VIII Simpósio de Iniciação Científica 2025

"Do conhecimento acadêmico à transformação sustentável: inovação com validação científica"

SISTEMA DE DETECÇÃO DE FRAUDES BANCÁRIAS

André Marcos Leifeld Raicoski[1] (EG), Egon Luiz Muller Júnior (PQ)[1]

[1]*Universidade Federal de Itajubá.*

Palavras-chave: CatBoost. Detecção de Fraudes. LightGBM. Machine Learning. XGBoost.

Introdução

As transações financeiras eletrônicas tornaram-se onipresentes na sociedade moderna, facilitando o comércio e as atividades econômicas em escala global. No entanto, essa conveniência é acompanhada por um aumento proporcional nos riscos de atividades fraudulentas. Fraudes em transações bancárias resultam em perdas financeiras substanciais para consumidores e instituições financeiras, além de minar a confiança no sistema financeiro digital (BOLTON; HAND, 2002).

A detecção precoce e precisa de transações fraudulentas é, portanto, uma prioridade crítica. Métodos tradicionais baseados em regras fixas são frequentemente inadequados para lidar com a natureza dinâmica e adaptativa das táticas de fraude (SANTOS, 2023). Nesse contexto, técnicas de Machine Learning emergiram como uma abordagem poderosa, capaz de aprender padrões complexos a partir de grandes volumes de dados transacionais e identificar anomalias que podem indicar fraude. Um desafio particular no desenvolvimento de sistemas de detecção de fraude reside na natureza dos dados bancários. Muitas informações são confidenciais e, por razões de privacidade e segurança, os conjuntos de dados disponíveis para pesquisa frequentemente contêm features anonimizadas ou transformadas, cuja semântica exata não é totalmente conhecida. Isso impõe limitações à interpretabilidade dos modelos e à engenharia de features.

Este trabalho visa enfrentar esse desafio por meio do desenvolvimento e análise de modelos de Machine Learning para a detecção de fraudes bancárias. O estudo compreende duas fases principais: primeiramente, uma análise comparativa de diferentes algoritmos de Machine Learning (XGBoost, LightGBM e CatBoost) utilizando um conjunto de dados inicial para identificar as abordagens mais promissoras. Em seguida, o desenvolvimento e otimização de um modelo final, utilizando um conjunto de dados mais robusto, com o objetivo de maximizar a capacidade de detecção. Adicionalmente, um serviço de backend (API - Application Programming Interface) foi implementado para demonstrar a aplicabilidade prática do modelo treinado. A relevância deste trabalho reside na contribuição para o avanço das técnicas de detecção de fraude, oferecendo uma análise prática de algoritmos e uma solução implementável, considerando as restrições impostas pela confidencialidade dos dados.

Metodologia

O conjunto de dados incialmente utilizado foi retirado do Kaggle, o conjunto "Credit Card Fraud Detection". Para os testes com a API funcionando, foi utilizado o conjunto "IEEE-CIS Fraud Detection".

O motivo da mudança de dataset durante o desenvolvimento foi estratégico: inicialmente, utilizou-se um conjunto de dados menor para a fase de comparação dos algoritmos, permitindo uma avaliação rápida e eficiente do comportamento de cada modelo, por isso, inicialmente utilizou-se o conjunto de dados "Credit Card Fraud Detection". Posteriormente, para o desenvolvimento do modelo final, optou-se pelo dataset "IEEE-CIS Fraud Detection", significativamente maior e mais detalhado, contendo mais de 590.000 transações.

Esta abordagem em duas fases permitiu otimizar o tempo de desenvolvimento, evitando o processamento desnecessário do dataset maior durante a fase inicial de experimentação com diferentes algoritmos.

O pré-processamento dos dados foi a etapa seguinte, onde incluiu-se:

- Carregamento e Amostragem de Dados: Para gerenciar o uso de memória e garantir a representatividade da classe minoritária (fraudes), os dados foram carregados em amostras. Foi aplicada uma estratégia de amostragem que considerou a proporção de fraudes, com undersampling da classe majoritária (não-fraudes) para obter um conjunto de treinamento mais balanceado.
- Fusão de Dados: Os arquivos de transação e identidade foram combinados utilizando a coluna TransactionID.
- Tratamento de Features Categóricas: As features categóricas foram identificadas e convertidas para representação numérica utilizando

LabelEncoder. Foi tomado cuidado para ajustar o encoder com a combinação dos dados de treino e teste para evitar categorias desconhecidas durante a inferência.

- Tratamento de Valores Ausentes: Valores ausentes(NaN) foram preenchidos com um valor placeholder (-999), uma estratégia comum em modelos baseados em árvores que conseguem lidar nativamente com esses valores ou aprender a sua significância.
- Seleção de Features: Colunas consideradas desnecessárias, como TransactionID após a fusão, foram removidas.

As colunas de features utilizadas no treinamento do modelo final foram salvas para garantir consistência durante a inferência na API.

Após o pré-processamento dos dados, foram explorados e comparados diferentes algoritmos de ML. Os principais algoritmos avaliados incluíram:

- CatBoost;
- LightGBM;
- XGBoost.

Um script também foi realizado para explorar combinações desses modelos. Esta fase teve como objetivo identificar o algoritmo ou a combinação de algoritmos com maior potencial para o problema em questão. Os resultados visuais desta comparação são apresentados a seguir.

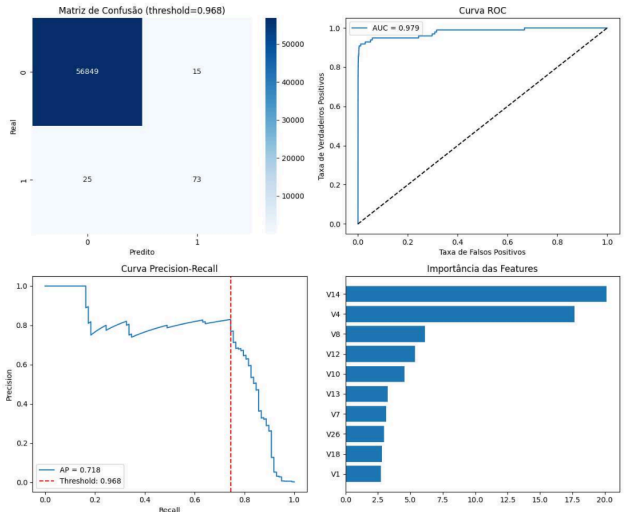

Figura 1 – Resultados do algoritmo CatBoost

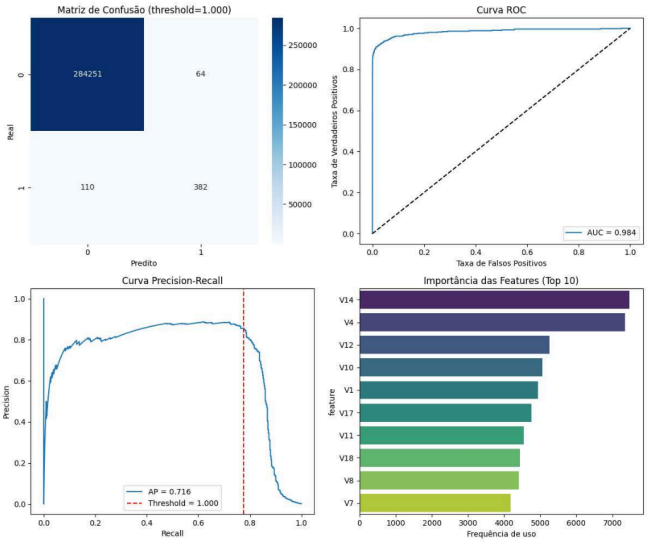

Figura 2 – Resultados do algoritmo LightGBM

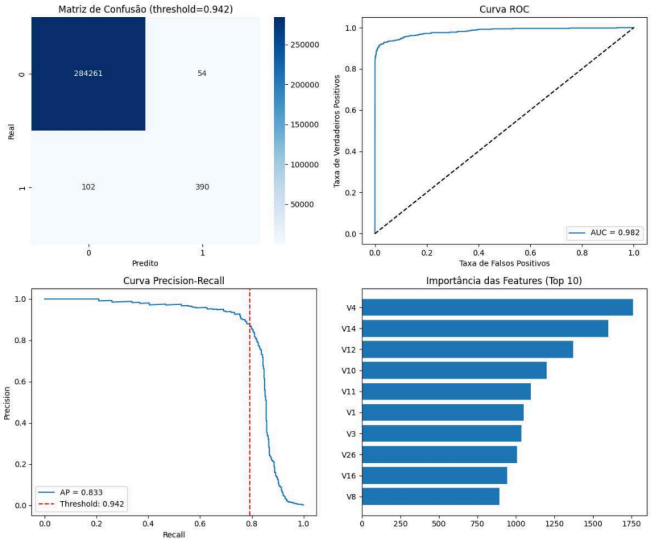

Figura 3 – Resultados do algoritmo XGBoost

Após uma análise comparativa dos algoritmos, XGBoost foi selecionado para o desenvolvimento do modelo final, principalmente pela maior precisão obtida, além de possuir um valor na curva AUC muito próximo dos demais algoritmos. As principais considerações no desenvolvimento deste modelo foram (CHEN; GUESTRIN, 2016):

- Configuração do Modelo: Foram definidos hiperparâmetros específicos para o XGBoost, como 'objective' ('binary:logistic'), 'eval_metric' ('auc'), 'learning_rate', 'max_depth', 'n_estimators', 'subsample', 'colsample_bytree', e 'scale_pos_weight'. O parâmetro 'scale_pos_weight' foi calculado com base no desbalanceamento das classes para dar

maior importância à classe minoritária (fraudes).
- Validação Cruzada: Foi utilizada validação cruzada estratificada (‘StratifiedKFold‘) com 5 folds para treinar o modelo e obter estimativas robustas de seu desempenho (OOF - Out-of-Fold).
- Treinamento e Early Stopping: O modelo foi treinado em cada fold com um mecanismo de ‘early_stopping_rounds‘ para evitar o overfitting e otimizar o número de árvores.
- Otimização de Threshold: Após obter as previsões OOF, foi realizada uma otimização do threshold de classificação utilizando a curva Precision-Recall. O threshold que maximizava o F1-Score foi selecionado para converter as probabilidades preditas em classificações binárias (fraude/não-fraude).

O modelo final treinado com todos os dados de treinamento (utilizando os parâmetros otimizados) foi salvo no formato JSON nativo do XGBoost e também como um wrapper PyTorch para facilitar sua integração com a API. O threshold otimizado e as colunas de features também foram salvos usando ‘pickle‘.

Para demonstrar a aplicabilidade prática do modelo treinado, foi desenvolvida uma API de backend. Esta API, construída com Node.js e Python, permite que novas transações sejam enviadas para o modelo, que retorna uma predição de fraude. O script ‘predict.py‘ carrega o modelo XGBoost salvo, o threshold otimizado e as colunas de features para realizar as previsões.

A arquitetura da API foi implementada seguindo princípios RESTful, utilizando Express.js como framework web. O servidor foi estruturado com os seguintes endpoints principais:
- GET /api/health: Endpoint de verificação de saúde que monitora a disponibilidade do modelo e seus arquivos auxiliares.
- POST /api/predict: Endpoint principal que recebe dados de transações e retorna predições de fraude.

A comunicação entre o servidor Node.js e o modelo Python é realizada através de um processo filho (child process), onde o servidor Node.js inicia um processo Python para cada predição, passando os dados da transação como argumentos. Esta abordagem garante isolamento e robustez, evitando que problemas no processamento do modelo afetem o servidor principal.

Para testar a API, foi desenvolvido um cliente de teste abrangente que simula diversos cenários de transações, incluindo 10 casos de teste de fraude com características específicas (horários suspeitos, valores altos, padrões anômalos) e 10 casos de teste de transações legítimas com padrões normais.

O cliente de teste não apenas verifica a funcionalidade da API, mas também calcula métricas de desempenho como precisão, recall e matriz de confusão para os casos de teste, permitindo uma avaliação contínua da qualidade das predições em um ambiente simulado.

O desempenho dos modelos foi avaliado utilizando um conjunto de métricas padrão para problemas de classificação, especialmente relevantes em cenários de dados desbalanceados:
- ROC-AUC: Mede a capacidade do modelo de distinguir entre as classes.
- Precisão: Proporção de transações classificadas como fraude que são de fato fraudes.
- Recall: Proporção de fraudes reais que foram corretamente identificadas pelo modelo.
- F1-Score: Média harmônica entre Precisão e Recall, útil para balancear ambos.
- Matriz de Confusão: Tabela que visualiza o desempenho do classificador, mostrando verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.
- Curva Precision-Recall: Visualiza o trade-off entre Precisão e Recall para diferentes thresholds.

Resultados e discussão

O modelo XGBoost final, treinado com o dataset "IEEE Fraud Detection" e otimizado conforme descrito na metodologia, demonstrou uma capacidade significativa na detecção de fraudes. As métricas de desempenho alcançadas, com o threshold otimizado, foram:
- AUC Geral (OOF): O valor de AUC obtido nas predições out-of-fold foi um indicador primário da performance do modelo antes da otimização do threshold.
- Threshold Otimizado: Um threshold de aproximadamente 0.2965 (valor exato pode variar ligeiramente entre execuções devido à natureza da otimização e dos dados) foi encontrado para maximizar o F1-Score.
- Métricas com Threshold Otimizado:
  - Acurácia: 90.9%;
  - Precisão (para classe fraude): 85.3%(significa que quando o modelo

prevê fraude, está correto em 85.3% dos casos);
- Recall/Sensibilidade (para classe fraude): 77.0%(o modelo conseguiu detectar 77.0% de todas as fraudes reais);
- F1-Score (para classe fraude): 81.0%;

Comparativamente, houve um aumento drástico no recall (de 38.7% para 77.0% em relação a uma configuração anterior de um threshold padrão de 0.5), dobrando a quantidade de fraudes detectadas. Isso indica que a otimização do threshold e possivelmente os ajustes nos hiperparâmetros do modelo foram eficazes em melhorar a detecção da classe minoritária, que é o principal objetivo em sistemas de detecção de fraude.

Os resultados obtidos demonstram a eficácia do modelo XGBoost em identificar transações fraudulentas no conjunto de dados IEEE-CIS. A melhoria significativa no recall é particularmente importante, pois significa que uma proporção maior de fraudes reais está sendo capturada, o que é crucial para minimizar perdas financeiras.

O desafio imposto pela confidencialidade dos dados, com muitas features anonimizadas, limitou a capacidade de realizar uma engenharia de features mais aprofundada e de interpretar completamente o comportamento do modelo. No entanto, mesmo sem o conhecimento completo do significado de todas as features, o modelo foi capaz de aprender padrões preditivos eficazes.

Os testes realizados com a API através do cliente de teste demonstraram a robustez e eficácia do sistema em um ambiente simulado, sendo apresentada uma performance consistente com o modelo treinado, mantendo níveis similares de precisão e recall. A latência média de resposta ficou abaixo de 200ms por requisição, demonstrando a viabilidade do sistema para uso em tempo real.

Um aspecto notável foi a capacidade do sistema em lidar com diferentes padrões de fraude, identificando corretamente transações suspeitas mesmo quando apresentadas com combinações de características não vistas durante o treinamento. Isso sugere que o modelo manteve boa capacidade de generalização quando implantado na API.

Na prática, a interação com a API de inferência se daria através de uma requisição HTTP POST para o endpoint /api/predict. O corpo da requisição esperaria um objeto JSON contendo as features da transação a ser analisada.

Conclusões

Este estudo reforça o valor do Machine Learning como uma ferramenta poderosa na luta contra fraudes financeiras. Para trabalhos futuros, sugere-se a exploração de técnicas mais avançadas de engenharia de features (especialmente se mais informações sobre os dados puderem ser obtidas), a investigação de algoritmos de deep learning, a implementação de sistemas de detecção em tempo real com monitoramento de concept drift (mudanças nos padrões de fraude ao longo do tempo) e a análise mais aprofundada da interpretabilidade dos modelos para fornecer insights acionáveis às instituições financeiras.

Agradecimentos

Os autores gostariam de agradecer à Universidade Federal de Itajubá pela oportunidade de desenvolver este trabalho e a FNDE.

Referências

BAHNSEN, A. C. et al. Feature engineering strategies for credit card fraud detection. Expert Systems with Applications, Elsevier, v. 51, p. 134–142, 2016.

BAUDER, R. A.; KHOSHGOFTAAR, T. M. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. Health Information Science and Systems, Springer, v. 6, n. 1, p. 1–15, 2018.

BHATTACHARYYA, S. et al. Data mining for credit card fraud: A comparative study. Decision Support Systems, Elsevier, v. 50, n. 3, p. 602–613, 2011.

BOLTON, R. J.; HAND, D. J. Statistical fraud detection: A review. Statistical Science, Institute of Mathematical Statistics, v. 17, n. 3, p. 235–255, 2002.

CARCILLO, F. et al. Combining unsupervised and supervised learning in credit card fraud detection. Information Sciences, Elsevier, v. 557, p. 317–331, 2019.

CARCILLO, Fabrizio et al. A survey of industrial and academic research in fraud detection. ACM Computing Surveys (CSUR), ACM New York, NY, v. 54, n. 2, p. 1–36, 2021.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: PROCEEDINGS of the 22nd ACM SIGKDD International Conference on Knowledge.

Discovery and Data Mining. [S.l.: s.n.], 2016. P. 785–794.

SANTOS, L. A. P. Aprendizado de máquina aplicado na detecção de fraudes em cartão de crédito. 2023. Universidade Federal de Ouro Preto, Ouro Preto.