Issue 16, 2024

https://doi.org/10.23939/sisn2024.16.251

UDC 004.6

GNN IMPLEMENTATION APPROACHES IN AWS CLOUD FOR RISK ASSESSMENT IN THE INSURANCE AREA

Oleksandr Lutsenko[1], Serhii Shcherbak[2]

[1, 2] Lviv Polytechnic National University,
Information Systems and Networks Department, Lviv, Ukraine
[1] E-mail: Oleksandr.V.Lutsenko@lpnu.ua, ORCID: 0009-0008-7644-6056
[2] E-mail: serhii.s.shcherbak@lpnu.ua, ORCID: 0000-0003-2914-2101

This article analyzes three most common approaches to the GNN architecture implementation on the AWS cloud for the use case of the risk assessment in the insurance area. The paper is split to several chapters, with the first one being the overview of 3 approaches to the GNN architecture, the second one describing prerequisites for the implementation, and finally development of the approaches on the cloud infrastructure, testing them on graph insurance data and comparison of all the approaches to select the most suitable for the risk assessment task.

The initial chapter introduces the three architectural approaches to GNN implementation being respectively Graph Convolutional Network (GCN), Graph Attention Network (GAT) and GraphSAGE (Graph Sample And AGgregatE). To conclude the chapter, it is decided to proceed with the further implementation of all three models on the AWS infrastructure and analyze the outputs on the same graph data to select the best suit for the risk assessment use case.

Then the article proceeds with considering the specifics of a realization of risk assessment in insurance on top of cloud infrastructure and preparing the data to use it for the GNN training and testing. After the analysis of the use case, it is decided to focus on only on the individuals' insurance. The main goal is to analyze the unique properties of every human which can affect the risk of insuring them as well as their connections with other individuals.

Further along, the development of all three approaches for risk assessment solution is described with first being GCN, then GAT and finally GraphSage. The models are then trained, tested and the output analysis is performed. Considering the analysis results, GAT and GraphSage provide the most correct results maintaining the test accuracy. However, considering model statistics, it is found that GraphSage has more distinct probabilities and additional insights through feature importance analysis which makes it the best fit for the risk assessment use case.

The article concludes by stating that out of all three analyzed architectures the most suitable for the risk assessment task is the GraphSAGE with a slight difference between this model and GAT, which will be used for further analysis and improvements. Furthermore, the article mentions a few steps for the potential future improvements of the models, which include using class weights or oversampling techniques to ensure the best performance, also mentioning the experiments that can be done with deeper architectures or different GNN layers. The last but not the least would be to focus on the testing and training on the larger dataset to make it more applicable for real-world applications.

Keywords: Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE (Sample and aggreGatE), Graph, Graph Neural Network (GNN), Underwriting, Insurance, Risk Assessment.

Introduction

Risk assessment in the insurance industry plays a pivotal role in determining premiums, managing exposure, and ensuring the financial stability of insurance companies. Accurate risk evaluation allows

insurers to make valuable decisions about policy issuance, pricing, and claim management. As the insurance landscape becomes increasingly complex, with interconnected risk factors and evolving societal trends, traditional methods of risk assessment are often found unsuitable from the performance standpoint. This has led to a growing interest in more sophisticated, data-driven approaches that can capture nuanced relationships between various risk factors and predict potential outcomes with greater accuracy.

In recent years, the insurance industry has recognized that risk factors are rarely isolated; instead, they form complex networks of interrelated variables. For instance, an individual's health risk might be influenced not only by their personal habits but also by their family history, social connections, and environmental factors. This realization has prompted insurers to view their data not as mere collections of individual records, but as interconnected graphs where nodes represent entities (such as policyholders), and edges represent relationships or shared attributes between these entities.

Graph Neural Networks (GNNs) are a powerful tool for analyzing and extracting insights from such graph-structured data. By leveraging the inherent structure of graphs, GNNs can capture complex dependencies and propagate information across the network, enabling more accurate predictions and risk assessments. This approach is particularly well-suited to the insurance domain, where understanding the connections between various risk factors is crucial both for accurate underwriting and claims prediction.

In this article, three prominent GNN architectures are analyzed – Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and GraphSAGE. The focus is also shifted to their application to risk assessment in insurance. Each of these approaches offers unique strengths in processing graph-structured data, from GCN's efficient neighborhood aggregation to GAT's attention mechanism and GraphSAGE's inductive learning capabilities. By comparing these architectures, aim is to provide insights into their relative performance and suitability for insurance risk assessment tasks, especially in the cloud infrastructure considering the tendency of ever-growing datasets and the ability of the cloud systems to handle such volumes at ease.

The analysis delves into the theoretical foundations of each architecture, their implementation details selecting the appropriate tools on the cloud, definite framework for the implementation, and analysis of the results after training and testing the models on a representative insurance dataset. Each model capturing and utilizing of the graph structure of insurance data, their ability to handle heterogeneous risk factors, and their performance in predicting high-risk cases is being analyzed. Additionally, the interpretability of these models is discussed, with it being a crucial factor in the highly regulated insurance industry where decisions often need to be explained and justified.

Through this comprehensive examination, it is sought to shed light on the potential of GNNs to revolutionize risk assessment in insurance. By using the power of graph-structured data, especially in the cloud environment to handle big data volumes and advanced neural network architectures for better scaling and infrastructural easiness, insurers can gain deeper insights into risk factors, improve the accuracy of their assessments, and ultimately offer more personalized and fair insurance products. Navigating through the individual characteristics of GCN, GAT, and GraphSAGE, the aim is to provide valuable insights for both researchers and practitioners in the insurance industry, paving the way for more sophisticated and effective risk assessment methodologies.

Formulation of the problem

In the realm of insurance, accurately assessing risk is inevitable for setting premiums and managing claims effectively. The risk assessment is the complex process required to understand if the certain individual or a company can be insured without losses for the insurance provider. As can be seen on Fig. 1, the calculation of the risk may take a lot of time and depends on a huge number of factors in the data which are intertangled among themselves.

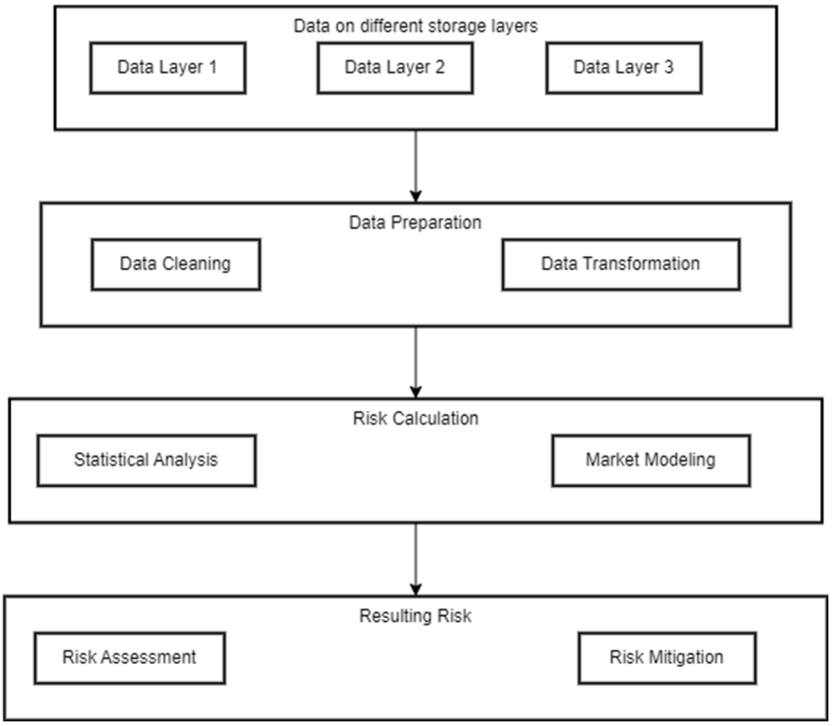

Fig. 1. The general description of the risk assessment process

One of the approaches to represent the data with a lot of the different connections such as insurer and the insured person in our use case is a graph representation. This approach focuses not only on the individual features of every data node such as the insured health state, bad habits etc., but even more on the connections between different people such as co-living or family relationships providing a more detailed look and the whole picture instead of nitpicking just one individual. These factors may impact the risk of insuring a certain individual, for example, if they live in the very dangerous area, the risk of health insurance will be higher.

There are various ways of calculation of the risk, with most common being complex algorithms and neural networks. The focus of the article is to analyze the potential benefit of using neural networks for the risk assessment. By leveraging the interconnected nature of client data, such as historical claims, policyholder interactions, and network relationships, presented in the graph structure, one of the methods that this article proposes to use for its analysis is GNN. GNNs can provide a more nuanced understanding of risk factors as it can be presented on Fig. 2, which is the more detailed view on part of the whole risk assessment process on the Fig. 1.

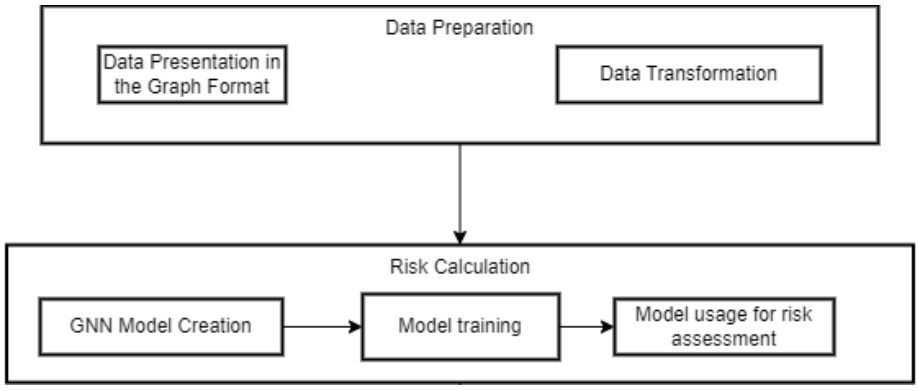

Fig. 2. The GNN usage in the insurance risk assessment process

Analysis of recent research and publications

Common Approaches to GNN Architecture in the context of risk assessment

The insurance area handles huge volumes of interconnected data such as information about the policyholders, potential insurers, companies and their intermediaries, etc. Especially it can be stated that in the process of underwriting and risk assessment a lot of the data is used for the analysis to predict the risk of the potential insurance of the new customer. The main reasons for that include the insurance company not wanting to lose money after insuring someone who has the risk of either tricking the insurance company for money or just happens to have factors that may impact individual's health, for example. Besides several individual factors that should be considered during the process of the risk assessment, the connections between individuals, their affiliations with companies, the family relationships with potentially dangerous individuals and even some non-obvious connections with already insured individuals [1].

Considering this intertangled structure of the data required for the risk assessment of insuring the individual, the most optimal storage for this particular task should be the graph structure. There are several ways to analyze the data in the graphs being respectively complex algorithms or using neural networks. The best suit of all neural networks designed specifically for the graph structures is the Graph Neural Networks (GNN). However, there are a few different GNN architectures, so it is necessary to choose the best one for the risk assessment task. The choice of GNN architecture can significantly impact the model's ability to capture relevant patterns that are necessary to make a decision on the risk impact of insuring a certain individual in the data, handle different types of graph structures (individual relationships, connections between companies etc.), and generalize to unseen data. Understanding the strengths and limitations of different GNN approaches allows to make informed decisions, balancing factors such as computational efficiency, interpretability, considering the specific requirements of the risk assessment task in the insurance underwriting process.

There are three main architectural designs of the GNN: Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE (Sample and aggreGatE) [2].

GCNs apply convolutional operations to graph-structured data. They aggregate information from a node's neighbors using a spectral convolution approach. GCNs use a simple weighted sum of neighbor features, typically with weight sharing across edges. They're efficient and work well for many tasks but may struggle with heterogeneous graphs or when node ordering is important.

GATs introduce attention mechanisms to GNNs. Instead of treating all neighbors equally, GATs learn to assign different importance to different neighbors when aggregating information. This allows the model to focus on the most relevant neighbors for each node. GATs can handle both homogeneous and heterogeneous graphs and are particularly effective when the importance of node relationships varies.

GraphSAGE (SAmple and aggreGatE) is an inductive framework for node embedding. It learns a function to generate embeddings by sampling and aggregating features from a node's local neighborhood. GraphSAGE can use various aggregator functions (mean, max, LSTM, etc.) and is particularly useful for large graphs as it can generalize to unseen nodes. It's efficient and scalable, making it suitable for dynamic or evolving graphs.

GCN approach in scope of risk assessment task

GCNs are a class of GNN designed to operate on graph-structured data. GCNs generalize the concept of convolution from grid-like data (e.g., images) to irregular graph structures. In the context of risk assessment in the insurance sector, irregular graph structures could arise from various complexities and anomalies in the data. For instance, highly interconnected nodes representing clients with multiple policies or frequent claims could form dense clusters, while isolated nodes might represent clients with minimal interaction or claims history. Additionally, irregularities might occur due to missing data or inconsistent reporting of client interactions and transactions, leading to incomplete edges or nodes with sparse

attributes. Such irregularities can challenge the model's ability to accurately learn and generalize the underlying patterns, potentially impacting the effectiveness of the risk assessment.

In GCN the first step of the application is the neighborhood aggregation: GCNs update node representations by aggregating information from their immediate neighbors, for example, in our case it will be immediate neighbors of the individuals living in the same address or first line relatives such as parents and children. After that Layer-wise Propagation is applied: Information is propagated through the graph in multiple layers, allowing nodes to incorporate information from higher-order neighborhoods.

It is also worth mentioning that GCNs are often motivated by spectral graph theory, although practical implementations use spatial approaches [2].

GCN can be described mathematically in the following way

Let G = (V, E) be a graph with nodes v in V and edges (v, w) in E. Each node has a feature vector x_v. The GCN layer can be described as follows:

Graph Convolution Operation:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} W^{(l)} h_u^{(l)}\right), \quad (1)$$

where:

- $h_v^{(l+1)}$ is the feature vector of node v at layer l
- $\mathcal{N}(v)$ is the set of neighbors of node v
- $W^{(l)}$ is the learnable weight matrix at layer l
- σ is a non-linear activation function (e.g., ReLU)

Matrix Form:

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

where:

– $\tilde{A} = A + I_N$ is the adjacency matrix with self-loops

– $\tilde{D}$ is the degree matrix of $\tilde{A}$

– $H^{(l)}$ is the matrix of node features at layer l

To address numerical instabilities and exploding/vanishing gradients, Kipf and Welling proposed a renormalization trick to replace the normalized adjacency matrix with the augmented normalized adjacency matrix [4]:

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right), \quad (3)$$

where $\hat{A} = \widetilde{D^{-\frac{1}{2}}}\tilde{A}\widetilde{D^{-\frac{1}{2}}}$

GCNs have proven to be highly effective in diverse applications such as node classification, link prediction, and graph classification within various domains, showcasing their adaptability and efficiency in handling graph-structured data. In the context of risk assessment in the insurance sector, the advantages of GCNs include their ability to efficiently propagate information across the graph, capture local graph structures, and implement parameter sharing across the graph, which enhances model generalization. These features are particularly beneficial for understanding and predicting risk based on the complex interrelations of policyholder data.

However, there are certain limitations associated with GCNs that can impact their performance in risk assessment tasks. These include challenges in capturing long-range dependencies within large and complex insurance networks, difficulties in dealing with heterogeneous graphs that contain diverse types of nodes and edges, and a fixed aggregation scheme that may not be ideally suited for all types of risk assessment scenarios. These limitations necessitate careful consideration and potential adaptations of the GCN approach to ensure optimal performance in specific risk assessment applications within the insurance industry.

GAT approach in scope of risk assessment task

On the other hand, one of the most used GNN architectures is GAT. It is a sophisticated class of Graph Neural Networks that introduce attention mechanisms to graph-structured data processing. GATs address some limitations of GCNs by allowing nodes to attend differently to their neighbors, enabling more flexible and adaptive feature aggregation. For the risk assessment it may mean that for risk of insuring a certain individual some of the relatives have more impact on the final risk then others. Another example of the attention to certain neighbors may be in analysis of the area where the person lives with specific attention to certain factors such as crime rate in the zone or connected individuals with crime records paying less attention to others that may have less impact as number of homeless cats in the region.

It is done by using attention mechanism: GATs compute attention coefficients for each neighbor, allowing the model to focus on the most relevant nodes [5]. Multiple independent attention mechanisms are employed to stabilize the learning process and enrich the model's expressive power. It is also important that nodes can attend to themselves, incorporating self-information in the update process, this possibility is called self-attention.

Considering previously described setup and prerequisites, attention mechanism can be described in the following way:

The attention coefficient e_{vw} between nodes v and w is computed as:

$$e_{vw} = a(Wh_v, Wh_w)\,, \tag{4}$$

where:

- W is a learnable weight matrix
- a is a single-layer feedforward neural network
- h_v and h_w are the feature vectors of nodes v and w

The attention coefficients are normalized using softmax [6]:

$$\alpha_{vw} = \mathrm{softmax}_w(e_{vw}) = \frac{\exp(e_{vw})}{\sum_{u\in\mathcal{N}(v)}\exp(e_{vu})}, \tag{5}$$

where $\mathcal{N}(v)$ is the neighborhood of node v.

On the step of feature aggregation, the updated feature vector for node v is computed as:

$$h'_v = \sigma\left(\sum_{w\in\mathcal{N}(v)}\alpha_{vw}Wh_w\right), \tag{6}$$

where σ is a non-linear activation function (e.g., ELU).

In the realm of risk assessment, Graph Attention Networks (GATs) offer significant advantages due to their adaptive feature aggregation, which prioritizes the influence of neighboring nodes based on their relevance. This capability is particularly useful in insurance where nodes (e.g., policyholders) have varying degrees of connectivity reflecting their different risk profiles. Additionally, GATs enhance model interpretability through attention weights, providing insights into the decision-making process by highlighting influential relationships. Their effectiveness in both transductive (learning from the entire graph) and inductive (generalizing to unseen parts of the graph) learning tasks makes them versatile tools for assessing risks in dynamic insurance environments.

However, GATs also present certain challenges in the context of risk assessment. Their increased computational complexity can be a drawback compared to simpler models like GCNs, particularly when rapid processing of data is required. There is also a heightened risk of overfitting when training on smaller graphs, which can lead to less reliable risk predictions. Moreover, handling very large graphs can be problematic due to memory constraints, potentially limiting their applicability in scenarios with extensive data networks.

GraphSAGE approach in scope of risk assessment task

GraphSAGE (Graph SAmple and aggreGatE) is an inductive framework for computing node embeddings that learns to generate embeddings by sampling and aggregating features from a node's local neighborhood. It's designed to be highly scalable and can generalize to unseen nodes, making it particularly suitable for large or dynamic graphs. For example, it could include temporal changes in the insured individual's connections or the individuals in highly populated areas to be able to analyze not all of the nodes, but finding the most important ones. Instead of using the entire neighborhood, GraphSAGE samples a fixed-size set of neighbors for each node. This process is called neighborhood sampling. After that the algorithm learns a set of aggregator functions that collect information from a node's local neighborhood. On the latest step GraphSAGE can generate embeddings for nodes that were not present during training [7].

Considering the graph representation, introduced in a previous paragraph, we can describe the SAGE in the following way.

For each node v, sample a set of neighbors $\mathcal{N}(v)$ up to a maximum size S.

The general form of the GraphSAGE layer is:

$$h_v^k = \sigma\left(W^k \cdot \text{CONCAT}\left(h_v^{k-1}, \text{AGG}_k\left(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)\right)\right), \tag{7}$$

where:

- h_v^k is the hidden state of node v at layer k;
- W^k is the weight matrix for layer k;
- AGG_k is the aggregator function for layer k;
- σ is a non-linear activation function.

GraphSAGE proposes several aggregator functions, among which the commonly used are:

Mean Aggregator:

$$\text{AGG}_{\text{mean}} = \frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} h_u^{k-1}\ . \tag{8}$$

And Max-pooling Aggregator:

$$\text{AGG}_{\text{max}} = \max\left(\{\sigma\left(W_{\text{pool}}h_u^{k-1} + b\right), \forall u \in \mathcal{N}(v)\}\right). \tag{9}$$

The final representation for node v is:

$$z_v = h_v^K / |h_v^K|_2\ , \tag{10}$$

where K is the number of layers in the GraphSAGE model.

GraphSAGE can be trained using various loss functions depending on the task. For unsupervised learning, it often uses a graph-based loss:

$$\mathcal{L} = -\log\left(\sigma(z_v^T z_u)\right) - Q \cdot E_{v_n\sim P_n(v)}\left[\log\left(\sigma\left(-z_v^T z_{v_n}\right)\right)\right], \tag{11}$$

where u is a neighbor of v, v_n is a negative sample, and Q is the number of negative samples.

In the insurance risk assessment context, GraphSAGE offers notable advantages that make it well-suited for analyzing large and complex datasets typical in this industry. Its high scalability is achieved through neighborhood sampling, allowing it to efficiently handle large graphs that represent extensive policyholder networks. Additionally, GraphSAGE's ability to generate embeddings for unseen nodes through its inductive capabilities is particularly valuable for assessing risks associated with new clients or emerging risk factors. The flexibility in choosing aggregator functions, such as sum or average, further enhances its adaptability to various data structures, making it effective for both transductive and inductive tasks.

However, there are inherent challenges with the GraphSAGE approach when applied to insurance risk assessment. The method's reliance on fixed-size neighborhood sampling can result in the loss of critical information, especially in densely connected regions of the graph where nuanced relationships

might indicate elevated risks. The performance of GraphSAGE is also sensitive to the choice of aggregator function, which can significantly influence the accuracy of risk predictions. Moreover, its focus on local sampling may hinder the model's ability to capture global graph structures, potentially overlooking broader risk patterns that emerge across the entire network.

Despite these limitations, GraphSAGE has demonstrated robust performance in various graph-based tasks such as node classification, link prediction, and graph classification. It excels particularly in scenarios that involve large-scale or dynamically changing graphs, where the ability to generalize from known to unknown data is crucial. This makes GraphSAGE a promising tool for evolving risk assessment models in the insurance sector, where the landscape of risk factors continuously shifts and expands.

Conclusion on the approaches to the GNN implementation

To compare the suitability of different architectural approaches to the specific risk assessment use case in the insurance, the GNN can be implemented using several methods, each varying in complexity and optimization within the cloud infrastructure. By comparing the results through testing and analyzing model output statistics, the performance, accuracy, and efficiency of each approach can be evaluated. This comparative analysis involves assessing the speed of convergence, the robustness of the model under different data conditions, and the scalability handling large datasets. Additionally, the impact of different hyperparameters, activation functions, and layer structures can be explored to fine-tune the models for optimal performance. Ultimately, this systematic evaluation helps in identifying the most effective GNN architecture for risk assessment in the insurance, ensuring that the chosen model not only predicts accurately but also integrates seamlessly with existing systems and scales effectively as data volume grows.

Goal formulation and task setting

The primary goal of this article is to conduct a comparative analysis of different GNN implementations for the risk assessment task in the insurance area deployed on the cloud environment. Being more specific, the objective is to evaluate and compare the performance of three distinct GNN architectures: Graph Convolutional Network (GCN), Graph Attention Network (GAT), and GraphSAGE. The comparison will focus on key metrics such as model output statistics, precision, and test accuracy to determine which implementation offers the most effective solution for enhancing risk assessment processes in the insurance sector. Considering the huge volumes of the data in the insurance sector and the complexity of the analysis in order to calculate the proper risks for certain individuals, the data should be deployed to a cloud infrastructure which can easily handle such tasks. The simplified architecture of the risk assessment task on a cloud infrastructure may look as following (Fig. 3).

As we can see from the diagram above, to achieve the formulated goal, the project can be divided into several structured tasks: firstly, collection and preprocessing the necessary data that will be used to train and test the GNN models should happen. This will allow to use the generalized data for every GNN, reducing the risk of bias in favor of one of the architectures. The next step is decision on the appropriate tools on the cloud for the graph data storage considering the specifics of the individual's data required for the risk assessment task, GNN development, training and finally processing of the data. The next step is the development of different GNN approaches which can solve the risk assessment problem and give the result as either the risk number or binary classification problem (e.g. risky/not risky). Each of the three models is then deployed, trained and tested on the cloud, utilizing appropriate computational resources and monitoring for performance and efficiency. Having done this, each model can be evaluated based on model output statistics, precision, and test accuracy. This involves running the models on a test dataset and capturing relevant metrics to assess performance for the risk assessment on the prepared graph where we definitely know which of the individuals are risky to insure.

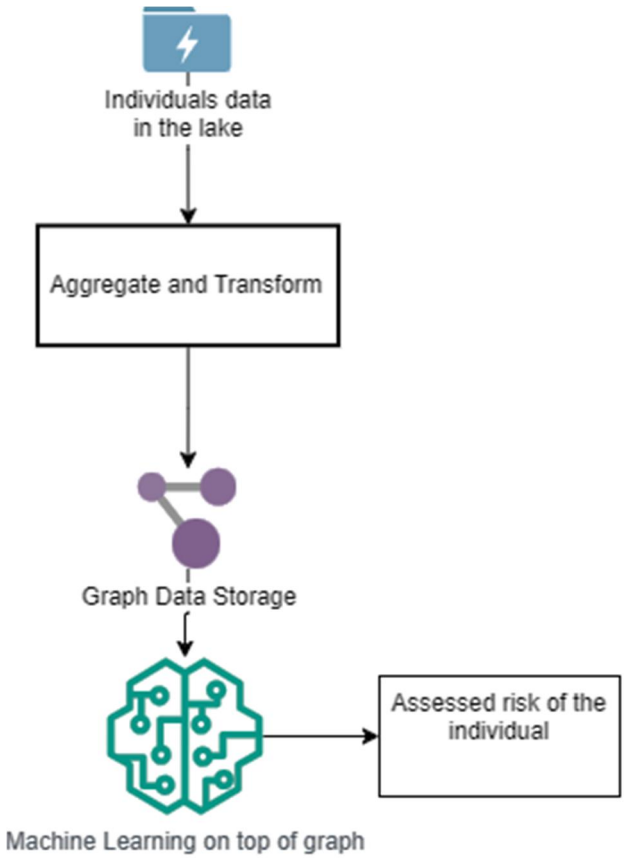

Fig. 3. The risk assessment task on a cloud infrastructure

Finally, comparative analysis should be conducted to confront the results obtained from each GNN architecture implementation to find the strengths and weaknesses of each model for risk assessment of the individual during underwriting process in the insurance area and then document them.

Presenting main material

The insurance data preparation and presentation for the risk assessment

In the realm of machine learning, particularly when dealing with GNNs, the quality and structure of the data used for training and testing are pivotal to the success of the model. For our project, which involves comparing different GNN implementations for risk assessment in the insurance sector using AWS cloud, we will utilize a specifically structured graph. This graph will encapsulate relevant features and relationships pertinent to insurance data, such as policyholder details, claim histories, and network interactions among clients.

The preparation of the correct data is crucial because GNNs leverage the relational information between nodes (data points) to generate predictions or classifications. A well-prepared graph ensures that the model can effectively learn and generalize from the interconnected nature of the data, rather than just individual data points in isolation [8]. This is particularly important in risk assessment, where the relationships and interactions between different entities can significantly influence risk profiles. By meticulously structuring and curating the graph, we aim to provide a resilient dataset that will allow the GNN models – GCN, GAT, and GraphSAGE – to accurately learn and predict risk, thereby enhancing the precision and reliability of the risk assessment process. This structured approach to data preparation not only aids in achieving high accuracy and performance but also ensures that the models are trained on data that closely mimics real-world scenarios, thus enhancing their applicability and effectiveness in practical settings.

To have appropriate use of the graph representation, individuals or companies can be used for the nodes of the graph. The nodes are then connected based on relationships (e.g., family members, business partnerships, shared demographics). To solve the binary classification task, which will be used as a simplification of the risk assessment, we have to rely on the node features, which include relevant attributes such as: age, health condition, income, social security, occupation, lifestyle habits for individuals and industry, size, revenue, number of employees, financial health indicators for companies.

For the specific use case, it has been decided to use the graph, which represents the individual wanting to insure his health, with the following setup: each node in our graph represents an individual. These nodes contain various attributes such as age, health status, lifestyle habits (e.g., smoking), and regular medical check-up information. The edges in the graph represent relationships between individuals. These could include family ties, shared living spaces (same zipcode), or other relevant connections that might influence health risks.

The simplified graph for the training of the model can be presented as such [Fig.4].

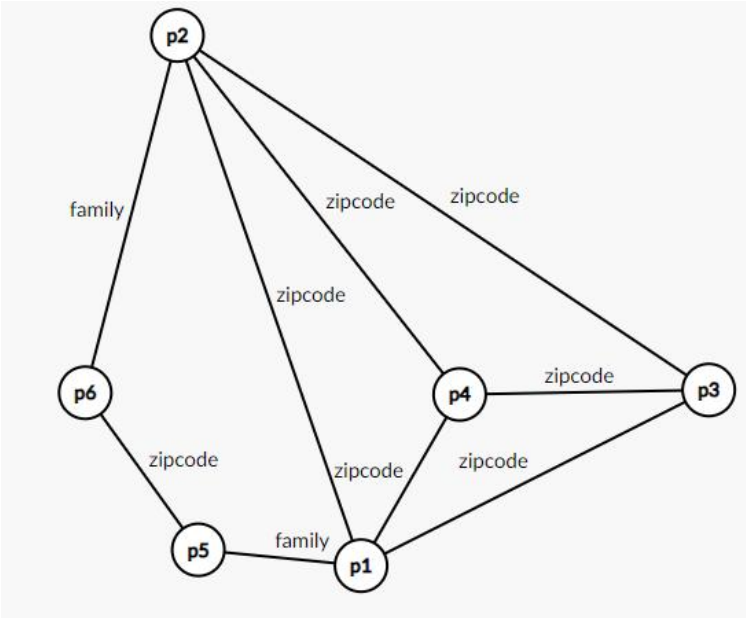

Fig. 4. The graph representation of the underwriting process

This graph represents a network of individuals and their relationships, focusing on health-related attributes and connections.

The graph consists of six nodes, each representing a person (p1 to p6). Each node has the following attributes:

- Unique identifier (id).
- Zipcode.
- Regular checkups (boolean).
- Smoking status (boolean).
- Health status (categorical: good or poor).
- Node characteristics include:
- Four people (p1, p2, p3, p5) have good health, regular checkups, and don’t smoke.
- One person (p4) has poor health, doesn’t have regular checkups, and smokes.
- Five people (p1, p2, p3, p4) share the zipcode ‘12345’.
- Two people (p5, p6) share a different zipcode ‘67890’.

The graph has nine edges representing two types of relationships: same zipcode and family relationship with 7 and 2 edges of two types respectively.

The important features that should be considered during the training process of the model: the graph shows two distinct communities based on zipcodes (12345 and 67890). There are family connections between the two zipcode communities. The person with poor health (p4) is connected to three people with good health in the same zipcode. All individuals except one have good health practices (regular checkups, non-smoking).

This graph structure allows for analyzing how health attributes and behaviors might be influenced by geographical proximity (same zipcode) and family relationships, which could be valuable for health risk assessment and understanding the spread of health-related behaviors within communities.

To prepare our data for use in a Graph Neural Network model, we need to transform it into an appropriate graph structure. This process involves several key steps:

- Node Feature Extraction: Convert individual attributes into a consistent feature vector for each node.
- Edge List Creation: Define the connections between individuals based on available relationship data.
- Adjacency Matrix Construction: Generate an adjacency matrix representing the graph structure.
- Feature Normalization: Normalize node features to ensure consistent scale across different attributes.
- Encoding Categorical Variables: Convert categorical data (like health status) into numerical representations.
- Handling Missing Data: Implement strategies to deal with any missing information in node features or relationships.

By carefully preprocessing our data into this graph structure, a rich representation can be created that captures both individual characteristics and network effects. This approach allows the GNN model to leverage both personal attributes and relational information in assessing health risks, potentially leading to more nuanced and accurate predictions.

Specifics of realization of risk assessment in insurance on top of cloud infrastructure

Having defined and preprocessed the graph, which can be used for the GNN development, the next step is to describe the specifics of the solution of the problem on the cloud infrastructure for proper processing of huge volumes of the interconnected insurance data. In our case the models will be deployed to AWS cloud, so the focus is shifted to the tools available there. However, it is worth noting that similar tools can be found on all major cloud providers such as Microsoft Azure and Google Cloud, the AWS was chosen only due to pricing and the prior knowledge. It is crucial to analyze the data properly and have the scalable and optimizable solution which can be used for the more complex risk assessment tasks on the real-world insurance data not only for the individuals but for the whole neighborhoods or even companies.

Amazon Neptune is the AWS service for graph storage and querying [9]. Employing Amazon Neptune for graph management and storage is particularly relevant for several reasons in the scenario of the risk assessment analytics in the insurance:

- Purpose-Built for Graphs: The service is specifically engineered as a high-performance graph database, ideal for efficiently managing and querying interconnected data, which is essential for the graph-structured data used in GNN application.
- Scalability: Neptune can manage billions of relationships and supports scaling to accommodate graph applications that require high query volumes. This feature is vital for handling large graphs or multiple graph instances.
- Gremlin Compatibility: The platform supports the Gremlin graph traversal language, which can be utilized in the code for the easy and reliable interaction with the graphs. This compatibility facilitates complex graph queries and traversals that might be too complex or less efficient in non-graph databases.

- AWS Ecosystem Integration: As part of the AWS suite, Neptune seamlessly integrates with other AWS services, enhancing the deployment of GNN models in the cloud, streamlining security measures, and managing data workflows.
- ACID Compliance: Neptune maintains data integrity through ACID compliance, ensuring the precision and reliability of the graph data.
- Optimized Performance: The database is optimized for graph-related operations, offering rapid query responses, even for complex queries on several connections typical in graph analysis.
- Managed Service: As a fully managed service, Neptune handles the burden of database management, allowing to concentrate on developing the GNN application.

In the specific context, most important features of Neptune for the underwriting include facilitating efficient storage and querying of graph structures (vertices and edges) along with their attributes. This capability is crucial for constructing node features, edge indices, and edge attributes necessary for creating PyTorch Geometric Data object.

Utilizing Neptune allows to decouple graph storage and querying from GNN processing, enabling independent scalability of each component. It also lays a solid foundation for application growth, such as managing larger graphs or integrating more complex graph operations in data preprocessing or feature engineering.

The next crucial tool to select is the framework which can be used for the GNN implementation and training. There are two main options: PyTorch PyG and Deep Graph Library (DGL).

By comparing the two tools for the specific use case and scenario of the risk assessment, the PyG has been chosen as the main tool to implement the GNN due to several reasons:

- PyTorch Integration: PyG is built specifically for PyTorch, ensuring seamless integration with PyTorch's ecosystem. This allows for easy use of PyTorch's optimization, loss functions, and other utilities.
- Simplicity and Intuitiveness: PyG offers a more straightforward and intuitive API for defining GNN models, especially for those already familiar with PyTorch. This can lead to cleaner, more readable code.
- Performance: PyG is known for its efficient implementation of graph operations, often providing better performance than DGL, especially for smaller to medium-sized graphs.
- Sparse Tensor Support: PyG has better support for sparse tensor operations, which are crucial for efficient processing of large, sparse graphs.
- Message Passing API: PyG's message passing API is more flexible and easier to customize, allowing for more control over the specifics of how information is propagated through the graph.
- Community and Ecosystem: PyG has a large and active community, resulting in extensive documentation, tutorials, and a wide range of pre-implemented GNN models and layers.
- Scalability: While DGL might have an edge for extremely large graphs, PyG's performance is more than sufficient for most practical applications, including our use case.

While DGL is also a powerful library with its own strengths, particularly in handling very large graphs and supporting multiple deep learning frameworks, PyG's tight integration with PyTorch, intuitive API, and efficiency for the scale of a problem made it the preferred choice for this implementation.

The next step in selecting the appropriate tool for the deployment part of the trained model to test and use for the risk assessment of graphs of different sizes.

This tool on the AWS cloud is Amazon SageMaker as it offers several advantages, among which the main ones are the following: managed infrastructure: SageMaker handles the complexities of provisioning and managing the infrastructure for model deployment. The important feature is automatic scalability of the number of instances based on traffic, ensuring good performance under varying loads. Furthermore, the Sagemaker is easily integrated into AWS Ecosystem: for services like Neptune, Lambda, and CloudWatch to implement a complete solution with the API, logging and monitoring system. Another important feature is built-in security like encryption at rest and in transit, IAM roles for access control, and VPC support [10].

Last but not the least feature is support for custom containers by allowing deployment of models with custom dependencies like PyTorch Geometric in the case of an application with GNN usage.

By using SageMaker for deployment, the focus can be shifted from management of the infrastructure to improving the GNN model and application logic, while leveraging AWS's expertise in scalable, secure, and efficient model serving. This approach aligns well with the use of Amazon Neptune for graph storage, creating a cohesive, fully managed graph-based machine learning solution within the AWS ecosystem.

Development of GCN model designed for risk assessment task

Having designed the graph for training and testing of the model and selected the tools for the implementation, the next logical step is the development of the model itself. There are several GNN architectures, which can be used for the solution of the task. To select the best option for the specific risk assessment use case, it has been decided to solve the binary classification task (risky/non-risky) and the prediction task (number from 0 to 1, which predicts the risk of insuring the individual with 1 being very risky and 0 being not risky at all) by three most common approaches of the GNN architectures and then compare the results of the execution with one another to find out the most suitable option.

The core idea of GNNs is to update node representations by aggregating information from their neighbors. To implement the GCN, GCNConv layers can be used in the Pytorch, simplifying the aggregation step to:

$$H^{i+1} = \sigma(\hat{A}H^{l}W^{l}), \tag{12}$$

where:

- H^{l} is the node feature matrix at layer l;
- $\hat{A} = D^{-1/2} (A + I) D^{(-1/2)}$ is the normalized adjacency matrix with self-loops;
- A is the adjacency matrix;
- D is the degree matrix;
- I is the identity matrix;
- $W^{(l)}$ is the learnable weight matrix at layer l;
- σ is the activation function (ReLU in our case);

The model consists of two GCN layers:

$$H^{(1)} = \mathrm{ReLU}(\hat{A}\ X\ W^{(0)}),$$
$$H^{(2)} = \hat{A}\ H^{(1)}\ W^{(1)},$$

where:

- X is the input feature matrix;
- $H^{(1)}$ is the hidden representation;
- $H^{(2)}$ is the output before softmax.

To optimize the model and adapt the learning rate for every parameter, the optimization function has to be used. Mainly used with GNN is Adam optimizer [11]:

$$\theta^{t} = \theta^{(t-1)} - \frac{\eta * \hat{m}^{t}}{\sqrt{\hat{v}^{t}} + \epsilon}, \tag{13}$$

where:

- θ^{t} is the parameter at time t;
- η is the learning rate;
- $\hat{m}^{t}$ is the first moment estimate;
- $\hat{v}^{t}$ is the second moment estimate;
- ϵ is a small constant for numerical stability.

Evaluation Metric: For binary classification (risky to insure the individual or non-risky), we can use accuracy:

$$Acc = \frac{TP+TN}{T}, \quad (14)$$

where:

- Acc – Accuracy;
- TP – True Positives;
- TN – True Negatives;
- T – Total Samples.

The definition of the GCN model designed for risk assessment and its forward function in PyTorch will look like this:

```
class RiskAssessmentGNN(torch.nn.Module):
    def __init__(self, num_node_features, hidden_channels):
        super(RiskAssessmentGNN, self).__init__()
        self.conv1 = GCNConv(num_node_features, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, 2)  # 2 classes: low risk and high risk
    def forward(self, x, edge_index, edge_attr):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, training=self.training)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)
```

Development of GAT model designed for risk assessment task

The next GNN architecture considered for the development and further comparison is Graph Attention Network (GAT). The approach is described using the previously defined formulas without repeating the same information, introducing only the changes that are different for the two approaches.

The key difference between GAT and GCN lies in how they aggregate information from neighboring nodes. While GCN uses a fixed weighting scheme based on the graph structure, GAT introduces an attention mechanism that allows for dynamic, adaptive weighting of neighbor information.

In contrast with GCN we can define the GAT update rule as following:

$$[\overrightarrow{h_i'} = \sigma\left(W \sum_{j\in\mathcal{N}_i} \frac{1}{\sqrt{d_i d_j}} \overrightarrow{h_j}\right)], \quad (15)$$

where d_i and d_j are the degrees of nodes i and j.

Key Differences between the two include adaptive weighting and multi-head attention. The first one can be explained as such:

GAT: α_{ij} is learned and can be different for each edge.

GCN: Uses fixed weighting $\frac{1}{\sqrt{d_i d_j}}$.

The second one is often employed by the GAT.

GAT often employs multi-head attention [12]:

$$[\overrightarrow{h_i'} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k W^k \overrightarrow{h_j}\right)], \quad (16)$$

where K is the number of attention heads.

It is also important to note that GAT includes self-loops implicitly in the attention mechanism while GCN needs to explicitly add self-loops to the graph.

In summary, GAT's key innovation is the introduction of the attention mechanism α_{ij}, which allows the model to assign different importance to different neighbors, potentially capturing more complex patterns.

Considering the information above, we can define the model and its forward function in Pytorch:

```
class SimpleGATModel(torch.nn.Module):
    def __init__(self, num_features, num_classes):
        super(SimpleGATModel, self).__init__()
        self.conv1 = GATConv(num_features, 16, heads=8, dropout=0.6)
        self.conv2 = GATConv(16*8, num_classes, heads=1, concat=False, dropout=0.6)
    def forward(self, x, edge_index):
        x = F.dropout(x, p=0.6, training=self.training)
        x = F.elu(self.conv1(x, edge_index))
        x = F.dropout(x, p=0.6, training=self.training)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)
```

Development of GraphSAGE model designed for risk assessment task

GraphSAGE has sampling-based approach to the message passing whereas GCN and GAT use all neighbors for each node:

$$[\mathcal{N}(v) = \text{sample}(neighbors(v), S)], \tag{17}$$

where S is a fixed sample size.

Besides that, SAGE also has flexible aggregation functions in comparison to the fixed weighted sum of GCN and attention-weighted sum of GAT:

$$\sigma\left(W^k \cdot \text{CONCAT}\left(h_v^{k-1}, \text{AGG}_k\left(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)\right)\right)], \tag{18}$$

where AGG_k can be mean, max-pooling, LSTM, or other appropriate functions.

Furthermore, GraphSAGE explicitly concatenates the node's own features with aggregated neighbor features and it is designed for inductive learning on unseen nodes. On the other hand, GCN and GAT are primarily transductive, though it's worth mentioning that GAT can be adapted for inductive tasks [13].

One more important characteristic to consider is feature normalization as GCN and GAT typically don't include this step:

$$[z_v = h_v^K / |h_v^K|_2], \tag{19}$$

GraphSAGE often uses a graph-based loss for unsupervised learning:

$$[\mathcal{L} = -\log\left(\sigma(z_v^T z_u)\right) - Q \cdot E_{v_n \sim P_n(v)}\left[\log\left(\sigma\left(-z_v^T z_{v_n}\right)\right)\right]], \tag{20}$$

In summary, GraphSAGE's key innovations are its sampling-based approach, flexible aggregation functions, and explicit design for inductive learning, distinguishing it from both GCN and GAT in terms of scalability and generalization to unseen nodes.

Considering the prior description of GraphSAGE, it can be implemented in the PyTorch library as following:

```
class GraphSAGEModel(torch.nn.Module):
    def __init__(self, num_features, hidden_dim, num_classes):
        super(GraphSAGEModel, self).__init__()
        self.conv1 = SAGEConv(num_features, hidden_dim)
        self.conv2 = SAGEConv(hidden_dim, num_classes)
    def forward(self, x, edge_index):
        x = F.relu(self.conv1(x, edge_index))
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)
```

The practical significance and ways of applying the results

In order to get the value from the implemented GNN architectures, we have to train it on our data defined above and deploy it on the AWS infrastructure. The first step here would be to create a graph described above using Gremlin framework to interact with the Neptune Database using a Python programming language on the Jupyter notebook. After that the created graph can be used for training and testing of all three of the models. The final architecture can be found on the Fig. 5.

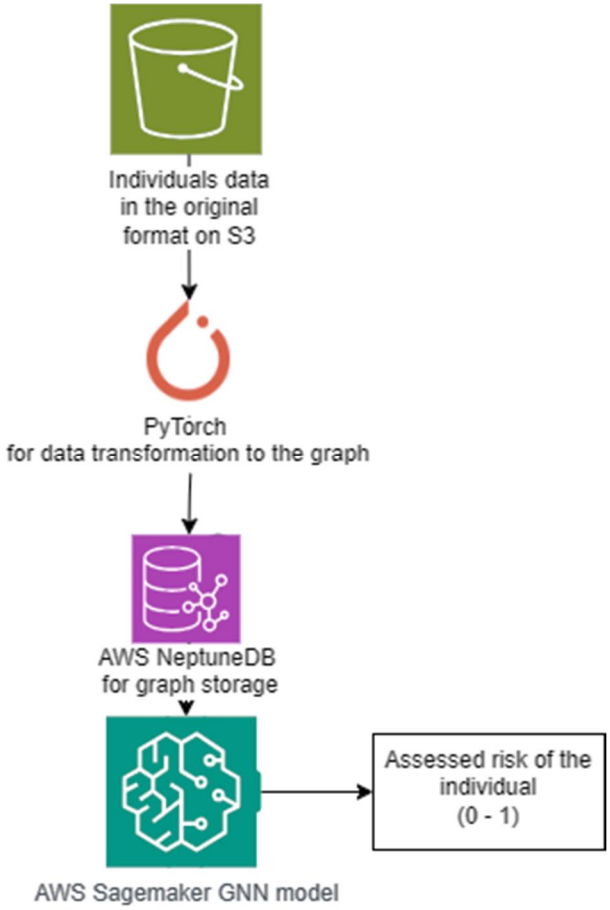

Fig. 5. Final architecture of the risk assessment on AWS infrastructure

The actual real-life company may train the GNN using historical insurance data. However, for our purposes, we will utilize generated and obfuscated data. This approach is necessary because using real insurance data for a model prototype is legally inappropriate. Securing the required approvals and navigating legal protocols to access real data would be time-consuming. To expedite the process, we will use synthetic data, which is based on the real data but obfuscated of any personally identifiable information (PII). The target variable for the specific use case of the trained model applied to the new graph could be a risk score from 0 to 1 or a binary classification (high risk vs. low risk).

It is important to mention that techniques such as dropout, cross-validation and regularization should be employed to prevent overfitting. Preventing overfitting is crucial in machine learning to ensure that models generalize well to unseen data. In this example dropout technique is applied for every GNN. Dropout is a regularization technique used to prevent overfitting in neural networks. For each forward pass during training, dropout randomly sets a fraction of input units to 0.

Typically, the keep probability (1 – dropout probability) for input layers is close to 1, with 0.8 often recommended. In hidden layers, a higher dropout probability leads to a sparser model, with 0.5 being an optimal keep probability, indicating a 50 % node dropout. This basically means that half of the nodes on

the hidden layers will be dropped (or in other words nullified) and only 20 % of the nodes on the input layer will be dropped.

A standard PyTorch training loop is being used for the GNN, with forward passes through the model, loss computation, and backpropagation for node feature updates. As the next step the model's performance is assessed using a separate test function, which computes predictions on a test set and calculates accuracy. Furthermore, some model statistics are collected during its training and testing processes and shown in the output.

Having run all three models on the same graph dataset, the results can be compared [Fig. 6].

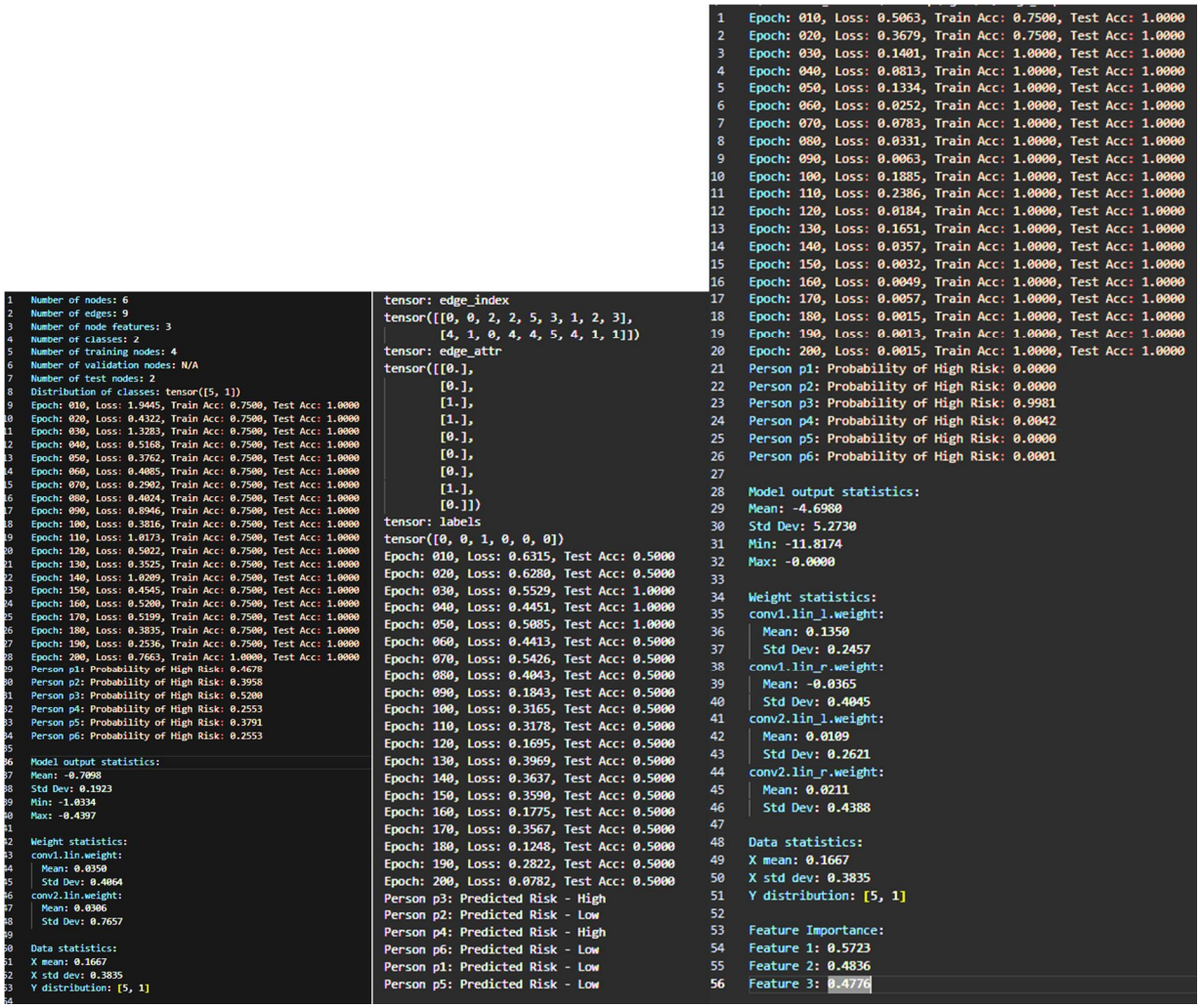

Fig. 6. The results of the execution of 3 GNN models on the same dataset. 1st is GAT, 2nd – GCN, 3rd – SAGE

The following can be said about the input data distribution for all three models: there are 6 nodes, 9 edges, and 3 node features in the graph that describes the individuals willing to insure themselves. There are 2 classes with an imbalanced distribution: 5 cases with low risk of insuring those individuals and only 1 case with high risk who has the bad preconditions for being insured. Finally, the original risk assessment graph is split into smaller datasets with 4 nodes being used for training and 2 of them being used for testing respectively.

To visualize the results of the training and testing of three different GNN model types for the risk assessment task in the insurance area, the comparison table can be built. The table should include the following output parameters of the models: test accuracy, training progression, training accuracy, final predictions and output model statistics (Table 1).

Table 1

Comparison table of the GNN approaches to the risk assessment task

GNN model	Training progression	Test Accuracy	Final Predictions	Model statistics
GCN	Loss generally decreased over epochs, starting at 0.6315 and ending at 0.0782.	Fluctuated between 0.5000 and 1.0000, settling at 0.5000 for the last 100 epochs.	Classified 3 individuals as low risk (p2, p1, p5, p6) and 2 as high risk (p3, p4).	Negative mean outputs (-0.056) with low standard deviation (0.2345).
GAT	Loss fluctuated but generally decreased, starting at 1.9445 and ending at 0.7663. Train accuracy improved from 0.7500 to 1.0000 by the final epoch.	Consistently maintained 1.0000 throughout all epochs.	Provided probabilities of high risk for everyone, with p3 having the highest probability (0.5200) and p4 and p6 having the lowest (0.2553).	Negative mean outputs (-0.7098) with relatively low standard deviation (0.1923).
GraphSage	Loss decreased rapidly, starting at 0.5063 and ending at 0.0015. Training Accuracy improved from 0.7500 to 1.0000 by the 30th epoch and maintained it.	Consistently maintained 1.0000 throughout all epochs.	Provided very distinct probabilities, with p3 having an extremely high probability of high risk (0.9981) and others having very low probabilities (close to 0).	Highly negative mean outputs (-4.6980) with large standard deviation (5.2730).

Considering the table above, it can be seen that the GCN model seems to have struggled with consistency, as evidenced by the fluctuating test accuracy. For GAT weight statistics indicate larger variance in the second layer compared to the first. GraphSAGE has some statistics not available for other tables due to specifics of this model such as weight statistics which indicate relatively balanced means and standard deviations across layers and feature importance analysis, which suggests that Feature 1 (individual health status) is the most important (0.5723), followed by Feature 2 (smoking habit) (0.4836) and Feature 3 (regular checkups) (0.4776) has the least importance according to the results.

In the conclusion, it can be stated based on the provided outputs that GraphSAGE appears to have provided the best results for the following reasons:

- Consistency: GraphSAGE achieved and maintained perfect train and test accuracy from early epochs, indicating stable learning.
- Loss Convergence: The loss decreased more consistently and to a lower final value (0.0015) compared to GCN and GAT.
- Clear Distinction in Predictions: GraphSAGE provided very distinct probabilities for high risk, clearly separating p3 as high risk from the others, which aligns with the original data distribution (5 low risk, 1 high risk).
- Feature Importance: GraphSAGE provided additional insight through feature importance analysis, which can be valuable for interpretation.
- Generalization: Both GraphSAGE and GAT maintained perfect test accuracy, but GraphSAGE's more distinct probabilities suggest better generalization.

GAT at the same time performed well, maintaining perfect test accuracy throughout, but its final predictions were less distinct than GraphSAGE's. GCN struggled with consistency in test accuracy, making it less reliable for this particular dataset.

The superior performance of GraphSAGE might be attributed to its flexible aggregation mechanism and its ability to sample and aggregate neighborhood information effectively, which seems particularly well-suited to this specific graph structure describing the individuals in the insurance area and task of the risk assessment during the underwriting process.

Further steps and improvements may start from handling the class imbalance. Currently, the model seems to handle the class imbalance well. However, it is still considered to use class weights or oversampling techniques to ensure robust performance for much more individuals with more features. The next step can be improving of the model architecture. To capture more complex patterns, experiments can be done with deeper architectures or different GNN layers. Finally, the model must be generalized: with such a small dataset, it's crucial to ensure the model generalizes well. Cross-validation or creating synthetic data to can be done to achieve further validation of the model's performance.

Overall, the models seem to be performing well on this small dataset. The next steps would be to create a larger, more diverse dataset and to validate the model on it, having implemented some of the additional analyses and improvements mentioned above.

Conclusions

In conclusion, the comparison analysis of 3 GNN architectures being Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and GraphSAGE for insurance risk assessment has yielded valuable insights into the application of Graph Neural Networks in the insurance domain. Three models have been described in detail and developed using the available technologies. The models have all been trained and tested on the same dataset, being a graph representing individuals with a task of assessing the risk of insuring them. The goal of analysis was for the models to focus not only on the individual's features, but also consider the connections between them to have unbiased risk assessment.

It is worth noting that considering the intertangled and very dependent on connections nature of the insurance data as well as the huge volumes of the real-world data of insurance companies it has been decided to build a scalable solution on top of the cloud infrastructure, which can handle such task. For this specific purpose, the AWS cloud can be chosen.

Based on the results of implementing three models and assessing the risk with them, the comparison table was built, which has revealed significant differences in the performance of the three models. While all showed promise in tackling the complex task of risk assessment, GraphSAGE emerged as the standout performer. Its superior ability to sample and aggregate neighborhood information proved particularly effective in capturing the nuanced relationships within the insurance dataset. This strength was evident in GraphSAGE's more distinct and confident risk predictions compared to the other models.

The Graph Attention Network (GAT) also demonstrated strong performance, maintaining perfect test accuracy throughout the evaluation. However, its final predictions lacked the clear differentiation seen in GraphSAGE's outputs. This suggests that while GAT's attention mechanism is powerful, it may not have fully captured the specific nuances of our insurance risk assessment task as effectively as GraphSAGE's flexible aggregation approach.

In contrast, the Graph Convolutional Network (GCN) struggled with consistency in test accuracy. This inconsistency makes GCN less reliable for the particular dataset and task at hand, highlighting the importance of choosing the right architecture for specific problem domains.

The superior performance of GraphSAGE can be attributed to its flexible aggregation mechanism and effective sampling of neighborhood information. These characteristics appear to be particularly well-suited to the graph structure representing individuals in the insurance domain and the specific task of risk assessment during the underwriting process. GraphSAGE's ability to capture and leverage the complex interrelationships between policyholders and their attributes has demonstrated its potential to significantly enhance the accuracy and reliability of risk predictions in insurance underwriting.

These findings underscore the importance of carefully selecting and tuning GNN architectures for specific applications in the insurance industry. As the sector continues to embrace data-driven decision-making, the integration of advanced graph-based machine learning models like GraphSAGE which shows the best performance for the risk assessment of the individuals could mark a significant leap forward in insurance capabilities. Future work should focus on further optimizing these models for other specific insurance use cases and improving the models for the current use case, exploring their interpretability to meet regulatory requirements. Investigating their performance on larger and more diverse datasets is required in the future with the focus being on ability to scale and handle data volumes as well as avoiding overfitting and underfitting respectively with different available methods. The parameters of the models may also be slightly improved to capture all the nuances of the bigger datasets with more features.

REFERENCES

1. Danish, A., Xie, Y., Umair, S. B. (2021). Insurers' risk management as a business process: a prospective competitive advantage or not? *Emerald, 31*(3), 345–366. DOI: https://doi.org/10.1108/EJMBE-08-2021-0221

2. Sanchez-Lengeling, et al. (2021). A Gentle Introduction to Graph Neural Networks, *Distill.* DOI: https://doi.org/10.23915/distill.00033

3. Uzair, A. B., Tang, H., Guilu, W., Marjan, S., Hussain, A. (2023). Deep Learning with Graph Convolutional Networks: An Overview and Latest Applications in Computational Intelligence. *International Journal of Intelligent Systems.* DOI: https://doi.org/10.1155/2023/8342104

4. Kipf, T. N., Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* DOI: https://doi.org/10.48550/arXiv.1609.02907.

5. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2017). Graph Attention Networks. *arXiv preprint arXiv:1710.10903v3.* DOI: https://doi.org/10.48550/arXiv.1710.10903.

6. Totaro, S., Hussain, A., Scardapane, S. (2020). A non-parametric softmax for improving neural attention in time-series forecasting. *Neurocomputing, 381,* 177–185. DOI: https://doi.org/10.1016/j.neucom.2019.10.084

7. Zeng, H., Zhang, M., Xia, Y., Srivastava, A., Malevich, A., Kannan, R., Prasanna, V., Jin, L., Chen, R. (2022). Decoupling the depth and score of Graph Neural Networks. *arXiv preprint arXiv: 2201.07858v1.* DOI: http://arxiv.org/abs/2201.07858v1

8. Liu, T., Chen, Y., Li, D., Wu, C., Zhu, Y., He, J., Peng, Y., Chen, H., Guo, G. (2021). BGL: GPU-Efficient GNN Training by Optimizing Graph Data I/O and Preprocessing. *arXiv preprint arXiv: 2112.08541.* DOI: https://doi.org/10.48550/arXiv.2112.08541

9. Tian, Y. (2022). The World of Graph Databases from an Industry Perspective. *arXiv preprint arXiv:2211.13170.* DOI: https://doi.org/10.48550/arXiv.2211.13170

10. Wang, Z., Ioannidis, V. (2022). *How AWS uses graph neural networks to meet customer needs*. Amazon Science. https://www.amazon.science/blog/how-aws-uses-graph-neural-networks-to-meet-customer-needs

11. Zhang, Z. (2018). Improved Adam Optimizer for Deep Neural Networks. *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS).* DOI: 10.1109/IWQoS.2018.8624183.

12. Cordonnier, J., Loukas, A., Jaggi, M. (2021). Multi-Head Attention: Collaborate Instead of Concatenate. *arXiv preprint arXiv:2006.16362.* DOI: https://doi.org/10.48550/arXiv.2006.16362

13. Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K. (2020). Inductive Representation Learning on Temporal Graphs. *arXiv preprint arXiv:2002.07962.* DOI: https://doi.org/10.48550/arXiv.2002.07962

СПИСОК ЛІТЕРАТУРИ

1. Санчез-Ленделінг та ін. (2021). Лагідний вступ до графових нейронних мереж. *Distill.* DOI: https://doi.org/10.23915/distill.00033

2. Узаір, А. Б., Танг, Н., Ґуілу, В., Мар'ян, С., Хуссейн, А. (2023). Глибоке навчання з графовими згортковими мережами: огляд та останні застосування в обчислювальному інтелекті. *Міжнародний журнал інтелектуальних систем.* DOI: https://doi.org/10.1155/2023/8342104

3. Кіпф, Т. Н., Welling, M. (2016). Напівконтрольована класифікація з графовими згортковими мережами. *arXiv препринт: arXiv:1609.02907* DOI: https://doi.org/10.48550/arXiv.1609.02907.

4. Велічковіч, П., Кукурулл, Ґ., Казанова, А., Ромеро, А., Ліо, П., Бенгіо, Й. (2017). Графові мережі уваги. *arXiv препринт arXiv:1710.10903v3.* DOI: https://doi.org/10.48550/arXiv.1710.10903.

5. Тотаро, С., Хуссейн, А., Скардапане, С. (2020). Непараметричний софтмакс для покращення нейронової уваги при прогнозуванні часових рядів. *Нейрокомп'ютинг, 381*, 177–185. DOI: https://doi.org/10.1016/j.neucom.2019.10.084

6. Зенг, Х., Жан, М., Чіа, Й., Срівастава, А., Малевіч, А., Каннан, Р., Прасанна, В., Джін, Л., Чен, Р. (2022). Відокремлення глибини та оцінки графових нейронних мереж. *arXiv препринт arXiv: 2201.07858v1*. DOI: http://arxiv.org/abs/2201.07858v1

7. Лю, Т., Чен, Й., Лі, Д., Ву, К., Жу, Й., Хі, Й., Пен, Й., Чен, Х., Ґуо, Ж. (2021). BGL: GPU-ефективне навчання GNN шляхом оптимізації введення-виведення та попередньої опрацювання графових даних. *arXiv препринт arXiv: 2112.08541*. DOI: https://doi.org/10.48550/arXiv.2112.08541

8. Тіан, Й. (2022). Світ графових баз даних з точки зору індустрії. *arXiv препринт: arXiv :2211.13170*. DOI: https://doi.org/10.48550/arXiv.2211.13170

9. Ван, З., Йоанідіс, В. (2022). *Як AWS використовує графові нейронні мережі для задоволення потреб клієнта*. Amazon Science. https://www.amazon.science/blog/how-aws-uses-graph-neural-networks-to-meet-customer-needs

10. Жан, З. (2018). Вдосконалений Оптимізатор АДАМ для глибоких нейронних мереж. *2018 IEEE/ACM 26-ий Міжнародний Симпозіум по якості послуг (IWQoS)*. DOI: 10.1109/IWQoS.2018.8624183.

11. Кордоньєр, Х., Лукас, А., Джаґґі, М. (2021). Увага до багатьох вершин: співпраця замість об'єднання. *arXiv препринт arXiv:2006.16362*. DOI: https://doi.org/10.48550/arXiv.2006.16362

12. Шу, Д., Руан, К., Корпеоґлу, Е., Кумар, С., Ашан, К. (2020). Навчання індуктивного представлення на часових графах. *arXiv препринт arXiv:2002.07962*. DOI: https://doi.org/10.48550/arXiv.2002.07962

ПІДХОДИ ВПРОВАДЖЕННЯ ГНМ У ХМАРНИХ СЕРВІСАХ AWS ДЛЯ ОЦІНКИ РИЗИКІВ У СФЕРІ СТРАХУВАННЯ

Олександр Луценко[1], Сергій Щербак[2]

[1, 2] Національний Університет "Львівська політехніка",
кафедра інформаційних систем та мереж, Львів, Україна
[1] E-mail: Oleksandr.V.Lutsenko@lpnu.ua, ORCID: 0009-0008-7644-6056
[2] E-mail: serhii.s.shcherbak@lpnu.ua, ORCID: 0000-0003-2914-2101

У статті проаналізовано три найпоширеніші підходи до реалізації архітектури ГНМ у хмарних сервісах AWS для оцінки ризиків у сфері страхування. Стаття поділена на кілька розділів, перший з яких містить огляд 3-х підходів до архітектури ГНМ, другий – описує передумови для впровадження і, нарешті, впровадження та порівняння всіх підходів для вибору найкращого.

У першому розділі представлено три архітектурні підходи до впровадження ГНМ, а саме: Графові Мережі Згортки (ГМЗ), Графові Мережі Уваги (ГМУ) і GraphSAGE (Графові збір та агрегація). Архітектури описані з акцентом на математичні формулювання. На завершення розділу вирішено продовжити подальше впровадження всіх трьох моделей в інфраструктуру AWS і проаналізувати результати на тих самих графових даних, щоб вибрати найкращий варіант для оцінки ризику для страхової компанії.

Далі стаття продовжує вибір інструментів і підготовку даних для їхнього подальшого використання в цілях навчання та тестування ГНМ. Після аналізу варіантів використання вирішено зосередитися лише на страхуванні фізичних осіб. Основна мета полягає в аналізі унікальних властивостей кожної людини, які можуть впливати на ризик її страхування, а також на її зв'язки з іншими особами. Основними інструментами для використання є NeptuneDB для зберігання графів і Sagemaker для розгортання та навчання моделі. Стаття також зосереджена на виборі відповідного інструменту для реалізації, порівнюючи два найбільш використовувані фреймворки Python PyTorch PyG і Deep Graph Library (DGL). До того ж пріоритет надано PyG.

Далі описано процес впровадження всіх трьох підходів: спочатку ГМЗ, потім ГМУ і, нарешті, GraphSage. Далі моделі піддаються навчанню та тестуванню з подальшим аналізом вихідних даних. З огляду на результати аналізу, ГМУ і GraphSage забезпечують найбільш точні результати, зберігаючи точність під час тестування. Однак враховуючи статистичні дані обох моделей, виявлено, що GraphSage має більшу різницю між імовірностями ризиків та додаткові відомості завдяки аналізу важливості особливостей даних, що робить його найкращим для сценарію використання оцінки ризику.

На завершення статті зазначено, що з усіх трьох проаналізованих архітектур найбільш придатною для завдання оцінки ризику є GraphSAGE з невеликою перевагою цієї моделі над ГМУ, відповідно її вирішено використати для подальшого аналізу та вдосконалення. Крім того, у статті згадується кілька кроків для потенційного майбутнього вдосконалення моделей, а також зосереджено увагу на тестуванні та навчанні на більшому наборі даних, щоб зробити його більш застосовним для реальних програм.

Ключові слова: Графові Згортальні Мережі (ГЗМ), Графові Мережі Уваги (ГМУ), GraphSAGE (Збір та агрегація), Граф, Графові нейронні мережі (ГНМ), Андеррайтинг, Страхування, Оцінка Ризиків.