Quantitative Finance and Economics

https://www.aimspress.com/journal/QFE

QFE, 9(2): 449–478.
DOI: 10.3934/QFE.2025015
Received: 05 January 2025
Revised: 16 April 2025
Accepted: 16 May 2025
Published: 04 June 2025

Research article

Optimizing B2B customer relationship management and sales forecasting with spectral graph convolutional networks: A quantitative approach

Shagufta Henna*, Shyam Krishnan Kalliadan and Mohamed Amjath

Department of Computing, Atlantic Technological University, Donegal, Ireland

* **Correspondence:** Email: shagufta.henna@atu.ie.

Abstract: Customer relationship management (CRM) in business-to-business (B2B) environments requires robust strategies and informed decision-making to cultivate strong inter-business relationships, which are pivotal for achieving competitive advantage and maximizing profitability. Traditional CRM analytics, which leverages conventional data mining, machine learning, and deep learning techniques, often fails to address the intricate and interdependent nature of B2B systems. To overcome this limitation, we proposed a spectral graph convolutional neural network (GCN) approach that utilized graph-based modeling to capture the structural complexity of B2B CRM. Companies were represented as nodes, and their interactions as edges within a graph, enriched with Eigenvector centrality and shortest-path graph features, which were particularly suited for spectral GCN operations. Using graph Laplacian-based convolutions, the spectral GCN effectively aggregated global and local relational information, enabling accurate and scalable B2B sales predictions. Experimental evaluations demonstrated that GCN models with spectral attributes significantly outperformed state-of-the-art machine learning and deep learning models, including random forests, convolutional neural networks, feed-forward neural networks, Extreme Gradient Boosting (XGBoost), and Categorical Boosting (CATboost), in terms of accuracy, F1 Score, precision, and specificity. Among the models, the GCN with Eigenvector features achieves the best classification performance, with a high Receiver Operating Characteristic (ROC) value of 0.924, further demonstrating its robustness against variations in feature correlations.

Keywords: graph convolutional neural networks; business-to-business CRM; spectral graph modeling; eigenvector centrality; sales prediction; machine learning analytics

JEL Codes: C45, M15

1. Introduction

An intelligent customer relationship management (CRM) system has the potential to significantly enhance the customer experience, improve brand awareness, and drive profitability. However, developing a cost-effective solution that meets the complex demands of a business-to-business (B2B) value chain remains a formidable challenge. A data-driven approach to model CRM systems can address customer-centric goals by extracting, storing, analyzing, and predicting potential customers and business opportunities. Graph-based modeling is particularly effective for this purpose, as it can abstract and capture complex data relationships that go beyond the limited scope of traditional business analytics. Despite the growing adoption of CRM data lakes, existing analytics methods often struggle to capture intricate relationships that are crucial to gain competitive advantage in complex B2B environments.

The rise of CRM data lakes has opened up new opportunities to leverage graph-based storage and analytics to improve business performance. Recent efforts to apply machine learning to CRM analytics have led to increased efficiency in certain areas. For instance, Zhang et al. (2019) employed semi-supervised spectral clustering for customer behavior analysis, while Yang et al. (2018) developed a recommender system for B2B sales and marketing. Furthermore, Konno et al. (2017) applied graph models to enhance business operations. However, despite these advances, the application of graph-based techniques in CRM remains underexplored, particularly within B2B contexts. Recent studies underscore the transformative potential of intelligent CRM systems that integrate artificial intelligence and big data for real-time strategic decision making (Nugmanova et al., 2019; Taleb et al. , 2020; Kim et al. , 2019).

Relational database management systems (RDBMS) fail to model the meaningful and critical relationships required for intelligent enterprise CRM solutions. In contrast, graph databases such as Neo4j are specifically designed to capture the inherent relationships present in complex CRM scenarios. Neo4j, a leading open source graph database, offers robust, scalable storage solutions and powerful graph querying and mining capabilities (Huang and Dong , 2013). Using a flexible schema, Atomicity, Consistency, Isolation, Durability (ACID) compliance, and Cypher query language to facilitate advanced graph analytics. Although graph databases have been extended to analyze customer relationships through centrality measures (Kimura et al., 2011; McClanahan and Gokhale , 2016), key graph features, such as Eigenvectors and shortest path distances, remain largely underutilized in CRM analytics.

Graph-based deep learning techniques, particularly graph convolutional networks (GCNs), have emerged as powerful tools to extract meaningful patterns and insights from graph data (Gheisari et al. , 2017; Scarselli et al., 2009). Spectral GCNs, which leverage Fourier transforms and Laplacian-based graph convolutions, efficiently aggregate neighborhood information while capturing global and local structural properties (Bruna et al., 2014; Defferrard et al., 2016).

Despite recent advances in CRM, which have incorporated machine learning and graph-based models, these efforts have been limited in their ability to capture the complex relational dependencies inherent in B2B CRM systems. Traditional models often focus on individual customer behaviors or static relationships, overlooking the broader, dynamic networks of inter-business connections that are central to B2B CRM. This limitation hinders the ability to model the full spectrum of relationships that impact sales outcomes, customer loyalty, and business opportunities. Moreover, critical graph features—such as Eigenvector centrality and shortest-path distances—are underexplored, despite their potential to capture important relational structures in B2B networks.

This study addresses these gaps by proposing a graph-based deep learning framework for CRM in B2B settings. We design a graph model using the Neo4j platform to represent B2B sales data, incorporating key relational features like Eigenvector centrality and shortest-path distances. We then applied spectral GCN to this graph model, enabling accurate and scalable sales predictions. The proposed approach demonstrates significant improvements over traditional machine learning and deep learning models, including random forests (RFs), Extreme Gradient Boosting (XGBoost), Categorical Boosting (CATboost), convolutional neural networks (CNNs), and feed-forward neural networks (FNNs). The major contributions of the paper are summarized as follows:

- We develop a graph model for a B2B sales dataset using Neo4j, which supports advanced graph data mining and querying using Cypher, providing valuable insights into CRM interactions.
- We apply spectral GCN to leverage graph features such as Eigenvector centrality and shortest-path distances for sales predictions, showcasing the effectiveness of graph-based learning.
- Experimental results demonstrate that spectral GCN outperforms RFs, gradient-boosting classifiers, CNNs, and FNNs across multiple evaluation metrics, including accuracy, F1 score, and Area Under the Curve (AUC).

The remainder of this paper is structured as follows. Section 2 critically reviews the existing literature on business analytics, with an emphasis on machine learning and deep learning approaches relevant to CRM systems. Section 3 details the B2B dataset employed in this study, highlighting its structure and relevance to the proposed approach. Section 4 introduces the design of the graph model, derived from the Neo4j platform, utilizing advanced graph data mining techniques. In Section 5, we describe the representation features of the graph model that are subsequently utilized for GCN-based inference. Section 6 elaborates on the proposed spectral GCN-based approach for CRM analytics, including its architecture and implementation. Section 7 presents a comprehensive evaluation of the proposed GCN-based model, encompassing exploratory data analysis using Cypher queries, as well as detailed results and performance analysis of the GCN model under various graph feature configurations. Finally, Section 8 concludes the paper by summarizing the key contributions and suggesting promising avenues for future research.

2. Related work

This section investigates various works relevant to graph databases and graph-based deep learning for business analytics.

2.1. Graph database and graph models

Traditional databases and data warehouses cannot process or store unstructured data. Not Only SQL (NoSQL) databases and Hadoop distributed storage systems are alternative solutions under such scenarios. NoSQL database framework, however, requires efficient graph data processing where a graph represents real-life entities and relationships between the entities in a more meaningful manner. Graph representation of big data has significant applications in genetics, social networks, molecular chemistry, finance, and drug testing (Huang and Dong , 2013). A graph database based on a graph model can store, process, and perform efficient graph analysis, such as graph-based deep learning. Huang and Dong (2013) analyzed the performance of Neo4j Cypher query in various B2B use cases. Neo4j stores the graph data as a record file and uses two caching mechanisms for faster data retrieval and visualization. The first mechanism stores the reference relationship of nodes with minimal information in the file system cache. The second mechanism stores major graph connectivity and node attribute information in the object cache. An authors group of researchers analyze the performance of different NoSQL databases and recommend Neo4j and Arango database as highly scalable graph databases for enterprises (Das et al., 2020). Neo4j-based graph analysis and data mining have applications in various domains including business-to-consumer (B2C) and B2B. Hoksza and JelínekIn (2015) applied Neo4j to perform data mining on various protein graphs. Despite its significant benefits, Neo4j has performance limitations in terms of graph model complexity proportional to dataset size. Similarly, the performance of Cypher queries is limited to larger subgraphs with a higher degree of connectivity. Neo4j graph database analysis has been widely investigated and adopted in the healthcare domain. Zhao et al. (2019) proposed a graph model in Neo4j with the help of Cypher queries for data analysis and disease prediction. Cypher queries can extract hidden patterns to reveal meaningful insights into various business trends. Zou and Liu (2020) analyzed air crash incidents from 1908 to date using Neo4j and Cypher query-enabled data mining to compare the performance of various data import methods in Neo4j. Among the various methods used for import operation, batch-wise import, and Neo4j admin import are suggested as the fastest methods for large datasets. Another work proposed by Needham and Hodler (2019) extracts and analyzes various hidden patterns and relationships in the file dataset using the Neo4j graph database platform and Cypher query. The application of graph representation learning and analysis for CRM based on the social network visual analytic tool (VisCRM) was introduced by Ye et al. (2008). The proposed model extracts hidden features from the customers in a social network with the help of a graph model. In its practical application, the model is limited to visual graphs and exploratory analysis and is unable to perform predictions. Another application of Neo4j in CRM is for goods recommendations using retail knowledge graph and jess reasoning engine (Konno et al., 2017). The work constructs a graph model based on the retail ontology that is queried and analyzed using the Neo4j framework with recommendations offered by the jess reasoning engine. According to Wang et al. (2014), a social network-based enterprise relationship graph can deliver higher customer value and business success rates.

Aasman (2017) presented a knowledge graph using customer data with a 360-degree view of a business's client data catalog. Work considered a customer perspective as an operational knowledge, product knowledge, and service knowledge graph with an analysis of the enterprise data lake. Saha and Sahoo (2018) implemented a graph clustering technique using the telecommunication call records dataset. The authors compared GCN with the K-means algorithm using various features. Zhang et al. (2019) proposed a convolutional graph neural network clustering method using a sentiment lexicon extracted from the social network graph model. In the work, authors implemented three specific models for constructing a topic-specific sentiment lexicon; a filtering text model, a sentiment relationship graph model, and a graph clustering model. The proposed graph model with clustering outperforms the traditional lexicon model with the help of sentiment analysis. The performance of the proposed model decreases with an exponential increase in data collected from the social network.

2.2. *Graph-based deep learning*

Grap-based deep learning has been widely adopted in different domains, including healthcare, businesses, and social networks. It is mainly used for graph representation learning and graph classification. Graph representation learning is the process of structural data encoding of a graph network (Cai et al. , 2018). The encoded information is mapped to a low dimensional vector space, such as adjacency or Laplacian matrices. These matrices are used in machine learning and data analysis operations. An example of graph representation learning using graph neural networks (GNNs) and its variations presented by Scarselli et al. (2009). Graph classification methods can perform node, link, and graph level classifications to classify latent features and knowledge based on graph model (Zhang et al. , 2022). The applications of graph classification methods include network behavior prediction, graph matching, and graph generation. Node-level classification is used for node clustering, node recommendation, link prediction, node prediction, and retrieval. Various variants of GCNs, such as spectral and spatial models, have been developed to address real-world challenges, including those in the B2B domain (Zhang et al., 2019; Bruna et al., 2014). Spectral GCNs, for instance, initially relied on Chebyshev polynomial-based approximations (Defferrard et al., 2016), which were later simplified by Kipf and Welling (2017) using first-order spectral propagation. While these methods introduced mathematical rigor to graph-based learning, their scalability to larger, complex graphs with dynamic relationships remains a challenge due to computational inefficiencies in spectral approaches. Spatial graph convolution networks aim to overcome these limitations by directly aggregating information from a node's local neighborhood. These models, which benefit from learnable parameters that are independent of graph size, generalize better across diverse scenarios (Duvenaud et al., 2015). However, spatial GCNs often rely on localized structural information, limiting their ability to capture global relational patterns that are critical for complex tasks like B2B CRM. More advanced models, such as the diffusion convolutional neural Network proposed by Zhuang and Ma (2018), compute node receptive fields based on diffusion transition probabilities. While this approach enhances relational modeling by incorporating probabilistic transitions, its performance often degrades when applied to dynamic and large-scale graphs due to inherent inefficiencies in diffusion-based methods. Similarly, approaches, such as random-walk-based convolution (Alomrani et al., 2024) exhibit constrained performance on larger graphs due to fixed walk lengths, which fail to fully exploit the underlying graph structure.

Graph sample and aggregated embeddings (GraphSAGE) is a spatial GCN designed for scalable representation learning on large and dynamic graphs by aggregating features from a node's neighborhood (Hamilton et al. , 2017). It introduces flexible aggregation techniques, including mean, long short-term memory (LSTM), and pooling functions, to capture local structural information. These aggregated features are then propagated through a neural network for downstream tasks such as prediction or classification. While GraphSAGE significantly improves scalability and adaptability for dynamic graphs, it primarily relies on localized feature aggregation, which can limit its ability to capture complex global graph structures. mixture model networks (MoNET) takes a step further by generalizing graph learning techniques through the integration of spatial and spectral graph convolution approaches (Monti et al., 2017). By employing a parametric kernel on pseudo-coordinates, MoNET efficiently models a node's neighborhood and learns shareable features across the graph. This hybrid approach enables MoNET to leverage both local and global structural properties, making it more versatile in addressing diverse graph-based problems.

Recently, self-attention and multi-head attention mechanisms have been integrated into GCNs to address their inherent limitations, giving rise to graph attention networks (GATs) (Veličković et al., 2018). These approaches employ attention mechanisms to dynamically identify critical nodes in variable-sized input graphs and learn node representations through weighted convolution operations. By extracting hidden features of each node using self-attention, GATs exhibit superior adaptability and learning capabilities, particularly for complex and unseen graphs. An advanced version of GATs incorporates a multi-head attention mechanism (Yu et al., 2018; Chaudhari et al., 2021), which employs parallelized self-attention layers to assign differentiated priorities to various sets of nodes. This enables the model to simultaneously learn from multiple perspectives within the graph, improving representational diversity and learning stability. While these attention-based methods have demonstrated impressive performance gains, particularly in capturing local dependencies and identifying key nodes, they exhibit notable limitations. One critical drawback is their constrained scalability, as the computational cost of attention mechanisms grows exponentially with the size of the graph, making them less suitable for large-scale or high-density datasets.

Martínez et al. (2019) employed GCNs for customer prediction within a customer-supplier graph network. The study developed a risk assessment model leveraging topological graph metrics, such as clustering coefficient, node degree, and PageRank, integrated with GCNs. The customer-supplier graph captured relationships based on contact sharing, financial flows, and other domain-specific features, with a dataset comprising 168,305 company nodes and 310,084 edges. While this model outperformed a baseline GCN model and emphasized the importance of relationship types in the graph structure, its approach was limited by its reliance on static graph metrics, which may not adequately capture the nuanced, evolving relationships critical in large-scale, dynamic B2B CRM scenarios. Similarly, Kim et al. (2019) proposed a GCN-based model for supply-demand prediction in a public bike-sharing environment, utilizing temporal and spatial node features to predict hourly demand. The model demonstrated responsiveness to sudden changes in global features, such as weather conditions, showcasing the utility of GCNs in dynamic business contexts. However, both studies focused primarily on spatial features and static graph metrics, neglecting the incorporation of spectral features—such as Eigenvector centrality or spectral clustering—which are crucial for capturing global graph properties and hierarchical relationships in complex networks.

An investigation of current approaches reveals that graph-based deep learning has demonstrated significant promise across various domains, including business analytics. Spectral GCNs, which utilize global graph properties derived from Laplacian matrices, have been effective in capturing overarching structural patterns. However, their scalability remains a challenge for large, dynamic graphs due to high computational costs (Kipf and Welling , 2017; Defferrard et al., 2016). On the other hand, spatial GCNs, such as GraphSAGE, focus on aggregating local neighborhood features, offering improved scalability but often at the cost of missing global relational insights crucial for complex B2B CRM tasks (Hamilton et al. , 2017).

Hybrid approaches, such as MoNET, aim to combine spectral and spatial methods to provide more comprehensive models, but still face limitations in handling large-scale datasets due to computational inefficiencies (Monti et al., 2017). Attention-based models like GATs introduce self- and multi-head attention mechanisms to enhance adaptability and learning from diverse node features (Veličković et al., 2018; Chaudhari et al., 2021). While these models show promise in identifying critical nodes and relationships, their scalability remains constrained by the exponential computational cost associated with attention mechanisms. In CRM applications, existing models such as those for customer prediction and supply-demand forecasting have predominantly relied on static graph metrics and localized node features (Martínez et al., 2019; Kim et al. , 2019). These methods fail to capture spectral features, like Eigenvector centrality and spectral clustering, which are essential for modeling global, hierarchical relationships in complex B2B networks.

Current B2B sales prediction models are limited by their inability to fully integrate spectral features and global relationships into graph-based learning frameworks. This oversight hinders their performance in capturing the intricate, dynamic relationships in B2B CRM scenarios. Therefore, there is a need for scalable graph-based models that combine both spectral and spatial features to enhance predictive accuracy and address the complexity of real-world B2B environments.

3. Description of dataset

In our work, we have used the publicly available B2B dataset, a real-world dataset from Salvirt Limited (B2B Dataset , 2020). The selected dataset does not contain any sensitive information regarding clients, business products, or strategies. It consists of anonymized sales data from a real-world organization trading in software solutions and services at the global level. The B2B marketing and sales process follows an auxiliary approach and a structural procedure for establishing connections among clients, which plays a significant role in CRM. The dataset includes 448 training instances with 23 features/attributes, with a sales status column as the class label. Initially, the raw dataset does not include a unique identifier for each sale. Therefore, we added a column, "sales_enquiry_id", during the data preprocessing step, representing the unique sales ID for each sale transaction. All features in the dataset are of the object data type, representing categorical variables. After preprocessing, the dataset consists of 449 training samples with 24 attributes, including one labeled sales status column and the newly created "sales_enquiry_id" column with 448 unique values. A detailed description of each dataset feature is presented in Table 1.

4. CRM graph model

To realize the full benefits of graph-based deep learning, in this section, we present the design of two graph models for the CRM called the exploratory data analysis graph model (EDA-Graph model) and GCN-Graph model, respectively. Graph models capture all the meaningful information from the CRM dataset (B2B Dataset , 2020). EDA-Graph model is useful for query-based data mining and exploratory analysis to identify pattern recognition coupled with interactive query-answering abilities. The second graph model represents the interconnectivity of "sale_enquiry_id" nodes. Both the graph models are the outcome of testing various graph models evaluated in terms of efficient data mining and querying capabilities. In both graph models, nodes and relationships are primarily modeled entities according to the selected use case. The EDA-Graph model is defined using 7 labels, and 7 different relationships between node entities. EDA-Graph model is presented in Figure 1, representing various entities/objects in the dataset. In the graph, the start node and target node depict the direction of the relationship. Neo4j has equal traversal performance in both directions that can query the association without specifying any direction (Robinson et al. , 2020).

To build the graph model, we consider 7 attributes from the dataset to represent real-life entities, i.e., nodes with labels. The remaining attributes are assigned as features to each node, or as relationship properties. For the sales label, the sales_enquiry_id column is assigned as the first node attribute that creates 448 nodes of unique sale inquiries. Similarly, for the label "product", 14 unique product labels are created by assigning the product column in the dataset as the first node attribute. Other features, i.e., "Forml_tend", "RFI", "RPF", and "Cross_sale" that define sales constraints are defined as node

Table 1. B2B sales features description.

No.	Feature Name	Code	Description
1	Product Name	Product	Offered product code
2	Seller Name	Seller	Name of the in-charge seller
3	Authority	Authority	Authority level at the client side
4	Company Size	$Comp\_size$	Size of the client or company
5	Competitors	Competitors	Competitors for a sale
6	Purchasing Department	$Purch\_dept$	Purchasing department involved
7	Partnership	Partnership	Product is being sold in partnership
8	Budget Allocation	$Budget\_alloc$	Reservation of budget by the client
9	Formal Tender	$Formal\_tend$	Tendering procedure
10	RFI	RFI	Request for Information
11	RFP	RFP	Request for Proposal
12	Growth of a Client	Growth	Growth status of the client
13	Positive Statements	$Posit\_statm$	Client's positive attitude
14	Source	Source	Source of sales inquiry
15	Client	Client	Type of client
16	Scope Clarity	Scope	Clarity of implementation scope
17	Strategic Deal	$Strat\_deal$	Deal with a strategic value
18	Cross Sale	$Cross\_sale$	Different product sold to a client
19	Up Scale	Up_sale	Upgrading or increasing an existing product
20	Deal Type	$Deal\_type$	Type of a sale or business requirement
21	Needs Defined	$Needs\_def$	Clearly expressed needs of a client
22	Attention to Client	Att_t_client	Attention/importance given to a client
23	Status	Status	Outcome of a sales opportunity
24	Sales Inquiry ID	$Sales\_enquiry\_id$	Unique sales inquiry ID

attributes of that label. Each sales node contains its corresponding values described by the assigned node attributes. Similarly, the client label is described by assigning the "client" column as a node value. Other dataset features like "Comp_size", "Budget_alloc", and "Growth" that describe the individual property of a client are set as node attributes. "Competitors" and "Strat_deal" columns of the dataset are assigned as the attributes of the relationship "selling_to" and are defined between the sale and client node. At the implementation level, constraints, nodes, relationships, and properties are defined using the Neo4j Cypher query. The dataset is imported and represented as the graph model using the Neo4j user interface. Figure 1 illustrates the EDA-Graph model. Figure 2 presents the graph model of sales id-1001 extracted using the Cypher query language. Each label and related nodes are represented using different colors. The number of nodes under a specific label is also evident in Figure 2.

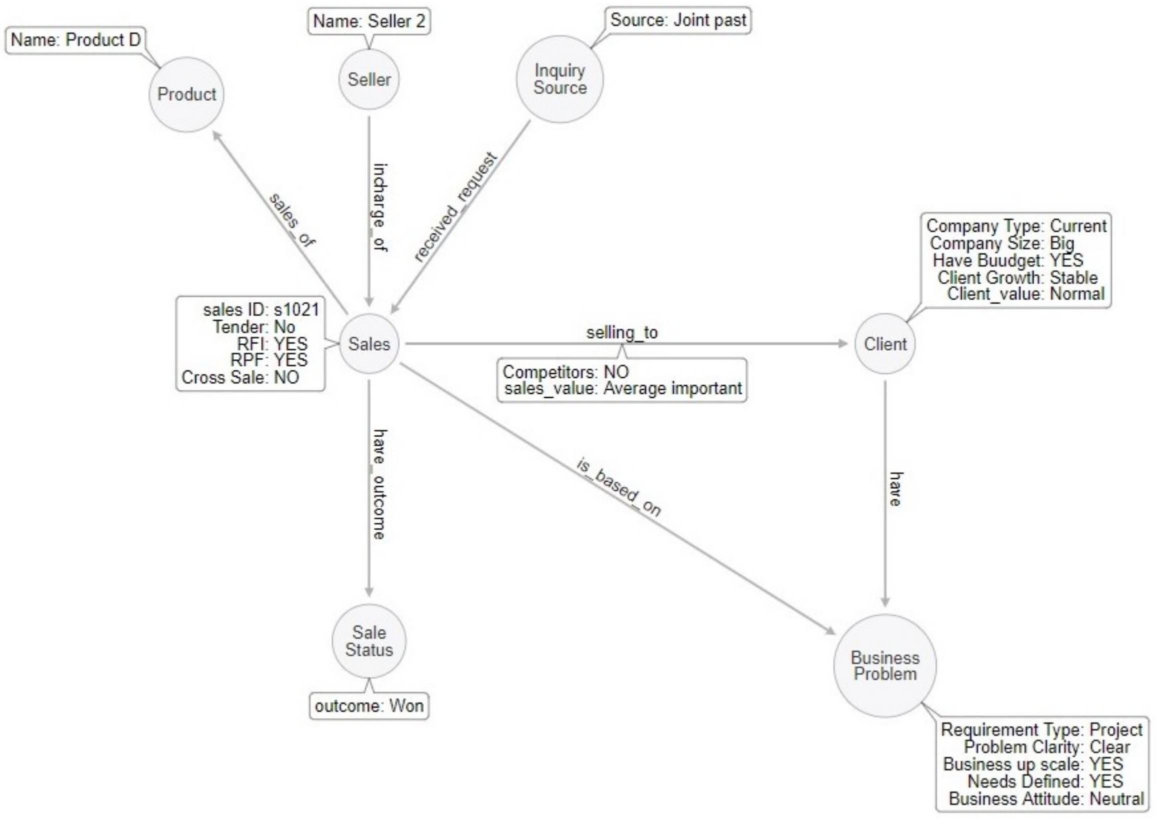


Figure 1. EDA-Graph model.

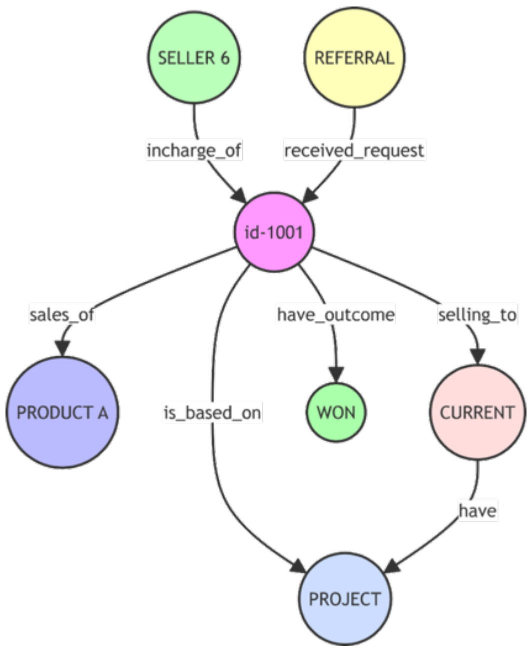


Figure 2. EDA-Graph model example: sale id: id-1001.

The value corresponding to each label for a node is inside the node circle. As shown in Figure 2, the sales id-1001 denotes the sale attempt of product A to an existing client based on business requirements.

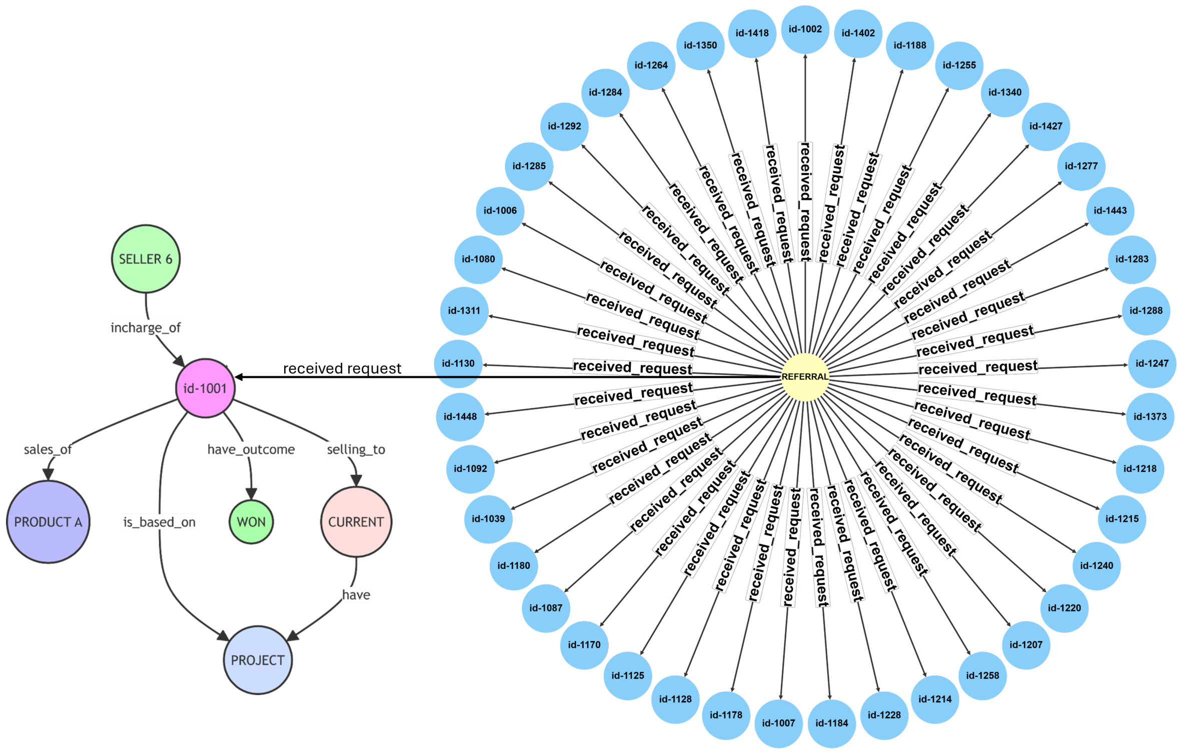


Figure 3. EDA-Graph model- sale ID: 1001 and all sale ID's of enquiry = REFERAL.

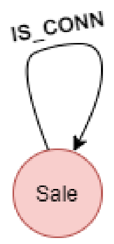


Figure 4. Single label graph schema.

Node "seller 6" represents a seller who is in charge of the sales, and the source of the sale inquiry is the node "referral". From the Figure, it is observed that sales id-1001 has a successful sales outcome and satisfies project requirements. Figure 3 presents the graph model of the sales id-1001 along with a list of all other sales nodes whose inquiry source is also "Referral". The EDA- Graph model extracts direct visualization instances using the Cypher query.

The second graph model, i.e., the GCN-Graph is extracted from the first graph model by defining the interconnectivity of "sale_enquiry_id" nodes. Figure 4 depicts the single-label graph schema visualization of the GCN-Graph model The arrow indicates the connection nodes of the sales label. A densely connected graph network of sales nodes is created using "Up_sale", "Client", "Competitors", "Product", and "Seller" attributes of the dataset. These features are selected based on their importance and influence on the sales status prediction, and we use these features to define the relational connectivity between the nodes. The newly created GCN-Graph model consists of 367 nodes and 423 relationships. Figure 5 depicts a 50-node instance of the GCN-Graph model. We have used this model to extract the node list and the attribute list to implement the GCN model.

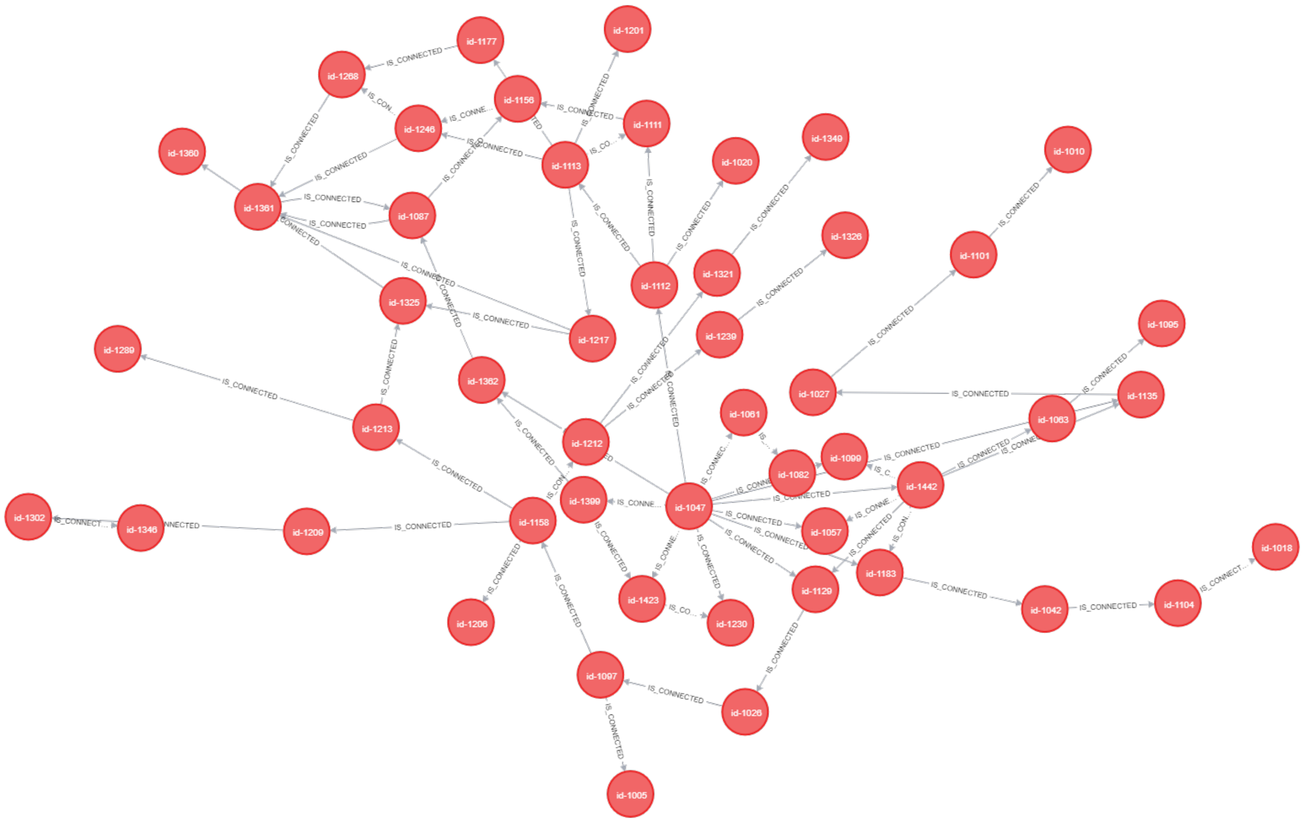


Figure 5. GCN-Graph model: graph model with 50 sales nodes.

5. Feature engineering

To prepare features appropriate for the GCN model, in this section, we discuss the feature engineering tasks. In the first instance, we have transformed the status column to an integer. Similarly, the class labels "Won" and "Loss" are converted to 1 and 0 binary values. We have also dropped the "sale_enquiry_id" column which does not influence the node classification or label prediction. Further, we have applied one-hot encoding on categorical features for training the GCN. The dataset is partitioned into X_train, y_train datasets with an 80:20 ratio. For a steady and unbiased execution, the GCN model is tested on unseen data to establish its generalization. The evaluation results demonstrate that the GCN model performs sales classification with generalization.

The GCN model is trained using the GCN-Graph model discussed earlier. GCN-Graph model consists of an edge list, "CRM.edgelist", features, "CRM.attributes" called the adjacency matrix and the label matrix. "CRM.edgelist" in the GCN-Graph model defines the node-to-node network connectivity/relationship in the GCN-Graph model. The "CRM.attributes" contains the list of sales nodes, a node label for performance evaluation, and a pair of labeled nodes, i.e., one node with the label "Won" and another node with the label "Loss" for GCN model training. The feature matrix of each node from the GCN-Graph model. is extracted dynamically.

We have selected two separate nodes with sale status as "Won" and "Loss" and flagged them separately in the dataset for training purposes. To train the GCN model, the GCN-Graph model embeds important features, hence, the node feature matrix is excluded to avoid bias in the prediction. Instead of using a feature matrix corresponding to dataset sales instances, a set of new graph-specific characteristics from the network have been extracted and used as the feature matrix in the GCN-Graph graph model. The graph-specific features are extracted using both the Neo4j graph data science extension and the python NetworkX library. The feature embeddings used for the GCN-Graph model are as follows:

- **Shortest-path** of a node in a network is the shortest path from that node to the specified target node. The shortest path of each node in the network is elicited by assigning the labeled training nodes as targets. The extracted feature matrix is an $N \times 2$ matrix containing the shortest path of each node to two separate target nodes, i.e., nodes with labels "Lost" and "Won".
- **Eigenvector Centrality** (Newman , 2008) of a node is the influence of that node in the graph. The relationship of a node with a high-scoring node is a reason to get a higher eigenvector score than its connection to a low-scoring node in the CRM.

6. GCN-based analytics

This section applies the GCN for classifications of sales based on the GCN-Graph model discussed in Section 4. The GCN is an advanced message-passing neural network popular for graph-based classification (Bruna et al., 2014). GCN model is powerful to perform convolutions on the graph model and aggregates the node information from the neighborhood. The expressive power of GCN in graph representation makes it a powerful tool for graph classification tasks, such as node and edge classifications. The goal of the GCN is to learn a function of signals or features on a graph G = (V, E) that takes a feature matrix and an adjacency node connection matrix as input. GCN-based classification using the GCN-Graph model extracted in Section 4 is given in Algorithm 1 and Algorithm 2.

Given a graph network G(V, E), where V represents nodes connected with edges E. Algorithm 1 extracts feature matrix F corresponding to the graph G with the $F \times N$ dimension, where N denotes the number of nodes and F is the number of features. Line 1 in Algorithm 1 generates an adjacency matrix A of size N denoted as $\hat{A}$. The adjacency matrix contains information on the weighted edges. A_{ij} is set to 1 if an edge exists between i and j, and, otherwise, 0. Line 4 to 5 in Algorithm 1 generates an $N \times L$ binary matrix Y where L represents the number of classes. The generated matrix is a single instance of each label. The labeled matrix corresponding to the test set is used later for performance evaluation.

The adjacency matrix A and feature matrix X are given as input to the forward propagation equation. The adjacency matrix is passed to the forward propagation in Equation 1 after adding a self-loop to include the features of the node along with neighboring node features. Forward propagation mechanism in Equation 1 generates node features as a summation of the features of the neighboring node along with its features. The equation also defines the addition of self-loop with the adjacency matrix using the identity matrix, and is given as follows in Equation 1:

$$\hat{A} = A + 1 \tag{1}$$

Lines 5 to 7 in Algorithm 1 normalize the features using the inverse degree matrix (Chung et al. , 2003), where D^{-1} is the inverse degree matrix and is expressed as given in Equation 2. This normalization ensures that each node's feature contribution is scaled relative to its connectivity, preventing high-degree nodes from dominating the aggregation process during graph convolution.

$$f(X, A) = D^{-1}\hat{A}X \tag{2}$$

Each hidden layer in GCN is defined as the propagation rule as given in Equation 3. In the Equation 3, H^i represents the i^{th} hidden layer and f is the propagation function where $H^0 = X$. The H^i is the propagated i^{th} row in a feature matrix $N \times F^i$ and is based on the spectral rule given Equation 3.

$$H^i = f(H^{i-1}, A) \tag{3}$$

The propagation function is given in Equation 4:

$$f(H^{i-1}, A) = \sigma(D^{-1}(\hat{A})H_i W_i) \tag{4}$$

Here, W^i is the weight matrix for the layer i and σ represents the activation function ReLU. Based on Kipf and Welling's (2017) spectral propagation rule, we can rewrite this equation as Equation 5, given in Lines 6 to 7. Each hidden layer in the GCN performs a localized spectral convolution, where node features are aggregated from their neighbors using the normalized adjacency matrix. Specifically, at each layer h, the model updates node embeddings by applying the propagation rule defined in Equation 5, which combines feature transformation through the weight matrix W^i and a nonlinear activation function, ReLU. The hidden layers progressively learn richer feature representations by capturing multi-hop dependencies in the graph structure, enabling the model to infer complex relational patterns critical for node classification and prediction tasks.

$$f(H^{i-1}, A) = \sigma(D^{-0.5}(\hat{A})D^{-0.5}H_i W_i) \tag{5}$$

Algorithm 1 GCN with Convolution Operation

```
Require: Adjacency matrix $A$, Feature matrix $X$, Label matrix $L$
Ensure: Latent feature representation $F_{latent}$
 1: $H \leftarrow H_L$                                ▷ Number of hidden neural network layers
 2: $\hat{A} \leftarrow A + 1$                        ▷ Generate adjacency matrix $A$ of size $N \times N$
 3: function GCNmodel($\hat{A}$)
 4:     $CON\_Agg \leftarrow \emptyset$
 5:     for $h = 1$ to $H$ do
 6:         for $i = 1$ to number_of_rows($L$) do
 7:             $CON\_Agg$.append($\sigma(D^{-0.5}(\hat{A})D^{-0.5}H_i W_i)$)    ▷ Kipf and Welling's spectral propagation
    rule
 8:         end for
 9:     end for
10:     $F_{latent} \leftarrow$ LogisticReg($CON\_Agg$)        ▷ Return aggregated features
11:     return $F_{latent}$
12: end function
```

Algorithm 2 GCN Model Training and Predictions

```
Require: $crm.edgelist$, $crm.attributes$
Ensure: Sales prediction
1: $network \leftarrow crm.edgelist$
2: $attributes \leftarrow crm.attributes$    ▷ Generate adjacency matrix $A$ of size $N \times N$
3: $X\_train, Y\_train \leftarrow$ attributes['churn']    ▷ Setting labeled nodes for training, i.e., 'Won' or 'Lost'
4: $X\_test, Y\_test \leftarrow$ attributes['churn'] == 'sales'    ▷ Setting all unlabeled nodes for testing
5: $X\_train, X\_test \leftarrow$ flatten($X\_train, X\_test$)    ▷ Partition data into test and train sets
6: train($GCNmodel, X, X\_train, Y\_train$)    ▷ Feed the GCN model training data
7: predict($GCNmodel, X, X\_test$)    ▷ Perform predictions
```

Algorithm 2 outlines the complete pipeline for training and testing the GCN model using the CRM dataset. The process begins by loading the network structure from the edge list and node-level features from the attribute file (lines 1–2). These are used to construct the graph and feature matrix required by the GCN. In line 3, the dataset is split into training data, consisting of labeled nodes that represent known sales outcomes (e.g., "Won" or "Lost"). Line 4 identifies test data: unlabeled nodes that need prediction. Both training and test features are then flattened (line 5) to match the input shape expected by the model. Line 6 trains the GCN model using the labeled node features and their corresponding churn labels. The model learns by optimizing a cross-entropy loss function through backpropagation, updating the weights to minimize classification error. The learned features (F_{latent}) are passed through layer-wise spectral propagation, where each GCN layer aggregates neighborhood information using the ReLU activation function.

Finally, in line 7, the trained GCN model is used to predict the labels of unseen nodes in the test set. The predictions are made using a logistic regression classifier applied to the final node embeddings, as shown in line 10 of Algorithm 1. This enables binary classification of sales outcomes (for example, predicting whether a sales attempt is likely to be successful).

7. Performance evaluations

This section evaluates the performance of GCN and compares it with CNN, FNN (LeCun et al. , 2015), an ensemble model, i.e., RF (Zhu et al. , 2017), and boosting classifiers, i.e., XGBoost and CATBoost. For performance evaluations, we performed hyperparameter tuning for each model using grid search and cross-validation to identify the best performing configurations. For the GCN model, hyperparameters such as the number of layers (ranging from 2 to 4 layers), number of hidden units per layer (32, 64, or 128 units), learning rate (0.001, 0.01, 0.1), dropout rate (0.2, 0.5), and activation functions (ReLU, Leaky ReLU) were tuned. Similarly, for the CNN model, we optimized the number of convolutional layers (2 to 4 layers), number of filters (32, 64, 128 filters per layer), kernel size (3x3, 5x5), learning rate (0.001, 0.01), and batch size (32, 64). For the FNN model, we tested different architectures with layer depths (2 to 4 hidden layers), units per layer (32, 64, 128 units), learning rate (0.001, 0.01), and activation functions (ReLU, Sigmoid). The RF model's hyperparameters, such as the number of trees (50, 100, 200 trees), maximum depth (10, 20, 30), and minimum samples split (2, 5, 10) were tuned. For XGBoost and CATBoost, learning rates (0.001, 0.01, 0.1), number of estimators (50, 100, 200), and tree depth (3, 6, 10) were optimized. For performance evaluations, we train the FNN after applying one-hot encoding on the training dataset and CNN with 2D convolution operation using 95 training features. Deep learning and machine learning classifiers are implemented and tested on an NVIDIA GeForce GTX-1050 Ti graphic processing unit (GPU). The GPU unit consists of 766 NVIDIA CUDA cores, 4 GB shared memory (VRAM), and a 1392 MHz clock speed. The implementation has used Python-3.6 and Cypher (Francis et al., 2018) for data analysis, model development, and data mining. Cypher by Neo4j is a declarative graph query language developed for efficient and expressive graph data processing, i.e., graph model creation and graph feature extraction. The exploratory analysis considers the use of Python and Cypher. GCN model is implemented and tested using Keras and Apache MXNet to support distributed learning. Other libraries include NetworkX, SKlearn, TensorFlow, py2neo, and APOC.

7.1. EDA graph model-based exploratory analysis

This section performs exploratory analysis on the graph models presented in Section Section 4. This section performs exploratory analysis using Cypher-based data mining on the EDA-Graph model that abstracts the B2B sales data as given in Figure 1. The GCN-Graph model represents the interconnectivity between the sales nodes identified using the "sales_enquiry_id". Connectivity between the sales nodes is defined using five features, i.e., Up_sale, Client, Competitors, Product, and seller.

EDA-Graph model-based inventory statistics are given in Table 2. The model consists of 1386 nodes with 7 labels. The total number of relationships/edges among nodes is 3136, with 7 relationship types. Among the 7 assigned labels, 5 are unique labels created using Cypher query "unique constraint". In the table, unique labels merge the same labeled nodes with similar values during the data import procedure. The database size is 2.03MB, larger than the raw dataset of size 67 KB. The sales label consists of 448 unique sales_enquiry_id_nodes, whereas the product label defines 14 unique products, sold by the company. There are 18 sellers represented using 18 distinct seller nodes in the graph model. The sales inquiry is integrated from eight different sources in the graph model as there are eight individual nodes with the sale status label with unique values of "Won" and "Lost".

Table 2. EDA-Graph model inventory statistics.

Parameter Name	Value	Unique Labels	Count
Number of nodes	1386	Sale	448
Number of labels	7	Product	14
No. of relationships	3136	Seller	18
No. of unique labels	5	Source	8
No. of relationship types	7	Sales Status	2
Size	2.03 MB	–	–

The inventory statistics of the GCN-Graph model are summarized in Table 3 and mined using the metadata and schema with the help of Cypher query. GCN-Graph model consists of a single sales label with 367 sale ID nodes. There are 423 relationships between sale nodes with only one type. The database is of size 1.53 MB, i.e., less than the size of the EDA-Graph model. This is due to less number of labels, nodes, and relationships. The average relationship count of the sales label is 2.305 per node, whereas the maximum relationship count is 33 (dense) with a minimum value of 1 (sparse), i.e., a graph with a maximum of 33 nodes and a minimum of 1 node.

Table 3. GCN-Graph model inventory statistics.

Parameter Name	**Value**
No. of nodes	367
No. of labels	1
No. of relationships	423
Min. relationship count	1.00
Max. relationship count	33.00
Avg. relationship count	2.305
Relationship types	1
Store size	1.53 MB
Name of label	Sales
Name of relationship	IS_CONNECTED

7.2. Exploratory data analysis

This section exploits the interactive capabilities of Neo4j Cypher query for exploratory analysis. As all the features in the CRM dataset are categorical, therefore, quantitative analysis based on statistical measures is not applicable here. Further, analysis in this section focuses only on critical features with particular attention to the sales status count. The dataset consists of 448 samples/training examples of sales_enquiry_id with the same number of "Loss" and "Won" sale outcomes. Both the EDA-Graph model and GCN-Graph model share the same characteristic. Here, the count of sale status class is evenly distributed. The sales status count distribution of sale ID is listed in Table 4. Some sales nodes are removed during the GCN-Graph model construction to avoid the formation of a separate graph network that has no relationship with the main graph network. In the generated graph model, the proportion of sales instances with positive and negative sales outcomes are approximately the same, as is evident from Figure 6.

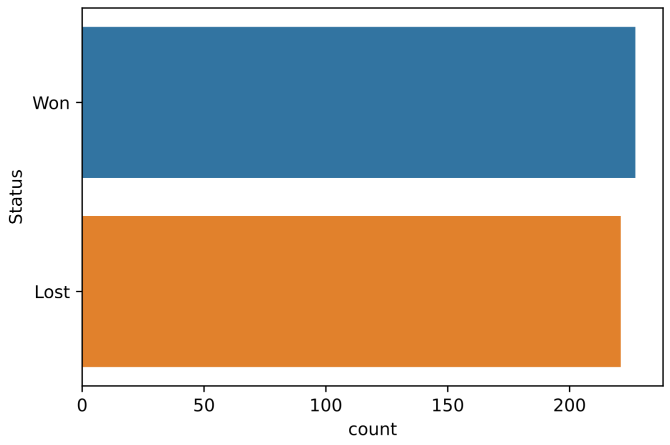
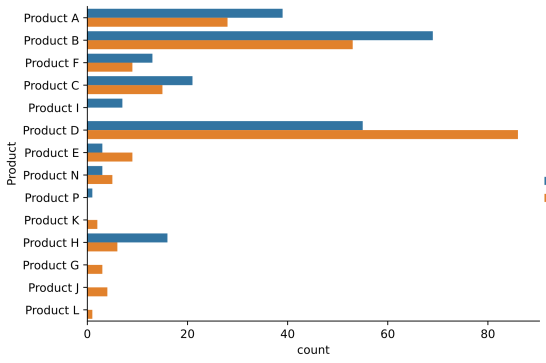
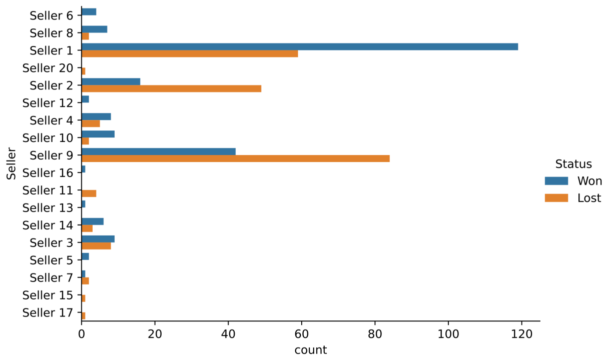


Figure 6. Class distributions in dataset.



(a) Product count based on sales status.



(b) Seller count based on sales status.

Figure 7. Product and seller with sales status.

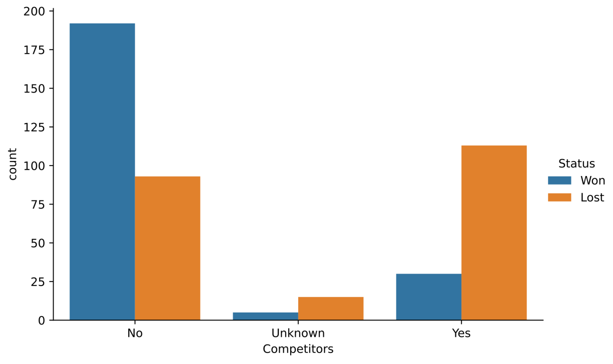
The EDA-Graph model consists of 227 successful sales attempts, with 221 sale instances with negative sale outcomes. On the other hand, the GCN-Graph model has a total of 367 sale nodes, with 177 sale instances/training samples with positive sale status and 190 instances with a negative outcome.

Table 4. Frequency of sales ID based on sales outcome.

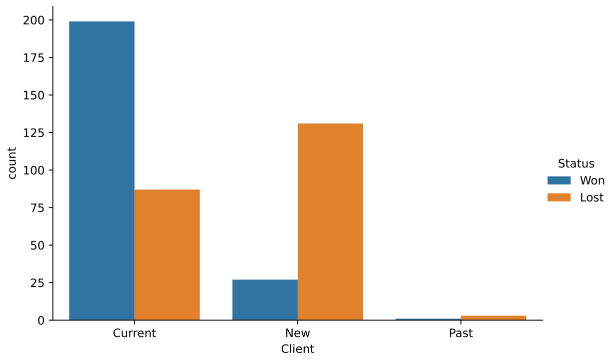
Sales Status	EDA-Graph Count	GCN-Graph Count
Won	227	177
Loss	221	190
Total	448	367

Figure 7 (a) shows the count of the seller and the product features with the sales status. It is clear from the figure that Product B and Product D are the two most successful products with a higher sales rate. Product D receives the highest number of requests. Product B still achieves a higher sales success count. In the figure, the positive sales status of product D is less than the negative sales status. Based on the selected dataset and use case, it is evident that the successful sales count of Product B is just below 70 and the number of failed sales attempts is below 60. Products L, J, G, and K have very few inquiries and no successful sales results. From the seller vs. sales status in Figure 7 (b), we can see that Seller 1 is the most successful salesman in charge. The number of positive sales performed by Seller 1 is just below 120 with less than 60 failed sales outcomes. Seller 2 and Seller 9 are the two other sellers with comparatively better positive sales outcomes than the remaining 15 sellers. The product and seller features are use-case specific and do not generalize to other B2B sales attempts. Although not considered in this work, it is also important to consider other external factors like innovation, geographic features, and market needs while making seller and product sales assumptions.

Figures 8 (a) and Figure 8 (b) show the features that generalize well in the B2B sales process. These features are important for CRM with a significant effect on sales outcomes. These features are transferable with an ability to generalize in similar business use cases. It can be seen in Figure 8 (a) that the success rate of a company with no competitors is higher than that of others. A similar trend is observed in Figure 8 (b) for the clients,that is, for current or existing clients, the sales success rate

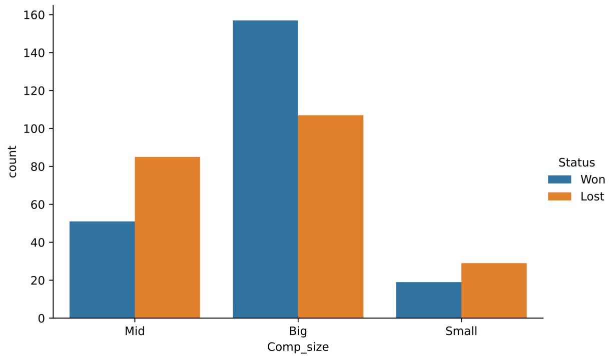


(a) Competitors with sales status.

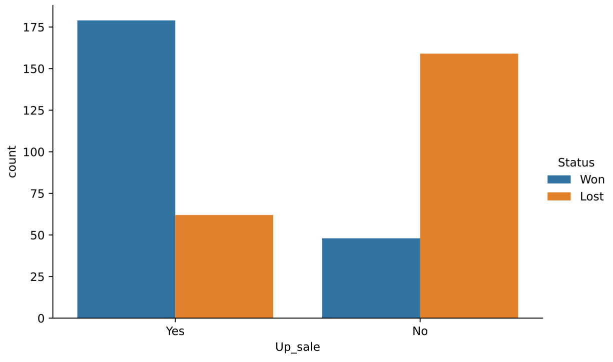


(b) Clients with sales status.

Figure 8. Competitors and clients with sales status.



(a) Company size and sales status.



(b) Up_sales and sales status.

Figure 9. Company size and up_sales status.

is twice the count of negative sales outcomes. In this scenario, the sales success count is positively impacted by the potential number of clients. For potential clients, the number of positive sales results is greater than the number of failed sales attempts. It is evident from Figure 9 (a) and Figure 10 that the size of the company results in increased sales in contrast to the small and medium-sized companies. From the analysis of the "UpScale" feature from Figure 9 (b), it is clear that all clients in need of "UpScale" products increase the sales success count. To conclude as given in Figure 11, it is concluded that sales attempts with no competitors, big company size, existing clients, good budget allocations,with good partnerships, and clients with upscale requirements positively impact the sales success rate.

7.3. Spectral GCN model evaluation and analysis

In this section, we compare the performance of the GCN models with RF, CNN, and FNN using several evaluation metrics, including accuracy, precision, sensitivity, specificity, F1 score, and the area under the ROC curve (AUC). Precision measures the proportion of correctly predicted positive sales outcomes among all instances predicted as positive. Sensitivity, also known as recall, indicates the percentage of actual positive sales outcomes that are correctly identified. Specificity represents the proportion of correctly classified high-priority negative sales outcomes relative to all actual negative

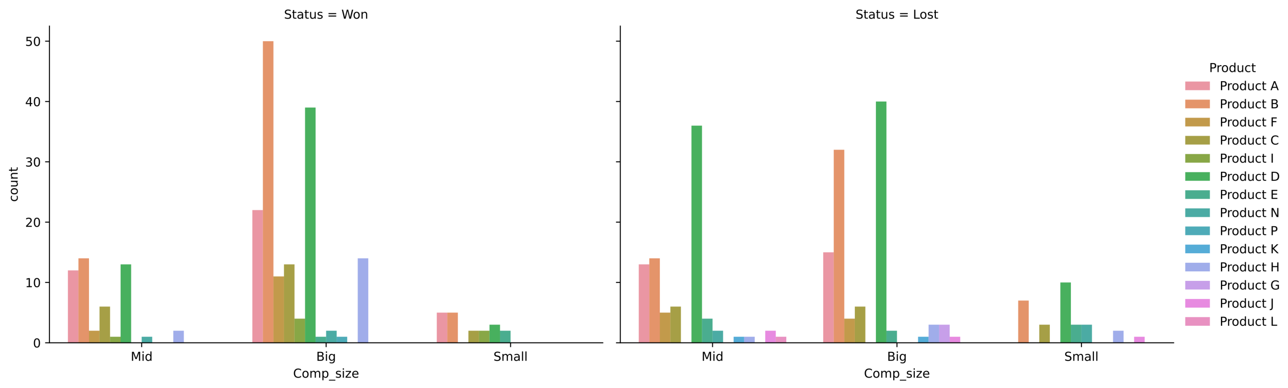


Figure 10. Company size and products with sales status.

predictions. The F1 score, calculated as the harmonic mean of precision and recall, provides a balanced metric that captures the trade-off between these two aspects of performance.

In addition to the comparisons presented above, we further evaluate the performance of GCNs using graph-level features introduced in Section 5, namely the shortest-path distances and eigenvectors. For ease of reference, we denote the GCN model utilizing shortest-path features as GCN-1, and the model leveraging eigenvectors as GCN-2. Moreover, we extend our analysis by comparing these GCN variants against traditional boosting-based classifiers, specifically XGBoost and CATboost.

Figure 12 presents the confusion matrices for all models. In the confusion matrix, the negative label represents the failed sales outcome as “0” and the positive class as “1” for successful sales. In the confusion matrices, the dark blue color indicates true positives (TPs) and true negatives (TNs), whereas the white color shows false positives (FPs) and false negatives (FNs). It is clear from Figures 12 (a), Figure 12 (b), and Figure 12 (f) that GCN-1 with the shortest-path feature has zero “FPs” and 0.14 “FNs” compared to the boosting machine classifiers, i.e., XGboost and CATboost, with CATboost 0.16 “FPs” and 0.2 “FN”, and XGboost 0.18 “FPs” and “0.17” FNs. Similarly for the selected ensemble model, RF in Figure 12 (c), misclassification is considerably lower for GCN-1. In addition, GCN-1 outperforms CNN and FNN (Figure 12 (d)) with fewer misclassification. This performance difference is primarily due to GCN’s ability to incorporate neighborhood information and relational dependencies between nodes in the graph structure—capabilities that are lacking in traditional models like CNN, FNN, and tree-based methods. Figures 12 (f) and Figure 12 (g) show that GCN with shortest-path graph feature performs better than Eigenvectors graph feature with only 0.14 “FNs” in contrast to 0.26 “FNs”. The shortest-path feature allows the model to effectively capture proximity-based influence in the graph, enhancing the prediction accuracy. This explains that the shortest path between nodes in a graph leads to better classification accuracy than the Eigenvectors as representative graph features.

Figure 13 (a) compares the performance of GCN-1 and GCN-2 with FNN and CNN with varying numbers of epochs from 10 to 60. It is clear from the figure that the GCN-1 with the shortest path outperforms all other deep learning models in terms of loss and accuracy. GCN-1 has the highest accuracy among all and converges after 20 epochs. This improved performance can be attributed to GCN-1’s ability to leverage the graph structure and shortest-path features, which provide rich relational information that is particularly valuable for structured data like sales interactions. GCN-2 with the Eigenvectors also performs better than other deep learning models; however, it takes longer for the

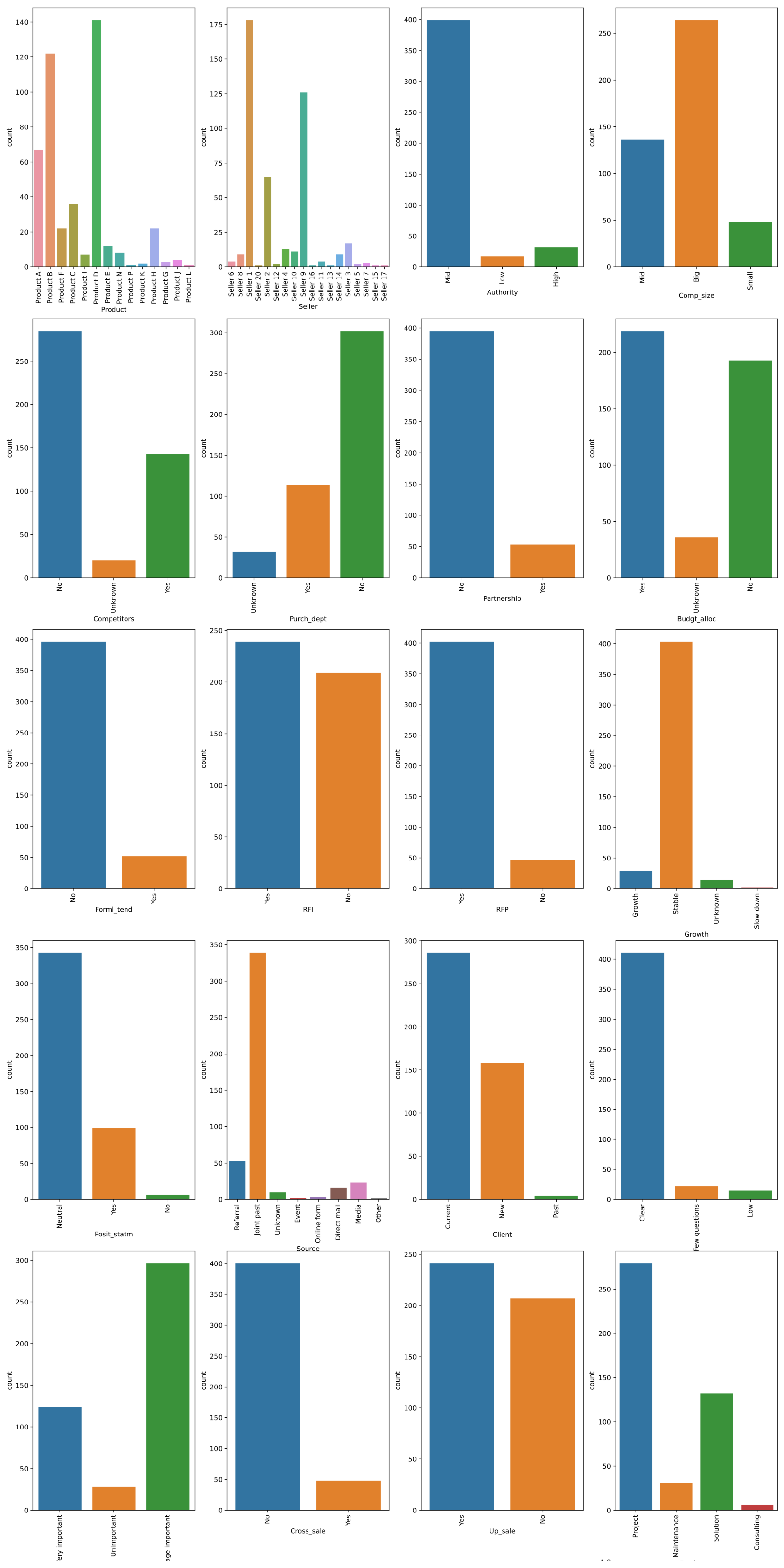
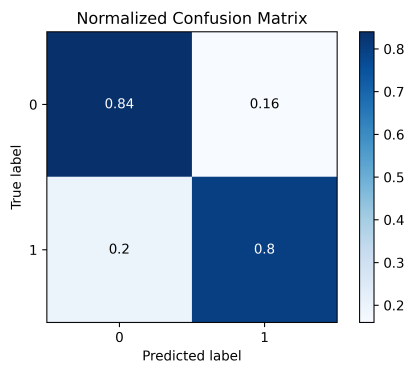
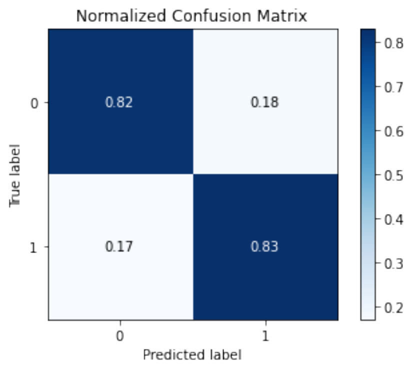


Figure 11. Exploratory analysis based on EDA-Graph model.

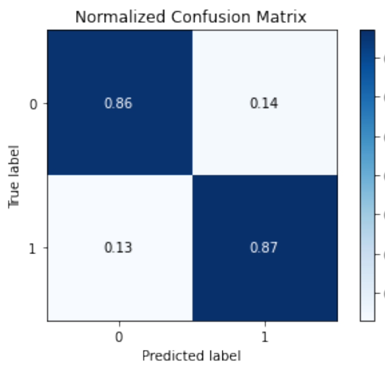
model to learn the right weights and converge in contrast to CNN, ensemble and gradient-boosting classifiers. In comparison, CNN is optimized for grid-like data (e.g., images) and does not capture the underlying graph relationships inherent in the dataset, leading to suboptimal feature extraction. Ensemble and gradient-boosting classifiers like RF and XGboost, while powerful for tabular data, fail to model inter-instance relationships, which limits their ability to generalize complex dependencies in graph-based sales data. A similar trend for the loss is observed for the GCN-2, as is evident in Figure 13 (b), where it is initially higher and takes longer to converge in contrast to other models. Figures 14(a), 14(b), 15(a), and 15(b) present a comparative analysis of GCN-1 and GCN-2 against RF, CNN, FNN, XGBoost, and CATboost across several performance metrics, including accuracy, precision, specificity, sensitivity, and F1 score. As illustrated, GCN-1 outperforms all other models in terms of accuracy, precision, specificity, and F1 score. Specifically, Figure 15 (b) highlights that GCN-2 exhibits a lower sensitivity (0.669) compared to GCN-1, FNN, CNN, RF, and the boosting-based methods. In contrast, GCN-1 demonstrates a better balance between metrics, with a high specificity and a sensitivity value of 0.857. In general, both GCN models surpass RF, FNN, and CNN in terms of precision, sensitivity, specificity, and F1 score. In particular, GCN-1 achieves perfect accuracy (1.0), outperforming all competing models. RF and GCN-2, by comparison, reach accuracies of 0.86 and 0.85, respectively. Interestingly, while GCN-2 lags in performance, FNN achieves a higher F1 score (0.85), suggesting better generalization compared to GCN-2 when using Eigenvector-based features as given in Figure 16. Furthermore, results shown in Figure 15(b) reveal that both CNN and FNN attain a sensitivity of 0.847, which is notably higher than that of GCN-2. However, CNN and FNN are inherently limited in modeling graph-structured relationships, which reduces their ability to capture context between sales instances. RF, while robust for tabular data, does not leverage any relational or topological information, leading to relatively lower performance when such structure exists in the data. Table 5 summarizes the number of correct and incorrect sales outcome predictions for all models. The true positive rate for all models, except GCN-2, exceeds 80%, indicating a generally strong ability to identify positive sales outcomes. Both GCN models achieve the highest true negative rates, demonstrating superior performance in correctly classifying unsuccessful sales attempts. Among all models, GCN-1 performs the best overall, with a perfect true negative rate of 1.00 and a true positive rate of 0.86. Although the true positive rate of GCN-1 is slightly lower than that of the RF model, RF achieves the highest success rate in predicting positive sales outcomes. However, both GCN-1 and GCN-2 outperform the remaining models in overall sales outcome classification performance, highlighting their effectiveness in handling both positive and negative cases.



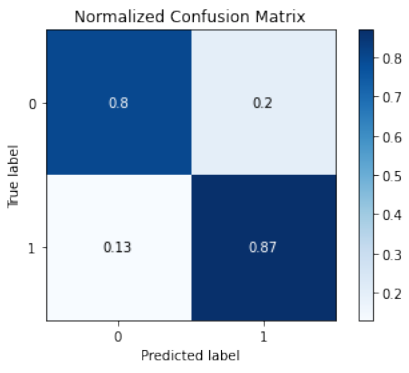
(a) CATboost.



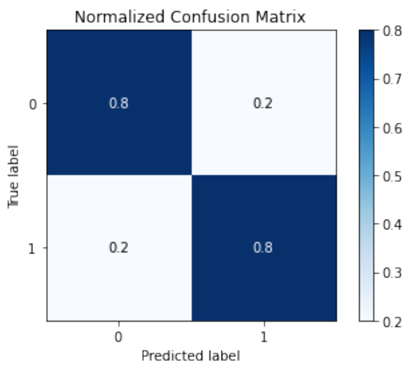
(b) XGboost.



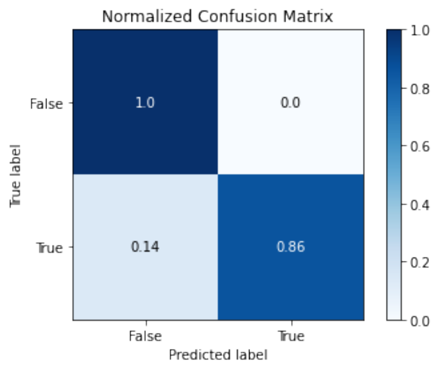
(c) RF.



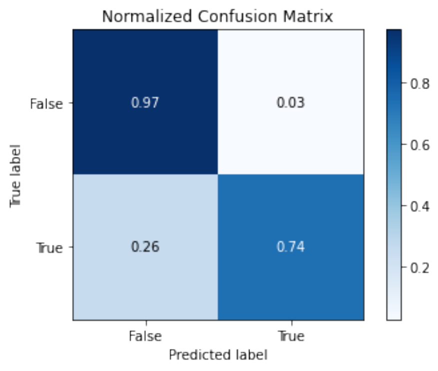
(d) FNN.



(e) CNN.

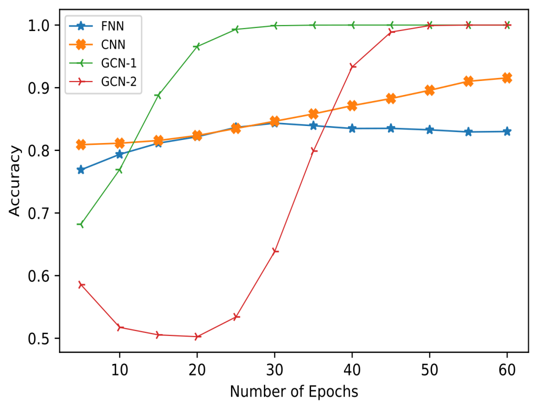


(f) GCN-1.

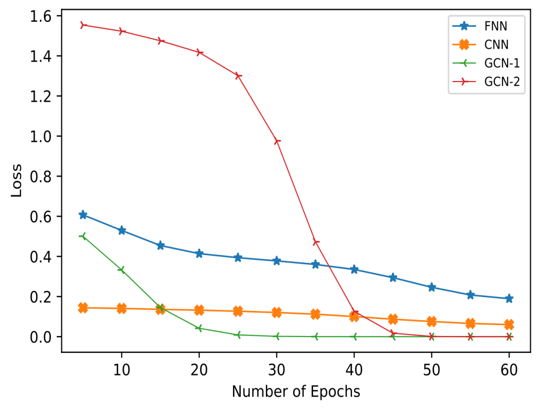


(g) GCN-2.

Figure 12. Confusion matrices comparison of GCN-1 and GCN-2 with baseline models.

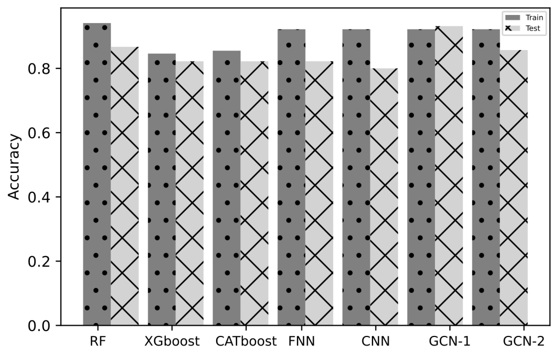


(a) Accuracy with varying number of epochs.

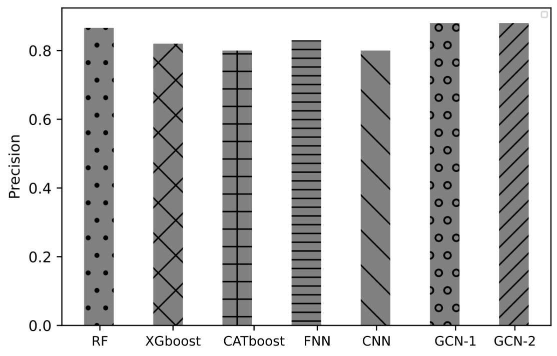


(b) Loss with varying number of epochs.

Figure 13. Comparison of GCN-1 and GCN-2 with deep learning models.



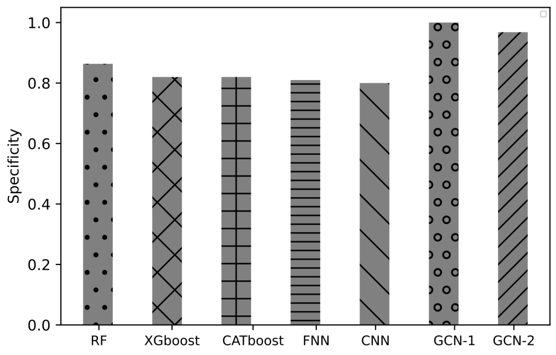
(a) Accuracy comparison of GCN-1 and GCN-2.



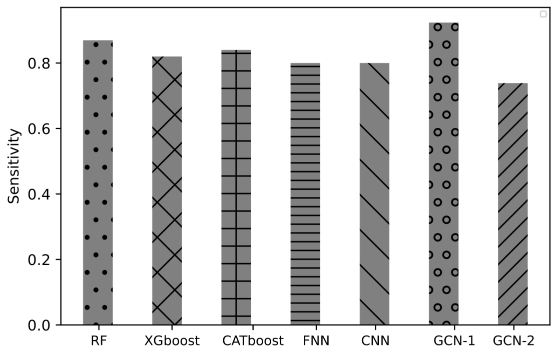
(b) Precision comparison of GCN-1 and GCN-2.

Figure 14. Accuracy and precision comparison of GCN-1 and GCN-2 with ensemble, gradient boosting, and deep learning models.

Figure 17 (a) shows the ROC curve of all the models. The figure shows that the GCN-1 model achieves the highest AUC value of 0.929. It can be observed from the figure that the GCN-2 underperforms RF and artificial neural network (ANN) with an AUC value of 0.853. These results are obtained from a balanced dataset for training and testing. Figure 17 (b) presents that the GCN-1 model achieves better trade-off between precision and F1 score with an area of 0.963. On the other hand the area of the GCN-2 is comparable with other deep learning models, gradient boosting, and ensemble models. In our selected use-case relevant to the CRM, prediction of both true positive and true negative is equally important to devise sales and marketing strategies. Here, the GCN-1 model demonstrates the best performance in predicting all true and negative sales outcomes. Based on GCN model evaluations and comparison with other models, it is clear that the GCN-1 with the shortest distance is the best-fit model for B2B sales outcome prediction The better performance of the GCN-1 model is attributed to the node-to-node relationship among sales nodes. We used labeled nodes as a target to calculate the



(a) Specificity comparison of GCN-1 and GCN-2.



(b) Sensitivity comparison of GCN-1 and GCN-2.

Figure 15. Specificity and sensitivity comparison of GCN-1 and GCN-2 with ensemble, gradient boosting, and deep learning models.

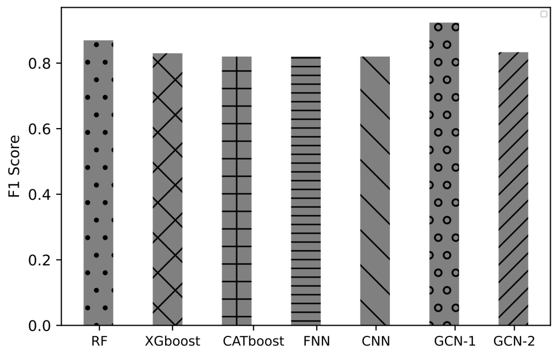
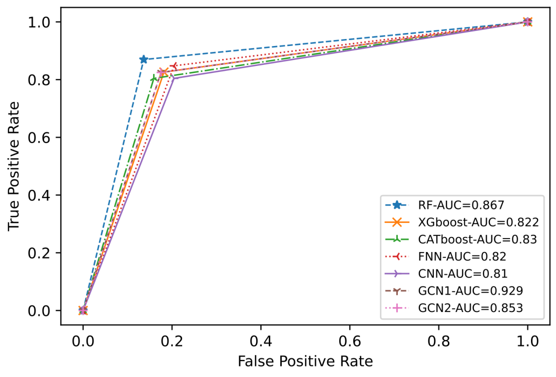


Figure 16. F1 score comparison of GCN-1 and GCN-2 with ensemble, gradient boosting, and deep learning models.

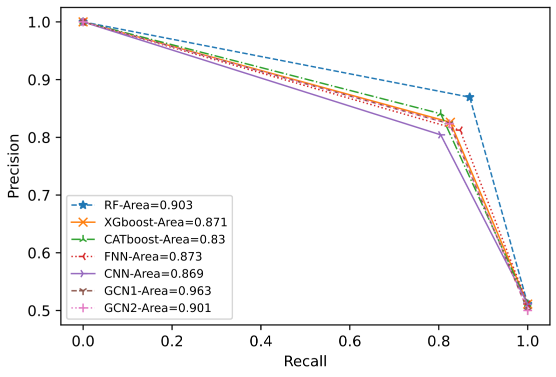
shortest path distance of each node. It means that the shortest-path graph feature has a stronger impact on node classification. The Eigenvector of a node allows the high-scoring nodes to contribute more toward the score of the target node, compared to the equal connections to low-scoring nodes. Node-wise relationship between sales nodes is an influencing factor in terms of its community/cluster behavior and contributes better to predictions. Experimental results reveal that GCN-1 learns by performing convolutions on a set of nodes to extract meaningful features/attributes from neighboring nodes. Here, the extracted features, like the shortest path and Eigenvectors, are graph-level features. In terms of convolutional learning, the GCN model can elicit insights from neighboring nodes. Using GCN, it is possible to process indirect features like shortest-path, Eigenvector, and other hidden graph features. These graph features show a significant impact on the classification performance in addition to the dataset features.

Table 5. Classification performance of all models.

Model	True Negative	False Positive	False Negative	True Positive
Random Forest	0.86	0.14	0.13	0.87
FNN	0.86	0.14	0.15	0.85
CNN	0.77	0.23	0.15	0.85
GCN-1	1.00	0.00	0.14	0.86
GCN-2	0.97	0.03	0.26	0.74
XGBoost	0.82	0.18	0.17	0.83
CATboost	0.84	0.16	0.20	0.80



(a) AUC comparison of GCN-1 and GCN-2.



(b) Recall vs Precision of GCN-1 and GCN-2.

Figure 17. AUC and Recall vs. Precision of GCN-1 and GCN-2 with ensemble, gradient boosting, and deep learning models.

The results clearly indicate that the GCN-1 model, which incorporates the shortest-path feature, significantly outperforms all other models, including RF, CNN, FNN, XGBoost, and CATboost—across multiple evaluation metrics such as accuracy, precision, sensitivity, specificity, F1 score, and AUC. In particular, GCN-1 achieves perfect accuracy (1.00), with zero false positives and a remarkably low FN rate of 0.14. This performance suggests that incorporating shortest-path information effectively captures the strength of node relationships within the graph, which is particularly important for B2B sales outcome prediction. In contrast, GCN-2, which uses eigenvector-based features, exhibits slightly lower performance, especially in terms of sensitivity and F1 score. The shorter convergence time and consistently higher performance of GCN-1 further emphasize the advantage of using graph features based on shortest-path. Moreover, both GCN models demonstrate strong capabilities in accurately predicting both TPs and TNs, an essential requirement for CRM applications in sales and marketing strategy. In general, the findings strongly support that GCN-1, leveraging shortest-path features, is the most effective model to predict B2B sales outcomes, outperforming both traditional machine learning and deep learning approaches.

8. Conclusions and future work

This study presents a novel application of graph-based deep learning, specifically spectral GCNs for predicting and analyzing customer relationships in B2B environments. Traditional CRM approaches, which typically rely on classical machine learning and deep learning models, often fail to capture the intricate, dynamic interdependencies inherent in B2B contexts. To address this gap, we propose a graph-centric methodology that models the complex relational structures among companies using two graph frameworks: EDA-Graph and GCN-Graph. These models were constructed from real-world sales datasets using the Cypher query language within the Neo4j graph database.

Our approach involved a two-phase process: (1) EDA to identify key features affecting B2B sales outcomes, and (2) the deployment of a spectral GCN on the GCN-Graph model for sales outcome classification. We further enhanced the GCN models by incorporating graph-level features, namely, shortest-path and Eigenvector centrality, which are particularly effective in spectral GCN operations. These enhanced models, referred to as GCN-1 and GCN-2, were evaluated against several state-of-the-art baseline models, including RF, CNN, XGBoost, and CATboost.

Experimental results demonstrate that the proposed GCN models, particularly GCN-1, consistently outperformed traditional and deep learning models across multiple performance metrics, including accuracy, F1 score, precision, specificity, and AUC. GCN-1 achieved a highest accuracy score and a true negative rate of 1.00, highlighting its exceptional generalizability. Meanwhile, GCN-2, although slightly lower in overall accuracy, showed superior performance in precision and specificity, making it valuable in contexts where identifying negative sales outcomes is critical.

Comparative analysis also revealed that GCN models outperform gradient boosting classifiers in identifying true negatives and detecting subtle relational patterns that conventional models often overlook. In particular, the performance of the GCN models remained robust across different feature subsets, indicating their resilience to feature redundancy and the high-dimensional nature of CRM data. In addition, we observed that the quality of features, particularly their relational significance, plays a more critical role than their quantity in influencing model performance.

Author contributions

Shgufta Henna (first author) contributed to the research, leading the overall study conception, methodology development, model implementation, and analysis. Shyam Krishnan Kalliadan contributed to the experimentation for the paper. Mohamed Amjath primarily contributed to the literature review and the overall structure of the paper.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare that there is no conflict of interest.

References

Aasman J (2017) Transmuting information to knowledge with an enterprise knowledge graph. *IT Professional* 19: 44–51. https://doi.org/10.1109/MITP.2017.4241469

Alomrani M, Biparva M, Zhang Y, et al. (2024) DyG2Vec: Efficient Representation Learning for Dynamic Graphs. *Trans Mach Learn Res*. https://doi.org/10.48550/arXiv.2210.16906

Asniar and Surendro K (2019) Predictive analytics for predicting customer behavior. 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT), Yogyakarta, Indonesia, 230–233. https://doi.org/10.1109/ICAIIT.2019.8834571

Brahim AB, Raboudi W (2020) Improving customer relationship management using machine learning techniques: A Tunisian case study. 2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA), Tunis, Tunisia, 1–16. https://doi.org/10.1109/OCTA49274.2020.9151465

Bruna J, Zaremba W, Szlam A, et al. (2014) Spectral networks and locally connected networks on graphs. *Int Conf Learn Representations*. https://doi.org/10.48550/arXiv.1312.6203

B2B Dataset (2020) A real-world dataset for B2B sales analysis. Salvirt Research. Available from: `http://www.salvirt.com/research/b2bdataset`.

Cai H, Zheng VW, Chang KC (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans Knowl Data Eng* 30: 1616–1637. https://doi.org/10.1109/TKDE.2018.2807452

Chaudhari S, Mithal V, Polatkan G, et al. (2021) An attentive survey of attention models. *ACM Trans Intell Syst Technol* 12: 1–32. https://doi.org/10.1145/3465055

Chung F, Lu L, Vu V (2003) Spectra of random graphs with given expected degrees. *Proceedings of the National Acad of Scien* 100: 6313–6318. https://doi.org/10.1073/pnas.0937490100

Das A, Mitra A, Bhagat SN, et al. (2020) Issues and concepts of graph database and a comparative analysis on list of graph database tools. 2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 1–6. https://doi.org/10.1109/ICCCI48352.2020.9104202

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. *Adv Neural Inf Process Sys*. `https://arxiv.org/abs/1606.09375`

Duvenaud DK, Maclaurin D, Iparraguirre J, et al. (2015) Convolutional networks on graphs for learning molecular fingerprints. *Adv Neural Inf Process Syst* 28. https://doi.org/10.48550/arXiv.1509.09292

Francis N, Green A, Guagliardo P, et al. (2018) Cypher: An evolving query language for property graphs. *Proceedings of the 2018 Int Conf on Manage of Data*, 1433–1445. https://doi.org/10.1145/3183713.3190657

Gheisari M, Wang G, Bhuiyan MZA (2017) A survey on deep learning in big data. *IEEE Int Conf Comput Sci Eng and IEEE Int Conf Embedded Ubiquitous Comput*: 173–180. https://doi.org/10.1109/CSE-EUC.2017.215

Global Customer Relationship Management (2020) Growth, Trends and Forecast to 2025. BusinessWire. Available from: `https://www.businesswire.com/news/home/20200416005517/en/Global-Customer-Relationship-Management-CRM-Market-Growth`.

Hamilton WL, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. *Adv Neural Inf Process Syst* 30. https://doi.org/10.48550/arXiv.1706.02216

Hoksza D, Jelínek J (2015) Using Neo4j for mining protein graphs: A case study. *26th Int Workshop Database Expert Syst Appl*, 230–234. https://doi.org/10.1109/DEXA.2015.59

Huang H, Dong Z (2013) Research on architecture and query performance based on distributed graph database Neo4j. *3rd Int Conf Consum Electron Commun Netw*, 533–536. https://doi.org/10.1109/CECNet.2013.6703387

Ibrahim A, Ermatita, Saparudin, et al. (2017) Analysis of weakness of data validation from social CRM. *Int Conf Data Softw Eng (ICoDSE)*, 1–5. https://doi.org/10.1109/ICODSE.2017.8285849

Kim TS, Lee WK, Sohn SY (2019) Graph convolutional network approach applied to predict hourly bike-sharing demands considering spatial, temporal, and global effects. *PLOS ONE* 14: e0220782. https://doi.org/10.1371/journal.pone.0220782

Kimura D, Gotoh T, Ikeda K (2011) Eliciting considerable requirements with word and customer graphs. *IEEE 35th Annu Comput Softw Appl Conf*, 476–485. https://doi.org/10.1109/COMPSAC.2011.68

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. *Int Conf Learn Representations (ICLR)*. https://doi.org/10.48550/arXiv.1609.02907

Kitchin R (2014) The data revolution: Big data, open data, data infrastructures & their consequences. *SAGE Publications Ltd*. https://doi.org/10.4135/9781473909472

Konno T, Huang R, Ban T, et al. (2017) Goods recommendation based on retail knowledge in a Neo4j graph database combined with an inference mechanism implemented in Jess. *IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, 1–8. https://doi.org/10.1109/UIC-ATC.2017.8397433

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521: 436–444. https://doi.org/10.1038/nature14539

Martínez A, Nin J, Tomás E, et al. (2019) Graph convolutional networks on customer/supplier graph data to improve default prediction. *Complex Netw and Their Appl VII*, 135–146. https://doi.org/10.1007/978-3-030-14459-3_11

Masci J, Boscaini D, Bronstein MM, et al. (2015) Geodesic convolutional neural networks on Riemannian manifolds. *IEEE Int Conf Comput Vis Workshop*: 832–840. https://doi.org/10.1109/ICCVW.2015.112

McClanahan B, Gokhale SS (2016) Centrality and cluster analysis of Yelp mutual customer business graph. *IEEE 40th Annu Comput Softw Appl Conf*, 592–601. https://doi.org/10.1109/COMPSAC.2016.79

Micheli A (2009) Neural network for graphs: A contextual constructive approach. *IEEE Trans Neural Networks* 20: 498–511. https://doi.org/10.1109/TNN.2008.2010350

Monti F, Boscaini D, Masci J, et al. (2017) Geometric deep learning on graphs and manifolds using mixture model CNNs. *2017 IEEE Conf Comput Vision Pattern Recognit*, 5425–5434. https://doi.org/10.1109/CVPR.2017.576

Needham M, Hodler AE (2019) *Graph Algorithms: Practical Examples in Apache Spark and Neo4j.* O'Reilly Media. ISBN: 9781492047681.

Newman MEJ (2008) Mathematics of networks. *In: Durlauf SN, Blume LE (eds) The New Palgrave Dictionary of Economics,* 1–8. https://doi.org/10.1057/978-1-349-95121-5_2565-1

Nugmanova A, Chernykh I, Bulusheva A, et al. (2019) Unsupervised training of automatic dialogue systems for customer support. *Int Conf Qual Manag Transp Inf Secur Inf Technol*, 436–438. https://doi.org/10.1109/ITQMIS.2019.8928445

Robinson I, Webber J, Eifrem E(2020) **Graph Databases*, 2nd Edition.* Available from: `https://www.oreilly.com/library/view/graph-databases-2nd/9781491930885/`.

Saha L, Sahoo L (2018) Adaptation of spectral clustering in telecommunication industry for customer relationship management. *2nd Int Conf Electron Mater Eng Nano-Technol*, 1–6. https://doi.org/10.1109/IEMENTECH.2018.8465187

Scarselli F, Gori M, Tsoi AC, et al. (2009) The graph neural network model. *IEEE Trans Neural Networks* 20: 61–80. https://doi.org/10.1109/TNN.2008.2005605

Taleb N, Salahat M, Ali L (2020) Impacts of big-data technologies in enhancing CRM performance. *2020 Int Conf Inf Manag*, 257–263. https://doi.org/10.1109/ICIM49319.2020.244708

Veličković P, Cucurull G, Casanova A, et al. (2018) Graph attention networks. *Int Conf Learn Representations*. https://doi.org/10.48550/arXiv.1710.10903

Wang L, Liu S, Pan L, et al. (2014) Enterprise relationship network: Build foundation for social business. *IEEE Int Congr Big Data*, 347–354. https://doi.org/10.1109/BigData.Congress.2014.57

Yang J, Liu C, Teng M, et al. (2018) A unified view of social and temporal modeling for B2B marketing campaign recommendation. *IEEE Trans Knowl Data Eng* 30: 810–823. https://doi.org/10.1109/TKDE.2017.2783926

Ye Q, Wang C, Wu B, et al. (2008) VisCRM: A social network visual analytic tool to enhance customer relationship management. *IEEE Int Conf Serv Oper Logistics Informatics*: 825–830. https://doi.org/10.1109/SOLI.2008.4686513

Yu A, Dohan D, Le Q, et al. (2018) [Qanet: Combining local convolution with global self-attention for reading comprehension]. *Int Conf Learn Representations*. https://doi.org/10.48550/arXiv.1804.09541

Zhang B, Xu D, Zhang H, et al. (2019) STCS lexicon: spectral-clustering-based topic-specific Chinese sentiment lexicon construction for social networks. *IEEE Trans Comput Soc Syst* 6: 1180–1189. https://doi.org/10.1109/TCSS.2019.2941344

Zhang Z, Cui P, Zhu W (2022) Deep learning on graphs: A survey. *IEEE Trans Knowl Data Eng*, 249–270. https://doi.org/10.1109/TKDE.2020.2981333

Zhao J, Hong Z, Shi M (2019) Analysis of disease data based on Neo4j graph database. *IEEE/ACIS 18th Int Conf Comput Inf Sci*, 381–384. https://doi.org/10.1109/ICIS46139.2019.8940247

Zhu J, San-Segundo R, Pardo JM (2017) Feature extraction for robust physical activity recognition. *Human-Centric Compt and Inf Scien* 7: 16. https://doi.org/10.1186/s13673-017-0097-2

Zhuang C, Ma Q (2018) Dual graph convolutional networks for graph-based semi-supervised classification. *Proc 2018 World Wide Web Conf*, 499–508. https://doi.org/10.1145/3178876.3186116

Zou Y, Liu Y (2020) The implementation knowledge graph of air crash data based on Neo4j. *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China*, 1699–1702. https://doi.org/10.1109/ITNEC48623.2020.9085182