Arquitetura de Tempo Real e Modelo de Aprendizado de Máquina para Detecção de Fraudes de Cartão de Crédito

Robson S. Santos[1], Robesvânia Araújo[1], Paulo A. L. Rego[1], José M. da S. M. Filho[1], Jarélio G. da S. Filho[2], José D. C. Neto[2], Nicksson C. A. de Freitas[2], Emanuel B. Rodrigues[1]

[1]Universidade Federal do Ceará (UFC)
Av. Humberto Monte, s/n, Pici - CEP 60440-593 – Fortaleza – CE – Brasil

[2]Sidi
Av. República do Líbano, 251 - CEP 51110-160 – Recife – PE – Brasil

{robson.santos, robesvania.araujo}@alu.ufc.br,
{paulo, monteiro}@dc.ufc.br, j.filho@sidi.org.br
{j.carneiro, nicksson.a}@sidi.org.br, emanuel@dc.ufc.br

Abstract. *Faced with the growing need for safe and efficient systems to detect fraud in credit card transactions in real time, this study proposes and evaluates a fraud detection approach based on machine learning, integrated into a real-time architecture of a virtual bank . The multilayer architecture used allows a clear division of responsibilities, resulting in optimized and efficient operations. With the use of the Random Forest machine learning model, the system is able to make accurate predictions about possible fraudulent transactions. This implementation provides robust and reliable results, with an accuracy of 99,98%, a precision of 99,97%, a recall of 100% and an F1-score of 99,88%, highlighting the potential of machine learning to significantly improve the detection effectiveness of cheats.*

Resumo. *Diante da necessidade crescente de sistemas seguros e eficientes para detectar fraudes em transações de cartões de crédito em tempo real, este estudo propõe e avalia uma abordagem de detecção de fraude baseada em aprendizado de máquina, integrada a uma arquitetura em tempo real de um banco virtual. A arquitetura multicamadas utilizada permite uma clara divisão de responsabilidades, resultando em operações otimizadas e eficientes. Com o uso do modelo de aprendizado de máquina Random Forest, o sistema é capaz de fazer previsões precisas sobre possíveis transações fraudulentas. Essa implementação fornece resultados robustos e confiáveis, com acurácia de 99,98%, precisão de 99,97%, revocação de 100% e F1-score de 99,88%, destacando o potencial do aprendizado de máquina para melhorar significativamente a eficácia da detecção de fraudes.*

1. Introdução

O setor financeiro global está passando por uma transformação notável, impulsionada principalmente pelo avanço dos pagamentos digitais. Este fenômeno é consequência da demanda crescente por soluções de pagamento convenientes e da contínua introdução de tecnologias modernas e inovadoras que estão reformulando a condução das transações financeiras [Cherif et al. 2022].

Em virtude da adoção crescente de serviços digitais e aplicativos de pagamento, as fraudes em operações financeiras tornaram-se ainda mais comuns [Bakhtiari, Nasiri e Vahidi 2023]. Isso elevou os riscos securitários, tornando-os uma preocupação premente para bancos e operadoras de cartões de crédito. A ocorrência de transações fraudulentas em cartões de crédito representa um desafio significativo para as instituições financeiras, resultando em perdas consideráveis.

Neste contexto, a rápida e efetiva detecção torna-se fundamental para evitar prejuízos financeiros. Contudo, a detecção de fraudes em transações de cartões de crédito apresenta inúmeros desafios de segurança, como, por exemplo: i) preocupações com a privacidade, uma vez que grandes volumes de dados dos usuários são utilizados para gerar os modelos preditivos, ii) vulnerabilidade a ataques *Distributed Denial of Service (DDoS)*, os quais podem interromper os sistemas de detecção [Cherif et al. 2022], iii) necessidade de dados rotulados e iv) falta de conhecimento e compreensão dos padrões e tendências das transações, que mudam constantemente [Chen e Han 2021]. Portanto, é imprescindível desenvolver soluções de detecção de fraude robustas, que funcionem em tempo real e que, ao mesmo tempo, busquem assegurar a privacidade de clientes e organizações.

Neste trabalho, propomos e avaliamos uma abordagem para detecção de fraudes em transações com cartões de crédito. Esta abordagem, baseada em algoritmos de aprendizado de máquina, pré-processamento dos dados e engenharia de atributos, foi projetada para identificar fraudes robustamente e em tempo real. Parte integrante deste processo foi a aplicação do modelo de aprendizado de máquina *Random Forest*, que se destacou em nossos testes, na arquitetura de tempo real de um banco virtual. A análise proposta apresentou resultados superiores ao estado da arte, com acurácia de 99,98%, precisão de 99,97%, revocação de 100% e *F1-score* de 99,88%. Este marco representa um passo importante na detecção de fraudes em tempo real, abrindo caminho para aplicações mais seguras e eficientes na indústria financeira.

Este artigo é estruturado da seguinte maneira: inicialmente, na Seção 2, discutimos os principais trabalhos relacionados. Em seguida, na Seção 3, detalhamos nossa abordagem proposta para a detecção de fraudes em cartões de crédito, além de discutirmos os experimentos realizados e os resultados obtidos. Na Seção 4, apresentamos a arquitetura de tempo real utilizada para a detecção de fraudes, bem como a implementação, implantação e os resultados dos experimentos realizados. Finalmente, na Seção 5, concluímos o artigo, apresentando nossas conclusões e possíveis direções para trabalhos futuros.

2. Trabalhos Relacionados

No estudo [Kewei et al. 2021], técnicas de aprendizado profundo e engenharia de atributos foram combinadas para detectar fraudes em transações online. A adição de novos atributos com base em medidas estatísticas e informações temporais resultou em um modelo mais eficiente, reduzindo a dimensionalidade do conjunto de dados original de 435 atributos.

O estudo em [Nguyen et al. 2022] apresenta um modelo híbrido usando CatBoost e Rede Neural Profunda para detectar fraudes, distinguindo entre usuários novos e antigos. O escopo do trabalho se concentrou na etapa de pré-processamento e na engenharia de atributos, com a atenção voltada para os dados pertinentes às transações, como valor, data

e horário, provenientes de clientes já conhecidos, além dos atributos ligados aos cartões, como tipo e país, quando aplicáveis aos novos usuários. Os resultados confirmaram a eficácia do modelo em termos de precisão e velocidade.

Em [Chen e Han 2021], os autores introduzem um método inovador de detecção de fraudes usando CatBoost. Esse método emprega pré-processamento e engenharia de atributos, criando novas características a partir de contagens, medidas estatísticas e valores extremos derivados dos dados originais. Para lidar com o tamanho do conjunto de dados, técnicas de compressão foram aplicadas para reduzir a memória durante o treinamento do modelo preditivo.

O estudo em [Bakhtiari, Nasiri e Vahidi 2023] apresentou resultados promissores na detecção de fraudes em transações de cartões de crédito, utilizando uma combinação de algoritmos *LightGBM* e *LiteMORT* após engenharia de atributos. A avaliação incluiu várias métricas, como AUC-ROC, precisão, revocação, F1-score e acurácia. Os modelos que passaram pela engenharia de atributos demonstraram desempenho consideravelmente melhor em comparação aos que não passaram.

O artigo [Abakarim, Lahby e Attioui 2018] ressalta a importância da predição em tempo real na detecção de fraudes em cartões de crédito, com destaque para o uso de *autoencoders* de aprendizado profundo para identificar transações fraudulentas através da detecção de anomalias. Essa abordagem em tempo real permite respostas rápidas das instituições financeiras contra fraudes, protegendo os clientes e minimizando perdas. O modelo proposto apresenta detalhes arquiteturais e metodológicos para abordar o fluxo de operações na detecção de fraudes, enfatizando o emprego de modelos de aprendizado de máquina.

O sistema proposto por [Sapozhnikova et al. 2017] tem um escopo amplo, visando a coleta e análise de dados do ambiente do usuário para combater fraudes. Ele utiliza informações como endereço IP, tamanho de tela e fontes instaladas para melhorar a precisão na identificação do usuário, e emprega análise de *cluster* para detectar padrões suspeitos. Em contraste, nossa arquitetura simula um banco virtual, não apenas coletando informações, mas também realizando a detecção em tempo real através dos serviços de Orquestração, Predição e Transação, ampliando a segurança na monitorização de transações.

O artigo [Menshchikov et al. 2022] investiga otimizações arquiteturais em sistemas anti-fraude baseados em Big Data, incorporando inteligência artificial para velocidade, escalabilidade e resiliência. Ele utiliza ferramentas Apache para processar fluxos de dados em tempo real e detectar fraudes eficientemente. Em contrapartida, nossa aplicação se dedica à detecção de fraudes em cartões de crédito, com ênfase no aprendizado de máquina em uma abordagem específica.

A metodologia de [Prusti, Das e Rath 2021] utiliza um banco de dados em grafo com Neo4j para detectar fraudes em cartões de crédito, incorporando características do grafo e dados de transações para melhorar a precisão. Vários algoritmos de aprendizado de máquina são aplicados para avaliar o desempenho. Por outro lado, nossa abordagem se diferencia ao combinar algoritmos de predição e automação, resultando em um modelo mais abrangente e robusto para a avaliação completa dos dados.

O estudo [Thennakoon et al. 2019] apresenta um sistema de detecção de fraudes

em cartões de crédito que combina aprendizado de máquina e análise preditiva para melhorar a eficácia na detecção. O sistema identifica padrões de transações fraudulentas e reage em tempo real a atividades suspeitas, além de possuir um módulo de aplicação e interface gráfica para suportar investigações. Apesar das semelhanças com nossa arquitetura, nossa proposta é mais abrangente, com análise preditiva, módulo de detecção em tempo real e seleção específica de algoritmos como *Logistic Regression* e *Decision Tree*, garantindo maior eficiência e resposta imediata.

O SCARFF é uma estrutura escalável em tempo real para detecção de fraudes, integrando ferramentas de Big Data (Kafka, Spark e Cassandra) e usando aprendizado de máquina para abordar desafios de desequilíbrio e não-estacionariedade em dados de *streaming* de fraudes em cartões de crédito. [Carcillo et al. 2018] avalia sua arquitetura e precisão. Por contraste, nossa aplicação prioriza o aprendizado de máquina e fluxos de dados, com foco em técnicas avançadas como redes neurais, evidenciando resultados detalhados que comprovam eficácia.

Vale destacar que a maioria dos trabalhos encontrados na literatura exploraram tanto o pré-processamento dos dados quanto a engenharia de atributos. Além disso, essas atividades impactaram significativamente o desempenho dos modelos avaliados. Na Tabela 1 é possível ver a comparação entre os trabalhos relacionados.

Tabela 1. Tabela de Comparação dos Trabalhos Relacionados

Artigos	Dataset	Modelo/Algoritmo	Aplicação/Ferramentas	Foco em cartão de crédito
Kewei et al.	IEEE-CIS Fraud Detected	NB, SVM & DL	✗	✓
Nguyen et al.	IEEE-CIS Fraud Detected	*Catboost* & DNN	✗	✓
Chen et al.	IEEE-CIS Fraud Detected	*Catboost*	✗	✓
Bakhtiari	IEEE-CIS Fraud Detected	*LightGBM* & *LiteMORT*	✗	✓
Abakarim	European Card	SVM, LR, ANN & DNN	Apache Kafka, TensorFlow & MemSQL	✓
Sapozhnikova	UCSD_FICO	MLP, SVM, RFC	✗	✗
Menshchikov	Credit Card Transactions Fraud Detection Dataset	LR, KNN, GNB, DT, MLP, RF AdaBoost, GradBoost, Bagging&XGBoost	Apache Kafka, ApacheSpark, ApacheCassandra PyFlink, Elasticsearch&Kibana	✓
Prusti	BankSim	DT, RF, KNN, MLP, VSM, LOF&LF	Neo4j	✓
Thennakoon	✗	SVM,NB, KNN &LR	✗	✓
Carcillo	Dataset DS	✗	Apache Kafka, ApacheSpark, ApacheCassandra&SCARFF	✓
Esta Proposta	IEEE-CIS Fraud Detected	LR, DT, KNN & RF&XGBoost	FastAPI, Docker, MongoDB Gunicorn&Locust	✓

3. Uma Abordagem para Detecção de Fraudes

Nesta seção, a abordagem concebida para detecção de fraudes em transações de cartões de crédito é detalhada. A Figura 1 ilustra a arquitetura da abordagem proposta. A seguir, iremos discutir cada um dos componentes desta arquitetura. As atividades de pré-processamento dos dados são apresentadas na Seção 3.1, a seleção de atributos na Seção 3.2, e por fim, a modelagem do *pipeline* de geração do modelo preditivo na Seção 3.3.

3.1. O Conjunto de Dados Utilizado

Este trabalho utiliza o conjunto de dados denominado *IEEE-CIS Fraud Detection*[1], o qual foi elaborado para possibilitar a construção e a avaliação de sistemas de detecção

[1]https://www.kaggle.com/competitions/ieee-fraud-detection

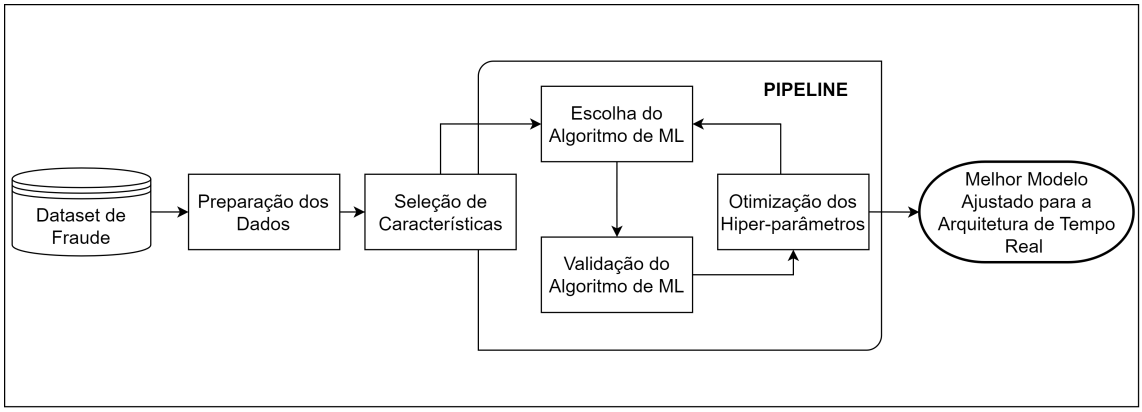

Figura 1. Arquitetura da Abordagem Proposta para Detecção de Fraudes.

de fraudes em comércio eletrônico. Esse conjunto de dados é formado por duas tabelas principais: a tabela de transações e a tabela de identidades. Na tabela de transações estão contidas informações cruciais sobre cada transação, incluindo o valor da transação, o cartão de crédito utilizado, a data e a hora da operação, além de outros atributos relevantes. Na tabela de identidades são fornecidos dados adicionais sobre os indivíduos envolvidos nas transações, incluindo características demográficas e informações sobre os dispositivos utilizados pelos usuários. O conjunto de dados *IEEE-CIS Fraud Detection* inclui 434 atributos, dos quais 400 são anonimizados (V1 a V339), a fim de proteger a privacidade de usuários e instituições. Cada linha da tabela de transações representa uma determinada transação eletrônica. Cada transação possui um rótulo, o qual pode assumir dois valores indicando: transação legítima ou transação fraudulenta. Neste sentido, o conjunto de dados pode ser utilizado para construir classificadores binários. Como o conjunto de dados *IEEE-CIS Fraud Detection* é bastante grande e desbalanceado, utilizamos apenas uma parte dos dados, mais precisamente um subconjunto formado pelo *train_identity* e o *train_transaction*, formando assim um novo conjunto de dados, o qual será utilizado nas etapas subsequentes desta pesquisa.

3.2. Preparação dos Dados

Como o conjunto de dados inclui dezenas de colunas, muitos valores ausentes, além de atributos alfa-numéricos, foi necessária uma etapa de pré-processamento de dados. Esta etapa incluiu diversas atividades, tais como: i) remoção de colunas com muitos dados ausentes, ii) imputação de valores faltantes (neste caso, utilizamos a mediana computada para o atributo), iii) remoção de linhas duplicadas, iv) codificação dos valores dos atributos categóricos alfa-numéricos em valores numéricos e v) agrupamento de atributos. Após esta etapa, o conjunto de dados, que originalmente continha 434 atributos, passou a ter 224 atributos.

3.3. Seleção de Atributos

Mesmo após a preparação dos dados, o conjunto de dados ainda apresentava uma alta dimensionalidade, com 224 atributos, o que pode influenciar o tempo de treinamento e o desempenho dos modelos preditivos. Além disso, alguns desses atributos podem ser irrelevantes ou redundantes para a tarefa de detecção de fraudes. Desta forma, uma etapa de seleção de atributos torna-se fundamental.

Uma das abordagens comuns para determinar a importância de características é o uso de árvores de decisão. Neste sentido, empregamos uma técnica de seleção de atributos baseada na importância dos atributos na divisão dos nós em uma árvore de decisão [2]. O limiar de importância de 0,01 estabelece um critério de seleção de características a partir dos dados iniciais. O algoritmo de Árvore de Decisão é usado para calcular a importância de cada característica dos dados, com isso, foi possível definir quais características têm a maior influência no modelo. Neste caso, um limiar de 0,01 é definido. Isso significa que estamos interessados apenas nas características, com uma importância maior que 0,01. Ao fazer isso, podemos excluir características menos importantes e, potencialmente, reduzir o ruído e melhorar a eficiência do nosso modelo. Ao final desta etapa, os 25 atributos mais relevantes foram selecionados.

Na aplicação, os atributos que continham informações da transação de compra são enviados pelo cliente na hora da solicitação, tais como: *transactionID*, *TransactionAmt*, *card1*, *card2*, *card5*, *card6*, *addr1* e *P_emaildomain*, os outros atributos são resgatados no banco de dados a partir do identificador (*TransactionID*) de cada usuário, tais como: *C1*, *C2*, *C4*, *C8*, *C13*, *C14*, *D1*, *D2*, *D3*, *D4*, *D10*, *D15*, *M4*, *V317*, *hour*, *day* e *dow*.

O atributo *TransactionAmt* representa o valor do pagamento em dólares. Os atributos *card1*, *card2*, *card5* e *card6* contêm informações relacionadas ao cartão de pagamento, como o tipo de cartão, a categoria, o banco emissor e o país associados. O atributo *addr1* corresponde ao endereço, enquanto *P_emaildomain* indica o domínio do e-mail. Por sua vez, *C1*, *C2*, *C4*, *C8*, *C13* e *C14* representam valores de contagem, embora seus significados reais tenham sido mascarados no conjunto de dados. *D1*, *D2*, *D3*, *D4*, *D10* e *D15* refletem a diferença em dias entre transações anteriores, enquanto *M4* se refere a correspondências, como o nome no cartão . O atributo *V317* representa um recurso avançado desenvolvido pela Vesta, abrangendo classificação, contagem e outras relações entre entidades, todas anonimizadas. Os demais atributos (*hour*, *day* e *dow*) foram derivados de uma coluna que armazenava o tempo em *timedelta* e representam, respectivamente, a hora, o dia e o dia da semana da transação.

3.4. Modelagem do Pipeline

Para obter modelos preditivos com uma adequada capacidade de generalização, realizamos a separação dos dados rotulados em um conjunto de treino, usado para otimizar os parâmetros dos modelos, e um conjunto de teste, utilizado para efetivamente avaliar o desempenho dos modelos. Comumente, a separação dos conjuntos de treino e teste é feita de forma randomizada. Contudo, como o conjunto de dados é desbalanceado, a proporção entre as classes foi preservada usando a função *StratifiedShuffleSplit* da biblioteca *Scikit-learn*. Além disso, 75% das instâncias foram separadas para formar o conjunto de treinamento e 25% para o conjunto de teste.

A fim de gerar os modelos preditivos, exploramos cinco algoritmos clássicos de classificação: *Logistic Regression*, *Decision Tree (DT)*, *K-Nearest Neighbors (KNN)*, *Random Forest (RF)* e XGBoost. Cada um desses algoritmos foi avaliado em três cenários diferentes: i) com os dados originais desbalanceados, ii) dados balanceados por meio de uma subamostragem aleatória e iii) dados balanceados por meio de uma sobreamostragem aleatória.

[2]https://scikit-learn.org/stable/

Além disso, para avaliar um determinado algoritmo de classificação em um cenário específico, é necessário realizar a otimização dos hiper-parâmetros deste algoritmo. Uma abordagem efetiva para encontrar bons hiper-parâmetros dentre uma vasta gama de possibilidades é a busca em grade. Realizamos a busca em grade para selecionar os hiper-parâmetros de cada um dos algoritmos utilizados, em cada cenário específico, subdividindo o conjunto de treino em um subconjunto de treino e de validação na proporção 80%-20% (ou seja, utilizamos uma validação cruzada com 5 *folds*) e realizando 10 experimentos de avaliação. Em cada experimento, um modelo é gerado utilizando os dados de treino e avaliado nos dados de validação. Após a busca em grade, o melhor conjunto de hiper-parâmetros, em termos de acurácia, é utilizado para treinar um modelo com os dados de treino e validação. Este modelo é então utilizado para fazer predições sobre os dados de teste.

3.5. Escolha do Melhor Modelo

Após o processo dos experimentos descritos anteriormente, torna-se necessário selecionar o modelo com o melhor desempenho. Para isso, é necessário utilizar uma ou mais métricas de desempenho, as quais são calculadas a partir das predições realizadas pelos modelos no conjunto de teste. Neste trabalho, exploramos as seguintes métricas: acurácia, precisão, revocação e *F1-score*.

Embora todas as métricas sejam relevantes e deseje-se obter o maior valor possível em todas, em nossa análise consideramos que um falso negativo é um erro mais crítico do que um falso positivo. Isso se deve ao fato de que uma possível aplicação de um modelo de detecção automática de fraudes seria alertar usuários humanos sobre o risco de uma determinada transação, de modo que este usuário possa eventualmente realizar uma checagem e confirmar ou não a predição do modelo. Neste contexto, um falso negativo seria entendido como uma transação fraudulenta ser considerada legítima. Por outro lado, o dano de bloquear uma transação incorretamente classificada como fraudulenta é menor do que permitir a execução de uma transação fraudulenta. Dessa forma, damos atenção especial às métricas de revocação e ao F1-*score* a fim de selecionar o melhor classificador. Todos os códigos utilizados nos experimentos estão disponíveis no *Github* [3].

3.6. Discussão dos Resultados

Na Tabela 2, é possível observar os resultados apresentados pelo melhor modelo encontrado pela abordagem proposta para cada um dos cinco algoritmos de classificação investigados (*Logistic Regression*, *Decision Tree (DT)*, *K-Nearest Neighbors (KNN)*, *Random Forest (RF)* e XGBoost, bem como os resultados reportados pelos principais trabalhos relacionados: [Deng et al. 2021], [Kewei et al. 2021], [Nguyen et al. 2022], [Chen e Han 2021], [Ni et al. 2023] e [Bakhtiari, Nasiri e Vahidi 2023]. Vale destacar que todos esses trabalhos utilizaram o conjunto de dados *IEE-CIS Fraud Detection*.

É importante lembrar ainda que, cada um dos cinco algoritmos de classificação investigados foi avaliado em três cenários diferentes: i) com os dados originais (desbalanceados), ii) subamostragem aleatória e iii) sobre-amostragem aleatória. Contudo, o melhor modelo encontrado para cada um dos cinco algoritmos avaliados foi obtido sempre no cenário baseado em dados balanceados por meio de uma sobre-amostragem aleatória.

[3] https://github.com/robsonsants/Credit_Card_Fraud-Detection_Pipeline

Analisando a Tabela 2, podemos observar que melhor modelo produzido pela abordagem proposta foi obtido utilizando-se o algoritmo RF com sobre-amostragem aleatória, o qual apresentou acurácia de 99,98%, precisão de 99,97%, revocação de 100% e *F1-score* de 99,88%, superando o estado da arte. Nota-se também que a grande maioria dos trabalhos relacionados utilizou a apenas a acurácia como métrica de desempenho. Contudo, sabe-se que a acurácia não é uma métrica adequada para dados desbalanceados.

Tabela 2. Resultados dos modelos de classificação binária (média em %) em termos de acurácia, precisão, revocação e F1-score.

Autor	Método	Acurácia	Precisão	Revocação	F1-score
Deng *et al.*	RF	96,80	–	–	–
	LR	92,10	–	–	–
Kewei *et al.*	NN	95,80	–	–	–
	SVM	94,00	–	–	–
Nguyen *et al.*	DNN	97,60	–	–	–
	CatBoost	98,64	–	–	–
Chen e Han	CatBoost	98,30	–	–	–
	SVM	95,00	–	–	–
Ni *et al.*	SOBT	97,87	88,19	44,42	59,08
Bakhtiari *et al.*	LightGBM + LiteMORT	99,44	92,79	91,67	90,65
Esta proposta	LR	69,58	69,24	70,46	69,84
	DT	99,52	98,13	100,00	99,06
	KNN	98,93	97,92	100,00	98.95
	RF	**99,98**	**99,97**	**100,00**	**99,88**
	XGBOOST	99,84	99,69	100,00	99,84

4. Arquitetura de Tempo Real para Detecção de Fraudes

A Figura 2 apresenta uma visão geral da arquitetura de software proposta para lidar com a detecção de fraudes em tempo real. Os componentes da solução e os fluxos de operação do sistema são ilustrados, com números identificando a sequência de etapas até a validação ou não da transação - simulando as operações de um banco. A arquitetura é composta por quatro serviços distintos que se comunicam entre si, nomeados como `Orquestração` (serviço 1), `Enriquecedor de Dados (Vesta)` (serviço 2) `Serviço de Predição` (serviço 3), `Serviço de Transação` (serviço 4) e o banco de dados `MongoDB`.

O componente de `Orquestração` foi criado baseado no padrão *Orchestration*, uma abordagem utilizada no desenvolvimento de microsserviços, em que tal componente é responsável por coordenar e controlar a interação entre diferentes microsserviços para executar uma funcionalidade ou um fluxo de trabalho mais complexo. Como um maestro em uma orquestra, o componente gerencia o fluxo de informações e a coordenação entre os componentes, aumentando a escalabilidade, confiabilidade e tolerância a falhas do sistema [Megargel, Poskitt e Shankararaman 2021].

O componente de `Enriquecedor de Dados` é responsável pelo enriquecimento dos dados da transação. Em um sistema bancário real, várias informações seriam coletas, por exemplo, sobre quem está fazendo o pagamento e sobre quem está recebendo,

a fim de ter dados suficientes para decidir se a transação é uma fraude ou não. Na nossa implementação, os dados complementares, coletados pela empresa Vesta, que disponibilizou o conjunto dados, foram armazenados num banco de dados MongoDb e servidos através do serviço 2. Uma vez enriquecidos os dados, o serviço 1 envia a transação para a predição.

O `Serviço de Predição` é responsável pela classificação das transações enviadas pelo cliente. Nele, é carregado o modelo de aprendizado de máquina RF, que obteve os melhores resultados na avaliação de desempenho. O serviço 3 então recebe as transações de cartão de crédito e executa uma classificação binária para a identificação da ocorrência ou não de fraude.

O componente de `Serviço de Transação` é responsável pela autorização da transação financeira. Quando recebe a resposta do serviço de predição, retorna a resposta da transação do cliente para a orquestração, que enfim indica ao cliente se a transação foi executado com sucesso ou bloqueada por suspeita de fraude.

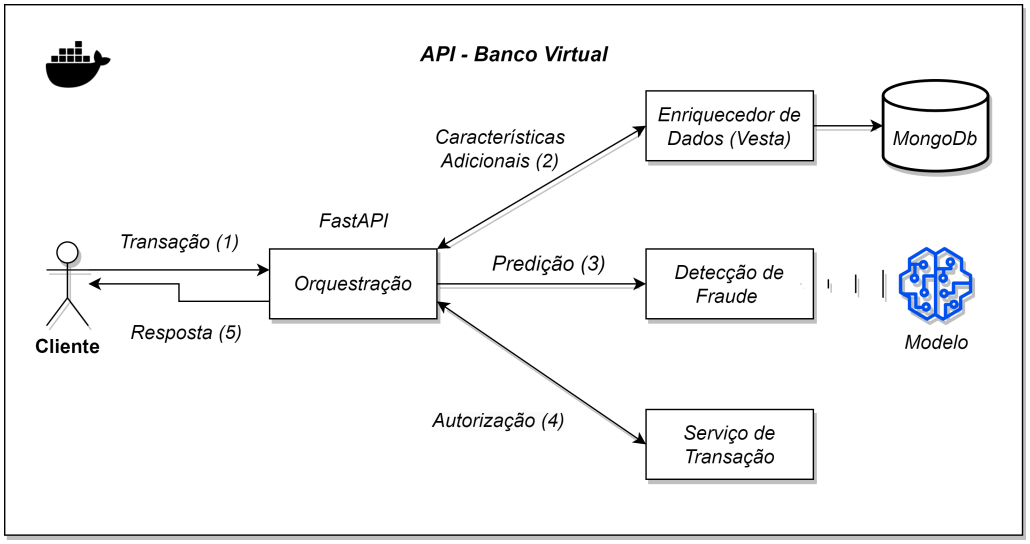

Figura 2. Arquitetura de Tempo Real para Detecção de Fraudes.

A primeira etapa do fluxo de execução é a requisição de compra, ou seja, o início da transação, que representa a etapa de coleta inicial de dados. O cliente interage inicialmente com a Orquestração, enviando as informações da transação de compra com o cartão de crédito, tais como: *TransactionID*, *TransactionAmt*, *card1*, *card2*, *card5*, *card6*, *addr1* e *P_emaildomain*. O serviço então chama o serviço 2 para enriquecer os dados da transação com o restante das informações do cliente, tais como: *C1*, *C2*, *C4*, *C8*, *C13*, *C14*, *D1*, *D2*, *D3*, *D4*, *D10*, *D15*, *M4*, *V317*, *hour*, *day* e *dow*. Nossa implementação simula um serviço da Vesta, onde mais características são incluídas na requisição para que a mesma fique equivalente ao conjunto de dados utilizado em treinamento, os 25 atributos. É importante ressaltar que no conjunto de dados, alguns dos dados estão anonimizados.

A segunda etapa envolve a análise de dados. O serviço 2 responde à solicitação do serviço 1, retornando as informações restantes necessárias para completar o conjunto de dados da transação. O serviço 1 então envia os dados completos para a Predição que é o serviço 3, sendo responsável pela detecção de fraude.

Na terceira etapa da detecção de fraudes, o serviço 3 emprega um modelo de aprendizado de máquina do tipo RF, previamente treinado e testado para a tarefa. A escolha do RF é justificada pelos resultados de um processo extensivo de avaliação e otimização de cinco algoritmos clássicos de classificação: LR, DT, KNN, RF e *XGBoost*. Estes algoritmos foram testados em três cenários de balanceamento de dados, e o RF se destacou com um desempenho superior nos dados balanceados por sobre-amostragem aleatória. Nessa análise, a relevância das métricas de desempenho Revocação e *F1-Score* foi enfatizada, pois, a priorização dessas métricas se dá pelo impacto mais crítico dos falsos negativos, que permitem transações fraudulentas serem consideradas legítimas, comparativamente a falsos positivos, que podem ser verificados e corrigidos, minimizando danos diretos.

A quarta etapa é a resposta ao cliente. Dependendo da resposta recebida do serviço 3, o serviço 1 solicita ao serviço 4 a autorização da transação, se a transação não for fraudulenta ou caso a transação seja fraudulenta. A decisão final se é autorizada ou não é então enviada de volta ao cliente pelo serviço 1.

Nesta arquitetura, temos uma sequência estruturada e bem definida de operações automatizadas que estabelecem uma interação entre os quatro serviços envolvidos. A eficiência e robustez desta metodologia, que tem como objetivo principal combater as fraudes de cartão de crédito, são garantidas pela sua ancoragem na arquitetura dos serviços interconectados e no modelo aprendizagem de máquina.

4.1. Implementação e Implantação

O projeto utilizou diversas bibliotecas para sua construção e execução, incluindo *FastAPI*, *Gunicorn*, *Uvicorn*, *Pandas*, *Scikit-learn*, *SciPy*, *Joblib*, *NumPy*, *Pydantic*, *PyMongo* e *Requests*. Essas bibliotecas desempenharam funções essenciais, desde a construção de *APIs* até a manipulação de dados e conexão com *mongoDB*. Utilizamos *Docker* e *Docker Compose* para garantir um ambiente de execução consistente e repetível. Essas ferramentas permitem que o software seja executado de forma confiável em diferentes ambientes, simplificando o desenvolvimento, teste e implantação.

O sistema foi implementado em instâncias de máquinas virtuais do tipo *t2.medium*, com 2 vCPUs e 4 GB de RAM, na região de São Paulo da nuvem da *Amazon Web Service (AWS)*. PAra cada serviço foi alocada uma máquina virtual, além de uma máquina virtual para o *Locust*. Cada serviço foi executado em um único container Docker.

Foi dada ênfase especial aos dados mantidos na parte da *Vesta*, para replicar a funcionalidade de uma aplicação bancária autêntica. A finalidade disso foi garantir a melhor simulação possível de uma aplicação bancária real na execução dos testes.

Todos esses elementos trabalham juntos para apoiar a aplicação, permitindo que todo o sistema seja iniciado, parado e gerenciado de maneira conveniente, proporcionando uma alta eficiência e portabilidade.

4.2. Experimentos

Para avaliar o desempenho e capacidade máxima de atendimento do sistema, realizamos experimentos utilizando a ferramenta *Locust*[4], uma estrutura de teste de carga de código

[4]https://locust.io/

aberto que permite simular ações simultâneas de múltiplos usuários em um serviço web. Durante os experimentos, o *Locust* foi empregado para capturar métricas críticas, tais como o tempo de resposta de cada requisição e o número total de requisições atentidas. Além disso, um registro detalhado foi mantido, incluindo a duração exata que cada serviço levou para processar e concluir uma transação e o consumo de CPU e memória das máquinas virtuais utilizadas. Com isso, foi possível obter uma visão abrangente do desempenho do sistema sob carga, identificando potenciais gargalos e identificando oportunidades de melhorias para trabalhos futuros.

A aplicação foi submetida a experimentos que variaram o número de clientes iniciais e a taxa de geração (i.e., quantidade de clientes iniciados por segundo). Foram executados 6 experimentos com configurações diferentes: utilizando 10 clientes e taxa de geração de 5 clientes/s; 20 clientes com taxa de 10 clientes/s, 30 clientes e taxa de 15, 40 clientes e taxa de 20, 50 clientes e taxa de 25 e, por fim, 60 clientes e 30 iniciados por segundo. O tempo de duração do experimento foi mantido em 3 minutos nas 6 configurações.

Para questões de análise, foram removidas 10% das requisições iniciais em todos os experimentos por ser uma fase de *warm up* do sistema. A Seção 5 apresenta os resultados obtidos durante o teste de carga do sistema.

4.3. Resultados

Esta seção apresenta os principais resultados obtidos a partir da avaliação de desempenho da arquitetura de tempo real para detecção de fraude de cartão de crédito. A carga de trabalho utilizada é uma transação de compra enviada por uma quantidade variada de clientes. As métricas analisadas foram tempo de execução da transação, tempo de resposta do serviço 3 (i.e., tempo de predição) e total de requisições atendidas.

A Figura 3 apresenta o tempo médio para executar a transação completa, englobando o período de espera na fila de entrada da orquestração, o processo de enriquecimento dos dados, a fase de predição e, por fim, a autorização da transação. Além disso, é calculado o intervalo de confiança de 95%. A figura mostra que os tempos de execução para as diferentes cargas de trabalho ficam abaixo de 2 segundos, com o menor tempo ficando abaixo de 300ms. Uma análise dos tempos de execução de cada serviço mostrou que uma sobrecarga começa a acontecer quando a taxa de requisições fica acima de 20 por segundo. Nesse ponto, o sistema começa a enfileirar as requisições no serviço de orquestração, resultando em um tempo médio de execução acima de 1 segundo.

A Figura 4 apresenta o tempo de execução apenas do serviço de predição. Pode-se perceber que o modelo executa rapidamente em todos os cenários (sempre em menos de 200ms), mas existe um aumento natural com o crescimento da taxa de clientes iniciados por segundo. O controle de fluxo exercido pelo serviço de orquestração impede que o serviço de predição seja sobrecarregado.

Podemos observar uma interseção entre os intervalos de confiança para os tempos de predição das transações com 20, 25 e 30 clientes iniciados por segundo. Por isso, realizamos um teste de Análise de Variância (ANOVA) *one-way* e o teste de intervalo de Tukey para avaliar as cargas de trabalho par-a-par. O resultado dos testes comprovou que não há diferença estatisticamente significativa entre a média dos tempos de predição para as referidas cargas, obtendo um *p-value* ajustado $> 0,05$ para cada par. Tal resultado cor-

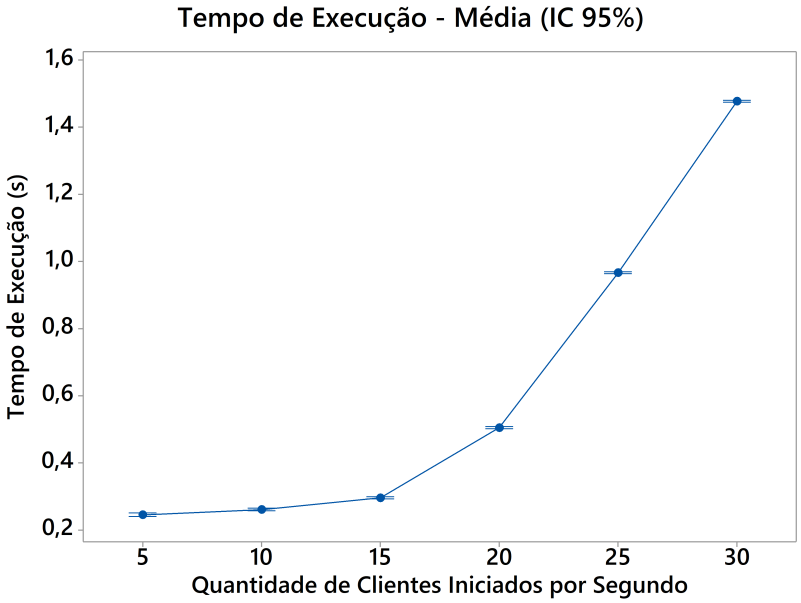

Figura 3. Média do Tempo de Execução das Transações

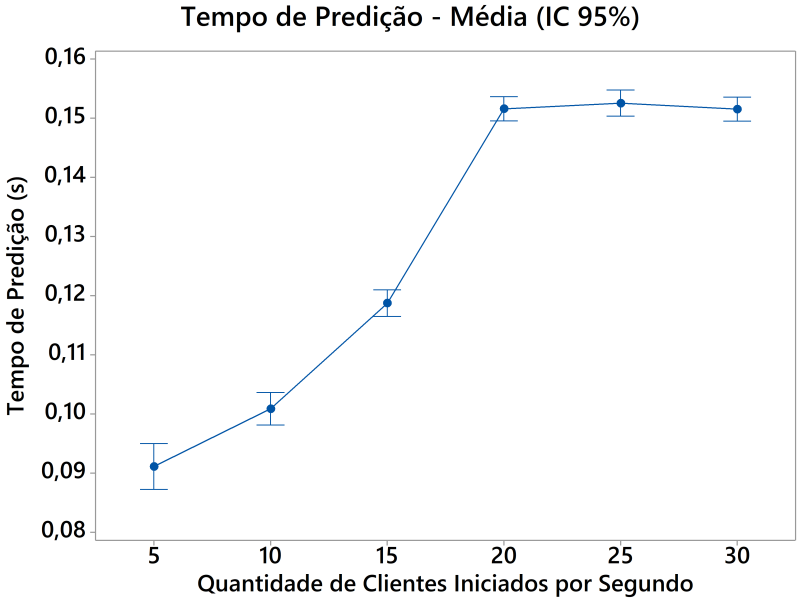

Figura 4. Média do Tempo de Predição

robora a análise dos intervalos de confiança da Figura 4 e indica que o enfileiramento das requisições está mantendo o desempenho do módulo de predição igual nos experimentos com 20 ou mais clientes por segundo.

Uma análise do consumo de CPU e memória das máquinas virtuais revelou que o componente de `Orquestração` é o que mais consome recursos, tornando-se um ponto crítico para o sistema, a ponto de causar o enfileiramento de várias requisições. Enquanto os demais serviços não ultrapassam 20% de uso de CPU, o serviço de `Orquestração` utiliza o máximo quando 20 ou mais clientes são iniciados por segundo.

Uma vez que cada experimento teve uma duração de 3 minutos e houve o enfileiramento de requisições no serviço 1, nem todas as requisições geradas puderam ser atendidas. Na Tabela 3, é possível observar o total de requisições atendidas em cada experimento. Podemos notar que, apesar de aumentarmos a quantidade de clientes gerando requisições por segundo, o sistema consegue atender, no máximo, a 3674 requisições dentro de um período de 3 minutos. Isso sugere que o sistema atinge sua capacidade máxima ao lidar com uma carga de trabalho de 20 clientes por segundo. Ao utilizar 25 e 30 clientes por segundo, a quantidade de requisições atendidas é praticamente igual à carga de 20 clientes, pois mais requisições foram enfileiradas e não puderam ser executadas dentro do intervalo definido para o experimento.

Experimentos	Total de Requisições atendidas
5 clientes iniciados por segundo	1039
10 clientes iniciados por segundo	2057
15 clientes iniciados por segundo	3015
20 clientes iniciados por segundo	3624
25 clientes iniciados por segundo	3631
30 clientes iniciados por segundo	3674

Tabela 3. Total de requisições atendidas em cada experimento

5. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada e avaliada uma abordagem para detecção de fraudes em transações de cartão de crédito, a qual explora algoritmos de aprendizado de máquina, o pré-processamento dos dados e a engenharia de atributos. A abordagem proposta apresentou resultados superiores ao estado da arte, com acurácia de 99,98%, precisão de 99,97%, revocação de 100% e *F1-score* de 99,88%. Além disso, utilizamos o melhor modelo de aprendizagem de máquina *random forest* na arquitetura em tempo real da aplicação do banco virtual para detecção de fraudes em transações financeiras de cartão de crédito. A aplicação atingiu um tempo médio de resposta de requisição excelente.

Como trabalhos futuros, pretende-se fazer experimentos com máquinas mais potentes e com mais usuários recorrentes, simulando o funcionamento de um banco mundial utilizando o serviço de predição de fraudes. Outro ponto a ser explorado é a replicação dos serviços e uma avaliação de diferentes estratégias de replicação e elasticidade dos containers do sistema. Ademais, utilizar outros conjuntos de dados e outros tipos de transações financeiras visando simular o ambiente real de uma instituição financeira.

6. Agradecimentos

Parte dos resultados apresentados neste trabalho foram obtidos através do projeto "Residência em Segurança da Informação", executado pela UFC, em parceria com o SiDi e financiado pela Samsung Eletrônica da Amazônia Ltda., no âmbito da Lei de Informática no. 8.248/91.

Referências

ABAKARIM, Y.; LAHBY, M.; ATTIOUI, A. An efficient real time model for credit card fraud detection based on deep learning. In: *Proceedings of the 12th international conference on intelligent systems: theories and applications*. [S.l.: s.n.], 2018. p. 1–7.

BAKHTIARI, S.; NASIRI, Z.; VAHIDI, J. Credit card fraud detection using ensemble data mining methods. *Multimedia Tools and Applications*, Springer, p. 1–19, 2023.

CARCILLO, F. et al. Scarff: a scalable framework for streaming credit card fraud detection with spark. *Information fusion*, Elsevier, v. 41, p. 182–194, 2018.

CHEN, Y.; HAN, X. Catboost for fraud detection in financial transactions. In: IEEE. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. [S.l.], 2021. p. 176–179.

CHERIF, A. et al. Credit card fraud detection in the era of disruptive technologies: A systematic review. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, 2022.

DENG, W. et al. A data mining based system for transaction fraud detection. In: IEEE. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. [S.l.], 2021. p. 542–545.

KEWEI, X. et al. A hybrid deep learning model for online fraud detection. In: IEEE. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. [S.l.], 2021. p. 431–434.

MEGARGEL, A.; POSKITT, C. M.; SHANKARARAMAN, V. Microservices orchestration vs. choreography: A decision framework. In: IEEE. *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC)*. [S.l.], 2021. p. 134–141.

MENSHCHIKOV, A. et al. Comparative analysis of machine learning methods application for financial fraud detection. In: IEEE. *2022 32nd Conference of Open Innovations Association (FRUCT)*. [S.l.], 2022. p. 178–186.

NGUYEN, N. et al. A proposed model for card fraud detection based on catboost and deep neural network. *IEEE Access*, IEEE, v. 10, p. 96852–96861, 2022.

NI, L. et al. Fraud feature boosting mechanism and spiral oversampling balancing technique for credit card fraud detection. *IEEE Transactions on Computational Social Systems*, IEEE, 2023.

PRUSTI, D.; DAS, D.; RATH, S. K. Credit card fraud detection technique by applying graph database model. *Arabian Journal for Science and Engineering*, Springer, v. 46, n. 9, p. 1–20, 2021.

SAPOZHNIKOVA, M. et al. Anti-fraud system on the basis of data mining technologies. In: IEEE. *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. [S.l.], 2017. p. 243–248.

THENNAKOON, A. et al. Real-time credit card fraud detection using machine learning. In: IEEE. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. [S.l.], 2019. p. 488–493.