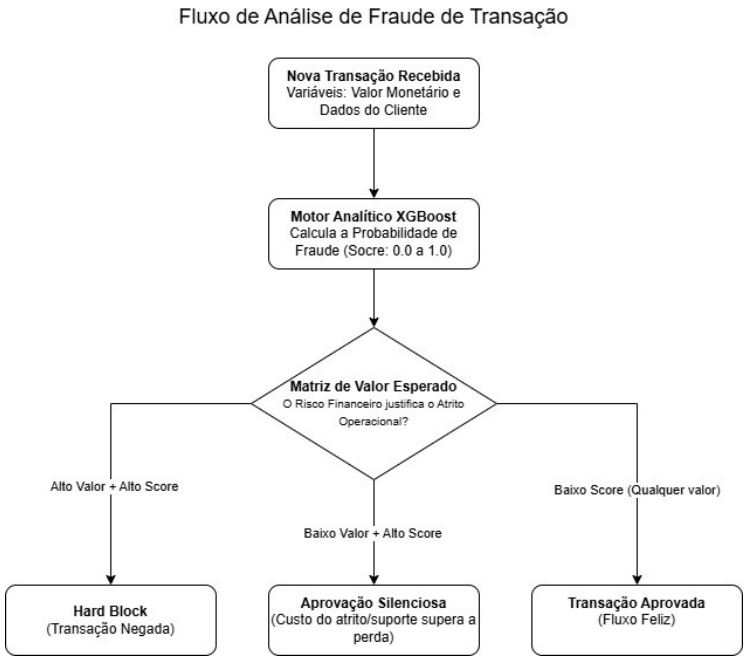


Figura 1. Fluxograma do *Expected Value Framework*: matriz de decisão dinâmica cruzando o *score* de risco preditivo com o impacto financeiro da transação.

5. Resultados e Discussões

A avaliação empírica do motor de inferência foi conduzida em ambiente controlado como meio de garantir a reprodutibilidade. O laboratório foi construído com uma máquina física atuante sob processador AMD Ryzen 7 5700U e 7,35 GB de memória RAM disponível, operante em ambiente local. Quanto ao motor preditivo e à API assíncrona, ambos foram desenvolvidos utilizando a linguagem Python na versão 3.12.19. Além disso, como meio de assegurar o isolamento de dependências e reduzir conflitos de versionamento, o ecossistema foi executado sob um ambiente virtual Python estrito, permitindo a medição direta da vazão ao explorar o limite do hardware local diretamente.

No escopo de análises, o motor construído com *XGBoost* demonstra notável capacidade de isolar anomalias sem deteriorar a base de transações. Ao considerar o desbalanceamento intrínseco aos dados de produção, a performance do modelo foi avaliada através da Área Sob a Curva de Precisão e Revocação, *PR-AUC*. A tabela 2 evidencia as métricas analíticas específicas para o artefato treinado com Paysim, o qual, após treinamento, retornou a *PR-AUC* expressiva de 0.9899, e precisão de 0.7614.

Tabela 2. Métricas Analíticas do Motor de Inferência via Paysim (XGBoost).

Métrica	Valor Obtido
Precision (Precisão)	0.7614
PR-AUC	0.9899

Ainda visando compreender a confiabilidade real da arquitetura, foi aplicada a análise em busca das mesmas métricas para o *dataset* IEEE-CIS, cujo resultado expõe a métrica de 0.1523 para precisão e 0.4798 de *PR-AUC*, o que confirma a eficácia da

aplicação do *Cost-Sensitive Learning* durante o treinamento, bem como torna explícita a divergência existente entre atuar com dados sintéticos e reais, conforme apresentado na tabela 3.

Tabela 3. Métricas Analíticas do Motor de Inferência via IEEE-CIS (XGBoost).

Métrica	Valor Obtido
Precision (Precisão)	0.1523
PR-AUC	0.4798

Destarte, as métricas indicam que mais de 76% dos alertas eram, de fato, fraudes genuínas simulando comportamento no *dataset* sintético, Paysim, taxa significativa para cenários de detecção não-supervisionada. Entretanto, em cenário ruidoso, o *dataset* da IEEE-CIS, essa taxa situou-se em 15,16%. Essa variação é reflexo direto da redução do payload para 20 variáveis contextuais, mas que ainda assim, essa taxa permitiu a captura de 73,84% de todas as fraudes do lote. Ratifica-se a necessidade de mitigação diferenciada por rotas dentro da arquitetura.

Como método de teste arquitetural da API, a camada de rede foi submetida a testes de estresse utilizando a ferramenta de automação Locust. O cenário desenhado foi pensado para simular o comportamento de ataques cibernéticos coordenados. A rampa de testes foi configurada com um teto de 1.000 usuários virtuais simultâneos, com uma taxa de injeção de 10 requisições por segundo. O objetivo do ataque foi saturar o *Event Loop* do servidor e levar o processamento concorrente da CPU ao seu limite.

Os resultados obtidos revelam dados críticos sobre o comportamento de *Machine Learning* em produção. Embora as métricas estejam em maiores detalhes na tabela 4, posterior a este parágrafo, vale ressaltar que, durante o pico do teste, a vazão máxima obtida foi de 699 requisições por segundo, processando um total de 26.285 requisições sem o registro de uma única falha da API. A latência mediana cravou uma marca de 100 milissegundos, a qual é excelente para o intuito do trabalho. Entretanto, o Percentil 99 atingiu 510 milissegundos no ápice de concorrência dos 1.000 usuários, isto é, até o ponto de estresse do hardware e recusa da conexão. Embora esta métrica ultrapasse as janelas de *timeout* mais agressivas do ecossistema PIX, a sustentação de zero falhas da API e uma mediana de 100 ms sob tráfego massivo provam, empiricamente, que a arquitetura obteve impacto sob a carga de forma resiliente, baseado no modelo gerado via treinamento do Paysim.

Tabela 4. Indicadores Chave de Performance (KPI) da API Sob Estresse - Paysim.

Métrica de Engenharia	Resultado Empírico
Total de Requisições	26.285
Requisições Falhas (API)	0
Requisições Falhas (SO)	100
Vazão Máxima (RPS)	699,4
Latência Mediana (p50)	100 ms
Latência Percentil 99 (p99)	510 ms

Utilizando o mesmo modelo, comprovado o limite máximo da API, faz-se ne-

cessário atestar a capacidade de resiliência; a tabela 5 demonstra bem os resultados do teste de carga. Neste sentido, o teto de usuários foi reduzido a 300 simultâneos, mantendo a taxa de injeção intacta. Dessa forma, a vazão máxima obtida foi de 329 requisições por segundo, processando um total de 56.438 requisições sem registros de falha da API. Para validar a resiliência, a API foi submetida ao teste durante 3 minutos ininterruptos, com critério de aceite sendo a ausência de erros de recusa da pilha TCP do sistema operacional.

Tabela 5. Indicadores Chave de Performance (KPI) da API sob carga constante - Paysim

Métrica de Engenharia	Resultado Empírico
Total de Requisições	56.438
Requisições Falhas (API)	0
Requisições Falhas (SO)	2049
Vazão Máxima (RPS)	329,5
Latência Mediana (p50)	570 ms
Latência Percentil 99 (p99)	950 ms

Os resultados obtidos para o *dataset* IEEE-CIS intensificam ainda mais a necessidade de roteamentos de modelos ao se visar a latência em rede. Para a experimentação mantiveram-se os mesmos parâmetros utilizados no *dataset* anterior, a fim de garantir comparação confiável dos cenários. Deste modo, sob simulação de ataque, visando estabelecer o limite máximo da API, isto é, até a saturação do hardware e da pilha TCP do sistema operacional, foram processadas 17.366 requisições, com latência mediana de 620 milissegundos e uma vazão de 192.2 requisições por segundo conforme detalhado na tabela 6 abaixo.

Tabela 6. Indicadores Chave de Performance (KPI) da API Sob Estresse - IEEE-CIS.

Métrica de Engenharia	Resultado Empírico
Total de Requisições	17.366
Requisições Falhas (API)	0
Requisições Falhas (SO)	2049
Vazão Máxima (RPS)	192,2
Latência Mediana (p50)	620 ms
Latência Percentil 99 (p99)	2200 ms

Ademais, para o teste de resiliência, aplicado sob o mesmo modelo, contendo os mesmos parâmetros e critérios de aceite do teste realizado no modelo anterior, processou o total de 66.013 requisições, com latência mediana de 330 milissegundos e vazão de 484 requisições por segundo. Essa diferenciação mostra como o comportamento computacional tende a se estabilizar com o uso do hardware alocado e torna explícito que os resultados obtidos com o teste de estresse são picos de processamento que, embora sejam próximos, não refletem o comportamento estável final.

Tabela 7. Indicadores Chave de Performance (KPI) da API sob Carga Constante - IEEE-CIS

Métrica de Engenharia	Resultado Empírico
Total de Requisições	66.013
Requisições Falhas (API)	0
Requisições Falhas (SO)	0
Vazão Máxima (RPS)	484
Latência Mediana (p50)	330 ms
Latência Percentil 99 (p99)	680 ms

6. Conclusão

Este trabalho apresenta uma arquitetura de engenharia projetada para sanar o dilema de tempo real na detecção de fraudes em pagamentos instantâneos. Para provar ser uma solução eficaz, foi necessário integrar um modelo preditivo baseado em *XGBoost*, com otimização via *Cost-Sensitive Learning* e uma API web assíncrona alimentada por matrizes contíguas em memória. Sob ótica analítica, o presente trabalho entrega uma assertividade nos bloqueios e minimiza a fricção operacional, sendo comprovado pelas métricas *PR-AUC* com valor de 0.9899 e *Precisão* com valor de 0.7614 para o artefato baseado no Paysim e métricas *PR-AUC* com valor de 0.4798 e *Precisão* com valor de 0.1523 para o artefato baseado no IEEE-CIS. Com olhar voltado para design de sistemas, os testes empíricos validaram a resiliência do código, com o mesmo processando a média de 122 mil transações, sustentando um *RPS* médio de 406,5 com zero falhas. Mesmo que a latência de cauda no percentil 99 tenha ultrapassado o limite desejado, isso reflete a saturação de rede e I/O do hardware local, não um gargalo matemático.

Os trabalhos futuros devem se concentrar na resiliência analítica, na elasticidade da infraestrutura e na efetivação do Roteamento de Modelo, analisando cabeçalhos HTTP. Assim sendo, o próximo passo consiste em virtualizar a aplicação e orquestrá-la em um cluster Kubernetes em nuvem pública. Dado que o tempo médio de treinamento foi inferior a 30 segundos em hardware local, viabilizado pela seletividade de maior importância, frente à vetorização das features dos *datasets*. Este *footprint* assegura uma viabilidade em cenários de alta elasticidade, isto é, torna-se apto para a implementação em ambientes serverless containers.

Referências

Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., and Ahmed, M. (2022). Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. *Ieee Access*, 10:39700–39715.

Hajek, P., Abedin, M. Z., and Sivarajah, U. (2023). Fraud detection in mobile payment systems using an xgboost-based framework. *Information Systems Frontiers*, 25(5):1985–2003.

Immadisetty, A. (2025). Real-time fraud detection using streaming data in financial transactions. *Journal of Recent Trends in Computer Science and Engineering (JRTCSE)*, 13(1):66–76.

Jesus, S., Pombal, J., Alves, D., Cruz, A., Saleiro, P., Ribeiro, R., Gama, J., and Bizarro, P. (2022). Turning the tables: Biased, imbalanced, dynamic tabular datasets for ml evaluation. *Advances in Neural Information Processing Systems*, 35:33563–33575.

Lopez-Rojas, E., Elmir, A., and Axelsson, S. (2016). Paysim: A financial mobile money simulator for fraud detection. In *28th European modeling and simulation symposium, EMSS, Larnaca*, pages 249–255. Dime University of Genoa.

Odufisan, O. I., Abhulimen, O. V., and Ogunti, E. O. (2025). Harnessing artificial intelligence and machine learning for fraud detection and prevention in nigeria. *Journal of Economic Criminology*, 7:100127.

Pelosi, D., Cacciagrano, D., and Piangerelli, M. (2025). Explainability and interpretability in concept and data drift: a systematic literature review. *Algorithms*, 18(7):443.

Sekar, J. (2023). Optimizing cloud infrastructure for real-time fraud detection in credit card transactions. *Journal Name*, 6:381–388.

Teles, D. A. G. (2025). Aplicação de floresta aleatória para identificação de fraudes em transações financeiras.