UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Identificação de fraude em transações financeiras com Redes Neurais para Grafos

Sabrina Vieira Guerra

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP
SÃO CARLOS

Sabrina Vieira Guerra

Identificação de fraude em transações financeiras com Redes Neurais para Grafos

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientadora: Profa. Dra. Tatiane Nogueira Rios

Versão original

São Carlos

2025

Guerra, Sabrina Vieira
S856m Identificação de fraude em transações financeiras com Redes Neurais para Grafos / Sabrina Vieira Guerra ; Orientadora Tatiane Nogueira Rios. – São Carlos, 2025.
67 p. : il. (algumas color.) ; 30 cm.

Monografia (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2025.

1. Redes Neurais para Grafos. 2. GNNs. 3. Fraudes financeiras. 4. Classificação binária. 5. Detecção de fraudes. I. Rios, Tatiane Nogueira, orient.

Sabrina Vieira Guerra

Fraud identification in financial transactions with Graph Neural Networks

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Original version

São Carlos

2025

Este trabalho é dedicado ao meu pai
que partiu precocemente e não pôde ver tudo isso acontecer de perto
mas tenho certeza que iria adorar.

AGRADECIMENTOS

Agradeço ao meu parceiro de vida, Tiago, por estar ao meu lado durante todas as madrugadas de estudo (e por me incentivar a me afastar quando necessário).

À minha família, pelo apoio incondicional e AMOR que me trouxeram até onde estou hoje.

Ao meu trabalho, que oferece tanto espaço para o crescimento e me impulsiona a aprender coisas novas - e totalmente fora da minha zona de conforto.

À minha orientadora, por inspirar minha fascinação pelos grafos.

E agradeço também às ferramentas de inteligência artificial utilizadas como auxiliadoras durante o entendimento de conceitos abstratos e busca por melhores metodologias, como o Gemini e o Data Science Agent aplicado ao Google Colab e BigQuery. Destaco que o trabalho é de autoria própria, sendo que a utilização de tais ferramentas restringe-se a materiais de estudo e refinamento de código.

"Estranho sobre aprender; quanto mais longe eu vou, mais vejo o que nunca soube que sequer existia. Algum tempo atrás, tolamente imaginei que poderia aprender tudo – todo o conhecimento existente. Agora espero apenas ser capaz de saber de sua existência e entender um mínimo disso."

Charlie Gordon, Flores para Algernon

RESUMO

GUERRA, S. **Identificação de fraude em transações financeiras com Redes Neurais para Grafos**. 2025. 67 p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2025.

As fraudes financeiras são um risco atual para comerciantes e clientes, que enfrentam prejuízos financeiros e experiências desagradáveis, principalmente em transações *on-line*. Os padrões entre fraudadores e seu métodos de operação podem oferecer alguma previsibilidade de como uma transação ilegítima se comporta. Este trabalho tem o objetivo de transformar um conjunto de dados tabular e altamente desbalanceado em um grafo com nós (sendo eles as transações e demais informações de identidade) e arestas, que representam o relacionamento entre as entidades. A partir dessa modelagem, aplicaram-se dois modelos de Redes Neurais em Grafos (Graph Neural Networks – GNN), o GraphSAGE e o Graph Attention Network (GAT), para avaliar os resultados segundo as métricas AUC (Area Under the Curve)–ROC (Receiver Operating Characteristic), AUC–PR (Precision–Recall), revocação, precisão e F1-Score. Além disso, the XGBoost classificato modelo was used para uma segunda comparação das métricas atingidas. Os experimentos com GNNs utilizaram a biblioteca Pytorch Geometric e sua classe HeteroConv para aplicar os dois diferentes modelos de redes neurais, SAGEConv e GATConv. As métricas apresentadas foram as de melhor resultado obtido com os respectivos hiperparâmetros e modelos. Os resultados demonstram que o modelo SAGEConv performou estatisticamente melhor do que o modelo GATConv em relação à métrica AUC-PR, não havendo diferença estatisticamente significante em relação ao modelo *baseline* XGBoost. No entanto, o valor médio de todos os seus parâmetros foi maior quando comparado aos dois modelos. Conclui-se que os modelos GNNs são um bom ponto de partida para a análise deste conjunto de dados, mesmo que enfrentem desafios pelo alto desbalanceamento e por serem o produto de uma conversão de dados tabulares. A eficiência temporal e complexidade dos algoritmos devem ser levadas em consideração, bem como a boa explicabilidade visual do grafo.

Palavras-chave: Redes Neurais para Grafos. GNNs. Fraudes financeiras. Classificação binária. Detecção de fraudes.

ABSTRACT

GUERRA, S. **Fraud identification in financial transactions with Graph Neural Networks**. 2025. 67 p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2025.

Financial fraud is a current risk for merchants and customers, who face financial losses and unpleasant experiences, especially in online transactions. The patterns among fraudsters and their methods of operation can offer some predictability of how an illegitimate transaction behaves. This work aims to transform a tabular and highly imbalanced dataset into a graph with nodes (being them the transactions and other identity information) and edges, which represent the relationship between the entities. From this modeling, two Graph Neural Network (GNN) models, GraphSAGE and Graph Attention Network (GAT), were applied to evaluate the results according to the AUC (Area Under the Curve)–ROC (Receiver Operating Characteristic), AUC–PR (Precision–Recall), recall, precision, and F1-Score metrics. Furthermore, the XGBoost classifier model was used for a second comparison of the achieved metrics. The experiments with GNNs used the Pytorch Geometric library and its HeteroConv class to apply the two different neural network models, SAGEConv and GATConv. The presented metrics were those of the best result obtained with the respective hyperparameters and models. The results demonstrate that the SAGEConv model performed statistically better than the GATConv model in relation to the AUC-PR metric, with no statistically significant difference in relation to the baseline XGBoost model. However, the average value of all its parameters was higher when compared to the two models. It is concluded that GNN models are a good starting point for the analysis of this dataset, even though they face challenges due to the high imbalance and being the product of a conversion from tabular data. The temporal efficiency and complexity of the algorithms should be taken into consideration, as well as the good visual explainability of the graph.

Keywords: Graph Neural Networks. GNNs. Financial frauds. Binary classification. Fraud detection.

LISTA DE FIGURAS

Figura 1 – Árvore de Decisão para classificação de possível fraude. Fonte: Narde (2024) 30
Figura 2 – Representação de um grafo, com 6 nós e 7 arestas. Fonte: Mariano (2020) 30
Figura 3 – Demonstração de um grafo sobre retweets em relação à política nos EUA. Fonte: Menczer, Fortunato e Davis (2020) 31
Figura 4 – Grafos com propriedades distintas de direção e peso. Fonte: Menczer, Fortunato e Davis (2020) 33
Figura 5 – Exemplo simples de classificação de pássaros com uma camada oculta e uma saída. Fonte: da autora 34
Figura 6 – Redes neurais para grafos com aspecto convolucional em tarefas de classificação para fraudes. Fonte: Adaptação de Motie e Raahemi (2024) 34
Figura 7 – Exemplo de grafo cujos nós são conectados pela semelhança de atributos. Fonte: da autora 35
Figura 8 – Comportamento da função de ativação ReLU em modelo com 3 camadas ocultas. Fonte: Google for Developers (2025c) 37
Figura 9 – Amostragem, agregação de features entre vizinhos e classificação de nós com GraphSAGE. Fonte: Xiao *et al.* (2022) 38
Figura 10 – Representação da propagação a partir de coeficientes de atenção normalizados aplicados a 3 *heads* (K = 3). Fonte: Xiao *et al.* (2022) 40
Figura 11 – Demonstração de métricas e curva ROC com threshold de 0.5 em *dataset* desbalanceado. Fonte: Google for Developers (2025b) 42
Figura 12 – Curva PR, com AUC-PR = 0,623. Útil para *dataset* desbalanceados. Fonte: Google for Developers (2025b) 42
Figura 13 – Modelagem do grafo a partir de dados tabulares e processamento da GNN. Fonte: da autora, adaptado de Wang *et al.* (2021) 48
Figura 14 – Alto desbalanceamento entre a quantidade de transações legítimas e fraudulentas. Fonte: da autora. 51
Figura 15 – Porcentagem de fraudes em relação aos tipos únicos da coluna ProductCD. Fonte: da autora. 52
Figura 16 – Porcentagem de fraudes em relação aos domínios de *e-mail* do comprador. Fonte: da autora. 52
Figura 17 – Porcentagem de fraudes em relação aos domínios de *e-mail* do recebedor. Fonte: da autora. 53
Figura 18 – Porcentagem de fraudes em relação à bandeira do cartão. Fonte: da autora. 53
Figura 19 – Porcentagem de fraudes em relação ao tipo do cartão. Fonte: da autora. 54

Figura 20 – Exemplo de amostras de transações legítimas e fraudulentas, bem como suas conexões com nós de identidade. Fonte: da autora. 56
Figura 21 – Comparação de diferenças estatisticamente significantes entre os valores de AUC-PR dos modelos SAGEConv, GATConv e XGBoost. Fonte: da autora. 59

LISTA DE TABELAS

Tabela 1 – Matriz de confusão para classificação de transações fraudulentas. Fonte: Adaptado de Google for Developers (2025d) 40
Tabela 2 – Tipo e descrição das *features* do *dataset train_transaction* 46
Tabela 3 – Tipo e descrição das *features* do dataset *train_identity* 46
Tabela 4 – Hiperparâmetros testados para melhor valor em GNNs 49
Tabela 5 – Hiperparâmetros testados para melhor valor com XGBoost 50
Tabela 6 – Exploração de transações legítimas e fraudulentas 51
Tabela 7 – Número de nós por entidade 55
Tabela 8 – Distribuição das Máscaras de Nós 56
Tabela 9 – Hiperparâmetros e métricas resultantes do treinamento com SAGEConv 57
Tabela 10 – Hiperparâmetros e métricas resultantes do treinamento com GATConv 57
Tabela 11 – Melhores resultados para cada modelo avaliado 58

LISTA DE ABREVIATURAS E SIGLAS

ML	*Machine Learning*
IA	Inteligência Artificial
IEEE-CIS	*Institute of Electrical and Electronics Engineers - Computational Intelligence Society*
XGBoost	*eXtreme Gradient Boosting*
CNNs	*Convolutional Neural Networks*
GNNs	*Graph Neural Networks*
GCN	*Graph Convolutional Networks*
ReLU	*Rectified Linear Unit*
GraphSAGE	*Graph SAmple and aggreGatE*
GAT	*Graph Attention Networks*
TP	*True Positive*
TN	*True Negative*
FP	*False Positive*
FN	*False Negative*
AUC-ROC	*Area under the curve - Receiver-operating characteristic curve*
AUC-PR	*Area under the curve - Precision-Recall*
CD	*Critical Distance*

SUMÁRIO

1 INTRODUÇÃO 25
1.1 Objetivos 26

2 FUNDAMENTAÇÃO TEÓRICA 27
2.1 Detecção de fraudes financeiras 27
2.1.1 Trabalhos relacionados 28
2.2 Modelos de classificação 29
2.2.1 XGBoost 29
2.3 Teoria dos Grafos e Redes Complexas 30
2.3.1 Propriedades de um grafo 32
2.4 Redes Neurais para Grafos 32
2.4.1 Graph Convolutional Networks 36
2.4.1.1 Funções de ativação e perda, *backpropagation* e otimizador 36
2.4.1.2 GraphSAGE 38
2.4.2 Graph Attention Networks 38
2.5 Métricas de avaliação 39
2.6 Comparação estatística 41

3 METODOLOGIA 45
3.1 Dataset IEEE-CIS Fraud Detection 45
3.1.1 Análise e preparação dos dados 45
3.2 Grafo modelado e processamento das GNNs 48
3.2.1 Classe HeteroGNN 48
3.2.2 Treinamento, validação e teste 49
3.3 *Dataset* aplicado ao XGBoost 50
3.4 Comparação estatística 50

4 AVALIAÇÃO EXPERIMENTAL 51
4.1 Análise do dataset 51
4.2 Estrutura do grafo 53
4.3 Segmentação dos experimentos 54
4.4 Resultados 57

5 CONCLUSÕES 61

REFERÊNCIAS 63

APÊNDICE A – REPOSITÓRIO DE CÓDIGO NO GITHUB 67

1 INTRODUÇÃO

Com o avanço da tecnologia em indústrias como as de serviços financeiros e varejista, a crescente virtualização de operações cotidianas é um contexto imparável. Projeções realizadas pela ABComm (Associação Brasileira de Comércio Eletrônico) indicam um crescimento expressivo do e-commerce no Brasil, esperando-se cerca de R$345 bilhões em faturamento para 2029 contra os R$204 bilhões de 2024 (ABComm, 2024). Sendo o ritmo de crescimento maior do que o esperado para o varejo como um todo, há fortes evidências de que as compras online tornam-se um padrão de consumo cada vez mais adotado pelos clientes. (Vicente, 2024). Diante de um panorama promissor, há também os desafios associados ao que parece ser o principal receio dos consumidores e lojistas: as fraudes. A possibilidade de pagar pelas compras online com o cartão de crédito - mediante cadastro de dados e código de segurança - traz também o risco de expor tais informações a sites maliciosos.

Em 2023, apenas no Brasil, aconteceram cerca de 3,7 milhões de tentativas de fraude, cujo valor acumulado atinge R$3,5 bilhões. Conclui-se, ainda, que o ticket médio (o número do faturamento total dividido pelo número de vendas do período) de tais tentativas foi duas vezes maior do que o ticket médio de compras legítimas (Chalegra, 2024). Em um contexto tão agressivo, não é surpresa que o consumidor esteja cada vez mais receoso. Mesmo com o crescimento do padrão de consumo on-line, em uma pesquisa realizada pela PSafe, 68% dos consumidores têm medo de realizar compras virtuais. Além disso, 50% utilizam o cartão de crédito como meio de pagamento pela praticidade, mas 54% dos respondentes têm medo de ter seu cartão clonado (Biancamano, 2021). Além do transtorno causado para o consumidor, o próprio lojista pode se deparar com cenários em que precise arcar com ressarcimentos, perda da mercadoria e até mesmo abalo da confiança do cliente, uma vez que sua experiência e visão da marca são prejudicadas por todo o contratempo causado.

Portanto, é de grande interesse dos comerciantes e empresas intermediárias de pagamentos desenvolver métodos robustos para a detecção de tentativas de fraude, para que tais ações possam ser barradas de maneira proativa e proteger a experiência de compradores legítimos. Por questões de confiabilidade e integridade, o monitoramento e registro de transações contam com dados - em sua maioria, tabulares e relacionais - que permitem identificar mais do que apenas os valores de compra, mas também os dados de onde ela foi realizada e por quem foi ordenada. Assim, diversos modelos de Machine Learning (como métodos de classificação) são utilizados atualmente para autorizar estratégias e investigações antifraude diante de uma probabilidade alta de anomalia, comparando-se padrões históricos (Shen; Tong; Deng, 2007).

1.1 Objetivos

Diante de todo o contexto, o objetivo geral deste trabalho é o aprofundamento na análise de transações por meio de redes neurais para grafos, a fim de verificar se as conexões (arestas) entre as transações e seus atributos relevantes (nós) ditam padrões de agrupamento para uma rede neural e trazem profundidade na detecção de fraudes. Utilizando os dados fornecidos pela competição IEEE-CIS Fraud Detection (IEEE Computational Intelligence Society, 2019), faz-se necessária a transformação de dados tabulares em um grafo com atributos e relacionamentos.

Uma vez concluídas as análises, os objetivos específicos incluem:

- Analisar os dados de transação para identificar campos com potencial de nós ou *features*;
- Modelagem dos dados tabulares em um grafo, a fim de aplicar dois modelos de redes neurais para grafos na tarefa de classificação de nós de transação;
- Comparar métricas de avaliação (AUC-ROC, AUC-PR, Revocação, Precisão, F1-Score) entre os modelos SAGEConv e GATConv, bem como o desempenho de diferentes hiperparâmetros;
- Comparar as mesmas métricas de avaliação e o desempenho computacional com um modelo de classificação *baseline*, o XGBoost.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão revisados conceitos fundamentais em relação à temática de fraudes, teoria dos grafos e modelos de redes neurais, bem como suas aplicações. Além disso, haverá análises sobre trabalhos desenvolvidos com fins semelhantes, a fim de demonstrar o atual estado da arte e sua relevância.

2.1 Detecção de fraudes financeiras

Há diversos tipos de fraudes detectáveis no âmbito financeiro. No cenário brasileiro atual, por exemplo, o surgimento do PIX simplificou o processo de recebimento, transferências e pagamentos pelo celular. No entanto, segundo Rimonato e Santos (2021), sua popularidade e democratização às operações financeiras acompanham também um aumento na aplicação de golpes e transações ilegítimas, uma vez que a devolução do dinheiro após o dano é um processo demorado e complexo. Por isso, medidas tecnológicas para limitar os valores de transferência, identificar transações suspeitas e facilitar a devolução do dinheiro são uma pauta extremamente relevante.

Diante desse primeiro cenário, é possível ampliar o contexto para diferentes públicos, desde os mais gerais até os mais específicos. O que as fraudes compartilham, assim como muitos outros comportamentos sociais, são os padrões. A análise de tais padrões é, muitas vezes, complexa. Afinal, há muitas variáveis a serem consideradas para estabelecer uma rotina observável que permita uma **classificação** acurada. É por isso que modelos de Machine Learning capazes de correlacionar parâmetros e predizer a autenticidade de uma transação são altamente estratégicos em cenários de fraude.

Ainda sobre a complexidade dos padrões em transações ilegítimas, muitos métodos e modelos tradicionais podem ser utilizados para detecção de fraudes; no entanto, apesar da boa performance, eles podem ter dificuldade em lidar com dados de alta dimensão, relacionamentos não lineares e a dita complexidade intrínseca entre os usuários e operações (Motie; Raahemi, 2024). A utilização de redes neurais desponta nessa temática como uma opção por sua alta capacidade em revelar padrões sutis e ocultos em dados brutos.

Segundo Wang *et al.* (2021), análises de grafos são interessantes para esse contexto devido à capacidade de considerar informações individuais de uma entidade e também sua informação estrutural, bem como possíveis agregações e atualizações nas relações existentes. Para além da eficácia, é importante considerar que as redes neurais para grafos (GNNs) podem enfrentar desafios de escalabilidade e tempo de processamento em grandes redes de transações financeiras (Nielsen, 2015).

O trabalho de Motie e Raahemi (2024) apresenta uma análise interessante sobre

as inúmeras fontes de fraudes financeiras analisadas por redes neurais para grafos na literatura. Em seu estudo, foram observadas cinco grandes áreas econômicas em que GNNs são aplicadas: **criptomoedas, transações on-line, cartões de crédito, apólices de seguro e taxação** Além disso, há uma forte prevalência de modelos supervisionados (em que as transações fraudulentas são rotuladas), bem como de modelos focados em classificação de nós. Tais conceitos serão aprofundados no decorrer deste capítulo.

2.1.1 Trabalhos relacionados

Dado que o *dataset* utilizado neste trabalho é focado em transações comerciais *on-line*, foram consultados vários artigos e monografias com temáticas semelhantes. Para garantir essa similaridade, os dois primeiros trabalhos em análise utilizam exatamente o IEEE-CIS Fraud Detection (IEEE Computational Intelligence Society, 2019) como fonte de dados para criação de grafos e descoberta de fraude.

A trabalho de Sergadeeva, Lavrova e Zegzhda (2022) traz o processo de modelagem do *dataset* para utilização em um grafo. A remoção de colunas majoritariamente nulas, normalização do valor de transações e comparação do número de épocas para melhores resultados demonstram que, para o modelo utilizado no trabalho, a quantidade de iterações durante o treino foi importante para melhoria das métricas.

Já na obra de Rao *et al.* (2021), além do IEEE-CIS Fraud Detection, utilizam-se outros dois *datasets* (BTFSD e Yelp-Chi) para comparar a performance de diferentes modelos para classificação de fraudes. O artigo é focado na comparação entre GraphSAGE, CARE-GNNN e um *framework* denominado *Knowledge-Guided Semi-supervised GNN*, ou KS-GNN. Esta última é o principal foco, com ênfase em diminuição de ruídos a cada época.

O recente artigo de Guo *et al.* (2025) aborda uma temática e comparação semelhante ao do presente trabalho, debruçando-se sobre a análise de diferentes métodos - tanto para grafos, com GCN, GAT e GIN, quanto para modelos como regressão logística, Transformers e CNNs tradicionais. A diferença principal é que o *dataset* utilizado (*Elliptic Dataset*) já é disponibilizado como um grafo por padrão, sem a necessidade de transformação de dados tabulares em nós, arestas e features.

Um dos interesses da atual pesquisa foi mapear produções acadêmicas sobre o assunto a nível nacional. Silva e Feitosa (2021) traz uma análise de modelos baseados em GCN e GAT, utilizando um *dataset* com forte criação de comunidades para observar a qualidade dos treinamentos. Isso porque, para o *dataset* real baseado em transações de *e-commerce*, a pesquisa foi limitada pelo tamanho do grafo. Isso demonstra que as técnicas empregadas na presente monografia são úteis para a academia como um todo, visto que técnicas para análise de vizinhos e uso de disco persistente para poupar memória RAM seriam valiosas neste caso de uso.

Assumpçao *et al.* (2022) analisa transações, assim como o dataset IEEE-CIS Fraud Detection. No entanto, o foco principal é a detecção de padrões relacionados à lavagem de dinheiro. A partir de dados devidamente confidencializados do banco Inter, um modelo *baseline* implementado com *Deep Graph Library - Knowledge Embedding* é comparado com uma metodologia de multi-task learning, que consiste em combinar duas etapas de aprendizado baseadas em GraphSAGE para captar diferentes padrões nas transações - como probabilidade de transação e estimativa de valor a ser transferido. O artigo foi um bom ponto de interesse em métodos mais sofisticados em pesquisas futuras.

Por fim, o trabalho de Mapel (2023) possui objetivos muito similares à presente monografia: comparar modelos de classificação tradicionais e também diferentes modelos de GNNs para classificar um *target* binário. Embora a temática dos dados seja diferente, com foco principal em saúde, os problemas de desbalanceamento e comparação de métricas são um desafio para ambos os casos de uso. Além disso, algumas das pautas de avaliação dos resultados e da metodologia (como a utilização de *frameworks* para explicação amigável dos critérios adotados pelo modelo de inteligência artificial) serão o foco para trabalhos futuros.

2.2 Modelos de classificação

Um algoritmo de classificação em aprendizado de máquina tem como objetivo definir um conjunto de instruções que permita ao sistema categorizar itens em uma ou mais classes pré-definidas. As características dos dados expostos ao algoritmo são examinadas e, com base em padrões aprendidos pelo modelo, é possível decidir a qual classificação o dado pertence. Um exemplo de aplicação é a proposta pela competição do dataset IEEE-CIS Fraud Detection, disponibilizada no *Kaggle* e utilizada nesta pesquisa. Neste conjunto de dados, há uma coluna (*label*) nomeada is_Fraud, em que a transação em questão deve ser analisada com base no treinamento do modelo e classificada como fraude (1) ou não fraude (0).

Um dos métodos para criação de um classificador é a chamada Árvore de Decisão, em que temos uma estrutura semelhante a um fluxograma. Na Figura 1, vê-se um exemplo de uma pergunta inicial (raiz da árvore). Cada resposta leva a um nó intermediário (folha da árvore), com uma pergunta que leva em conta outros atributos hierárquicos e tenta levar à classificação correta em relação à classe alvo (rejeitar ou não uma transação com base na sua tendência à fraude).

2.2.1 XGBoost

O algoritmo XGBoost propõe-se a utilizar a técnica de Gradient Boosting, que cria árvores de decisão sequenciais com o objetivo de corrigir os erros cometidos pelas árvores de decisão (modelos) anteriores e, assim, reduzir a função de perda (Mapel, 2023). Para

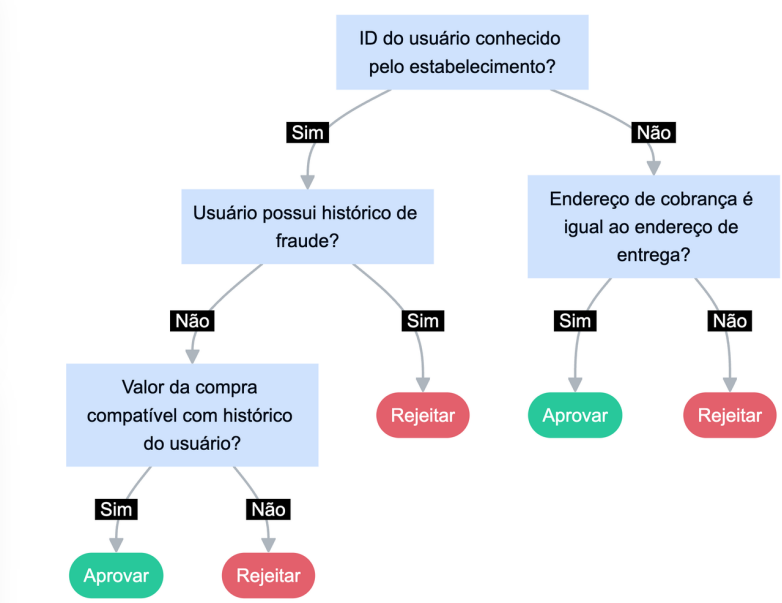

Figura 1 – Árvore de Decisão para classificação de possível fraude. Fonte: Narde (2024)

além disso, seu maior diferencial está na sua escalabilidade: segundo Chen e Guestrin (2016), o sistema é 10 vezes mais rápido em execução do que outras soluções populares em uma única máquina, isso por conta de importantes otimizações feitas a nível de algoritmo e sistema - como um novo algoritmo para lidar com dados esparsos e computação paralela e distribuída. Isso faz com que sua eficiência seja aumentada e a velocidade, aprimorada.

2.3 Teoria dos Grafos e Redes Complexas

A análise de relacionamento é uma ferramenta poderosa em uma sociedade cada vez mais conectada. A conexão entre indivíduos, ferramentas e quaisquer unidades observáveis traz informações importantes sobre seu comportamento em um ambiente controlado. Um exemplo disso são as redes sociais, em que amigos e interesses comuns podem ditar padrões de consumo de conteúdo e até mesmo compra. Uma maneira de representar tais elementos e seus relacionamentos é a utilização de grafos, como representado na Figura 2.

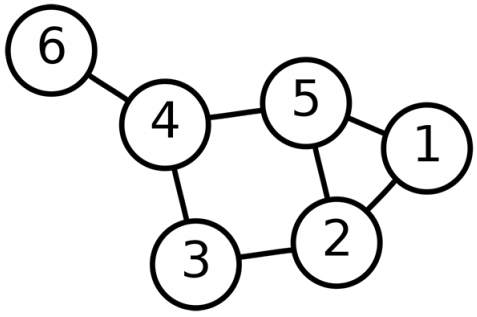

Figura 2 – Representação de um grafo, com 6 nós e 7 arestas. Fonte: Mariano (2020)

Segundo Menczer, Fortunato e Davis (2020)

> Em termos bem genéricos, uma rede, ou grafo, é um conjunto de elementos, que chamamos de nós, juntamente com um conjunto de conexões entre pares de nós, que chamamos de arestas. As arestas representam a presença de uma relação entre os elementos representados pelos nós. [...] As arestas podem corresponder a interações sociais, físicas, de comunicação, geográficas, conceituais, químicas, biológicas ou outras.

Matematicamente falando, um grafo G pode ser representado como $G(N, L)$, um conjunto em que N é o número de nós e L é o conjunto de arestas, também chamado de lista de adjacências (Gomes, 2024), e pode ser representado como (i,j) - descrevendo a conexão entre os nós i e j. No campo de redes complexas, segundo Metz *et al.* (2007), o termo "refere-se a um grafo que apresenta uma estrutura topológica não trivial", de forma que quando o conceito é aplicado a redes do mundo real, sua topologia e evolução "apresentam propriedades organizacionais bastante robustas".

Ainda segundo Metz *et al.* (2007), é interessante ressaltar que um grafo só pode ser considerado uma rede complexa caso apresente propriedades topográficas que não estão presentes em grafos simples. Um exemplo é o coeficiente de aglomeração, que dita a tendência de conexão entre os nós A e C, caso ambos sejam conectados a B. No campo de análises relacionadas a redes complexas, a criação de comunidades (ou *clustering*) demonstra a tendência de agrupamento entre nós com conexões similares. Na Figura 3, demonstra-se a criação de grupos de afinidade em relação às redes sociais, mas isso também se reflete em temáticas de artigos, agrupamentos geográficos e páginas da *internet*.

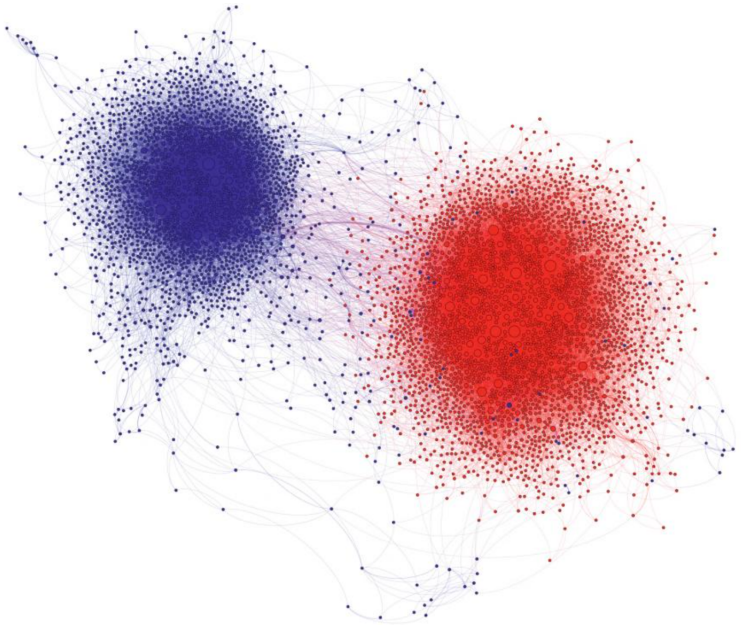

Figura 3 – Demonstração de um grafo sobre retweets em relação à política nos EUA. Fonte: Menczer, Fortunato e Davis (2020)

2.3.1 Propriedades de um grafo

Há diversos comportamentos a serem notados em um grafo, que dizem muito sobre as características dos dados nele inseridos. Desde a configuração de ligação entre os nós até mesmo suas tendências de agrupamento podem trazer informações valiosas sobre o conjunto. Um grafo é chamado **direcionado** quando a direção das arestas importa quando ligam dois nós; já quando apenas é necessário sinalizar a ligação, sem que a direção importe, o grafo é chamado **não direcionado**. As duas configurações são ilustradas na Figura4. Além disso, as arestas podem apresentar **pesos** distintos, de maneira que a relação entre nós específicos seja mais forte ou significativa para o modelo do que as demais.

Quando se fala sobre configurações dos nós em um grafo não direcionado, pode-se definir o conceito de **grau** k do vértice i na Equação 2.1, dado pela quantidade de arestas conectadas a i em uma rede com m arestas.

$$k_i = \sum_{j=1}^{n} A_{ij} \quad \rightarrow \quad \sum_i k_i = 2m \tag{2.1}$$

Para toda a rede não direcionada, calcula-se o grau médio partindo da quantidade de arestas m, como na Equação 2.2:

$$m = \frac{1}{2}\sum_i k_i = \frac{1}{2}\sum_i \sum_j A_{ij} \quad \rightarrow \quad \mu = 1/n \sum_i k_i \quad \rightarrow \quad \mu = \frac{2m}{n} \tag{2.2}$$

Já numa rede direcionada, é importante diferenciar o grau de entrada k_{in} e o grau de saída k_{out}, já que a direção é relevante para a análise. Sendo $k_i^{in} = \sum_{j=1}^{n} A_{ij}$ e $k_j^{out} = \sum_{i=1}^{n} A_{ij}$, o número de arestas m na rede é dada pela somatória das arestas de entrada e saída (veja a Equação 2.3).

$$m = \sum_i k_i^{in} = \sum_i k_j^{out} = \frac{1}{2}\sum_i \sum_j A_{ij} \tag{2.3}$$

Há, ainda, o conceito de comunidades. Diante de um conjunto de nós com alto grau de conexão entre si, é possível observar as tendências de agrupamento desses elementos e a influência exercida, tanto pelos nós quanto por suas conexões. Em uma rede cujos nós tem uma forte inclinação a serem mais conectados quando são semelhantes entre si, é possível observar o princípio de **homofilia**. Já quando se vê uma tendência de agrupamento entre nós que não compartilham tantas semelhanças, observa-se a **heterofilia**.

2.4 Redes Neurais para Grafos

Quando um ser humano busca aprender novos conceitos, uma das maneiras de memorizar conceitos é a repetição. Para identificar um pássaro, por exemplo, há algumas estruturas fundamentais que uma criança começa a associar: a existência de um bico,

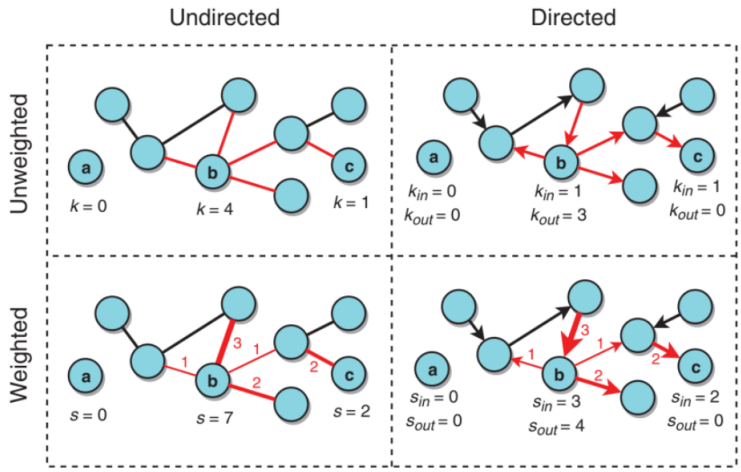

Figura 4 – Grafos com propriedades distintas de direção e peso. Fonte: Menczer, Fortunato e Davis (2020)

corpo coberto de penas, canto característico. Quanto mais pássaros lhe são apresentados, mais concreta se torna a ideia. Da mesma forma, pode-se ensinar a um sistema sobre um ou mais conceitos específicos.

No entanto, como abordado por Nielsen (2015), para além da complexidade de transformar intuições comuns em algoritmos, há muitos casos especiais e exceções difíceis de serem traduzidos. Ainda no exemplo dos pássaros, um ornitorrinco possui um bico, mesmo que seja outro tipo de animal. Para trazer uma outra abordagem a esse problema, as redes neurais trazem ao sistema vários exemplos de pássaros (entradas, ou *inputs*) e os usam para inferir regras para o seu reconhecimento (Nielsen, 2015). A Figura 5 ilustra, de maneira didática, o processo de decisão. Quanto mais exemplos, o modelo poderá aprender ainda mais sobre o que realmente classifica um pássaro e, assim, melhorar sua acurácia.

Cada uma das camadas de aprendizado (as chamadas camadas ocultas, ou *hidden layers*) podem ter pesos diferentes. Se o animal não tem penas, como o ornitorrinco, muito provavelmente não será um pássaro; poderíamos atribuir um peso maior ($w_1 = 2$) a essa variável. Já a capacidade de canto pode estar presente em aves como a galinha, que não é um pássaro por definição. Portanto, seu peso ($w_2 = 1$) será menor. Além da inferência de pesos, as camadas poderiam ser divididas em outras sub-redes. A questão sobre o canto, por exemplo, poderia ser decomposta em relação a cacarejos ou melodias. Redes com uma estrutura de várias camadas – duas ou mais camadas ocultas – são chamadas de redes neurais profundas *(deep neural networks)*.

Valendo-se do conceito de grafos, pode-se trabalhar com o aprendizado de padrões por meio das relações. Consideram-se, ainda, os conceitos de redes neurais em relação às camadas de aprendizado e pesos, mas com cada um dos nós possuindo características próprias, que podem ser enriquecidas pelos atributos dos vizinhos com os quais se relaci-

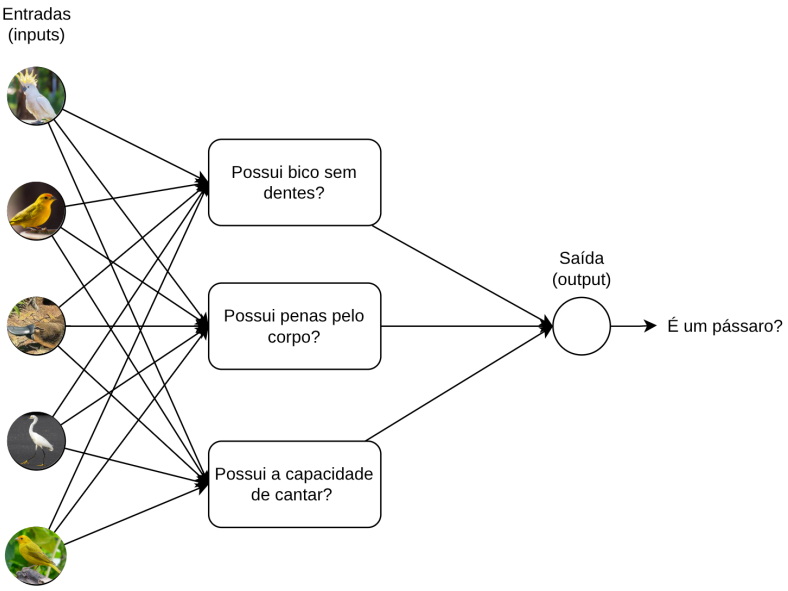

Figura 5 – Exemplo simples de classificação de pássaros com uma camada oculta e uma saída. Fonte: da autora

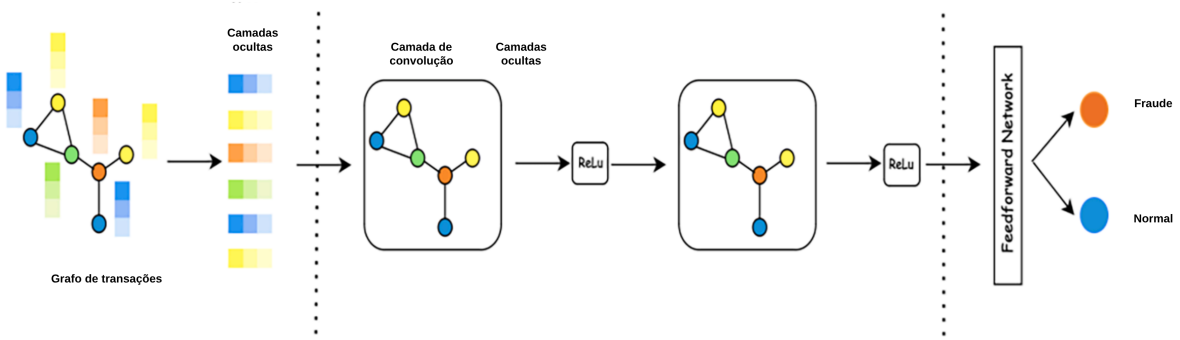

Figura 6 – Redes neurais para grafos com aspecto convolucional em tarefas de classificação para fraudes. Fonte: Adaptação de Motie e Raahemi (2024)

onam. Assim, é possível trabalhar em tarefas supervisionadas (classificação, regressão), não-supervisionadas (agrupamento de comunidades) e semi-supervisionadas. É possível, ainda, classificar nós, arestas e grafos completos. Neste trabalho, nos aprofundaremos em um modelo supervisionado para classificação de nós, como exemplificado na Figura 6.

Sendo o grafo constituído por nós que representam diversos tipos de ave, pode-se classificar um nó como sendo um pássaro baseado em seus atributos e relações. A Figura 7 exemplifica o cenário em que os nós poderiam se relacionar pelos tipos de atributos ou vizinhos compartilhados - como um canário e um sabiá - e, assim, fornecer ao modelo insumos significativos para predição do que é um pássaro ou não.

No cenário financeiro, a classificação de nós é muito eficiente em detectar fraudes de seguro e cartão de crédito analisando padrões de transações e relações entre reclamante

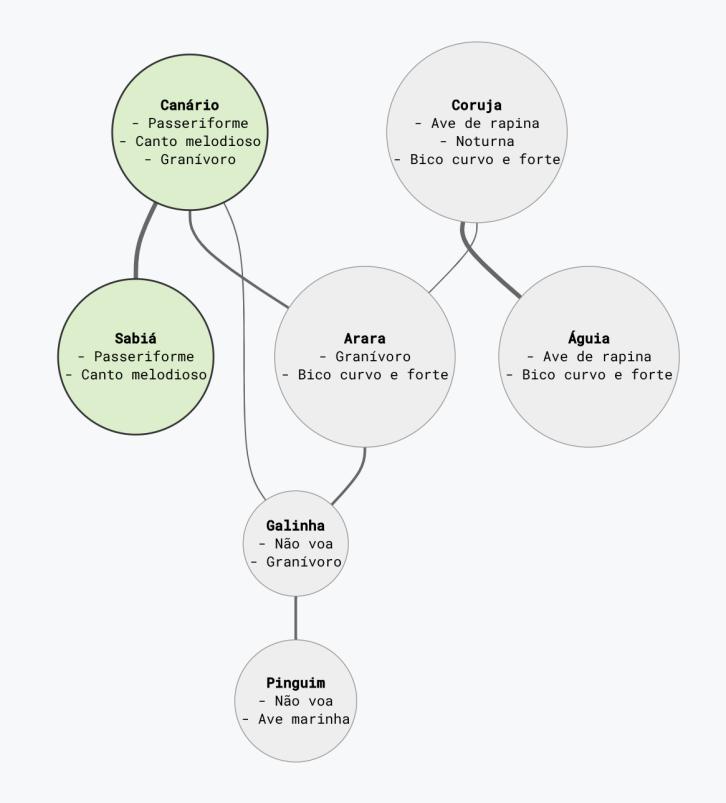

Figura 7 – Exemplo de grafo cujos nós são conectados pela semelhança de atributos. Fonte: da autora

e provedor (Cheng *et al.*, 2025). Ainda segundo Cheng *et al.* (2025):

> Dados de transações são essenciais para revelar atividades fraudulentas. A criação de uma rede de transações (grafo), com nós representando contas ou entidades individuais (como indivíduos, empresas) e arestas representando transações (por exemplo, transferências, pagamentos), permite a identificação eficaz de padrões anormais de transações. Por exemplo, pequenas transações frequentes ou grandes fluxos de fundos em um curto período podem indicar atividade fraudulenta. Analisar a topologia da rede, como a centralidade dos nós e as divisões da comunidade, pode ajudar a identificar redes de fraude ou contas fraudulentas importantes.

O principal mecanismo por trás do funcionamento de redes neurais para grafos é a passagem de mensagens (Tozato, 2021). Para cada nó do grafo, a intenção é calcular um novo vetor (ou *embedding*) que leve em consideração não apenas as próprias características, mas as características dos seus vizinhos. Ainda segundo Tozato (2021), "As camadas de uma rede neural baseada em grafos podem ser entendidas como uma série de agregações de representações em conjunto do mecanismo de passagem de mensagens, com o intuito de calcular a representação de um dado vértice no grafo".

A mensagem passada de um nó vizinho j para o nó-alvo i é dada pela Equação 2.4:

$$\mathbf{m}_{j \to i} = \phi\left(\mathbf{h}_j^{(k)}\right) = \mathbf{W}^{(k)} \cdot \mathbf{h}_j^{(k)}, \tag{2.4}$$

em que $h_j^{(k)}$ é o vetor de características na camada k e $W^{(k)}$ é uma matriz de pesos usada pela rede.

A função de agregação $a_i^{(k)}$ utilizada para combinar as mensagens em um único vetor pode se valer de mecanismos de soma, média e *max-pooling*, que representam diferentes meios para lidar com os valores vindos dos vizinhos. Após esse processo, para atualizar o vetor final da próxima camada, espera-se que haja uma combinação entre o vetor original e vetor agregado pelos vizinhos.

2.4.1 Graph Convolutional Networks

O modelo GCN é inspirado nas chamadas *Convolutional Neural Networks* (CNNs), valendo-se de operações convolucionais ou de agrupamento na estrutura do grafo para extrair uma representação para cada nó e, em seguida, utilizá-la na classificação Xiao *et al.* (2022). A regra de propagação numa GCN de múltiplas camadas é dada por:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2.5}$$

onde $\widetilde{A} = A + l_N$ é a matriz de adjacências do grafo não direcionado G com autoconexões adicionadas, l_N é a matriz de identidade, $\widetilde{D}_{ii} = \Sigma_j \widetilde{A}_{ij}$ e $W^{(l)}$ é uma matriz de pesos treinável (Motie; Raahemi, 2024). σ representa a função de ativação, $H^{(l)} \in R^{N \times D}$ é a matriz de ativações na camada l; sendo $H(0) = X$ a representação da primeira camada convolucional (Kipf, 2016).

2.4.1.1 Funções de ativação e perda, *backpropagation* e otimizador

A **função de ativação** mencionada anteriormente desempenha um papel importante na definição da saída do modelo, uma vez que sua utilização insere a não-linearidade necessária no sistema (Ultralytics, 2025). Isso quer dizer que o sinal advindo de um neurônio específico é disparado e modulado em intensidade para a próxima camada seguindo uma dita função não-linear, a fim de captar padrões e relações não triviais para um modelo simples de regressão linear.

Há algumas funções que se destacam no campo de redes neurais, como a função sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$ e tangente hiperbólica $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Atualmente, uma das soluções mais implementadas é a função ReLU, cuja fórmula é dada pela Equação 2.6. Em resumo, caso a entrada seja positiva, é retornado seu próprio valor; caso contrário, é retornado o valor 0.

$$f(x) = \max(0, x) \tag{2.6}$$

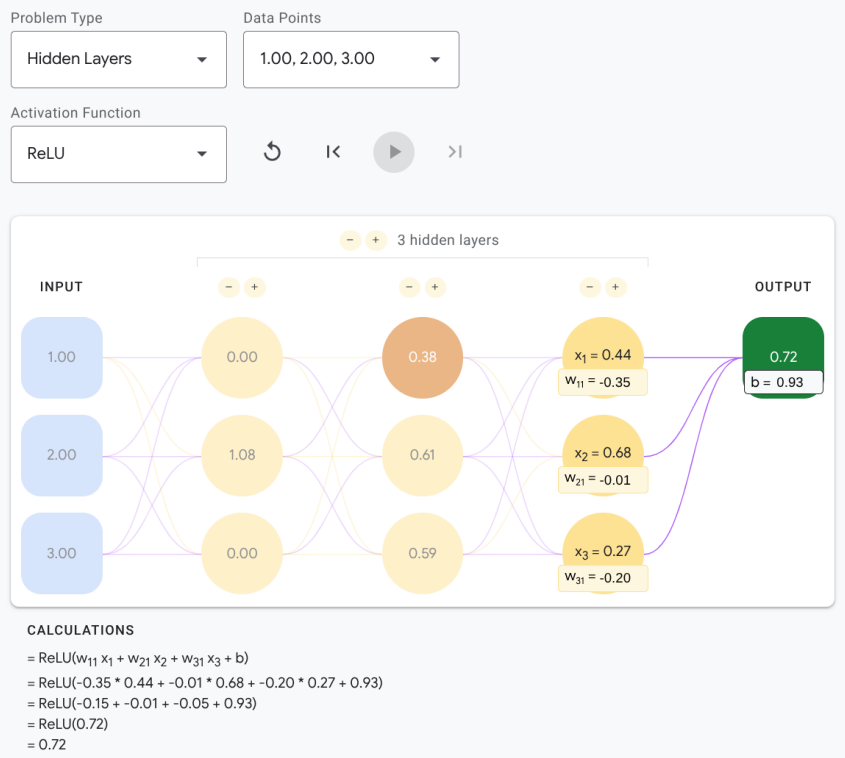

Figura 8 – Comportamento da função de ativação ReLU em modelo com 3 camadas ocultas. Fonte: Google for Developers (2025c)

Sua eficiência é tanto computacional quanto em relação ao contorno do fenômeno do gradiente de desaparecimento. Ele se dá quando os gradientes - ou seja, as derivadas da função de ativação - tornam-se muito pequenos ou nulos durante as multiplicações através das camadas. Isso pode acontecer em funções como a sigmoide, em que o valor máximo da derivada é alcançado em $x = 0$ e vai diminuir ao se afastar desse valor. A Figura 8 demonstra como os cálculos são feitos entre camadas ocultas com a função ReLU.

Já com a **função de perda**, podemos fazer uma comparação entre a saída $y(x^{(i)})$ fornecida pelo treinamento da rede após a entrada $(x^{(i)})$ e a resposta esperada $a(x^{(i)})$ (Goulart, 2022). Um exemplo de função muito utilizada é a *Cross Entropy Loss*, ou função perda de entropia cruzada, dada pela Equação 2.7. Nela, se considera p_i o indicador binário para a classe correta da observação i e q_i é a probabilidade prevista na observação específica da classe i.

$$H(p, q) = -\sum_{i} p_i \log q_i. \tag{2.7}$$

O ***backpropagation*** é uma técnica sofisticada para calcular as derivadas parciais - os já mencionados gradientes - por entre as camadas da rede neural sem que seja extremamente custoso computacionalmente (Goulart, 2022). Após a propagação do modelo (ou seja, já passada a transformação linear, aplicação de função de ativação e também a de

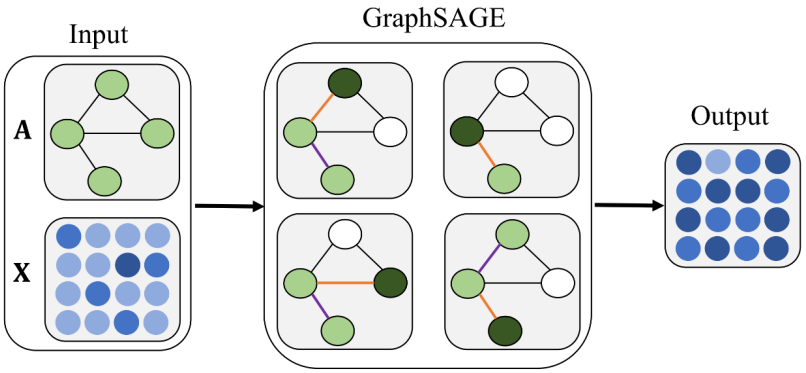

Figura 9 – Amostragem, agregação de features entre vizinhos e classificação de nós com GraphSAGE. Fonte: Xiao *et al.* (2022)

perda), o valor da perda é utilizado para cálculo de um gradiente, que será propagado para a última camada da rede - de "trás para frente". Recursivamente, a perda é distribuída de maneira eficiente a cada peso individual da rede.

E, por sua vez, os **otimizadores** buscam minimizar o valor da função de perda, ajustando os parâmetros aprendidos da rede. Um exemplo é o Adam, um método eficiente que requer pouca memória. Segundo (Kingma; Ba, 2014), "ele calcula taxas de aprendizagem adaptativa individuais para diferentes parâmetros a partir de estimativas", o que é muito útil e eficiente para redes com gradientes esparsos e numerosos.

2.4.1.2 GraphSAGE

O **GraphSAGE** é um exemplo de algoritmo que vale-se do princípio de GCNs para criar um processo **indutivo** sobre os nós de um grafo, usando agregações de características da vizinhança. Com ele, é possível gerar *embeddings* para dados não vistos anteriormente, visto que seu objetivo é treinar um conjunto de funções agregadoras que combinam informações de diferentes saltos (*hops*) (Hamilton; Ying; Leskovec, 2017). O processo de propagação deste modelo é demonstrado pela Equação 2.6:

$$h_v^k = \sigma\left(W_k \cdot \text{CONCAT}\left(AGG\left(h_u^{k-1}, \forall u \in N(\nu)\right), h_v^{k-1}\right)\right), \forall k \in \{1, \ldots, K\} \qquad (2.8)$$

em que $CONCAT$ demonstra o processo de combinação entre os vetores descrito anteriormente e o método AGG refere-se à função de agregação escolhida para o algoritmo (Tozato, 2021). A Figura 9 demonstra o funcionamento da arquitetura.

2.4.2 Graph Attention Networks

As arquiteturas GAT são versáteis - devido à aplicabilidade em cenários indutivos e transdutivos - e orientadas a atribuir diferentes importâncias aos nós de uma mesma

vizinhança, especificando implicitamente pesos diferentes a cada um deles (Veličković *et al.*, 2017). Na Equação 2.7, demonstra-se o cálculo do coeficiente de atenção e_{vu}, que indica a importância das features do nó u para o nó v.

$$e_{vu} = a\left(W_k h_u^{k-1}, W_k h_v^{k-1}\right) \tag{2.9}$$

Para que os coeficientes possam ser facilmente comparáveis entre diferentes nós, é necessário normalizá-los através de todas as escolhas de v usando uma função chamada *softmax* (Veličković *et al.*, 2017) - uma generalização da função sigmoide para casos não-binários (Ceccon, 2020) - dada pela Equação 2.8. É interessante ressaltar que, relacionando as equações 2.7 e 2.8, o mecanismo de atenção a é uma rede neural de camada única - baseada em um vetor de pesos $\overrightarrow{\mathbf{a}}$ com função de ativação aplicada - a LeakyReLU, que aplica um fator de divisão nos valores negativos para torná-los muito pequenos, ao invés de zero (Ceccon, 2020).

$$\alpha_{vu} = \frac{\exp\left(e_{vu}\right)}{\sum_{k \in N(v)} \exp\left(e_{vk}\right)} \tag{2.10}$$

Para estabilizar o aprendizado, é possível, ainda, trabalhar com uma diferente quantidade de cabeças (*heads*) de atenção. Na prática, isso quer dizer que cada uma das *heads* calculará seus próprios coeficientes de atenção normalizados α_{vu} de maneira paralela para cada nó. Ao final, as operações de concatenação e/ou cálculo da média aplicadas nas features agregadas (Veličković *et al.*, 2017) resultam na seguinte função de propagação:

$$h_v^k = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} W_k h_u^{k-1}\right), \forall k \in \{1, \ldots, K\} \tag{2.11}$$

Em que σ é uma função de não-linearidade qualquer (Tozato, 2021). A Figura 10 ilustra o processo de cálculo de vários coeficientes de atenção, até o resultado final em $h_i^{'}$.

2.5 Métricas de avaliação

Em modelos de classificação, é indispensável avaliar a qualidade das predições e o desempenho alcançado. Isso significa entender quão bem o modelo é capaz de distinguir as classes através de métricas calculáveis a partir de equações matemáticas. Nem todo número é relevante para casos de uso específicos; a escolha de métricas a se considerar leva em conta a tarefa em questão, o custo de diferentes classificações erradas e se o *dataset* utilizado é balanceado ou desbalanceado - ou seja, se há muita diferença entre a quantidade de dados de cada classe (Google for Developers, 2025a).

Os cálculos são feitos usando os conceitos de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). Como os nomes sugerem,

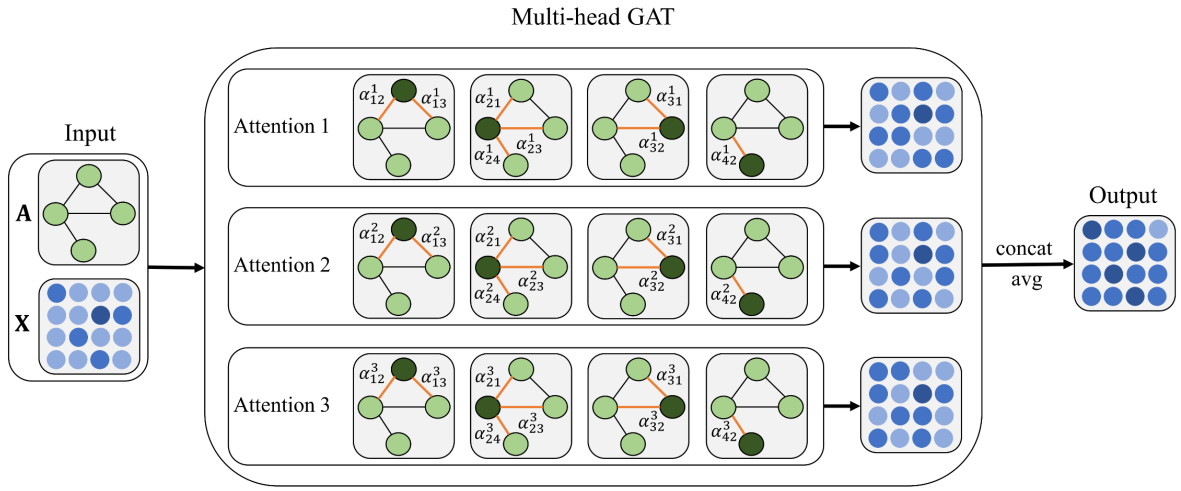

Figura 10 – Representação da propagação a partir de coeficientes de atenção normalizados aplicados a 3 *heads* (K = 3). Fonte: Xiao *et al.* (2022)

eles apresentam os resultados alcançados pelo modelo no que diz respeito à classificação binária em relação a uma *label* específica. É possível visualizá-los em quadrantes para entender a distribuição real e a distribuição prevista. A Tabela 1 ilustra uma matriz de confusão em um exemplo aplicável para este trabalho.

Tabela 1 – Matriz de confusão para classificação de transações fraudulentas. Fonte: Adaptado de Google for Developers (2025d)

	Positivo real	**Negativo real**
Positivo previsto	**Verdadeiro positivo (VP)**: uma transação fraudulenta classificada corretamente como fraude. A transação será interrompida.	**Falso positivo (FP)**: uma transação legítima classificada incorretamente como fraude. Necessitará de confirmação pelo cliente (fricção).
Negativo previsto	**Falso negativo (FN)**: uma transação fraudulenta classificada incorretamente como legítima. Causa prejuízo (para o negócio e o cliente).	**Verdadeiro negativo (VN)**: uma transação legítima classificada corretamente como normal. Será concretizada normalmente.

Com tais informações, é possível calcular métricas como as seguintes:

- **Revocação, *recall* ou True Positive Rate (TPR):** $TPR = \frac{TP}{TP+FN}$
  - Razão entre os reais positivos classificados corretamente e todos os reais positivos da amostra.
  - Boa métrica para observar em um *dataset* desbalanceado e casos de uso em que o custo de um falso negativo é maior do que o de um falso positivo.

- **Precisão ou *precision*:** $Precisão = \frac{TP}{TP+FP}$

  - Razão entre os reais positivos classificados corretamente e tudo o que foi classificado como positivo.
  - Métrica menos significativa em *datasets* desbalanceados (Google for Developers, 2025a). Relevante em casos de uso em que o custo de um falso positivo é maior do que o de um falso negativo.

- **F1-*Score*:** $\mathrm{F1} = 2 * \frac{\text{precision } * \text{ recall}}{\text{precision + recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$
  - Média harmônica entre precisão e revocação.
  - Boa métrica para observar em um *dataset* desbalanceado. Quando a precisão e a revocação tem valores próximos, o valor de *F1-Score* terá um valor semelhante ao deles. Quando estão muito distantes, a métrica ficará mais próxima do pior valor entre precisão e revocação.

Outro conceito importante é a curva ROC (*Receiver-operation characteristic curve*), que representa visualmente o desempenho do modelo por meio da plotagem da taxa de verdadeiros positivos sobre a taxa de falsos positivos em cada possibilidade de limite do modelo - também chamado de probabilidade de *threshold*, que se refere ao valor de corte para que uma classe seja considerada positiva ou negativa. A área sob a curva ROC, dada pela métrica AUC-ROC, representa a probabilidade de que o modelo, ao receber um exemplo positivo e um negativo escolhidos aleatoriamente, classifique o positivo com uma probabilidade maior do que o negativo (Google for Developers, 2025b).

A Figura 11 demonstra o porquê de AUC-ROC ser uma métrica difícil de confiar em *datasets* desbalanceados. Em problemas desse tipo, caso o modelo seja altamente permissivo e tenha a classe majoritária como palpite unânime em cada iteração, a acurácia será expressiva (mesmo que o modelo não consiga detectar a classe positiva). Neste caso, há também a curva PR (*Precision-Recall*), em que são plotados os valores de precisão no eixo Y e o valor de revocação no eixo X, passando por todos os *thresholds* do modelo. O valor da área sob essa curva (AUC-PR), como ilustrado pela Figura 12 é mais uma possibilidade de avaliação da qualidade do modelo.

2.6 Comparação estatística

Para que as métricas de avaliação possam ser comparadas, é necessário adotar critérios claros. Para isso, muitas obras acadêmicas valem-se de técnicas estatísticas diferentes e até do senso comum na decisão entre diferenças reais ou aleatórias entre algoritmos (Demšar, 2006). Assim, trabalharemos com duas hipóteses principais, considerando inicialmente a métrica AUC-PR: H_0 = os resultados são estatisticamente iguais; ou H_1 = os resultados são estatisticamente diferentes.

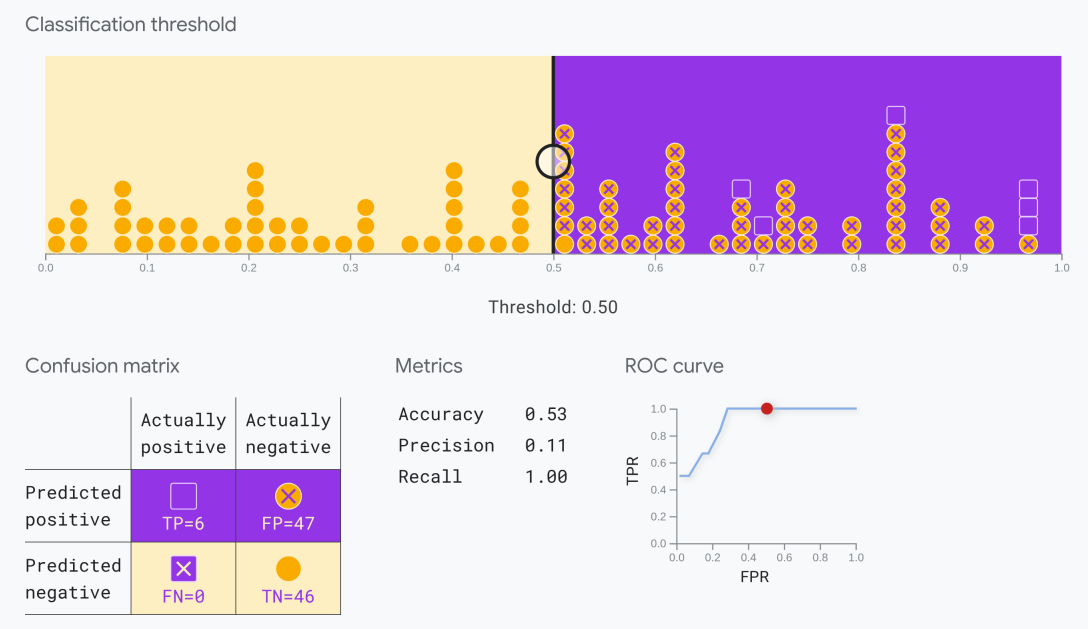

Figura 11 – Demonstração de métricas e curva ROC com threshold de 0.5 em *dataset* desbalanceado. Fonte: Google for Developers (2025b)

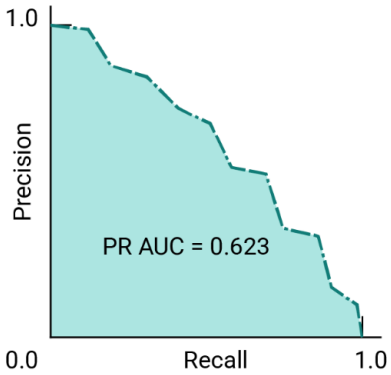

Figura 12 – Curva PR, com AUC-PR = 0,623. Útil para *dataset* desbalanceados. Fonte: Google for Developers (2025b)

A fim de testar tais hipóteses, será aplicado o Teste de Friedman. Com ele, utiliza-se a Equação 2.12, em que N é o número de métricas avaliadas (linhas) e k é o número de classificadores avaliados (colunas). O valor de χ_F^2, conhecido como "qui-quadrado", é dado pela Equação 2.13, com a nova variável R, referente ao valor de *ranking* - os valores de cada uma das métricas são *rankeados* de i_1 a i_k, do maior ao menor, e lhe é atribuída uma média (Wallis, 2021).

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{2.12}$$

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{2.13}$$

O valor calculado na Equação 2.12 será comparado a um número denominado valor crítico, que pode ser encontrado em livros de estatística (Demšar, 2006). Ele é distribuído em relação a um nível de significância α entre dois graus de liberdade dados por $k-1$ e $(k-1) \times (N-1)$. Um exemplo seria uma tabela com nível de significância $\alpha = 0,05$, cujos dados correspondem a 5 métricas calculadas para 4 classificadores. Assim, os graus de liberdade serão 4 $(5-1)$ e 12 $((5-1) \times (4-1))$, respectivamente. Se o valor crítico for menor do que F_F, a hipótese H_0 é refutada (Wallis, 2021).

Por fim, coloca-se em prática o teste de Nemenyi, em que a performance de dois classificadores será considerada diferente se a média correspondente dos *rankings* mencionados anteriormente diferem em um valor maior ou igual à distância crítica (CD), calculada pela Equação 2.14.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{2.14}$$

3 METODOLOGIA

Neste capítulo, descrevem-se os materiais e procedimentos metodológicos utilizados para a aplicação, bem como a argumentação em relação às escolhas dentre modelos e ferramentas.

3.1 Dataset IEEE-CIS Fraud Detection

O *dataset* utilizado foi publicado na plataforma Kaggle para uma competição baseada em submissões de detecção de fraude com AUC-ROC mais representativos. Conforme publicado por IEEE Computational Intelligence Society (2019), "os dados são provenientes de transações reais de e-commerce da Vesta e contêm uma ampla gama de recursos, desde o tipo de dispositivo até as características do produto".

Ao explorar os canais de discussão sobre os dados, é possível aprimorar a análise pelos esclarecimentos do *host* da competição: o *dataset* contém transações variadas em relação a *e-commerces*, desde transferências de dinheiro até serviços de presente e outros tipos de consumo. As Tabelas 2 e 3 descrevem o tipo e descrição das *features* dos datasets *train_transaction* e *train_identity*, respectivamente.

3.1.1 Análise e preparação dos dados

Para iniciar o processo de análise do dataset, foram adotadas diferentes metodologias para navegação em suas linhas e colunas. Em relação às linguagens utilizadas, duas foram significativas para o processo:

1. Python, com bibliotecas *numpy* e *pandas* no Google Colab Enterprise
2. SQL, utilizando a ferramenta de *data warehouse* do Google Cloud, o BigQuery

É interessante ressaltar que o **Data Science Agent do Google Colab Enterprise** desempenhou um excelente papel na validação do código e sugestões de melhorias para performance. Em ambas as metodologias, os datasets de treino e teste, tanto para transações quanto para identidades, foram carregados nas plataformas em formato CSV. No BigQuery, há a detecção automática de schema e também insights com IA generativa sobre as colunas. Tal perspectiva foi interessante para a definição das entidades candidatas a nós. Já no Google Colab, as análises realizadas foram orientadas a aspectos quantitativos, como:

- Quantidade de linhas e colunas dos dados de treinamento (*Transaction* e *Identity*)
- Quantidade de valores únicos para cada coluna nos dados de treinamento

Tabela 2 – Tipo e descrição das *features* do *dataset train_transaction*

Feature	**Tipo**	**Descrição**
TransactionID	Numérica	Identidade da transação. Utilizada para merge com Identity.
isFraud	Numérica	Feature alvo - fraude (1) ou não fraude (0)
TransactionDT	Numérica	Tempo corrido (time delta) dado um horário de referência qualquer
TransactionAMT	Numérica	Montante de pagamento feito pela transação (em USD)
ProductCD	Categórica	Código (caractere) do produto na transação
card1-card6	Categórica	Informações de cartão de pagamento. Inclui tipo de cartão, categoria, banco, país.
addr1, addr2	Categórica	Ambos sobre o comprador, sendo (1) para região e (2) para país.
dist	Numérica	Distâncias entre endereço de cobrança, endereço de e-mail, código postal, endereço IP, área de telefone, etc.
P_emaildomain	Categórica	Endereço de e-mail do comprador
R_emaildomain	Categórica	Endereço de e-mail do recebedor
C1-C14	Numérica	Contagem, como quantos endereços são encontrados associados ao cartão de pagamento, etc. O significado real é mascarado.
D1-D15	Numéricas	Timedelta, como dias entre transações anteriores
M1-M9	Categórica	Correspondência, como nomes no cartão e endereço, etc.
Vxxx	Numéricas	Vesta desenvolveu recursos avançados, incluindo classificação, contagem e outras relações entre entidades. Por exemplo, quantas vezes o cartão de pagamento associado a um IP e e-mail ou endereço apareceu em um período de 24 horas

Tabela 3 – Tipo e descrição das *features* do dataset *train_identity*

Feature	**Tipo**	**Descrição**
DeviceType	Categórica	Transação feita por mobile ou desktop.
DeviceInfo	Categórica	Informação sobre o celular ou computador de onde a transação foi feita
id_01 - id_11, id_12 - id_38	Numérica, Categórica	Dados coletados pela Vesta e parceiros de segurança, como classificação de dispositivos, classificação de domínio IP, classificação de proxy, etc. Também são registradas impressões digitais comportamentais, como horários de login/falha ao fazer login na conta, tempo de permanência da conta na página, etc.

- Quantidade de colunas com maior porcentagem de dados ausentes, para tratamento posterior
- Porcentagem de transações fraudulentas em relação às transações legítimas (grau de desbalanceamento)
- Análise da correlação entre as colunas do dataset e o rótulo *isFraud*
- Modelagem de nós com base na relevância das colunas

Foram realizados experimentos com a seguinte configuração de nós:

- Um grafo heterogêneo com nó alvo *transaction* e nós de entidades individuais (card1, card2, card3, card4, card5, card6, addr1, addr2, P_emaildomain, R_emaildomain, ProductCD, DeviceInfo, DeviceType e id_01 a id_38). O nó *transaction* é conectado às entidades específicas da transação.

Além disso, foram adotados os processamentos de *features* abaixo. A arquitetura do grafo é ilustrada pela Figura 13.

- Colunas que não são nós e que possuem menos de 90% dos valores diferentes de *NaN* são features;
- *Features* categóricas nulas são preenchidas com *Missing*. Quando apresentam baixa cardinalidade (ou seja, poucas categorias distintas) foram tratadas com One-Hot Encoding e as de maior cardinalidade foram tratadas com LabelEncoder (para evitar a criação de muitas colunas extra sem grandes ganhos). Em um segundo momento, cada um dos nós de identidade serão processados por uma camada de *embedding* (etapa para transformá-los em vetores densos visualizáveis pela rede neural);
- *Features* numéricas nulas são preenchidas com a mediana para não serem afetados por *outliers*. Elas são colocadas em uma escala comum usando StandardScaler para dados de entrada mais padronizados, ajudando na convergência das redes neurais;
- A feature *TransactionAmt* teve transformação logarítmica aplicada para normalização. Caso haja $log(0)$, o valor é substituído por *NaN* e tratado com a regra de *features* numéricas acima.

Para que o sistema fosse mais facilmente replicável, foram adotadas medidas para diminuir o consumo de memória RAM:

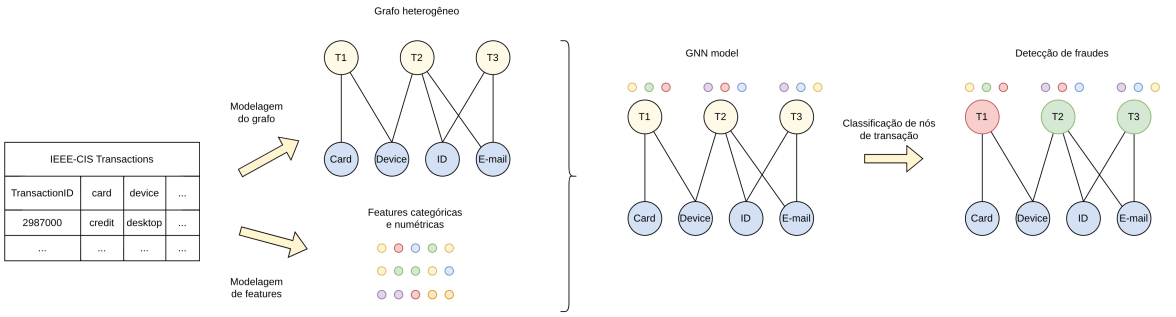

Figura 13 – Modelagem do grafo a partir de dados tabulares e processamento da GNN. Fonte: da autora, adaptado de Wang *et al.* (2021)

- Centralização de *features* e *tags* em arquivos .csv, para permitir a divisão da modelagem em mais de um *notebook*, uma vez que os arquivos persistem no disco. A abordagem foi inspirada pelo código de Wang (2021).

- Utilização da biblioteca **gc** para coleta de lixo (*garbage collector*), permitindo a deleção de *DataFrames* após sua utilização no código e liberação de memória RAM.

3.2 Grafo modelado e processamento das GNNs

Com as dados tratados, os nós escolhidos e as features modeladas, o método *HeteroData()* da biblioteca *Pytorch Geometric* foi utilizado na construção do grafo, salvo em um arquivo .pt. Assim, as features e rótulos de acordo com a variável alvo (*isFraud*) são carregados em cada um dos nós de transação como parâmetros x e y, respectivamente. Para simplificação devido ao tamanho dos arquivos, a partir dos nós criados com o merge de *train_transaction* e *train_identity*, cria-se máscaras distintas para treino (60%), validação (20%) e teste (20%). O código pode ser visualizado no repositório referenciado pelo Apêndice A.

3.2.1 Classe HeteroGNN

A classe HeteroGNN é a base para todos os experimentos realizados, sendo ela a principal fonte das alterações de parâmetros e também de métodos para GNNs. Ela conta com as funções:

- ___init___ , em que são inicializadas as camadas de entrada para cada tipo de nó (transações ou demais entidades), de forma que as transações utilizam uma camada linear e os outros tipos, camada de embedding. Além disso, é aqui que um dos métodos da biblioteca *Pytorch Geometric* para processamentos de *Neural Networks* será utilizado.

Tabela 4 – Hiperparâmetros testados para melhor valor em GNNs

Hiperparâmetro	**Valores testados**	**Melhor(es) valor(es)**
Hidden channels	32, 64, 128	64, 128
Layers	2, 3, 4	2, 3
Dropout rate	0,3, 0,4, 0,5	0,3
Neighbors	[15, 10], [20, 15], [25, 10]	[15, 10]
Heads (GAT)	2, 4, 6	2, 4

- forward, que cuida da aplicação das transformações, normalizações e ativações a serem realizadas em cada nó (tanto lineares quanto embeddings). Utiliza-se a função $ReLU$ para ativação.

3.2.2 Treinamento, validação e teste

Devido ao tamanho do grafo, a metodologia de treino com *mini batches* definidos no método *NeighborLoader* foi adotada para melhorar a eficiência computacional e temporal. Uma parte importante do trabalho experimental foi a otimização de hiperparâmetros para a métrica AUC-PR. Foram testados valores de camadas ocultas, números de camadas do modelo GNN, a taxa de *dropout* para regularização e o número de vizinhos amostrados no NeighborLoader por 10 épocas para visualizar os melhores resultados. Aqui, o Data Science Agent do Google Colab, baseado em Gemini, foi de grande auxílio nos experimentos de código. A Tabela 4 mostra os valores testados e escolhidos para os experimentos posteriores.

Para todas as GNNs, o otimizador *Adam* foi utilizado. Seus parâmetros assumiram diferentes valores durante os experimentos para chegar à melhor configuração. Além disso, a função de perda é calculada usando *Cross Entropy Loss*, para que a ponderação penalize os erros cometidos em classes minoritárias. Durante o treinamento, AUC-PR e AUC-ROC foram as métricas escolhidas como mais relevantes, para traduzir a eficácia do modelo em distinguir as fraudes e também o melhor desempenho diante do desbalanceamento.

Embora a métrica AUC–ROC seja amplamente utilizada, sua interpretação em cenários como este requer cautela. Nessa pesquisa, foi percebido que seu aumento geralmente implica em melhores resultados conjuntos. Assim, utilizaram-se 100 épocas e *coeficiente de paciência* (número de épocas para finalizar o treinamento caso a métrica escolhida não seja aprimorada) definido em 10 para chegar aos melhores resultados.

Cada uma das linhas de treinamento foi executada 5 vezes, sendo apresentados os valores de média e desvio padrão como resultados. As métricas avaliadas ao final são F1-Score, Precisão, AUC-ROC, Recall, AUC-PR e a matriz de confusão para visualizar a disposição das predições de acordo com os rótulos verdadeiros.

Tabela 5 – Hiperparâmetros testados para melhor valor com XGBoost

Hiperparâmetro	**Valores testados**	**Melhor(es) valor(es)**
learning_rate	0,01, 0,05, 0,1	0,05
n_estimators	500, 1000, 1500	500
max_depth	7, 9, 11	9
subsample	0.7, 0.8	0,7
colsample_bytree	0.7, 0.8	0,7
gamma	0, 0.1, 0.2	0,2

3.3 *Dataset* aplicado ao XGBoost

Para o modelo XGBoost, os melhores valores de hiperparâmetros foram obtidos por meio do método *RandomizedSearchCV*, inserido na biblioteca *sklearn.model_selection*. É importante ressaltar que essa busca foi feita procurando maximizar o valor de *AUC-PR*. Com ele, foram testadas 10 iterações com combinações diferentes dos parâmetros setados. A Tabela 5 mostra os valores pré-definidos e os resultados obtidos.

A partir de tais parâmetros, os experimentos também foram realizados cinco vezes para apresentação dos valores de média e desvio padrão em relação a eles como resultado. As mesmas *features* e *tags* foram utilizadas para criação do grafo e para o modelo *XGBoost*. Embora muitas outras manipulações temporais e identitárias pudessem tornar a classificação mais eficiente, a intenção dos testes é captar como os dois tipos de modelos se comportam com uma quantidade menor de engenharia de *features*.

3.4 Comparação estatística

Para que as métricas de avaliação possam ser comparadas, foram adotados alguns critérios de relevância. Em primeiro lugar, para definir o melhor experimento entre todos os hiperparâmetros testados em SAGEConv, GATConv e XGBoost, analisa-se a diferença estatística entre os valores de AUC-PR utilizando o teste de Friedman e, caso não sejam estatisticamente iguais, aplica-se o teste de Nemenyi.

Tais métodos foram implementados utilizando a biblioteca *SciPy*. É interessante ressaltar que foram utilizados os 5 valores obtidos para cada experimento. Se os resultados não forem significantemente diferentes, adotam-se três critérios para escolher o melhor experimento, respectivamente: os melhores resultados de AUC-PR e revocação, considerando menor tempo computacional para processamento.

Em segundo lugar, comparam-se os melhores resultados de cada modelo entre si, utilizando a mesma metodologia estatística. Ao final, analisam-se também AUC-PR e revocação, a fim de trazer maior ênfase na capacidade proporcional de detectar fraudes (classe minoritária) e evitar falsos negativos, visto que trazem maior prejuízo neste caso.

4 AVALIAÇÃO EXPERIMENTAL

Neste capítulo, evidenciam-se os experimentos realizados com base na metodologia discutida anteriormente, bem como suas observações e resultados.

4.1 Análise do dataset

Para tratar devidamente as *features* do *dataset* e trazer interpretações de bons candidatos a nós, o experimento começou com uma análise exploratória das colunas. Após o *merge* dos *datasets* de transações e identidades, obteve-se o total de 590540 linhas e 434 colunas. Os dados são altamente desbalanceados, como evidenciado pela Tabela 6 e a Figura 14.

Durante a exploração, alguns comportamentos chamaram a atenção. O primeiro deles foi em relação ao código do produto comprado (*ProductCD*), cujo real valor é mascarado. O produto de código 'C' tem quase 12% de sua totalidade em fraude, o dobro do segundo maior produto representativo em fraudes, 'S' (Figura 15). Além disso, o domínio de *e-mail* 'protomail.com' possui grande relação com transações fraudulentas, visto que em 40% dos *e-mails* de compradores com esse sufixo há indícios de fraudes

Tabela 6 – Exploração de transações legítimas e fraudulentas

Total de transações	590540
Quantidade de transações fraudulentas	20663
Quantidade de transações legítimas	569877
Porcentagem de transações fraudulentas	3,499

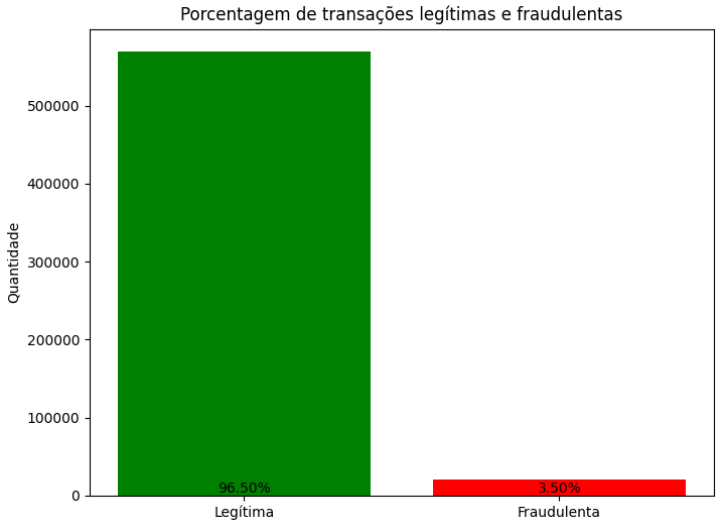

Figura 14 – Alto desbalanceamento entre a quantidade de transações legítimas e fraudulentas. Fonte: da autora.

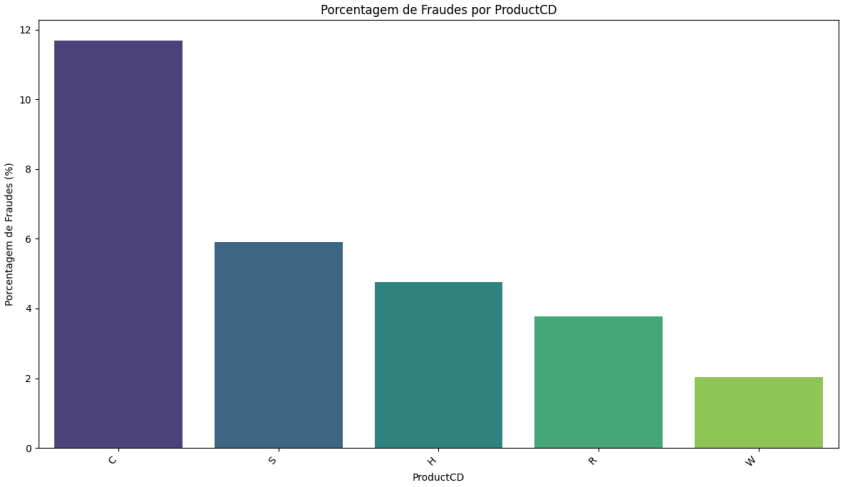

Figura 15 – Porcentagem de fraudes em relação aos tipos únicos da coluna ProductCD. Fonte: da autora.

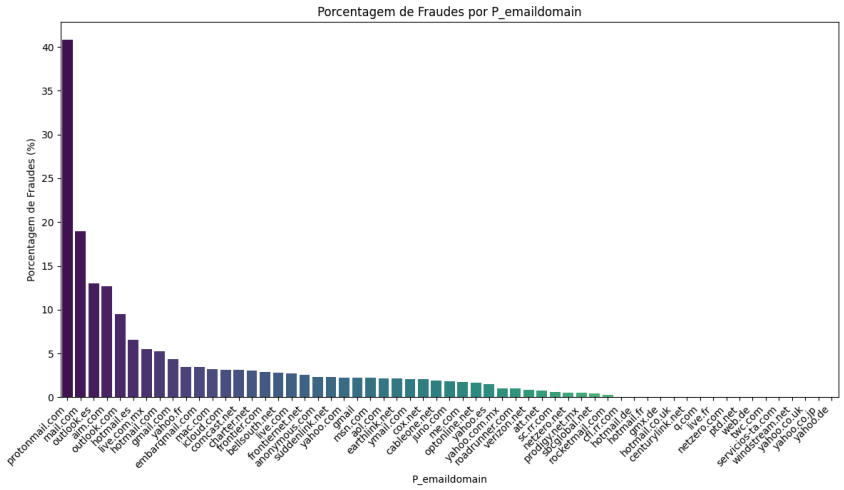

Figura 16 – Porcentagem de fraudes em relação aos domínios de *e-mail* do comprador. Fonte: da autora.

(Figura 16). Quando se trata do *e-mail* cadastrado para o recebedor da mercadoria, o valor é ainda mais chocante, atingindo quase 90% de sua utilização como um indicativo de fraude (Figura 17).

Por fim, quando se trata de cartões, a bandeira 'Discover' possui quase 8% do total de transações como sendo fraudes, o dobro da segunda maior bandeira de cartão representativa em fraudes, 'Visa', que está bem próxima da bandeira 'Mastercard' (Figura 18). Já no tipo de operação realizada pelo cartão, 'crédito' apresenta quase 7% de suas transações como fraude, mais do que o dobro em relação às operações de 'débito' (Figura 19). Esse confortamento reitera a problemática de cartões de crédito em operações *on-line* discutidas na Introdução deste trabalho.

Tais dados foram importantes para fixar a escolha de nós da metodologia. Afinal,

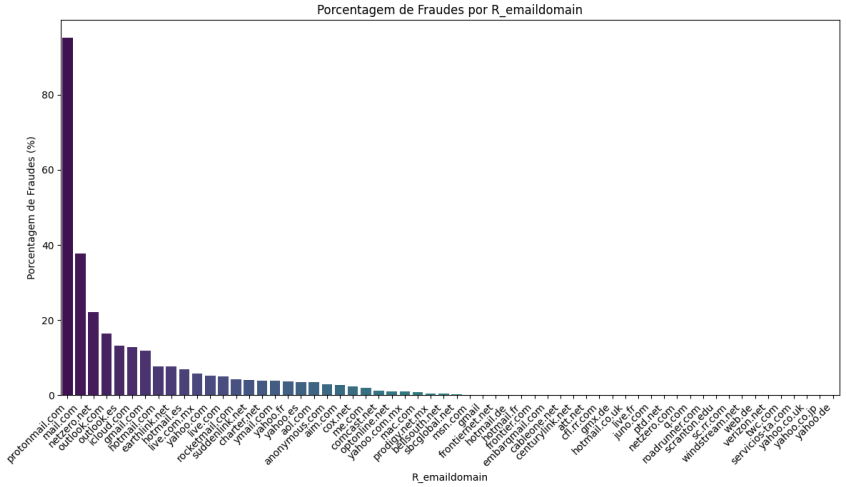

Figura 17 – Porcentagem de fraudes em relação aos domínios de *e-mail* do recebedor. Fonte: da autora.

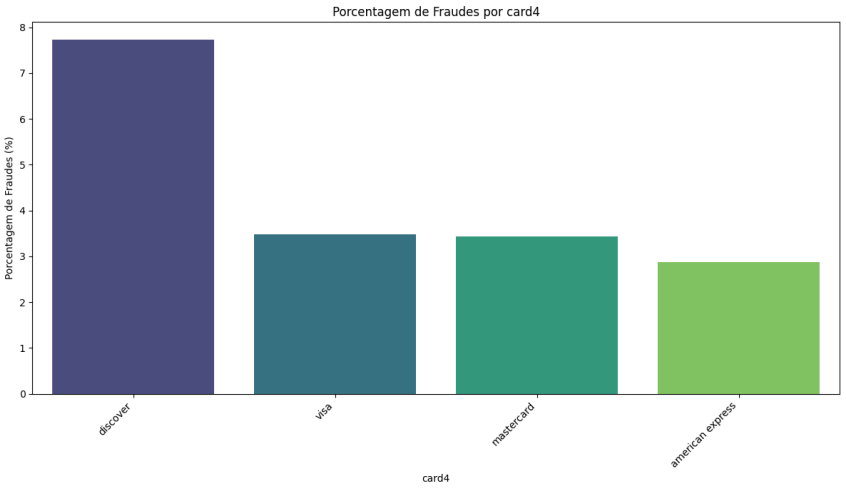

Figura 18 – Porcentagem de fraudes em relação à bandeira do cartão. Fonte: da autora.

as colunas que envolvem informações sobre a identidade envolvida na transação, embora não explicitamente, são peça fundamental para o entendimento de como as fraudes se propagam. As análises de campos como operação, tipo de e-mail e produto são as que saltam aos olhos em relação ao senso comum humano; o que se busca compreender com o modelo de redes neurais para grafos é a influência de outras categorias, como as de identidades numéricas (cujo significado não é tão claro para interpretação), nesse contexto malicioso de fraudes.

4.2 Estrutura do grafo

Com os nós retratados na Metodologia e as arestas conectando transações e demais entidades de identificação (de maneira não-direcionada, pelo método ToUndirected()), o grafo resultante possui 725199 nós e 18836226 arestas. A Tabela 7 demonstra a quantidade

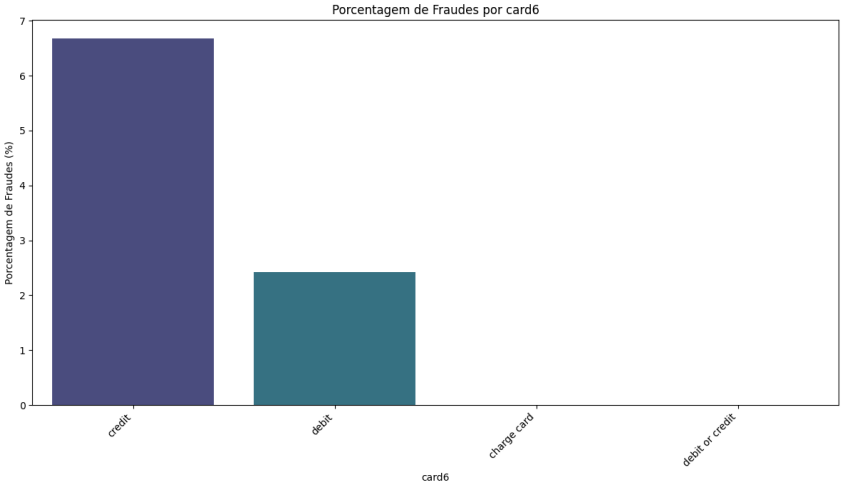

Figura 19 – Porcentagem de fraudes em relação ao tipo do cartão. Fonte: da autora.

de nós por cada entidade. Além disso, com a divisão dos nós em treino, validação e teste, a Tabela 8 ilustra a distribuição em formato de quantidade e porcentagem em relação aos nós de transação.

Pelas quantidades de nós e arestas, é possível perceber a grandeza e densidade do grafo criado. Para melhor visualização, a Figura 20 demonstra a configuração de uma amostragem com 5 nós cuja *label* $isFraud = 1$ (em vermelho) e 5 nós cuja *label* $isFraud = 1$ (em verde). Deles, partem as arestas que os ligam a determinados nós de identidade, como *ProductCD* e *card4* - que analisamos anteriormente.

Algo interessante nessa visualização é que alguns dos nós de identidade têm ligações apenas com fraudes ou transações legítimas; outros, como *card4*, têm ligações com os dois tipos de nós. Nesse momento, os reais atributos desses nós já estão em formato numérico e traduzidos em um *embedding*, mas supondo que esse fosse um nó que representa o cartão da bandeira '*Discovery*', é evidente que apenas isso não poderia classificar a legitimidade de uma transação. No entanto, sua ligação com outros nós podem ampliar a análise com uma rede neural apropriada.

4.3 Segmentação dos experimentos

Os métodos escolhidos para treinamento do modelo, **SAGEConv** e **GATConv**, tiveram a mesma modelagem de nós, arestas e *features* e os parâmetros associados ao treinamento foram manipulados com os valores descritos pelas Tabelas 9 e 10, que evidenciam também as métricas obtidas com a aplicação do melhor modelo em AUC-ROC para o conjunto de testes.

Para apresentação nas tabelas, foram calculados as médias e desvios padrão em relação aos 5 experimentos com cada configuração. É importante lembrar que os hiperparâmetros escolhidos foram testados anteriormente para buscar pela melhor performance

Tabela 7 – Número de nós por entidade

Entidade	**Número de nós**
transaction	590540
id_02	115655
card1	13553
DeviceInfo	1786
id_19	522
card2	500
id_20	394
id_11	365
addr1	332
id_33	260
id_31	130
card5	119
card3	114
id_17	104
id_06	101
id_05	93
id_01	77
id_30	75
addr2	74
id_10	62
R_emaildomain	60
P_emaildomain	59
id_13	54
id_09	46
id_14	25
id_03	24
id_18	18
id_04	15
ProductCD	5
card4	4
card6	4
id_32	4
id_34	4
id_37	2
id_15	3
DeviceType	2
id_36	2
id_28	2
id_16	2
id_29	2
id_12	2
id_38	2
id_35	2

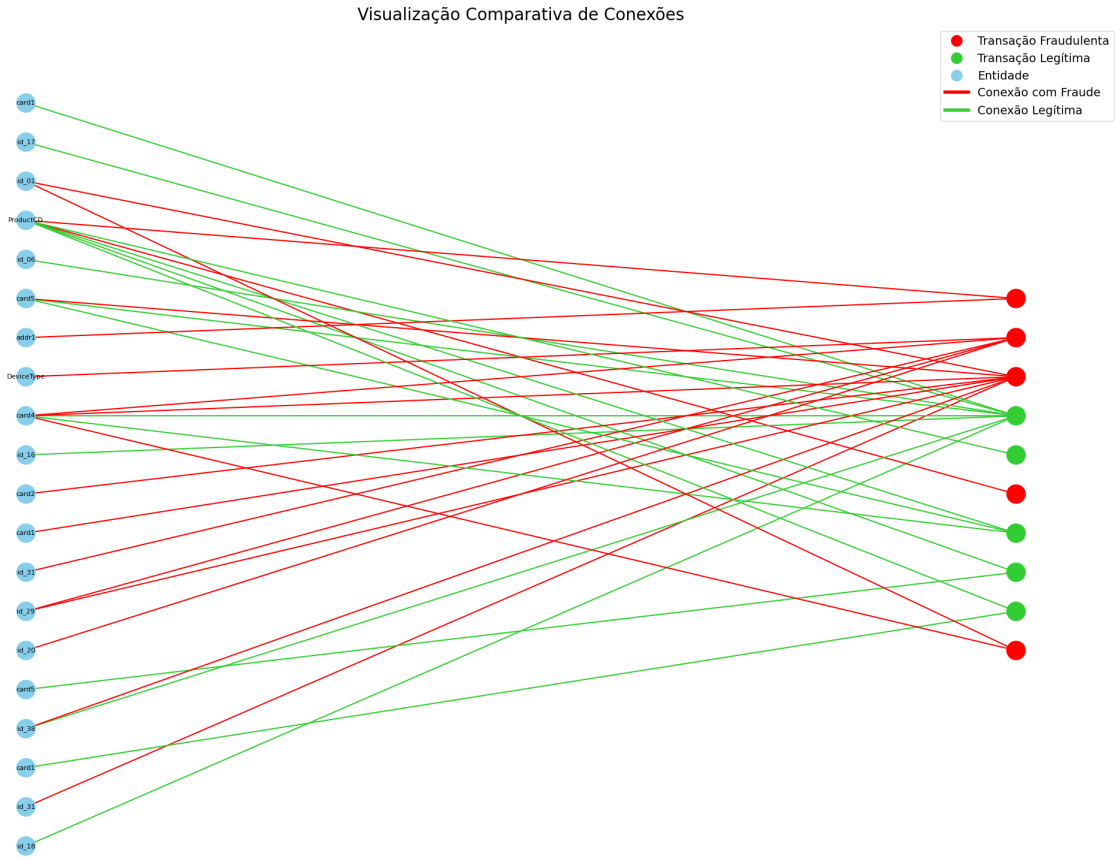

Figura 20 – Exemplo de amostras de transações legítimas e fraudulentas, bem como suas conexões com nós de identidade. Fonte: da autora.

Tabela 8 – Distribuição das Máscaras de Nós

Tipo de Máscara	**Nós**	**Percentual**
Treino	354324	60.00%
Validação	118108	20.00%
Teste	118108	20.00%
Total de nós mascarados	**590540**	100%

Tabela 9 – Hiperparâmetros e métricas resultantes do treinamento com SAGEConv

Hidden Channels	Num Layers	Learning Rate	AUC-ROC	AUC-PR	F1-Score	Recall	Precisão
64	2	0.0001	0.9463 ± 0.0011	0.6958 ± 0.0037	0.4889 ± 0.0209	0.7943 ± 0.0065	0.3537 ± 0.0228
		0.001	0.9477 ± 0.0009	0.6784 ± 0.0033	0.3771 ± 0.0337	0.8468 ± 0.0209	0.2438 ± 0.0307
	3	0.0001	0.9484 ± 0.0011	0.6944 ± 0.0039	0.4656 ± 0.0216	0.8057 ± 0.0092	0.3274 ± 0.0221
		0.001	0.9450 ± 0.0008	0.6652 ± 0.0045	0.3799 ± 0.0333	0.8343 ± 0.0170	0.2470 ± 0.0294
128	2	0.0001	0.9523 ± 0.0014	0.7346 ± 0.0066	0.5321 ± 0.0018	0.7945 ± 0.0026	0.4000 ± 0.0027
		0.001	0.9486 ± 0.0002	0.6790 ± 0.0029	0.4201 ± 0.0130	0.8220 ± 0.0098	0.2823 ± 0.0129
	3	0.0001	0.9507 ± 0.0015	0.7255 ± 0.0168	0.4971 ± 0.0534	0.8023 ± 0.0156	0.3601 ± 0.0615
		0.001	0.9463 ± 0.0021	0.6739 ± 0.0036	0.3889 ± 0.0557	0.8309 ± 0.0311	0.2555 ± 0.0506

Tabela 10 – Hiperparâmetros e métricas resultantes do treinamento com GATConv

Hidden Channels	Num Layers	Heads	AUC-ROC	AUC-PR	F1-Score	Recall	Precisão
64	2	2	0.9295 ± 0.0008	0.6110 ± 0.0001	0.3319 ± 0.0100	0.8096 ± 0.0069	0.2088 ± 0.0084
		4	0.9250 ± 0.0028	0.5835 ± 0.0138	0.3245 ± 0.0335	0.8077 ± 0.0158	0.2036 ± 0.0272
	3	2	0.9286 ± 0.0010	0.5889 ± 0.0229	0.3480 ± 0.0148	0.7993 ± 0.0148	0.2226 ± 0.0131
		4	0.9280 ± 0.0047	0.5999 ± 0.0197	0.3455 ± 0.0595	0.8013 ± 0.0265	0.2219 ± 0.0505
128	2	2	0.9303 ± 0.0004	0.6149 ± 0.0070	0.3566 ± 0.0317	0.8020 ± 0.0224	0.2299 ± 0.0281
		4	0.9246 ± 0.0022	0.5922 ± 0.0261	0.3334 ± 0.0460	0.8005 ± 0.0259	0.2116 ± 0.0385
	3	2	0.9298 ± 0.0024	0.5953 ± 0.0066	0.3540 ± 0.0055	0.8023 ± 0.0010	0.2271 ± 0.0044
		4	0.9212 ± 0.0009	0.5843 ± 0.0148	0.3596 ± 0.0114	0.7823 ± 0.0052	0.2335 ± 0.0101

de AUC-PR.

Para o experimento com GATConv, foram adotadas as mesmas métricas; no entanto, ao invés da variação de *Learning Rate*, esse parâmetro foi fixado em 0.01 (pela performance e velocidade de treinamento) e as cabeças (*heads*) de treinamento foram variadas entre os valores de 2 e 4.

Devido ao alto tempo empregado no treinamento dos modelos GATConv e SAGEConv (sendo o maior tempo registrado em 4 horas e 32min), foi testada a performance da adoção de uma GPU NVIDIA L4 x 1. Um experimento com os mesmos hiperparâmetros (GATConv, 128 canais ocultos, 3 camadas e 4 cabeças de atenção) alcançou 52 minutos. No entanto, é importante ressaltar que a inicialização é feita ao acaso e também pode interferir no tempo do treinamento.

4.4 Resultados

Uma das primeiras observações ao analisar os resultados é que a diminuição do parâmetro *learning rate*, além de tornar o treinamento do modelo mais rápido, leva a resultados de revocação mais altos e precisão mais baixa. Isso porque, ao tentar aprender mais rápido, o modelo pode priorizar a classificação majoritária (não-fraude) e levar a um valor maior de AUC-ROC (pelo acerto da maioria), mas aumentando o número de falsos positivos.

Destaca-se também o desempenho superior do modelo SAGEConv na maioria das métricas em relação ao modelo GATConv. Com o alto desbalanceamento dos dados, o aumento de cabeças de atenção pode tornar a propagação ruidosa e exigir maior quantidade de canais ocultos e recursos computacionais para chegar a uma performance sem diferenças

Tabela 11 – Melhores resultados para cada modelo avaliado

	SAGEConv	**GATConv**	**XGBoost**
AUC-ROC	0.9523 ± 0.0014	0.9303 ± 0.0004	0.9356 ± 0.0003
AUC-PR	0.7346 ± 0.0066	0.6149 ± 0.0070	0.6779 ± 0.0014
Recall	0.7945 ± 0.0026	0.8020 ± 0.0224	0.7480 ± 0.0021
Precisão	0.4000 ± 0.0027	0.2299 ± 0.0281	0.3670 ± 0.0031
F1-Score	0.5321 ± 0.0018	0.3566 ± 0.0317	0.4925 ± 0.0029

estatisticamente significantes.

A fim de escolher os melhores resultados para cada modelo, o teste de Friedman sinalizou o modelo SAGEConv com diferença significativa da métrica AUC-PR entre os experimentos 5 (128 canais ocultos, 2 camadas e taxa de aprendizado = 0,0001) e 7 (128 canais ocultos, 3 camadas e taxa de aprendizado = 0,0001) quando comparados ao experimento 4 (64 canais ocultos, 3 camadas e taxa de aprendizado = 0,001). Assim, o experimento 5 foi escolhido pela eficiência temporal e pelo resultado AUC-PR ligeiramente superior.

Já para o modelo GATConv, as mudanças de hiperparâmetros sem variação da taxa de aprendizado não apresentaram resultados estatisticamente diferentes. A escolha do melhor modelo foi baseada no maior valor médio de AUC-PR, seguido pela eficiência temporal e revocação. O modelo XGBoost foi executado 5 vezes com os hiperparâmetros evidenciados na Metodologia e, portanto, os valores médios foram adotados para avaliação na Tabela 11.

Usando o teste de Friedman e, posteriormente, o teste de Nemenyi, conclui-se que os valores de AUC-PR entre SAGEConv e GATConv possuem diferença significativa em diferentes valores críticos, como demonstrado na Figura 21. É possível visualizar, também, que não há diferença estatísticamente significante entre o modelo SAGEConv e XGBoost. No entanto, ao comparar puramente os valores médios obtidos em todas as métricas, nota-se uma superioridade geral do modelo SAGEConv em relação aos outros dois métodos.

Há hipóteses relevantes que podem embasar essa diferença entre SAGEConv e GATConv, mesmo porque este último método possui a complexidade intrínseca das cabeças de atenção. Uma observação que pode ser útil nesse caso é que, embora o modelo de GNNs melhore com o aumento de camadas ocultas, isso não é verdade quando são adicionadas mais camadas (veja Tabelas 9 e 10). Esse é um indicativo de que o grafo altamente desbalanceado se beneficia do aumento de capacidade do aprendizado, mas não necessariamente de um maior campo de visão da vizinhança.

Se observada a Figura 18, é possível entender que muitos nós legítimos e fraudulentos compartilham vizinhos. Aumentar a ênfase nessa análise não se mostrou útil, neste caso.

Figura 21 – Comparação de diferenças estatisticamente significantes entre os valores de AUC-PR dos modelos SAGEConv, GATConv e XGBoost. Fonte: da autora.

De maneira semelhante, um modelo mais complexo como o GATConv pode valer-se de cabeças de atenção e diluir os sinais mais importantes de fraude enquanto se apega a padrões sutis detectados, mas que não levam a melhores resultados na classificação.

Além disso, o modelo GATConv pode enfrentar um comportamento de *overfitting* devido à sua maior complexidade, apegando-se aos poucos dados de fraude e internalizando seu comportamento, o que diminui sua capacidade de generalizar e detectar transações ilegítimas que tenham características ligeiramente distintas do padrão antes aprendido.

Por último, destaca-se o tempo computacional empregado no treinamento dos modelos baseados em GNNs. O trabalho foi idealizado para requerir o mínimo de sofisticação e recursos pagos, valendo-se de CPUs do Google Colab na maioria dos treinamentos. No entanto, alguns experimentos alcançaram o tempo de 4 horas para finalização. O modelo *baseline* XGBoost destaca-se nesse aspecto, embora não possua o mesmo caráter visual para demonstração e explicabilidade dos critérios adotados pelo modelo para reconhecer uma fraude.

5 CONCLUSÕES

Diante da exploração dos dados, testes de código e resultados obtidos pelos treinamentos, torna-se possível avaliar o desempenho de redes neurais para grafos diante de um *dataset*, antes tabular, transformado em nós e arestas de conexões entre entidades. Apesar do teor altamente desbalanceado do conjunto de dados, com uma modelagem muito básica das *features* (transformação de variáveis categóricas e numéricas em *embeddings*) e análise de informações de identidade, foi possível encontrar um modelo que atende ao proposto: a detecção de transações fraudulentas, priorizando os falsos negativos. Dado que o valor *baseline* de AUC-PR para o *dataset* de transações seria de 0.035%, alcançar valores superiores a 50% é significativamente bom.

Os dados transformados em nós foram aqueles diretamente ligados à identidade da transação, como informações de cartão, dispositivo, endereço e domínios de *e-mail* registrados para compra e recebimento. Os demais dados, principalmente os que têm o real significado anônimo e relevância alta no contexto da transação em si, foram transformados em *features* para enriquecer os nós a serem classificados. Apesar de apresentar-se em um modelo tabular, os testes e experimentos com configurações para grafos demonstram que existem diversas maneiras de apresentar um conjunto de dados em que há relacionamentos claros e torná-los relevantes para análise.

Em relação aos diferentes modelos de GNNs, utilizando a mesma modelagem de grafo, os resultados demonstram que a acurária precisão-revocação (AUC-PR) do modelo GraphSAGE com maiores valores de canais ocultos é superior em relação ao modelo GATConv, enquanto que a diferença em relação ao modelo XGBoost não é estatisticamente relevante, mesmo que maior numericamente. O modelo GAT teve boa performance, mas sua característica de atenção em um caso de uso tão desbalanceado pode assumir o comportamento de propagação de ruídos.

No entanto, é importante ressaltar que o tempo empregado no treinamento do melhor modelo GraphSAGE e GAT com maior quantidade de canais pode chegar a ser 2 vezes maior. Levando em consideração que o trabalho emprega diversas técnicas para diminuir a complexidade e exigência computacional, esse é um fator importante a se considerar. Caso haja disponibilidade de GPU, o processamento poderá ser altamente otimizado em tempo.

Já o modelo XGBoost, assim como na maioria das publicações acadêmicas consultadas, performou de maneira eficiente em relação a aumentar a precisão com menos prejuízo na revocação e, por sua vez, o AUC-PR. O tempo empregado no treinamento também é um ponto forte, já que foi muito eficiente nesse aspecto. Ao explorar diferentes soluções

dadas à competição no *Kaggle*, vê-se que uma engenharia de *features* mais aprofundada em relação ao tempo das transações e a tentativa de individualizar as identidades por trás das transações foram as abordagens que contribuíram para melhores resultados. Para atender ao objetivo específico do trabalho, a mesma engenharia de *features* foi empregada para os dois tipos de arquitetura (GNNs e XGBoost).

Por fim, para abordagens futuras, idealiza-se o modelo GNN com hospedagem em uma infraestrutura *Cloud*. Com base nessa solução, tanto o *dataset* original quanto novas transações com dados tabulares semelhantes poderão ser classificados de maneira dinâmica e orientada a eventos (disparadas com o acionamento de um gatilho específico, como a entrada de uma nova fonte de dados na solução de armazenamento). Além disso, outro tópico relevante para diminuir a fricção em casos de falsos positivos (em que o cliente é impedido de concluir uma transação legítima) seria a implementação de métodos visuais facilmente interpretáveis para trazer transparência ao usuário final sobre a classificação do modelo.

REFERÊNCIAS

ABCOMM. **Previsão de vendas no e-Commerce para os Próximos 5 anos**. 2024. https://dados.abcomm.org/previsao-de-vendas-online.

ASSUMPÇAO, H. S. *et al.* Delator: Detecção automática de indícios de lavagem de dinheiro por redes neurais em grafos de transacões. *In*: SBC. **Brazilian Workshop on Social Network Analysis and Mining (BRASNAM)**. [*S.l.: s.n.*], 2022. p. 13–24.

BIANCAMANO, P. **Dia dos Pais: 3 a cada 5 pessoas têm medo de realizar compras online**. 2021. https://www.psafe.com/blog/dia-dos-pais-3-a-cada-5-pessoas-tem-medo-de-realizar-compras-online/.

CECCON, D. **Funções de ativação: definição, características, e quando usar cada uma**. [*S.l.: s.n.*]: IA Expert Academy, 2020. https://iaexpert.academy/2020/05/25/funcoes-de-ativacao-definicao-caracteristicas-e-quando-usar-cada-uma/.

CHALEGRA, J. **Tentativas de fraudes no e-commerce atingem R$ 3,5 bi em 2023**. 2024. https://consumidormoderno.com.br/fraude-comercio-eletronico/.

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.

CHENG, D. *et al.* Graph neural networks for financial fraud detection: a review. **Frontiers of Computer Science**, Springer Science and Business Media LLC, v. 19, n. 9, jan. 2025. ISSN 2095-2236. Disponível em: http://dx.doi.org/10.1007/s11704-024-40474-y.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, v. 7, n. Jan, p. 1–30, 2006.

GOMES, P. F. Uma introdução à ciência de redes e teoria de grafos. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 46, p. e20240190, 2024.

Google for Developers. **Classification: Accuracy, recall, precision, and related metrics**. [*S.l.: s.n.*]: Google for Developers, 2025. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall.

Google for Developers. **Classification: ROC and AUC**. [*S.l.: s.n.*]: Google, 2025. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

Google for Developers. **Neural networks: Nodes and hidden layers**. [*S.l.: s.n.*]: Google, 2025. https://developers.google.com/machine-learning/crash-course/neural-networks/nodes-hidden-layers.

Google for Developers. **Thresholds and the confusion matrix**. [*S.l.: s.n.*]: Google for Developers, 2025. https://developers.google.com/machine-learning/crash-course/classification/thresholding.

GOULART, V. M. A. Treinamento de redes neurais com incorporação da técnica backpropagation ao fdipa. 2022.

GUO, X. *et al.* Graph-based representation learning for identifying fraud in transaction networks. *In*: IEEE. **2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)**. [*S.l.: s.n.*], 2025. p. 1598–1602.

HAMILTON, W.; YING, Z.; LESKOVEC, J. Inductive representation learning on large graphs. **Advances in neural information processing systems**, v. 30, 2017.

IEEE Computational Intelligence Society. **IEEE-CIS Fraud Detection**. 2019. https://www.kaggle.com/c/ieee-fraud-detection/overview.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KIPF, T. Semi-supervised classification with graph convolutional networks. **arXiv preprint arXiv:1609.02907**, 2016.

MAPEL, E. F. Comparação entre modelos de classificação convencionais e baseados em gnn aplicados a dados tabulares de covid-19. Serra, 2023.

MARIANO, D. **Networkx - Analisando grafos com Python e a biblioteca Networkx**. 2020. https://diegomariano.com/networkx/.

MENCZER, F.; FORTUNATO, S.; DAVIS, C. A. **A first course in network science**. [*S.l.: s.n.*]: Cambridge University Press, 2020.

METZ, J. *et al.* Redes complexas: conceitos e aplicações. 2007.

MOTIE, S.; RAAHEMI, B. Financial fraud detection using graph neural networks: A systematic review. **Expert Systems with Applications**, Elsevier, v. 240, p. 122156, 2024.

NARDE, W. **Como utilizar machine learning no combate à fraude**. 2024. Disponível em: https://glassdata.io/blog/como-utilizar-machine-learning-no-combate-a-fraude/.

NIELSEN, M. A. **Neural networks and deep learning**. [*S.l.: s.n.*]: Determination press San Francisco, CA, USA, 2015. v. 25.

RAO, Y. *et al.* Knowledge-guided fraud detection using semi-supervised graph neural network. *In*: SPRINGER. **International Conference on Web Information Systems Engineering**. [*S.l.: s.n.*], 2021. p. 385–393.

RIMONATO, I. P. d. O. S.; SANTOS, J. P. dos. Pix solução tecnológica de inclusão financeira. **Research, Society and Development**, v. 10, n. 13, p. 1–9, 2021.

SERGADEEVA, A.; LAVROVA, D. S.; ZEGZHDA, D. P. Bank fraud detection with graph neural networks. **Automatic Control and Computer Sciences**, Springer, v. 56, n. 8, p. 865–873, 2022.

SHEN, A.; TONG, R.; DENG, Y. Application of classification models on credit card fraud detection. *In*: IEEE. **2007 International conference on service systems and service management**. [*S.l.: s.n.*], 2007. p. 1–4.

SILVA, L. de A.; FEITOSA, E. L. Avaliando modelos de graph neural networks para detecção de usuários fraudulentos em e-commerce. *In*: SBC. **Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)**. [*S.l.: s.n.*], 2021. p. 280–287.

TOZATO, J. M. **Análise de redes neurais baseadas em grafos**. 2021. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2021.

Ultralytics. **Função de ativação**. 2025. https://www.ultralytics.com/pt/glossary/activation-function.

VELIČKOVIĆ, P. *et al.* Graph attention networks. **arXiv preprint arXiv:1710.10903**, 2017.

VICENTE Ângelo. **O impulso do e-commerce no Brasil: perspectivas até 2027**. 2024. https://www.ecommercebrasil.com.br/artigos/o-impulso-do-e-commerce-no-brasil-perspectivas-ate-2027.

WALLIS, D. **Comparing classifiers (Friedman and Nemenyi tests)**. [*S.l.: s.n.*]: Medium, 2021. https://medium.com/@diogeneswallis/comparing-classifiers-friedman-and-nemenyi-tests-32294103ee12.

WANG, J. *et al.* A review on graph neural network methods in financial applications. **arXiv preprint arXiv:2111.15367**, 2021.

WANG, Z. **graph-fraud-detection**. [*S.l.: s.n.*]: GitHub, 2021. https://github.com/waittim/graph-fraud-detection/tree/main.

XIAO, S. *et al.* Graph neural networks in node classification: survey and evaluation. **Machine Vision and Applications**, Springer, v. 33, n. 1, p. 4, 2022.

APÊNDICE A – REPOSITÓRIO DE CÓDIGO NO GITHUB

Autora: Sabrina Vieira Guerra

Ano: 2025

Título: graph-fraud-detection

Descrição: Código fonte utilizado para os experimentos descritos na Metodologia.

***Link*:** https://github.com/sahvieirag/graph-fraud-detection