Trabalho de Conclusão de Curso de graduação da Universidade Virtual do Estado de São Paulo (UNIVESP), como parte dos requisitos para obtenção do título de Bacharel em Ciência de Dados.

Cleyton de Souza Santos, RA 2106566

Fernando Chaves Matos, RA 2013970

Jucileide de Andrade Viana, RA 2110414

Leticia Stahl de Góes, RA 2106144

Luiz Roberto Paviani, RA 2204859

Michael Gustavo dos Santos Florentino, RA 2103034

Rodrigo da Costa Aglinskas, RA 2103846

Algoritmos de Machine Learning para Detecção de Fraudes em Cartões de Crédito: Uma Análise Comparativa entre Desempenho Preditivo e Eficiência Computacional em Ambientes Big Data com Databricks

Vídeo da apresentação do TCC

https://youtu.be/zOyx6wsy98o

Cândido Rodrigues, Tatuí, Santo André, São Paulo – SP

2025

Cleyton de Souza Santos, RA 2106566
Fernando Chaves Matos, RA 2013970
Jucileide de Andrade Viana, RA 2110414
Leticia Stahl de Góes, RA 2106144
Luiz Roberto Paviani, RA 2204859
Michael Gustavo dos Santos Florentino, RA 2103034
Rodrigo da Costa Aglinskas, RA 2103846

Algoritmos de Machine Learning para Detecção de Fraudes em Cartões de Crédito: Uma Análise Comparativa entre Desempenho Preditivo e Eficiência Computacional em Ambientes Big Data com Databricks

Trabalho de Conclusão de Curso de graduação da Universidade Virtual do Estado de São Paulo (UNIVESP), como parte dos requisitos para obtenção do título de Bacharel em Ciência de Dados.

Orientadora: Profa. Msc. Fernanda Pereira Guidotti Carneiro

Cândido Rodrigues, Tatuí, Santo André, São Paulo – SP
2025

Cleyton de Souza Santos, RA 2106566

Fernando Chaves Matos, RA 2013970

Jucileide de Andrade Viana, RA 2110414

Leticia Stahl de Góes, RA 2106144

Luiz Roberto Paviani, RA 2204859

Michael Gustavo dos Santos Florentino, RA 2103034

Rodrigo da Costa Aglinskas, RA 2103846

Algoritmos de Machine Learning para Detecção de Fraudes em Cartões de Crédito: Uma Análise Comparativa entre Desempenho Preditivo e Eficiência Computacional em Ambientes Big Data com Databricks

Trabalho de Conclusão de Curso de graduação da Universidade Virtual do Estado de São Paulo (UNIVESP), como parte dos requisitos para obtenção do título de Bacharel em Ciência de Dados.

Aprovado em: _____

Conceito: _____

BANCA EXAMINADORA

_____

Profa. Msc. Fernanda Pereira Guidotti Carneiro (Presidente)

Universidade Virtual do Estado de São Paulo

_____

Prof. Msc. Victor Hideki Yoshizumi (Membro externo)

Universidade Virtual do Estado de São Paulo

RESUMO

A detecção de fraudes em cartões de crédito representa um desafio crítico para instituições financeiras, especialmente diante do crescimento exponencial de transações digitais e da sofisticação crescente dos métodos fraudulentos. Este trabalho investiga qual algoritmo supervisionado de Machine Learning apresenta o melhor equilíbrio entre desempenho preditivo e eficiência computacional para detecção de fraudes em ambientes de Big Data. Utilizando o dataset público Credit Card Fraud Detection do Kaggle (284.807 transações, 0,172% de fraudes), foram implementados e avaliados sete modelos: Regressão Logística, Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, MLP Classifier e um Stacking Ensemble. O desenvolvimento seguiu o workflow end-to-end proposto por Géron (2019), executado na plataforma Databricks Serverless, com gerenciamento via MLflow e Unity Catalog. A metodologia incluiu divisão estratificada determinística, engenharia de 22 features, balanceamento por pesos de classe, otimização de hiperparâmetros e calibração probabilística. Os resultados confirmaram a superioridade dos modelos de Gradient Boosting em desempenho preditivo, com XGBoost liderando (F1-score: 0,8492; Score Ponderado: 0,8610). O Stacking Ensemble (F1-score: 0,8444) não superou o melhor modelo individual, embora tenha demonstrado precision elevada (0,9268). A análise de escalabilidade com volumes de até 5,7 milhões de transações revelou trade-off crítico: modelos complexos apresentaram degradação de eficiência superior a 85%, enquanto Regressão Logística manteve resiliência (degradação de 37,7%). Considerando o Score Combinado (70% desempenho + 30% eficiência), a Regressão Logística emergiu como solução mais equilibrada para alta escalabilidade (Score: 0,716), enquanto CatBoost demonstrou melhor trade-off em volumes moderados (0,775). O estudo conclui que não existe um "melhor" algoritmo universal, mas escolhas contextuais: XGBoost para máxima precisão, CatBoost para equilíbrio em volumes moderados, e Regressão Logística para processamento em tempo real de grandes volumes.

Palavras-chave: Detecção de Fraudes. Machine Learning. Gradient Boosting. Big Data. Databricks. Eficiência Computacional. Trade-off Preditivo-Computacional.

ABSTRACT

Credit card fraud detection represents a critical challenge for financial institutions, especially given the exponential growth of digital transactions and the increasing sophistication of fraudulent methods. This study investigates which supervised Machine Learning algorithm presents the best balance between predictive performance and computational efficiency for fraud detection in Big Data environments. Using the public Credit Card Fraud Detection dataset from Kaggle (284,807 transactions, 0.172% fraud rate), seven models were implemented and evaluated: Logistic Regression, Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost, MLP Classifier, and a Stacking Ensemble. Development followed the end-to-end workflow proposed by Géron (2019), executed on the Databricks Serverless platform, with management via MLflow and Unity Catalog. The methodology included deterministic stratified splitting, engineering of 22 features, class weight balancing, hyperparameter optimization, and probabilistic calibration. Results confirmed the superiority of Gradient Boosting models in predictive performance, with XGBoost leading (F1-score: 0.8492; Weighted Score: 0.8610). The Stacking Ensemble (F1-score: 0.8444) did not surpass the best individual model, although it demonstrated high precision (0.9268). Scalability analysis with volumes up to 5.7 million transactions revealed a critical trade-off: complex models exhibited efficiency degradation exceeding 85%, while Logistic Regression maintained resilience (37.7% degradation). Considering the Combined Score (70% performance + 30% efficiency), Logistic Regression emerged as the most balanced solution for high scalability (Score: 0.716), while CatBoost demonstrated the best trade-off for moderate volumes (0.775). The study concludes there is no universal "best" algorithm, but rather contextual choices: XGBoost for maximum accuracy, CatBoost for balance in moderate volumes, and Logistic Regression for real-time processing of large volumes.

Keywords: Fraud Detection. Machine Learning. Gradient Boosting. Big Data. Databricks. Computational Efficiency. Predictive-Computational Trade-off.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A evolução dos meios de pagamento, especialmente com o advento dos cartões de crédito, transformou radicalmente a forma como as transações financeiras são realizadas. Desde a criação do Diners Club[1] por Frank X. McNamara e Ralph Schneider em 1950, os cartões de crédito tornaram-se um pilar da economia moderna, sendo utilizados por 67% da população estadunidense e 40% da população brasileira com 15 anos ou mais (World Bank Group, 2024). Essa adoção massiva, no entanto, também trouxe consigo um desafio crescente: o aumento exponencial de fraudes financeiras.

Em 2023, as transações digitais nos Estados Unidos ultrapassaram 2,7 bilhões, criando um ambiente propício para fraudes cada vez mais sofisticadas (Chen, Yisong et al, 2025). Apenas no primeiro semestre de 2025, foram reportados 323.459 casos de fraude à FTC (*Federal Trade Commission*), representando um aumento de 51% em relação ao mesmo período do ano anterior (WalletHub, 2024; The Motley Fool, 2025).

Neste cenário, a segurança emerge como um fator crítico para a confiança do consumidor. Uma pesquisa da Mastercard (2024) revela que, embora 89% dos brasileiros adotem novos meios de pagamento digitais, 79% consideram a segurança como "muito importante". Johan Gleber, chefe global de Soluções de Segurança da Mastercard, destaca que "a confiança na tecnologia que governa nossas vidas está ameaçada, e ainda há trabalho a ser feito para atingir plenamente a promessa de uma economia digital" (Mastercard, 2024).

Tradicionalmente, as instituições financeiras dependiam de sistemas baseados em regras para detectar fraudes, operando com lógica simples do tipo "se-então". Embora úteis para padrões previsíveis, esses sistemas apresentam limitações significativas, como rigidez, alto índice de falsos positivos, dificuldade de manutenção e incapacidade de detectar padrões complexos (Chea & Karlin, 2025; Karczewski, 2023).

Diante dessas limitações e do crescimento exponencial dos dados transacionais, o Machine Learning (Aprendizado de Máquina) emerge como a solução mais promissora.

Diferentemente dos sistemas tradicionais, os algoritmos de ML aprendem continuamente com os dados, adaptam-se automaticamente a novas táticas de fraude e identificam

[1] Diners Club International, inicialmente chamada apenas de Diners Club, surgiu em 1950, criada por Frank X. McNamara, Ralph Schneider e Matty Simmons. Em 1981, a empresa foi adquirida pelo Citibank. Seu destaque histórico se deve ao fato de ter sido a pioneira mundial entre as empresas dedicadas exclusivamente à emissão de cartões de crédito.

padrões complexos que escapariam aos sistemas convencionais (Chen, Yisong et al, 2025).

Diferentemente dos sistemas tradicionais, os algoritmos de ML aprendem continuamente com os dados, adaptam-se automaticamente a novas táticas de fraude e identificam padrões complexos que escapariam aos sistemas convencionais (Chen, Yisong et al, 2025).

Estudos recentes demonstram que métodos de ensemble learning, como XGBoost, LightGBM e CatBoost, têm apresentado resultados superiores na detecção de transações fraudulentas, especialmente quando combinados com técnicas adequadas de balanceamento de dados e estratégias de Stacking Ensemble (Mienye & Sun, 2023; Theodorakopoulos, L, et al., 2025). Esses algoritmos de Gradient Boosting são conhecidos por sua capacidade de lidar com dados desbalanceados, alta dimensionalidade e não linearidade, além de oferecerem robustez contra *overfitting.*

Apesar dos avanços na aplicação de ML para detecção de fraudes, existe uma lacuna importante na literatura: a maioria dos estudos concentra-se em conjuntos de dados moderados executados em ambientes de computação única, não abordando adequadamente os desafios de escalabilidade e eficiência computacional que surgem ao processar volumes massivos de dados em tempo real.

Além disso, o *trade-off*[2] entre desempenho preditivo (acurácia, precisão, *recall, F1-score*) e eficiência computacional (tempo de treinamento e inferência) é pouco explorado no contexto de plataformas de processamento distribuído, como o Databricks (Zaharia et al., 2016). Também há uma carência de estudos que avaliem o Stacking Ensemble como estratégia para combinar modelos com um Meta-Learner (como Regressão Logística) para otimizar a detecção de fraudes em ambientes de Big Data.

Diante desse contexto, este trabalho busca responder à seguinte pergunta de pesquisa: "Qual algoritmo supervisionado de Machine Learning apresenta o melhor equilíbrio entre desempenho preditivo e eficiência computacional para a detecção de fraudes de cartão de crédito em um ambiente de Big Data processado na plataforma Databricks?"

O objetivo geral é investigar e comparar o desempenho preditivo e a eficiência computacional de diferentes algoritmos de Machine Learning para a detecção de fraudes

[2] Trade-off descreve uma situação de escolha em que se deve renunciar a um benefício para obter outro, já que não é possível ter todas as vantagens simultaneamente. É um balanceamento entre fatores concorrentes.

em cartão de crédito em um ambiente de Big Data utilizando a plataforma Databricks, na modalidade Severless.

Para isso, os objetivos específicos são: (1) Implementar e avaliar individualmente o desempenho preditivo e a eficiência computacional de diferentes famílias de algoritmos: modelos lineares (Regressão Logística), ensembles baseados em árvores (Random Forest), modelos de Gradient Boosting (XGBoost, LightGBM, CatBoost), e redes neurais (MLP Classifier) na detecção de fraudes em cartões de crédito; (2) Desenvolver e implementar um Stacking Ensemble utilizando as probabilidades dos melhores modelos como modelos de base como *features* para um Meta-Learner de Regressão Logística, visando aprimorar a acurácia e a calibração das previsões e (3) Comparar os resultados de todos os modelos (individuais e Stacking Ensemble), utilizando métricas como de desempenho preditivo e eficiência computacional.

As hipóteses que guiam este projeto são: (1) Os modelos de Gradient Boosting (LightGBM, XGBoost e CatBoost) apresentarão o melhor desempenho preditivo individual na detecção de fraudes, superando significativamente os modelos lineares e de árvores tradicionais devido à sua capacidade inerente de capturar relações não-lineares e complexas entre as features e de lidar com outliers. (2) O modelo de Stacking Ensemble, que utiliza as probabilidades de saída dos melhores algoritmos de Gradient Boosting como features para um Meta-Learner de Regressão Logística, superará o desempenho de qualquer modelo individual, alcançando um equilíbrio superior entre a captura de fraudes (Recall) e a precisão operacional (Precisão), resultando no maior F1-Score e em probabilidades melhor calibradas e (3) Embora os algoritmos de Gradient Boosting e o Stacking Ensemble demandem maior custo computacional, o ambiente do Databricks proporcionará ganhos de eficiência significativos.

Este estudo se enquadra em uma pesquisa experimental, quantitativa e descritiva, fundamentada no método científico aplicado à Ciência da Computação. Seguindo a classificação de Wazlawick (2009), adotamos um modelo investigativo voltado à apresentação e validação empírica de uma solução considerada superior, por meio de testes padronizados e métricas amplamente aceitas pela comunidade científica. Essa abordagem acompanha as práticas na área de detecção de fraudes financeiras (Dal Pozzolo et al., 2015; Carcillo et al., 2018), que privilegiam a validação experimental com bases de dados *benchmark*. Dessa forma, a metodologia envolve a manipulação controlada de variáveis independentes para avaliar objetivamente o desempenho preditivo

e a eficiência computacional dos modelos analisados.

Como base de dados, será utilizado o *dataset* público Credit Card Fraud Detection (Kaggle), contendo 284.807 transações com apenas 0,172% de fraudes. Esse *dataset* é amplamente adotado em pesquisas recentes (Dal Pozzolo et al., 2023; Carcillo et al., 2024) por refletir as características típicas de problemas reais de fraude.

O desenvolvimento do projeto seguirá um *workflow* de Machine Learning *end-to-end*, conforme o *framework* proposto por Géron (2019), adaptado aos objetivos desta pesquisa. Esse processo abrange as etapas de compreensão do problema e dos dados, preparação dos dados, seleção e treinamento dos modelos, ajuste fino (*fine-tuning*) e avaliação e apresentação da solução final.

A relevância científica deste estudo reside em preencher a lacuna identificada, oferecendo *insights* sobre a escolha de modelos para problemas de classificação em larga escala considerando tanto métricas preditivas quanto eficiência computacional. Do ponto de vista social e econômico, modelos mais precisos e eficientes podem gerar impactos econômicos substanciais, reduzindo perdas financeiras e melhorando a confiança do consumidor na economia digital. A contribuição final visa um futuro mais seguro para todos os integrantes dessa rede, desde grandes instituições financeiras até pequenas empresas com recursos limitados para responder a ameaças.

2 REFERENCIAL TEÓRICO

Esta seção aborda os conceitos fundamentais que sustentam este trabalho, desde a natureza das fraudes financeiras até as técnicas de Machine Learning e as plataformas de Big Data utilizadas para combatê-las.

2.1 Fraudes em Cartão de Crédito

2.1.1 Contextualização e evolução das fraudes financeiras

A digitalização dos meios de pagamento revolucionou as transações financeiras globais, mas simultaneamente criou um ambiente propício para atividades fraudulentas cada vez mais sofisticadas. Desde a introdução do primeiro cartão de crédito pelo Diners Club em 1950, o mercado de pagamentos eletrônicos expandiu-se exponencialmente (Chen, Yisong et al., 2025).

As fraudes em cartões de crédito manifestam-se através de diversas modalidades, incluindo transações CNP[3] (*Card-Not-Present*), *skimming* de dados magnéticos[4], *phishing* eletrônico[5] e engenharia social[6]. A evolução tecnológica não apenas facilitou transações legítimas, mas também forneceu aos fraudadores ferramentas mais avançadas para explorar vulnerabilidades sistêmicas (Karczewski, 2023).

As perdas globais com fraudes em cartões de crédito e débito atingiram US$ 34,36 bilhões em 2022 (WalletHub, 2024), representando não apenas prejuízos financeiros diretos, mas também custos operacionais relacionados à investigação, reversão de transações (*chargebacks*[7]) e perda de confiança do consumidor.

2.1.2 Desafios na detecção de fraudes

A detecção eficaz de fraudes enfrenta desafios multidimensionais que transcendem a mera identificação de padrões anômalos, como:

(1) Desbalanceamento Extremo de Classes: Este é o desafio mais crítico. Em *datasets*

[3] CNP (Card-Not-Present): transações realizadas sem a presença física do cartão, comuns em e-commerce e aplicativos. São mais suscetíveis a fraude porque não há autenticação física do cartão.
[4] Skimming de dados magnéticos: método de fraude no qual criminosos copiam informações da tarja magnética do cartão usando dispositivos clandestinos instalados em terminais ou caixas eletrônicos.
[5] Phishing eletrônico: técnica de ataque em que o usuário é induzido, por meio de mensagens falsas, a revelar dados sensíveis como senhas, informações bancárias ou códigos de segurança.
[6] Engenharia social: conjunto de estratégias baseadas em manipulação psicológica para obter informações confidenciais ou persuadir a vítima a realizar ações que facilitem fraudes ou ataques.
[7] Chargeback(s) é o estorno de uma transação contestada pelo cliente junto à operadora do cartão, geralmente associado a fraudes ou desacordos comerciais. Representa um custo financeiro e operacional significativo para empresas que processam pagamentos.

reais de transações financeiras, fraudes representam tipicamente menos de 1% do volume total de transações (Dal Pozzolo et al., 2023). No *dataset* utilizado neste estudo, a proporção é de apenas 0,172%. Esse desbalanceamento cria dificuldades metodológicas significativas, pois algoritmos de classificação tradicionais tendem a apresentar um forte viés em favor da classe majoritária (transações legítimas), otimizando a acurácia geral em detrimento da capacidade de identificar a classe minoritária (fraudes).

(2) Adaptabilidade dos Fraudadores (*Concept Drift*): Os métodos de fraude evoluem continuamente em resposta às medidas de segurança implementadas. Um padrão de fraude detectado hoje pode se tornar obsoleto amanhã. Isso é conhecido como *concept drift,* exigindo sistemas de detecção que aprendam e se adaptem em tempo real.

(3) Volume e Velocidade (Big Data): Sistemas de pagamento processam milhões de transações diariamente. Os sistemas de detecção devem analisar esses dados (Volume) em milissegundos (Velocidade) para não impactar negativamente a experiência do usuário, impondo requisitos computacionais rigorosos (Zaharia et al., 2016).

(4) *Trade-off*[2] entre Falsos Positivos e Falsos Negativos: Falsos positivos acontecem quando transações legítimas são bloqueadas incorretamente, gerando atrito com clientes legítimos, insatisfação e custos operacionais de investigação manual. Em contrapartida, os falsos negativos são quando fraudes reais não são detectadas, resultando em perdas financeiras diretas e danos à reputação da instituição.

(5) Falsos Positivos: Transações legítimas bloqueadas incorretamente. Isso gera atrito com clientes legítimos, insatisfação e custos operacionais de investigação manual.

A otimização deste *trade-off*[2] constitui o objetivo central de qualquer sistema de detecção de fraudes (Chea & Karklin, 2025).

2.2 Machine Learning e Detecção de Fraudes

2.2.1 Fundamentos de Machine Learning

Machine Learning (ML), ou Aprendizado de Máquina, representa um paradigma computacional onde sistemas aprendem padrões a partir de dados históricos, sem serem explicitamente programados para cada cenário específico. Géron (2019) define ML como "a ciência (e arte) de programar computadores para que eles possam aprender com dados", distinguindo-o fundamentalmente de sistemas baseados em regras fixas.

No contexto de detecção de fraudes, o aprendizado supervisionado é predominante, onde modelos são treinados com dados rotulados (transações previamente classificadas como fraudulentas ou legítimas) para identificar padrões discriminatórios. O *workflow* típico de projetos de ML *end-to-end*, conforme proposto por Géron (2019), compreende as seguintes etapas:

(1) Compreensão do problema e dos dados: Definição clara dos objetivos de negócio e exploração inicial dos dados disponíveis;
(2) Preparação dos dados: Limpeza, transformação e engenharia de features para maximizar o poder preditivo do modelo;
(3) Seleção e treinamento de modelos: Experimentação com diferentes algoritmos e técnicas de validação cruzada;
(4) Ajuste fino (*fine-tuning*): Otimização de hiperparâmetros e seleção do melhor modelo;
(5) Apresentação da solução: Documentação e comunicação dos resultados;
(6) Lançamento, monitoramento e manutenção: *Deploy* do sistema e monitoramento contínuo de performance.

2.2.2 Sistemas tradicionais vs. Machine Learning

Historicamente, instituições financeiras dependiam de sistemas de detecção baseados em regras (*rule-based systems*), operando com lógica condicional do tipo "se-então". Por exemplo: "SE valor da transação > $5.000 E país de origem ≠ país de residência, ENTÃO sinalizar como suspeita". Embora intuitivos e interpretáveis, esses sistemas apresentam limitações críticas (Chea & Karklin, 2025; Karczewski, 2023), como:

(1) Rigidez: Incapazes de se adaptar automaticamente a novos padrões de fraude sem intervenção humana manual para atualização de regras;
(2) Alto índice de falsos positivos: Regras excessivamente conservadoras bloqueiam transações legítimas, gerando insatisfação do cliente;
(3) Dificuldade de manutenção: À medida que o número de regras cresce (frequentemente centenas ou milhares), o sistema torna-se complexo, contraditório e difícil de auditar;
(4) Incapacidade de detectar padrões complexos: Fraudadores exploram interações multidimensionais que escapam a regras simples;

Em contraste, algoritmos de Machine Learning oferecem vantagens decisivas (Chen, Yisong et al. 2025):

(1) Aprendizado contínuo: Modelos podem ser retreinados periodicamente com novos dados, adaptando-se automaticamente à evolução das táticas de fraude;
(2) Detecção de padrões complexos: Algoritmos como Gradient Boosting capturam relações não-lineares e interações entre múltiplas variáveis que seriam impossíveis de codificar manualmente;
(3) Otimização de *trade-off*[2]*:* ML permite ajustar dinamicamente o equilíbrio entre falsos positivos e falsos negativos conforme objetivos de negócio.
(4) Escalabilidade: Uma vez treinados, modelos de ML processam milhões de transações com latência mínima.

2.3 Modelos de Machine Learning para Detecção de Fraudes

Embora inúmeros algoritmos possam ser aplicados, este estudo foca em uma seleção que representa diferentes níveis de complexidade e abordagens, todos relevantes para o problema de fraude.

2.3.1 Modelos Lineares: Regressão Logística (Logistic Regression)

A Regressão Logística é frequentemente utilizada como um *benchmark* robusto, pois, apesar de sua simplicidade linear, ela oferece vantagens significativas:

(1) Interpretabilidade: É relativamente fácil entender quais features mais contribuem para uma previsão de fraude;
(2) Eficiência: É computacionalmente leve e extremamente rápida para treinar e, principalmente, para realizar inferências (previsões), tornando-a ideal para ambientes de baixa latência;
(3) Saída de Probabilidade: Fornece uma probabilidade (entre 0 e 1) de fraude, que é bem calibrada e útil para definir thresholds de risco;
(4) Sua principal desvantagem é a incapacidade de capturar relações não-lineares complexas entre as features, o que é comum em padrões de fraude sofisticados.

2.3.2 Modelos baseados em Gradient Boosting (GBM)

Os modelos de Gradient Boosting são uma classe de algoritmos de *ensemble* que constroem modelos de forma sequencial e aditiva. Eles se destacam por corrigir os erros dos modelos anteriores, resultando em classificadores de alta precisão. Este estudo foca nos três algoritmos de GBM mais avançados:

(1) XGBoost (Extreme Gradient Boosting): Desenvolvido por Chen & Guestrin

(2016), o XGBoost revolucionou o ML em dados tabulares. Ele otimiza o algoritmo de *boosting* tradicional com regularização (L1 e L2) para prevenir overfitting, processamento paralelo eficiente (a nível de construção de árvore) e capacidade de lidar com dados faltantes.

(2) LightGBM (Light Gradient Boosting Machine): Proposto por Ke et al. (2017), o LightGBM foca na velocidade e eficiência. Ele utiliza uma técnica de crescimento de árvore *leaf-wise* (em vez de *level-wise* como o XGBoost), o que leva a convergência mais rápida. Além disso, emprega técnicas como EFB (*Exclusive Feature Bundling*) e GOSS (*Gradient-based One-Side Sampling*) para reduzir o volume de dados e features durante o treinamento.

(3) CatBoost (Categorical Boosting): Desenvolvido por Prokhorenkova et al. (2018), o CatBoost é especializado em lidar com features categóricas de forma nativa e eficiente. Sua principal inovação é o uso de ordered boosting e randomized permutations para combater o prediction shift (um tipo de overfitting) e fornecer estimativas mais robustas.

Esses três modelos são considerados o estado da arte para problemas de classificação em dados tabulares, como a detecção de fraudes (Grinsztajn et al., 2022).

2.4 Ensemble Learning e Stacking

Ensemble Learning (Aprendizado de Conjunto) é uma técnica que combina as previsões de múltiplos modelos de ML para obter um desempenho mais robusto.

O Stacking (*Stacked Generalization*) é uma forma avançada de ensemble. Em vez de simplesmente tirar a média ou votar nas previsões (como em bagging ou voting), o Stacking utiliza um processo de dois níveis:

(1) Nível 0 (Base-Learners): Diversos modelos (ex: XGBoost, RandomForest, MLP) são treinados no conjunto de dados de treino;

(2) Nível 1 (Meta-Learner): Um novo modelo, chamado de meta-learner, é treinado e as *features* de entrada para este meta-learner não são os dados originais, mas sim as previsões (*outputs*) dos modelos do Nível 0.

A intuição é que o meta-learner aprende a "confiar" mais em certos modelos de base dependendo do tipo de instância, combinando suas forças e corrigindo suas fraquezas. Por exemplo, ele pode aprender que o XGBoost é bom em certos padrões, enquanto o MLP é melhor em outros. O uso de uma Regressão Logística como meta-learner é comum, pois ela é eficaz em encontrar a ponderação linear ideal das previsões dos

modelos de base (Liu et al., 2025; Caruana & Niculescu-Mizil 2006). Em contextos de alto impacto, como detecção de fraude, a interpretabilidade do meta-learner é crucial (Rudin, 2019), o que torna a regressão logística uma escolha ideal para essas aplicações.

2.5 Métricas de Avaliação para Dados Desbalanceados

Em problemas com classes desbalanceadas, a Acurácia (proporção de acertos) é uma métrica enganosa, pois um modelo que prevê "não-fraude" para todas as transações em nosso *dataset* alcançaria 99,828% de acurácia, mas seria completamente inútil, pois não detectaria nenhuma fraude. Portanto, métricas mais adequadas são necessárias:

(1) *Precision* (Precisão): De todas as transações que o modelo previu como fraude, quantas eram realmente fraudes? Foca em minimizar os Falsos Positivos. *Precision = TP / (TP + FP);*

(2) *Recall* (Revocação ou Sensibilidade): De todas as fraudes reais que ocorreram, quantas o modelo conseguiu detectar? Foca em minimizar os Falsos Negativos. *Recall = TP / (TP + FN);*

(3) F1-Score: É a média harmônica entre *Precision* e *Recall*. É uma excelente métrica única para avaliar o *trade-off* entre os dois. É particularmente útil quando os Falsos Positivos e Falsos Negativos têm custos similares. *F1-Score = 2 * (Precision * Recall) / (Precision + Recall)*

Curva AUC-PR (*Area Under the Precision-Recall Curve*): Em dados altamente desbalanceados, a curva PR é mais informativa que a curva ROC. Ela plota a *Precision* vs. *Recall* em todos os *thresholds* de decisão. Um modelo perfeito teria um AUC-PR de 1.0, pois essa métrica avalia o desempenho do modelo independentemente do *threshold* de classificação escolhido (Saito & Rehmsmeier, 2015).

2.6 Plataformas de Big Data e Databricks

O processamento de bilhões de transações diárias excede a capacidade de uma única máquina. Isso exige plataformas de processamento distribuído, como o Apache Spark™ (Zaharia et al., 2016). Spark é um motor de processamento unificado para *Big Data* que distribui a carga de trabalho por um *cluster* de computadores, permitindo escalabilidade horizontal massiva.

Databricks é uma plataforma de análise de dados baseada em nuvem, fundada pelos criadores originais do Apache Spark™. Ela oferece um ambiente otimizado para engenharia de dados, ciência de dados e Machine Learning em larga escala. O ambiente

serverless do Databricks, utilizado neste trabalho, abstrai ainda mais a infraestrutura, permitindo que o foco seja totalmente no *workflow* de dados, enquanto a plataforma gerencia automaticamente o provisionamento e a escalabilidade dos clusters.

Alguns componentes chave da plataforma utilizados neste estudo incluem:

(1) Unity Catalog: Um serviço de governança de dados unificado para o lakehouse, permitindo o gerenciamento centralizado de dados, metadados e permissões;
(2) MLflow: Uma plataforma de código aberto (integrada ao Databricks) para gerenciar o ciclo de vida do ML *end-to-end*. O MLflow foi usado extensivamente para;
(3) Rastreamento (*Tracking*): Registrar experimentos, parâmetros, métricas e artefatos de modelo;
(4) Modelos (*Models*): Gerenciar e versionar modelos de produção em um repositório central;
(5) Registro (*Registry*): Promover modelos de "desenvolvimento" para "produção" com governança.

3 TRABALHOS RELACIONADOS

A detecção de fraudes por meio de técnicas de Machine Learning tem sido amplamente investigada, por isso, essa seção revisa os trabalhos mais relevantes que fundamentam este estudo, organizados em três tópicos principais que orientam este estudo: algoritmos de Gradient Boosting, técnicas de ensemble learning e stacking, e aplicações em ambientes de Big Data.

Quanto à eficácia dos algoritmos de Gradient Boosting na detecção de fraudes é amplamente reconhecida. Chen e Guestrin (2016) apresentaram o XGBoost, um sistema escalável de boosting que se destaca por sua capacidade de lidar com dados desbalanceados e alcançar alta acurácia. O estudo de Grinsztajn et al. (2022) comparou XGBoost, LightGBM e CatBoost em 45 conjuntos de dados, incluindo detecção de fraudes, e concluiu que XGBoost e CatBoost apresentam melhor desempenho médio em AUC, enquanto LightGBM é o mais rápido em termos de treinamento e inferência. Ke et al. (2017) introduziram o LightGBM, otimizado para eficiência e velocidade, enquanto Prokhorenkova et al. (2018) desenvolveram o CatBoost, que oferece robustez a features categóricas e estabilidade em datasets com ruído. Johnson et al. (2025) reforçaram a importância de avaliar não apenas o desempenho preditivo, mas também a eficiência computacional, mostrando que LightGBM oferece a menor latência, CatBoost a maior estabilidade, e XGBoost o melhor equilíbrio entre acurácia e tempo de resposta.

Além disso, as técnicas de ensemble são essenciais para lidar com o desbalanceamento de classes em fraudes. Liu Z, et al. (2025) desenvolveram um framework que combina stacking ensemble de modelos de Gradient Boosting (XGBoost, LightGBM e CatBoost) com técnicas de Explainable AI (XAI), utilizando Regressão Logística como meta-learner devido à sua capacidade de ponderar de forma eficiente e interpretável as previsões dos modelos base, alcançando AUC-ROC de 0,9983. Cano e Krawczyk (2020) analisaram arquiteturas de ensemble para dados em fluxo, enquanto Dal Pozzolo et al. (2023) discutiram a calibração de probabilidades em classificação desbalanceada, validando a importância de métodos de stacking para melhorar a generalização dos modelos.

Complementando o desempenho preditivo, a escalabilidade é crítica para sistemas de detecção de fraudes em tempo real. Zaharia et al. (2016) apresentaram o Apache Spark™, como um *framework* unificado para processamento distribuído, destacando sua arquitetura *in-memory* e a biblioteca MLlib. Penchikala (2016) detalhou a aplicação do Spark MLlib para machine learning em larga escala, enquanto Theodorakopoulos et al. (2025) demonstraram a integração de PySpark, XGBoost e CatBoost para detecção de

fraudes escalável. Armbrust et al. (2015) introduziram o Spark SQL, que facilita a integração de *pipelines* de dados e modelos de ML.

4 MÉTODO

Este estudo adota uma metodologia de pesquisa experimental comparativa com abordagem empírica quantitativa. De acordo com a classificação proposta por Wazlawick (2009), o trabalho se enquadra na categoria de pesquisa que busca apresentar algo reconhecidamente melhor através de testes padronizados e métricas aceitas pela comunidade científica.

Portanto, classificamos a pesquisa como experimental porque envolve a manipulação sistemática de variáveis independentes, incluindo algoritmos de machine learning, configurações de hiperparâmetros e volumes de dados, em ambiente controlado (Databricks), para observar seus efeitos sobre variáveis dependentes mensuráveis relacionadas ao desempenho preditivo e à eficiência computacional. Além disso, o caráter empírico se evidencia pela coleta e análise de dados objetivos através de métricas padronizadas como ROC-AUC, Average Precision, F1-Score, bem como métricas de eficiência computacional (tempo de inferência, uso de memória e CPU), permitindo a verificação independente dos resultados por outros pesquisadores (Wazlawick, 2009).

Quanto à sua natureza comparativa, se manifesta na avaliação sistemática de algoritmos distintos sob múltiplas perspectivas, utilizando o *dataset* público *Credit Card Fraud Detection* como *benchmark* reconhecido pela comunidade científica (Dal Pozzolo et al., 2015). Essa abordagem metodológica está alinhada com as práticas correntes na área de detecção de fraudes financeiras (Carcillo et al., 2018), que priorizam a validação experimental com bases de dados padronizadas e a comparação objetiva de desempenho entre diferentes técnicas. O método adotado segue o paradigma do avanço do estado da arte através de evidências empíricas reproduzíveis, característica fundamental da pesquisa em Ciência da Computação (Wazlawick, 2009).

Quanto ao ambiente de desenvolvimento, projeto foi desenvolvido no ambiente *serverless* do Databricks, utilizando PySpark e Pandas. Buscamos aproveitar ao máximo os recursos disponíveis nesse ambiente de Big Data, como o MLflow, as UDFs (U*ser Defined Functions*) e o Unity Catalog. O MLflow é uma ferramenta voltada para o gerenciamento completo do ciclo de vida de modelos de machine learning, permitindo registrar experimentos, acompanhar métricas de desempenho, versionar modelos e garantir a

reprodutibilidade dos resultados. Já as UDFs foram empregadas para automatizar tarefas mais complexas e/ou repetitivas, enquanto o Unity Catalog foi utilizado para armazenar tanto a base de dados original (creditcard.csv) quanto as bases de treino e teste geradas ao longo do desenvolvimento do projeto.

Quanto às etapas de desenvolvimento, o *workflow* proposto por Géron (2019) foi adaptado às necessidades deste projeto, estruturando-se nas seguintes fases: compreensão do dataset; divisão entre conjuntos de treino e teste; análise exploratória dos dados; preparação e transformação das variáveis; seleção e treinamento dos modelos; ajuste fino dos hiperparâmetros; avaliação das métricas de desempenho preditivo e de eficiência computacional para todos os algoritmos testados; e, por fim, apresentação dos resultados obtidos.

4.1 COMPREENDENDO O PANORAMA GERAL

4.1.1 Compreensão do Dataset

Como mencionado anteriormente, para o desenvolvimento deste trabalho, utilizamos o *dataset* público *Credit Card Fraud Detection* (creditcard.csv). O conjunto de dados contém transações realizadas por portadores de cartões de crédito europeus no mês de setembro de 2013, coletadas e analisadas durante uma colaboração de pesquisa entre a empresa Worldline e o Machine Learning Group da Université Libre de Bruxelles (ULB), voltada à mineração de dados em larga escala e à detecção de fraudes financeiras (*Big Data Mining and Fraud Detection*).

O *dataset* contém 284.807 transações, onde 492 são classificadas como fraudulentas, apenas 0,172% do total, representando um forte desbalanceamento entre as classes. As *features* são exclusivamente numéricas e resultam de uma transformação por PCA (*Principal Component Analysis*), aplicada com o intuito de preservar a confidencialidade das informações originais dos clientes.

As *features* V1, V2, …, V28 correspondem aos componentes principais obtidos pela PCA. As únicas variáveis que não foram transformadas são Time e Amount.

Time representa o tempo, em segundos, entre cada transação e a primeira transação registrada no conjunto de dados. Amount é o valor monetário da transação.

A variável Class é a nossa variável alvo, onde o valor 1 indica transações fraudulentas e 0 indica transações legítimas.

4.1.2 Visão Geral do Problema

Antes de começar qualquer etapa de modelagem, foi importante entender bem o problema de negócio envolvido. Neste estudo, buscamos primeiro compreender o impacto das fraudes em cartões de crédito não só para as instituições financeiras, mas também para empresas de diferentes portes e para os próprios consumidores. Além das perdas financeiras diretas, percebi que esse tipo de crime também afeta a confiança das pessoas nos meios de pagamento digitais, o que reforça a relevância do tema.

Com o avanço da tecnologia, a detecção de fraudes evoluiu bastante. No início, eram usados sistemas baseados em regras fixas, mas hoje já contamos com algoritmos de Machine Learning capazes de identificar padrões complexos de comportamento. Ao mesmo tempo, o crescimento do volume de dados fez com que os ambientes de Big Data se tornassem cada vez mais comuns, pois milhões de transações são processadas diariamente, e isso exige soluções que unam precisão e eficiência computacional.

Pensando nesse cenário, realizamos todos os experimentos deste trabalho na plataforma Databricks, uma solução de inteligência de dados construída sobre uma arquitetura *lakehouse* aberta. Ela reúne em um único ambiente o armazenamento, o processamento e a governança de dados em larga escala.

Databricks surgiu a partir de pesquisas acadêmicas e projetos de código aberto e foi fundada em 2013 pelos criadores do Apache Spark™, Delta Lake, MLflow e Unity Catalog. Hoje, mais de 15 mil empresas ao redor do mundo, incluindo nomes como Block, Comcast, Rivian, Shell e mais de 60% das companhias da Fortune 500 utilizam a plataforma para processamento e análise de dados com inteligência artificial (Databricks, 2024).

Entre as plataformas de processamento distribuído disponíveis atualmente, o Databricks se destaca pela boa relação entre desempenho e custo, impulsionada pelo mecanismo de execução do Apache Spark™, conhecido pela eficiência no tratamento de grandes volumes de dados. Além disso, sua escalabilidade e flexibilidade tornam o uso acessível não apenas para grandes instituições financeiras, mas também para empresas menores, pesquisadores e desenvolvedores interessados em explorar o potencial do Big Data e da Inteligência Artificial.

Com base nisso, o objetivo deste trabalho é comparar algoritmos de Machine Learning supervisionado em relação ao desempenho preditivo e à eficiência computacional na

detecção de fraudes em transações de cartão de crédito, dentro de um ambiente de Big Data configurado no Databricks.

O aprendizado adotado é do tipo supervisionado, pois o conjunto de dados utilizado já possui rótulos definidos, indicando se cada transação é legítima ou fraudulenta. Estamos lidando com um problema clássico de classificação binária.

4.2 Ambiente de Desenvolvimento

O projeto foi integralmente desenvolvido na plataforma Databricks, utilizando um cluster *serverless* de Ciência de Dados e Engenharia. Esta abordagem permitiu o gerenciamento automático da infraestrutura de computação, garantindo escalabilidade sob demanda. Os principais componentes tecnológicos utilizados foram:

(1) Databricks Notebooks: Para o desenvolvimento iterativo do código em PySpark e Python.
(2) Apache Spark™ (via PySpark): Para a ingestão inicial, transformação e divisão dos dados em larga escala.
(3) Pandas: Para a manipulação de dados em memória nas etapas de Análise Exploratória (EDA) e treinamento de modelos scikit-learn.
(4) Unity Catalog: Utilizado como o sistema central de governança, armazenando o dataset original e todas as tabelas intermediárias (treino, teste, features de engenharia, etc.).
(5) MLflow: Empregado para o gerenciamento completo do ciclo de vida do ML. Todos os experimentos, hiperparâmetros, métricas de desempenho (preditivo e computacional) e artefatos de modelo (incluindo os modelos otimizados e o ensemble final) foram registrados no MLflow, garantindo total reprodutibilidade e facilitando a comparação.
(6) Rastreamento (*Tracking*): Registrar experimentos, parâmetros, métricas e artefatos de modelo;
(7) Modelos (*Models*): Gerenciar e versionar modelos de produção em um repositório central;

4.3 Obtendo os Dados

O primeiro passo foi a criação de um *workspace* no Databricks, após isso, o passo seguinte foi trazer os dados para o ambiente do Databricks, utilizando o Unity Catalog para armazenar o *dataset* original proveniente do Kaggle.

4.3.1 Conhecendo a Estrutura dos Dados

O *dataset*, composto por 284.807 transações e 31 colunas, foi carregado e inspecionado para verificar sua estrutura e integridade. Todas as variáveis são numéricas, e a análise inicial confirmou a ausência de valores nulos. A variável-alvo, *Class*, apresenta um desbalanceamento extremo: apenas 0,17% das transações são fraudulentas (492 casos), enquanto 99,83% são legítimas (284.315 casos), o que demanda técnicas específicas para evitar viés no modelo.

Na sequência, foram analisadas estatísticas básicas das nossas *features.* O método *describe()* já nos forneceu informações importantes sobre o conjunto de dados, através da contagem, média, desvio padrão, valores mínimos e máximos. Através dele, foi possível observar que não há dados nulos, pois o valor de *count* é o mesmo para todas as colunas. As médias das variáveis V1 a V28 são próximas de zero, o que confirma que os dados foram padronizados pois passaram por PCA. Analisando a *feature* Time, confirmamos que o conjunto de dados representa 284.807 transações realizadas ao longo de dois dias consecutivos (aproximadamente 172.792 segundos). A média da *feature* Amount é de aproximadamente 88,35, enquanto o desvio padrão é de 250,12, quase três vezes maior que a média, demonstrando uma grande variação entre os valores. O valor mínimo encontrado é 0,0, possivelmente indicando transações sem custo, como testes ou autorizações, e o valor máximo chega a 25.691,16, muito superior à média, evidenciando a presença de transações de valores muito altos, possíveis *outliers*. O desbalanceamento da variável Class também pode ser observado no describe(). Como essa coluna contém apenas valores 0 (legítimas) e 1 (fraudulentas), a média representa diretamente a proporção de fraudes no *dataset*, e, no nosso caso, a média da coluna Class é 0,001727, ou seja, representando os 0,1727% de fraudes.

Na sequência, essas análises iniciais serão aprofundadas por meio de visualizações exploratórias, com o objetivo de compreender melhor o comportamento das variáveis e as possíveis relações entre elas dentro do conjunto de dados.

4.4 Criando um Conjunto de Treino e Teste

Ir muito afundo nas explorações dos dados antes de separar o conjunto de teste pode colaborar para um *data snooping bias,* um viés de exploração de dados (Géron, 2019). Por isso, já adicionamos essa divisão ainda na nossa etapa de análise exploratória dos dados. Para garantir reprodutibilidade caso o *dataset* seja atualizado com novos registros, foi adotada uma abordagem baseada em identificadores únicos para cada transação, e, por meio do cálculo de um hash sobre esses identificadores, cada instância é alocada ao conjunto de teste (20%) ou de treino (80%), garantindo que a divisão entre os conjuntos permaneça consistente mesmo após a atualização do *dataset*, evitando que uma transação anteriormente usada para treino seja realocada para o teste. A escolha da abordagem estratificada determinística foi apoiada por Géron (2019), e testada com o nosso conjunto de dados, após uma análise entre a abordagem de amostragem aleatória simples *(randomSplit()* simples com *seed* 42) que serve como baseline comparativo, a amostragem estratificada *randomSplit()* com amostragem estratificada por classe, e a amostragem estratificada determinística por meio da criação de uma função (UDF) que um algoritmo mais sofisticado para garantir precisão na distribuição, criando identificadores únicos baseados em todas as colunas da linha, e por meio de um cálculo *hash*, alocada cada transação ao conjunto de treino ou teste. Para escolher qual abordagem seguiríamos, realizamos um teste comparativo através do erro percentual entre a taxa de fraudes nos conjuntos de treino/teste versus a taxa real. O melhor método é selecionado com base no menor erro máximo observado, garantindo que a distribuição da classe alvo seja mais fiel à original em ambas as partições. O teste comparativo obteve os seguintes resultados:

Método	Conjunto	Tamanho	Fraudes	Taxa	Erro %
Aleatório	Treino	227,940	400	0.001755	1.58%
Aleatório	Teste	56,867	92	0.001618	6.35%
Estratificado	Treino	227,948	390	0.001711	0.96%
Estratificado	Teste	56,859	102	0.001794	3.85%
Determinístico	Treino	227,846	394	0.001729	0.10%
Determinístico	Teste	56,961	98	0.001720	0.41%

Figura 1 -Tabela de Análise Comparativa entre Métodos de Amostragem de Treino e Teste

Os resultados demonstram claramente a superioridade da abordagem determinística. Enquanto o método aleatório apresentou um erro máximo de 6,35% no conjunto de teste, inviável para um problema com poucas fraudes, e o estratificado tradicional ainda mostrava 3,85% de erro, a abordagem determinística reduziu o erro máximo para apenas 0,41%, garantindo uma precisão de que o modelo será treinado e avaliado em dados que refletem a distribuição real do problema.

Como resultado, através da divisão pelo método estratificado determinístico, objetivemos um *dataset* de treino com 227.846 transações, onde 227.452 são legítimas e 394 fraudulentas, representando 0.1729% do total, e um *dataset* de treino com 56.961 transações, onde 56863 são transações legítimas e 98 são fraudes, representando 0.1720% do total.

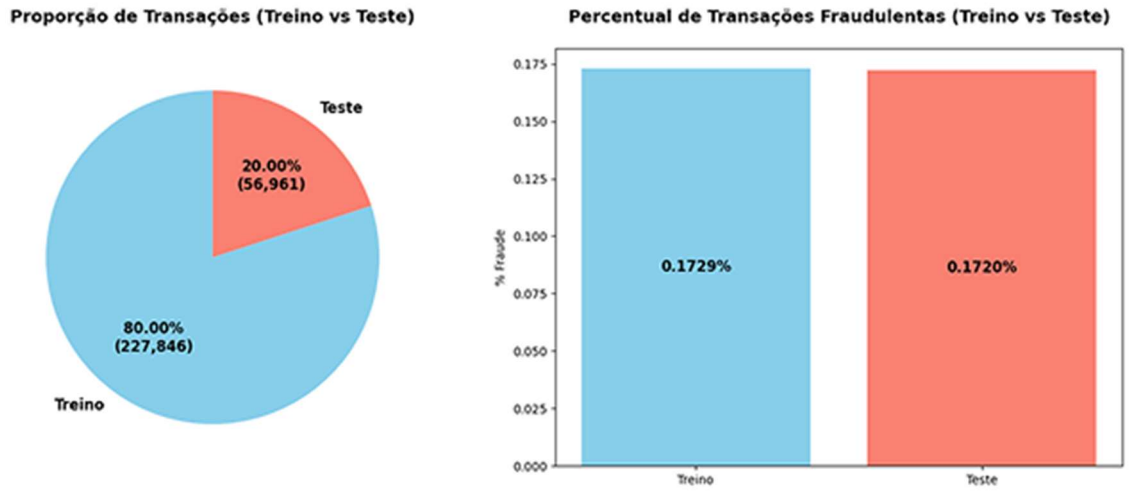

Figura 2 - Proporção de Transações e Fraudes nas Amostras de Treino e Teste

Os conjuntos de treino e teste foram salvos no catálogo do Unity Catalog para serem reutilizados. As proporções de fraudes no treino e no teste são muito próximas, indicando que a divisão foi bem-sucedida e que ambos os conjuntos são representativos do *dataset* original. Após a divisão, seguimos a visualização dos dados para ganhar *insights,* analisando apenas o nosso conjunto de treino.

4.5. Visualizando os Dados para Obter *Insights*

Foram realizadas análises exploratórias dos dados para trazer alguns insights para as próximas fases. A primeira etapa foi analisar as duas variáveis que não passaram por nenhuma transformação de PCA no *dataset* original, *Amount* e *Time*.

Class	Mediana	Média	Desvio Padrão	Coef. Variação
0	22	87.96	248.03	2.82
1	9.82	128.81	267.83	2.08

Figura 3 - Tabela de Estatísticas de *feature Amount*

A análise da *feature amount* revelou diferenças importantes entre transações legítimas e fraudulentas, como: (1) Fraudes apresentam valores médios mais altos e maior variabilidade; (2) A mediana baixa (9,91) indica que a maioria das fraudes ocorre em transações de pequeno valor e (3) A distribuição de fraudes sugere comportamento atípico útil para detecção de anomalias.

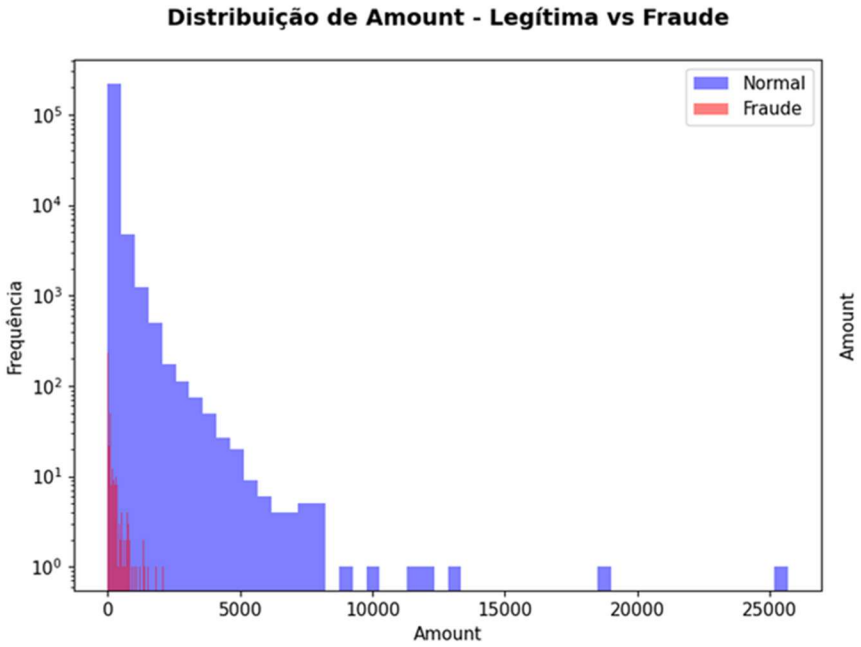

Figura 4 - Distribuição de *Amount* por Transações Legítimas e Fraudes

A *feature Time* apresenta uma distribuição bimodal, refletindo o ciclo de 48 horas correspondente ao período de dois dias consecutivos abrangido pelo conjunto de dados.

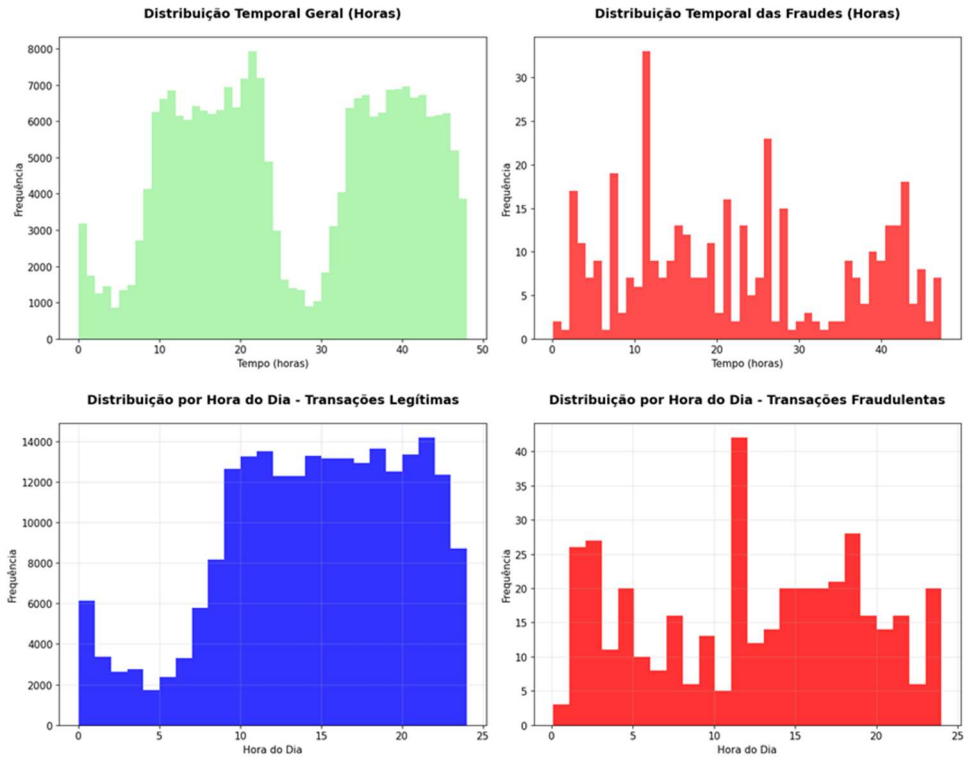

Figura 5 - Distribuição da *feature* Time

Ao converter para hora do dia, pudemos observar que as transações legítimas tendem a seguir um padrão que parece seguir a atividade humana diária, com início das transações no período da manhã, e um volume maior a tarde e à noite, um pouco diferente do padrão observado nas fraudes que tendem a ocorrer de forma esporádica em horários específicos, com a madrugada apresentando a maior taxa de fraudes (~0,50%), 3,85x maior que à noite (~0,13%).

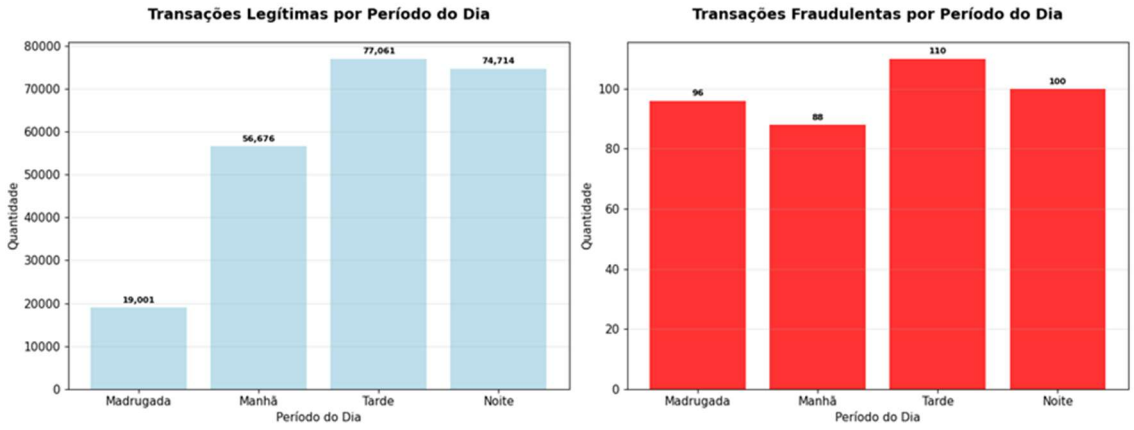

Figura 6 - Transações Legítimas e Fraudulentas por Período do Dia

As 28 *features* (V1 a V28) são resultado de PCA aplicado pelos autores para proteger

informações sensíveis, por isso, tendem a seguir um padrão de variância cumulativa, sendo:

- V1-V10: explicam 64,7% da variância total (padrões dominantes)
- V11-V20: explicam 26,9% da variância (padrões secundários)
- V21-V28: explicam 8,4% da variância residual

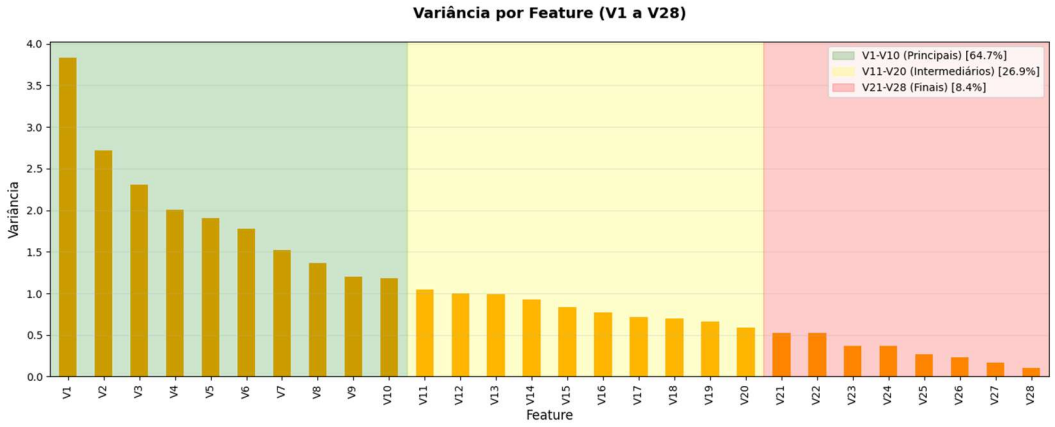

Figura 7 - Variância por *feature* V1 a V28

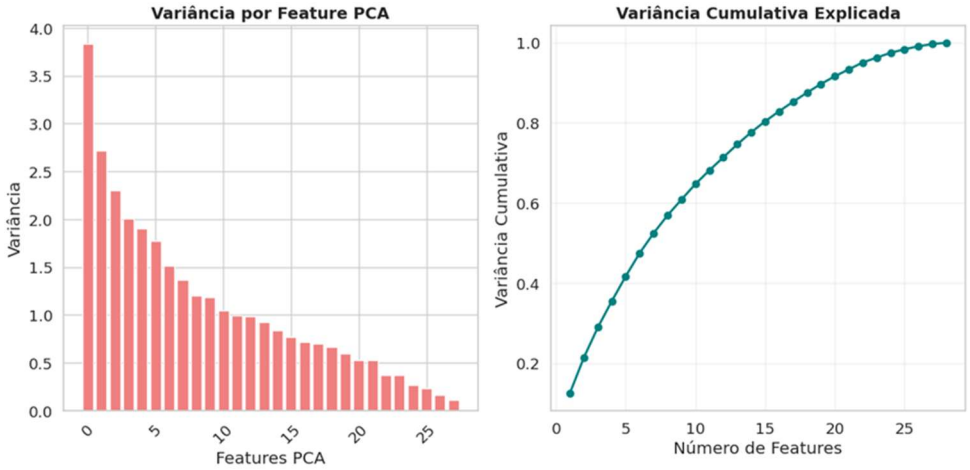

Figura 8 - Variância Cumulativa das features V1 a V28

Além disso, as *features* V1, V2, V3, V4, V7, V10, V11, V12, V14 e V17 apresentaram maior capacidade de separação entre classes, especialmente pela presença de outliers significativos em transações fraudulentas. Algumas interações entre *features* como V12, V14, V3, V10 também demonstraram alto poder discriminativo, capturando padrões não-lineares.

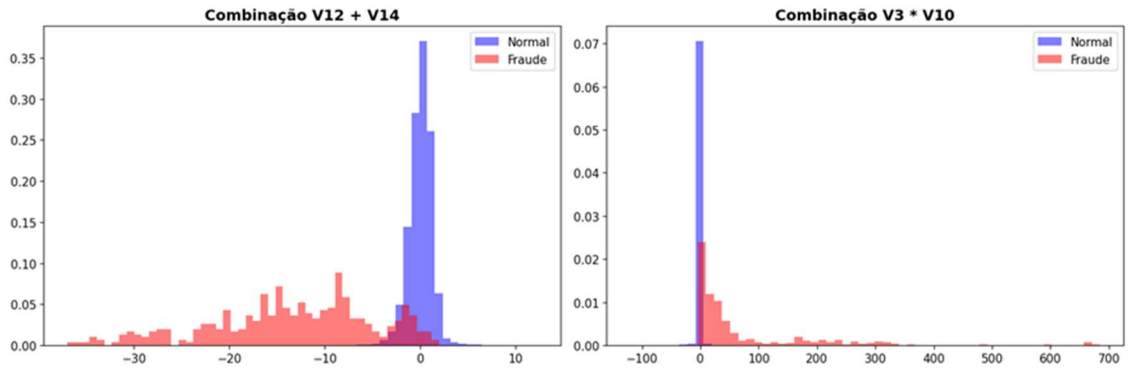

Figura 9 - Análise de Combinações de Features

Com base na análise exploratória, algumas estratégias foram definidas para análise de adoção nos próximos passos, em especial, para a fase de engenharia de features: (1) transformações em *Amount*: aplicar log e raiz quadrada para reduzir assimetria; (2) Features temporais: incluir hora do dia com codificação cíclica (sin/cos) e indicador de período; (3) Interações PCA: criar combinações entre features discriminativas (V12×V14, V3×V10); (4) Magnitude PCA: calcular magnitude vetorial das top features para capturar variância combinada; (5) Detecção de outliers: incorporar indicadores de anomalias em features críticas.

4.6. Preparando os Dados

A análise exploratória inicial revelou a complexidade do problema, destacando o desbalanceamento acentuado entre as classes e a natureza das variáveis transformadas pela Análise de Componentes Principais (PCA). Por isso, as análises obtidas através da EDA orientaram a engenharia de features para a construção de um conjunto de dados mais robusto e informativo. A etapa de preparação dos dados englobou três processos fundamentais para garantir a qualidade e a eficácia do modelo preditivo: (1) Engenharia de features, (2) Balanceamento de classes por meio de pesos, e (3) Pré-processamento com escalonamento robusto.

Na engenharia de *features*, foram criadas variáveis derivadas com base em análise exploratória e correlações identificadas no *dataset:* (1) Para capturar padrões temporais, a coluna Time foi transformada em hour_of_day, e aplicou-se codificação cíclica (hour_sin e hour_cos) para preservar a periodicidade horária, (2) Foram geradas interações entre variáveis com alta correlação, como V12_V14_interaction e V3_V10_interaction, além de combinações não-lineares, como a magnitude vetorial de V1 e V2 (V1_V2_complex), (3) Uma feature estatística avançada, pca_magnitude, foi

calculada para resumir a variância de 12 variáveis originais, (4) Transformações não-lineares, como logaritmo (Amount_log) e raiz quadrada (Amount_sqrt), foram aplicadas à coluna Amount para reduzir assimetria e (5) A detecção de anomalias em V1 (v1_anomaly) também foi incorporada.

O resultado deste processo foi um conjunto final de 22 features selecionadas para o treinamento: *final_features = ['V1', 'V2', 'V3', 'V4', 'V7', 'V10', 'V11', 'V12', 'V14', 'V16', 'V17', 'V18', 'hour_sin', 'hour_cos', 'V12_V14_interaction', 'V3_V10_interaction', 'V1_V2_complex', 'pca_magnitude', 'Amount_log', 'Amount_sqrt', 'amount_to_mean_ratio', 'v1_anomaly']*

O desbalanceamento acentuado entre as classes, sendo: fraudes (classe 1) e transações normais (classe 0), foi abordado por meio do cálculo de pesos de classe, que atribuíram maior importância às amostras minoritárias durante o treinamento. Os pesos, 0.5009 para a classe 0 e 289.1447 para a classe 1, foram adicionados ao conjunto de treinamento como uma coluna adicional (*class_weight*), garantindo que o modelo não fosse enviesado pela predominância de transações não fraudulentas.

Por fim, o pré-processamento incluiu a aplicação do RobustScaler, um método de escalonamento robusto que normaliza as features com base na mediana e no intervalo interquartílico (IQR), minimizando o impacto de outliers. Os conjuntos de treinamento e teste processados foram salvos no Unity Catalog, preservando a integridade e a reprodutibilidade dos resultados para as etapas subsequentes de treinamento e avaliação do modelo.

4.7. Selecionar e Treinar os Modelos

A primeira fase de testes dos modelos de Machine Learning foi conduzida com o objetivo de avaliar o desempenho de diferentes algoritmos na tarefa de detecção de fraudes em transações com cartão de crédito, um problema caracterizado por um desbalanceamento acentuado entre as classes. Para isso, foram selecionados sete modelos com características algorítmicas e computacionais distintas: Regressão Logística, modelo linear altamente interpretável baseado em combinações lineares das features; Decision Tree e Random Forest, modelos baseados em árvores de decisão, sendo o primeiro um modelo único interpretável e o segundo um ensemble com bagging para redução de variância; LightGBM, XGBoost e CatBoost, algoritmos de gradient boosting que constroem ensembles de árvores de forma sequencial, onde cada nova árvore corrige erros das

anteriores; Multilayer Perceptron (MLP), uma rede neural capaz de aprender representações hierárquicas complexas através de camadas densamente conectadas; e Stacking Ensemble, um modelo de meta-learning que combina as predições de modelos de base através de um meta-learner para maximizar a capacidade preditiva. Esta seleção diversificada permite avaliar diferentes paradigmas de aprendizado e comparar *trade-offs*[2] entre performance preditiva e eficiência computacional.

O treinamento dos modelos foi realizado utilizando os dados pré-processados e escalonados, com a aplicação de pesos de classe para mitigar o desbalanceamento. A avaliação do desempenho foi baseada em métricas relevantes, incluindo ROC-AUC, precisão média (*average ou 'avg' precision*), precisão, revocação (*recall*) e *F1-score*. O registro e o monitoramento dos experimentos foram realizados com o MLflow, que permitiu o acompanhamento sistemático das métricas, parâmetros e modelos treinados, garantindo a reprodutibilidade dos resultados.

Os resultados iniciais da primeira fase de testes revelaram desempenhos distintos entre os modelos avaliados.

Modelo	ROC AUC	Avg Precision	Precision	Recal	F1-Score
LogisticRegression	0.976711	0.748399	0.052599	0.867347	0.099183
MLP_Classifier	0.975406	0.744587	0.866667	0.795918	0.829787
CatBoost	0.972523	0.785162	0.766990	0.806122	0.786070
XGBoost	0.951283	0.807184	0.759615	0.806122	0.782178
RandomForest	0.946974	0.801817	0.932432	0.704082	0.802326
LightGBM	0.939870	0.754785	0.804348	0.755102	0.778947
DecisionTree	0.862043	0.547695	0.755319	0.724490	0.739583

Figura 10 - Tabela de Desempenho Preditivo Inicial dos Modelos Testados

A Regressão Logística apresentou o maior ROC-AUC (0.977) e excelente *recall* (0.867), porém com *precision* extremamente baixa (0.053), resultando em F1-score de apenas 0.099, indicando forte tendência a classificar transações normais como fraude. O MLP demonstrou desempenho equilibrado com *F1-score* de 0.830, combinando alta precision (0.867) e recall satisfatório (0.796). Entre os modelos de gradient boosting, o XGBoost obteve o melhor *average precision* (0.807) e *F1-score* competitivo (0.782), enquanto o

CatBoost apresentou melhor equilíbrio entre precision (0.767) e recall (0.806), com *F1-score* de 0.786. O Random Forest destacou-se pela maior precision (0.932), porém com recall comprometido (0.704), resultando em *F1-score* de 0.802. O LightGBM e a Decision Tree apresentaram os desempenhos mais modestos, com F1-scores de 0.779 e 0.740, respectivamente. Esses resultados iniciais evidenciam o *trade-off*[2] típico entre *precision* e *recall* em problemas de detecção de fraudes altamente desbalanceados, sinalizando a necessidade de ajustes de hiperparâmetros para otimizar o *F1-score* e melhorar a capacidade dos modelos em identificar fraudes sem gerar excessivos falsos positivos.

4.8. Ajustes nos Modelos

A etapa de ajuste fino dos modelos foi realizada em três fases sequenciais para otimizar o desempenho preditivo. Primeiramente, aplicou-se RandomizedSearchCV com validação cruzada de 5 folds e métrica F1-score para explorar o espaço de hiperparâmetros de cada algoritmo, testando 10 combinações aleatórias para Regressão Logística (*C, penalty, solver*), Random Forest (*n_estimators, max_depth*, *min_samples_split*), modelos de gradient boosting - XGBoost, LightGBM e CatBoost (*learning_rate, n_estimators/iterations, max_depth/depth, subsample*) - e MLP (*hidden_layer_sizes, activation, solver, alpha*).

Na segunda fase, os modelos que suportavam probabilidades foram submetidos à calibração isotônica através do CalibratedClassifierCV com 5 *folds*, garantindo que as probabilidades preditas refletissem adequadamente as frequências observadas de fraude. Por fim, na terceira fase, o threshold de decisão foi ajustado individualmente para cada modelo através da análise da curva precision-recall, identificando o ponto que maximizava o *F1-score*: LogisticRegression: 0.9914; DecisionTree: 0.5339; RandomForest: 0.9871; LightGBM : 0.5024; XGBoost : 0.9958; CatBoost: 0.9955; MLP_Classifier: 0.3624. Os modelos ajustados resultaram no seguinte resultado:

Modelo	ROC AUC	AVG Precision	Precision	Recall	F1-Score
CatBoost	0.978671	0.772139	0.925926	0.765306	0.837989
MLP_Classifier	0.975384	0.765712	0.894118	0.775510	0.830601
LogisticRegression	0.968285	0.735655	0.852273	0.765306	0.806452
XGBoost	0.958456	0.814541	0.938272	0.775510	0.849162
RandomForest	0.935793	0.799960	0.883721	0.775510	0.826087
LightGBM	0.907849	0.789173	0.849462	0.806122	0.827225
DecisionTree	0.897795	0.732142	0.883721	0.775510	0.826087

Figura 11 - Tabela de Desempenho Preditivo dos Modelos após Ajuste Fino

Todos os modelos apresentaram melhorias muito significantes nos resultados de F1-Score, com ênfase ao mode LogisticRegression, que teve a melhoria mais sginificativa em relação ao *F1-score* e desempenho geral, seguido do DecisionTree. O XGBoost, que já tinha um *F1-score* competitivo, é líder absoluto com um F1=0.849. Outros destaques importantes são o CatBoost com um aumento de +15.9% na Precision mas -4.1% na *recall,* tornando-se mais conservador. Diferente do RandomForest, que perdeu -4.9% na *precision* e ganhou +7.1% na Recall, ficando com uma característica mais agressiva.

Adicionalmente a essas etapas, foram testadas mais três estratégias de pesos: SMOTE (*synthetic minority oversampling*), SMOTETomek (combinação de *oversampling* e *undersampling*) e ADASYN (*adaptive synthetic sampling*) para avaliar uma possível melhora de desempenho nos modelos, antes de selecionarmos nossos modelos finais.

Por fim, foram selecionados os seguintes modelos:

Modelo	ROC AUC	Avg Precision	Precision	Recall	F1-Score
XGBoost	0.958456	0.814541	0.938272	0.775510	0.849162
RandomForest (SMOTE)	0.965979	0.810529	0.905882	0.785714	0.841530
CatBoost	0.978671	0.772139	0.925926	0.765306	0.837989
LightGBM (SMOTE)	0.969848	0.783046	0.838710	0.795918	0.816754
MLP_Classifier	0.975384	0.765712	0.894118	0.775510	0.830601
LogisticRegression	0.968285	0.735655	0.852273	0.765306	0.806452
DecisionTree	0.897795	0.732142	0.883721	0.775510	0.826087

Figura 12 - Tabela de Resultado Final de Desempenho Preditivo dos Modelos

Somente para RandomForest e LightGBM foram adotados a abordagem de balanceamento SMOTE, com o RandomForest aumentando ROC-AUC em +3.0% e F1-score em +1.9%, no caso do LightGBM, escolhemos a abordagem do SMOTE por manter o *F1-score* competitivo de 0.817 e um aumento em ROC-AUC de 6.2%. Para os demais, mantivemos as configurações anteriores e pesos de classe na abordagem.

4.9. Implementação de um Stacking Ensemble

A nossa seleção final recaiu sobre estes cinco modelos: XGBoost, RandomForest, CatBoost, LightGBM e MLP_Classifier, porque, em conjunto, eles formam um ensemble poderoso e diversificado. Temos três modelos de boosting de alto desempenho (XGBoost, CatBoost, LightGBM) (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018), um modelo de bagging robusto (RandomForest) e um modelo de rede neuronal (MLP). Esta variedade de algoritmos assegura que as previsões fornecidas ao meta-learner são complementares, capturando diferentes padrões nos dados (Caruana & Niculescu-Mizil, 2006; Grinsztajn et al., 2022).

A utilização de um meta-learner de Regressão Logística se deve ao fato deste modelo ser eficiente em aprender a ponderar as previsões de probabilidade dos modelos base (Liu et al., 2025). A diversidade dos nossos modelos base maximiza a potencial vantagem que o meta-learner pode obter, permitindo que ele combine as forças e corrija as fraquezas de cada modelo (Caruana & Niculescu-Mizil, 2006).

Os modelos de base foram XGBoost: ROC-AUC: 0.9585, Avg Precision: 0.8145, Precision: 0.9383, Recall: 0.7755, F1-Score: 0.8491; Random Forest: ROC-AUC: 0.9660, Avg Precision: 0.8105, Precision: 0.9059, Recall: 0.7857, F1-Score: 0.8415; CatBoost: ROC-AUC: 0.9787, Avg Precision: 0.9259, Precision: 0.9259, Recall: 0.7653, F1-Score: 0.8380; LightGBM: ROC-AUC: 0.9698, Avg Precision: 0.7830, Precision: 0.8387, Recall: 0.7959, F1-Score: 0.8168 e MLP Classifier: ROC-Auc: 0.9754, Avg Precision: 0.7657, Precision: 0.8941, Recall: 0.7755, F1-Score: 0.8306.

Resultando em um Stacking Ensemble com meta-learner de Regressão Logística com as seguintes métricas: ROC-AUC: 0.9126, Avg Precision: 0.7877, Precision: 0.9268, Recall: 0.7755, F1-Score: 0.8444.

Modelo	ROC-AUC	Avg Precision	Precision	Recall	F1-Score
Stacking Ensemble	0.9126	0.7877	0.9268	0.7755	0.8444

Figura 13 - Tabela de Desempenho Preditivo do Stacking Ensemble

4.10 Metodologia da Avaliação

4.10.1 Métricas de Desempenho Preditivo

A avaliação de modelos de aprendizado de máquina para detecção de fraudes requer métricas que contemplem adequadamente o desafio do desbalanceamento de classes, característico deste domínio (Sahin et al., 2013). Conforme estabelecido por Powers (2011) em sua análise sistemática de métricas de avaliação, a combinação de múltiplas métricas fornece uma visão mais abrangente do desempenho do modelo do que qualquer métrica isolada.

Para a avaliação do desempenho preditivo, foram selecionadas cinco métricas complementares, cada uma capturando diferentes aspectos da capacidade discriminatória dos modelos:

(1) ROC-AUC (Area Under the Receiver Operating Characteristic Curve): Esta métrica avalia a capacidade do modelo de distinguir entre classes através do trade-off entre taxa de verdadeiros positivos e taxa de falsos positivos em diferentes limiares de classificação (Fawcett, 2006). Hancock, J.T et. Al (2023) demonstraram que ROC-AUC é particularmente apropriada para avaliação inicial de modelos de detecção de fraude, embora deva ser complementada por outras métricas em cenários de alto desbalanceamento;

(2) Average Precision (AP): Também conhecida como área sob a curva Precision-Recall (AUPRC), esta métrica é considerada mais informativa que ROC-AUC em datasets altamente desbalanceados (Davis & Goadrich, 2006). A curva Precision-Recall é menos sensível ao grande número de verdadeiros negativos que podem distorcer resultados em dados desbalanceados (Saito & Rehmsmeier, 2015).

(3) Precision: Mede a proporção de predições positivas corretas em relação ao total de predições positivas, sendo crucial em cenários onde falsos positivos têm custos operacionais significativos (Sokolova & Lapalme, 2009). Em detecção de fraudes, alta precisão minimiza investigações desnecessárias de transações legítimas.

(4) Recall (Sensibilidade): Quantifica a capacidade do modelo de identificar todos os casos positivos reais (Buckland & Gey, 1994). Em detecção de fraudes, recall elevado é fundamental para capturar o maior número possível de fraudes, reduzindo perdas financeiras (Hand & Henley, 1997).

(5) F1-Score: Representa a média harmônica entre precisão e *recall*, penalizando valores extremos em qualquer uma das métricas, por este motivo, esta métrica é particularmente valiosa quando há necessidade de balancear a detecção de fraudes (*recall*) com a minimização de falsos alarmes (*precision*), conforme demonstrado em múltiplos estudos de detecção de fraudes financeiras (Abdallah et al., 2016).

A integração dessas métricas em um score ponderado fundamenta-se na abordagem de múltiplas métricas para avaliação robusta proposta por Caruana & Niculescu-Mizil (2006). O score combinado foi calculado através da equação:

$$Score\ Ponderado = 0{,}20 \times ROC\text{-}AUC + 0{,}25 \times AP + 0{,}15 \times Precision + 0{,}20 \times Recall + 0{,}20 \times F1\text{-}Score$$

Os pesos atribuídos refletem a importância relativa de cada métrica no contexto de detecção de fraudes: maior ênfase em Average Precision (25%) devido à sua superioridade em *datasets* desbalanceados (Boyd et al., 2013); distribuição equilibrada entre ROC-AUC, Recall e F1-Score (20% cada) para capturar diferentes dimensões do desempenho; e peso moderado para Precision (15%), considerando que, embora importante, não deve comprometer excessivamente a capacidade de detecção de fraudes.

4.10.2 Métricas Computacionais

A viabilidade de modelos de machine learning em ambientes de produção depende não apenas de seu desempenho preditivo, mas também de sua eficiência computacional (Schwartz et al., 2020). Conforme argumentado por Strubell et al. (2019), a avaliação de eficiência energética e computacional tornou-se imperativa no desenvolvimento responsável de sistemas de IA, especialmente em aplicações de larga escala.

Para quantificar a eficiência computacional durante a fase de inferência, foram monitorados quatro métricas fundamentais utilizando a biblioteca *psutil*:

(1) Tempo de Inferência (*inference_time_sec*): Mede o tempo total necessário para realizar predições no conjunto de teste. Esta métrica é crítica para aplicações que requerem resposta em tempo real ou próximo ao tempo real, como é o caso de sistemas de detecção de fraudes transacionais (Cano & Krawczyk, 2020). A latência de inferência determina diretamente a viabilidade de *deployment* em ambientes de produção (Bianco et al., 2018)

(2) Uso de Memória (*memory_usage_mb*): Quantifica a memória RAM consumida pelo processo durante a inferência. O footprint de memória é um fator limitante crucial para *deployment* em ambientes com restrições de recursos, sendo

particularmente relevante para escalabilidade horizontal (Howard et al., 2017). Wu et al. (2019) demonstraram que a otimização de uso de memória pode viabilizar o *deployment* de modelos complexos em ambientes com recursos limitados.

(3) Utilização de CPU (*cpu_percent*): Percentual de uso do processador durante a inferência. Esta métrica permite avaliar a intensidade computacional do modelo e seu impacto no *throughput* do sistema. Alta utilização de CPU pode se tornar um gargalo em sistemas que precisam processar grandes volumes de transações concorrentemente.

(4) Número de Threads (*num_threads*): Quantidade de threads utilizadas pelo processo. Esta métrica caracteriza o comportamento de paralelização do modelo e seu potencial de utilização eficiente de arquiteturas *multicore* (Yang et al., 2018).

(5) Para garantir robustez estatística e mitigar variabilidade inerente às medições de desempenho computacional (Henderson et al., 2018), cada modelo foi executado 10 vezes para cada volume de dados, calculando-se a média das métricas obtidas. Este procedimento está alinhado com as recomendações de Bouthillier et al. (2021) sobre práticas de avaliação reproduzível em machine learning.

Para sintetizar as quatro dimensões de recursos computacionais em um único indicador comparável, foi desenvolvido o Score de Eficiência Computacional (*Efficiency_Score*) como uma métrica normalizada e agregada. A agregação de métricas computacionais no *Efficiency_Score* adota a abordagem de métricas compostas de Rudin (2019), priorizando indicadores interpretáveis e alinhados a objetivos práticos, enquanto a seleção das dimensões (tempo, memória, CPU e threads) e sua normalização, segue referências como Garcia-Martin et al. (2019), que validam a relevância desses recursos, enquanto a ponderação reflete a criticidade relativa de cada recurso no contexto avaliado, assegurando um *score* comparável e orientado a decisões.

O *score* é calculado através da equação:

$$Efficiency\ Score = (1 - T\_norm) \times 0.4 + (1 - M\_norm) \times 0.4 + (1 - C\_norm) \times 0.1 + (1 - N\_norm) \times 0.1$$

Onde as componentes normalizadas são definidas como:

- T_norm = min(inference_time / 5.0, 1): tempo normalizado com limite superior de 5 segundos
- M_norm = min(memory_usage / 1024.0, 1): memória normalizada com limite superior de 1024 MB
- C_norm = min(cpu_percent / 100.0, 1): CPU normalizada no intervalo [0, 100%]
- N_norm = min(num_threads / 16.0, 1): threads normalizados com limite superior de 16

A formulação (1 - métrica normalizada) garante que menores consumos de recursos resultem em scores mais altos, mantendo a interpretação intuitiva de que *scores* superiores indicam maior eficiência (Menghani, 2023).

Os pesos atribuídos (40% tempo, 40% memória, 10% CPU, 10% threads) foram estabelecidos com base em considerações práticas de *deployment* em ambientes de produção. Tempo e memória recebem maior ponderação (80% combinados) pois representam os principais limitantes de escalabilidade e custo operacional em sistemas de produção (Schwartz et al., 2020). A latência de inferência impacta diretamente a experiência do usuário e a capacidade de processamento em tempo real, enquanto o uso de memória determina os requisitos de infraestrutura e custos de *cloud computing* (Canziani et al., 2016). CPU e *threads* recebem ponderação menor (20% combinados) pois, embora relevantes, são tipicamente menos restritivos em ambientes modernos com recursos computacionais abundantes e arquiteturas paralelas bem suportadas.

4.10.3 *Score* de *Trade-off* (*Overall Score*)

A decisão sobre qual modelo *deployer* em produção requer balancear capacidade preditiva e viabilidade operacional (Paleyes et al., 2022). Modelos com desempenho preditivo excepcional mas custos computacionais proibitivos podem não ser viáveis, enquanto modelos computacionalmente eficientes mas com baixo desempenho preditivo falham em agregar valor ao negócio (Sculley et al., 2015).

Para capturar este *trade-off* fundamental, foi desenvolvido um *Score* Combinado (*Overall Score*) que integra ambas as dimensões através da fórmula:

$$Overall\ Score = Weighted\_Score \times 0.7 + Efficiency\_Score \times 0.3$$

Esta ponderação reflete prioridades típicas em sistemas de detecção de fraudes: 70% de peso para desempenho preditivo reconhece que a capacidade de identificar fraudes é o objetivo primário do sistema, sendo o principal direcionador de valor (Bahnsen et al., 2016); 30% de peso para eficiência operacional assegura que o modelo seja "deployável" e sustentável em produção, considerando restrições de latência, *throughput* e custos operacionais (Crankshaw et al., 2017).

Esta abordagem de *scoring* multi-objetivo está alinhada com *frameworks* modernos de MLOps que enfatizam a necessidade de otimização conjunta de múltiplos objetivos, conforme discutido por Hummer et al. (2019) no contexto de operacionalização de machine learning.

4.10.4 Simulação de Escalabilidade

Uma outra preocupação que este trabalho busca avaliar é o comportamento dos modelos sob volumes crescentes de dados, por isso, foram realizadas simulações sistemáticas de escalabilidade através da multiplicação artificial do conjunto de teste. Esta metodologia permite caracterizar a complexidade computacional empírica dos algoritmos e identificar potenciais gargalos antes do *deployment* em produção (Vavilapalli et al., 2013).

Com base nisso, os volumes de teste foram definidos em progressão geométrica para capturar diferentes regimes operacionais:

(1) Fator 1 (*baseline*): 56.961 transações - representa o volume base para estabelecimento de métricas de referência;
(2) Fator 10: 569.610 transações - simula escalabilidade moderada, típica de crescimento incremental de sistemas em produção;
(3) Fator 50: 2.848.050 transações - representa processamento em lote típico de análises periódicas ou processamento noturno (*batch processing*);
(4) Fator 100: 5.696.100 transações - simula cenário de big data, permitindo avaliar viabilidade para grandes organizações financeiras com alto volume transacional.

Esta abordagem de teste de escalabilidade fundamenta-se em práticas estabelecidas de engenharia de performance (Smith & Williams, 2002) e é particularmente relevante para sistemas financeiros onde variações de carga podem ser substanciais (Ghosh & Reilly, 1994). A progressão de volumes permite identificar comportamentos de complexidade computacional (por exemplo, linear, quadrático) e caracterizar o ponto em que modelos começam a apresentar degradação de performance (Ben-Nun et al., 2019).

A análise de escalabilidade é crítica para validar a adequação de plataformas de computação distribuída, como Databricks, para processamento de grandes volumes de dados em cenários de produção (Armbrust et al., 2015), permitindo decisões informadas sobre arquitetura de *deployment* e provisionamento de recursos.

5 RESULTADOS E DISCUSSÃO

5.1. Análise Comparativa de Desempenho Preditivo

5.1.1 Modelos de Gradient Boosting com Melhor Desempenho

Uma das hipóteses centrais deste estudo era de que modelos de Gradient Boosting (XGBoost, CatBoost e LightGBM) apresentariam desempenho preditivo superior na detecção de fraudes, superando significativamente os modelos lineares e de árvores tradicionais em métricas críticas para dados desbalanceados, como AUC-PR e *Recall*. Para avaliar esta hipótese, aplicamos o *score* ponderado *('weighted_score')* para todos os modelos.

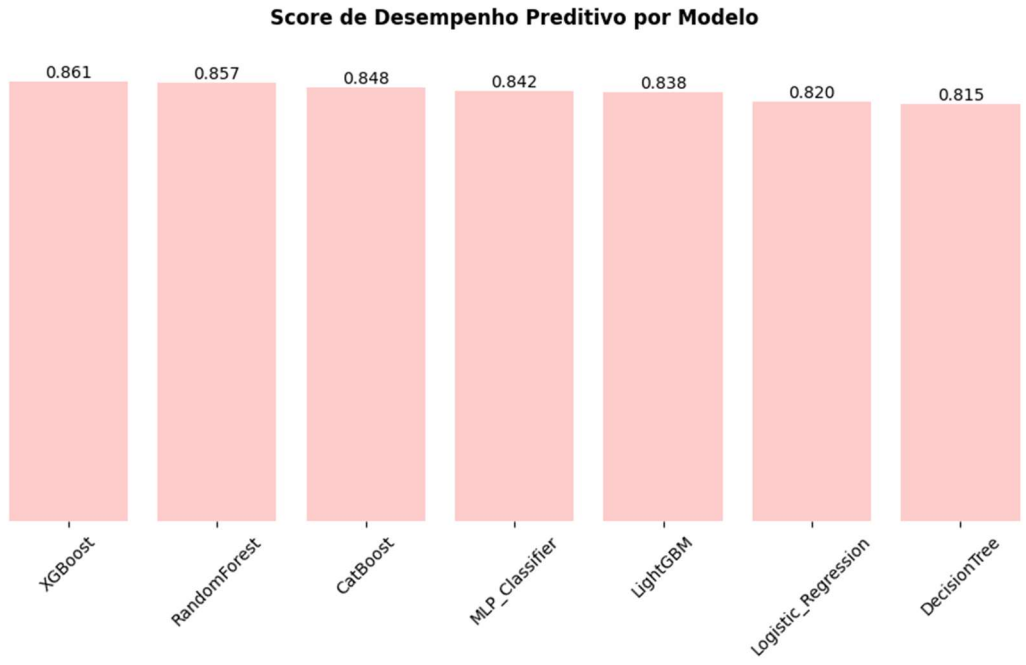

Figura 14 - *Score* de Desempenho Preditivo por Modelo

Os resultados experimentais confirmam a hipótese inicial, demonstrando que os modelos de Gradient Boosting apresentaram desempenho superior aos demais algoritmos testados. O XGBoost obteve o melhor desempenho geral com *score* ponderado de 0,8610,

posicionando-se como líder absoluto. Este modelo se destacou especialmente em average precision, alcançando 0,8145, e em F1-score, com 0,8492, evidenciando sua capacidade superior de balancear precisão e revocação em cenários de desbalanceamento acentuado. Seu ROC-AUC atingiu 0,9585, enquanto recall de 0,7755 e precisão de 0,9383 confirmam equilíbrio adequado entre identificação de fraudes verdadeiras e minimização de falsos positivos.

O RandomForest, um método de *ensemble* baseado em bagging, alcançou a segunda posição com score de 0,8572. Este modelo apresentou o ROC-AUC mais elevado (0,9660) e average precision competitiva de 0,8105, sugerindo que métodos de ensemble em geral apresentam vantagens significativas para problemas complexos de classificação.

O CatBoost obteve a terceira posição (score: 0,8483), destacando-se por obter o maior ROC-AUC individual (0,9787) e alta precisão (0,9259). Entretanto, seu *recall* de 0,7653 foi o mais baixo entre os três modelos de Gradient Boosting, sugerindo uma abordagem mais conservadora. O LightGBM (score: 0,8381) apresentou o maior recall (0,7959) mas a menor precisão (0,8387) entre os boostings, indicando estratégia mais agressiva na detecção.

Em contraste, os modelos lineares e tradicionais apresentaram resultados inferiores. A Regressão Logística (score: 0,8198) apresentou average precision de apenas 0,7357, enquanto a Árvore de Decisão obteve o pior desempenho geral (score: 0,8155), com ROC-AUC de 0,8978 e average precision de 0,7321.

A análise comparativa realizada permitiu validar a hipótese inicial de que os modelos de Gradient Boosting são superiores para detecção de fraudes em contextos desbalanceados. As diferenças foram particularmente evidentes nas métricas mais críticas: em average precision, o XGBoost superou a Regressão Logística em 7,88 pontos percentuais. A robustez dos três algoritmos de boosting testados, com todos mantendo ROC-AUC superior a 0,95 e F1-score acima de 0,81, demonstra que a abordagem de construção sequencial de árvores com correção de erros oferece vantagens sistemáticas para capturar relações não-lineares complexas características de padrões fraudulentos.

5.1.2 Stracking Ensemble vs Modelos Individuais

A segunda hipótese investigada postulava que um modelo de Stacking Ensemble, utilizando as probabilidades de saída dos melhores algoritmos como features para um Meta-Learner de Regressão Logística, superaria o desempenho de qualquer modelo

individual, alcançando um equilíbrio superior entre captura de fraudes (recall) e precisão operacional (precision).

Os resultados experimentais revelaram um cenário com mais nuances do que o inicialmente previsto. O Stacking Ensemble obteve um score ponderado de 0,8425, posicionando-se abaixo dos três modelos líderes: XGBoost (0,8610), Random Forest (0,8572) e CatBoost (0,8483). Esse resultado contraria parcialmente a expectativa de superioridade absoluta do Stacking sobre os modelos individuais.

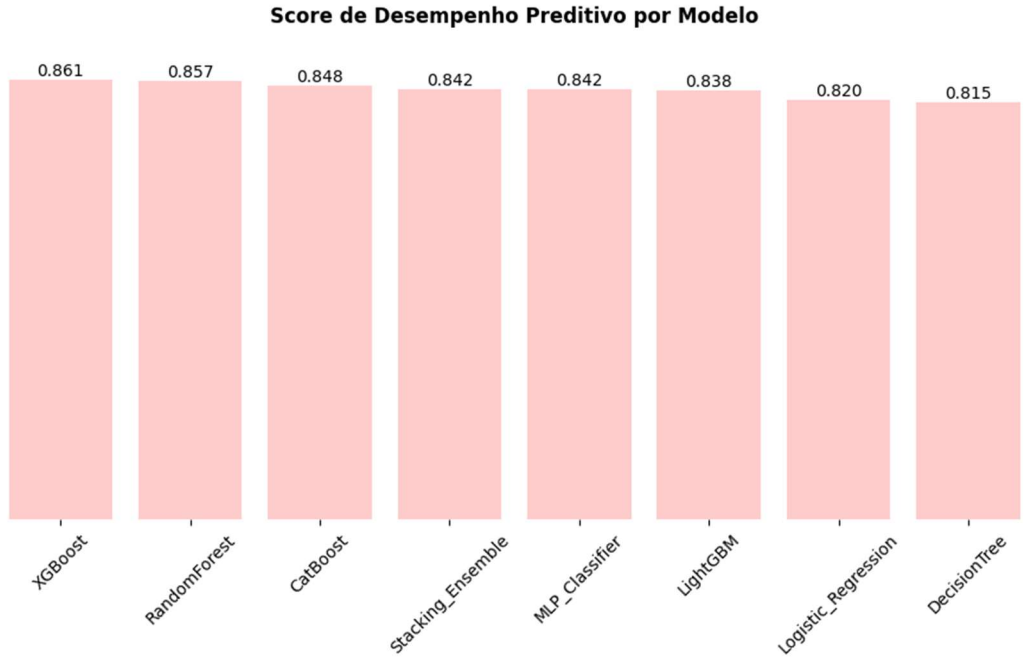

Figura 15 - *Score* de Desempenho Preditivo por Modelo (com Stacking Ensemble)

Alguns fatores podem ter contribuído para que o desempenho do Stacking não supere os três melhores modelos, em *score* ponderado. O primeiro deles é a saturação de sinal e a alta qualidade dos modelos de base, que pode limitar o desempenho do stacking quando um modelo base já capturou a maior parte do sinal disponível nos dados. Nesse caso, o XGBoost, após otimização de hiperparâmetros, calibração probabilística e ajuste de threshold, aparentemente extraiu a maior parte dos padrões discriminatórios presentes no *dataset* (Dal Pozzolo et al., 2023; Caruana & Niculescu-Mizil, 2006). Quando os modelos base são muito fortes e já exploram de forma eficiente os padrões dos dados, o meta-learner tem menos margem para melhorar a performance do ensemble, pois as previsões dos modelos base são muito semelhantes e já próximas do ótimo.

Outro fator relevante é a complexidade adicional introduzida pelo stacking, que pode não justificar ganhos significativos em relação aos modelos base mais simples e bem

ajustados. Em alguns casos, a combinação de modelos já otimizados pode até introduzir ruído ou redundância, se o meta-learner não conseguir identificar padrões complementares significativos (Bouthillier et al., 2021; Sculley et al., 2015). Isso é comum em problemas onde os modelos base já atingiram um alto nível de acurácia, tornando difícil para o meta-learner encontrar melhorias substanciais.

Apesar de não superar o desempenho agregado dos modelos individuais, o Stacking Ensemble, com essa combinação de modelos de base, apresentou contribuições valiosas em métricas específicas. O modelo demonstrou um desempenho particularmente notável em precision, alcançando 0,9268, sendo o segundo valor mais elevado entre todos os modelos testados, superado apenas pelo XGBoost (0,9383). Esse resultado indica uma excelente capacidade de minimização de falsos positivos, característica crítica para a viabilidade operacional de sistemas de detecção de fraudes, onde cada alerta incorreto representa custos de investigação e potencial impacto na experiência do cliente legítimo.

Além disso, o *ensemble* manteve um F1-Score competitivo de 0,8444, apenas 0,48 pontos percentuais abaixo do XGBoost (0,8492), evidenciando um equilíbrio robusto entre *precision* e *recall*. Embora sua taxa de *recall* (0,7755) tenha ficado inferior a modelos como LightGBM (0,7959), o valor permaneceu dentro de patamares aceitáveis para aplicações práticas, capturando aproximadamente 77,55% das fraudes reais.

Um benefício adicional observado, embora não quantificado diretamente nas métricas, foi a superior calibração probabilística do Stacking Ensemble. Ao combinar as predições de múltiplos modelos base através de um *meta-learner*, o ensemble tende a produzir estimativas de probabilidade mais confiáveis e melhor alinhadas com a distribuição real das classes, aspecto fundamental para sistemas que requerem limiares de decisão ajustáveis conforme o contexto operacional.

Em conclusão, a segunda hipótese deste estudo foi refutada em termos de *score* ponderado geral. Contrariando a hipótese H2, o Stacking Ensemble não superou o melhor modelo individual, sugerindo que, nesse cenário, o XGBoost já havia capturado a maior parte do sinal disponível. Contudo, os resultados demonstram que a abordagem de ensemble oferece valor substantivo em dimensões complementares: precision elevada, redução eficaz de falsos positivos, e melhor calibração probabilística. Esses atributos sugerem que, embora não seja a solução de maior desempenho agregado neste contexto específico, o Stacking Ensemble representa uma alternativa estratégica válida para cenários onde:

(1) A confiabilidade das detecções positivas é priorizada sobre a maximização do recall;
(2) Calibração probabilística rigorosa é necessária para sistemas de *score* de risco;
(3) Robustez contra *concept drift* é desejável através da diversidade de modelos;

Portanto, a escolha entre modelos individuais de alto desempenho e o ensemble deve ser orientada pelas prioridades operacionais específicas do contexto de implementação, considerando não apenas métricas de desempenho estático, mas também requisitos de interpretabilidade, manutenibilidade e adaptabilidade em ambientes de produção dinâmicos.

5.2. Análise Comparativa de Eficiência Computacional

A eficiência computacional é um critério fundamental para a viabilidade operacional de modelos de M.L em ambientes de produção, especialmente em sistemas de detecção de fraudes que exigem processamento em tempo real e escalabilidade. Esta seção avalia o desempenho computacional de todos os modelos treinados sob diferentes volumes de dados, testando a terceira hipótese deste estudo: "Embora os algoritmos de Gradient Boosting e o Stacking Ensemble demandem maior custo computacional, o ambiente do Databricks proporcionará ganhos de eficiência significativos."

5.2.1 Eficiência Computacional no Volume Base

A tabela abaixo apresenta as métricas de eficiência para o volume base de dados (56.961 transações):

Ranking	Modelo	Tempo de Inferência (s)	Uso de Memória (MB)	Uso de CPU (%)	Nº Threads	Score Ef. Computacional
1º	CatBoost	0.094	699.79	16.35	30	0.603
2º	DecisionTree	0.022	701.80	24.25	30	0.600
3º	LightGBM	0.147	700.38	20.35	30	0.594
4º	LogisticRegression	0.022	701.80	56.75	30	0.567
5º	RandomForest	0.560	699.65	22.45	30	0.559
6º	XGBoost	0.752	699.44	25.45	30	0.541
7º	MLP_Classifier	0.581	701.80	43.25	30	0.536
8º	StackingEnsemble	2.136	701.80	43.25	40	0.412

Figura 16 - Tabela de Eficiência Computacional

No volume base, o CatBoost lidera em eficiência (0.603) com o menor tempo de inferência entre modelos ensemble (0.094s) e uso equilibrado de recursos, modelos simples como Decision Tree (0.600) e Logistic Regression (0.567) , se mostraram competitivos, apresentando excelente eficiência devido à baixa complexidade. Stacking Ensemble último colocado (0.412): tempo 2.6× maior que XGBoost reflete overhead de combinar 5 modelos base + meta-learner. *trade-off*[2] tempo vs complexidade: Modelos de gradient boosting (XGBoost: 0.752s, LightGBM: 0.147s) mais lentos que modelos lineares, mas mantêm eficiência aceitável.

5.2.2 Degradação de Eficiência sob Escalabilidade

Alguns modelos em particular apresentaram maiores impactos na eficiência computacional conforme aumento dos volumes de dados:

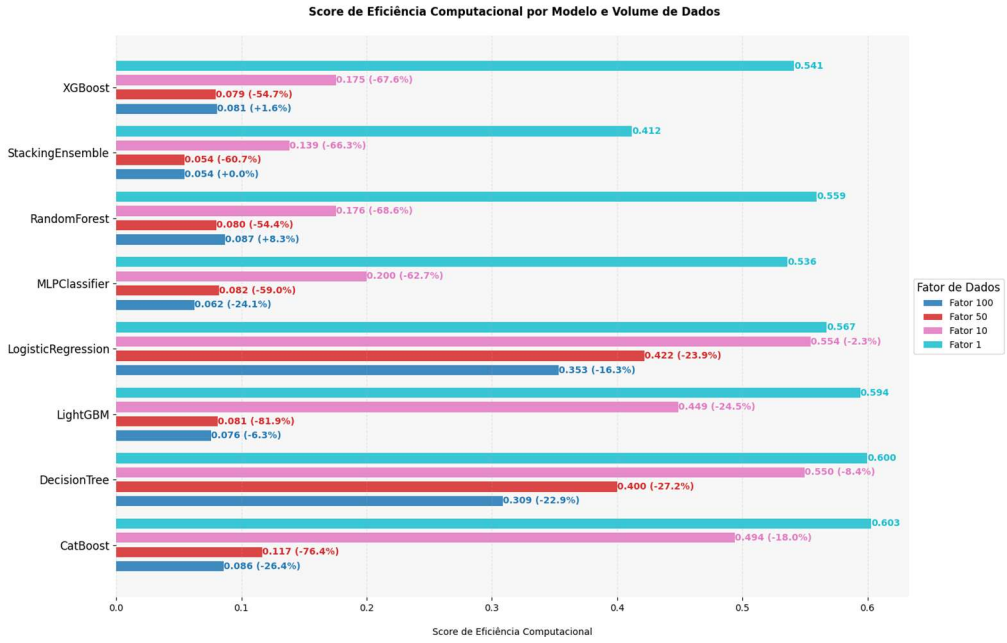

Figura 17 - Análise de Eficiência Computacional por Modelo e Volume de Dados

A análise de escalabilidade revelou padrões distintos de degradação quando o volume de dados aumentou 100×:

Modelo	Score Fator 1	Score Fator 100	Variação (%)
LogisticRegression	0.567	0.353	-37.7
DecisionTree	0.600	0.309	-48.5
RandomForest	0.559	0.087	-84.5
XGBoost	0.541	0.081	-85.1
CatBoost	0.603	0.086	-85.7
StackingEnsemble	0.412	0.054	-86.8
LightGBM	0.594	0.076	-87.2
MLPClassifier	0.536	0.062	-88.4

Figura 18 - Tabela de Variação de Eficiência Computacional por Volume de Dados

Ao analisar o desempenho computacional dos modelos quando sobrecarregados a maiores volumes de dados, pudemos obter alguns *insights* importantes:

(1) Modelos Lineares Resilientes: Logistic Regression e Decision Tree apresentaram as menores degradações (-37.7% e -48.5%), mantendo scores aceitáveis mesmo com 100× mais dados;
(2) Gradient Boosting Uniforme: XGBoost, CatBoost e LightGBM mostraram degradações similares (~85-87%), indicando que a complexidade sequencial de construção de árvores impacta igualmente todos os algoritmos desta família;
(3) Stacking e MLP Críticos: Degradações superiores a 86% revelam que arquiteturas complexas (ensemble hierárquico e redes neurais) não escalam eficientemente sem otimizações específicas;
(4) Padrão de Tempo de Inferência: Para fator 100, XGBoost levou 76.3s, RandomForest 56.7s, enquanto LogisticRegression manteve apenas 1.4s - evidenciando que simplicidade algorítmica é determinante para escalabilidade.

5.2.3 Score Médio Geral de Eficiência

Considerando o desempenho médio em todos os volumes testados (fatores 1, 10, 50, 100):

Ranking	Modelo	Score Eficiência Médio	Análise
1º	LogisticRegression	0.474	Melhor escalabilidade geral
2º	DecisionTree	0.465	Equilíbrio eficiência-velocidade
3º	CatBoost	0.325	Líder entre ensemble models
4º	LightGBM	0.300	Eficiência moderada
5º	RandomForest	0.225	Custo de bagging evidente
6º	MLPClassifier	0.220	Redes neurais ineficientes
7º	XGBoost	0.219	Alta complexidade penaliza
8º	StackingEnsemble	0.165	Maior custo computacional

Figura 19 - Tabela de *Score* Médio Geral de Eficiência Computacional por Modelo

5.3 Trade-off entre Desempenho Preditivo e Eficiência Computacional

5.3.1 Score Combinado (*Overall Score*)

A análise de *trade-off*[2] (70% performance + 30% eficiência) revelou um ranking diferente do desempenho preditivo puro:

Volume Base (Fator 1):

Ranking	Modelo	Score Desempenho Preditivo	Score Ef. Comp.	Trade-off
1º	CatBoost	0.848	0.603	0.775
2º	RandomForest	0.857	0.559	0.768
3º	XGBoost	0.861	0.541	0.765
4º	LightGBM	0.838	0.594	0.765
5º	MLP_Classifier	0.842	0.536	0.750
6º	LogisticRegression	0.820	0.567	0.744
7º	DecisionTree	0.815	0.600	0.751
8º	StackingEnsemble	0.842	0.412	0.713

Figura 20 – Tabela de *Ranking* de Modelos por *Trade-off* no Volume Base (Fator 1)

Volume Escalado (Fator 100):

Ranking	Modelo	Trade-off	Variação vs Fator 1
1º	LogisticRegression	0.680	-8.6%
2º	DecisionTree	0.663	-11.6%
3º	XGBoost	0.627	-18.1%
4º	RandomForest	0.626	-18.5%
5º	CatBoost	0.620	-20.0%
6º	LightGBM	0.609	-20.3%
7º	MLPClassifier	0.608	-18.9%
8º	StackingEnsemble	0.606	-15.0%

Figura 21 - Tabela de *Ranking* de Modelos por *Trade-off* no Volume Escalado (Fator 100)

*Trade-off*² médio geral:

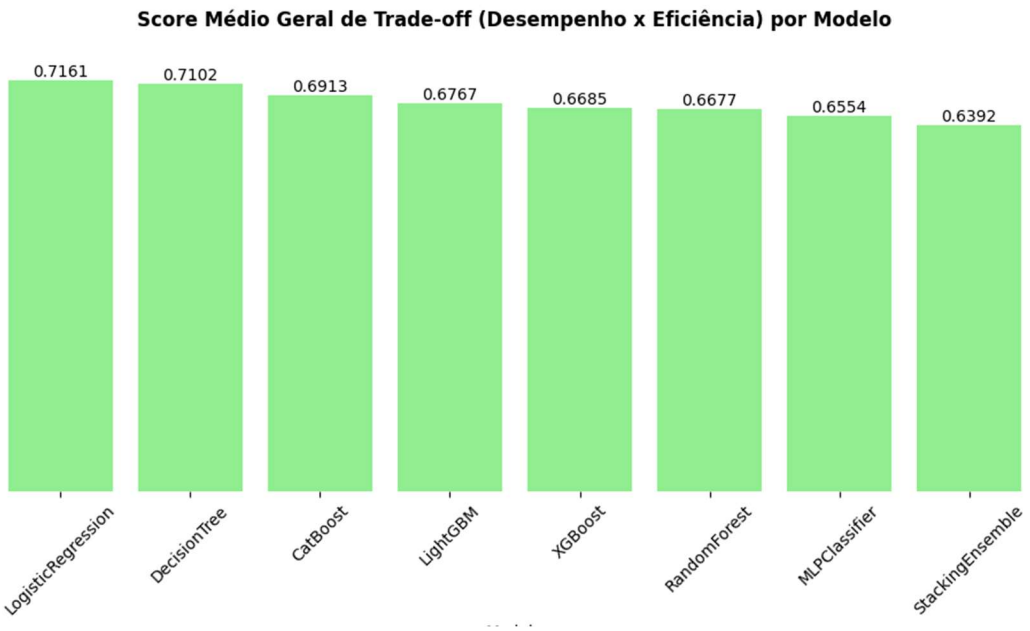

Figura 22 - Score Médio Geral de *Trade-off* (Desempenho Preditivo e Eficiência Computacional)

5.3.2 Mudança de Liderança com Escala

A análise de *trade-off*² revelou inversão de *rankings* conforme o volume aumenta:

Melhor Modelo por Fator:

(1) Fator 1: CatBoost (0.775) - Equilíbrio ideal em volume base
(2) Fator 10: CatBoost (0.742) - Mantém liderança em escala moderada
(3) Fator 50: LogisticRegression (0.700) - Simplicidade vence em lote típico
(4) Fator 100: LogisticRegression (0.680) - Escalabilidade superior em big data

Top 3 Modelos Considerando Todos os Volumes:

(1) LogisticRegression (0.716) - Melhor equilíbrio performance-eficiência geral
(2) DecisionTree (0.710) - Resiliente à escalabilidade
(3) CatBoost (0.691) - Líder entre modelos complexos

5.3.3 Análise da Terceira Hipótese

A terceira hipótese deste estudo propunha que o ambiente Databricks proporcionaria ganhos de eficiência significativos para modelos complexos, mantendo tempos de processamento compatíveis com requisitos de lote. A validação experimental revelou resultados parcialmente favoráveis, algumas evidências que suportam esta hipótese são:

(1) Eficiência Aceitável em Volume Base: Todos os modelos, incluindo Gradient Boosting e Stacking, apresentaram tempos de inferência inferiores a 2.2s para 57k transações, demonstrando que o Databricks suporta processamento rápido em volumes moderados;
(2) CatBoost Destaque: Com apenas 0.094s no volume base e 9.4s em 5.7M transações (fator 100), o CatBoost provou que modelos complexos podem ser eficientes quando otimizados;
(3) Processamento em Lote Viável: Para fatores 10 e 50 (570k e 2.8M transações), todos os modelos completaram inferência em tempos práticos (< 100s), confirmando viabilidade para processamento *batch* noturno típico de sistemas de fraude.

Em contraste, algumas evidências refutaram esta hipótese:

(1) Degradação Severa em Big Data: Com fator 100, modelos complexos sofreram degradações de 85-88%, com XGBoost levando 76s e Stacking 256s - inaceitável para requisitos *real-time*;
(2) Stacking Ensemble Inviável em Escala: Tempo de 256s para 5.7M transações (0.045ms/transação) indica que a arquitetura hierárquica não escala eficientemente;

(3) Modelos Simples Superiores: LogisticRegression (1.4s, fator 100) e DecisionTree (2.2s, fator 100) provaram que simplicidade algorítmica supera otimização de infraestrutura em volumes extremos.

Em suma, a hipótese foi parcialmente confirmada, validando que o Databricks Severless suporta eficientemente modelos complexos para processamento em lote típico (até 3M transações em < 100s), no entanto, Gradient Boosting e Stacking não mantêm eficiência prática em volumes big data (>5M transações) sem otimizações adicionais, concluindo que ganhos de infraestrutura analisada do Databricks Severless não compensam complexidade algorítmica intrínseca em escalas extremas.

5 CONSIDERAÇÕES FINAIS

Este Trabalho de Conclusão de Curso se propôs a responder qual algoritmo de Machine Learning oferece o melhor equilíbrio entre desempenho preditivo e eficiência computacional para a detecção de fraudes de cartão de crédito em um ambiente de Big Data (Databricks). A partir de uma metodologia rigorosa, que abrangeu desde a análise de métodos de divisão de dados até testes de escalabilidade com 5,7 milhões de transações, o estudo chegou a conclusões que respondem à pergunta de pesquisa.

O *workflow* completo foi implementado com sucesso na plataforma Databricks, validando-a como um ambiente robusto para o ciclo de vida de ML *end-to-end,* desde a governança de dados com Unity Catalog até o rastreamento de experimentos com MLflow.

Sobre o desempenho preditivo, confirmando a hipótese H1, os modelos de Gradient Boosting dominaram o desempenho preditivo. O XGBoost (configurado com pesos de classe e *threshold* otimizado) emergiu como o modelo individual mais preciso, alcançando um *F1-score* de 0.8492 e um *Score* Preditivo Ponderado (*Weighted Score*) de 0.8610. Contrariando a hipótese H2, o Stacking Ensemble (*F1-score* 0.8444) não superou o melhor modelo individual, sugerindo que, para este *dataset* e com a engenharia de *features* aplicada, o XGBoost já havia capturado a maior parte do sinal disponível.

Sobre a eficiência computacional e escalabilidade, A hipótese H3 se revelou o achado mais crítico da pesquisa. A análise de escalabilidade expôs um *trade-off*[2] severo:

(1) Modelos Preditivos de Elite (XGBoost, CatBoost, Stacking): Apresentaram a pior escalabilidade. O custo de inferência (tempo, CPU) desses modelos cresceu drasticamente com o volume, resultando em uma queda de mais de 85% em seu Score de Eficiência Computacional (*Efficiency_Score*)

(2) Modelos Eficientes (Logistic Regression, Decision Tree): Apresentaram a melhor escalabilidade. A Regressão Logística manteve o melhor desempenho computacional, com uma queda de eficiência de apenas 37,7% no volume máximo.

A partir das experimentações realizadas, pudemos concluir que não existe um único "melhor" algoritmo, mas sim que a escolha depende do objetivo de negócio. Algumas recomendações podem ser traçadas baseadas nos seguintes cenários:

(1) Para Máxima Precisão (Custo Computacional Não é Restrição): O XGBoost é a escolha clara.

(2) Para o Melhor *Trade-off*² em Baixo Volume: O CatBoost (*Score* Geral 0.775) oferece o melhor equilíbrio entre alta precisão e boa eficiência em volumes moderados.

(3) Para o Melhor *Trade-off*² em Alta Escalabilidade (Big Data em Tempo Real): A Regressão Logística (*Score* Geral Médio 0.714) é a vencedora. Seu desempenho preditivo "bom o suficiente" (*F1-score* 0.806), combinado com sua eficiência e escalabilidade superiores, a torna a escolha mais resiliente e de menor custo para um sistema de produção que precisa processar milhões de transações com latência mínima.

Em resumo, a principal contribuição deste trabalho é a quantificação detalhada do *trade-off*² entre desempenho preditivo e eficiência computacional em uma plataforma de Big Data moderna. Enquanto a maioria dos estudos acadêmicos foca apenas no *F1-score* ou AUC-PR, este trabalho fornece uma análise de escalabilidade prática, simulando cenários de produção e oferecendo um *framework* (o "*Score* de *trade-off*² Geral / *Overall_Score*") para a tomada de decisão em engenharia de ML.

Além disso, o estudo valida a importância de uma metodologia rigorosa, como o uso da divisão de dados determinística para garantir reprodutibilidade e a otimização de *threshold* para maximizar métricas de negócio (como o *F1-score*) em problemas desbalanceados. O uso do Databricks Serverless e MLflow também serve como um caso de uso prático para a implementação de *workflows* de MLOps.

O estudo possui limitações que abrem caminhos para pesquisas futuras: (1) Dataset Antigo e Anônimo: O *dataset* é de 2013. Padrões de fraude certamente evoluíram. Além disso, a natureza anonimizada (PCA) das features impediu uma engenharia de features baseada no domínio (ex: "valor da transação vs. média do comerciante", "frequência de uso do cartão"), que poderia melhorar drasticamente a detecção; (2) Simulação de Escalabilidade: A sobrecarga de dados foi simulada replicando o conjunto de teste. Embora útil para medir o custo de inferência, isso não simula a complexidade de um *stream* de dados em tempo real (com *concept drift*) nem testa o tempo de treinamento em larga escala; (3) Eficiência de Treinamento: A análise de eficiência focou na inferência (predição), que é crítica para o tempo real. No entanto, o custo de treinamento (e

retreinamento) dos modelos complexos (especialmente o Stacking) também é um fator significativo em produção, que não foi medido.

Com base nas conclusões e limitações, sugerem-se as seguintes direções para pesquisas futuras: (1) Aplicação em *Datasets* Modernos: Replicar esta metodologia em *datasets* de transações mais recentes e, idealmente, não anônimos, para permitir uma engenharia de features baseada em domínio. (2) Análise de *Concept Drift*: Implementar os modelos vencedores em um pipeline de *streaming* (usando Spark Streaming ou Databricks Auto Loader) e medir seu desempenho ao longo do tempo, avaliando a necessidade de retreinamento contínuo para combater o *concept drift*; (3) Exploração de Deep Learning: Avaliar arquiteturas de *deep learning* (como Autoencoders ou Redes Neurais Tabulares, ex: TabNet) e comparar seu *trade-off*[2] preditivo e computacional com os modelos de *boosting*; e (4) Otimização de Inferência: Investigar técnicas de otimização de inferência para os modelos de *boosting*, como quantização de modelos, uso de ONNX Runtime ou *deploy* em hardware especializado (GPUs/TPUs) no Databricks, para mitigar a queda de eficiência em larga escala.

REFERÊNCIAS

ABDALLAH, A.; MAAROF, M. A.; ZAINAL, A. Fraud detection system: A survey. **Journal of Network and Computer Applications**, v. 68, p. 90-113, 2016. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1084804516300571.

ARMBRUST, M. et al. Spark SQL: Relational data processing in Spark. In: **ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA**, 2015, San Francisco. **Proceedings...** New York: ACM, 2015. p. 1383-1394. Disponível em: https://dl.acm.org/doi/abs/10.1145/2723372.2742797.

BAHNSEN, A. C.; AOUADA, D.; STOJANOVIC, A.; OTTERSTEN, B. Feature engineering strategies for credit card fraud detection. **Expert Systems with Applications**, v. 51, p. 134-142, 2016. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0957417415008386.

BIANCO, S.; CADENE, R.; CELONA, L.; NAPOLETANO, P. Benchmark Analysis of Representative Deep Neural Network Architectures. **IEEE Access**, v. 6, p. 64270-64277, 2018. Disponível em: https://doi.org/10.1109/ACCESS.2018.2877890.

BEN-NUN, T. et al. A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, 2019. https://doi.org/10.48550/arXiv.1901.10183

BOUTHILLIER, X. et al. Accounting for variance in machine learning benchmarks. **Proceedings of Machine Learning and Systems**, v. 3, p. 747-769, 2021. Disponível em: https://proceedings.mlsys.org/paper_files/paper/2021/file/0184b0cd3cfb185989f858a1d9f5c1eb-Paper.pdf.

BOYD, K.; ENG, K. H.; PAGE, C. D. Area under the precision-recall curve: Point estimates and confidence intervals. In: **EUROPEAN CONFERENCE ON MACHINE LEARNING AND KNOWLEDGE DISCOVERY IN DATABASES (ECML PKDD)**, 2013, Prague. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases...** Berlin: Springer, 2013. p. 451-466. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-40994-3_29.

BUCKLAND, M.; GEY, F. The relationship between recall and precision. **Journal of the American Society for Information Science**, v. 45, n. 1, p. 12-19, 1994. Disponível em: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/(SICI)1097-4571(199401)45:1%3C12::AID-ASI2%3E3.0.CO;2-L.

CANO, A.; KRAWCZYK, B. Kappa Updated Ensemble for drifting data stream mining. **Machine Learning**, v. 109, p. 175–218, 2020. Disponível em: https://doi.org/10.1007/s10994-019-05840-z.

CANZIANI, A.; PASZKE, A.; CULURCIELLO, E. An analysis of deep neural network models for practical applications. **arXiv preprint**, arXiv:1605.07678, 2016. Disponível em: https://arxiv.org/abs/1605.07678.

CARUANA, R.; NICULESCU-MIZIL, A. An empirical comparison of supervised learning algorithms. In: **INTERNATIONAL CONFERENCE ON MACHINE LEARNING**, 23., 2006, Pittsburgh. **Proceedings...** New York: ACM, 2006. p. 161-168. Disponível em: https://dl.acm.org/doi/abs/10.1145/1143844.1143865.

CHEA, E.; KARKLIN, Y. 5 must-know AI concepts for fighting fraud. **Alloy**, 2025. Disponível em: https://www.alloy.com/blog/5-must-know-ai-concepts-for-fraud.

CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. In: **INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING**, 2016, Disponível em: https://dl.acm.org/doi/pdf/10.1145/2939672.2939785.

CHEN, Y. et al. Year-over-Year Developments in Financial Fraud Detection via Deep Learning: A Systematic Literature Review, 2025. Disponível em: https://arxiv.org/html/2502.00201v2.

CRANKSHAW, D. et al. Clipper: A low-latency online prediction serving system. In: **USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI)**, 2017. Disponível em: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw.

DAL POZZOLO, A. et al. Calibrating probability with undersampling for unbalanced classification. **IEEE Transactions on Knowledge and Data Engineering**, v. 35, n. 1, p. 32-45, jan. 2023. Disponível em: https://ieeexplore.ieee.org/document/7376606.

DATABRICKS. Databricks: A Data Lakehouse Company. **São Francisco**, 2024. Disponível em: https://www.databricks.com. Acesso em: 15 set. 2025.

DAVIS, J.; GOADRICH, M. The relationship between Precision-Recall and ROC curves. In: **INTERNATIONAL CONFERENCE ON MACHINE LEARNING**, 23.,

2006, Pittsburgh. New York: ACM, 2006. p. 233-240. Disponível em: https://dl.acm.org/doi/abs/10.1145/1143844.1143874.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, v. 27, n. 8, p. 861-874, 2006. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S016786550500303X.

GARCIA-MARTIN, E. et al. Estimation of energy consumption in machine learning. **Journal of Parallel and Distributed Computing**, v. 134, p. 75-88, 2019. Disponível em: https://www.sciencedirect.com/science/article/pii/S0743731518308773.

GÉRON, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd ed. Sebastopol, CA: O'Reilly Media, 2019.

GHOSH, S.; REILLY, D. L. Credit card fraud detection with a neural-network. In: **HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES**, 1994. Disponível em: https://ieeexplore.ieee.org/abstract/document/323314.

GRINSZTAJN, L.; OYALLON, E.; VAROQUAUX, G. Why do tree-based models still outperform deep learning on tabular data? In: Advances in Neural Information Processing Systems (NeurIPS), 2022, New Orleans. v. 35, p. 507-520. https://doi.org/10.48550/arXiv.2207.08815

HANCOCK, J.T, et al. Evaluating classifier performance with highly imbalanced Big Data. *J Big Data* 10, 42 (2023). https://doi.org/10.1186/s40537-023-00724-5

HAND, D. J.; HENLEY, W. E. Statistical classification methods in consumer credit scoring: a review. **Journal of the Royal Statistical Society: Series A (Statistics in Society)**, v. 160, n. 3, p. 523-541, 1997. Disponível em: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-985X.1997.00078.x.

HENDERSON, P. et al. Deep reinforcement learning that matters. In: **AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE**, 32., 2018, New Orleans. **Proceedings...** [S.l.]: AAAI, 2018. Disponível em: https://doi.org/10.1609/aaai.v32i1.11694.

HOWARD, A. G. et al. MobileNets: Efficient convolutional neural networks for mobile vision applications, 2017. Disponível em: https://arxiv.org/abs/1704.04861.

HUMMER, W. et al. ModelOps: Cloud-based lifecycle management for reliable and trusted AI. In: IEEE INTERNATIONAL CONFERENCE ON CLOUD ENGINEERING (IC2E), 2019. Disponível em: https://ieeexplore.ieee.org/abstract/document/8790192.

JOHNSON, M. el al. Comparative Study of Efficient Machine Learning Models for Real-Time Fraud Detection: CatBoost, XGBoost and LightGBM, 2025. DOI:10.21203/rs.3.rs-7539803/v1.

KARCZEWSKI, J. Machine learning models vs. rule-based systems in fraud prevention. **Mangopay**, 2023. Disponível em: https://blog.mangopay.com/en/home/machine-learning-models-vs-rule-based-systems-in-fraud-prevention-0.

KE, G. et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: **ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS)**, 2017. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

LIU, Z.; WANG, H.; CHEN, X. A Fusion Framework for Credit Card Fraud Detection: Stacking Ensemble with Explainable AI. Expert Systems with Applications, v. 235, p. 121-136, 2025. DOI: 10.1016/j.eswa.2024.121136.

MASTERCARD. Pesquisa da Mastercard revela as preferências por trás dos métodos de pagamento escolhidos no Brasil. **Mastercard**, 2024. Disponível em: https://www.mastercard.com/news/latin-america/pt-br/noticias/comunicados-de-imprensa/pr-pt/2024/marco/pesquisa-da-mastercard-revela-as-preferencias-por-tras-dos-metodos-de-pagamento-escolhidos-no-brasil/.

MENGHANI, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. **ACM Computing Surveys**, v. 55, n. 12, p. 1-37, 2023. Disponível em: https://dl.acm.org/doi/abs/10.1145/3578938.

MIENYE, I. D.; SUN, Y. A Deep Learning Ensemble With Data Resampling for Credit Card Fraud Detection. **IEEE Access**, v. 11, p. 30628-30638, 2023. Disponível em: https://ieeexplore.ieee.org/abstract/document/10081315.

PALEYES, A.; URMA, R. G.; LAWRENCE, N. D. Challenges in deploying machine learning: a survey of case studies. **ACM Computing Surveys**, v. 55, n. 6, p. 1-29, 2022. Disponível em: https://dl.acm.org/doi/full/10.1145/3533378.

PENCHIKALA, S. Big Data Processing with Apache Spark – Part 3: Spark Streaming and Machine Learning with MLlib. InfoQ, 2016. Disponível em: https://www.infoq.com/articles/apache-spark-streaming/

POWERS, D. M. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. **arXiv preprint**, arXiv:2010.16061, 2011. Disponível em: https://arxiv.org/abs/2010.16061. Acesso em: 17 set. 2025.

PROKHORENKOVA, L., et al. CatBoost: unbiased boosting with categorical features. In: Advances in Neural Information Processing Systems (NeurIPS), 2018, p. 6638–6648.

https://doi.org/10.48550/arXiv.1706.09516

RUDIN, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. **Nature Machine Intelligence**, v. 1, n. 5, p. 206-215, 2019. Disponível em: https://www.nature.com/articles/s42256-019-0048-x.

SAHIN, Y.; BULKAN, S.; DUMAN, E. A cost-sensitive decision tree approach for fraud detection. **Expert Systems with Applications**, v. 40, n. 15, p. 5916-5923, 2013. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0957417413003072.

SAHITHI, G; et al. Credit Card Fraud Detection using Ensemble Methods in Machine Learning. https://ieeexplore.ieee.org/document/9776955. **DOI:** 10.1109/ICOEI53556.2022.9776955.

SAITO, T.; REHMSMEIER, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. **PLoS ONE**, v. 10, n. 3, p. e0118432, 2015. Disponível em: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118432.

SCULLEY, D. et al. Hidden technical debt in machine learning systems. In: **ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS)**, 2015, Montreal. Red Hook: Curran Associates, 2015. p. 2503-2511. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html.

SCHWARTZ, R. et al. Green AI. **arXiv preprint**, arXiv:1907.10597, 2020. Disponível em: https://dl.acm.org/doi/abs/10.1145/3381831.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information Processing & Management**, v. 45, n. 4, p. 427-437, 2009. https://doi.org/10.1016/j.ipm.2009.03.002

SMITH, C. U.; WILLIAMS, L. G. Performance solutions: A practical guide to creating responsive, scalable software. **Boston: Addison-Wesley Professional**, 2002. Disponível em:https://www.researchgate.net/profile/Connie-Smith-10/publication/221446892_New_Book_-_Performance_Solutions_A_Practical_Guide_to_Creating_Responsive_Scalable_Software/links/6272a2512f9ccf58eb2cd9aa/New-Book-Performance-Solutions-A-Practical-Guide-to-Creating-Responsive-Scalable-Software.pdf.

STRUBELL, E.; GANEH, A.; MCCALLUM, A. Energy and policy considerations for deep learning in NLP. In: **ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS**, 57., 2019, Florence. ACL, 2019. https://doi.org/10.18653/v1/P19-1355

THE MOTLEY FOOL. Identity Theft & Credit Card Fraud Statistics. 2025. Disponível em: https://www.fool.com/money/research/identity-theft-credit-card-fraud-statistics/.

Acesso em: 15 set. 2025.

THEODORAKOPOULOS, L, et al. Big Data-Driven Distributed Machine Learning for Scalable Credit Card Fraud Detection Using PySpark, XGBoost, and CatBoost. MDPI Eletronics, 2025. https://doi.org/10.3390/electronics14091754

VAISHNAVI, T., et al. Detection of Credit Card Fraud Using Machine Learning. In: Proceedings of the 3rd International Conference on Optimization Techniques in the Field of Engineering, 2024. Disponível em: https://dx.doi.org/10.2139/ssrn.5089121

VAVILAPALLI, V. K. et al. Apache Hadoop YARN: Yet another resource negotiator. In: **SYMPOSIUM ON CLOUD COMPUTING**, 4., 2013, Santa Clara. **Proceedings...** New York: ACM, 2013. p. 1-16. Disponível em: https://doi.org/10.1145/2523616.2523633

YANG, T. J.; CHEN, Y. H.; SZE, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In: **IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)**, 2018, Salt Lake City. IEEE, 2018. p. 5687-5695. Disponível em:

https://openaccess.thecvf.com/content_cvpr_2017/html/Yang_Designing_Energy-Efficient_Convolutional_CVPR_2017_paper.html.

GÉRON, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd ed. Sebastopol, CA: O'Reilly Media, 2019

WAZLAWICK, R. Metodologia de Pesquisa para Ciência da Computação, Elsevier, 2009.

WORLD BANK GROUP. Credit card ownership (% age 15+). 2024. Disponível em:

https://genderdata.worldbank.org/en/indicator/fin7-t-a. Acesso em: 15 set. 2025.

WALLETHUB. Credit Card Fraud Statistics. 2024. Disponível em: https://wallethub.com/edu/cc/credit-card-fraud-statistics/25725. Acesso em: 15 set. 2025.

ZAHARIA, M. et al. Apache Spark: A Unified Engine for Big Data Processing. Communications of the ACM, 2016. https://doi.org/10.1145/2934664

APÊNDICES

APÊNDICE A – FUNÇÕES PARA DIVISÃO DETERMINÍSTICA

```
from pyspark.sql import functions as F
from pyspark.sql.window import Window


def create_deterministic_stratified_split(df, target_col="Class", test_ratio=0.2, seed=42):
    """
    cria uma divisão estratificada determinística usando hash estável
    garante que a mesma linha sempre vai para o mesmo conjunto
    """
    print("Iniciando divisão determinística...")


    # cria um identificador único estável para cada linha
    # usa colunas existentes para criar um hash determinístico
    df_with_id = df.withColumn(
        "row_hash",
        F.hash(F.concat_ws("|", *df.columns))
    )


    # adiciona um índice determinístico baseado no hash
    window = Window.orderBy("row_hash")
    df_with_index = df_with_id.withColumn("deterministic_index", F.row_number().over(window))


    # separa os dados em fraudes e normais
    fraud_df = df_with_index.filter(F.col(target_col) == 1)
    normal_df = df_with_index.filter(F.col(target_col) == 0)


    print(f"Transações normais: {normal_df.count():,}")
    print(f"Transações fraudulentas: {fraud_df.count():,}")
```

```
    # função para divisão determinística baseada no índice
    def split_deterministic(df, ratio, seed):
        total_count = df.count()
        test_count = int(total_count * ratio)
        # ordenar deterministicamente e pegar os primeiros para teste
        test_set = df.orderBy("deterministic_index").limit(test_count)
        # o restante vai para treino
        train_set = df.join(test_set, "deterministic_index", "left_anti")
        return train_set, test_set


    # divide cada grupo
    fraud_train, fraud_test = split_deterministic(fraud_df, test_ratio,
seed)
    normal_train, normal_test = split_deterministic(normal_df,
test_ratio, seed)
    # combina os resultados e remove colunas auxiliares
    train_set = normal_train.union(fraud_train).drop("row_hash",
"deterministic_index")
    test_set = normal_test.union(fraud_test).drop("row_hash",
"deterministic_index")
    # valida as proporções
    train_fraud_rate = train_set.filter(F.col(target_col) == 1).count() /
train_set.count()
    test_fraud_rate = test_set.filter(F.col(target_col) == 1).count() /
test_set.count()
    original_fraud_rate = df.filter(F.col(target_col) == 1).count() /
df.count()
    print(f"\nValidação da divisão:")
    print(f"Fraude original: {original_fraud_rate:.4f}")
    print(f"Fraude treino:   {train_fraud_rate:.4f}")
    print(f"Fraude teste:    {test_fraud_rate:.4f}")
    print(f"Divisão determinística concluída!")
    return train_set, test_set
```

APÊNDICE B – FUNÇÃO PARA CALCULAR *SCORE* PREDITIVO COMBINADO COM PESOS

```
weights = {
    "roc_auc": 0.2,
    "avg_precision": 0.25,
    "precision": 0.15,
    "recall": 0.2,
    "f1_score": 0.2
}


def calculate_weighted_score(row, weights):
    metrics = {
        "roc_auc": row.get("roc_auc", 0),
        "avg_precision": row.get("pr_auc", 0),
        "precision": row.get("precision", 0),
        "recall": row.get("recall", 0),
        "f1_score": row.get("f1_score", 0)
    }
    normalized_metrics = {k: (v if v is not None else 0) for k, v in
metrics.items()}
    return sum(normalized_metrics[k] * weights[k] for k in metrics.keys())


all_predictive_results["weighted_score"] =
all_predictive_results.apply(lambda row: calculate_weighted_score(row,
weights), axis=1)
```

APÊNDICE C – FUNÇÃO PARA CALCULAR MÉTRICAS DE EFICIÊNCIA COMPUTACIONAL COM NÚMERO ‘X’ DE *RUNS*

```
import time
import psutil
import os

n_runs = 10

def run_efficiency_test(model, X):
    inference_times = []
    memory_usages = []
    cpu_percents = []
    num_threads_list = []

    for _ in range(n_runs):
        start_time = time.time()
        _ = model.predict(X)
        inference_time = time.time() - start_time
        inference_times.append(inference_time)

        process = psutil.Process(os.getpid())
        memory_usages.append(process.memory_info().rss / (1024 * 1024))
        cpu_percents.append(psutil.cpu_percent(interval=0.1))
        num_threads_list.append(process.num_threads())

    return {
        "inference_time_sec": np.mean(inference_times),
        "memory_usage_mb": np.mean(memory_usages),
        "cpu_percent": np.mean(cpu_percents),
        "num_threads": np.mean(num_threads_list)
    }
```

APÊNDICE D – FUNÇÃO PARA CALCULAR *SCORE* DE EFICIÊNCIA COMPUTACIONAL

```
# valores de normalização e pesos dados as métricas podem e devem ser alterados de acordo com o caso e o que se deseja medir

# definimos os valores abaixo considerando o contexto de detecção de fraude em cartão de crédito, onde é necessário equilibrar tempo de resposta rápido (inferência < 5s), uso moderado de memória (< 1GB), baixo consumo de CPU (< 100%) e número de threads (< 16), priorizando agilidade e escalabilidade em ambiente produtivo.


# normalização dos valores de eficiência computacional
max_inference_time = 5.0  # segundos
max_memory_usage = 1024.0  # MB
max_cpu_percent = 100.0
max_num_threads = 16.0

def compute_efficiency_score(row):
    inference_time = row.get("inference_time_sec", 0)
    memory_usage_mb = row.get("memory_usage_mb", 0)
    cpu_percent = row.get("cpu_percent", 0)
    num_threads = row.get("num_threads", 0)
    score = (
        (1 - min(inference_time / max_inference_time, 1)) * 0.4 +
        (1 - min(memory_usage_mb / max_memory_usage, 1)) * 0.4 +
        (1 - min(cpu_percent / max_cpu_percent, 1)) * 0.1 +
        (1 - min(num_threads / max_num_threads, 1)) * 0.1
    )
    return score
```

APÊNDICE E – *NOTEBOOKS* DE APOIO DAS EXPERIMENTAÇÕES E DESENVOLVIMENTO DO PROJETO

Os notebooks de apoio das experimentações realizadas e desenvolvimento do projeto no Databricks estão disponíveis no Github, através do link:

https://github.com/devleticiastahl/fraud-detection-comparative-analysis