ÍTALO DELLA GARZA SILVA

REDES NEURAIS DE GRAFOS APLICADAS À DETECÇÃO DE LAVAGEM DE DINHEIRO

LAVRAS – MG

2023

ÍTALO DELLA GARZA SILVA

REDES NEURAIS DE GRAFOS APLICADAS À DETECÇÃO DE LAVAGEM DE DINHEIRO

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Sistemas de Computação, para a obtenção do título de Mestre.

Prof. DSc. Luiz Henrique Andrade Correia
Orientador

Prof. DSc. Erick Galani Maziero
Coorientador

LAVRAS – MG
2023

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo próprio autor.

Silva, Ítalo Della Garza

Redes neurais de grafos aplicadas à detecção de lavagem de dinheiro / Ítalo Della Garza Silva. – 2023.

63 p. : il.

Orientador(a): Luiz Henrique Andrade Correia.

Corientador(a): Erick Galani Maziero.

Dissertação (mestrado acadêmico) – Universidade Federal de Lavras, 2023.

Bibliografia.

1. Aprendizagem Profunda. 2. Crimes Financeiros. 3. Aprendizagem de Máquina. I. Correia, Luiz Henrique Andrade. II. Maziero, Erick Galani. III. Título.

ÍTALO DELLA GARZA SILVA

REDES NEURAIS DE GRAFOS APLICADAS À DETECÇÃO DE LAVAGEM DE DINHEIRO
GRAPH NEURAL NETWORKS APPLIED TO MONEY LAUNDERING DETECTION

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Sistemas de Computação, para a obtenção do título de Mestre.

APROVADA em 14 de Abril de 2023.

Prof. DSc. Luiz Henrique Andrade Correia	UFLA
Prof. DSc. Erick Galani Maziero	UFLA
Prof. DSc. Mayron César de Oliveira Moreira	UFLA
Prof. DSc. Daniel Fernandes Macedo	UFMG

Prof. DSc. Luiz Henrique Andrade Correia
Orientador

Prof. DSc. Erick Galani Maziero
Co-Orientador

LAVRAS – MG
2023

AGRADECIMENTOS

Novamente, tenho de agradecer primeiramente a Deus, que me iluminou e me deu forças para continuar nessa jornada, à minha família, que tanto me incentivou e me apoiou nos momentos mais difíceis, e aos meus orientadores, por me mostrarem o caminho a ser seguido. Descobri, durante esses dois anos, que a jornada do mestrado não é fácil. Porém, aqueles que estão ao seu lado podem fazer toda a diferença. Agradeço também à FAPEMIG, CAPES e à CNPq.

RESUMO

Crimes financeiros existem em todos os países do mundo, e um dos mais recorrentes é a Lavagem de Dinheiro. Esta é capaz de causar enormes prejuízos, tanto financeiros quanto relacionados à reputação, às empresas e agências governamentais envolvidas no processo. Atualmente, tais organizações utilizam algoritmos envolvendo técnicas de Inteligência Artificial para detectar transações financeiras suspeitas de Lavagem de Dinheiro. No entanto, tais métodos geram várias transações suspeitas, sendo frequentemente necessária uma posterior avaliação humana para a confirmação da suspeição, aumentando os custos financeiros e o tempo gasto. A literatura tem apresentado métodos alternativos mais robustos para resolver esses problemas, frequentemente envolvendo técnicas de *Machine Learning*. Nesse cenário, uma vez que é possível representar transações financeiras por meio de grafos, métodos envolvendo Redes Neurais de Grafos (GNN) têm se mostrado uma solução promissora para a detecção de transações suspeitas de Lavagem de Dinheiro. É possível representar transações tanto como vértices quanto como arestas através de grafos, impactando na escolha do modelo de GNN para o processo de detecção. Este trabalho avalia as conhecidas arquiteturas de Rede Convolucional de Grafos (GCN) e Skip-GCN, assim como a recente Rede Neural de Nós e Arestas (NENN), para a solução do problema de detecção automatizada de Lavagem de Dinheiro, testando-as em transações financeiras geradas pelo simulador AMLSim. Foram geradas quatro bases de dados para testar a influência do desbalanceamento de classe sobre a qualidade da detecção: AMLSim 1/3, AMLSim 1/5, AMLSim 1/10, e AMLSim 1/20, com taxas de desbalanceamento de 3, 5, 10, e 20, respectivamente. Inicialmente, os modelos de GNN foram testados sobre todos os conjuntos de dados, com a classificação feita tanto por Softmax quanto por XGBoost. Em seguida, foi realizada uma otimização de hiperparâmetros nos modelos sobre a base de dados AMLSim 1/20, visando melhorar os resultados para a taxa de desbalanceamento mais alta. Foi também avaliado o aumento de precisão através da classificação feita pela combinação Softmax + XGBoost disposta em cascata de forma que o classificador seguinte confirmasse se a detecção de suspeição por parte do anterior. Nos resultados iniciais, embora os modelos GCN e Skip-GCN tenham se saído melhor no geral, a combinação NENN + XGBoost atingiu resultados melhores para o conjunto AMLSim 1/20, com uma macro-F1 de 86,69%, indicando a influência positiva da representação das transações como arestas do grafo. Após a otimização de hiperparâmetros, todos os modelos melhoraram seus resultados, sendo que a combinação com a F1 mais alta (88,77%) passou a ser Skip-GCN + Softmax. Com o uso da combinação de classificadores Softmax + XGBoost, o modelo Skip-GCN obteve a melhor F1 (88,90%).

Palavras-chave: Aprendizagem Profunda. Crimes Financeiros. Aprendizagem de Máquina.

ABSTRACT

Financial crimes exist in all world countries, and one of the most recurrent ones is Money Laundering. It is capable of causing enormous damage, both financial and reputational, to the companies and government agencies involved in the process. Currently, such organizations use algorithms involving Artificial Intelligence techniques to detect suspicious Money Laundering financial transactions. However, such methods generate many suspicious transactions, often requiring a posterior human evaluation to confirm the suspicion, increasing financial costs and time spent. The literature has presented more robust alternative methods to solve these problems, often involving Machine Learning techniques. In this scenario, since it is possible to represent financial transactions through graphs, methods involving Graph Neural Networks (GNN) have proven to be a promising solution for detecting suspicious Money Laundering transactions. It is possible to represent transactions both as vertices and edges through graphs, impacting the choice of the GNN model for the detection process. This study evaluates the well-known Convolutional Graph Network (GCN) and Skip-GCN, as well as the recent Node and Edge Neural Network (NENN), for the Money Laundering automated detection problem solution, testing them in financial transactions generated by the AMLSim simulator. Four databases were generated to test the influence of class imbalance on detection quality: AMLSim 1/3, AMLSim 1/5, AMLSim 1/10, and AMLSim 1/20, with imbalance rates of 3, 5, 10, and 20, respectively. Initially, the GNN models were tested on all datasets, with the classification done by Softmax and XGBoost. Then, a hyperparameter optimization was performed on the models on the AMLSim 1/20 database, aiming to improve the results for the highest imbalance rate. The precision increase through classification performed by Softmax + XGBoost combination arranged in cascade was also evaluated so that the next classifier confirms the detection of suspicion by the previous one. In the initial results, although the GCN and Skip-GCN models performed better overall, the combination NENN + XGBoost achieved better results for the AMLSim 1/20 set, with a macro-F1 of 86.69%, indicating the positive influence of the representation of transactions as edges of the graph. After hyperparameter optimization, all models improved their results, and the combination with the highest F1 (88.77%) became Skip-GCN + Softmax. Using the combination of Softmax + XGBoost classifiers, the Skip-GCN model obtained the best F1 (88.90%).

Keywords: Deep Learning. Financial Crimes. Machine Learning.

SUMÁRIO

1 INTRODUÇÃO

Crimes financeiros ocorrem em todos os países do mundo, e o dever de identificá-los e puni-los cabe a determinadas agências federais. Essas o fazem por meio de uma série de processos investigativos, geralmente bem estabelecidos. No Brasil, os possíveis crimes financeiros estão listados na Lei nº 7.492/1986 do Direito Civil ou Comercial (BRASIL, 1986) que "define os crimes contra o sistema financeiro nacional, e dá outras providências". Pode-se definir um crime financeiro como aquele praticado por uma entidade (i.e., pessoa física ou jurídica) através do mercado financeiro, objetivando ganho próprio às custas de outras entidades daquele negócio (DREŻEWSKI; SEPIELAK; FILIPKOWSKI, 2012). Outra definição de crime financeiro é dada por Gottschalk (2010) e descreve o crime financeiro como qualquer crime que não envolva o uso de violência e produza uma perda financeira, incluindo uma série de atividades ilegais como sonegação de impostos e lavagem de dinheiro, sendo o crime financeiro um tipo de abuso financeiro.

Gottschalk (2010) lista as principais atividades que se enquadram na categoria. A Fraude Financeira pode ser descrita como um engodo, envolvendo documentos financeiros, realizado com o intuito de se obter lucro em detrimento do patrimônio alheio, incluindo fraude sobre cheque, cartão de crédito, esquema de pirâmide, entre outros. A Sonegação de Impostos ocorre quando uma entidade pagadora de impostos omite dados sobre seus rendimentos e economias, visando diminuir ou até mesmo anular o valor do imposto a ser pago. O Contorno de Restrições Cambiais engloba qualquer atividade que envolva burlar as leis cambiais vigentes com o intuito se de se obter benefícios ilegais. A Manipulação de Ações define-se como qualquer atividade fraudulenta realizada sobre o mercado de ações visando a obtenção de lucros ou causar danos a alguma outra entidade. Já a Lavagem de Dinheiro, foco deste trabalho, pode ser descrita como uma atividade financeira, envolvendo uma parte externa, que visa ocultar o patrimônio ilegalmente obtido. Existem várias outras atividades que podem se enquadrar na categoria de crime financeiro, dependendo sobretudo do sistema legal e jurídico de cada país.

A Lavagem de Dinheiro é geralmente tida como um tipo de crime financeiro. A Legislação Brasileira possui uma lei específica para tratar desse tipo de atividade. A Lei nº 9.613, de 3 de março de 1998, que discorre sobre "crimes de *lavagem* ou ocultação de bens, direitos e valores", define a Lavagem de Dinheiro como “Ocultar ou dissimular a natureza, origem, localização, disposição, movimentação ou propriedade de bens, direitos ou valores provenientes,

direta ou indiretamente, de crime." (BRASIL, 1998). A Lei nº 12.683, de 9 de julho de 2012, alterou a definição original, trocando "crime", por "infração penal", tornando-a mais abrangente (BRASIL, 2012). A Lavagem de Dinheiro visa ocultar dados provando a ilicitude do patrimônio de forma a evitar a ocorrência dos devidos processos legais e eventual uso dos dados como uma prova de crime. A atividade pode ocorrer com ou sem conhecimento da instituição financeira envolvida. No Brasil, a Lavagem de Dinheiro, juntamente com outros crimes financeiros, tem sido objeto recorrente de discussão. A 11ª edição do Relatório Global de Fraude e Risco da *Kroll* mostra que o Brasil ultrapassa todos os países com relação à quantidade de incidentes de Lavagem de Dinheiro (KROLL, 2019). Além disso, o mesmo relatório coloca a Lavagem de Dinheiro na categoria dos incidentes que mais afetaram as organizações no último ano. Isso mostra a relevância do tema em nível nacional e mundial.

Tanto instituições financeiras quanto agências governamentais fazem uso de métodos automatizados para a detecção de transações suspeitas de Lavagem de Dinheiro (HAN et al., 2020). Atualmente, a maioria dos sistemas implementados, embora frequentemente envolvam Inteligência Artificial ou Mineração de Dados, costumam ser simplistas e baseados em regras, fazendo com que seja necessária uma análise humana após a detecção para que se confirme a ilicitude das transações. Por isso, é comum que a detecção automatizada gere um número consideravelmente grande de transações, fazendo com que o tempo e dinheiro gasto no processo para discriminar as transações ilícitas seja alto e sua qualidade seja negativamente impactada. A inclusão de técnicas mais robustas de Inteligência Artificial tende a diminuir esse número de transações, tornando o processo mais viável.

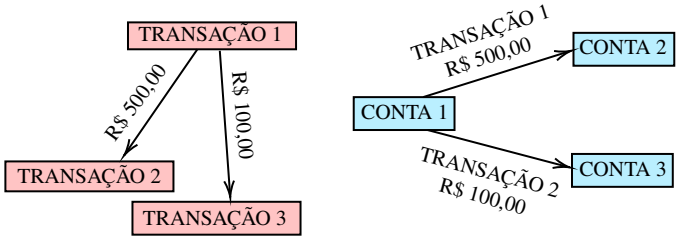
Conforme Han et al. (2020), a literatura tem apresentado várias alternativas aos sistemas de detecção de Lavagem de Dinheiro baseados em regras. Uma das novas abordagens é a Análise de Redes, que estuda a aplicação de conceitos da Teoria de Redes objetivando analisar redes de diferentes naturezas (tais como redes sociais, Internet, etc.). A Análise de Redes pode ser aplicada a dados relacionais de transação financeira para se obter ligações diretas ou ocultas a partir de um nó, onde já se confirma a Lavagem de Dinheiro. A Análise de *Links*, que compreende uma área da Análise de Dados dedicada a avaliar relações entre nós de uma rede, trata os dados de transação financeira como um grafo conexo, onde os nós são as entidades (contas bancárias, pessoas, empresas, etc.) e as transações são as arestas. A Detecção de *Outliers* é uma área da Análise de Dados que busca a detecção de dados com comporta-

mento (conjunto de atributos) muito diferente do restante da base a qual pertencem. Pode ser aplicada à detecção de Lavagem de Dinheiro identificando as transações que possam ser consideradas "anormais"através de seus parâmetros e de limítrofes predefinidos. Geralmente, usa-se *clustering* para aplicar Detecção de *Outliers* em bases de dados de transações. Muitos trabalhos fizeram também o uso de métodos de *Machine Learning* para classificação/*scoring* de risco com o objetivo de determinar as transações suspeitas. Por fim, há também o uso de Aprendizagem de Grafos aplicada à detecção de Lavagem de Dinheiro, que compreende principalmente Redes Neurais de Grafos, sendo o foco deste trabalho.

A principal motivação da aplicação de Aprendizagem de Grafos nesse contexto é a possibilidade de se representar um conjunto de transações através de diferentes tipos de grafos (WEBER et al., 2018). Uma opção é fazer com que os vértices do grafo sejam as transações financeiras. Essa forma é bastante utilizada quando o conjunto de dados vem de um sistema de criptomoedas, como o *Bitcoin Blockchain* (WEBER et al., 2019), e, nesse caso, as arestas do grafo representam o fluxo monetário. Para representar dados de transações bancárias comuns dessa maneira, é possível ligar as transações (vértices) entre si quando ambos tiverem a mesma origem ou destino em comum (PEREIRA; MURAI, 2021). A outra opção para representar o conjunto de dados é fazer com que os vértices sejam as contas bancárias e as arestas sejam as transações. Dessa maneira, é possível utilizar tanto os atributos das próprias contas financeiras quanto os atributos das transações, de forma que a arquitetura de Rede Neural de Grafos (*Graph Neural Network* - GNN) incorpore os dados das arestas e vértices vizinhos separadamente. No entanto, nesse cenário, é necessário utilizar uma arquitetura de GNN que seja capaz de gerar *embeddings* (representações vetoriais do conjunto de atributos geradas pela rede neural) para as arestas. A Figura 1.1 ilustra um exemplo dessas representações.

As GNNs compreendem um subgrupo de redes neurais que recebem um grafo como entrada e vão atualizando, a cada camada de neurônios, o conjunto de atributos dos vértices, incorporando em cada vértice os atributos de seus respectivos vizinhos. Dessa forma, cada vértice é aprendido dentro de seu contexto no grafo, e esse conjunto de instâncias gerado pode ser usado como entrada em qualquer outro algoritmo de *Machine Learning*. A aplicação de GNNs em tarefas envolvendo crimes financeiros ainda é recente na literatura. Alguns estudos criaram arquiteturas de GNN simples e as aplicaram em bases de dados financeiras, muitas vezes rotuladas por meio de heurísticas (WEBER et al., 2019; ALARAB; PRAKOONWIT; NACER,

Figura 1.1 – Representações de transações bancárias em grafos.



(a) (b)

Um exemplo para quando o vértice é a transação bancária (a) e quando a aresta é a transação bancária (b).
Fonte: Do autor (2023).

2020). Além disso, embora tais estudos apresentem alta precisão em seus reconhecimentos, não cobrem de maneira satisfatória o conjunto de instâncias que indicam as atividades ilícitas (baixa revocação). Sendo assim, um dos objetivos deste estudo é encontrar uma solução envolvendo o uso de GNN que atinja um melhor equilíbrio entre precisão e revocação no contexto de crimes financeiros.

A maior parte dos trabalhos sobre detecção de Lavagem de Dinheiro na literatura foca-se em bases de dados de criptomoedas, tais como a *Bitcoin Blockchain*. O modelo de operação financeira sob o qual trabalham essas criptomoedas consiste em transações que realizam sucessivos fluxos monetários entre si, tornando possível sua representação em um grafo no qual o cada nó é uma transação. No entanto, este trabalho tem como foco transações bancárias comuns, uma vez que a maior parte da Lavagem de Dinheiro no cenário brasileiro ocorre por meio delas. Nesse modelo de operação financeira, cada transação está associada a somente um único fluxo monetário entre duas contas bancárias, fazendo com que a aplicação de Redes Neurais de Grafos nesse cenário tenha de levar em consideração arquiteturas que façam uso, em suas operações, de dados associados às arestas do grafo.

Os maiores desafios atualmente envolvendo a aplicação de técnicas de *Machine Learning* no contexto de detecção de Lavagem de Dinheiro são o desequilíbrio de classes e o desvio de conceito (PEREIRA; MURAI, 2021). O desequilíbrio de classes está relacionado ao fato de existirem, em bases de dados de transações, uma quantidade muito maior de transações lícitas do que transações ilícitas, fazendo com que o treinamento dos algoritmos seja negativamente afetado e consequentemente comprometa sua qualidade ao ser aplicado no mundo real.

O algoritmo terá exemplos insuficientes de transações ilícitas e não será capaz de minerar suas características. O desvio de conceito, por outro lado, se refere à mudança nas características estatísticas da distribuição de dados ao longo do tempo. Isso ocorre porque as conexões em uma rede de transações financeiras se altera com o passar do tempo. Novas conexões são formadas, conexões antigas são removidas, e as sub-redes que operam esquemas de lavagem de dinheiro também mudam. Isso faz com que haja uma certa imprevisibilidade no comportamento dos conjuntos de transações sucessivas, também afetando negativamente a qualidade das classificações.

Neste trabalho foi realizada uma série de pesquisas na literatura em busca de um melhor aprofundamento no tema da pesquisa e visando um melhor entendimento das soluções para o problema de detecção de Lavagem de Dinheiro já disponíveis na literatura. As diferentes arquiteturas dessas soluções foram comparadas entre si por meio de métricas clássicas de *Machine Learning* (acurácia, precisão, revocação, etc.).

Muitos dos trabalhos sobre detecção de Lavagem de Dinheiro realizam a detecção baseando-se unicamente em uma única transação, o que, por si só, não é capaz de informar todo o contexto daquela transação, essencial para reconhecer um esquema de Lavagem de Dinheiro. Os trabalhos que utilizam Aprendizagem de Grafos, a qual considera o contexto da transação, a fazem considerando unicamente os atributos das transações, ou não realizando uma separação estrutural entre os atributos das contas bancárias e das transações na própria representação dos dados. Dessa forma, este trabalho avaliou o impacto da utilização de ambos os atributos de transações e contas bancárias, através de uma arquitetura de GNN que incorpora atributos tanto de nós quanto de arestas. Os modelos foram avaliados pela sua capacidade tanto de detectar o maior número possível de transações ilícitas em um conjunto de dados de transações, quanto pela capacidade de distinguir corretamente as transações lícitas das ilícitas. O impacto nos resultados de diferentes classificadores usados em conjunto com o modelo também foi avaliado por este trabalho, bem como a influência da otimização de hiperparâmetros e do uso de dois classificadores sobre os resultados.

1.1 Objetivos

Para este trabalho, foi adotada uma abordagem voltada mais ao aprimoramento dos modelos em si (*model-centric*) do que nos dados (*data-centric*). Vale destacar a importância de

abordagens focadas em *data-centric*, uma vez que um cientista de dados, por exemplo, gasta em média 80% do tempo apenas preparando os dados para serem inseridos em procedimentos posteriores (GUPTA et al., 2021). No entanto, como os dados utilizados neste trabalho foram gerados por um simulador, a qualidade e a padronização dos dados já estava satisfatória, sendo necessárias somente algumas transformações para que os dados pudessem ser inseridos nos modelos de rede neural. Sendo assim, estabelecem-se, para este trabalho, os objetivos gerais e específicos descritos a seguir.

1.1.1 Objetivo geral

Este trabalho tem por objetivo avaliar a performance de modelos de GNN para o problema de detecção automática de Lavagem de Dinheiro em um conjunto de dados de transação financeira, classificando cada transação como "lícita" ou "ilícita", e comparando os resultados em ambas as maneiras de se representar o conjunto de dados em grafos (vértices como transações financeiras ou como contas financeiras), em diferentes níveis de desbalanceamento de classe.

1.1.2 Objetivos específicos

Foram definidos os seguintes objetivos específicos para este trabalho:

a) avaliar um conjunto de arquiteturas de GNN no contexto de detecção automática de Lavagem de Dinheiro em uma base de dados de transações bancárias, buscando encontrar a melhor solução;
b) avaliar o impacto do uso de atributos de nós (contas financeiras) e arestas (transações) na performance da detecção, por meio de uma arquitetura GNN que incorpore atributos de nós e arestas na detecção;
c) testar o impacto de diferentes classificadores utilizados juntamente com as arquiteturas GNN na performance da detecção;
d) avaliar a influência da hiperparametrização e do uso de dois classificadores (Softmax e XGBoost) nos resultados.

1.2 Organização do texto

Este trabalho está organizado como descrito a seguir. O Capítulo 2 deste texto apresenta toda a base teórica necessária para a compreensão da metodologia de pesquisa desenvolvida e experimentos executados. O Capítulo 3 mostra os trabalhos relacionados. O Capítulo 4 aborda a metodologia adotada ao longo da execução. O Capítulo 5 descreve e discute os resultados obtidos e, por fim, o Capítulo 6 conclui este texto abordando as contribuições teóricas e práticas, evidenciando as limitações da solução encontrada, e propondo possíveis projetos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a teoria e os conceitos por trás da metodologia utilizada neste trabalho. Inicialmente, é mostrado o conceito de Anti-Lavagem de Dinheiro e como a Inteligência Artificial é usada nesse domínio (SEÇÃO 2.1). A Seção 2.2 aborda conceitos fundamentais de Teoria de Grafos usados neste trabalho. Em seguida, a Seção 2.3 mostra os fundamentos de Inteligência Artificial e *Machine Learning*. A Seção 2.4 explica sobre o XGBoost, um dos métodos de *Machine Learning* usados neste trabalho. A Seção 2.5 aborda os conceitos gerais de Redes Neurais Artificiais. A Seção 2.6 detalha as Redes Neurais de Grafos, o tipo de arquitetura de rede neural na qual este trabalho foca, descrevendo as arquiteturas utilizadas por este estudo.

2.1 Anti-Lavagem de Dinheiro (*Anti-Money Laundering*)

Os sistemas Anti-Lavagem de Dinheiro (*Anti-Money Laudering* – AML) são implementados por vários tipos de instituições financeiras e agências governamentais para prevenir atividades de Lavagem de Dinheiro em seu sistema financeiro (HAN et al., 2020). Um sistema AML de uma instituição precisa ser suficientemente eficaz para prevenir que a existência de transações aprovadas em seu sistema sejam rotuladas como "ilegais" futuramente pelo sistema de outra instituição. Quando isso ocorre, o sistema de segurança da instituição pode estar comprometido, consequentemente afetando sua reputação e valor de mercado. Para lidar com isso, a maioria dos sistemas de segurança no mundo seguem as recomendações da *Financial Action Task Force* (FATF), chamadas de "Quarenta Recomendações em Lavagem de Dinheiro" (*Forty Recommendations on Money Laudering*), cujo desenvolvimento foi iniciado em 1990 (FATF, 2003). Os sistemas de segurança que seguem as regras da FATF geram uma série de relatórios de atividade suspeita (*Suspicious Activity Reports* – SAR), os quais contêm informações sobre as transações, contas envolvidas e o tipo de atividade suspeita.

Esses sistemas baseados em regras normalmente geram um grande número de atividades suspeitas, sendo que a análise humana posterior é frequentemente necessária para filtrar as transações com um real potencial de serem suspeitas. Logo, a detecção leva um tempo considerável para ser executada, além de ser necessário um gasto financeiro com os especialistas que realizam a análise posterior. Para solucionar esses problemas, a literatura apresenta várias soluções propondo técnicas alternativas de Inteligência Artificial para compor os sistemas AML. Uma

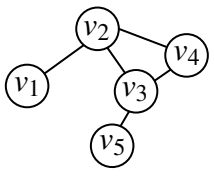
das abordagens é a Análise de Redes, que pode ser aplicada a dados financeiros transacionais de forma a se obter links ocultos ou diretos de um nó no qual já foi detectada a Lavagem de Dinheiro (DREŻEWSKI; SEPIELAK; FILIPKOWSKI, 2015; COLLADON; REMONDI, 2017). Outra alternativa é a Análise de Links, que trata os dados financeiros como um grafo conexo e avalia as relações entre os nós desse grafo (LOPEZ-ROJAZ; AXELSSON, 2012). A Detecção de *Outliers* busca detectar transações com um comportamento (i.e, seu conjunto de atributos) consideravelmente diferente do restante das transações (transações "anormais") (LE-KHAC; MARKOS; KECHADI, 2009; LARIK; HAIDER, 2011). Muitos trabalhos também fazem uso de métodos de *Machine Learning* para classificação/*scoring* de risco, usando o resultado como base para detectar as transações suspeitas (CHEN et al., 2018). Por fim, há também o uso de Aprendizado de Grafos aplicado à detecção de Lavagem de Dinheiro, que compreende majoritariamente Redes Neurais de Grafos e é o foco deste trabalho (ALARAB; PRAKOONWIT, 2022).

2.2 Conceitos de Teoria de Grafos

Conforme descreve Goldbarg e Goldbarg (2012), um grafo (ou rede) é uma abstração matemática que descreve relações de dependência dentre os elementos de um conjunto, sendo uma estrutura bem útil na resolução de diversos problemas. Tais elementos, denominados nós ou vértices, são graficamente representados por círculos ou pontos, enquanto a dependência entre dois elementos, denominada aresta, é graficamente representada por um segmento de reta ligando esses elementos.

Formalmente, um grafo é denotado por uma tupla $G = (V, E)$, na qual $V = \{v_1, v_2, ..., v_n\}$ é o conjunto de vértices do grafo e $E = \{e_1, e_2, ..., e_m\}$ é o conjunto de arestas. A Figura 2.1 exemplifica a representação de um grafo.

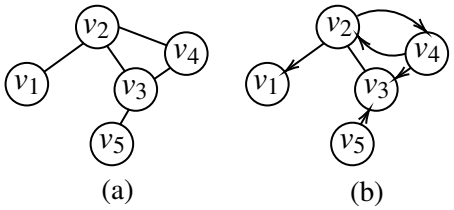
Figura 2.1 – Representação gráfica de um grafo.



Fonte: Do autor (2023).

Os grafos podem ser direcionados ou não direcionados. Um grafo é dito direcionado quando os pares de vértices que compõem uma aresta são ordenados, isto é, a aresta possui uma origem e um destino. Dessa forma, a existência de uma ligação (v_i, v_j) no grafo não implica a existência de uma ligação (v_j, v_i). As arestas de um grafo direcionado, chamadas de arcos, são representadas graficamente como setas apontadas para o vértice destino. Por outro lado, nos grafos não direcionados, os pares de vértices não possuem uma ordem, não havendo, portanto, uma origem e um destino para cada aresta. Graficamente, as arestas de um grafo não direcionado não contêm uma seta, como as arestas de um grafo direcionado. A Figura 2.2 ilustra a diferença entre grafos direcionados e não direcionados.

Figura 2.2 – Grafo não direcionado e direcionado.



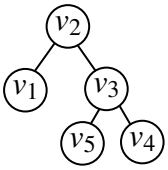
Exemplo de grafo não direcionado (a) e direcionado (b).
Fonte: Do autor (2023).

Os componentes de um grafo podem possuir relações de vizinhança. Dois vértices de um grafo são considerados vizinhos (ou adjacentes) se, e somente se possuem uma aresta conectando-os. Já duas arestas são consideradas vizinhas (ou adjacentes) quando compartilham um vértice em comum. O conjunto de vizinhos (ou conjunto vizinhança) de um vértice v é denotado por $\mathcal{N}(v)$.

Existem diversos tipos especiais de grafos, sendo um deles a árvore. Uma árvore pode ser definida como um grafo conexo (i.e., todos os vértices do grafo possuem ao menos uma aresta conectando-o a outro vértice) e acíclico (não possui ciclos, i.e., um caminho formado por um conjunto de arestas ligando um vértice a si mesmo). Os nós de grau 1 de uma árvore são chamados de nós-folha. As árvores são estruturas utilizadas em vários métodos de *Machine Learning*, como o XGBoost (SEÇÃO 2.4). A Figura 2.3 ilustra uma árvore.

Para todo grafo, é possível gerar um grafo aresta correspondente. Um grafo aresta $L(G)$ de um grafo G representa as adjacências entre as arestas de G. Cada nó de $L(G)$ corresponde a uma aresta de G, sendo que dois nós de $L(G)$ estão conectados se, e somente se suas arestas

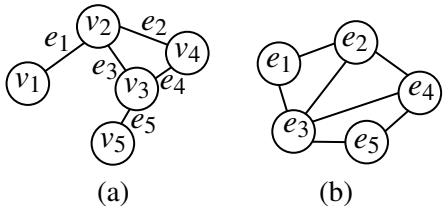
Figura 2.3 – Exemplo de árvore.



Fonte: Do autor (2023).

correspondentes em G forem vizinhas. A Figura 2.4 ilustra um grafo e seu respectivo grafo aresta.

Figura 2.4 – Exemplo de grafo aresta.



Exemplo de um grafo (a) e seu respectivo grafo aresta (b).
Fonte: Do autor (2023).

Existem diversas maneiras de se representar grafos computacionalmente, como Listas de Adjacência, Listas de Incidência, e Matrizes de Adjacência, que é a estrutura utilizada pelos métodos deste trabalho. Para um Grafo G, a matriz de adjacência $A = [a_{ij}]$ obedece às seguintes propriedades:

i $a_{ij} = 1$, se $(i, j) \in E$;

ii $a_{ij} = 0$, caso contrário.

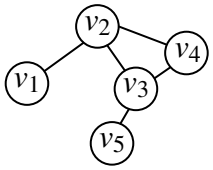
Dessa forma, a matriz de adjacências para um grafo não direcionado será sempre simétrica. A Figura 2.5 ilustra um grafo e sua respectiva matriz de adjacência.

Os conceitos apresentados nesta seção são utilizados em várias metodologias importantes para este trabalho (SEÇÕES 2.4 e 2.6).

2.3 Inteligência Artifical e *Machine Learning*

Este trabalho faz uso de técnicas de Inteligência Artificial (IA), mais especificamente de Aprendizado de Máquina (*Machine Learning* – ML). A IA é uma ciência que estuda princípios

Figura 2.5 – Exemplo de matriz de adjacência.



(a)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(b)

Exemplo de um grafo (a) e sua respectiva matriz de adjacência (b).
Fonte: Do autor (2023).

e técnicas que possibilitam aos computadores capacidade de raciocínio e tomada de decisões de maneira inteligente (RUSSEL; NORVIG, 2013). É possível abordar a IA de quatro formas:

a) fazer com que o computador pense racionalmente;
b) fazer com que o computador aja racionalmente;
c) fazer com que o computador pense como ser humano;
d) fazer com que o computador aja como ser humano.

O ML é uma sub-área da IA cujas técnicas objetivam fazer com que as máquinas aprendam a partir de experiências anteriores e sejam capazes de tomar decisões cada vez mais inteligentes na medida em que se dá esse aprendizado (MARSLAND, 2015). O ML possui diversos algoritmos que podem ser classificados em quatro tipos diferentes:

a) **Aprendizado Evolutivo**: Um mecanismo de aprendizado baseado na seleção natural. No cenário do Aprendizado Evolutivo há um determinado problema a ser resolvido e um algoritmo que o resolva, no qual seja possível modificar parâmetros. Inicialmente, várias instâncias desse algoritmo são criadas aleatoriamente, cada qual com um conjunto distinto de parâmetros. Os algoritmos são então testados no cenário em questão (problema a ser resolvido) e seu sucesso ou fracasso são avaliados. Os parâmetros dos conjuntos que tiverem melhores resultados no cenário são combinados entre si, gerando novos conjuntos, e a avaliação recomeça. A instância com os melhores resultados é escolhida na última iteração;
b) **Aprendizado Não-supervisionado**: O algoritmo se corrige utilizando similaridades entre as entradas, de forma que entradas similares produzam a mesma resposta. Nesse caso, não há um conjunto de dados contendo a resposta esperada, fazendo com que a avaliação da qualidade das respostas deva ser feita através de métricas estatísticas apropriadas, como a entropia por exemplo. As principais tarefas para este tipo são o Agru-

pamento (ou Clusterização), que visa agrupar os elementos de uma base de dados em *clusters* a partir de seus atributos, e a Mineração de Regras de Associação, que busca encontrar elementos que ocorrem simultaneamente no conjunto de dados;

c) **Aprendizado por Reforço**: De maneira similar ao Aprendizado Evolutivo, há um cenário (um problema a ser resolvido) e um algoritmo para resolvê-lo com parâmetros passíveis de modificação. Inicialmente, o modelo é criado com um conjunto inicial de parâmetros e testado no cenário em questão. O algoritmo é informado quando sua resposta ou ação está correta ou não, e corrige seus parâmetros com base nessa informação;

d) **Aprendizado Supervisionado**: Há uma etapa de treinamento na qual as respostas corretas são fornecidas, e o algoritmo corrige seus parâmetros comparando suas próprias respostas com as respostas corretas. Podem ser de Regressão ou Classificação. A Regressão objetiva encontrar um resultado numérico aproximado para um dado com base em seus outros atributos. Já a Classificação busca atribuir um rótulo (valor discreto, também denominado classe) a um dado, com base nos atributos desse dado.

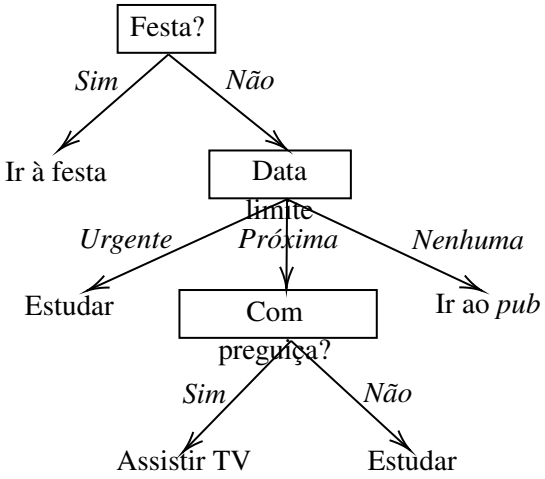
Este trabalho utiliza técnicas de Aprendizado Supervisionado, objetivando a tarefa de Classificação. O objetivo é rotular um conjunto de transações de licitude desconhecida com base em um conjunto de transações cuja licitude já é conhecida pelo algoritmo. As seções seguintes detalham as técnicas de ML utilizadas neste trabalho.

2.4 XGBoost

Um dos classificadores utilizados neste trabalho é o *Gradient Boosted Trees*, através de sua implementação mais popular fornecida pela biblioteca XGBoost (CHEN; GUESTRIN, 2016). O nome "XGBoost" também é usado para se referir ao algoritmo em si, sendo utilizado dessa forma neste trabalho. O XGBoost pertence à categoria dos algoritmos denominada "decisão por comitê" (*decision by comitee*), que utiliza um conjunto de classificadores de baixa acurácia, atribuindo um peso à saída de cada um, de forma que a resposta com maior somatório de pesos seja a escolhida, o que faz com que a saída final do algoritmo tenha alta acurácia. No caso do XGBoost, são utilizadas árvores de decisão, i. e., classificadores estruturados em árvore que utilizam uma sequência de decisões baseadas nos atributos de entrada para obter a resposta final. Cada nó folha dessa árvore corresponde a uma classe e os outros nós correspondem às

decisões. A profundidade da árvore é a quantidade máxima de perguntas para se chegar a um rótulo. A Figura 2.6 é um exemplo de árvore de decisão de profundidade 3.

Figura 2.6 – Exemplo de árvore de decisão de profundidade 3.



Fonte: Adaptado de Marsland (2015).

O treinamento do conjunto de classificadores é feito de maneira sequencial, i.e, o classificador seguinte é treinado baseado nos erros de seu predecessor. Para isso, cada classificador é avaliado através de uma função de perda, sendo a Entropia Cruzada uma das mais utilizadas. A Equação 2.1 mostra como se calcula o erro entre o conjunto de valores preditos por um modelo ($\hat{y}$) e os valores reais (y) para um conjunto de N elementos.

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)]. \tag{2.1}$$

Durante o treinamento, o XGBoost cria os modelos preditivos na iteração seguinte de forma que o erro do conjunto de classificadores seja menor que o erro na iteração anterior. Para isso, é utilizado um método conhecido como *Gradient Descend* (RUDER, 2016), que utiliza derivadas para calcular a influência de cada modelo no erro como um todo, e minimizá-lo a cada passo de execução. Mais detalhadamente, o objetivo é criar um novo conjunto de modelos F_m na iteração m conforme descrito na Equação 2.2,

$$F_m = F_{m-1} - \eta * \frac{\partial L(F_{m-1})}{\partial F_{m-1}}, \tag{2.2}$$

na qual η é a taxa de aprendizado do XGBoost, um valor que evita uma convergência muito abrupta do algoritmo, o que pode ocasionar problemas como sobreajuste do modelo ao conjunto de treinamento. Por outro lado, se a taxa de aprendizado for muito baixa, cada etapa do treinamento se torna pouco eficiente, também afetando negativamente a qualidade do modelo treinado.

2.5 Redes Neurais Artificiais

De acordo com Haykin (2001), uma Rede Neural Artificial (Artificial Neural Network - ANN) é um modelo de ML inspirado no funcionamento do cérebro humano. Uma ANN constitui-se de um conjunto de neurônios artificiais, que nada mais são que unidades de processamentos simples.

O primeiro modelo de ANN é chamado de *Perceptron* e contém somente um neurônio (ROSENBLATT, 1958). Esse neurônio é formado por um conjunto de entradas x, um conjunto de pesos w (cada peso associado a uma entrada), um somador e uma função de ativação $\varphi(.)$. Para computar uma saída y para um determinado conjunto de atributos, a ANN recebe na sua entrada esses atributos, multiplica cada valor da entrada pelo seu respectivo peso, soma esse conjunto de resultados, e calcula a função de ativação usando o resultado da soma como parâmetro, gerando assim a saída. A ANN pode também conter um valor de *bias* b. A Equação 2.3 descreve um *Perceptron* simples.

$$y = \varphi(wx + b). \tag{2.3}$$

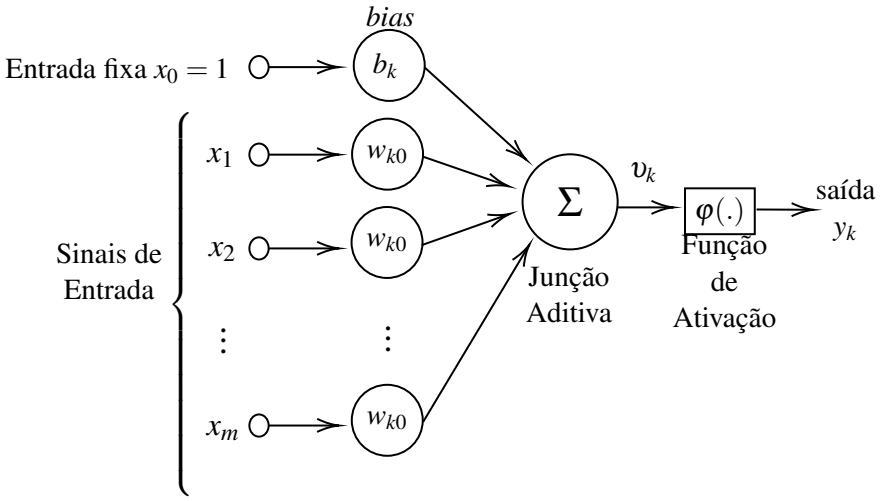
O *bias* é um peso fixo associado a uma entrada fixada em +1 e serve para modificar a relação entre o potencial de ativação do neurônio e a saída da função de ativação. A Figura 2.7 ilustra o funcionamento do *Perceptron* simples.

Com relação à função de ativação, existem diversas funções que podem ser usadas para tal. Seu objetivo é normalizar a saída do somatório para um intervalo específico de valores. Um exemplo de função de ativação é a função sigmoide, que pode ser descrita pela Equação 2.4,

$$\varphi(\upsilon) = \frac{1}{1 + \exp(-a\upsilon)}, \tag{2.4}$$

na qual a é o parâmetro de inclinação da função e υ é a variável a ser ativada.

Figura 2.7 – Ilustração de um *Perceptron* simples.



Fonte: Adaptado de Haykin (2001).

As ANNs realizam o aprendizado modificando o conjunto de pesos e o *bias* de seus neurônios para produzir a saída desejada. A cada iteração, um novo conjunto de atributos é inserido e os pesos são ajustados. Para isso, usa-se a saída y da rede neural e o valor real esperado d para calcular uma função de erro (também conhecida como função de custo). Um exemplo de função de erro, no caso de um *Perceptron* simples, é a subtração do valor esperado pela saída da rede ($e = d - y$). Os pesos são então atualizados com base nesse erro. A variação Δw_i para o peso w_i pode ser obtida pela Equação 2.5,

$$\Delta w_i = \eta e_k x_i, \tag{2.5}$$

na qual x_i é a entrada associada ao peso w_i e η é a taxa de aprendizado, usada para evitar que os pesos se ajustem abruptamente à saída a cada iteração.

A partir do surgimento do *Perceptron*, foram então desenvolvidas diversas arquiteturas de ANN. Na literatura as ANNs organizadas por camadas são muito usadas, na qual uma camada é composta por neurônios que processam o conjunto de entradas e geram um conjunto de saídas, sendo que o conjunto de saídas é usado como entrada para cada neurônio da camada seguinte, e assim sucessivamente, até que a última camada, conhecida como camada de saída, gere a saída final da ANN. Quando há apenas uma camada, o processo de aprendizado é conhecido como *Shalow Learning*, e quando há duas ou mais camadas, o processo é denominado *Deep*

Learning. A ANN de múltiplas camadas mais simples é denominada *Multi-Layer Perceptron* (MLP).

Para que os pesos de cada neurônio em cada camada sejam atualizados da maneira correta em um MLP, usa-se um processo denominado Retropropagação do Erro (*Error Backpropagation*), no qual o erro final é propagado para as camadas anteriores através da derivada dos pesos de cada neurônio sobre a derivada do erro.

Existem diversas arquiteturas de MLP. As Redes Alimentadas Adiante (*Feed Forward*) possuem uma camada de entrada, uma camada de saída e um conjunto de camadas ocultas, e o fluxo de informação sempre ocorre a partir da entrada em direção à saída. Já as Redes Neurais Recorrentes realimentam seus neurônios usando suas respectivas saídas como parte de sua entrada, o que funciona como uma espécie de "memória", sendo bastante usado para processar entradas com tamanhos arbitrários. Uma variação dessas Redes Recorrentes é a LSTM (*Long short-term Memory*), capaz de lidar com sequências de tamanho arbitrário, sendo muito úteis quando não se sabe o tamanho da série temporal. As Máquinas de Boltzmann são redes neurais probabilísticas que usam o conceito de "energia", muito usadas para detecção de fraudes e sistemas de recomendação. As *Deep Belief Networks* também são redes probabilísticas, que constituem-se de um conjunto de camadas de variáveis latentes. Os *Autoencoders* são treinados para gerar um conjunto de saída igual ao de entrada e são usados para codificação, sendo muito úteis quando se deseja reduzir o tamanho do conjunto de entrada. As GANs (*Generative Adversarial Networks*) são arquiteturas formadas por duas redes neurais que aprendem através de uma competição entre si. As Redes Convolucionais realizam sucessivas convoluções na sua entrada com o objetivo de abstrair suas características, sendo muito usadas para processar imagens e sinais de áudio. Por fim, também há a Rede Neural de Grafos (*Graph Neural Network* – GNN), foco de estudo deste trabalho.

2.5.1 Classificação com Softmax

Para realizar a tarefa de classificação em uma rede neural artificial, a alternativa mais utilizada é a inclusão de uma camada no fim da rede que tenha um número de neurônios igual ao número de classes possíveis e que tenha como função de ativação a função Softmax (MARSLAND, 2015). A função Softmax realiza uma normalização dos valores de saída de cada neurônio através do cálculo de seu exponencial, e divide esse valor pelo somatório dos exponenciais

da saída de todos os neurônios. Após a função, todos esses valores estarão entre 0 e 1 e seu somatório resultará em 1. Dessa forma, o neurônio com o valor mais alto corresponderá a classe escolhida. Sejam os valores de saída dos neurônios de uma rede $h_1, h_2, \ldots, h_N$. A Equação 2.6 mostra como calcular o valor de saída normalizado y_k do neurônio k através da função Softmax.

$$y_k = \frac{exp(h_k)}{\sum_{i=1}^{N} exp(h_i)}. \tag{2.6}$$

Uma outra alternativa para realizar a tarefa de classificação consiste em inserir a saída da rede em um outro algoritmo de classificação, como o XGBoost, por exemplo (SEÇÃO 2.4). Logo, pode-se considerar a camada contendo a função Softmax como um classificador, o que é feito nos experimentos deste trabalho (CAPÍTULO 5).

2.6 Redes Neurais de Grafos

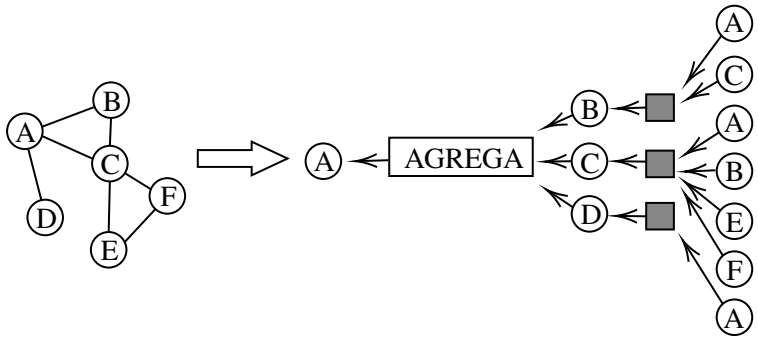
Uma Rede Neural de Grafos (*Graph Neural Networks* – GNN) é uma rede neural usada para processar conjuntos de dados estruturados em grafos (HAMILTON, 2020). Uma GNN irá receber como entrada, além do conjunto de instâncias e seus atributos como toda a ANN, a informação sobre as ligações entre as instâncias por meio de matrizes de adjacências. Ou seja, a GNN recebe um grafo como entrada. A cada iteração, a GNN realiza uma operação de convolução nesse grafo baseada na ligação entre os vértices, e cada vértice recebe um novo conjunto de atributos (conhecido como *hidden embedding*) contendo o resultado da agregação entre o conjunto de atributos daquele vértice e o conjunto de atributos de seus vizinhos. Essa operação pode ser representada pela Equação 2.7.

$$\begin{aligned} h_u^{(k+1)} &= \text{ATUALIZA}^{(k)}\left(h_u^{(k)},\ \text{AGREGA}^{(k)}\left(\left\{h_v^{(k)}, \forall v \in \mathscr{N}(u)\right\}\right)\right) \\ &= \text{ATUALIZA}^{(k)}\left(h_u^{(k)},\ m_{N(u)}^{(k)}\right) \end{aligned} \tag{2.7}$$

Ou seja, cada *hidden embedding* $h_v^{(k)}$ do vértice v, para todo v vizinho do vértice u, i.e., $v \in \mathscr{N}(u)$, será agregado por uma função de agregação $\text{AGREGA}^{(k)}$ produzindo a "mensagem" $m_{N(u)}^{(k)}$. Esse conjunto de valores será combinado com o *hidden embedding* $h_{u(k)}$ do vértice u, através da função $\text{ATUALIZA}^{(k)}$, gerando um novo *hidden embedding* $h_u^{(k+1)}$. Essa operação é conhecida como *message passing*, e é ilustrada pela Figura 2.8. Se ela for executada somente

uma vez, cada vértice do grafo terá, ao fim de sua execução, a sua própria informação agregada à de seus vizinhos imediatos. Quando há mais iterações (i.e. mais camadas de *message passing*), os vértices também englobam informações sobre a vizinhança de seus vizinhos, gerando um conjunto de atributos que inclui uma informação de localização daquele vértice no grafo. Esses novos conjuntos de atributos poderão ser usados para uma operação futura qualquer, como regressão, classificação etc., ou serem enviadas para as camadas seguintes da GNN, caso haja alguma.

Figura 2.8 – Ilustração do funcionamento do *message passing* para duas execuções.



O nó A agrega os valores de seus vizinhos, sendo que esses vizinhos imediatos, também agregam valores de seus vizinhos imediatos.

Fonte: Adaptado de Hamilton (2020).

As GNNs possuem diversos usos, tais como aplicações na área de Química (COLEY et al., 2019; YANG; CHAKRABORTY; WHITE, 2021), Genética (LI et al., 2019), etc. Porém este trabalho focará em conjuntos de dados envolvendo crimes financeiros.

2.6.1 Rede Convolucional de Grafo

Uma das implementações mais simples e populares de GNN é a Rede Convolucional de Grafo (*Graph Convolutional Network – GCN*), descrito pela primeira vez em Kipf e Welling (2017). Esse modelo realiza uma operação de convolução baseada nas ligações entre os diferentes vértices do grafo. A operação que ocorre a cada iteração em uma GCN é a seguinte: para cada vértice u do grafo, os *hidden embeddings* h_v de cada vértice v, tal que v seja vizinho de u ou o próprio u, são normalizados de acordo com seu próprio grau e o grau de u. Os *hidden embeddings* normalizados são então somados entre si e o vetor resultante é multiplicado pelo conjunto de pesos $W^{(k)}$ e inserido como entrada em uma função de ativação σ, para que se torne o novo *hidden embedding* de u. Esse processo pode ser representado pela Equação 2.8.

$$h_u^{(k)} = \sigma\left(\sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{h_v}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}}\right), \tag{2.8}$$

na qual $\mathcal{N}(.)$ corresponde aos vizinhos de determinado vértice. Uma outra maneira de representar esse processo é visualizando-o no grafo como um todo, de forma que são utilizadas matrizes para descrever as transformações. Seja A a matriz de adjacências do grafo de entrada, I uma matriz identidade da mesma ordem de A, $H_t^{(l)}$ e $W_t^{(l)}$ a matriz de *hidden embeddings* desse grafo e a matriz de pesos, respectivamente, na camada (l) e no tempo t. Seja $\tilde{A}$ a matriz de adjacências A incluindo um *loop* em cada nó ($\tilde{A} = A + I$), $\tilde{D}$ a matriz diagonal formada pelo somatório das colunas de $\tilde{A}$ ($\tilde{D} = diag(\sum_j \tilde{A}_{ij})$), e $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Considerando o grafo como um todo, o modelo de GCN pode ser representado pela Equação 2.9.

$$H_t^{(l+1)} = \sigma\left(\hat{A}_t H_t^{(l)} W_t^{(l)}\right) \tag{2.9}$$

2.6.2 Rede Neural de Nós e Arestas (NENN)

Introduzida por Yang e Li (2020), a Rede Neural de Nós e Arestas (*Node and Edge Neural Networks* - NENN) utiliza em suas operações tanto atributos de nós quanto de arestas, gerando *hidden embeddings* para ambos. Para compreender a NENN, é necessário definir os seguintes conceitos sobre vizinhança em grafos:

a) **Vizinhos Baseados em Nó**: Para um grafo $G = (V, E)$, os vizinhos baseados em nó $\mathcal{N}_i$ de um nó i são os nós ligados diretamente a este por uma aresta, e o próprio nó i. Para uma aresta j desse mesmo grafo, seus vizinhos baseados em nó $\mathcal{N}_j$ são os nós conectados por essa aresta;

b) **Vizinhos Baseados em Aresta**: Para um grafo $G = (V, E)$, os vizinhos baseados em aresta $\mathcal{E}_i$ de um nó i são as arestas que se conectam a esse nó. Para uma aresta j desse mesmo grafo, os seus vizinhos baseados em aresta são as arestas que se conectam aos nós conectados por esta aresta, e a própria aresta j.

A NENN trabalha com mecanismos de atenção e possui dois tipos de camadas. As Camadas de Atenção no Nível de Nó (*Node-Level Attention Layer*) geram *hidden embeddings* para os nós através de seus vizinhos baseados em nó e em aresta. Já as Camadas de Atenção no Nível de Aresta (*Edge-Level Attention Layer*) geram os *hidden embeddings* para as arestas.

Ambas as camadas calculam coeficientes de importância para os vizinhos baseados em nó e em aresta, para cada nó ou aresta em questão. Esses coeficientes são então normalizados pela função Softmax e são utilizados para calcular os *hidden embeddings*, de forma que os *embeddings* relativos aos vizinhos baseados em nó e em aresta são calculados separadamente, e unidos em seguida formando um único *hidden embedding* para aquele nó ou aresta.

A Camada de Atenção no Nível de Nó calcula um coeficiente de importância α_{ij}^{n} de cada nó j para cada nó i. Esse coeficiente, para um dado nó i é obtido multiplicando-se os conjuntos de atributos x_i do nó i e x_j do nó j pela matriz de pesos $W_n \in \mathbb{R}^{d_v^l + d_v^{l+1}}$, sendo d_v^l o tamanho do *embedding* de um nó na camada l. Os resultados dessas multiplicações sofrem uma concatenação (nesta seção representados pelo símbolo $||$) e multiplicado pelo vetor de parâmetros $a_n^T \in \mathbb{R}^{2d_v^{l+1}}$ de uma Rede Alimentada Adiante de camada única. Em seguida, o resultado passa pela função de ativação σ (p. ex. LeakyReLU), sendo normalizado via Softmax, conforme mostra a Equação 2.10.

$$\alpha_{ij}^{n} = \frac{\exp\left(\sigma\left(a_n^T \left[W_n x_i \,\|\, W_n x_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\sigma\left(a_n^T \left[W_n x_i \,\|\, W_n x_k\right]\right)\right)} \tag{2.10}$$

Outro coeficiente de importância calculado por esta camada é o α_{ik}^{e}, que quantifica a importância de cada aresta k para cada nó i. Multiplica-se o conjunto de atributos e_k da aresta k pela matriz de pesos $W_e \in \mathbb{R}^{d_e^l + d_e^{l+1}}$ (sendo d_e^l o tamanho do *embedding* da aresta e na camada l). As multiplicações são então concatenadas e multiplicadas pelo vetor de parâmetros $a_e^T \in \mathbb{R}^{d_v^{l+1} + d_e^{l+1}}$ de uma Rede Alimentada Adiante de camada única. Por fim, o resultado é ativado pela função σ e normalizado via Softmax, como ilustrado na Equação 2.11.

$$\alpha_{ik}^{e} = \frac{\exp\left(\sigma\left(a_e^T \left[W_n x_i \,\|\, W_e e_k\right]\right)\right)}{\sum_{j \in \varepsilon_i} \exp\left(\sigma\left(a_e^T \left[W_n x_i \,\|\, W_e e_j\right]\right)\right)} \tag{2.11}$$

Os coeficientes α_{ij}^{n} são multiplicados pelos atributos de vetor x_j para todo nó j vizinho de i, para obter uma média ponderada pelas importâncias de cada atributo entre os vizinhos de i. O resultado então é multiplicado pela matriz de pesos W_n e é ativada pela função σ, gerando o conjunto de valores $x_{\mathcal{N}_i}$, conforme mostra a Equação 2.12.

$$x_{\mathcal{N}_i} = \sigma\left(W_n \cdot \text{MEDIA}\left(\left\{\alpha_{ij}^{n} x_j, \forall_j \in \mathcal{N}_i\right\}\right)\right) \tag{2.12}$$

Já os coeficientes α_{ik}^{e} são multiplicados pelos atributos de aresta e_k para toda aresta k vizinha do nó i, para obter uma média ponderada pelas importâncias de cada atributo entre as arestas vizinhas de i. O resultado então é multiplicado pela matriz de pesos W_e e é ativada pela função σ, gerando o conjunto de valores x_{ε_i}, conforme mostrado na Equação 2.13.

$$x_{\varepsilon_i} = \sigma\left(W_e \cdot \text{MEDIA}\left(\{\alpha_{ik}^{e} e_k, \forall_k \in \varepsilon_i\}\right)\right) \tag{2.13}$$

Por fim, os resultados das equações 2.12 e 2.13 são concatenados, como mostra a Equação 2.14, para formar o *embedding* $x_i^{(l+1)}$.

$$x_i^{(l+1)} = x_{\mathcal{N}_i}^{(l)} \| x_{\varepsilon_i}^{(l)} \tag{2.14}$$

A Camada de Atenção no Nível de Aresta trabalha de maneira similar à Camada de Atenção no Nível de Nó, porém calculando as atenções e *embeddings* dos vizinhos de cada aresta. Dada uma aresta i (com um conjunto de atributos e_i) qualquer do grafo e uma aresta k vizinha de i (com um conjunto de atributos e_k), multiplicam-se seus conjuntos de atributos pela matriz de pesos W_e e concatenam-se os resultados. Esse valor é então multiplicado pelo vetor de parâmetros $q_e^T \in \mathbb{R}^{2d_e^{l+1}}$, ativado pela função σ e normalizado pela função Softmax para se obter a importância β_{ik}^{e}, como mostra a Equação 2.15

$$\beta_{ik}^{e} = \frac{\exp\left(\sigma\left(q_e^T \left[W_e e_i \| W_e e_k\right]\right)\right)}{\sum_{j \in \varepsilon_i} \exp\left(\sigma\left(q_e^T \left[W_e e_i \| W_e e_j\right]\right)\right)} \tag{2.15}$$

Da mesma forma, calcula-se a importância β_{ij}^{n} de cada nó j (com um conjunto de atributos x_j) para cada aresta i (com um conjunto de atributos e_i). O conjunto de atributos x_j é multiplicado pela matriz de pesos W_n e o conjunto de atributos e_i é multiplicado pela matriz de pesos W_e. Os dois resultados são concatenados e multiplicados pelo vetor de parâmetros $q_n^T \in \mathbb{R}^{d_v^{l+1}+d_e^{l+1}}$. Ativa-se o resultado com a função σ e normaliza-se via Softmax, como ilustra a Equação 2.16.

$$\beta_{ij}^{n} = \frac{\exp\left(\sigma\left(q_n^T \left[W_e e_i \| W_n x_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\sigma\left(q_n^T \left[W_e e_i \| W_n x_k\right]\right)\right)} \tag{2.16}$$

A importância $\beta_{ik}^{n} e_k$ é multiplicada pelo conjunto de atributos e_k para cada aresta k vizinha de i, para obter uma média ponderada para cada atributo entre as arestas k. O resultado é multiplicado pela matriz de pesos W_e e ativado pela função σ para obter o conjunto de valores e_{ε_i}. Isso é ilustrado pela Equação 2.17.

$$e_{\varepsilon_i} = \sigma\left(W_e \cdot \text{MEDIA}\left(\{\beta_{ik}^{n} e_k, \forall_k \in \varepsilon_i\}\right)\right) \tag{2.17}$$

Da mesma forma, a importância $\beta_{ij}^{e} x_j$ é multiplicada pelo conjunto de atributos e_k para cada nó j vizinho da aresta i, para obter uma média ponderada para cada atributo entre os nós j. O resultado é multiplicado pela matriz de pesos W_n e ativado pela função σ, gerando o conjunto de valores $e_{\mathcal{N}_i}$, como mostra a Equação 2.18.

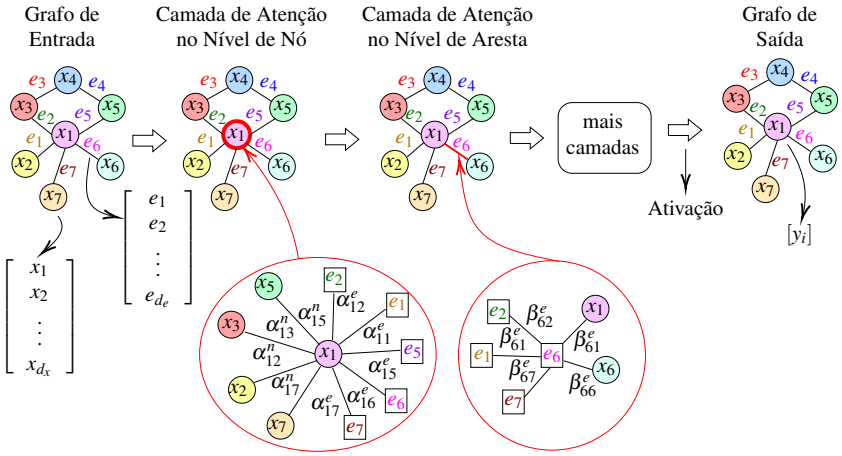
$$e_{\mathcal{N}_i} = \sigma\left(W_n \cdot \text{MEDIA}\left(\left\{\beta_{ij}^{e} x_j, \forall_j \in \mathcal{N}_i\right\}\right)\right) \tag{2.18}$$

Por fim, o resultado das equações 2.17 e 2.18 são concatenados para se obter o *embedding* $e_i^{(l+1)}$.

$$e_i^{(l+1)} = e_{\mathcal{N}_i}^{(l)} \,\|\, e_{\varepsilon_i}^{(l)} \tag{2.19}$$

Ambas as camadas no nível de aresta e de nó podem ser combinadas sequencialmente e/ou com outros tipos de camadas para a construção do modelo desejado. Os *embeddings* calculados em uma camada são usados como atributos pela camada seguinte, sejam eles associados aos nós ou às arestas. A Figura 2.9 ilustra um modelo simples baseado na arquitetura NENN, no qual têm-se uma Camada de Atenção no Nível de Nó seguida uma Camada de Atenção no Nível de Aresta, além de uma abstração de possíveis camadas adicionais. No fim, para a resolução de um problema de classificação, os *embeddings* de nós e arestas são combinados e inseridos em um modelo de classificação, podendo ser uma camada MLP simples, por exemplo.

Figura 2.9 – Ilustração da arquitetura NENN em um modelo de exemplo



Fonte: Adaptado de Yang e Li (2020).

A teoria detalhada neste capítulo é necessária para o entendimento dos trabalhos relacionados apresentados no capítulo seguinte (CAPÍTULO 3) bem como para a metodologia explicada no Capítulo 4.

3 TRABALHOS RELACIONADOS

O conjunto de referências deste trabalho foi construído pesquisando-se por soluções similares que envolvessem Redes Neurais de Grafos presentes na literatura. As soluções pesquisadas deveriam lidar com problemas cujas abordagens sejam o mais parecidas possíveis com o problema alvo deste trabalho. Ao mesmo tempo em que se pesquisou por tais soluções, foram buscados trabalhos da literatura nos quais foram usadas bases de dados de Transações Bancárias, contendo informações sobre Lavagem de Dinheiro. Este capítulo descreve os trabalhos relacionados encontrados nessa pesquisa e faz uma comparação com este trabalho.

Foi desenvolvida uma base de dados em grafos em Weber et al. (2019) para a detecção de crimes financeiros a partir de dados reais da *Bitcoin Blockchain*. As classes dessa base foram geradas artificialmente por uma heurística, separando os dados em "lícito", "ilícito"ou "desconhecido", indicando a presença ou não de lavagem de dinheiro na transação. Para efetuar a classificação nessa base, foi usada uma Rede Convolucional de Grafo simples com duas camadas. Os *embedding vectors* gerados pela rede também foram incluídos posteriormente no conjunto de atributos para testar a influência da presença da localidade na informação dentro de outros métodos de classificação, nomeadamente Regressão Logística, *Random Forest* e *Multi-Layer Perceptron* simples. Também foi testada a influência da temporalidade nos dados com o uso de uma abordagem recorrente para a Rede Convolucional de Grafo.

Os métodos de Weber et al. (2019) foram comparados entre si através de sua precisão, revocação e F1 para a classe "ilícito" e micro-F1. O melhor resultado foi obtido pelo *Random Forest* com o uso dos *embedding vectors* (Precisão de 0,971), porém com baixa revocação (0,624). Nenhum dos métodos propostos, como mencionado pelo próprio autor, conseguiu lidar com o fechamento do mercado negro ocorrido em um dos *timesteps* da base de dados. Além disso, a base de dados utilizada pelos autores foi artificialmente rotulada, e os resultados podem mudar quando os métodos forem testados em uma base de dados real. Os autores deixaram também para discussões futuras novas ideias para se combinar Redes Neuras de Grafos com *Random Forest* e Regressão Logística.

Gerou-se uma base de dados de contas ilícitas em Farrugia, Ellul e Azzopardi (2020) a partir de transações reais da Ethereum Blockchain e uma lista de contas que comprovadamente praticaram atividades ilegais naquele domínio. Utilizou XGBoost para detecção dessas contais ilegais, atingindo uma acurácia média de 0,963 e uma área sob a curva (AUC) média de 0,994.

O método proposto, no entanto, não é capaz de detectar a lógica computacional de contratos inteligentes realizados com o intuito de ocultar a atividade ilícita. Além disso, a abordagem em Farrugia, Ellul e Azzopardi (2020) foi feita por meio de contas financeiras, enquanto a utilizada neste trabalho será a nível de transação.

Uma abordagem própria de Rede Convolucional de Grafo foi usada em Alarab, Prakoonwit e Nacer (2020) para lidar com a base de dados desenvolvida por Weber et al. (2019). A abordagem utilizada modifica a Rede Convolucional de Grafo padrão para lidar com grafos direcionados, através do Laplaciano normalizado simétrico. Foi obtida uma precisão para a classe "ilícito" de 0,899, porém uma revocação para a mesma classe de 0,624. A abordagem também não levou em consideração o caráter temporal da base de dados. Os autores sugeriram ainda que a inclusão das datas e horas reais das transações poderiam ser muito mais significativas para o aprendizado que os *timesteps* colocados na base de dados.

Uma abordagem para avaliar o desempenho de Redes Neurais de Grafo para detecção de fraude foi feita por Pereira e Murai (2021). Foram testados alguns modelos comuns de Redes Neurais de Grafo (como o GCN), dois algoritmos clássicos de *boosting* (*XGBoost* e *CatBoost*), EvolveGCN (PAREJA et al., 2020) (um modelo de GCN no qual foi adicionada a capacidade de lidar com memória a longo prazo) e PC-GNN (LIU et al., 2021), que usa técnicas de amostragem para lidar com o desequilíbrio de classe durante a propagação da mensagem. As bases de dados testadas foram Elliptic (WEBER et al., 2019), YelpChi (RAYANA; AKOGLU, 2015), e conjuntos de dados gerados pelo simulador AMLSim (SUZUMURA; KANEZASHI, 2021). Para adaptar os dados do AMLSim, que têm o formato de transações bancárias, para um grafo em que os vértices são transações bancárias, as transações foram conectadas de forma que se ligassem a outras transações do grafo em que a conta de origem ou destino fosse a mesma. O melhor desempenho, em se tratando de F1 para a classe "ilícito" foi obtido pelo modelo GraphSAINT (ZENG et al., 2020) sobre a base AMLSim (F1 de 0,975$\pm$0,041). Porém, o modelo que melhor lidou com o desbalanceamento de classe foi o EvolveGCN, também sobre a base AMLSim (F1 de 0,826 ± 0,037). No entanto, foi constatado que nenhum modelo conseguiu lidar de maneira satisfatória com a presença do desvio de conceito.

Uma nova abordagem que utiliza *multi-task learning* para a geração de *embeddings* foi proposta por Assumpção et al. (2022). O sistema, conhecido como DELATOR, utiliza o modelo

GraphSAGE para a obtenção dos *embeddings* através de ambos aprendizado não supervisionado e regressão, que são combinados para se gerar os *embeddings* de saída. Para a classificação, foram testados tanto um classificador único quanto dois classificadores combinados (o segundo classificador "filtra" os casos considerados suspeitos pelo primeiro, buscando uma maior precisão). O delator foi testado em uma base de dados real, com transações do banco Inter, sendo que os resultados obtidos foram maiores do que os *baselines* comparados.

O Quadro 3.1 lista os trabalhos citados neste capítulo e resume sua principal contribuição, comparando-os com este trabalho.

Quadro 3.1 – Trabalhos relacionados e suas principais contribuições.

Trabalho	Arquitetura	Dados utilizados	Principal contribuição
Weber et al. (2019)	MLP, GCN, Skip-GCN e EvolveGCN	Base de dados de criptomoedas coletada da Bitcoin Blockchain (Elliptic).	Criação da base de dados Elliptic e testagem de um modelo recorrente de GNN.
Farrugia, Ellul e Azzopardi (2020)	XGBoost (não utilizou RNA).	Base de dados de criptomoedas coletada da Ethereum Blockchain.	Criação da base de dados Ethereum.
Alarab, Prakoonwit e Nacer (2020)	Arquitetura própria de GCN	Base de dados de criptomoedas coletada da Bitcoin Blockchain (Elliptic).	Desenvolvimento de uma variante própria da GCN para lidar com grafos direcionados.
Pereira e Murai (2021)	GCN, EvolveGCN, PC-GNN e GraphSAINT	Dados bancários simulados, gerados via AMLSim.	Testagem de diferentes modelos no contexto de detecção de Lavagem de Dinheiro, inclusive alguns modelos que lidam com o desequilíbrio de classe.
Assumpção et al. (2022)	GraphSAGE	Dados bancários reais, coletados pelo Banco Inter.	Desenvolvimento de um método *multi-task learning* utilizando GraphSAGE e classificação feita através de dois classificadores combinados.
Nosso trabalho	**GCN, Skip-GCN e NENN**	**Dados bancários simulados, gerados via AMLSim.**	**Testagem de uma arquitetura que suporta atributos de nós e arestas.**

Fonte: Do autor (2023).

A maioria dos trabalhos relacionados citados neste capítulo utilizam bases de dados de transações feitas em criptomoedas, enquanto o foco deste trabalho são as transações bancárias. Os trabalhos que utilizam transações bancárias não divulgam o seu conjunto de dados, uma

vez que a natureza de tais dados é sigilosa. Este trabalho utiliza dados gerados pelo simulador AMLSim, utilizado por Pereira e Murai (2021). Além disso, avalia as técnicas de classificadores combinados proposta por Assumpção et al. (2022). No entanto, diferentemente dos trabalhos citados, este trabalho avalia a arquitetura NENN, que incorpora atributos de nós e arestas, no contexto da detecção de Lavagem de Dinheiro. A metodologia adotada é detalhada no capítulo seguinte (CAPÍTULO 4).

4 METODOLOGIA

Este capítulo descreve os experimentos realizados neste trabalho. A Seção 4.1 descreve como foi feita a obtenção dos conjuntos de dados e as transformações neles realizadas. As métricas utilizadas neste trabalho estão dispostas na Seção 4.2. Inicialmente, foram feitos testes utilizando um conjunto de parâmetros *default* para cada modelo, que estão descritos na Seção 4.3. Em seguida, foi feita uma otimização destes parâmetros para o conjunto de dados com maior desbalanceamento de classe, que está detalhada na Seção 4.4. Por fim, como mostrado na Seção 4.5, foram feitos testes com dois classificadores, com o conjunto de parâmetros já otimizado. Todos os experimentos realizados foram repetidos 100 vezes e as métricas foram obtidas com um intervalo de confiança de 95%.

4.1 Conjuntos de dados

Os dados utilizados neste trabalho foram gerados por um simulador de transações financeiras baseado em agentes, denominado AMLSim (SUZUMURA; KANEZASHI, 2021) que tem sido utilizado em diversos trabalhos na literatura (WEBER et al., 2018; PEREIRA; MURAI, 2021). O AMLSim cria, inicialmente, um grafo de transações através de um Gerador de Grafo de Transações (*Transaction Graph Generator* – TGG) e executa sobre o grafo criado um Simulador de Transações (*Transaction Simulator* – TS), gerando transações lícitas e ilícitas. O TGG gera, a partir de um arquivo de listas de contas, os vértices do grafo de transação. Sobre esse conjunto de vértices ele gera um grafo de transações base utilizando um arquivo de graus de distribuição de parâmetros. Por fim, sobre esse grafo base, ele gera o grafo de transações completo, através de um arquivo de parâmetros de alerta. O TS faz com que os agentes associados às contas financeiras realizem, a cada iteração, transações com seus respectivos vizinhos.

Através do AMLSim, é possível configurar a quantidade total de *timesteps* (dias) da simulação, a data de início, o valor mínimo e máximo das transações, entre outros parâmetros estatísticos. Para este estudo, foram geradas transações diárias para o ano de 2020 (365 *timesteps*). Durante a simulação, são geradas informações sobre o identificador da conta, se é uma conta suspeita a priori, suas datas de abertura e fechamento, o depósito inicial, o identificador do padrão de comportamento, o identificador do banco, entre outros. As informações referentes à transação são o seu identificador, as contas de origem e destino, o tipo de transação, o dia, e

se a transação é ou não suspeita. Para este trabalho, as informações utilizadas foram somente o valor transferido (*base_amt*), *timestep* (*tran_timestamp*) e tipo de transação (*tx_type*), para cada transação (além da informação sobre sua ilicitude (*is_sar*) e a conta de origem e destino, *orig_acct* e *bene_acct*, usadas na construção dos grafos), e informações de depósito inicial (*initial_deposit*) e categoria de comportamento (*tx_behavior_id*), para cada conta bancária. Uma amostra dos dados de transações e contas financeiras gerados pelo AMLSim, somente com os atributos utilizados neste trabalho, pode ser vista nas tabelas 4.1 e 4.2.

Tabela 4.1 – Amostra de dados de transações bancárias geradas pelo AMLSim.

tran_id	**orig_acct**	**bene_acct**	**tx_type**	**base_amt**	**tran_timestamp**	**is_sar**
1	743	409	TRANSFER	727.06	2020-01-01	False
2	883	751	TRANSFER	397.13	2020-01-01	False
3	993	678	CASH_IN	933.56	2020-01-01	False
18	774	217	TRANSFER	199.22	2020-01-01	True
19	774	883	DEPOSIT	199.22	2020-01-01	True

Fonte: Do autor (2023).

Tabela 4.2 – Amostra de dados de contas bancárias geradas pelo AMLSim.

acct_id	**initial_deposit**	**tx_behavior_id**
7	66450.68	1
8	71285.46	1
9	93684.83	1
10	52558.76	1
11	70139.99	1

Fonte: Do autor (2023).

De maneira similar ao que foi realizado por (PEREIRA; MURAI, 2021), os dados gerados foram posteriormente selecionados de forma a variar em cada conjunto a taxa de desbalanceamento de classe, para que fosse possível testar a performance de cada modelo em diferentes taxas de desbalanceamento. A Tabela 4.3 mostra cada conjunto de dados, sua respectiva taxa de desbalanceamento e a quantidade de exemplos que cada classe possui. Vale ressaltar que, dado que os conjuntos de transações do mundo real possuem taxas consideravelmente baixas de desbalanceamento, o conjunto AMLSim 1/20 é o conjunto de dados que mais se aproxima do mundo real.

Os dados gerados foram então normalizados via *MinMax*, um método de normalização que redimensiona os valores para o intervalo $[0, 1]$. Os dados categóricos foram convertidos em

Tabela 4.3 – Informações sobre os conjuntos de dados utilizados.

Conjunto de dados	# de transações lícitas	# de transações ilícitas	Total de transações	Taxa de desbalanceamento
AMLSim 1/3	2142	1071	3213	3
AMLSim 1/5	4284	1071	5355	5
AMLSim 1/10	9639	1071	10710	10
AMLSim 1/20	20349	1071	21420	20

Fonte: Do autor (2023).

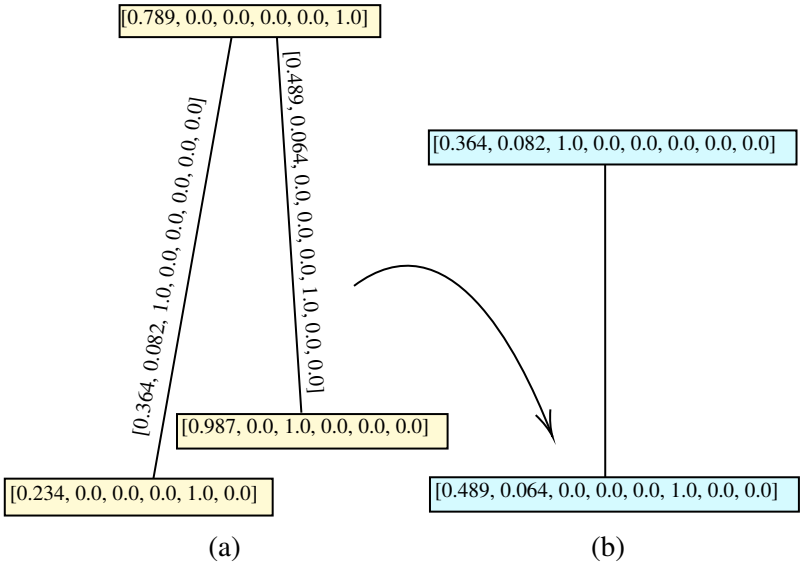
múltiplas tabelas de dados binários via *one-hot-encoding*, ou seja, cada um correspondendo a uma categoria, e, para determinado atributo pertencente a uma determinada categoria, atribui-se 1 para a coluna referente àquela categoria e 0 para as demais. Os dados foram então convertidos para grafos. Para a testagem nas arquiteturas GCN e *Skip*-GCN, os vértices do grafo são as transações e dois vértices (ou transações) estão ligadas por uma aresta se tiverem alguma conta em comum entre elas, seja como depositante ou como beneficiário do depósito.

Para a testagem na arquitetura NENN, cada vértice do grafo é uma conta financeira e as arestas são as transações entre as diferentes contas. Dessa forma, os dados relativos as contas financeiras são utilizados somente nos testes feitos com a NENN. Todas as transações em todos os conjuntos de dados estão classificadas como "lícita" (0) ou "ilícita" (1). A Figura 4.1 ilustra a diferença entre os tipos de representações geradas para cada conjunto de dados. É possível afirmar que a representação na qual a conta bancária é o vértice do grafo é o grafo aresta da representação na qual a transação é o vértice do grafo.

4.2 Métricas

As métricas utilizadas neste trabalho foram Precisão (Equação 4.1), Revocação (Equação 4.2) e F1 (Equação 4.3). As equações estão descritas em função do número de verdadeiros positivos (*true positives* – TP), falsos positivos (*false positives* – FP) e falsos negativos (*false negatives* – FN). Todas as métricas foram calculadas considerando ambas as classes (macro), e também somente para a classe "ilícito". A Precisão mostra a capacidade de discriminação por parte do modelo, uma vez que ela evidencia a proporção de instâncias atribuídas a uma classe que foram corretamente rotuladas. Já a Revocação mostra o quanto um modelo consegue cobrir as instâncias de uma classe, uma vez que ela traz a proporção de instâncias pertencentes uma classe que foram corretamente classificadas. Para comparar os resultados, este trabalho foca na

Figura 4.1 – Diferença entre as representações via grafo do conjunto de dados utilizado.



Conta bancária como um vértice do grafo (a) e transação como vértice do grafo (b).
Fonte: Do autor (2023)

F1 da classe "ilícito", uma vez que ela consegue avaliar tanto a precisão quanto a revocação das classificações para esta classe. Ambas as métricas seriam importantes numa aplicação real, uma vez que, ao passo que se deseja obter o maior número possível de transações ilícitas, também é desejável que o número de falsos positivos seja o menor possível.

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.1}$$

$$\text{Revocação} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.2}$$

$$\text{F1} = 2 * \frac{\text{Precisão} * \text{Revocação}}{\text{Precisão} + \text{Revocação}}. \tag{4.3}$$

4.3 Experimentos com vários níveis de desbalanceamento

Para os experimentos iniciais realizados neste trabalho, foram utilizadas as arquiteturas GCN, Skip-GCN e NENN com uma camada totalmente conectada e um Softmax para fazer a classificação. Além disso, o algoritmo *XGBoost* foi treinado e testado com os *embeddings* gerados por essas três arquiteturas. Tanto a GCN quanto a Skip-GCN utilizadas possuem duas

camadas e 16 neurônios nas camadas escondidas. A NENN foi construída com duas camadas de atenção de arestas (no início e no fim do conjunto de camadas escondidas) e uma camada de atenção de nós (no centro do conjunto de camadas escondidas).

Para as três arquiteturas o otimizador utilizado foi o *Adam*, sendo que a taxa de aprendizagem para as arquiteturas baseadas em GCN foi fixada em $1,0 \times 10^{-3}$, enquanto para a arquitetura NENN foi estabelecida em $1,0 \times 10^{-4}$. Foi utilizada a Entropia Cruzada com Pesos como função de erro. Para as arquiteturas GCN e Skip-GCN, os pesos foram fixados em 0,7 para a classe "ilícito" e 0,3 para a classe "lícito". Esses valores de hiperparâmetros foram obtidos experimentalmente, utilizando como base valores de outros trabalhos relacionados (WEBER et al., 2019; PEREIRA; MURAI, 2021) e alterando esses valores por outros próximos, verificando se havia melhoria nos resultados. A arquitetura NENN se mostrou consideravelmente sensível à variação da taxa de desbalanceamento e, por isso, foram fixados pesos diferentes para cada conjunto de dados testado. Para o AMLSim 1/3 não foram estabelecidos pesos. Para o AMLSim 1/5 foi fixado 0,6 para a classe "ilícito" e 0,4 para a classe "lícito". Para o AMLSim 1/10 foi fixado 0,85 para a classe "ilícito" e 0,15 para a classe "lícito". Por fim, para o AMLSim 1/20 foi fixado 0,96 para a classe "ilícito" e 0,04 para a classe "lícito". Experimentaram-se vários pesos diferentes para cada conjunto de dados até que fosse encontrado um ponto de equilíbrio entre a precisão e a revocação do modelo.

Os testes foram implementados em Python 3.8, sendo que as bibliotecas utilizadas para implementação dos modelos foram Pytorch 1.10 e Pytorch-Geometric 2.0.2. O XGBoost foi implementado com a biblioteca XGBoost 1.6.1, e as métricas foram coletadas via Scipy 1.8 e Scikit-learn 1.0.2. Os códigos utilizados encontram-se disponíveis *on-line* para fins de estudo[1].

4.4 Otimização de hiperparâmetros

A Otimização de Hiperparâmetros, ou Hiperparametrização, consiste na busca automática pela melhor configuração dos parâmetros iniciais dos modelos de ML, i. e. aqueles parâmetros que são definidos na criação e configuração inicial do modelo, antes do treinamento, denominados hiperparâmetros. (FEURER; HUTTER, 2019). Neste trabalho, cada uma das combinações entre modelos de GNN e classificadores foi submetida a uma Otimização de Hi-

[1] Disponível em: <https://github.com/italodellagarza/GCNMoneyLaundering>. Acesso em: 017/05/2023

perparâmetros considerando somente o conjunto de dados AMLSim 1/20 (o conjunto de dados mais realista).

O conjunto de dados foi dividido em treino, validação e teste. A otimização para cada modelo foi feita utilizando o método de busca denominado *Random Search*. Esse método consiste de múltiplas iterações, sendo que cada iteração neste trabalho foi procedida da seguinte maneira: uma configuração de hiperparâmetros é gerada aleatoriamente e usada para criar um modelo. O modelo então é treinado no conjunto de treino e testado no conjunto de validação 5 vezes, e a F1 média para a classe ilícito é obtido. Após as iterações, a melhor configuração de hiperparâmetros encontrada foi treinada com os conjuntos de treino e validação e testada no conjunto de teste. As combinações de GNN com Softmax foram otimizadas com 100 iterações, ao passo que as combinações com XGBoost, por sua otimização incluir hiperparâmetros do próprio XGBoost, foram otimizadas com 200 iterações. A implementação do *Random Search* utilizada neste trabalho foi feita utilizando a biblioteca Ray Tune 2.1.0 (LIAW et al., 2018).

Nos modelos GCN e Skip-GCN, os hiperparâmetros otimizados foram a taxa de aprendizado, o número de neurônios na camada escondida e o peso da classe "ilícito" no cálculo do erro durante o treinamento. No modelo NENN, além do peso da classe ilícito e da taxa de aprendizado, foram otimizados os tamanhos de *embedding* de saída para o nó e para a aresta. Nas combinações com XGBoost, além dos hiperparâmetros dos próprios modelos, também foram otimizados a profundidade máxima da árvore de decisão do XGBoost e sua taxa de aprendizagem (η). Os hiperparâmetros otimizados em cada arquitetura juntamente com seu espaço de busca estão listados na Tabela 4.4.

4.5 Combinações de classificadores

Outro teste realizado consistiu em realizar a classificação através de combinações entre classificadores. Inspirado por Assumpção et al. (2022), cada modelo foi testado com um primeiro classificador, responsável por realizar uma classificação inicial das transações, e um segundo classificador, executado sobre as transações do primeiro classificador que foram classificadas como "ilícitas", com o objetivo de filtrar as transações rotuladas como "ilícitas" e deixar o modelo mais preciso. Para o primeiro classificador foi utilizado o Softmax (por ser uma

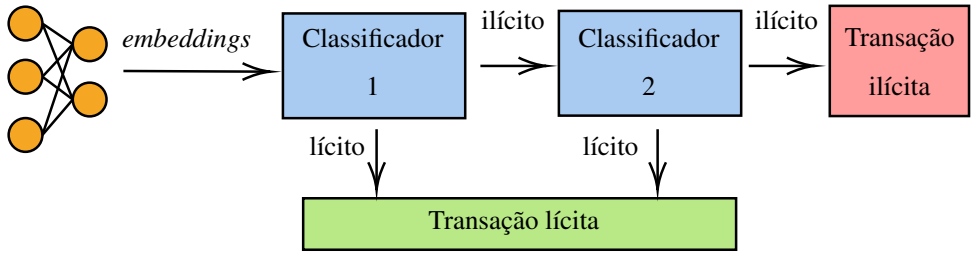
Tabela 4.4 – Hiperparâmetros otimizados em cada arquitetura.

Arquitetura	Hiperparâmetro	Intervalo para busca
GCN e Skip-GCN	Taxa de aprendizado	$[1 \times 10^{-2}, 1 \times 10^{-4}]$
	# de neurônios na camada escondida	[8; 28]
	Peso da classe "ilícito"	[0,6; 1,0]
NENN	Tamanho do embedding do nó	[3, 12]
	Tamanho do embedding da aresta	[4, 16]
	Peso da classe "ilícito"	[0,8; 0,9]
	Taxa de aprendizado	$[1 \times 10^{-3}, 1 \times 10^{-5}]$
XGBoost*	Profundidade máxima da árvore de decisão	[3, 10]
	Taxa de aprendizado do XGBoost (η)	[0,01; 0,2]

Fonte: Do autor (2023).

camada diretamente ligada à saída da rede), e para o segundo o XGBoost. Esse método de combinação de dois classificadores é ilustrado pela Figura 4.2.

Figura 4.2 – Classificação utilizando dois classificadores.



Fonte: Do Autor (2023).

Foram utilizadas nesse teste as configurações de hiperparâmetros otimizadas na Seção 4.4. Os hiperparâmetros utilizados nas GNN foram os resultados ótimos obtidos com o Softmax, ao passo que os hiperparâmetros utilizados no XGBoost foram os obtidos na sua combinação com a respectiva GNN. Ou seja, os hiperparâmetros para a GCN, por exemplo, são os hiperparâmetros ótimos da combinação GCN + Softmax, e os hiperparâmetros para o XGBoost utilizado como o segundo classificador são os hiperparâmetros ótimos da combinação GCN + XGBoost.

Adotando a metologia descrita neste capítulo, foram obtidos os resultados listados e discutidos no capítulo seguinte (CAPÍTULO 5).

5 RESULTADOS E DISCUSSÃO

Este capítulo descreve os resultados obtidos para os experimentos realizados. A Seção 5.1 descreve os resultados obtidos nos testes iniciais, com parâmetros fixos e somente um classificador. Os resultados obtidos após a otimização de hiperparâmetros estão presentes na Seção 5.2. Os resultados obtidos nos testes com múltiplos classificadores estão presentes na Seção 5.3. Por fim, o capítulo se encerra com uma discussão sobre os resultados na Seção 5.4.

5.1 Resultados para vários níveis de desbalanceamento

A partir dos experimentos descritos na Seção 4.3, foi coletada a precisão, revocação e F1, tanto no nível macro como para a classe "ilícito", bem como seus respectivos intervalos de confiança (IC), com o nível de confiança de 95%. As métricas resultantes para cada modelo do experimento e separadas pelo conjunto de dados estão dispostas na Tabela 5.1. Em todos os resultados descritos neste capítulo, quando não se menciona o classificador, foi utilizada a função Softmax.

Nota-se que, dentre todas métricas obtidas nos experimentos realizados nas bases de dados com taxa de desbalanceamento mais baixas, i.e., AMLSim 1/3 e AMLSim 1/5, os modelos baseados em GCN se mostraram superiores, com destaque para a combinação GCN + XGBoost, que obteve maior precisão e F1 para esses dois conjuntos de dados. Também vale destacar as revocações obtidas pela arquitetura Skip-GCN, que foram mais altas que as demais, indicando que essa arquitetura é que obteve melhor cobertura para os conjuntos de dados supracitados, ainda que não tenha sido a mais precisa ao classificá-los. A arquitetura NENN obteve métricas mais baixas para esses conjuntos de dados (ainda que tenham sido satisfatórias), sobretudo nos testes em que a classificação foi realizada com o Softmax. Dessa forma, nota-se que a representação das transações como nós do grafo pode ter resultados positivos, ao menos para as taxas de desbalanceamento mais baixas.

O efeito do desbalanceamento começa a ser perceptível quando a taxa de desbalanceamento é igual a 10, sobretudo nos valores de precisão para a classe "ilícito". No cenário com as taxas de desbalanceamento mais altas, i.e, AMLSim 1/10 e AMLSim 1/20, a combinação NENN + XGBoost obteve precisão e F1 superiores às demais, exceto para o conjunto AMLSim 1/20, para o qual a maior precisão foi obtida pela combinação Skip-GCN + XGBoost. A ar-

Tabela 5.1 – Métricas coletadas nos testes sobre os conjuntos de dados.

Base de Dados	**Arquitetura**	**Precisão**		**Revocação**		**F1**	
		macro	classe "ilícito"	macro	classe "ilícito"	macro	classe "ilícito"
AMLSim 1/3	GCN	95,31±0,12	91,30±0,23	97,27±0,06	98,43±0,00	96,21±0,09	94,73±0,13
	Skip-GCN	95,34±0,17	91,35±0,34	**97,28±0,08**	**98,44±0,00**	96,23±014	94,76±0,18
	NENN	94,15±1,49	89,36±2,34	96,34±1,70	97,52±3,00	95,13±1,54	93,26±2,14
	GCN+XGBoost	**95,79±0,65**	**92,80±0,99**	96,98±1,00	97,10±2,03	**96,35±0,77**	**94,89±1,10**
	Skip-GCN+XGBoost	95,74±0,63	92,72±1,14	96,94±0,89	97,04±1,86	96,30±0,69	94,83±0,98
	NENN+XGBoost	95,39±1,36	91,79±2,44	97,00±1,24	97,65±2,08	96,14±1,26	94,61±1,74
AMLSim 1/5	GCN	91,67±1,05	83,97±2,14	96,60±0,28	97,32±0,43	93,87±0,73	90,15±1,15
	Skip-GCN	91,70±1,07	84,01±2,16	**96,62±0,30**	**97,35±0,28**	93,90±0,76	90,19±1,19
	NENN	90,34±1,61	82,34±3,10	94,14±1,60	92,70±3,18	92,07±1,42	87,20±2,28
	GCN+XGBoost	**93,13±1,26**	**87,76±2,39**	95,23±1,48	93,35±2,94	**94,12±1,18**	**90,46±1,93**
	Skip-GCN+XGBoost	93,08±1,24	87,64±2,49	95,23±1,38	93,40±2,89	94,10±1,06	90,42±1,73
	NENN + XGBoost	92,75±2,16	87,32±4,33	94,47±1,41	91,91±2,79	93,57±1,57	89,53±2,52
AMLSim 1/10	GCN	85,78±2,91	72,38±5,85	**94,23±0,50**	91,99±0,94	89,39±2,00	80,98±3,49
	Skip-GCN	85,54±2,80	71,90±5,64	94,20±0,49	92,02±1,04	89,23±1,92	80,69±3,34
	NENN	79,26±2,59	59,19±5,12	93,57±1,71	**93,60±3,17**	84,41±2,40	72,49±4,07
	GCN+XGBoost	88,53±2,57	78,62±5,16	89,76±1,77	84,56±4,03	89,76±1,77	81,44±3,17
	Skip-GCN+XGBoost	88,20±2,52	77,95±5,03	91,16±1,55	84,73±3,16	89,60±1,68	81,17±2,99
	NENN+XGBoost	**90,63±1,76**	**82,79±3,53**	91,44±2,19	84,65±4,49	**91,02±1,49**	**83,68±2,73**
AMLSim 1/20	GCN	83,15±3,08	67,32±6,17	88,20±1,01	78,19±2,04	85,45±1,90	72,31±3,56
	Skip-GCN	83,09±2,87	67,20±5,73	88,22±0,82	78,24±1,62	85,42±1,80	72,27±3,36
	NENN	65,36±2,81	31,10±5,63	**91,42±3,14**	**92,57±6,71**	70,59±3,87	46,46±6,30
	GCN+XGBoost	89,92±4,27	81,31±8,35	83,86±3,15	68,45±6,10	86,60±3,33	74,29±6,39
	Skip-GCN+XGBoost	**89,98±3,46**	**81,45±6,86**	83,67±2,06	68,06±4,05	86,51±2,19	74,12±4,20
	NENN+XGBoost	88,21±3,17	77,74±6,34	85,35±2,90	71,66±5,89	**86,69±2,20**	**74,51±4,21**

Fonte: Do autor (2023).

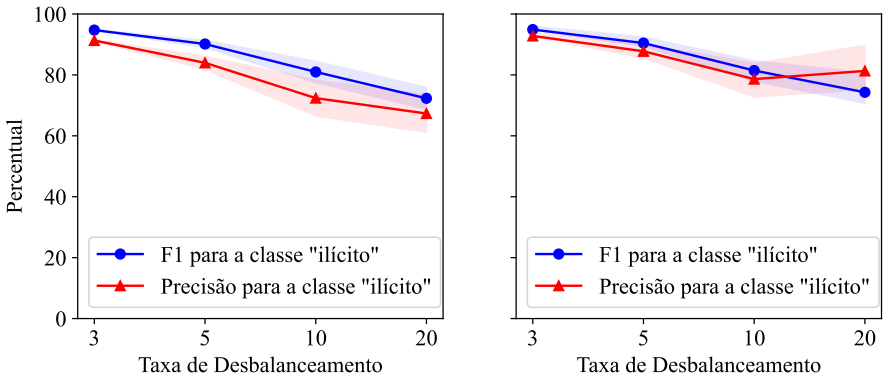
quitetura NENN combinada com a classificação por Softmax atingiu a melhor macro-revocação para o conjunto de dados AMLSim 1/20 e a melhor revocação para a classe “ilícita” no conjunto AMLSim 1/10. No entanto, a arquitetura apresenta baixa precisão em ambos os conjuntos de dados, o que indica que ela está atribuindo muitas instâncias para a classe “ilícito”, sem necessariamente filtrar por aquelas que realmente são ilícitas. A baixa precisão dessa arquitetura afetou também a F1 do modelo, piorando-o em relação aos demais. Por fim, A GCN combinada com Softmax atingiu a melhor macro-revocação no conjunto AMLSim 1/10. É notável que a representação das transações como arestas do grafo se saiu melhor para taxas de desbalanceamento maiores.

A adição do XGBoost em todos os modelos provocou um aumento expressivo nos resultados. O XGBoost é um modelo que geralmente tem resultados positivos sobre dados tabulares (GRINSZTAJN; OYALLON; VAROQUAUX, 2022) e sua inclusão nos modelos incrementou a capacidade dos mesmos ao lidar com a complexidade dos conjuntos de dados e sua resistência

ao desequilíbrio de classe. Considerando-se os resultados sobre os conjuntos de dados AML-Sim 1/10 e AMLSim 1/20, nota-se que a adição do XGBoost à arquitetura NENN fez com que sua F1, que de maneira geral era a pior em relação aos outros modelos, se tornasse a melhor.

A Figura 5.1 mostra o impacto do desbalanceamento de classe na F1 e na precisão para a classe "ilícito" para o modelo GCN, com e sem XGBoost. Nota-se uma queda significativa em ambas as métricas quando a taxa de desbalanceamento sobe de 5 para 10. Além disso, destaca-se o efeito positivo do uso do XGBoost como classificador para valores mais altos de desbalanceamento de classe, sobretudo na precisão.

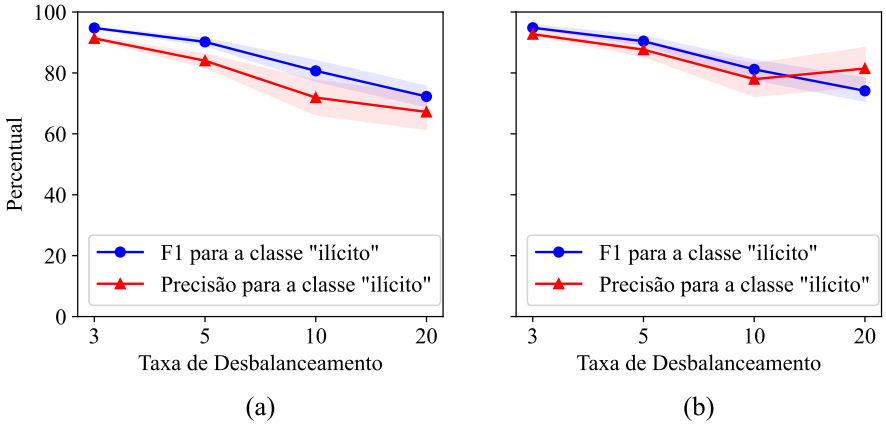
Figura 5.1 – F1 e Precisão para a classe "ilícito" do modelo GCN



Resultados para os classificadores Softmax (a) e XGBoost (b). (IC = 95%).
Fonte: Do Autor (2023).

Os efeitos do desbalanceamento de classe na F1 e na precisão para a classe "ilícito" para o modelo Skip-GCN são evidenciados pela Figura 5.2. Nota-se que as métricas foram muito similares às obtidas pelo modelo GCN, de forma que percebe-se o mesmo efeito positivo da adição do XGBoost ao modelo.

Figura 5.2 – F1 e Precisão para a classe "ilícito" do modelo Skip-GCN.

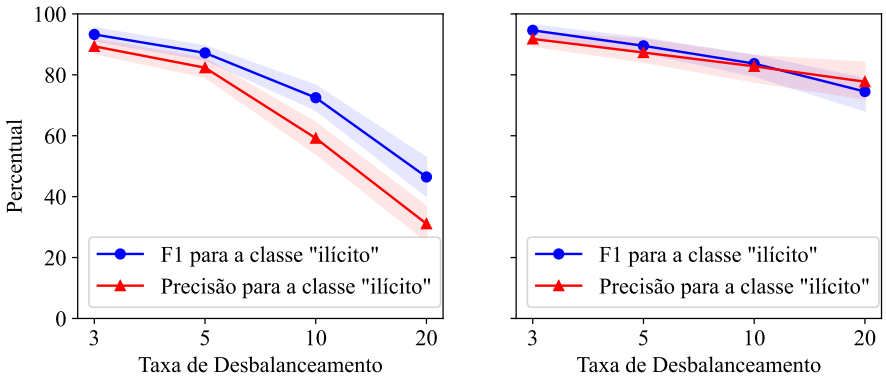


Resultados para os classificadores Softmax (a) e XGBoost (b). (IC = 95%).
Fonte: Do Autor (2023).

A Figura 5.3 mostra o efeito do desbalanceamento de classe na precisão e F1 para a classe "ilícito" do modelo NENN, com e sem a adição do XGBoost. Nesse modelo, quando o XGBoost não é usado, percebe-se uma queda mais acentuada nas métricas obtidas em relação

aos demais modelos. Logo, o modelo NENN, quando utilizado juntamente com o Softmax para classificação, se mostrou, no cenário abordado por este estudo, mais sensível aos efeitos do desbalanceamento de classe. Porém, com a adição do XGBoost, o modelo se mostra resistente a esses efeitos, inclusive superando os demais em termos de F1 e precisão. Pode-se dizer que a combinação NENN + XGBoost é a mais indicada para a tarefa de detecção apresentada neste estudo.

Figura 5.3 – F1 e Precisão para a classe "ilícito" do modelo NENN.



Resultados para os classificadores Softmax (a) e XGBoost (b). (IC = 95%).
Fonte: Do Autor (2023).

5.2 Resultados após a otimização de hiperparâmetros

Os resultados calculados para o conjunto AMLSim 1/20 após a otimização de hiperparâmetros, descrita na Seção 4.4, se encontram na Tabela 5.2. É possível notar uma melhoria geral nos resultados dos modelos, indicando que um efeito positivo da hiperparametrização. Também observa-se que a diferença entre os resultados dos diferentes modelos foi reduzida. O modelo NENN, por exemplo, quando utilizado com o classificador Softmax, anteriormente estava com uma revocação consideravelmente alta e uma precisão consideravelmente baixa, indicando um forte desequilíbrio. Com a otimização de hiperparâmetros, ao passo em que a revocação sofreu uma pequena diminuição, a precisão aumentou consideravelmente.

Com a otimização, o "melhor" modelo passou a ser a Skip-GCN, por ter as maiores F1, tanto em nível macro quanto para a classe "ilícito". Da mesma forma, o modelo com maior

Tabela 5.2 – Resultados após a otimização de hiperparâmetros para o conjunto AMLSim 1/20.

Arquitetura	**Precisão**		**Revocação**		**F1**	
	macro	classe "ilícito"	macro	classe "ilícito"	macro	classe "ilícito"
GCN	87,45±3,03	76,10±6,16	87,64±1,26	76,52±2,84	87,52±1,25	76,25±2,32
Skip-GCN	89,84±2,04	81,00±4,14	86,66±1,32	74,22±2,77	**88,17±0,95**	**77,43±1,82**
NENN	84,70±1,73	70,25±3,48	**89,70±3,35**	**80,93±6,85**	86,99±1,75	75,16±3,38
GCN+XGBoost	**94,77±2,44**	**91,27±4,94**	82,58±1,94	65,48±3,97	87,59±1,33	76,21±2,57
Skip-GCN+XGBoost	93,11±3,65	88,11±7,43	80,91±3,25	62,25±6,68	85,83±2,14	72,84±4,15
NENN+XGBoost	90,85±2,28	83,05±4,68	84,30±3,44	69,23±7,04	87,22±1,91	75,43±3,72

Fonte: Do autor (2023).

precisão passou a ser a combinação GCN + XGBoost. Apenas o modelo com maior revocação continuou sendo o mesmo (NENN).

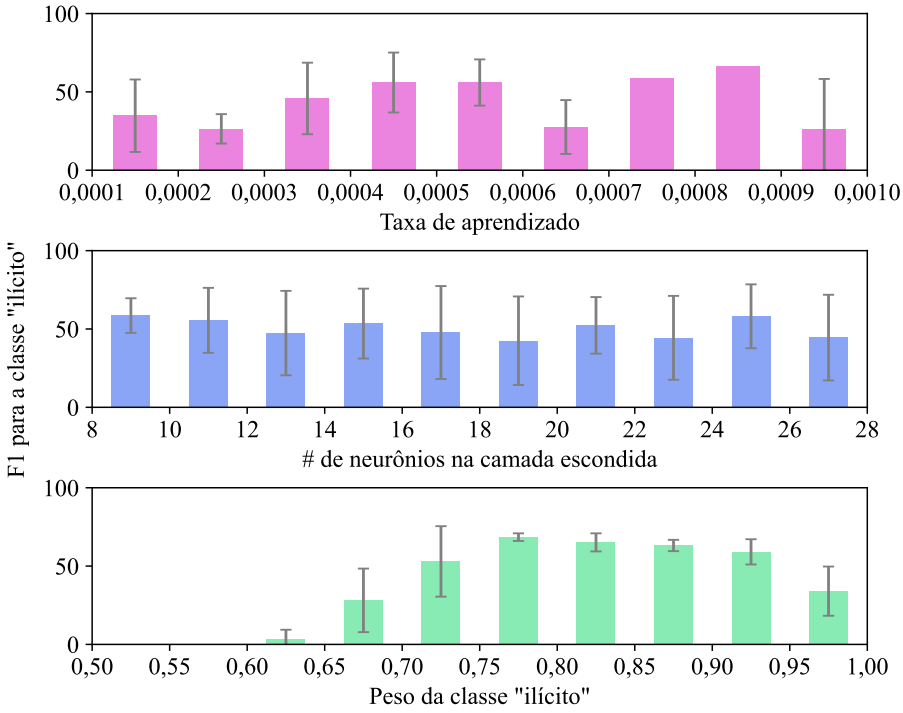
Além das métricas para a otimização de hiperparâmetros, foi também obtida a correlação entre cada hiperparâmetro sobre a F1 para a classe "ilícito", em cada modelo. Isso foi feito calculando-se uma média da F1, obtida durante a otimização, para intervalos de valores da cada hiperparâmetro testado. Com isso, foram gerados os gráficos da Figura 5.4 à Figura 5.9, com uma margem de erro de um desvio-padrão (positivo e negativo).

A Figura 5.4 mostra a F1 para a classe "ilícito" média obtido de acordo com os hiperparâmetros otimizados no modelo GCN, com o classificador Softmax. As taxas de aprendizado que resultaram em melhores F1 estão mais próximas do intervalo entre 0,0060 e 0,0081, aproximadamente. Os valores para número de neurônios na camada escondida mais adequados estão entre 15 e 18. A F1 média se mantém relativamente constante entre os valores 0,75 e 0,95 de peso para a classe "ilícito".

Para o modelo GCN com classificação via Softmax, a melhor configuração de hiperparâmetros foi de 16 neurônios na camada escondida, peso de 0,757966 para a classe "ilícito" e taxa de aprendizado de 0,008078.

A variação da F1 média para a classe "ilícito" para diferentes valores de hiperparâmetros no modelo Skip-GCN com classificação via Softmax é mostrada na Figura 5.5. Observa-se que, para uma taxa de aprendizado acima de 0,004 e menor do que 0,01, a F1 média se mantém constante, em seus patamares mais altos. Para o número de neurônios na camada escondida próximo de 11 ou 22, a F1 também atinge seus maiores valores. O peso para a classe "ilícito" mostra a maior correlação com a F1 média em comparação com os demais hiperparâmetros, sendo que

Figura 5.4 – Influência dos hiperparâmetros na F1 da classe "ilícito" para o conjunto de validação, para o modelo GCN com o classificador Softmax.



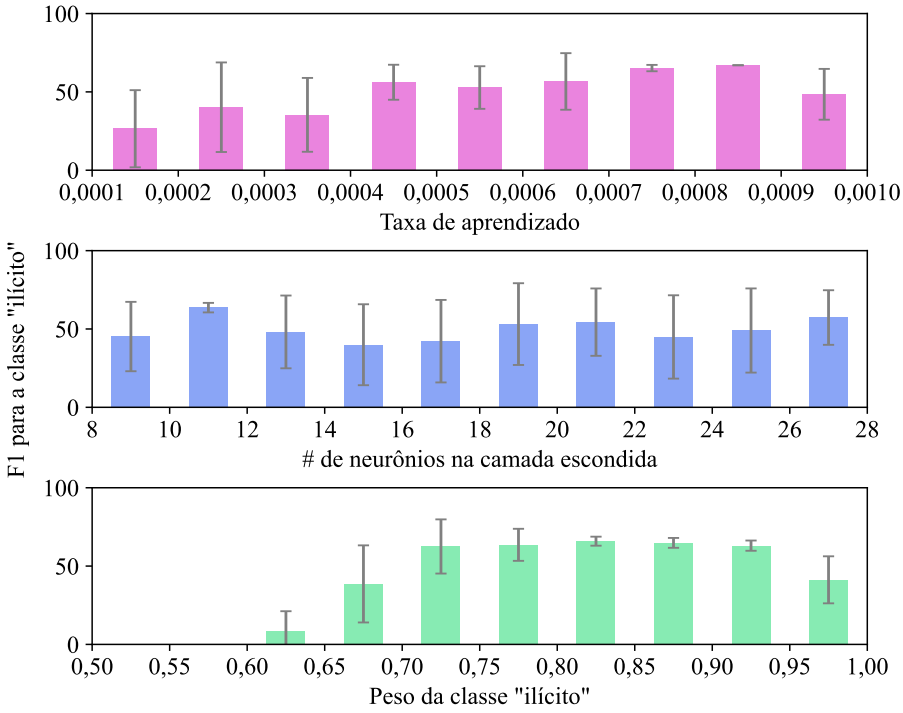
Fonte: Do autor (2023).

os valores que implicam numa maior F1 média estão no intervalo entre 0,72 e 0,90, aproximadamente.

Para o modelo Skip-GCN com classificação via Softmax, uma taxa de aprendizado de 0,006703, um peso para a classe "ilícito" de 0,73601 e 22 neurônios na camada escondida foi a configuração de hiperparâmetros que resultou na F1 mais alta.

A Figura 5.6 mostra os diferentes valores de F1 média para a classe "ilícito" conforme variaram os diferentes valores de hiperparâmetros no processo de otimização, para o modelo NENN com classificação via Softmax. Os valores para taxa de aprendizado que resultaram em maior F1 média estão no intervalo um pouco acima de 0 e um pouco abaixo de 0,0003. Até 9, quanto maior o tamanho do *embedding* do nó, maior é a F1 média para a classe "ilícito", com exceção dos valores 4 e 8. No entanto, o inverso parece ocorrer em relação ao tamanho do *embedding* da aresta, pois em geral quanto mais baixo esse número maior é a F1 resultante (com exceção do valor 6). Quanto ao peso para a classe "ilícito", os valores que resultaram em maiores F1 estão aproximadamente entre 0,87 e 0,90.

Figura 5.5 – Influência dos hiperparâmetros na F1 da classe "ilícito" para o conjunto de validação, para o modelo Skip-GCN com o classificador Softmax.

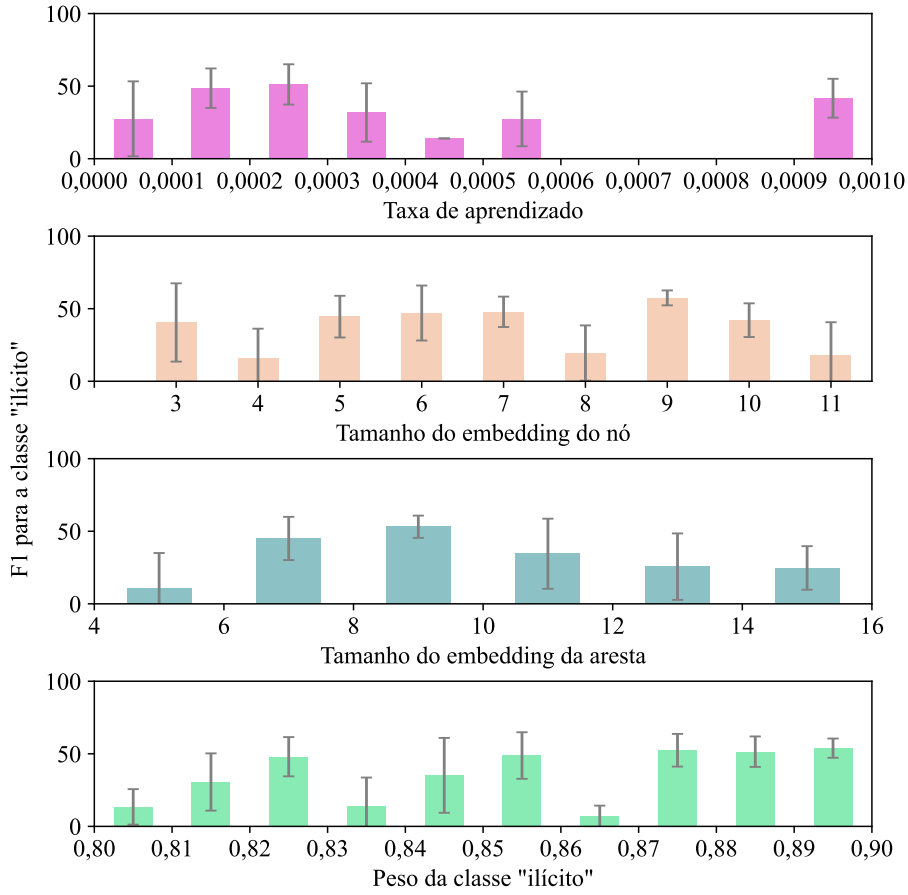


Fonte: Do autor (2023).

Para o modelo NENN com classificação via Softmax, a configuração de hiperparâmetros que resultou na F1 para a classe "ilícito" mais alta foi uma taxa de aprendizado em 0,000363, 5 e 8 para os tamanhos dos *embeddings* de nó e aresta, respectivamente, e um peso para a classe "ilícito" de 0,810695.

A influência dos diferentes valores de hiperparâmetros na F1 média para a classe "ilícito", quando a classificação é feita via XGBoost, é mostrada na Figura 5.7. É Possível notar que, no caso das taxas de aprendizado da rede neural, do XGBoost, e do peso da classe "ilícito", a maior parte dos casos nos quais a F1 se iguala ou aproxima de 0 ocorre quando esses são os menores dentro dos intervalos testados. A F1 atinge seus maiores valores para uma taxa de aprendizado da rede neural entre 0,006 e 0,008, um número de neurônios na camada escondida entre 25 e 27 e uma profundidade máxima da árvore de decisão mais próxima de 9. Para os outros hiperparâmetros, não se vê tanta influência sobre a F1 média, mas sim intervalos grandes dentro do intervalo testado onde há pouca variação da F1.

Figura 5.6 – Influência dos hiperparâmetros na F1 da classe "ilícito" para o conjunto de validação, para o modelo NENN com o classificador Softmax.

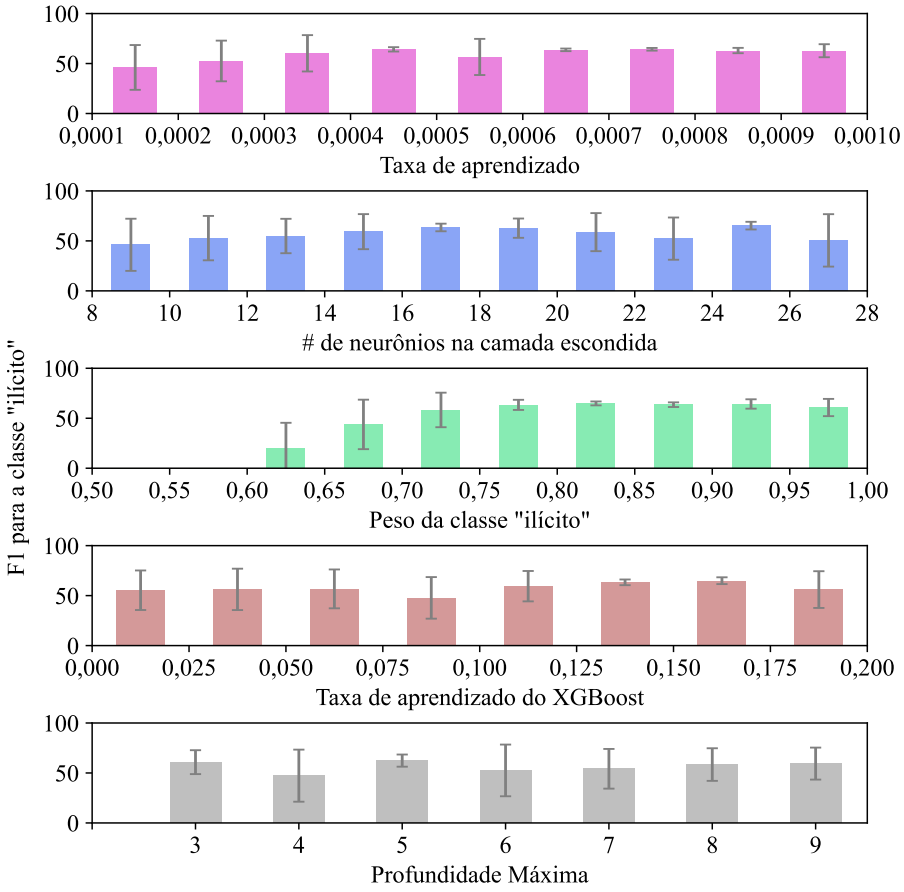


Fonte: Do autor (2023).

Para o modelo GCN com classificação via XGBoost, uma taxa de aprendizado da rede fixada em 0,007664, um peso para a classe "ilícito" de 0,747424, uma taxa de aprendizado do XGBoost de 0,055712, 12 neurônios na camada escondida, e 7 como a profundidade máxima da árvore de decisão resultaram na F1 para a classe "ilícito" mais alta.

A Figura 5.8 mostra a F1 para a classe "ilícito" média obtido de acordo com os hiperparâmetros otimizados no modelo Skip-GCN, com o classificador XGBoost. O comportamento da F1 em relação aos hiperparâmetros de mostrou bastante similar o que foi constatado na otimização feita com a combinação GCN + XGBoost, com relação aos intervalos nos quais há estabilidade da F1 ou quando ela atinge seus valores mais altos. No entanto, para a taxa de aprendizado do XGBoost, vê-se uma instabilidade maior, com o surgimento de valores 0 para diversas taxas de aprendizado ao longo do intevalo testado. Isso pode ser explicado pelo fato

Figura 5.7 – Influência dos hiperparâmetros na F1 da classe "ilícito" para o conjunto de validação, para o modelo GCN com o classificador XGBoost.

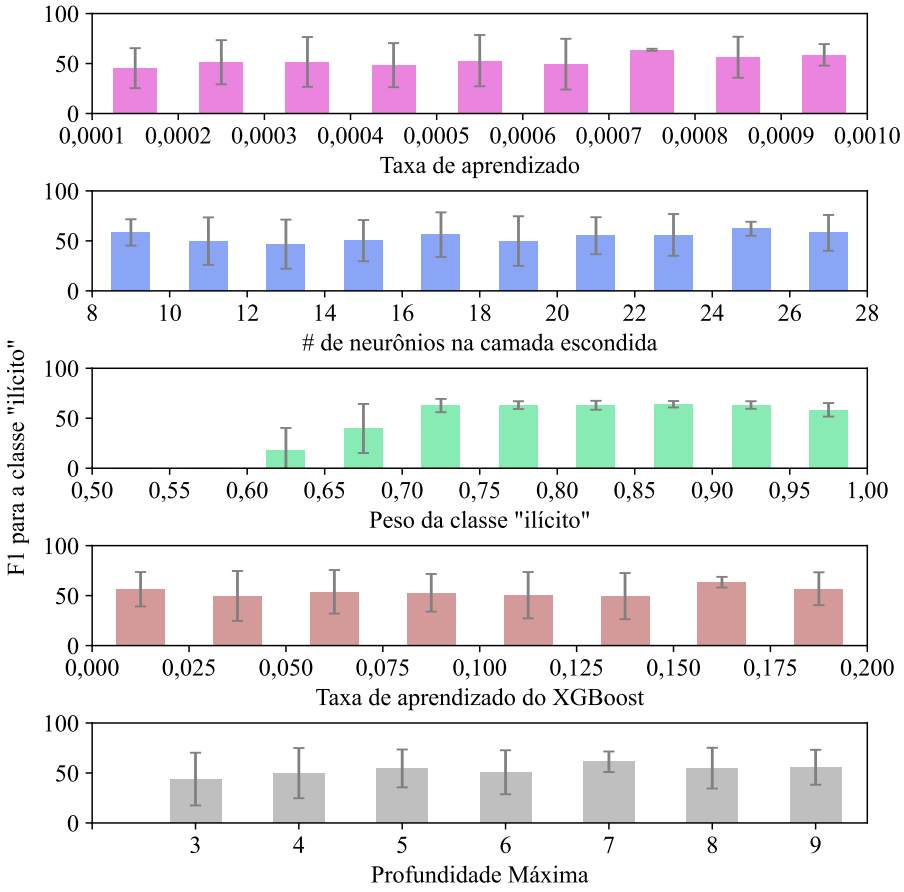


Fonte: Do autor (2023).

de diversos valores de taxa de aprendizagem do XGBoost terem sido testados em configurações nas quais outros hiperparâmetros influenciaram negativamente a F1 média.

Para o modelo Skip-GCN com classificação via XGBoost, a melhor configuração de hiperparâmetros foi uma taxa de aprendizado da rede de 0,005559, uma taxa de aprendizado do XGBoost de 0,040558, um peso para a classe "ilícito" fixado em 0,914568, uma profundidade máxima de 4, e 27 neurônios na camada escondida.

Figura 5.8 – Influência dos hiperparâmetros na F1 da classe "ilícito" para o conjunto de validação, para o modelo Skip-GCN com o classificador XGBoost.

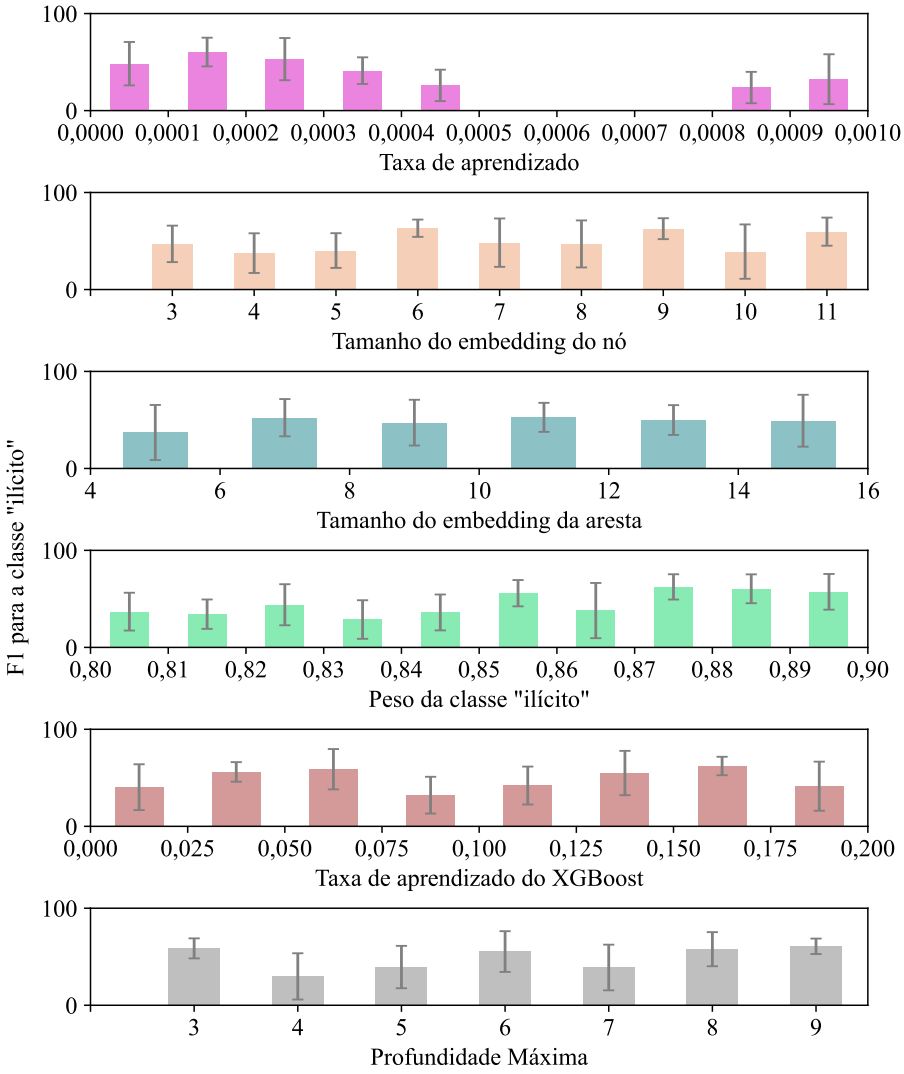


Fonte: Do autor (2023).

A variação da F1 média para a classe "ilícito" para diferentes valores de hiperparâmetros no modelo NENN com classificação via XGBoost é mostrada na Figura 5.9. É possível notar uma variação maior, em relação aos outros modelos combinados com XGBoost, da F1 média conforme a alteração das taxas de aprendizado da rede neural e do XGBoost, e do peso da classe "ilícito". A F1 atinge seus maiores valores para uma taxa de aprendizado da rede neural entre 0 e 0,0002, um peso para a classe "ilícito"entre 0,87 e 0,90, um tamanho do *embedding* da aresta menor que 8, e uma profundidade máxima da árvore de decisão maior que 7. Para os outros hiperparâmetros, não se vê um intervalo claro no qual a F1 sofre um constante aumento ou diminuição.

Para o modelo NENN com classificação via XGBoost, a configuração que resultou em uma melhor F1 para a classe "ilícito" foi uma taxa de aprendizado da rede de 0,000266, uma

Figura 5.9 – Influência dos hiperparâmetros na F1 da classe "ilícito" para o conjunto de validação, para o modelo NENN com o classificador XGBoost.



Fonte: Do autor (2023).

taxa de aprendizado do XGBoost de 0,014164, um peso para a classe "ilícito"de 0,829702, uma profundidade máxima para a árvore de decisão de 5, e o tamanho do *embeddings* do nó e da aresta fixados em 8 e 11.

Os valores de hiperparâmetros para cada modelo obtidos pela hiperparametrização estão dispostos na Tabela 5.3.

Tabela 5.3 – Conjunto de valores ótimos de hiperparâmetros para cada modelo após a hiperparametrização.

Arquitetura	Hiperparâmetro	Valor da Melhor configuração de hiperparâmetros	
		com Softmax	com XGBoost
GCN	Taxa de aprendizado da rede neural	0,008078	0,007664
	# de neurônios na camada escondida	16	12
	Peso da classe "ilícito"	0,757966	0,747424
	Profundidade máxima da árvore de decisão	-	7
	Taxa de aprendizado do XGBoost (η)	-	0,055712
Skip-GCN	Taxa de aprendizado da rede neural	0,006703	0,005559
	# de neurônios na camada escondida	22	27
	Peso da classe "ilícito"	0,73601	0,914568
	Profundidade máxima da árvore de decisão	-	4
	Taxa de aprendizado do XGBoost (η)	-	0,040558
NENN	Tamanho do embedding do nó	5	8
	Tamanho do embedding da aresta	8	11
	Peso da classe "ilícito"	0,810695	0,829702
	Taxa de aprendizado da rede neural	0,000363	0,000266
	Profundidade máxima da árvore de decisão	-	5
	Taxa de aprendizado do XGBoost (η)	-	0,014164

Fonte: Do autor (2023).

5.3 Resultados dos *ensembles* de classificadores

A Tabela 5.4 mostra os resultados de cada arquitetura com o processo de classificação sendo realizado pela combinação Softmax + XGBoost (SEÇÃO 4.5). No geral, em relação aos modelos com classificação via Softmax, observa-se um aumento da precisão, tanto para a classe "ilícito" quanto em nível macro, acompanhadas de uma diminuição da revocação. Ou seja, o uso de dois classificadores gera modelos que cobrem uma quantidade menor de casos, se comparado com o uso de um único classificador, mas torna os modelos consideravelmente mais precisos, por outro lado. Em todos os casos a F1 aumentou ligeiramente. Priorizar um equilíbrio entre as duas métricas ou uma precisão mais alta é uma decisão que deve ser tomada de acordo com as políticas internas da organização que irá utilizar o modelo, bem como o contexto no qual o modelo estará inserido.

Em comparação com os modelos que realizaram a classificação através do XGBoost, observa-se o inverso: A revocação aumentou, ao passo que a precisão diminuiu. No entanto, a F1 também teve um ligeiro aumento. Logo, pode-se afirmar que o uso somente do XGBoost como classificador torna a classificação mais precisa em relação à combinação Softmax + XGBoost.

Tabela 5.4 – Resultados para cada modelo, combinando dois classificadores, para o conjunto AMLSim 1/20.

Arquitetura	Precisão		Revocação		F1	
	macro	classe "ilícito"	macro	classe "ilícito"	macro	classe "ilícito"
GCN	**93,91±1,85**	**89,55±3,16**	83,86±1,72	68,16±3,44	88,15±1,38	77,39±2,64
Skip-GCN	93,11±2,02	87,76±4,14	**85,55±2,06**	**71,65±4,26**	**88,90±1,20**	**78,85±2,31**
NENN	93,58±2,03	88,82±4,07	84,45±3,55	69,39±7,16	88,38±2,42	77,85±4,65

Fonte: Do autor (2023).

5.4 Discussão

O combate à Lavagem de Dinheiro é um tema de extrema relevância para diversas organizações, tanto financeiras – para evitar prejuízos e perda de credibilidade perante a sociedade) quanto governamentais – para contribuir com a diminuição da corrupção e promover uma sociedade mais justa, transparente, e com menores índices de criminalidade. Substituir os processos de detecção de Lavagem de Dinheiro utilizados atualmente por tais instituições é de suma importância para a constante evolução e inovação do seu ambiente organizacional, no que tange ao seu meio digital.

Utilizando como exemplo os resultados do modelo Skip-GCN otimizados (que se saíram melhor no conjunto de dados mais desbalanceado), têm-se aproximadamente 81% de respostas corretas dentre as transações que foram classificadas como ilícitas, sendo que aproximadamente 74% do total de instâncias ilícitas do conjunto foram corretamente classificadas. Esses resultados equilibrados evidenciam uma solução que conseguiria, em um ambiente real, detectar uma imensa maioria de transações ilícitas e gerar apenas 20% de transações ilícitas, fazendo a discriminação manual de que são transações realmente ilícitas feita posteriormente diminua consideravelmente o consumo de tempo e recursos. Vale lembrar que este trabalho fez o uso de dados gerados por um simulador, e os resultados podem sofrer variações se os mesmos modelos forem aplicados a conjuntos de transações reais.

Levando em consideração os resultados, conclui-se que, para taxas de desbalanceamento menores, os modelos baseados em GCN são mais apropriados. Para taxas de desbalanceamento maiores, a combinação NENN + XGBoost se mostra mais eficaz.

Utilizando os resultados obtidos neste capítulo, o trabalho é concluído no Capítulo 6, no qual também são apresentados potenciais trabalhos futuros.

6 CONCLUSÃO

Este trabalho comparou o desempenho de um conjunto de modelos de Redes Neurais de Grafos e também duas maneiras diferentes de representação das transações via grafos. Realizou a otimização dos hiperparâmetros dos modelos de maneira a evitar que configurações inadequadas afetassem negativamente os resultados, e testou os impactos do uso de mais de um classificador sobre os *embeddings* gerados pelos modelos. De uma maneira geral, a representação das transações como nós do grafo apresentou resultados melhores do que a representação das transações como arestas do grafo. A escolha do classificador mais apropriado deve levar em consideração o que se busca em termos de precisão e revocação do modelo. As contribuições teóricas deste trabalho são detalhadas na Seção 6.1, ao passo que as contribuições práticas se encontram na Seção 6.2. Por fim, a Seção 6.3 menciona possíveis trabalhos futuros.

6.1 Contribuições teóricas

Foi apresentado, neste trabalho, o comportamento de três arquiteturas de Rede Neural de Grafos, em um contexto de classificação, em vários níveis de desbalanceamento de classe. Além disso, foi testado o impacto de melhorias e combinações sobre esses modelos sobre os resultados, evidenciando qual foi a melhor configuração em cada caso. Foi avaliado também a influência do uso de nós e arestas na geração dos vetores *embedding* a serem inseridos no classificador.

O que foi apresentado neste texto poderá servir como embasamento teórico a qualquer trabalho que aplique GNNs em um contexto de desbalanceamento de classe, ou que pretenda fazer uso de nós e arestas na geração dos vetores *embeddings*.

6.2 Contribuições práticas

Este trabalho apresenta uma solução, utilizando Redes Neurais de Grafos, para que organizações possam melhorar seu processo de detecção de Lavagem de Dinheiro. Obviamente, a solução deverá ser adaptada para o contexto no qual se deseja inseri-la, o que alterará a configuração tanto dos modelos quanto dos hiperparâmetros utilizados. Além disso, como a arquitetura NENN gera *embeddings* também para os nós, é possível facilmente adaptar o modelo para classificar as contas financeiras, sendo este um importante diferencial em uma situação real.

6.3 Trabalhos futuros

Como trabalhos futuros, serão incluídas novas arquiteturas neste experimento, inclusive arquiteturas propositalmente resistentes ao desequilíbrio de classe, além da inclusão de outros métodos de ML (como o XGBoost, por exemplo), sem o uso dos *embeddings*. Além disso, pretende-se acrescentar aos experimentos dados de transações financeiras reais, além dos dados gerados automaticamente, o que poderá afetar significativamente os resultados.

REFERÊNCIAS

ALARAB, I.; PRAKOONWIT, S. Graph-based lstm for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data. **Neural Processing Letters**, v. 54, p. 1–19, 06 2022.

ALARAB, I.; PRAKOONWIT, S.; NACER, M. I. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In: **Proceedings of the 2020 5th International Conference on Machine Learning Technologies**. New York, NY, USA: Association for Computing Machinery, 2020. (ICMLT 2020), p. 23–27. ISBN 9781450377645.

ASSUMPÇÃO, H. et al. Delator: Detecção automática de indícios de lavagem de dinheiro por redes neurais em grafos de transações. In: **Anais do XI Brazilian Workshop on Social Network Analysis and Mining**. Porto Alegre, RS, Brasil: SBC, 2022. p. 13–24. ISSN 2595-6094. Disponível em: <https://sol.sbc.org.br/index.php/brasnam/article/view/20513>.

BRASIL. Lei nº 7.492, de 16 de junho de 1986. Define os crimes contra o sistema financeiro nacional, e dá outras providências. **Diário Oficial da República Federativa do Brasil**, Brasília, DF, 16 jun. 1986. Disponível em: <https://www.planalto.gov.br/ccivil_03/leis/l7492.htm>. Acesso em: 09/02/2023.

BRASIL. Lei nº 9.613, de 3 de março de 1998. Dispõe sobre os crimes de "lavagem"ou ocultação de bens, direitos e valores; a prevenção da utilização do sistema financeiro para os ilícitos previstos nesta Lei; cria o Conselho de Controle de Atividades Financeiras - COAF, e dá outras providências. **Diário Oficial da República Federativa do Brasil**, Brasília, DF, 4 mar. 1998. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/l9613.htm>. Acesso em: 28/07/2021.

BRASIL. Lei nº 12.683, de 9 de julho de 2012. Altera a Lei nº 9.613, de 3 de março de 1998, para tornar mais eficiente a persecução penal dos crimes de lavagem de dinheiro. **Diário Oficial da República Federativa do Brasil**, Brasília, DF, 10 jun. 2012. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2012/Lei/L12683.htm>. Acesso em: 28/07/2021.

CHEN, T.; GUESTRIN, C. Xgboost. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. ACM, 2016. Disponível em: <https://doi.org/10.1145\%2F2939672.2939785>.

CHEN, Z. et al. Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: a review. **Knowledge and Information Systems**, v. 57, p. 245–285, 11 2018.

COLEY, C. W. et al. A graph-convolutional neural network model for the prediction of chemical reactivity. **Chem. Sci.**, The Royal Society of Chemistry, v. 10, p. 370–377, 2019. Disponível em: <http://dx.doi.org/10.1039/C8SC04228D>.

COLLADON, A. F.; REMONDI, E. Using social network analysis to prevent money laundering. **Expert Systems with Applications**, v. 67, p. 49–58, 2017. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417416305139>.

DREŻEWSKI, R.; SEPIELAK, J.; FILIPKOWSKI, W. System supporting money laundering detection. **Digital Investigation**, v. 9, n. 1, p. 8–21, 2012. ISSN 1742-2876. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1742287612000230>. Acesso em: 28/07/2021.

DREŻEWSKI, R.; SEPIELAK, J.; FILIPKOWSKI, W. The application of social network analysis algorithms in a system supporting money laundering detection. **Information Sciences**, v. 295, p. 18–32, 2015. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025514009979>.

FARRUGIA, S.; ELLUL, J.; AZZOPARDI, G. Detection of illicit accounts over the ethereum blockchain. **Expert Systems with Applications**, v. 150, p. 113318, 2020. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417420301433>.

FATF. **FATF 40 Recommendations**. Paris, France: FATF, 2003. Disponível em: <https://www.fatf-gafi.org/media/fatf/documents/FATF%20Standards%20-%2040%20Recommendations%20rc.pdf>.

FEURER, M.; HUTTER, F. Hyperparameter optimization. In: ____. **Automated Machine Learning: Methods, Systems, Challenges**. Cham: Springer International Publishing, 2019. p. 3–33. ISBN 978-3-030-05318-5. Disponível em: <https://doi.org/10.1007/978-3-030-05318-5_1>.

GOLDBARG, M.; GOLDBARG, E. **Grafos**: Conceitos, algoritmos e aplicações. 1. ed. Rio de Janeiro – RJ, BR: Elsevier, 2012.

GOTTSCHALK, P. Categories of financial crime. **Journal of Financial Crime**, v. 17, p. 441–458, 10 2010.

GRINSZTAJN, L.; OYALLON, E.; VAROQUAUX, G. **Why do tree-based models still outperform deep learning on tabular data?** arXiv, 2022. Disponível em: <https://arxiv.org/abs/2207.08815>.

GUPTA, N. et al. Data quality for machine learning tasks. In: **Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining**. New York, NY, USA: Association for Computing Machinery, 2021. (KDD '21), p. 4040–4041. ISBN 9781450383325. Disponível em: <https://doi.org/10.1145/3447548.3470817>. Acesso em: 25/04/2022.

HAMILTON, W. L. **Graph Representation Learning**. San Rafael – CA, USA: Morgan & Claypool, 2020. (Synthesis Lectures on Artifical Intelligence and Machine Learning, 46).

HAN, J. et al. Artificial intelligence for anti-money laundering: a review and extension. **Digital Finance 2020 2:3**, Springer, v. 2, n. 3, p. 211–239, jun 2020. ISSN 2524-6186. Disponível em: <https://link.springer.com/article/10.1007/s42521-020-00023-1>. Acesso em: 28/07/2021.

HAYKIN, S. **Redes neurais**: princípios e prática; trad. Paulo Martins Engel. 2. ed. Porto Alegre – RS, BR: Bookman, 2001.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. In: **International Conference on Learning Representations (ICLR)**. Toulon, France: OpenReview.net, 2017.

KROLL. **Global Fraud and Risk Report 2019/20**: Mapping the new risk landscape. 11. ed. Boston - MA, US, 2019. Disponível em: <https://www.kroll.com/en/insights/publications/global-fraud-and-risk-report-2019>. Acesso em: 28/07/2021.

LARIK, A. S.; HAIDER, S. Clustering based anomalous transaction reporting. **Procedia Computer Science**, v. 3, p. 606–610, 2011. ISSN 1877-0509. World Conference on Information Technology. Disponível em: <https://www.sciencedirect.com/science/article/pii/S187705091000476X>.

LE-KHAC, N.-A.; MARKOS, S.; KECHADI, T. Towards a new data mining-based approach for anti-money laundering in an international investment bank. In: **Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering**. Albany, Ny, USA: Springer, 2009. v. 31, p. 77–84. ISBN 978-3-642-11533-2.

LI, Y. et al. Pgcn: Disease gene prioritization by disease and gene embedding through graph convolutional neural networks. **bioRxiv**, Cold Spring Harbor Laboratory, 2019. Disponível em: <https://www.biorxiv.org/content/early/2019/01/28/532226>.

LIAW, R. et al. Tune: A research platform for distributed model selection and training. **arXiv preprint arXiv:1807.05118**, 2018.

LIU, Y. et al. Pick and choose: A gnn-based imbalanced learning approach for fraud detection. In: **Proceedings of the Web Conference 2021**. [S.l.: s.n.], 2021. p. 3168–3177.

LOPEZ-ROJAZ, E. A.; AXELSSON, S. Money laundering detection using synthetic data. In: **Linköping Electronic Conference Proceedings, No. 71**. Linköping, Sweden: Linköping University Electronic Press, 2012. p. 33 – 40.

MARSLAND, S. **Machine Learning: An Algorithmic Perspective**. 2. ed. Boca Raton – FL, EUA: CRC Press, 2015. ISBN 9781498759786.

PAREJA, A. et al. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In: **Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2020.

PEREIRA, R.; MURAI, F. Quão efetivas são redes neurais baseadas em grafos na detecção de fraude para dados em rede? In: **Anais do X Brazilian Workshop on Social Network Analysis and Mining**. Porto Alegre, RS, Brasil: SBC, 2021. p. 205–210. ISSN 2595-6094. Disponível em: <https://sol.sbc.org.br/index.php/brasnam/article/view/16141>. Acesso em: 25/04/2022.

RAYANA, S.; AKOGLU, L. Collective opinion spam detection: Bridging review networks and metadata. In: **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2015. (KDD '15), p. 985–994. ISBN 9781450336642.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65 – 386, 1958.

RUDER, S. An overview of gradient descent optimization algorithms. **ArXiv**, abs/1609.04747, 2016.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**; trad. Regina Célia Simille. 3. ed. Rio de Janeiro – RJ, BR: Elsevier, 2013. ISBN 0137903952.

SUZUMURA, T.; KANEZASHI, H. **Anti-Money Laundering Datasets: InPlusLab Anti-Money Laundering DataDatasets**. 2021. Disponível em: <http://github.com/IBM/AMLSim>. Acesso em: 25/04/2022.

WEBER, M. et al. Scalable graph learning for anti-money laundering: A first look. **CoRR**, abs/1812.00076, 2018.

WEBER, M. et al. **Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics**. Anchorage, Alaska, USA: [s.n.], 2019. Disponível em: <https://drive.google.com/drive/folders/1r_iJYFJru-jdDdgpB-KZ1N0Zathy2LD2>.

YANG, Y.; LI, D. Nenn: Incorporate node and edge features in graph neural networks. In: PAN, S. J.; SUGIYAMA, M. (Ed.). **Proceedings of The 12th Asian Conference on Machine Learning**. Bangkok, Thailand: PMLR, 2020. (Proceedings of Machine Learning Research, v. 129), p. 593–608. Disponível em: <https://proceedings.mlr.press/v129/yang20a.html>. Acesso em: 25/04/2022.

YANG, Z.; CHAKRABORTY, M.; WHITE, A. D. Predicting chemical shifts with graph neural networks. **Chem. Sci.**, The Royal Society of Chemistry, v. 12, p. 10802–10809, 2021. Disponível em: <http://dx.doi.org/10.1039/D1SC01895G>.

ZENG, H. et al. GraphSAINT: Graph sampling based inductive learning method. In: **International Conference on Learning Representations**. [s.n.], 2020. Disponível em: <https://openreview.net/forum?id=BJe8pkHFwS>. Acesso em: 25/04/2022.