UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

LUÍSA CALEGARI DE BARROS CIZILIO

DETECÇÃO DE CONTAS BANCÁRIAS FRAUDULENTAS COM REDES NEURAIS DE GRAFO

Ouro Preto, MG
2025

LUÍSA CALEGARI DE BARROS CIZILIO

DETECÇÃO DE CONTAS BANCÁRIAS FRAUDULENTAS COM REDES NEURAIS DE GRAFO

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Ouro Preto, MG
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Luísa Calegari de Barros Cizilio

Detecção de contas bancárias fraudulentas com Redes Neurais de Grafo

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 29 de Agosto de 2025.

Membros da banca

Eduardo José da Silva Luz (Orientador) - Doutor - Universidade Federal de Ouro Preto
Andressa Oliveira Souza (Examinadora) - Mestre - Programa de Pós-Graduação em Ciência da Computação - UFOP
Vander Luis de Souza Freitas (Examinador) - Doutor - Universidade Federal de Ouro Preto

Eduardo José da Silva Luz, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 29/08/2025.

Documento assinado eletronicamente por **Eduardo Jose da Silva Luz**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 02/09/2025, às 12:12, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0965549** e o código CRC **79A5011B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Agradecimentos

A Deus, pela oportunidade da vida, pela força e pela luz em todos os momentos dessa caminhada.

Aos meus pais Aniely e Gilmar, meus irmãos Guilherme, Lívia e Felipe, e toda a minha família, pelo amor incondicional, pelo apoio constante e por sempre me proporcionarem as melhores oportunidades.

Ao Thiago, que esteve ao meu lado durante toda a graduação, me oferecendo apoio, paciência e incentivo nos momentos mais desafiadores.

Ao meu orientador, professor Eduardo, pela dedicação, pelos valiosos conselhos, pelas ideias que enriqueceram este trabalho e por toda paciência e compreensão ao longo deste percurso.

À Andressa, pelo auxílio fundamental em algumas etapas deste projeto.

Ao CSILab, pelo auxílio, espaço e máquinas que possibilitaram a realização desta pesquisa.

À Universidade Federal de Ouro Preto (UFOP), por toda a estrutura e conhecimento oferecidos, que foram essenciais para minha formação acadêmica e pessoal.

Resumo

A detecção de fraudes em contas bancárias representa um desafio crescente, demandando abordagens que transcendam a análise de transações isoladas. Modelos tradicionais, embora eficazes até certo ponto, falham em capturar a complexa rede de interações entre contas, onde padrões fraudulentos sofisticados se manifestam. Este trabalho propõe a utilização de Redes Neurais de Grafos (GNNs) para aprimorar a identificação de contas fraudulentas, modelando a estrutura relacional e a dinâmica das transações financeiras. O objetivo principal é investigar a eficácia das GNNs em comparação com métodos de classificação convencionais, como árvores de decisão e Redes Neurais Convolucionais (CNNs), que não consideram a topologia das interações. Para isso, uma base de dados tabular foi convertida em uma representação de grafo, permitindo que o modelo explore as conexões entre as entidades. Os resultados indicam que a estrutura do grafo e os atributos influenciam o desempenho, com o modelo GraphSAGE se destacando. Embora os modelos de grafos ainda não tenham superado os métodos tabulares tradicionais, eles se mostraram promissores, principalmente quando o problema foi focado em uma classificação binária. A incorporação da estrutura relacional dos dados resultou em um desempenho que, com refinamentos futuros na modelagem e nos hiperparâmetros, tem o potencial de superar as abordagens convencionais, contribuindo para o desenvolvimento de sistemas de segurança financeira mais robustos e proativos.

Palavras-chave: Detecção de Fraude. Contas Bancárias. Redes Neurais de Grafos.

Abstract

Fraud detection in bank accounts represents a growing challenge, demanding approaches that transcend the analysis of isolated transactions. Traditional models, while effective to a certain extent, fail to capture the complex network of interactions between accounts, where sophisticated fraudulent patterns emerge. This work proposes the use of Graph Neural Networks (GNNs) to enhance the identification of fraudulent accounts by modeling the relational structure and dynamics of financial transactions. The main objective is to investigate the effectiveness of GNNs compared to conventional classification methods, such as decision trees and Convolutional Neural Networks (CNNs), which do not consider the topology of interactions. To achieve this, a tabular database was converted into a graph representation, allowing the model to explore the connections between entities. The results indicate that the graph structure and attributes influence performance, with the GraphSAGE model showing notable results. Although the graph-based models have not yet surpassed traditional tabular methods, they have shown promise, especially when the problem was focused on a binary classification. Incorporating the relational structure of the data has resulted in performance that, with future refinements in modeling and hyperparameters, has the potential to outperform conventional approaches, contributing to the development of more robust and proactive financial security systems.

Keywords: Fraud Detection. Bank Accounts. Graph Neural Networks.

Sumário

1 Introdução

A detecção de fraudes em contas bancárias é um desafio crescente, impulsionado pelo avanço das táticas fraudulentas e pela complexidade das relações financeiras. Diferentemente da detecção de anomalias em transações isoladas, identificar contas fraudulentas exige a análise da estrutura e da evolução das interações entre usuários (SOUZA, 2023). Este estudo propõe a utilização de Redes Neurais de Grafos (GNNs) (MOTIE; RAAHEMI, 2024) para capturar padrões ocultos em redes financeiras, explorando as conexões entre contas para aprimorar a identificação de fraudes.

Estudos anteriores demonstram a eficácia de abordagens tradicionais na classificação de contas bancárias fraudulentas. O estudo de Souza (2023) investigou a utilização de uma taxonomia hierárquica para classificação de contas fraudulentas, avaliando abordagens de classificação plana, hierárquica local e global, além da aplicação de modelos tabulares e baseados em árvores de decisão. Os resultados indicaram que métodos como SMOTE e Borderline SMOTE melhoram a sensibilidade dos classificadores e que modelos hierárquicos apresentaram melhor desempenho em termos de precisão e F-score, porém sem capturar a evolução temporal das fraudes.

A revisão de Motie e Raahemi (2024) apontou o crescente uso de Redes Neurais de Grafos (GNNs) na detecção de fraudes financeiras, mas ressaltou a escassez de estudos que exploram grafos heterogêneos e dinâmicos. Além disso, observou-se uma lacuna na aplicação de técnicas de aprendizado não supervisionado e semi-supervisionado, que podem ser fundamentais para detecção de padrões fraudulentos sem a necessidade de grandes volumes de dados rotulados.

Neste estudo, será utilizada a base de dados do estudo de Souza (2023) para avaliar o desempenho de GNNs na detecção de contas fraudulentas, explorando a estrutura das interações financeiras e incorporando conceitos de grafos para montar uma rede direcionada com os principais dados de contas bancárias e transações, conforme sugerido por Motie e Raahemi (2024). A proposta busca superar as limitações de abordagens tabulares e baseadas em árvores, permitindo uma análise mais profunda da estrutura temporal das fraudes bancárias e contribuindo para avanços na detecção proativa de atividades fraudulentas.

1.1 Justificativa

A detecção de fraudes em contas bancárias é um desafio contínuo que demanda soluções cada vez mais sofisticadas. Modelos tradicionais, como os analisados por Souza (2023), têm apresentado bons resultados na identificação de contas fraudulentas, mas operam principalmente com abordagens tabulares e baseadas em árvores de decisão, sem considerar a complexidade das interações financeiras entre contas. A crescente sofisticação dos esquemas fraudulentos exige

métodos capazes de capturar as complexas relações entre as contas envolvidas.

A revisão de Motie e Raahemi (2024) destaca que Redes Neurais de Grafos (GNNs) vêm ganhando espaço na detecção de fraudes financeiras, devido à sua capacidade de modelar relações complexas entre entidades. Essa abordagem permite a detecção de padrões emergentes e anomalias nas conexões entre contas, o que pode levar a uma identificação mais precisa de fraudes, especialmente em cenários com múltiplas interações.

Este estudo visa explorar o uso de redes neurais de grafos (GNNs) para a detecção de fraudes financeiras, preenchendo as lacunas identificadas na literatura. O impacto esperado é a melhoria da precisão dos modelos, contribuindo para a segurança e confiabilidade do sistema financeiro.

1.2 Objetivos

Este trabalho visa investigar a eficácia de Redes Neurais de Grafos (GNN) na detecção de fraudes bancárias. A performance dos modelos GNN, que incorporam a estrutura relacional dos dados, será comparada com a de abordagens que não consideram tais interações, como algoritmos baseados em árvores de decisão e Redes Neurais Convolucionais (CNN).

1.3 Objetivos específicos

- Propor um modelo baseado em Redes Neurais de Grafos (GNNs) para a classificação de contas bancárias suspeitas de fraude a partir de suas interações financeiras.
- Avaliar a eficácia da modelagem de transações financeiras como uma estrutura de grafo para a identificação de padrões fraudulentos complexos.
- Investigar o impacto da conectividade e das características das transações (a topologia do grafo) na capacidade do modelo em identificar contas fraudulentas.
- Quantificar o desempenho preditivo do modelo de GNNs por meio de métricas consolidadas para problemas de classificação desbalanceada.
- Comparar a performance do modelo de GNNs com abordagens de aprendizado de máquina tradicionais aplicadas em dados tabulares.

1.4 Questões de pesquisa

1. As Redes Neurais de Grafos (GNN) apresentam desempenho superior na detecção de fraudes em contas bancárias em comparação com modelos tradicionais baseados em dados tabulares?

2. Em que medida a modelagem das interações entre contas contribui para a melhoria da capacidade de detecção de fraudes?
3. A inclusão das interações entre contas pode reduzir falsos positivos ou falsos negativos em relação aos modelos que tratam as contas de forma isolada?

2 Fundamentação Teórica

2.1 Fraudes Financeiras

Fraude é definida como qualquer ato ardiloso, enganoso ou de má-fé, com o objetivo de prejudicar outra pessoa ou evitar o cumprimento de uma obrigação. No contexto financeiro, a fraude caracteriza-se como um comportamento ilícito ou ilegal que resulta em benefício econômico para indivíduos ou organizações. Exemplos incluem manipulação de transações bancárias, uso indevido de cartões de crédito, lavagem de dinheiro, fraudes em seguros e outros (AL-HASHEDI; MAGALINGAM, 2021). O art. 14 da Lei 8.078 de 1990, no Brasil, garante que as instituições financeiras devem se responsabilizar pela reparação dos danos sofridos pelo consumidor, independente da existencia de culpa.

Esse tipo de crime afeta não apenas o setor financeiro, mas também governos, empresas e consumidores. A evolução tecnológica, como a popularização de serviços eletrônicos e dispositivos móveis, contribuiu significativamente para a complexidade e a escala desses atos fraudulentos, tornando-os mais difíceis de identificar e combater. A detecção de fraudes financeiras, portanto, tornou-se um tema de grande relevância, com implicações que vão desde perdas econômicas até impactos na confiança pública no sistema financeiro (BOLTON; HAND, 2002).

2.2 Métricas de Avaliação

A avaliação de um classificador vai além da simples medição de acertos. Em muitos domínios, o custo associado a diferentes tipos de erro é assimétrico e o desbalanceamento de classes é frequente. A matriz de confusão é a ferramenta primária para decompor o desempenho do modelo em quatro desfechos: Verdadeiros Positivos (TP), instâncias positivas corretamente identificadas; Verdadeiros Negativos (TN), instâncias negativas corretamente classificadas; Falsos Positivos (FP), instâncias negativas incorretamente classificadas como positivas; e Falsos Negativos (FN), instâncias positivas incorretamente classificadas como negativas.

A partir desta matriz, derivam-se métricas mais informativas. A Precisão ($TP/(TP + FP)$) mede a proporção de predições positivas que estavam corretas, sendo importante para reduzir erros de classificação em instâncias negativas. A Revocação ou Sensibilidade ($TP/(TP + FN)$) mede a proporção de todas as instâncias positivas que o modelo conseguiu identificar, sendo essencial para avaliar a capacidade de detecção. Existe um trade-off inerente entre estas duas métricas: aumentar a sensibilidade do modelo para detectar mais instâncias positivas geralmente leva a um aumento de falsos alarmes (diminuição da Precisão). O F1-Score, a média harmônica de ambas, oferece um ponto de equilíbrio. Para cenários com classes desbalanceadas, a análise

da Curva Precisão-Revocação (PR Curve) e a área sob ela (AUPRC) pode ser mais elucidativa do que a tradicional curva ROC, pois fornece uma visão mais clara sobre o desempenho em relação às classes minoritárias (GRANDINI; BAGLI; VISANI, 2020).

2.3 Random Forests

O método *Random Forests*, proposto por Breiman (2001), consiste em um conjunto (*ensemble*) de classificadores do tipo árvore de decisão, onde cada árvore é construída a partir de amostras aleatórias e independentes, garantindo diversidade entre os modelos individuais. A predição final é obtida por votação majoritária no caso de classificação ou pela média das saídas no caso de regressão. Formalmente, um *Random Forest* é definido como um conjunto de classificadores estruturados em árvores $\{h(x, \Theta_k), k = 1, 2, \ldots\}$, onde cada Θ_k é um vetor aleatório independente e identicamente distribuído (i.i.d.). Assim, para uma entrada x, cada árvore $h(x, \Theta_k)$ emite um voto unitário para a classe mais provável, sendo a classe final escolhida pelo voto majoritário:

$$H(x) = \text{mode}\{h(x, \Theta_k)\}_{k=1}^{K} \tag{2.1}$$

Para analisar a acurácia, Breiman (2001) introduz a função margem de um ensemble de classificadores $h_1(x), h_2(x), \ldots, h_K(x)$, definida como

$$mg(X, Y) = \mathbb{E}_k \, I(h_k(X) = Y) - \max_{j \neq Y} \mathbb{E}_k \, I(h_k(X) = j), \tag{2.2}$$

, onde $I(\cdot)$ é a função indicadora. A margem mede a diferença entre a proporção média de votos para a classe correta Y e a maior proporção de votos para qualquer outra classe. O erro de generalização é então dado por

$$PE^* = P_{X,Y}(mg(X, Y) < 0). \tag{2.3}$$

Segundo Breiman (2001), à medida que o número de árvores tende ao infinito, o erro de generalização converge quase certamente para um limite, o que explica por que *Random Forests* não sofrem de sobreajuste com o aumento de árvores. A acurácia do modelo também depende de dois fatores fundamentais: (i) a força dos classificadores individuais e (ii) a correlação entre eles. A força média é definida como

$$s = \mathbb{E}_{X,Y} \, mr(X, Y), \tag{2.4}$$

onde a margem do ensemble é

$$mr(X, Y) = P_\Theta(h(X, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(X, \Theta) = j). \tag{2.5}$$

Com base nessa formulação, Breiman (2001) derivou um limite superior para o erro de generalização em termos de s e da correlação média $\bar{\rho}$ entre classificadores:

$$PE^* \leq \frac{\bar{\rho}(1 - s^2)}{s^2}. \tag{2.6}$$

Essa relação mostra que, para reduzir o erro, é necessário aumentar a força dos classificadores individuais enquanto se reduz a correlação entre eles. Nesse contexto, a principal inovação do método está na introdução de aleatoriedade na escolha dos atributos. Em cada nó da árvore, em vez de considerar todas as variáveis, seleciona-se aleatoriamente um subconjunto de tamanho F e determina-se a melhor divisão apenas dentro desse subconjunto. Essa técnica reduz a correlação entre árvores e aumenta a robustez do modelo. Breiman (2001) mostrou que valores baixos de F (como $F = 1$ ou $F = \lfloor \log_2 M + 1 \rfloor$, sendo M o número total de atributos) já produzem resultados próximos ao ótimo, com custo computacional bastante reduzido.

Outra característica notável dos *Random Forests* é a possibilidade de obter estimativas internas de erro sem a necessidade de um conjunto de teste separado. Aproximadamente um terço das instâncias é deixado de fora em cada amostragem bootstrap, formando o conjunto denominado *out-of-bag* (OOB). O erro OOB é calculado utilizando essas instâncias não vistas, e estudos demonstram que ele constitui uma estimativa não tendenciosa do erro de generalização, comparável ao uso de uma base de validação independente. Além disso, o procedimento OOB permite calcular medidas adicionais, como a força s do classificador, a correlação média $\bar{\rho}$ e a importância das variáveis, obtida pela aleatorização dos valores de um atributo e pela observação do impacto no erro.

2.4 XGBoost (Extreme Gradient Boosting)

O XGBoost (Extreme Gradient Boosting), proposto por Chen e Guestrin (2016), é um algoritmo de aprendizado de máquina baseado em um modelo de conjunto de árvores (*tree ensemble model*). Nesse modelo, a previsão final para uma determinada amostra é a soma das previsões de cada árvore de decisão que compõe o conjunto. Para melhorar a precisão, o XGBoost incorpora técnicas como regularização de segundo grau, otimização da função de perda e estratégias de poda.

Considerando um conjunto de dados $\mathcal{D} = (x_i, y_i)$, com n amostras e m atributos, onde $|\mathcal{D}| = n$, $x_i \in \mathbb{R}^m$, e $y_i \in \mathbb{R}$, um modelo de conjunto de árvores utiliza K funções aditivas para prever a saída $\hat{y}$:

$$\hat{y} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}\,, \tag{2.7}$$

onde $\mathcal{F}$ é o espaço das árvores de regressão, também conhecidas como CART (*Classification and Regression Trees*). Cada função f_k corresponde a uma estrutura de árvore independente q e pesos de folha w.

Para aprender o conjunto de funções, o modelo minimiza um objetivo regularizado $\mathcal{L}(\phi)$:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)\,. \tag{2.8}$$

Aqui, l é uma função de perda convexa e diferenciável que mede a diferença entre a previsão $\hat{y}_i$ e o valor real y_i. O termo Ω penaliza a complexidade do modelo, ajudando a suavizar os pesos finais e a evitar o *overfitting*.

O modelo é treinado de forma aditiva, pois não pode ser otimizado por métodos tradicionais no espaço euclidiano. Denotando $\hat{y}_i^{(t)}$ como a previsão da i-ésima instância na t-ésima iteração, adiciona-se f_t para minimizar o seguinte objetivo:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)\,. \tag{2.9}$$

Utilizando uma aproximação de Taylor de segunda ordem, o objetivo pode ser simplificado para:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)\,, \tag{2.10}$$

onde g_i e h_i são as estatísticas do gradiente de primeira e segunda ordem na função de perda.

Para uma estrutura de árvore $q(x)$ fixa, o peso ótimo w_j^* para a folha j é calculado como:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}\,, \tag{2.11}$$

e o valor ótimo correspondente da função de perda é:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \left[\frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} \right] + \gamma T\,. \tag{2.12}$$

Essa equação serve como uma função de pontuação para avaliar a qualidade de uma estrutura de árvore q. Como é impossível enumerar todas as estruturas de árvore possíveis, um algoritmo guloso é utilizado, que começa com uma única folha e adiciona ramos iterativamente.

A redução da perda após uma divisão, $\mathcal{L}_{split}$, é usada para avaliar os candidatos à divisão:

$$\mathcal{L}_{split} = \tfrac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i \right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i \right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i \right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma\,, \tag{2.13}$$

onde I_L e I_R são os conjuntos de instâncias nos nós esquerdo e direito após a divisão. O algoritmo também emprega técnicas como *shrinkage* (retração) e subamostragem de colunas para prevenir ainda mais o *overfitting*.

2.5 Redes Neurais Convolucionais (CNNs)

As Redes Neurais Convolucionais (CNNs) surgiram inicialmente com a proposta de Fukushima (1980) e foram posteriormente aprimoradas por LeCun et al. (1989). Elas constituem

um tipo específico de rede neural projetado para lidar com dados organizados em forma de grade, como ocorre em séries temporais (estrutura unidimensional) ou em imagens (arranjo bidimensional de pixels) (GOODFELLOW et al., 2016).

Segundo Goodfellow et al. (2016), a denominação "rede neural convolucional" decorre do uso da operação de convolução, uma forma particular de operação linear. Dessa maneira, as CNNs diferenciam-se por substituir a multiplicação matricial tradicional pela convolução em pelo menos uma de suas camadas.

Conforme Goodfellow et al. (2016), a convolução é denotada pelo símbolo de asterisco ($*$) e consiste em uma operação entre duas funções com argumentos de valor real. A convolução das funções $x, w : \mathbb{R}^d \to \mathbb{R}$ é definida por $s(t)$:

$$s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(a)\, w(t-a)\, da\,. \tag{2.14}$$

Ou seja, a operação mede a sobreposição entre x e w quando uma das funções é "invertida" e deslocada por um fator t. Para objetos discretos, a integral é substituída por um somatório. De modo geral, a convolução é definida para quaisquer funções para as quais a integral acima seja válida, podendo ser utilizada para outros fins além da obtenção de médias ponderadas.

Na terminologia de redes convolucionais, o primeiro argumento (x) da convolução é referido como a entrada (*input*), e o segundo argumento (w) como o *kernel*. A saída da operação é chamada de mapa de características (*feature map*).

Assumindo que x e w são definidos apenas em índices inteiros t, podemos definir a convolução discreta como:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a) \tag{2.15}$$

Em aplicações de aprendizado de máquina, a entrada é geralmente um *array* multidimensional de dados, e o *kernel* é um *array* multidimensional de parâmetros que são ajustados pelo algoritmo de aprendizado. Esses *arrays* multidimensionais são também chamados de tensores. Como cada elemento da entrada e do *kernel* deve ser armazenado explicitamente, é comum assumir que essas funções são nulas em todos os pontos, exceto no conjunto finito de pontos para os quais os valores são armazenados. Isso permite, na prática, que o somatório infinito seja implementado como uma soma sobre um número finito de elementos.

Frequentemente, a convolução é aplicada em mais de um eixo simultaneamente. Por exemplo, se utilizarmos uma imagem bidimensional I como entrada, pode-se usar um *kernel* bidimensional K:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)\, K(i-m, j-n)\,. \tag{2.16}$$

Dado que a convolução é comutativa, a equação acima é equivalente a:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)\, K(m,n)\,. \tag{2.17}$$

A propriedade comutativa da convolução emerge da inversão do *kernel* em relação à entrada, no sentido de que, à medida que m aumenta, o índice na entrada aumenta enquanto o índice no *kernel* diminui. Embora a comutatividade seja útil para provas matemáticas, ela não é crucial na implementação de redes neurais. Muitas bibliotecas implementam a correlação cruzada (*cross-correlation*), que é idêntica à convolução, mas sem a inversão do *kernel*:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)\, K(m,n)\,. \tag{2.18}$$

Na Figura 2.1, apresenta-se um exemplo de convolução (sem a inversão do *kernel*) aplicada a um tensor 2D. A imagem mostra uma matriz de entrada 3x4 composta pelos elementos de 'a' até 'l', e um *kernel* 2x2 contendo os pesos 'w', 'x', 'y' e 'z'. O processo de convolução envolve a sobreposição do *kernel* sobre regiões 2x2 da matriz de entrada, multiplicando cada elemento da região pelo peso correspondente do *kernel* e somando os produtos obtidos. Essa operação é repetida deslizando o *kernel* horizontal e verticalmente sobre toda a matriz de entrada, produzindo uma matriz de saída 2x3, em que cada elemento é a soma ponderada dos elementos da região correspondente da entrada pelos pesos do *kernel*. Dessa forma, o resultado final mostra como os valores da matriz de entrada interagem com o *kernel*, evidenciando a extração de características locais da entrada pelo processo de convolução.

Uma camada típica de uma rede convolucional consiste em três estágios. No primeiro, a camada realiza várias convoluções em paralelo para produzir um conjunto de ativações lineares. No segundo estágio, cada ativação linear passa por uma função de ativação não linear, como a Unidade Linear Retificada (ReLU ($f(x) = \max(0, x)$)). Esse estágio é por vezes chamado de detecção ou ativação. No terceiro estágio, utiliza-se uma função de *pooling* para modificar ainda mais a saída da camada.

A camada de convolução utiliza o operador de convolução para extração de características. Um pequeno *kernel* é aplicado sobre a entrada (tensor), e um produto escalar entre cada elemento do *kernel* e a região correspondente do tensor de entrada é calculado. Os resultados formam o mapa de características, repetido com múltiplos *kernels* para gerar diferentes atributos. Os principais hiperparâmetros são o tamanho e o número de *kernels*, determinando a profundidade da saída.

A camada de ativação seleciona apenas os atributos relevantes. Funções não lineares como ReLU são aplicadas às saídas lineares para introduzir não linearidade.

A camada de *pooling* reduz a dimensionalidade dos mapas de características, diminuindo a dimensão espacial e controlando o sobreajuste. Operações comuns incluem *max pooling* e *average pooling*.

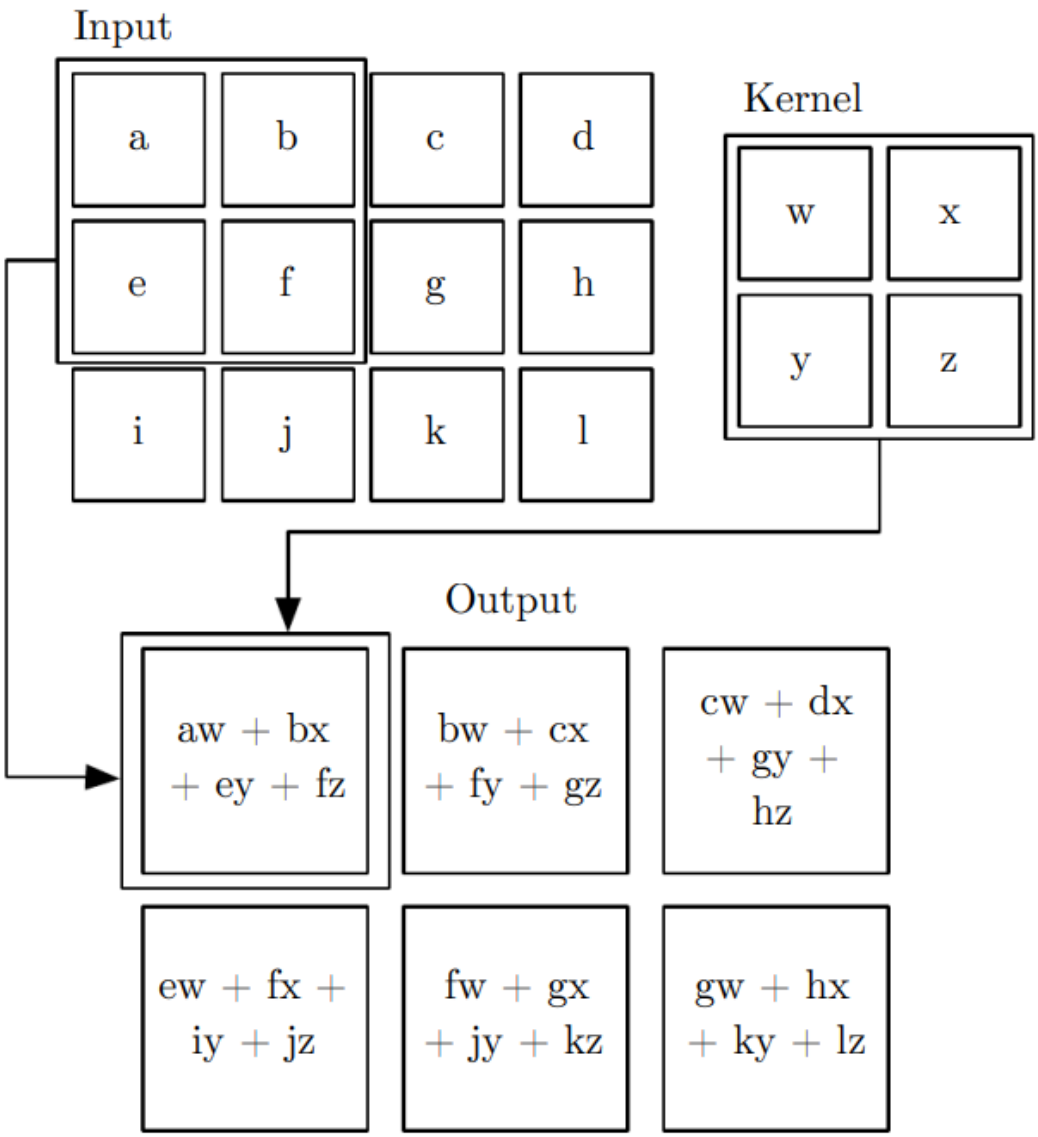

Figura 2.1 – Exemplo de convolução 2D sem inversão do kernel, onde o resultado é calculado apenas nas posições em que o kernel se ajusta totalmente à imagem (convolução "válida"). As caixas com setas mostram como o primeiro elemento do tensor de saída é obtido a partir da região correspondente do tensor de entrada. Fonte: (GOODFELLOW et al., 2016).

O *padding* adiciona zeros nas bordas da entrada para controlar seu tamanho, evitando redução rápida da dimensão espacial. O *stride* define o passo do *kernel* ou da janela de *pooling* sobre a entrada, manipulando a sobreposição entre regiões.

Os mapas de características resultantes da última camada de convolução ou *pooling* são transformados em vetores unidimensionais e conectados a camadas totalmente conectadas (*fully connected*). Nessas camadas, todas as entradas estão conectadas a cada saída por pesos aprendíveis, produzindo as saídas finais da rede, como probabilidades para cada classe.

2.6 TabNet: Aprendizado Profundo para Dados Tabulares

O TabNet, proposto por Arik e Pfister (2021), é uma arquitetura de rede neural profunda projetada especificamente para o aprendizado de dados tabulares, que busca combinar alta performance com interpretabilidade. Diferentemente de abordagens tradicionais que dominam o cenário de dados tabulares, como os ensembles de árvores de decisão, o TabNet introduz um método mais dinâmico e análogo ao funcionamento de modelos de aprendizado profundo em outras áreas.

Cada passo de decisão dentro do codificador TabNet é composto pelas seguintes partes que operam de forma integrada.

Transformador Atento (Attentive Transformer)

A principal função deste componente é determinar quais características devem ser processadas no passo atual. Ele recebe como entrada a informação processada do passo anterior ($a[i-1]$) e gera uma "máscara" de atenção ($M[i]$), que é um vetor de pesos indicando a importância de cada característica para a decisão.

O termo *Prior Scale* ($P[i]$) é utilizado para evitar que o modelo selecione repetidamente as mesmas características em todos os passos. Essa escala rastreia o quanto cada característica já foi utilizada nos passos anteriores, incentivando o modelo a explorar novos atributos a cada etapa. Além disso, a máscara gerada passa por uma função de normalização *Sparsemax*, que zera os pesos das características menos importantes, resultando em uma seleção de características esparsa, fundamental para a interpretabilidade do modelo.

Máscara de Características

A máscara esparsa $M[i]$, gerada pelo transformador atento, é aplicada multiplicativamente às características de entrada (f). Essa operação, representada por $M[i] \cdot f$, seleciona efetivamente as características mais salientes para aquele passo específico, enquanto ignora as demais, otimizando a capacidade de aprendizado do modelo.

Transformador de Características (Feature Transformer)

As características selecionadas pela máscara são então processadas pelo transformador de características, cuja arquitetura é projetada para ser eficiente e robusta. O transformador é composto por uma sequência de blocos de processamento, sendo que alguns possuem pesos compartilhados entre todos os passos de decisão, aprendendo representações gerais, enquanto outros possuem pesos específicos para cada passo, aprendendo representações especializadas.

Cada bloco consiste em uma camada totalmente conectada (FC), seguida por Normalização em Lote (Batch Normalization) e uma não linearidade GLU (Gated Linear Unit). O modelo também utiliza conexões residuais normalizadas para estabilizar o treinamento.

Bloco de Divisão (Split Block)

A saída do transformador de características é então dividida em duas partes:

1. Uma parte ($a[i]$) é enviada para o transformador atento do próximo passo de decisão ($i+1$), servindo como base para a seleção das características seguintes.
2. A outra parte ($d[i]$) é utilizada para compor a saída final do modelo.

Após a execução de todos os N_{steps} passos de decisão, as saídas destinadas à decisão final de cada passo ($d[i]$) são agregadas. Isso é realizado somando-se as saídas após a aplicação de uma função de ativação ReLU, conforme a equação:

$$d_{\text{out}} = \sum_{i=1}^{N_{\text{steps}}} \text{ReLU}(d[i])\,. \tag{2.19}$$

Finalmente, uma camada linear mapeia essa representação agregada (d_{out}) para a saída desejada, como as probabilidades de classe em um problema de classificação ou um valor numérico em uma tarefa de regressão.

O TabNet também inclui uma arquitetura de decodificador, utilizada para o pré-treinamento não supervisionado. Este decodificador é composto por transformadores de características em cada passo, seguidos por camadas totalmente conectadas. Sua função é reconstruir as características de entrada que foram intencionalmente mascaradas durante a fase de pré-treinamento, a partir da representação latente gerada pelo codificador. As saídas de cada passo do decodificador são somadas para obter a reconstrução final das características.

Adicionalmente, o TabNet foi pioneiro ao introduzir o aprendizado auto-supervisionado para dados tabulares. Neste modo de pré-treinamento, o modelo aprende a prever características que foram intencionalmente mascaradas (omitidas) a partir das que estão visíveis. Este processo, ilustrado na Figura 2.2, utiliza uma arquitetura de codificador-decodificador onde o TabNet atua como codificador para aprender representações robustas da estrutura dos dados. Esse pré-treinamento melhora significativamente o desempenho e acelera a convergência quando o modelo é posteriormente ajustado (fine-tuning) para uma tarefa supervisionada com dados rotulados, sendo especialmente útil em cenários com abundância de dados não rotulados.

2.7 SMOTE (Synthetic Minority Over-sampling Technique)

O método SMOTE (Synthetic Minority Over-sampling Technique), proposto por Chawla et al. (2002), é uma das abordagens mais utilizadas para lidar com o problema de desbalanceamento de classes em tarefas de aprendizado supervisionado. Esse desbalanceamento ocorre quando a quantidade de exemplos de uma classe é muito menor que a de outra, o que pode

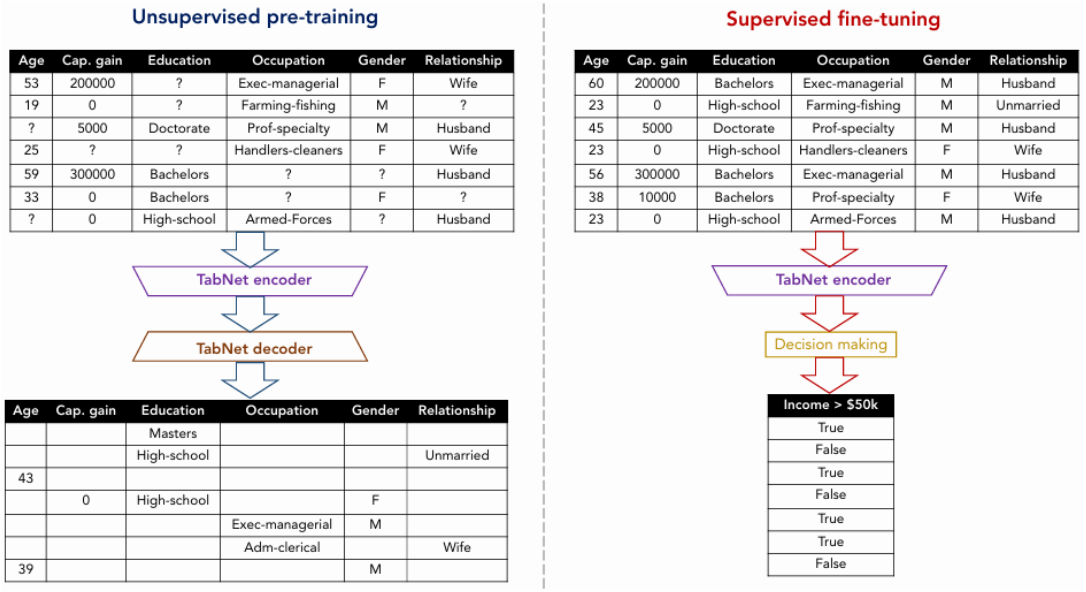

Figura 2.2 – Exemplo do processo de aprendizagem auto-supervisionada com TabNet. À esquerda, o pré-treinamento não supervisionado utiliza dados tabulares com valores ausentes, onde o codificador TabNet aprende representações ocultas e o decodificador tenta reconstruir os valores mascarados. À direita, o ajuste fino supervisionado aplica o codificador pré-treinado a uma tarefa específica de classificação (renda > $50k), integrando a tomada de decisão para gerar previsões finais. (ARIK; PFISTER, 2021)

levar o modelo a ter um viés em favor da classe majoritária, comprometendo sua capacidade de generalização.

O SMOTE busca mitigar esse problema por meio da geração de novas amostras sintéticas da classe minoritária, em vez de simplesmente replicar instâncias já existentes. Para isso, a técnica seleciona uma amostra da classe minoritária e identifica seus vizinhos mais próximos. A partir deles, gera novos exemplos intermediários, localizados no espaço de atributos entre a instância original e os vizinhos escolhidos. Dessa forma, os novos dados produzidos não são cópias, mas sim pontos adicionais que enriquecem a região ocupada pela classe minoritária, ampliando a sua representatividade no conjunto de dados.

Essa característica faz com que o SMOTE se diferencie das estratégias tradicionais de oversampling, que apenas repetem instâncias, evitando problemas como overfitting. Além disso, ao aumentar a diversidade da classe minoritária, a técnica favorece a aprendizagem de fronteiras de decisão mais bem definidas pelos modelos.

No entanto, é importante ressaltar que, embora seja bastante eficaz, o SMOTE pode gerar exemplos sintéticos em regiões ruidosas ou próximas da classe majoritária, o que pode introduzir ambiguidades e reduzir a qualidade do modelo se não for utilizado com cuidado. Assim, o SMOTE representa uma solução robusta e amplamente aplicada no pré-processamento de dados desbalanceados, sendo encontrado em diversos domínios, como detecção de fraude,

diagnóstico médico e reconhecimento de padrões, sempre com o objetivo de tornar o processo de aprendizado de máquina mais justo e eficiente. (FERNÁNDEZ et al., 2018)

2.8 Teoria dos Grafos

De acordo com Bondy e Murty (2008) em *Graph Theory*, a teoria dos grafos é uma área da matemática discreta que estuda estruturas denominadas grafos, compostas por vértices (ou nós) conectados por arestas (ou ligações). Essa teoria é utilizada para modelar e resolver problemas em diversas áreas, como ciência da computação, redes sociais e logística.

Seja $G = (V, E)$ um grafo, onde

$$V = \{v_1, v_2, \ldots, v_n\} \text{ e } E \subseteq \{(v_i, v_j) \mid v_i, v_j \in V\} \tag{2.20}$$

Os grafos podem ser classificados de diferentes maneiras. Grafos simples são aqueles que não possuem laços (arestas da forma (v_i, v_i)) nem múltiplas arestas entre os mesmos pares de vértices. Grafos direcionados, ou dígrafos, possuem arestas como pares ordenados (v_i, v_j), indicando a direção da conexão. Já os grafos ponderados associam a cada aresta (v_i, v_j) um peso $w_{ij} \in \mathbb{R}$, que pode representar custo, distância, similaridade ou outra métrica relevante.

O grau de um vértice v, denotado por $d(v)$, é o número de arestas incidentes a ele. Em grafos direcionados, distingue-se o grau de entrada $d^{in}(v)$, que conta as arestas que chegam a v, do grau de saída $d^{out}(v)$, que conta as arestas que saem de v. A vizinhança de um nó v é definida como $N(v) = \{v_j \in V \mid (v_i, v_j) \in E\}$, representando todos os nós diretamente conectados a v. Um caminho em G é uma sequência de vértices $P = (v_1, v_2, \ldots, v_k)$ tal que $(v_i, v_{i+1}) \in E$ para todo $1 \leq i < k$, enquanto um ciclo é um caminho fechado em que $v_1 = v_k$ e todos os demais vértices são distintos. Um grafo é considerado conexo se, para qualquer par de vértices $v_i, v_j \in V$, existe um caminho que os conecta.

Um *subgrafo* $G' = (V', E')$ de $G = (V, E)$ satisfaz $V' \subseteq V$ e $E' \subseteq E$, sendo que todas as arestas em E' conectam apenas vértices em V'. Particularmente, um subgrafo induzido por um conjunto de vértices $V' \subseteq V$ contém todas as arestas de G cujos extremos estão em V'.

Em muitas aplicações de GNNs, cada nó $v_i \in V$ possui um vetor de características ou atributos $x_i \in \mathbb{R}^f$, e cada aresta $(v_i, v_j) \in E$ pode possuir atributos e_{ij}. Essas informações são essenciais para a propagação de mensagens e a atualização das representações dos nós. A propagação normalmente considera a vizinhança do nó, podendo ser estendida para múltiplos níveis de vizinhos (*hops*) para capturar informações mais globais do grafo.

Para análises algébricas, os grafos podem ser representados através de matrizes, das quais destacamos:

Matriz de Adjacência

A *matriz de adjacência* $A \in \mathbb{R}^{n \times n}$ é definida por:

$$A_{ij} = \begin{cases} w_{ij}, & \text{se } (v_i, v_j) \in E \\ 0, & \text{caso contrário} \end{cases} \tag{2.21}$$

Para grafos não direcionados, a matriz A é simétrica ($A = A^T$). A normalização da matriz de adjacência, como $\tilde{A} = D^{-1/2} A D^{-1/2}$, é utilizada em GNNs para estabilizar a propagação de informações entre os nós.

Matriz de Graus

A *matriz de graus* $D \in \mathbb{R}^{n \times n}$ é uma matriz diagonal onde cada elemento da diagonal representa o grau do vértice correspondente:

$$D_{ii} = \sum_{j=1}^{n} A_{ij} \tag{2.22}$$

Para grafos direcionados, pode-se definir separadamente o grau de entrada d_i^{in} e o grau de saída d_i^{out}.

Matriz Laplaciana

A *matriz Laplaciana* L é uma ferramenta fundamental na Teoria Espectral dos Grafos e no funcionamento das GNNs. Ela é definida como:

$$L = D - A \tag{2.23}$$

Uma variante normalizada muito comum é a Laplaciana Simétrica:

$$L_{sym} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2.24}$$

onde I é a matriz identidade. Essa normalização é especialmente importante para estabilizar o fluxo de informação nas redes neurais sobre grafos. Grafos desconexos ou com múltiplos componentes conectados também podem ser tratados individualmente durante a propagação de mensagens.

Finalmente, os grafos podem ser utilizados em diferentes tipos de tarefas de aprendizado: a classificação de nós (*node classification*), a previsão de arestas (*edge prediction*) ou a classificação de grafos inteiros (*graph classification*), dependendo se os alvos estão associados a nós, arestas ou ao grafo como um todo.

2.9 Graph Neural Networks (GNN)

As redes neurais de grafos (GNNs), conforme introduzidas por Scarselli et al. (2008), constituem uma classe de arquiteturas neurais projetadas para operar diretamente sobre estruturas de grafos, sejam elas acíclicas, cíclicas, direcionadas ou não direcionadas. A motivação central reside no fato de que, em muitas aplicações do mundo real, os dados possuem natureza relacional e são mais adequadamente representados por grafos do que por vetores em espaços euclidianos. Diferentemente de abordagens tradicionais que exigem a vetorização dos grafos, a arquitetura das GNNs preserva explicitamente a topologia dos dados durante o processo de aprendizado, permitindo a modelagem direta de dependências estruturais.

Considere um grafo $G = (V, E)$, em que V representa o conjunto de vértices e E o conjunto de arestas. Suponha que cada vértice $v \in V$ esteja associado a um vetor de características $l(v) \in \mathbb{R}^p$, e que cada aresta $e = (u, v) \in E$ possua um vetor de características $l(e) \in \mathbb{R}^q$. A arquitetura GNN define duas funções principais: uma função de transição local, responsável por determinar o estado oculto h_v de um vértice v com base em sua vizinhança, e uma função de saída local, que produz a saída o_v a partir desse estado.

Matematicamente, essas funções podem ser descritas por:

$$h_v = f\left(l(v), l(e_{vu}), h_u, l(u) \mid u \in \mathcal{N}(v)\right) \tag{2.25}$$

$$o_v = g(h_v) \tag{2.26}$$

A função f promove a difusão de informação entre os vértices conectados, permitindo que cada nó atualize seu estado interno a partir dos estados de seus vizinhos de maneira iterativa, até que um ponto de equilíbrio seja alcançado. A função g, por sua vez, transforma o estado latente do vértice em uma saída utilizável na tarefa de interesse, como classificação ou regressão.

A atualização conjunta dos estados de todos os vértices pode ser representada por uma função global de transição, expressa por:

$$H = F(H, L) \tag{2.27}$$

Neste contexto, H é o vetor concatenado dos estados ocultos de todos os vértices do grafo, e L representa o conjunto de rótulos ou características dos vértices e arestas. Para assegurar que o processo iterativo convirja para uma solução única e estável, a arquitetura assume que F é uma aplicação contrativa. Isso implica a existência de uma constante $\alpha \in (0, 1)$ tal que:

$$|F(H_1, L) - F(H_2, L)| \leq \alpha |H_1 - H_2|, \quad \forall H_1, H_2 \tag{2.28}$$

A convergência da sequência iterativa definida por $H^{(t+1)} = F(H^{(t)}, L)$ para um ponto fixo H^* é garantida pelo Teorema do Ponto Fixo de Banach. Esse ponto fixo representa o estado

de equilíbrio da arquitetura, refletindo um processamento estável da informação propagada ao longo do grafo.

O treinamento da arquitetura é realizado em um contexto supervisionado, a partir de um conjunto de exemplos de treinamento constituído por trios da forma (G_k, v_k, y_k), em que G_k é um grafo de entrada, v_k é um vértice específico dentro desse grafo, e y_k é o rótulo esperado associado a esse vértice. O objetivo é ajustar os parâmetros das funções f e g de modo a minimizar a discrepância entre as saídas produzidas pela arquitetura e os rótulos esperados.

A função de custo comumente adotada é o erro quadrático médio:

$$\mathcal{L} = \sum_{k=1}^{N} |g(h_{v_k}) - y_k|^2 \tag{2.29}$$

Após a convergência do processo iterativo de atualização dos estados, o gradiente do erro é retropropagado pelos parâmetros da arquitetura. Esse processo utiliza uma adaptação do algoritmo de Almeida–Pineda, permitindo computar os gradientes de maneira eficiente sem armazenar todas as atualizações temporais das representações.

A arquitetura GNN generaliza tanto redes neurais recursivas quanto modelos baseados em caminhadas aleatórias. Diferente das redes recursivas, que operam apenas sobre estruturas acíclicas e exigem pré-processamento hierárquico, a GNN é capaz de lidar diretamente com grafos arbitrários, inclusive cíclicos e com topologias complexas. Em comparação com modelos de caminhada aleatória, a arquitetura GNN incorpora aprendizado paramétrico e supervisionado, o que a torna mais expressiva e adaptável a contextos específicos.

Sob condições regulares, a arquitetura GNN possui a propriedade de aproximação universal sobre funções definidas em grafos. Isso significa que, dada complexidade suficiente, ela é capaz de aproximar qualquer função contínua que dependa de um vértice e de sua vizinhança. Tal propriedade fundamenta sua aplicação em tarefas como classificação de nós, rotulagem de grafos e detecção de padrões estruturais, especialmente em domínios onde a relação entre elementos é tão importante quanto os elementos em si.

2.9.1 Arquitetura GCN (Graph Convolutional Network)

A arquitetura conhecida como *Graph Convolutional Network* (GCN), proposta por Kipf e Welling (2016), representa um avanço fundamental na generalização de convoluções para dados estruturados em grafos. Diferentemente de abordagens espectrais que requerem a decomposição espectral completa do Laplaciano do grafo, operação computacionalmente dispendiosa, a GCN baseia-se em uma aproximação de primeira ordem de convoluções espectrais localizadas, tornando-a escalável a grafos de grande porte.

A propagação de informações entre os nós do grafo é realizada camada a camada por meio da seguinte regra de atualização:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) \tag{2.30}$$

Neste contexto, $H^{(l)} \in \mathbb{R}^{n\times d_l}$ representa a matriz de ativações dos vértices na l-ésima camada, com $H^{(0)} = X$ correspondendo às características iniciais dos vértices. A matriz $\tilde{A} = A+I_n$ é a matriz de adjacência com auto-laços adicionados, assegurando que cada vértice também leve em conta suas próprias características durante a propagação. A matriz $\tilde{D}$ é a matriz diagonal de graus correspondente, definida por $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. O termo $W^{(l)} \in \mathbb{R}^{d_l\times d_{l+1}}$ representa a matriz de pesos treinável da camada, e $\sigma(\cdot)$ é uma função de ativação não-linear, usualmente a ReLU.

A expressão $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ realiza uma normalização simétrica da matriz de adjacência, essencial para estabilizar o fluxo de gradientes em grafos com alta variância nos graus dos vértices, além de preservar a escala espectral da operação. Essa normalização resulta de um processo denominado *truque de renormalização*, que substitui a forma original $I + D^{-1/2}AD^{-1/2}$, cuja aplicação repetida pode causar instabilidades numéricas.

A operação de convolução espectral aproximada, na qual se baseia a GCN, pode ser entendida como uma aplicação sucessiva de filtros lineares localizados sobre o espectro do Laplaciano normalizado do grafo. Em vez de utilizar uma parametrização explícita via polinômios de Chebyshev, como em abordagens anteriores, a GCN adota uma forma simplificada, restrita ao primeiro vizinho (K=1), mas com profundidade de camadas capaz de expandir o campo receptivo ao longo das vizinhanças mais distantes.

Na prática, o modelo completo de uma GCN com duas camadas para classificação semi-supervisionada de vértices pode ser expresso como:

$$Z = \text{softmax}\left(\hat{A}\cdot\sigma\left(\hat{A}\cdot XW^{(0)}\right)W^{(1)}\right) \tag{2.31}$$

onde $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ é calculada previamente, $W^{(0)}$ e $W^{(1)}$ são as matrizes de pesos das camadas, e $\text{softmax}(\cdot)$ é aplicada linha a linha sobre a saída para produzir probabilidades sobre as classes.

Essa arquitetura permite o aprendizado simultâneo das estruturas do grafo e das características dos vértices, possibilitando a generalização para vértices não rotulados com alto desempenho. A complexidade computacional do modelo é linear no número de arestas do grafo, o que a torna aplicável em cenários com grande escala e alta esparsidade, como redes de citações, conhecimento e redes sociais.

2.9.2 Mecanismo de Atenção em Grafos (Graph Attention Network - GAT)

A arquitetura *Graph Attention Network* (GAT), introduzida por Veličković et al. (2017), propõe uma generalização do mecanismo de autoatenção para o domínio de grafos, permitindo

que vértices atribuam pesos distintos às informações recebidas de seus vizinhos. Essa abordagem supera limitações de métodos anteriores que impõem pesos fixos ou dependem da estrutura global do grafo, ao mesmo tempo em que evita operações espectrais custosas, como a decomposição do Laplaciano.

Considere um grafo $G = (V, E)$ com $|V| = N$ vértices, onde cada vértice $v_i \in V$ possui uma representação de entrada $\vec{h}_i \in \mathbb{R}^F$, com F sendo a dimensionalidade das características. A camada de atenção produz vetores transformados $\vec{h}'_i \in \mathbb{R}^{F'}$, onde F' é a nova dimensionalidade. O primeiro passo consiste em aplicar uma transformação linear compartilhada a cada vetor de entrada, por meio de uma matriz de pesos $W \in \mathbb{R}^{F' \times F}$:

$$\vec{h}_i^W = W\vec{h}_i \tag{2.32}$$

Em seguida, calcula-se o coeficiente de atenção não normalizado e_{ij} entre os vértices i e j, com base na concatenação dos vetores transformados:

$$e_{ij} = \text{LeakyReLU}\left(\vec{a}^\top [\vec{h}_i^W, |, \vec{h}_j^W]\right) \tag{2.33}$$

onde $\vec{a} \in \mathbb{R}^{2F'}$ é um vetor de pesos treinável, $\|$ denota a concatenação, e a função LeakyReLU é utilizada com inclinação negativa $\alpha = 0.2$.

Para preservar a estrutura local do grafo, os coeficientes e_{ij} são computados apenas para os vizinhos diretos de v_i, isto é, para $j \in \mathcal{N}(i)$. Esses coeficientes são então normalizados via uma função softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})} \tag{2.34}$$

A nova representação do vértice v_i é obtida como uma combinação linear ponderada das representações transformadas dos seus vizinhos, seguida de uma função de ativação σ (geralmente ELU ou ReLU):

$$\vec{h}i' = \sigma\left(\sum j \in \mathcal{N}(i)\alpha_{ij}\vec{h}_j^W\right) \tag{2.35}$$

Para estabilizar o treinamento e enriquecer a capacidade expressiva do modelo, os autores propõem o uso de atenção multi-cabeça. Neste caso, K mecanismos de atenção independentes são aplicados em paralelo, e suas saídas podem ser concatenadas (nas camadas intermediárias) ou agregadas por média (na camada de saída). A formulação com concatenação é dada por:

$$\vec{h}i' = \|k = 1^K \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k)} W^{(k)} \vec{h}_j\right) \tag{2.36}$$

e, na última camada (onde é comum realizar classificação), a agregação é feita por média:

$$\vec{h}i' = \sigma\left(\frac{1}{K}\sum k = 1^{K}\sum_{j\in\mathcal{N}(i)}\alpha_{ij}^{(k)}W^{(k)}\vec{h}_j\right) \tag{2.37}$$

Essa arquitetura é particularmente eficiente computacionalmente, pois permite paralelização sobre todas as arestas do grafo e elimina a necessidade de conhecer a estrutura global antecipadamente. Além disso, ao permitir que vértices de uma mesma vizinhança tenham pesos distintos, a GAT proporciona maior capacidade de modelagem e interpretabilidade, sendo aplicável tanto em cenários transdutivos quanto indutivos.

2.9.3 Amostragem e Agregação em Grafos (GraphSAGE)

A arquitetura *GraphSAGE* (Sample and Aggregate), proposta por Hamilton, Ying e Leskovec (2017), é um framework indutivo projetado para gerar representações de baixa dimensionalidade (*embeddings*) para vértices em grafos. Diferentemente de abordagens transdutivas que aprendem um vetor de *embedding* para cada vértice individualmente, o GraphSAGE aprende uma função que mapeia as características locais de um vértice para uma representação vetorial. Isso permite que o método generalize para vértices não vistos durante o treinamento, uma capacidade crucial para grafos dinâmicos e de larga escala.

O processo consiste em iterativamente agregar informações da vizinhança de um vértice. Dado um grafo $G = (V, E)$ e um conjunto de características de entrada $\vec{x}_v, \forall v \in V$, onde $\vec{x}_v \in \mathbb{R}^F$, a representação do vértice v na camada k é denotada por $\vec{h}_v^k$. A representação inicial é a própria característica de entrada, $\vec{h}_v^0 = \vec{x}_v$. Para cada camada $k \in 1, \ldots, K$, a representação $\vec{h}_v^k$ é atualizada em duas etapas:

Primeiro, ocorre a agregação. Uma função agregadora AGGREGATE_k coleta as representações dos vizinhos de v, $\mathcal{N}(v)$, da camada anterior e as combina em um único vetor $\vec{h}_{\mathcal{N}(v)}^{k-1}$. Para manter a complexidade computacional fixa, em vez de usar a vizinhança completa, o GraphSAGE realiza uma amostragem uniforme de um número fixo de vizinhos:

$$\vec{h}_{\mathcal{N}(v)}^{k-1} = \text{AGGREGATE}_k(\{\vec{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \tag{2.38}$$

Em seguida, ocorre a atualização. O vetor agregado $\vec{h}_{\mathcal{N}(v)}^{k-1}$ é concatenado com a representação do próprio vértice da camada anterior, $\vec{h}_v^{k-1}$, e a combinação é passada por uma camada densa com uma matriz de pesos W^k e uma função de ativação não linear σ:

$$\vec{h}_v^k = \sigma\left(W^k \cdot [\vec{h}_v^{k-1} \| \vec{h}_{\mathcal{N}(v)}^{k-1}]\right) \tag{2.39}$$

Onde $\|$ denota a concatenação. Opcionalmente, uma normalização L_2 pode ser aplicada a $\vec{h}_v^k$ após a atualização. A representação final do vértice, $\vec{z}_v$, é a saída da última iteração, $\vec{h}_v^K$.

A flexibilidade do GraphSAGE reside na escolha da função agregadora, que deve ser capaz de operar sobre um conjunto não ordenado de vetores. Três arquiteturas principais foram investigadas: o Agregador de Média (Mean Aggregator) calcula a média elemento a elemento dos vetores dos vizinhos; o Agregador LSTM (LSTM Aggregator) aplica uma LSTM a uma permutação aleatória dos vetores da vizinhança, explorando correlações mais complexas entre os vizinhos; e o Agregador de Agrupamento (Pooling Aggregator), uma função simétrica e treinável, processa cada vetor de vizinho independentemente por uma rede neural (perceptron) e depois realiza uma operação de *max-pooling* elemento a elemento para agregar a informação:

$$\text{AGGREGATE}_{\text{pool}} = \max(\sigma(W_{\text{pool}}\vec{h}_u + \vec{b}), \forall u \in \mathcal{N}(v)) \tag{2.40}$$

Para o treinamento dos parâmetros do modelo (as matrizes W^k e os parâmetros dos agregadores), pode-se utilizar uma abordagem não supervisionada. Neste caso, uma função de perda baseada no grafo é empregada para forçar que vértices próximos tenham representações similares, enquanto vértices distantes tenham representações distintas:

$$J_{\mathcal{G}}(\vec{z}_u) = -\log(\sigma(\vec{z}_u^{\top}\vec{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\vec{z}_u^{\top}\vec{z}_{v_n})) \tag{2.41}$$

onde v é um vizinho de u (obtido por caminhada aleatória), P_n é uma distribuição de amostragem negativa, e Q é o número de amostras negativas. Alternativamente, o modelo pode ser treinado de forma totalmente supervisionada, substituindo essa perda por um objetivo específico da tarefa, como a classificação de vértices. A capacidade de gerar representações para novos vértices sem a necessidade de retreinamento completo torna o GraphSAGE uma arquitetura eficiente e escalável para aplicações em grafos do mundo real.

2.10 Aplicações de GNNs em Detecção de Fraude

As Redes Neurais de Grafos (GNNs) são utilizadas na detecção de fraudes financeiras por sua capacidade de modelar transações e contas como um grafo, onde os nós representam contas e as arestas direcionadas indicam transações, com pesos que podem refletir valores ou outros atributos relevantes. Através de algoritmos de passagem de mensagens, as GNNs propagam informações entre os nós, capturando padrões complexos e relações não lineares que podem indicar comportamentos fraudulentos, como lavagem de dinheiro. Esses modelos permitem o aprendizado automático de características, eliminando a necessidade de engenharia extensiva de recursos, e possibilitam a integração de dados multimodais, como transações numéricas, registros de comunicação textual e atributos categóricos de usuários, em um único framework. Além disso, as GNNs geram pontuações de fraude para contas ou transações, facilitando a análise de atividades suspeitas por analistas humanos e aprimorando as estratégias de detecção de fraudes (MOTIE; RAAHEMI, 2024; CHENG et al., 2024).

3 Trabalhos Relacionados

O estudo de Souza (2023) explorou o uso de uma taxonomia hierárquica para identificar contas fraudulentas em um conjunto de dados de contas digitais financeiras, alcançando uma acurácia de 92,3% e *F1-score* de 89% na classificação binária. Os objetivos incluíram a investigação de três estratégias de classificação: plana, hierárquica local e hierárquica global, sendo que a abordagem hierárquica global alcançou a maior precisão ponderada (91,7%), superando a abordagem plana (89,2%). Também foi analisado o impacto de técnicas de balanceamento de dados, com o *SMOTE* aumentando a sensibilidade de 78,5% para 85,9%, enquanto o *Borderline SMOTE* elevou o *F-score* para 87%, em comparação com 82% sem balanceamento. Por fim, exploraram-se novas formas de representação e o uso de *deep learning* com dados matriciais, visando aprimorar a detecção de fraudes. Questões como a eficácia de métodos de sobreamostragem, a contribuição da taxonomia hierárquica e o desempenho de redes neurais convolucionais em comparação com classificadores tradicionais foram analisadas. Os resultados apresentados demonstraram que os métodos de balanceamento *SMOTE* e *Borderline SMOTE* aumentaram a sensibilidade e o *F-score* na identificação de fraudes para classificadores tradicionais, enquanto métodos como *CTGAN* e *TVAE* tiveram impacto limitado ou negativo. Na classificação binária, redes neurais convolucionais associadas ao *SMOTE* obtiveram a maior sensibilidade (88,4%), superando os classificadores tradicionais como *Random Forest* (84,1%). Já na classificação multiclasse, a abordagem hierárquica destacou-se em métricas ponderadas de precisão e *F-score* (*F-score* ponderado de 85%), enquanto a abordagem plana obteve melhor sensibilidade (82,3%). Esses achados reforçam a importância do balanceamento de dados e a eficácia das abordagens hierárquicas e de *deep learning* na detecção de fraudes financeiras.

Entre as possíveis extensões do estudo, Souza (2023) sugere a investigação da interpretabilidade dos modelos, com foco nos atributos mais influentes e no uso de mecanismos de atenção; a exploração de outras arquiteturas de *CNNs* por meio de busca automatizada; o emprego de aprendizado multitarefa para aprimorar a abordagem hierárquica global das redes; a aplicação de técnicas de detecção *Out-Of-Distribution* para identificação de contas fraudulentas; e a adoção de arquiteturas baseadas em *Transformers* para esse fim.

O estudo de Motie e Raahemi (2024) realizou uma revisão sistemática sobre o uso de Redes Neurais em Grafos (*GNNs*) na detecção de fraudes financeiras, identificando os tipos de *GNNs* propostos e lacunas na pesquisa atual. Após uma seleção rigorosa, foram analisados 33 estudos que abordam a aplicação de *GNNs* em diferentes contextos de fraude financeira, como transações *online* e contas bancárias, onde modelos como *GraphSAGE* alcançaram *AUC-ROC* de 0,93 em transações bancárias, enquanto *GATs* (*Graph Attention Networks*) obtiveram precisão de 89,5% em fraudes de cartão de crédito. Os resultados indicaram um aumento significativo no número de publicações sobre *GNNs* para detecção de fraudes financeiras nos últimos anos, evidenciando

um crescente interesse acadêmico na área. Apesar de avanços promissores, limitações como a escassez de estudos sobre grafos heterogêneos (apenas 28% dos estudos analisados) – que apresentaram ganhos de 3-5% em *F1-score* – e a exclusão de literatura não revisada por pares ainda persistem. O estudo enfatiza a necessidade de aprofundamento na aplicação de *GNNs* a grafos dinâmicos e no uso de aprendizado não supervisionado ou semi-supervisionado (que melhoraram a detecção em 7,2% em média) para melhorar a eficácia desses modelos na detecção de fraudes.

O estudo de Cheng et al. (2024) apresenta uma revisão abrangente sobre a aplicação de *GNNs* na detecção de fraudes financeiras, analisando mais de 100 estudos onde esses modelos superaram métodos tradicionais em 12-18% no *AUC* para fraudes complexas. Os autores argumentam que métodos tradicionais de detecção, baseados em regras ou aprendizado de máquina clássico, enfrentam dificuldades para lidar com padrões complexos e dinâmicos de fraude. Em contrapartida, *GNNs* demonstram uma capacidade superior de modelar relações entre entidades financeiras, capturando conexões ocultas entre transações e participantes do sistema, com modelos como *GTNs* (*Graph Temporal Networks*) atingindo 93,1% de acurácia em dados dinâmicos. O artigo categoriza abordagens existentes, analisando mais de 100 estudos e explorando desafios técnicos, como escalabilidade (afetando 63% dos modelos), interpretabilidade e adaptação a padrões dinâmicos.

Além de revisar modelos de *GNNs* aplicados à detecção de fraudes, como *Graph Convolutional Networks* (*GCNs*), *Graph Attention Networks* (*GATs*) e *Graph Temporal Networks* (*GTNs*), Cheng et al. (2024) enfatizam a importância da modelagem de dados financeiros como grafos heterogêneos e dinâmicos. Técnicas como aprendizado por reforço e pré-treinamento (que reduziram falsos positivos em 22%) podem aumentar a precisão e adaptabilidade dos modelos. O estudo também sugere a fusão de dados multimodais, combinando informações transacionais, comportamentais e contextuais para uma análise mais robusta de fraudes financeiras, abordagem que aumentou a precisão para 91,4% (*vs.* 86,7% em grafos simples).

O artigo de Šiljak (2008) propõe um modelo matemático para analisar a evolução dinâmica de grafos em sistemas complexos interconectados, com estabilidade conectiva garantida matematicamente para sistemas com até 30% de perturbações estruturais. O autor introduz o conceito de estabilidade conectiva para garantir a funcionalidade de sistemas interligados diante de perturbações estruturais, utilizando a teoria de Lyapunov e funções vetoriais de estabilidade. Além disso, propõe um coordenador de interconexões que ajusta dinamicamente os níveis de conexão entre subsistemas, assegurando o equilíbrio estável do sistema (mantendo 87,5% das conexões críticas estáveis durante ataques coordenados em simulações). A modelagem de grafos como matrizes de adjacência dinâmicas permite representar matematicamente as conexões variáveis entre agentes do sistema. A aplicação desse estudo à detecção de fraudes financeiras é relevante, pois a estabilidade conectiva pode ser empregada para modelar a evolução de transações suspeitas ao longo do tempo. Assim, algoritmos baseados em *GNNs* podem capturar padrões

dinâmicos de fraude.

No estudo de Ren et al. (2023), um modelo baseado em *GNNs* dinâmicas, chamado *CORE-DGNN*, foi proposto para detectar fraudes colaborativas em redes de telecomunicações, explorando a evolução temporal das interações fraudulentas e atingindo 94,2% de *AUC*. Embora focado em fraudes telefônicas, seus conceitos são altamente aplicáveis à detecção de fraudes financeiras, especialmente no combate a esquemas de lavagem de dinheiro e fraudes transacionais em rede. A modelagem dinâmica de redes e o uso de técnicas como filtros de frequência (reduzindo a heterofilia em 41%) e convolução temporal com autoatenção (81,3% de precisão na atenção dinâmica) podem ser adaptados para analisar transações bancárias ao longo do tempo, aprimorando a detecção de fraudes em cenários financeiros complexos, com adaptações do modelo alcançando 91,8% de *F1-score* em detecção de lavagem de dinheiro em dados simulados.

Os estudos revisados fornecem um suporte teórico e metodológico fundamental para a proposta deste estudo, tanto ao evidenciar a relevância das GNNs na detecção de fraudes quanto ao destacar desafios ainda não resolvidos. O estudo de Souza (2023), em particular, será utilizado como baseline, fornecendo um conjunto de dados já testado com abordagens tradicionais e redes convolucionais, o que permitirá uma comparação direta com a aplicação de GNNs. Enquanto pesquisas como Motie e Raahemi (2024) e Cheng et al. (2024) reforçam o potencial das GNNs na captura de relações complexas entre transações, estudos como Ren et al. (2023) e Šiljak (2008) trazem insights sobre a modelagem dinâmica de grafos, uma direção que pode ser explorada para aprimorar a detecção de fraudes bancárias. Assim, este estudo busca contribuir ao avaliar empiricamente o desempenho das GNNs nesse contexto específico, analisando se sua capacidade de representar interações entre contas bancárias pode superar abordagens anteriores na identificação de contas fraudulentas.

4 Materiais e métodos

4.1 Conjunto de dados

Os dados empregados neste estudo foram fornecidos pela instituição financeira brasileira Efí S.A., estruturados em um banco de dados relacional contendo atributos numéricos, booleanos e categóricos. O conjunto inclui informações cadastrais de 45.209 clientes, tanto pessoas físicas quanto jurídicas, abrangendo diferentes padrões de comportamento, bem como registros de suas transações financeiras. Entre os clientes, há aqueles que realizam movimentações diariamente, semanalmente, mensalmente, e outros que possuem conta, mas não a utilizam. O período abrangido pelos dados vai de agosto de 2019 a outubro de 2021. Todas as informações foram anonimizadas e normalizadas, garantindo que os valores reais permanecessem protegidos.

As transações disponíveis no conjunto de dados incluem envio e recebimento de Pix (Pagamento Instantâneo Brasileiro), envio e recebimento de TED (Transferência Eletrônica Disponível), pagamentos de contas e emissão de boletos.

As contas do conjunto de dados foram rotuladas por especialistas do setor financeiro da instituição parceira. Essa rotulação divide as contas em seis classes distintas. As contas não fraudulentas são subdivididas em quatro classes: A, B, C e D. Já as classes E e $Fraude$ representam, respectivamente, contas suspeitas e contas fraudulentas confirmadas.

Tabela 4.1 – Descrição das classes de contas

Classe	**Descrição**
A	Contas não fraudulentas, classificadas como de baixo risco e com comportamento financeiro estável.
B	Contas não fraudulentas, mas com pequenas inconsistências ou características que exigem monitoramento.
C	Contas não fraudulentas, mas com maior risco de fraude futura devido a comportamentos suspeitos ou anômalos.
D	Contas não fraudulentas, porém com características que dificultam a análise de risco e exigem atenção especial.
E	Contas suspeitas, que apresentam fortes indícios de fraude, mas sem confirmação definitiva.
$Fraude$	Contas confirmadas como fraudulentas, utilizadas para treinar e validar os modelos de detecção.

A Tabela 4.2 apresenta a distribuição das classes no conjunto de dados. Nota-se que o conjunto é fortemente desbalanceado, com a classe D representando a maior parte das contas.

Tabela 4.2 – Distribuição das classes do conjunto de dados

	Quantidade	**Percentual (%)**
Classe A	37	0.08
Classe B	323	0.71
Classe C	3.397	7.51
Classe D	36.993	81.83
Classe E	3.646	8.07
Classe F	813	1.80
Total	**45.209**	**100**

4.2 Construção do grafo inicial (G_1)

O primeiro grafo, denotado por G_1, foi construído considerando todas as classes do conjunto de dados. Para representar as relações financeiras entre documentos e contas, foi utilizada uma abordagem baseada em grafos direcionados. O processo iniciou-se com o carregamento e integração de diversos conjuntos de dados em formato CSV, contendo informações de contas, documentos, níveis de análise, transações via PIX e TED, e pagamentos de contas.

Em seguida, foi realizado o pré-processamento dos dados, no qual cada documento recebeu um conjunto de atributos agregados. Estes atributos incluem informações cadastrais das contas associadas, como número, cidade, estado, grupo e classe; o nível mais relevante do documento; métricas de transações, como número de transações realizadas e recebidas, valor médio, desvio padrão, número de estornos e cancelamentos; além do total e quantidade de pagamentos de contas realizados.

Com os dados preparados, foi construído um grafo direcionado, em que cada nó representa um documento e contém seus atributos agregados. As arestas representam transações financeiras entre documentos, com métricas associadas, incluindo valor médio, número de transações e valor total transacionado. As transações do tipo PIX e TED foram adicionadas como arestas entre os documentos correspondentes, enquanto os pagamentos de contas, que não possuem destinatário específico, foram contabilizados nas métricas do nó correspondente. A Tabela 4.1 apresenta os atributos dos nós utilizados nesse grafo, enquanto a Tabela 4.2 detalha os atributos das arestas.

Antes da exportação para GML, todos os atributos de nós e arestas foram convertidos para `string` para garantir compatibilidade com o formato GML, e o grafo resultante G_1 foi salvo. Estatísticas do grafo, como número total de nós e arestas, foram registradas, permitindo a análise estrutural e a identificação de padrões financeiros que podem indicar potenciais fraudes.

A seguir, são detalhadas nas Tabelas 4.3 e 4.4 as features de nós e arestas utilizadas no grafo, respectivamente.

Tabela 4.3 – Features dos nós G_1

Atributo	**Descrição**
document_id	Identificador do documento.
documento	Número/valor do documento (ex.: CPF/CNPJ anonimizado).
documento_tipo	Tipo do documento (PF/PJ, quando disponível).
profile_id	Identificador do perfil associado (quando existir).
corporation_id	Identificador de corporação (se aplicável).
nivel_relevante	Nível mais relevante do *level* (agregado por documento).
status_nivel	Status associado ao nível relevante.
conta_numeros	Lista dos números de conta vinculados ao documento.
conta_cidades	Lista de cidades das contas vinculadas.
conta_estados	Lista de estados das contas vinculadas.
conta_grupo	Lista/valor do grupo da conta (quando disponível).
conta_classe	Classes associadas (A–F) às contas do documento.
pix_env_qtd	Quantidade de transações PIX enviadas pelo documento.
pix_rec_qtd	Quantidade de transações PIX recebidas pelo documento.
pix_env_valor_total	Soma dos valores de PIX enviados.
pix_rec_valor_total	Soma dos valores de PIX recebidos.
pix_env_valor_medio	Valor médio dos PIX enviados.
pix_rec_valor_medio	Valor médio dos PIX recebidos.
ted_env_qtd	Quantidade de TEDs enviadas.
ted_rec_qtd	Quantidade de TEDs recebidas.
ted_env_valor_total	Soma dos valores de TED enviadas.
ted_rec_valor_total	Soma dos valores de TED recebidas.
ted_env_valor_medio	Valor médio das TED enviadas.
ted_rec_valor_medio	Valor médio das TED recebidas.
ted_estornos_qtd	Número de estornos (TED) associados ao documento.
ted_cancelamentos_qtd	Número de cancelamentos (TED) associados ao documento.
pagto_contas_qtd	Quantidade de pagamentos de contas realizados.
pagto_contas_valor_total	Soma dos valores de pagamentos de contas.
valor_medio_global	Média dos valores transacionados (todas as operações).
desvio_padrao_global	Desvio padrão dos valores transacionados.

4.3 Construção do segundo grafo (G_2)

O segundo grafo, denotado por G_2, foi construído a partir de G_1 aplicando-se um filtro para reter apenas as classes de maior interesse para o estudo: classe C e classe Fraude. Essa escolha possibilitou a criação de um cenário binário de classificação. As demais classes não foram utilizadas, pois apresentavam distribuições desbalanceadas, algumas com quantidade excessiva de dados e outras com amostras muito reduzidas, o que comprometeria a análise.

Para lidar com situações em que um mesmo documento estivesse associado simultaneamente a diferentes rótulos de nível, foi estabelecida uma regra de prioridade entre eles. O rótulo *Fraude* foi definido como prioritário em relação ao rótulo *C*, de forma que, sempre que ambos estivessem presentes para o mesmo documento, o nó correspondente no grafo recebesse

Tabela 4.4 – Features das arestas G_1

Atributo	**Descrição**
source	Documento de origem (remetente).
target	Documento de destino (favorecido).
tipo_transacao	Tipo da transação agregada na aresta (PIX ou TED).
qtd_transacoes	Número total de transações entre *source* e *target*.
valor_total	Soma dos valores transacionados na aresta.
valor_medio	Valor médio das transações na aresta.
valor_desvio_padrao	Desvio padrão do valor das transações na aresta.
qtd_estornos	Quantidade de estornos (aplicável a TED).
qtd_cancelamentos	Quantidade de cancelamentos (aplicável a TED).
data_primeira	Data da primeira transação entre os documentos.
data_ultima	Data da última transação entre os documentos.
tipo_origem	Tipo do documento de origem (PF/PJ), quando disponível.
tipo_destino	Tipo do documento de destino (PF/PJ), quando disponível.
contas_origem	Conjunto/lista de contas de origem observadas na aresta.
contas_destino	Conjunto/lista de contas de destino observadas na aresta.

apenas o rótulo *Fraude*. Essa escolha foi feita para evitar inconsistências e ambiguidades na representação, já que a presença de indícios de fraude é mais crítica para o objetivo do estudo do que a classificação por retorno financeiro. Dessa forma, cada documento permanece no grafo associado a um único rótulo, reduzindo ruídos nos dados e assegurando que os nós representem corretamente o status de risco analisado.

O grafo foi então construído de forma direcionada, incluindo apenas documentos pertencentes a uma dessas duas classes ou diretamente conectados a eles por transações. Com isso, foram removidos nós e arestas sem relevância para a análise de fraude, resultando em uma estrutura mais enxuta e focada.

Essas restrições tornam G_2 mais representativo para a investigação de padrões transacionais relacionados a potenciais fraudes, além de reduzir significativamente a complexidade estrutural em comparação a G_1.

4.4 Construção do terceiro grafo (G_3)

O terceiro grafo, denotado por G_3, corresponde a uma versão enriquecida e filtrada, contendo apenas as classes C e Fraude, mas incorporando atributos adicionais de natureza comportamental. O objetivo foi criar uma estrutura de dados detalhada, capturando padrões temporais e estatísticos das transações, para então aplicar um filtro e concentrar a análise nas classes de maior relevância para o estudo: “C” e “Fraude”. A metodologia foi estruturada em etapas sequenciais, conforme descrito a seguir.

Inicialmente, os dados transacionais, dispersos em múltiplos arquivos de origem, foram

unificados em conjuntos de dados coesos por tipo de operação (transferências eletrônicas e pagamentos de contas). Em seguida, aplicou-se um critério de filtragem rigoroso, retendo-se apenas as transações cujos participantes, tanto remetente quanto destinatário, estivessem presentes em um conjunto de dados cadastrais de referência. Essa etapa foi fundamental para garantir a integridade do ecossistema a ser modelado, assegurando que o grafo representasse apenas interações entre entidades conhecidas.

Os registros de transferências que atenderam ao critério de filtragem foram então padronizados em uma estrutura de dados unificada. Essa estrutura contemplava campos essenciais como documento de origem, documento de destino, valor, data e tipo da transação. A consolidação resultou em um repositório centralizado de todas as interações financeiras diretas, que serviu como base para a posterior criação das arestas do grafo.

Para o enriquecimento dos nós do grafo, que representam as entidades (documentos), foi conduzido um processo de engenharia de atributos a partir do histórico de pagamentos de contas. Para cada entidade, foram calculadas métricas comportamentais que descrevem seus padrões de pagamento, como a distribuição das operações ao longo do dia, da semana e do mês. Adicionalmente, foram computadas estatísticas sobre os valores transacionados e a frequência de ocorrências como estornos e cancelamentos. Esses novos atributos foram, então, integrados aos dados cadastrais de referência, formando uma base de dados consolidada e enriquecida para os nós. Para as entidades sem histórico de pagamento de contas, os valores desses atributos foram definidos como nulos (zero), mantendo a consistência estrutural do conjunto de dados.

De maneira análoga, as arestas do grafo, que representam as interações financeiras entre as entidades, também foram enriquecidas. As transações foram agrupadas por par de interação (origem-destino), e para cada relação, foram computadas métricas estatísticas e temporais semelhantes às calculadas para os nós. Este processo transformou cada aresta em um vetor de características que descreve o perfil comportamental da relação entre duas entidades, incluindo a frequência, o volume financeiro e os padrões temporais de suas transações.

A etapa final consistiu na montagem da estrutura do grafo. Inicialmente, o conjunto de nós foi filtrado para incluir apenas as entidades classificadas como “C” ou “Fraude”. Com este subconjunto de nós definido, as arestas foram subsequentemente filtradas para reter apenas as conexões existentes entre eles. Finalmente, G_3 foi exportado para o formato GEXF, contendo tanto atributos cadastrais quanto estatísticos e temporais. Essa estrutura apresenta maior riqueza informacional em comparação a G_2, permitindo uma análise mais aprofundada sobre os padrões de fraude.

Tabela 4.5 – Features dos nós G_3

Atributo	**Descrição**
document_id	Identificador único do documento.
profile_id	Identificador do perfil associado.
corporation_id	Identificador da corporação vinculada.
documento	Identificador do documento (ex.: CPF/CNPJ anonimizado).
documento_tipo	Tipo de documento (PF ou PJ).
level	Classe atribuída ao documento (C ou Fraude).
cpf	CPF associado ao perfil (anonimizado).
profile_created_at	Data de criação do perfil.
profile_birth	Data de nascimento do titular.
profile_career	Profissão ou carreira declarada.
profile_is_pep	Indicador de Pessoa Politicamente Exposta (PEP).
profile_is_foreign	Indicador de pessoa estrangeira.
profile_cellphone_ddd	DDD do celular cadastrado.
profile_dominio_email	Domínio do e-mail do perfil.
profile_length_email	Comprimento do e-mail.
profile_name_in_email	Indicação de presença de nome no e-mail.
profile_number_initial_email	Dígito inicial no e-mail (se houver).
profile_number_final_email	Dígito final no e-mail (se houver).
profile_number_in_email	Quantidade de números no e-mail.
users	Número de usuários associados.
contact_by_phone	Indica se há contato telefônico.
pct_00_05, pct_06_11, pct_12_17, pct_18_23	Percentual de transações em cada faixa horária.
pct_dias_uteis, pct_fins_semana	Percentual de transações em dias úteis e finais de semana.
pct_seg, pct_ter, pct_qua, pct_qui, pct_sex, pct_sab, pct_dom	Percentual de transações por dia da semana.
pct_1_10, pct_11_20, pct_21_31	Percentual de transações por faixas do mês.
pct_estornos, pct_cancelamentos	Percentual de estornos e cancelamentos.
valor_total_pago	Valor total pago em transações.
valor_medio_pago	Valor médio das transações.
valor_total_sem_estorno_cancelamento	Valor total desconsiderando estornos/cancelamentos.
valor_medio_sem_estorno_cancelamento	Valor médio desconsiderando estornos/cancelamentos.
qtd_total	Quantidade total de transações.
qtd_total_sem_estorno_cancelamento	Quantidade de transações válidas (sem estorno/cancelamento).

Tabela 4.6 – Features das arestas G_3

Atributo	**Descrição**
transacao_id	Identificador único da transação.
conta_origem	Identificador único da conta de origem.
conta_destino	Identificador único da conta de destino.
tipo_transacao	Tipo da transação (PIX ou TED).
pct_00_05	Percentual de transações realizadas entre 00h e 05h59min (madrugada).
pct_06_11	Percentual de transações realizadas entre 06h e 11h59min (manhã).
pct_12_17	Percentual de transações realizadas entre 12h e 17h59min (tarde).
pct_18_23	Percentual de transações realizadas entre 18h e 23h59min (noite).
pct_dias_uteis	Percentual de transações realizadas em dias úteis (segunda a sexta-feira).
pct_fins_semana	Percentual de transações realizadas aos fins de semana (sábado e domingo).
pct_seg	Percentual de transações realizadas nas segundas-feiras.
pct_ter	Percentual de transações realizadas nas terças-feiras.
pct_qua	Percentual de transações realizadas nas quartas-feiras.
pct_qui	Percentual de transações realizadas nas quintas-feiras.
pct_sex	Percentual de transações realizadas nas sextas-feiras.
pct_sab	Percentual de transações realizadas nos sábados.
pct_dom	Percentual de transações realizadas nos domingos.
pct_1_10	Percentual de transações realizadas entre os dias 1 e 10 do mês.
pct_11_20	Percentual de transações realizadas entre os dias 11 e 20 do mês.
pct_21_31	Percentual de transações realizadas entre os dias 21 e 31 do mês.
valor_total_pago	Valor total das transações realizadas.
valor_medio_pago	Valor médio das transações realizadas.
qtd_total_transacoes	Quantidade total de transações realizadas.

4.5 Experimentos

4.5.1 Baseline com Modelos de Classificação Tabular

Antes de prosseguir para a modelagem baseada em grafos, foi estabelecida uma linha de base (*baseline*) de desempenho utilizando uma adaptação da metodologia proposta por Souza (2023). A principal modificação em relação ao trabalho original, que lidava com múltiplas classes, consistiu em filtrar o conjunto de dados para um problema de classificação binária, retendo exclusivamente as classes `C` (legítima) e `Fraude`.

Foram avaliados quatro modelos de classificação distintos, com hiperparâmetros definidos como: *Random Forest* (150 árvores, critério de entropia), *XGBoost* (200 estimadores, profundidade máxima de 5), e *TabNet* (treinado por até 100 épocas com parada antecipada).

Uma alteração notável em relação ao trabalho original foi a substituição do modelo baseado em CNN com duas camadas convolucionais (*CNN2Conv*) por uma arquitetura de Rede Neural Convolucional unidimensional (*CNN1D*). Nosso modelo foi composto por uma camada

`Conv1D` (com 32 filtros), seguida por `MaxPooling1D`, uma camada densa e um mecanismo de `Dropout` para regularização, sendo otimizado com Adam e perda de entropia cruzada binária.

O protocolo experimental empregou uma validação cruzada estratificada com 10 folds (`StratifiedShuffleSplit`). Adicionalmente, foram investigados dois cenários distintos para cada modelo: um com os dados em seu desbalanceamento natural e outro com a aplicação da técnica de oversampling SMOTE no conjunto de treinamento para equilibrar as classes. Os resultados obtidos nestes experimentos serviram como um benchmark robusto para comparar e contextualizar o desempenho dos modelos de GNN apresentados na sequência.

4.5.2 Configuração Experimental

Os experimentos foram implementados utilizando dois scripts principais, cada um adaptado para um formato específico de grafo: um para arquivos `.gml` e outro para `.gexf`. A base para a execução incluiu bibliotecas essenciais como *PyTorch* e *PyTorch Geometric* para a definição e treinamento dos modelos, *NetworkX* para o carregamento e a representação inicial dos grafos, *Scikit-learn* para o pré-processamento de dados e cálculo de métricas, além de *Pandas* e *NumPy* para a manipulação eficiente dos atributos. Foram avaliadas três arquiteturas distintas de GNNs, proeminentes na literatura: a *Graph Convolutional Network (GCN)*, que agrega informações dos vizinhos de forma espectral; a *GraphSAGE (Graph Sample and AggreGaTe)*, uma abordagem indutiva que aprende funções de agregação a partir da vizinhança local; e a *Graph Attention Network (GAT)*, que utiliza mecanismos de atenção para ponderar a importância dos vizinhos.

4.5.3 Pré-processamento e Preparação dos Dados

O tratamento dos dados brutos dos grafos foi uma etapa fundamental e seguiu um pipeline bem definido. O processo se inicia com o carregamento do grafo em memória utilizando a biblioteca `NetworkX`. Na sequência, realiza-se a engenharia de features, na qual os atributos de cada nó são processados para formar um vetor de características numéricas. Para o grafo em formato `.gml`, funções customizadas extraem features de strings complexas, enquanto para o formato `.gexf`, atributos já estruturados foram utilizados diretamente. Atributos categóricos foram convertidos em representações numéricas através de *one-hot encoding*, e valores ausentes foram imputados com a média da respectiva coluna. Por fim, o conjunto de nós foi dividido em subconjuntos de treinamento (70%), validação (15%) e teste (15%). Essa divisão foi realizada de forma estratificada para garantir que a proporção de nós fraudulentos e legítimos fosse mantida em todos os subconjuntos.

4.5.4 Treinamento e Otimização dos Modelos

O processo de treinamento foi projetado para encontrar o melhor modelo com base no desempenho no conjunto de validação. Para isso, foi utilizado o otimizador `Adam` com decaimento

de peso (regularização L2) e a função de perda *Negative Log-Likelihood Loss* (`NLLLoss`). Para um ajuste fino do modelo, implementou-se também um agendador, `ReduceLROnPlateau`, que monitorava a métrica F1-Score no conjunto de validação e reduzia a taxa de aprendizado caso não houvesse melhora após um número definido de épocas. Durante as 300 épocas de treinamento, o estado do modelo que alcançou o maior F1-Score no conjunto de validação foi salvo, sendo este"melhor model" o utilizado para a avaliação final.

4.5.5 Tratamento de Desbalanceamento de Classes

Dado que a detecção de fraude é um problema com classes inerentemente desbalanceadas, duas estratégias foram implementadas e poderiam ser selecionadas. A primeira, Ponderação de Classes (*Class Weighting*), atribui um peso maior à classe minoritária (fraude) na função de perda, forçando o modelo a penalizar mais severamente os erros de classificação nesta classe de interesse. A segunda abordagem foi a Subamostragem da Classe Majoritária (*Undersampling*). Nesta técnica, a cada época de treinamento, um lote de dados era formado por todas as amostras da classe minoritária e uma subamostra aleatória da classe majoritária de mesmo tamanho. Isso resulta em um treinamento com lotes perfeitamente balanceados, evitando que o modelo seja enviesado para a classe dominante.

4.5.6 Avaliação de Desempenho

A avaliação final foi realizada sobre o conjunto de teste. Um passo crucial foi a otimização do limiar de classificação. Em vez de usar o valor padrão de 0.5 para converter as probabilidades geradas pelo modelo em uma classificação binária, buscou-se o limiar ótimo no intervalo $[0.01, 1.0]$ que maximizava o F1-Score no conjunto de validação. Utilizando o melhor modelo e o limiar otimizado, o desempenho final foi então aferido com base nas métricas de Acurácia, Precisão, Recall, F1-Score e ROC-AUC. Adicionalmente, para uma análise qualitativa dos erros, uma matriz de confusão foi gerada para o conjunto de teste, permitindo visualizar a distribuição de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.

5 Resultados e discussão

5.1 Caracterização dos Grafos

Os grafos foram construídos a partir das interações financeiras entre contas, conforme descrito no Capítulo 4. As métricas estruturais estão resumidas na Tabela 5.1.

Tabela 5.1 – Principais métricas dos grafos gerados.

Métrica	G_1 **(todas as classes)**	G_2 **(C e Fraude)**	G_3 **(C e Fraude enriquecido)**
Número de nós	124.293	2.640	3.234
Número de arestas	128.827	1.991	3.230

O grafo G_1 (todas as classes) apresenta o maior número de nós e arestas, refletindo a abrangência completa do conjunto de dados, que inclui todas as classes de contas. O grafo G_2 (classes C e Fraude) contém um subconjunto mais restrito, reduzindo a densidade e o número de nós ao focar exclusivamente no cenário binário de classificação. Por fim, o grafo G_3 (classes C e Fraude enriquecido) mantém apenas essas duas classes, mas incorpora atributos adicionais, resultando em um número maior de nós e arestas em relação a G_2.

5.1.1 Visualização dos Grafos

A visualização dos grafos permite identificar padrões relevantes nas relações entre contas e documentos bancários. Observa-se uma diferença significativa entre os nós considerados totalmente legítimos (classe A) e os nós classificados como fraudulentos. Os nós de classe A, como pode ser visualizado na Figura 5.4, possuem muitas transações com diversos outros nós, indicando um alto grau de conectividade dentro da rede. Em contraste, os nós fraudulentos, representados em vermelho nas Figuras 5.1 e 5.3, apresentam poucas conexões, concentrando-se em interações mais restritas. À medida que o nível de legitimidade diminui de A para E, conforme ilustrado nas Figuras 5.5 a 5.8, a densidade de transações também diminui, tornando-se cada vez mais semelhante ao comportamento dos nós fraudulentos.

As Figuras 5.1 e 5.2 foram plotadas para análise visual das relações entre os documentos bancários e suas transações. O primeiro corresponde ao grafo G_1, contendo todas as classes de contas, enquanto o segundo representa o grafo G_3, restrito apenas às classes C e Fraude, mas enriquecido com atributos adicionais. A visualização evidencia a densidade de conexões e a distribuição das classes nos subgrafos, permitindo observar padrões estruturais que poderão ser explorados nos experimentos de detecção de fraude.

As Figuras 5.1 e 5.2 mostram a representação geral do grafo, enquanto a Figura 5.3 apresenta um subgrafo com um nó aleatório e seus 100 vizinhos mais próximos, permitindo

observar a distribuição local das conexões. Já as Figuras 5.4 a 5.8 detalham subgrafos contendo combinações de nós fraudulentos e nós de diferentes classes legítimas, evidenciando como a conectividade diminui à medida que os nós se aproximam de perfis suspeitos.

Essas observações fornecem uma visão inicial sobre padrões de comportamento e conectividade, servindo como base para análises subsequentes e identificação de potenciais sinais de fraude nas interações financeiras.

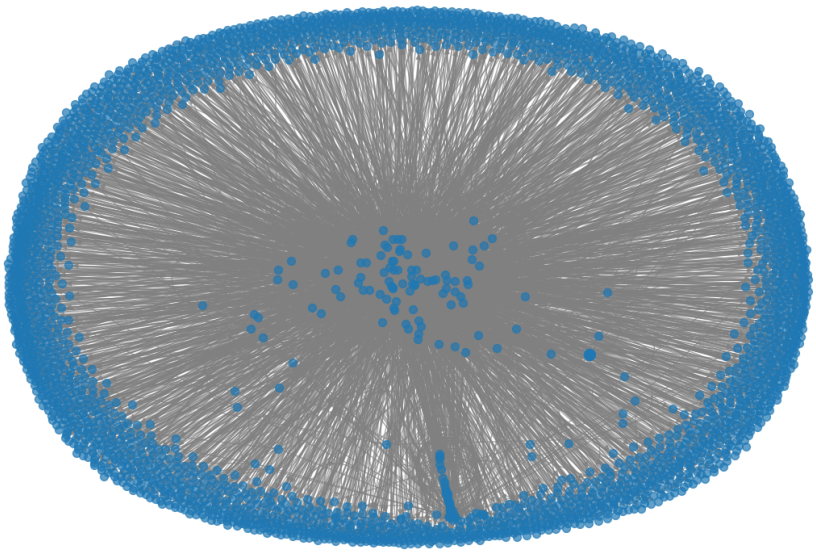

Figura 5.1 – Representação parcial do grafo G_1 construído para modelar as relações entre documentos bancários e transações financeiras.

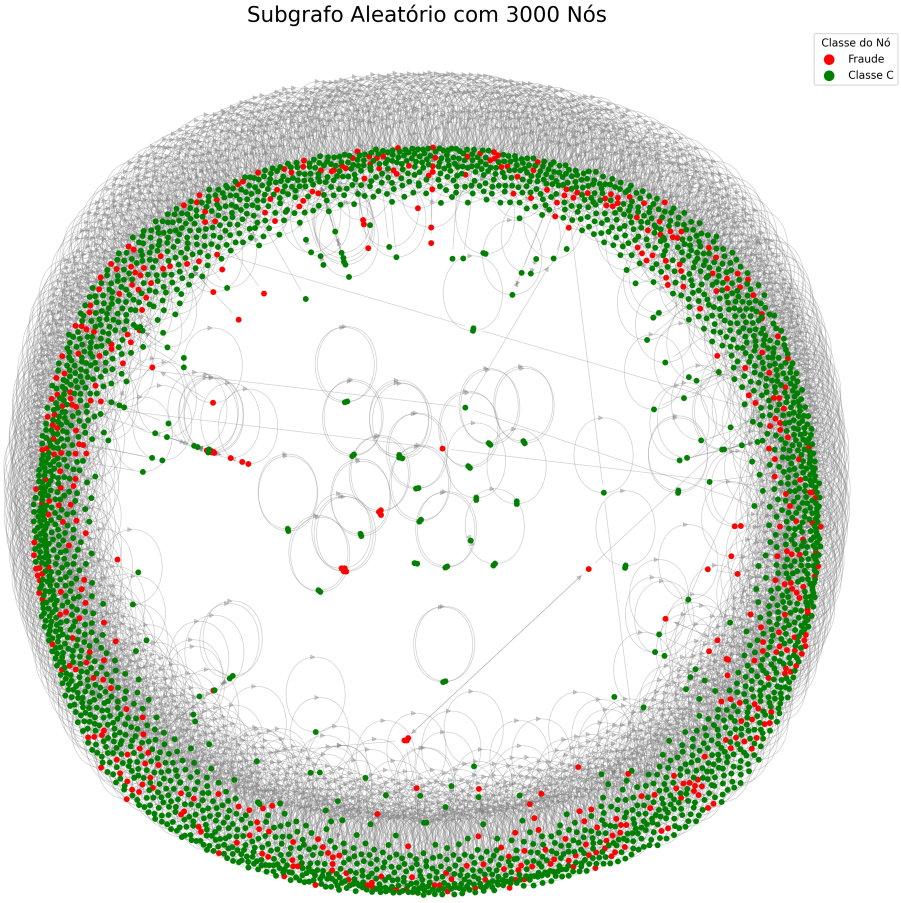

Figura 5.2 – Representação parcial do grafo G_3 com as classes C (verde) e Fraude (vermelho).

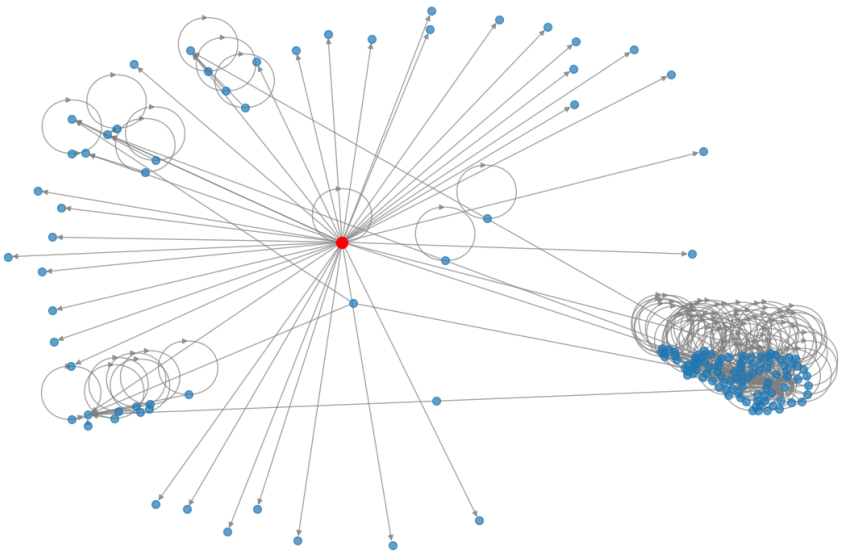

Figura 5.3 – Subgrafo com um nó aleatório selecionado (vermelho) e seus 100 vizinhos mais próximos.

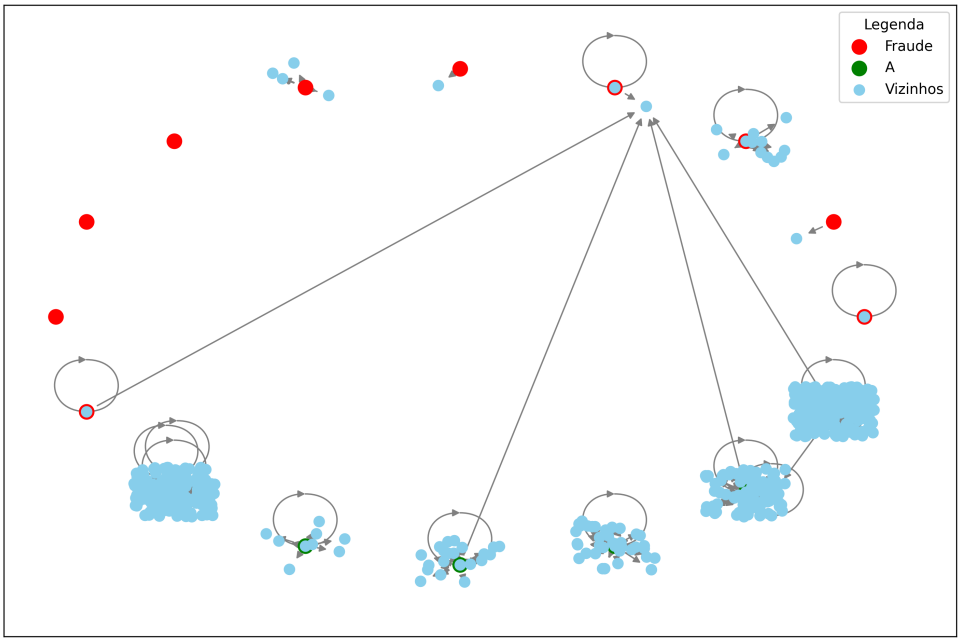

Figura 5.4 – Subgrafo contendo 10 nós classificados como “Fraude” e 6 nós de nível “A” com seus vizinhos.

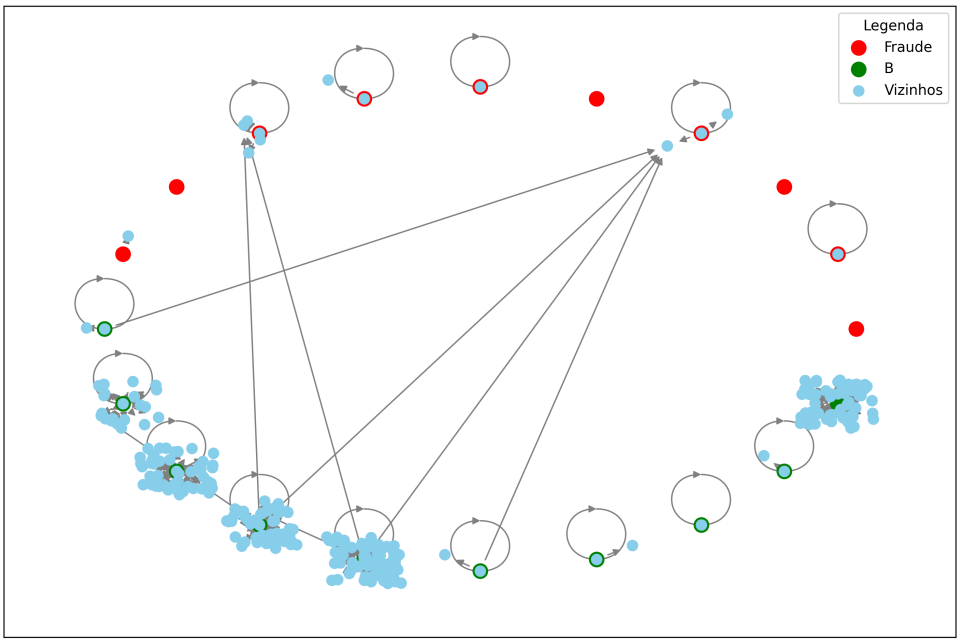

Figura 5.5 – Subgrafo contendo 10 nós classificados como “Fraude” e 10 nós de nível “B” com seus vizinhos.

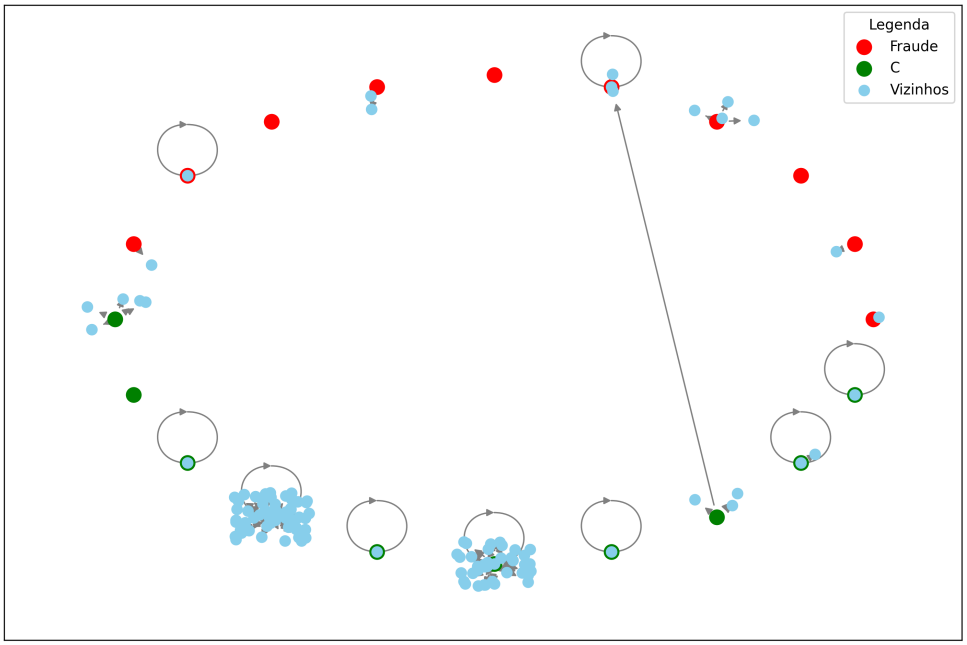

Figura 5.6 – Subgrafo contendo 10 nós classificados como “Fraude” e 10 nós de nível “C” com seus vizinhos.

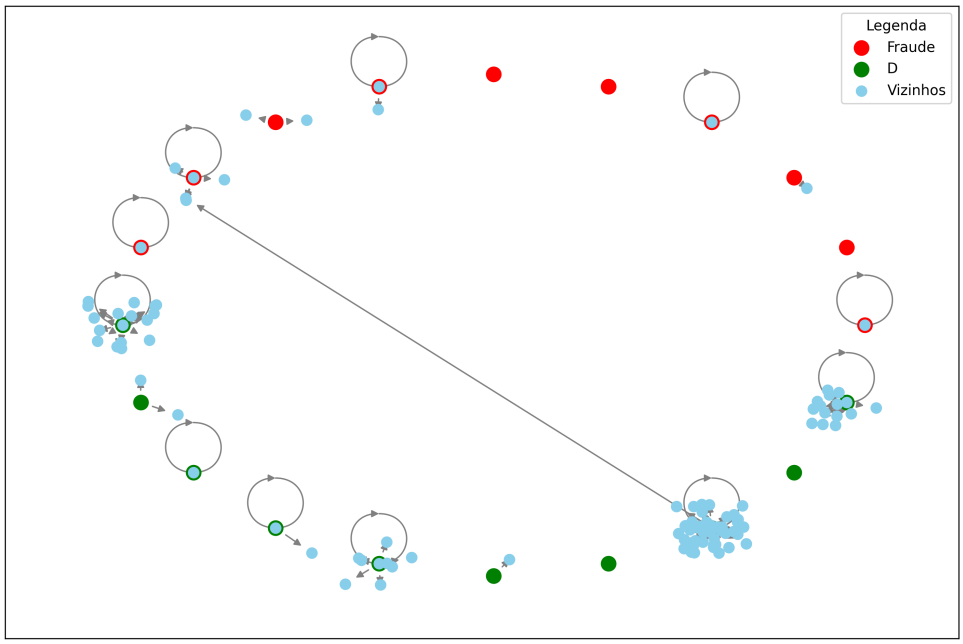

Figura 5.7 – Subgrafo contendo 10 nós classificados como “Fraude” e 10 nós de nível “D” com seus vizinhos.

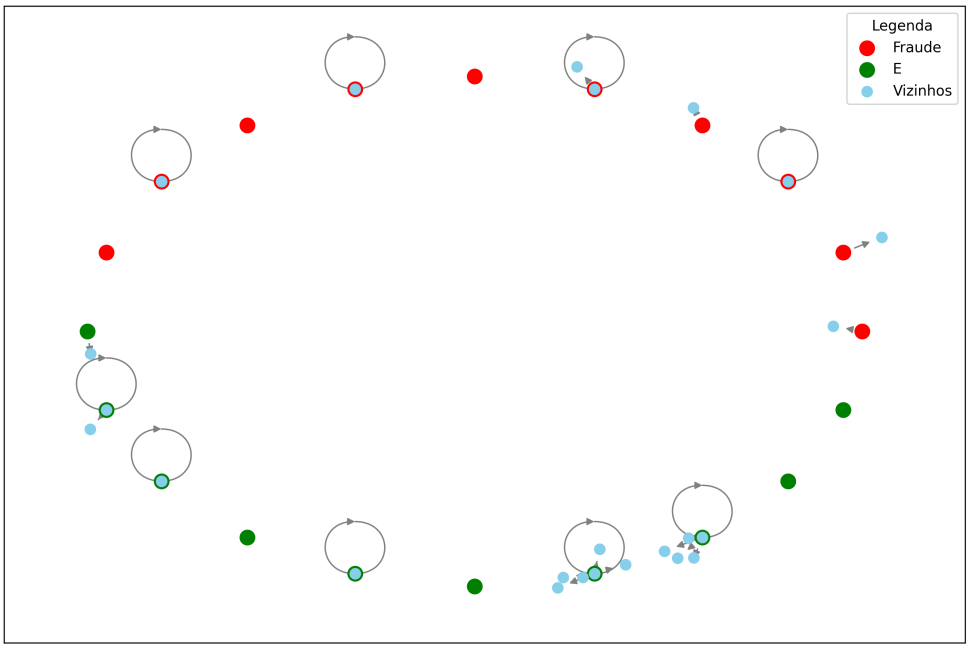

Figura 5.8 – Subgrafo contendo 10 nós classificados como “Fraude” e 10 nós de nível “E” com seus vizinhos.

5.1.2 Análise das imagens e resultados

A visualização dos grafos construídos possibilita compreender melhor os padrões de interação entre documentos bancários e transações financeiras, além de fornecer indícios relevantes sobre o comportamento das contas classificadas como legítimas e fraudulentas.

A análise das imagens mostra que as contas legítimas, especialmente aquelas associadas ao nível A, possuem um padrão de conexões mais diversificado, estabelecendo transações com múltiplos vizinhos. Esse comportamento sugere uma rede de interações mais ampla e heterogênea, compatível com o funcionamento esperado de contas comuns, que tendem a realizar diferentes tipos de operações financeiras ao longo do tempo. Em contraste, os nós classificados como fraudulentos exibem um comportamento estruturalmente distinto: suas conexões são mais restritas, com menor número de vizinhos, indicando um padrão de utilização limitado e possivelmente orquestrado para fins ilícitos.

Outro aspecto relevante identificado é a forma como os nós intermediários, situados em níveis mais próximos da fraude, apresentam características mistas. Embora não sejam diretamente classificados como fraudulentos, o padrão de conexões desses nós se aproxima gradualmente daquele observado nas contas de fraude, reforçando a hipótese de que esses níveis podem representar perfis de risco intermediário ou potenciais alvos de monitoramento mais atento.

Além da análise estrutural, também foram observadas características de atributos associados às contas. Um dado significativo é o uso do domínio de e-mail: apenas 56% das contas gerais utilizam o domínio "GMAIL", enquanto entre as contas fraudulentas esse número sobe para 80%. Essa discrepância sugere uma preferência marcante por esse provedor em esquemas de fraude, possivelmente relacionada à facilidade de criação de contas e à ampla aceitação em cadastros digitais. Outro fator de destaque é que todos os documentos classificados como fraudulentos possuem exatamente uma conta associada, diferentemente de perfis legítimos, que podem estar vinculados a múltiplas contas.

No que diz respeito ao comportamento transacional, foi constatado que apenas 8% das contas fraudulentas realizaram operações tanto de PIX quanto de TED. A esmagadora maioria (92%) utilizou exclusivamente um único tipo de transação, indicando um padrão de comportamento mais restrito e potencialmente automatizado. Já nas contas legítimas, observa-se maior diversidade no uso dos diferentes meios de transferência.

De modo geral, a análise das representações gráficas e dos atributos observados confirma a existência de diferenças estruturais e comportamentais claras entre contas legítimas e fraudulentas. Essas distinções, tanto no nível de conectividade quanto no uso de atributos como domínio de e-mail e diversidade de transações, reforçam a relevância da abordagem baseada em grafos para a detecção de fraude.

5.2 Resultados dos Modelos

Os resultados foram organizados em duas tabelas para melhor compreensão. A Tabela 5.2 apresenta os experimentos realizados sobre o grafo G_1, contendo todas as classes, em comparação com os resultados tabulares com todas as classes. Já a Tabela 5.3 mostra os resultados referentes aos grafos filtrados para as classes C e Fraude, G_2 e G_3, comparados aos experimentos tabulares

filtrando as mesmas classes. Em ambas, são apresentadas métricas de sensibilidade, especificidade, taxa de falsos positivos (TFP), precisão e F-score, considerando média e desvio padrão a partir de validação cruzada de 10 folds para os modelos tabulares.

Tabela 5.2 – Média em (%) e desvio padrão dos resultados para a classe Fraude para os diferentes modelos e estratégias no grafo geral.

Modelo	**Sensibilidade**	**Especificidade**	**TFP**	**Precisão**	**F-score**
Random Forest	$50,19 \pm 1,91$	$97,12 \pm 0,23$	$2,88 \pm 0,23$	$74,63 \pm 1,64$	$60,00 \pm 1,61$
RF + SMOTE	$62,38 \pm 1,71$	$93,67 \pm 0,36$	$6,33 \pm 0,36$	$62,50 \pm 1,40$	$62,43 \pm 1,27$
XGBoost	$57,58 \pm 1,10$	$96,68 \pm 0,26$	$3,32 \pm 0,26$	$74,60 \pm 1,66$	$64,99 \pm 1,20$
XGBoost + SMOTE	$68,96 \pm 2,22$	$93,50 \pm 0,26$	$6,50 \pm 0,26$	$64,18 \pm 1,42$	$66,48 \pm 1,73$
TabNet	$0,00 \pm 0,00$	$100,00 \pm 0,01$	$0,00 \pm 0,01$	$0,00 \pm 0,00$	$0,00 \pm 0,00$
TabNet + SMOTE	$53,90 \pm 26,62$	$84,18 \pm 9,58$	$15,83 \pm 9,58$	$52,04 \pm 18,74$	$54,60 \pm 19,83$
CNN	$17,20 \pm 17,30$	$98,70 \pm 1,35$	$1,30 \pm 1,35$	$0,00 \pm 0,00$	$0,00 \pm 0,00$
CNN + SMOTE	$70,79 \pm 3,61$	$82,90 \pm 1,32$	$17,10 \pm 1,32$	$41,17 \pm 0,92$	$52,01 \pm 0,82$
GCN (default)	$0,00$	$100,00$	$0,00$	$0,00$	$0,00$
GCN (class weights)	$50,00$	$99,49$	$0,51$	$5,41$	$9,76$
GCN (undersample)	$50,00$	$99,24$	$0,76$	$3,70$	$6,90$
GraphSAGE (default)	$50,00$	$99,74$	$0,26$	$10,00$	$16,67$
GraphSAGE (class weights)	$50,00$	$99,65$	$0,35$	$7,69$	$13,33$
GraphSAGE (undersample)	$50,00$	$99,65$	$0,35$	$7,69$	$13,33$
GAT (default)	$0,00$	$100,00$	$0,00$	$0,00$	$0,00$
GAT (class weights)	$75,00$	$98,91$	$1,09$	$3,90$	$7,41$
GAT (undersample)	$0,00$	$99,68$	$0,32$	$0,00$	$0,00$

5.3 Discussão dos Resultados

Os resultados obtidos evidenciam que a escolha da estrutura do grafo e dos atributos considerados influencia de maneira significativa o desempenho dos modelos. A comparação direta entre o cenário GraphSAGE com undersampling em G_2 e o cenário GraphSAGE com undersampling em G_3 mostra que, mesmo sem alterar o modelo ou os hiperparâmetros, apenas a modificação na estrutura do grafo levou a ganhos relevantes de desempenho. Esse comportamento indica que a representação das relações financeiras desempenha papel central na detecção de fraude, pois um grafo mais bem estruturado captura melhor os padrões de interações suspeitas entre contas e documentos.

Ao analisar os modelos, percebe-se que o GraphSAGE se destacou em relação ao GCN e ao GAT nos experimentos conduzidos. Esse resultado sugere que a capacidade do GraphSAGE de amostrar vizinhos e agregar informações de forma adaptativa o torna mais adequado para lidar com grafos financeiros esparsos e heterogêneos, onde diferentes contas possuem graus de conexão muito variados. Já o GCN, embora consistente, mostrou desempenho inferior, possivelmente devido à sua limitação em capturar variações locais de vizinhança. O GAT, por sua vez, apresentou resultados intermediários, beneficiando-se da atenção para priorizar conexões mais relevantes, mas não superando a robustez do GraphSAGE na configuração utilizada.

Tabela 5.3 – Média em (%) dos resultados para os diferentes modelos e estratégias.

Modelo	Sensibilidade	Especificidade	TFP	Precisão	F-score
Random Forest	$86,54 \pm 1,95$	$98,68 \pm 0,34$	$1,32 \pm 0,34$	$\mathbf{93,99} \pm 1,35$	$90,09 \pm 1,41$
RF + SMOTE	$90,86 \pm 1,63$	$97,96 \pm 0,53$	$2,04 \pm 0,53$	$91,41 \pm 2,32$	$91,12 \pm 1,52$
XGBoost	$91,17 \pm 1,36$	$98,46 \pm 0,69$	$1,54 \pm 0,69$	$93,42 \pm 2,86$	$92,27 \pm 1,99$
XGBoost + SMOTE	$\mathbf{92,35} \pm 1,77$	$98,21 \pm 0,56$	$1,79 \pm 0,56$	$92,50 \pm 2,40$	$92,41 \pm 1,58$
TabNet	$0,00 \pm 0,00$	$100,00 \pm 0,00$	$0,00 \pm 0,00$	$0,00 \pm 0,00$	$0,00 \pm 0,00$
TabNet + SMOTE	$80,80 \pm 9,11$	$94,43 \pm 2,56$	$5,57 \pm 2,56$	$78,82 \pm 7,68$	$79,05 \pm 2,87$
CNN 1D	$81,24 \pm 5,28$	$96,69 \pm 1,25$	$3,31 \pm 1,25$	$85,83 \pm 4,71$	$83,23 \pm 2,05$
CNN 1D + SMOTE	$82,10 \pm 3,23$	$96,90 \pm 0,88$	$3,10 \pm 0,88$	$86,41 \pm 3,31$	$84,15 \pm 2,49$
GCN (G2, default)	59,38	97,53	2,47	67,86	63,33
GCN (G2, class weights)	53,12	98,08	1,92	70,83	60,71
GCN (G2, undersampling)	68,75	98,08	1,92	75,86	72,13
GCN (G3, default)	76,62	94,87	5,13	73,75	75,16
GCN (G3, class weights)	**85,71**	91,69	8,31	66,00	74,58
GCN (G3, undersampling)	80,52	95,11	4,89	75,61	**77,99**
GraphSAGE (G2, default)	31,25	98,63	1,37	66,67	42,55
GraphSAGE (G2, class weights)	59,38	96,15	3,85	57,58	58,46
GraphSAGE (G2, undersampling)	40,62	97,80	2,20	61,90	49,06
GraphSAGE (G3, default)	72,73	96,82	3,18	81,16	76,71
GraphSAGE (G3, class weights)	80,52	95,84	4,16	78,48	79,49
GraphSAGE (G3, undersampling)	79,22	97,07	2,93	**83,56**	81,33
GAT (G2, default)	53,12	92,03	7,97	36,96	43,59
GAT (G2, class weights)	43,75	98,63	1,37	73,68	54,90
GAT (G2, undersampling)	43,75	98,63	1,37	73,68	54,90
GAT (G3, default)	88,31	89,24	10,76	60,71	71,96
GAT (G3, class weights)	88,31	87,78	12,22	57,63	69,74
GAT (G3, undersampling)	88,31	88,51	11,49	59,13	70,83

Outro ponto importante está relacionado ao impacto das estratégias de balanceamento de classes. O undersampling mostrou-se a mais eficaz entre as abordagens testadas, principalmente em cenários com forte desbalanceamento entre contas legítimas e fraudulentas. Entretanto, observou-se que o undersampling pode gerar maior variabilidade entre execuções, dependendo do formato do grafo adotado, reforçando a necessidade de ajustes mais finos. Em contrapartida, as estratégias de class weights e undersampling, embora úteis, não apresentaram a mesma capacidade de reduzir o viés do modelo frente às classes minoritárias.

Do ponto de vista das métricas, nota-se que os resultados alcançados com grafos ainda não superaram aqueles obtidos nos treinamentos tabulares tradicionais, mas mostraram-se promissores. A principal diferença está no recall: enquanto os modelos tabulares já oferecem valores mais elevados para essa métrica, os modelos baseados em grafos ainda precisam de refinamento para alcançar desempenho equivalente. Por outro lado, a especificidade se manteve em níveis satisfatórios, mostrando que o modelo consegue identificar bem as contas legítimas. Isso reforça a hipótese de que, com melhorias adicionais na estrutura do grafo e no ajuste de hiperparâmetros, é possível alcançar desempenhos superiores aos modelos tabulares, especialmente em tarefas que envolvem relações complexas como as transações financeiras.

Portanto, a análise indica que os modelos baseados em grafos estão no caminho certo para

lidar com o problema de detecção de fraude. O uso de informações relacionais entre documentos e transações enriquece a representação dos dados, permitindo capturar padrões que modelos tabulares não conseguem explorar de forma direta. Apesar das limitações atuais, principalmente no recall, os resultados mostram que há espaço para avanços consideráveis ao se refinar a modelagem do grafo e ajustar parâmetros de treinamento. Assim, a discussão evidencia não apenas os ganhos já obtidos, mas também o potencial de evolução dessa abordagem em comparação com métodos tradicionais.

As figuras a seguir mostram a comparação entre o resultado GraphSAGE com undersampling em G_2, Figura 5.9, e o resultado GraphSAGE com undersampling em G_3, Figura 5.10.

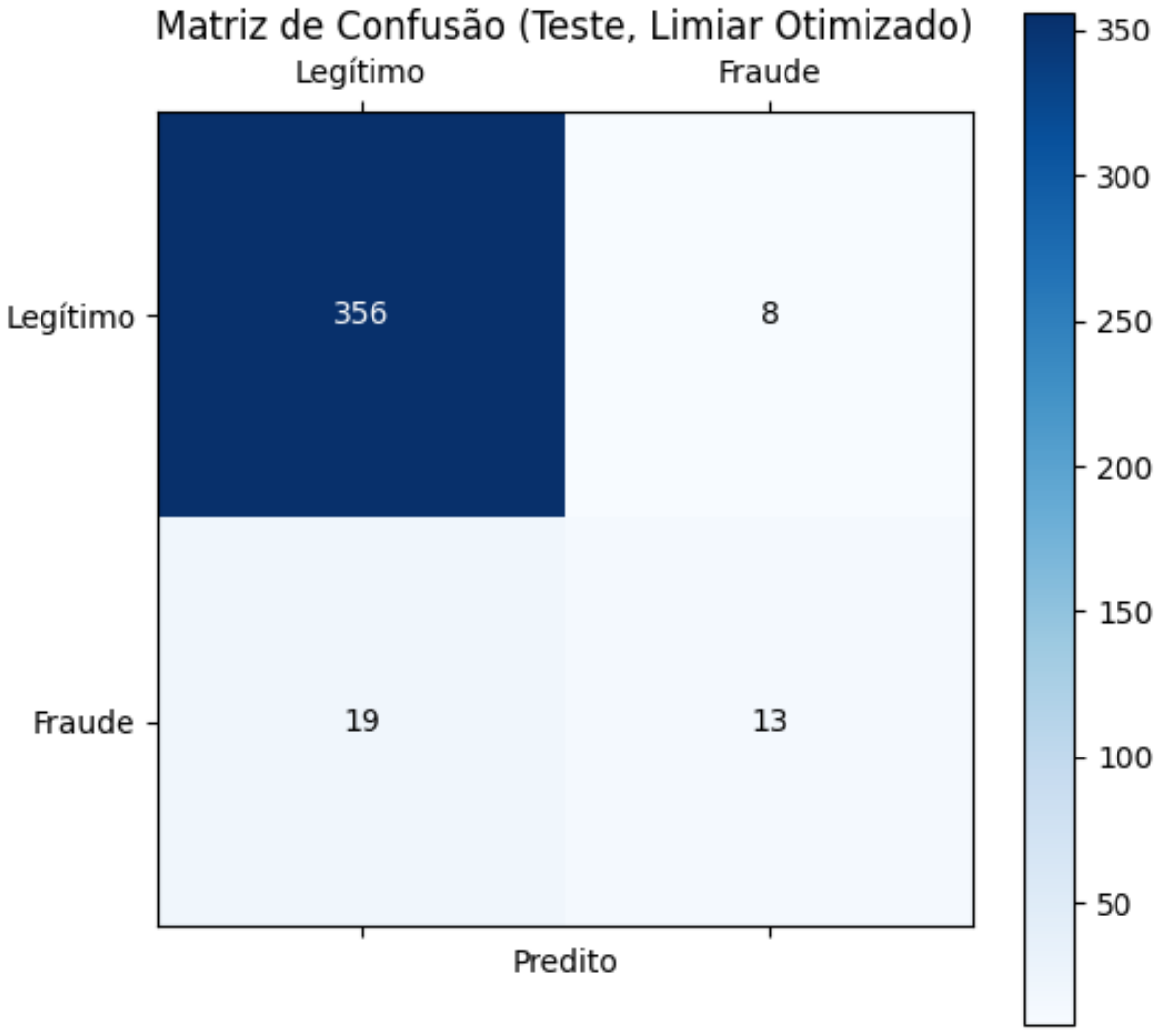

Figura 5.9 – Matriz de confusão GraphSAGE com undersample em G_2 (classes C e Fraude).

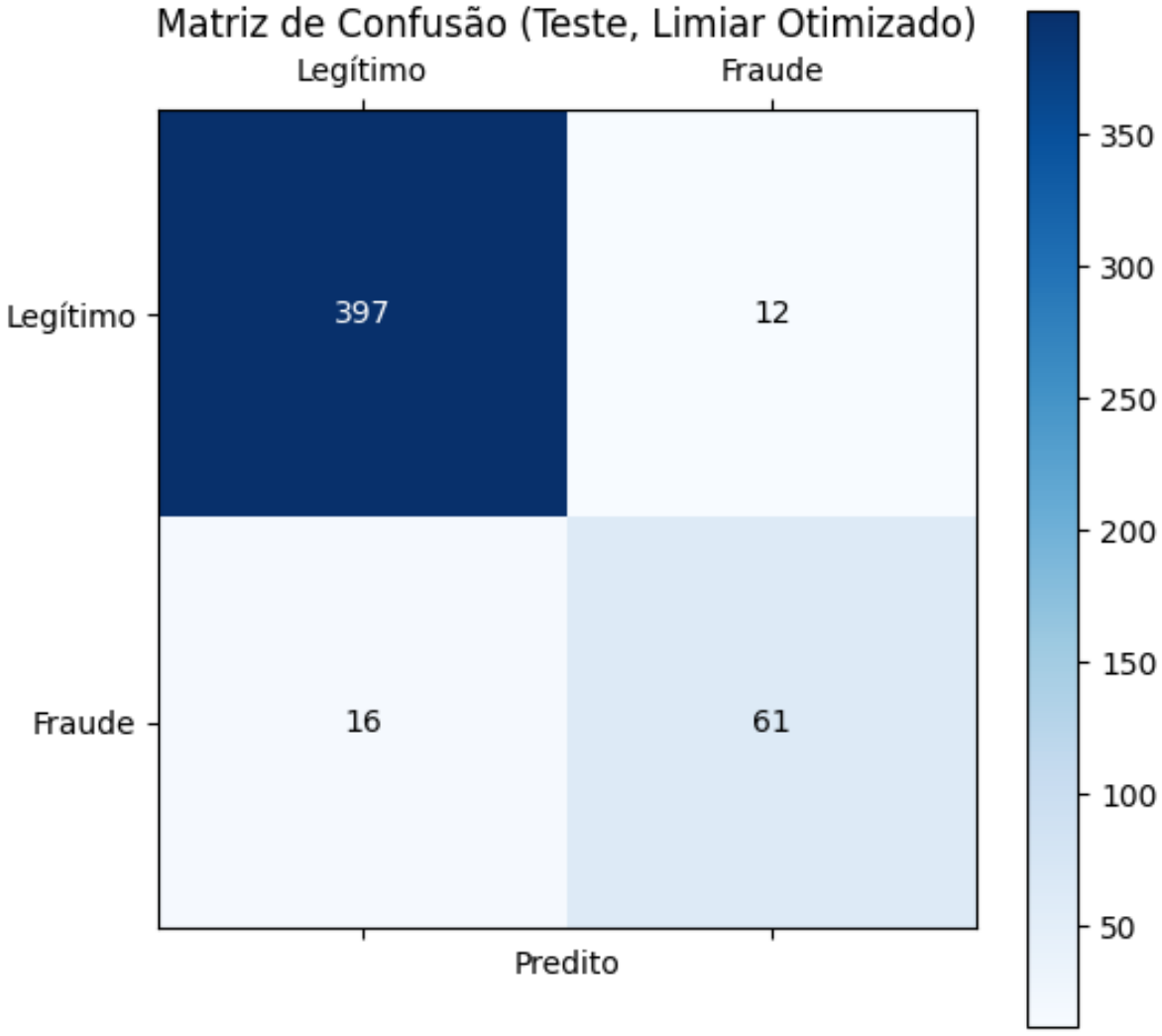

Figura 5.10 – Matriz de confusão GraphSAGE com undersample em G_3 (classes C e Fraude).

6 Considerações Finais

Este trabalho apresentou a construção de diferentes representações em grafos a partir de dados financeiros, explorando suas propriedades estruturais e a aplicação de modelos de aprendizado de máquina baseados em redes neurais em grafos. A análise envolveu tanto grafos abrangendo todas as classes de contas quanto versões filtradas, contendo apenas as classes C e Fraude. Foram realizadas visualizações, cálculo de métricas estruturais e experimentos de classificação, comparando o desempenho de abordagens tabulares tradicionais com modelos de aprendizado em grafos.

Os resultados evidenciaram que, no cenário geral, os modelos tabulares ainda apresentam desempenho superior, especialmente em termos de precisão e F-score. Destaca-se que, apenas modificando a estrutura e os atributos do grafo, sem alterar o código de treinamento, foi possível observar ganhos expressivos em desempenho, reforçando a importância de escolhas adequadas na modelagem do grafo.

Esses achados confirmam o potencial da abordagem baseada em grafos para a detecção de fraudes financeiras. Apesar de ainda não superarem os resultados obtidos pelos métodos tabulares tradicionais, os modelos de grafos mostraram ser capazes de capturar relações estruturais que não estão explicitamente disponíveis nas abordagens convencionais, o que os torna promissores como alternativa ou complemento em cenários reais.

Como trabalhos futuros, acredita-se que há espaço para avanços significativos com a melhoria da estrutura do grafo, por meio da incorporação de atributos adicionais, filtragem mais criteriosa de conexões e estratégias de agregação mais sofisticadas. Além disso, a exploração de diferentes hiperparâmetros de treinamento, arquiteturas mais recentes de Graph Neural Networks e técnicas avançadas de balanceamento de classes podem contribuir para ganhos adicionais de desempenho. Dessa forma, espera-se que, com tais ajustes, os modelos baseados em grafos não apenas se tornem competitivos, mas possam até mesmo superar os resultados de abordagens tabulares tradicionais, consolidando-se como uma ferramenta eficaz na detecção de fraudes financeiras.

Referências

AL-HASHEDI, K. G.; MAGALINGAM, P. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review*, Elsevier, v. 40, p. 100402, 2021.

ARIK, S. Ö.; PFISTER, T. Tabnet: Attentive interpretable tabular learning. In: *Proceedings of the AAAI conference on artificial intelligence*. [S.l.: s.n.], 2021. v. 35, n. 8, p. 6679–6687.

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

BONDY, J. A.; MURTY, U. S. R. *Graph theory*. [S.l.]: Springer Publishing Company, Incorporated, 2008.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002.

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.; ZOU, Y.; XIANG, S.; JIANG, C. Graph neural networks for financial fraud detection: A review. *arXiv preprint arXiv:2411.05815*, 2024.

FERNÁNDEZ, A.; GARCIA, S.; HERRERA, F.; CHAWLA, N. V. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, v. 61, p. 863–905, 2018.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, Springer, v. 36, n. 4, p. 193–202, 1980.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; BENGIO, Y. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1.

GRANDINI, M.; BAGLI, E.; VISANI, G. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

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

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

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, MIT Press, v. 1, n. 4, p. 541–551, 1989.

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

REN, L.; HU, R.; LI, D.; LIU, Y.; WU, J.; ZANG, Y.; HU, W. Dynamic graph neural network-based fraud detectors against collaborative fraudsters. *Knowledge-Based Systems*, Elsevier, v. 278, p. 110888, 2023.

SCARSELLI, F.; GORI, M.; TSOI, A. C.; HAGENBUCHNER, M.; MONFARDINI, G. The graph neural network model. *IEEE transactions on neural networks*, IEEE, v. 20, n. 1, p. 61–80, 2008.

ŠILJAK, D. Dynamic graphs. *Nonlinear Analysis: Hybrid Systems*, Elsevier, v. 2, n. 2, p. 544–567, 2008.

SOUZA, A. O. Detecção de fraudes financeiras em contas digitais: explorando abordagens hierárquicas e técnicas de aprendizado profundo. *125 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, 2023.*, 2023.

VELIČKOVIĆ, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIO, P.; BENGIO, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.