UNIVERSITÀ
DI TRENTO
Department of
Information Engineering and Computer Science

Doctoral School in
Information and Communication Technology

Relational Learning with Graph Neural Networks

Interpretability and handling incomplete data

Francesco Ferrini

Advisor
Prof. Andrea Passerini
Università di Trento

Co-Advisor
Prof. Manfred Jaeger
Aalborg University

2026

Abstract

Graph-structured data provide a natural representation for many real-world domains, including relational databases, knowledge graphs, and complex information systems. Learning from such data requires models that can effectively exploit relational structure while remaining robust, interpretable, and reliable under realistic data imperfections. Although Graph Neural Networks (GNNs) have emerged as a powerful paradigm for graph-based learning, their application to complex relational settings raises fundamental challenges related to scalability, interpretability, and robustness to incomplete information.

This thesis addresses these challenges through two complementary lines of research. The first part focuses on learning from heterogeneous graphs. It introduces a principled framework for learning informative meta-paths directly from data, enabling GNNs to reason explicitly over semantically meaningful relational patterns. Building on this idea, the thesis further proposes a self-explainable heterogeneous GNN that extends meta-path reasoning by incorporating learnable statistics over relational paths, yielding faithful explanations that quantify both which relational structures are relevant and how they influence predictions.

The second part of the thesis studies learning under missing node features, a pervasive characteristic of real-world relational data that has received limited principled treatment in the graph learning literature. Rather than directly addressing this problem in fully multi-relational settings, the thesis deliberately considers the simpler yet non-trivial case of homogeneous graphs as a foundational step. This controlled setting allows for a systematic analysis of how missing features interact with message passing, existing GNN architectures, and common evaluation practices. The thesis revisits standard benchmarks and missingness assumptions, showing that they often fail to reflect realistic scenarios, and introduces improved evaluation protocols based on structured and feature-dependent missingness mechanisms. In addition, it proposes a simple yet effective modeling strategy that explicitly exposes missing information within the message passing process. Extensive empirical results demonstrate improved robustness and more stable training dynamics across diverse missingness patterns and train–test distribution shifts.

Overall, this thesis advances graph learning by combining relational modeling, interpretability, and a principled analysis of robustness to incomplete data. By jointly addressing expressive relational reasoning and controlled investigations of missing-feature learning, the proposed contributions lay the groundwork for more transparent and reliable graph neural networks in real-world relational domains.

Keywords

Graph Neural Networks; Relational Learning; Interpretability; Missing Data; Robustness

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Graph-structured data constitute a natural representation for a wide range of real-world systems, in which entities interact through explicit relationships and are often associated with attribute information describing their properties. Depending on the nature of these interactions, graphs can be broadly categorized into *homogeneous* and *heterogeneous* representations. Homogeneous graphs are characterized by a single type of node and a single type of relation, whereas heterogeneous graphs allow for multiple node types and/or multiple relation types.

Many real-world datasets are more faithfully modeled using heterogeneous graphs. Examples include bibliographic networks, where authors, papers, and venues are connected by relations such as *writes*, *cites*, and *published_in*, as well as social platforms composed of users, posts, and topics linked by interactions like *follows*, *likes*, or *mentions*. Relational databases naturally fall into this category: each table can be interpreted as a set of nodes of a specific type, while primary and foreign key relationships induce typed edges between records across tables. By explicitly representing different entity types and relation semantics, heterogeneous graphs provide a richer and more faithful abstraction of complex systems. At the same time, this increased structural richness introduces additional challenges for learning, as models must account for diverse relational roles and interaction patterns.

Graph Neural Networks (GNNs) have emerged as a powerful framework for learning over graph-structured data by propagating and transforming information along edges through message-passing mechanisms. GNNs have demonstrated strong performance on tasks such as node classification, link prediction, and graph-level inference, supported by both empirical results and theoretical analyses, and offering a unified framework to learn from graph structure and node attributes. However, the majority of early

GNN models were developed under the assumption of graph homogeneity, implicitly targeting settings with a single node type, a single relation type, and complete feature information. As a result, these architectures do not directly account for the semantic diversity of relations and entities present in heterogeneous graphs, and naively treating them as homogeneous, for instance by collapsing or ignoring relation types, can obscure important relational patterns and limit both performance and interpretability.

As graph representations move from homogeneous settings to heterogeneous ones, the complexity of the underlying relational structure increases substantially. This shift raises a set of fundamental challenges that limit the applicability of existing Graph Neural Network models to real-world relational data. In this thesis, these challenges are not treated merely as open problems, but rather as the organizing principles of the contributions presented.

A first challenge concerns the ability of graph learning models to effectively account for a large number of semantically distinct relations. In heterogeneous graphs, nodes may be connected through many different relation types. Designing models that can scale to such settings while preserving the semantic distinctions between relations remains a non-trivial problem, and naive strategies that collapse or ignore relation types often lead to a loss of crucial relational information. This challenge motivates the development of models that explicitly reason over multiple relations without sacrificing their semantic identity.

A second challenge concerns *interpretability*. Standard GNN architectures are typically opaque, offering limited insight into the relational mechanisms driving their predictions. This issue becomes particularly critical in heterogeneous graphs, where different relations may contribute in qualitatively different ways. Addressing this challenge requires models whose relational reasoning process is transparent by design, allowing one to identify which relations, or combinations of relations, are responsible for a given prediction.

A third challenge relates to *data quality*, and in particular to the widespread presence of missing node features. In relational and heterogeneous data sources, attribute information is frequently incomplete, systematically missing, or unevenly distributed across node types. Such missingness patterns are intrinsic to real-world relational data and can significantly degrade both predictive performance and model reliability if not explicitly taken into account. This challenge motivates the study of learning strategies that are robust to incomplete feature information under realistic missingness mechanisms.

Taken together, these three challenges define the core contributions of this thesis.

The remainder of this chapter provides an overview of the proposed methods and results, each addressing one of these challenges through principled modeling choices and empirical evaluation.

1.1 Contributions

The contributions of this thesis are organized around three main research directions. Each contribution is presented in a dedicated subsection, reflecting both the methodological advances and the conceptual motivations underlying the work.

1.1.1 Meta-Path Graph Neural Network Learning

The first contribution introduces **MP-GNN** [34], an interpretable Graph Neural Network designed for learning on multi-relational graphs. The key idea of MP-GNN is to identify and exploit *meta-paths*, i.e., sequences of relations that capture meaningful relational patterns between node types.

Unlike traditional relational GNNs, which aggregate information from all relation types using relation-specific parameters but treat them as simultaneously relevant, MP-GNN explicitly learns which relational paths are relevant for a given prediction task. These meta-paths serve a dual role: they guide information propagation during learning and provide a human-interpretable explanation of the model's predictions. Importantly, MP-GNN does not rely on manually specified meta-paths, but instead learns them automatically from data.

This approach represents one of the earliest attempts to integrate interpretability directly into the architecture of multi-relational GNNs. By grounding predictions in a small set of learned relational patterns, MP-GNN demonstrates that it is possible to reason not only *over* graphs, but also *about* the structure of relational dependencies that drive model decisions.

1.1.2 A Self-Explainable GNN for Heterogeneous Graphs

While MP-GNN establishes the feasibility of interpretable learning through meta-paths, it relies on a simplifying assumption: the influence of a meta-path is modeled primarily through its existence. However, in many real-world applications, relational influence is cumulative, and the number or aggregate properties of relational patterns can be crucial.

The second contribution addresses this limitation through **MPS-GNN** [35, 32], the Meta-Path Statistics Graph Neural Network. MPS-GNN extends MP-GNN by incorporating learnable *meta-path statistics*, such as counts of meta-path occurrences or aggregated attributes along relational chains.

By moving beyond binary existence-based reasoning, MPS-GNN significantly increases the power of interpretable graph models. Predictions can now be explained not only in terms of *which* relational patterns matter, but also *how much* they matter. This results in more faithful explanations and improved performance, particularly in relational database settings where repeated interactions and cumulative effects are common.

Empirical evaluations show that MPS-GNN yields clearer and more informative explanations, while maintaining scalability and interpretability in complex heterogeneous graphs.

1.1.3 Learning with Missing Node Features: Benchmarks and GNNmim

The third contribution addresses a complementary but increasingly critical problem: learning on graphs with missing node features. While missing attributes naturally arise in many real-world graph-structured datasets, this work adopts a principled approach by first studying the problem in homogeneous graphs, where confounding factors can be more carefully controlled.

This contribution proceeds along two directions. First, it introduces improved evaluation protocols and benchmark datasets designed to rigorously assess GNN robustness to missing features. Unlike standard benchmarks that assume missingness completely at random, the proposed evaluation framework considers more realistic missingness mechanisms, including structured and feature-dependent patterns. A theoretical analysis complements this evaluation, clarifying the assumptions under which different methods succeed or fail.

Second, the contribution introduces **GNNmim** [33], a simple yet effective method for node classification with incomplete features. Rather than relying on complex generative models or explicit imputation, GNNmim leverages neighborhood-based propagation to compensate for missing information. Despite its simplicity, GNNmim consistently achieves strong performance across datasets and missingness regimes, often outperforming more elaborate baselines.

This contribution highlights that robustness to missing data is not merely a mod-

eling detail, but a fundamental requirement for deploying GNNs in realistic settings. Moreover, it lays the groundwork for future extensions to heterogeneous graphs, where missingness interacts with multiple node and relation types.

Thesis Overview

In summary, this thesis advances the state of graph learning along two complementary axes: interpretability and robustness. The first two contributions focus on making heterogeneous GNNs more transparent by learning and quantifying meaningful relational patterns through meta-paths. The third contribution addresses the pervasive issue of missing node features, providing both principled evaluation tools and a robust learning framework.

Together, these works move toward graph learning systems that are not only powerful, but also explainable and reliable when applied to complex, imperfect relational data.

1.2 Publications

The research activities carried out during the doctoral program have resulted in the following publications, which are directly related to the contributions presented in this thesis.

- **Francesco Ferrini**, Antonio Longa, Andrea Passerini, Manfred Jaeger.
  Meta-Path Learning for Multi-Relational Graph Neural Networks. Proceedings of the Second Learning on Graphs Conference (LoG), 2024. [34] (Chapter 3)

- **Francesco Ferrini**, Antonio Longa, Andrea Passerini, Manfred Jaeger.
  A Self-Explainable Heterogeneous GNN for Relational Deep Learning. Transactions on Machine Learning Research (TMLR), 2025. [35] (Chapter 4)

- **Francesco Ferrini**, Veronica Lachi, Antonio Longa, Bruno Lepri, Andrea Passerini, Xin Liu, Manfred Jaeger.
  Beyond Sparse Benchmarks: Evaluating GNNs with Realistic Missing Features. New Perspectives in Graph Machine Learning Workshop (NeurIPS Workshop), 2025. [33] (Chapter 5)

A substantially extended and revised version of the latter work is available as a preprint and is currently under peer review:

- **Francesco Ferrini**, Veronica Lachi, Antonio Longa, Bruno Lepri, Matono Akiyoshi, Andrea Passerini, Xin Liu, Manfred Jaeger.
  Rethinking GNNs and Missing Features: Challenges, Evaluation and a Robust Solution. arXiv preprint arXiv:2601.04855, 2026. [32] (Chapter 5)

In addition, the doctoral program has led to further publications on related topics that are not included in this thesis.

- Veronica Lachi, **Francesco Ferrini**, Antonio Longa, Bruno Lepri, Andrea Passerini. *A Simple and Expressive Graph Neural Network Based Method for Structural Link Representation.* ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling. [57]

- Veronica Lachi, **Francesco Ferrini**, Antonio Longa, Bruno Lepri, Andrea Passerini, Manfred Jaeger. *Bridging Theory and Practice in Link Representation with Graph Neural Networks.* Advances in Neural Information Processing Systems (NeurIPS), 2025. [58]

- **Francesco Ferrini**, Veronica Lachi, Antonio Longa, Bruno Lepri, Andrea Passerini. *GNNs Meet Sequence Models Along the Shortest-Path: an Expressive Method for Link Prediction.* arXiv preprint, 2025.

Chapter 2

This chapter provides the background and preliminary concepts necessary to contextualize the contributions of this thesis. We begin by introducing graphs as a general modeling paradigm for relational data, formalizing their mathematical foundations and illustrating how graph-based representations naturally arise across a wide range of real-world domains. We then review the foundations of graph learning, tracing the evolution from early feature-based and spectral approaches to modern GNNs, with particular emphasis on the message passing framework and its fundamental properties.

Building on this foundation, we discuss the transition from homogeneous graphs to heterogeneous and multi-relational graphs, highlighting the challenges introduced by multiple node and relation types. We review relational GNN architectures, with a focus on Relational Graph Convolutional Networks (R-GCNs) and their extensions, which form a central methodological reference for the models developed in later chapters. Subsequently, we introduce meta-paths as a key abstraction for capturing higher-order semantic relationships in multi-relational graphs, discussing their intuition, usefulness, and role in existing approaches.

Finally, we address the problem of learning with missing data in graph-structured domains. We review classical perspectives on missingness, examine how missing features manifest and propagate in graphs, and discuss the limitations of existing GNN approaches under incomplete information. Together, these topics establish the conceptual and methodological groundwork for the interpretability and robustness-oriented contributions presented in the remainder of this thesis.

2.1 Graphs as a Modeling Paradigm

Graphs offer a flexible and expressive framework for representing structured data. Formally, a graph is defined as a pair $G = (V, E)$, where V is a set of nodes (or vertices)

and $E \subseteq V \times V$ is a set of edges describing relationships between nodes. Depending on the application, graphs may be directed or undirected, weighted or unweighted, and static or dynamic.

The appeal of graph-based representations stems from their ability to explicitly encode relational dependencies. Many real-world systems are inherently relational: social networks describe interactions between individuals; biological networks capture relationships between genes, proteins, or molecules; transportation systems encode connectivity between physical locations; and knowledge bases represent semantic relationships between entities. In all these cases, the data cannot be adequately modeled as independent samples, as the relationships between entities carry crucial information.

2.1.1 Mathematical Foundations

More precisely, let us denote an undirected graph as $G = (V, E)$ where $|V| = n$ nodes and $|E| = m$ edges. The graph can be represented through its adjacency matrix $A \in \{0, 1\}^{n \times n}$, where $A_{ij} = 1$ if there exists an edge between nodes i and j, and $A_{ij} = 0$ otherwise. For undirected graphs, the adjacency matrix is symmetric: $A = A^T$. The degree of a node $v \in V$, denoted $\deg(v)$, is the number of edges incident to v, which corresponds to $\sum_{u \in V} A_{vu}$. The degree matrix D is a diagonal matrix where $D_{ii} = \deg(v_i)$.

In the case of directed graphs, we distinguish between in-degree and out-degree, where the in-degree of node v counts the number of incoming edges, and the out-degree counts outgoing edges. The adjacency matrix for directed graphs is generally asymmetric.

Attributed graphs extend the basic graph definition by associating features with nodes and/or edges. Formally, an attributed graph can be written as $G = (V, E, X)$, where $X \in \mathbb{R}^{n \times d}$ is a node feature matrix, with each row $x_v \in \mathbb{R}^d$ representing the attributes of node v. In many applications, these attributes encode domain-specific information: in social networks, node features might include user demographics or activity statistics; in molecular graphs, they might represent atomic properties; in citation networks, they could be document embeddings derived from text.

2.1.2 Graph-Based Data in Practice

The ubiquity of graph-structured data across disciplines underscores the importance of developing effective learning methods for graphs. In computational biology, protein-protein interaction networks model functional relationships between proteins, enabling

the study of cellular processes and disease mechanisms [8, 90]. In neuroscience, brain connectivity networks (connectomes) represent structural or functional connections between brain regions, facilitating investigation into cognitive functions and neurological disorders [89, 16].

In the domain of chemistry and drug discovery, molecules are naturally represented as graphs where atoms are nodes and chemical bonds are edges [26, 39]. This representation has enabled machine learning models to predict molecular properties, toxicity, and drug-target interactions with increasing accuracy [52, 107]. Similarly, in materials science, crystal structures and material compositions can be encoded as graphs to predict physical properties such as conductivity, stability, and reactivity [105, 20].

Knowledge graphs [46] have become central to organizing and reasoning over large-scale structured knowledge. Systems such as Google's Knowledge Graph, Wikidata, and domain-specific ontologies [101, 24] represent entities and their relationships as massive heterogeneous graphs, supporting tasks like question answering, semantic search, and recommendation [29, 76]. In these settings, the graph structure encodes rich semantic information that traditional bag-of-words or vector space models cannot capture [104].

Social network analysis [56] represents another major application area. Platforms such as Facebook, Twitter, and LinkedIn generate graphs where users are nodes and various types of interactions (friendships, follows, messages, endorsements) form edges [61, 93]. Understanding and predicting behavior in these networks, such as influence propagation, community detection, or user engagement, requires methods that can leverage both individual user attributes and the relational structure.

Traditional machine learning methods typically rely on the assumption that data points are independent and identically distributed (i.i.d.). This assumption is fundamentally violated in relational settings, where the attributes or labels of an entity may depend on those of related entities [88]. This phenomenon, known as relational autocorrelation or homophily, is pervasive in real-world networks: connected nodes tend to be similar, either because similar entities are more likely to form connections or because connected entities influence each other over time [70].

Graphs provide a principled way to model such dependencies, enabling learning algorithms to jointly reason over entities and their interactions. As increasingly complex and interconnected datasets have become available, learning over graphs has emerged as a central problem in modern machine learning.

2.2 Graph Learning

Graph learning refers to a family of methods designed to operate directly on graph-structured data. The history of graph learning can be traced back to early work on graphical models, relational learning, and spectral graph theory.

2.2.1 Early Approaches to Graph Learning

Early approaches to learning on graphs focused on handcrafted features derived from graph topology. Node-level features include degree centrality, betweenness centrality, closeness centrality, clustering coefficients, and PageRank scores. These measures capture various aspects of a node's position and importance within the network. For instance, degree centrality simply counts the number of neighbors, while betweenness centrality quantifies how often a node lies on shortest paths between other nodes, reflecting its role as a bridge in the network.

Spectral methods offer another classical approach, leveraging the eigendecomposition of graph-related matrices [100]. The graph Laplacian, defined as $L = D - A$ (where D is the degree matrix and A is the adjacency matrix), encodes structural information about the graph. The eigenvectors of L provide a basis for representing nodes in a lower-dimensional space, with the eigenvalues capturing information about graph connectivity and clustering structure. Spectral clustering, for example, partitions a graph by analyzing the eigenvectors corresponding to the smallest non-zero eigenvalues of the Laplacian.

Graph kernels [86, 14] represent another line of work, particularly prominent in chemoinformatics and bioinformatics. Methods such as the Weisfeiler-Lehman kernel, random walk kernels, and shortest-path kernels compute similarity measures between graphs by comparing local structures or paths. These kernels enable the use of kernel-based learning algorithms (such as Support Vector Machines) on graph-structured data.

While these features and methods capture important structural information, they suffer from several limitations. First, they are typically task-agnostic and require substantial domain expertise to design and select. Second, they do not naturally scale to complex or heterogeneous graphs with multiple node and edge types. Third, they treat feature extraction and learning as separate stages, rather than jointly optimizing representations for a specific task.

2.2.2 The Graph Neural Network model

Graph Neural Networks address these limitations by learning representations of graph elements in an end-to-end manner. The foundational idea, introduced in early work by [40] and [83], is to compute vector representations for nodes by iteratively aggregating information from neighboring nodes. These representations can then be used for downstream tasks such as classification, regression, or ranking.

Modern GNNs draw inspiration from several sources. Convolutional Neural Networks (CNNs), which have revolutionized computer vision, operate on regular grid-structured data (images) by applying local filters that aggregate information from neighboring pixels. GNNs extend this notion to irregular graph structures, where the concept of a "neighborhood" is defined by edges rather than spatial proximity. Similarly, Recurrent Neural Networks (RNNs) update hidden states by incorporating information from previous time steps; GNNs update node representations by incorporating information from neighboring nodes.

The expressiveness of GNNs is intimately connected to their ability to distinguish between different graph structures. A fundamental result in this area, established by [106] through the Graph Isomorphism Network (GIN), shows that the expressive power of message-passing GNNs is bounded by the Weisfeiler-Lehman (WL) graph isomorphism test [103, 74]. This test iteratively refines node labels based on the multiset of neighbor labels, and two graphs are considered WL-equivalent if this process produces the same label distributions. GNNs that aggregate neighbor information through summation (rather than mean or max pooling) can be as powerful as the WL test, meaning they can distinguish any pair of graphs that the WL test can distinguish [11]. However, certain graph structures (such as regular graphs with specific symmetries) remain indistinguishable even to WL-equivalent GNNs, motivating research into higher-order GNN architectures that go beyond standard message passing.

2.2.3 Message Passing Neural Networks

A unifying abstraction for many GNN architectures is provided by the Message Passing Neural Network (MPNN) framework, formalized by [39] In this formulation, graph learning proceeds through a sequence of message passing layers, each consisting of two main steps: message computation and node state update.

The MPNN Framework

Let $h_v^{(k)}$ denote the representation (or embedding) of node v at layer k, where $h_v^{(0)} = x_v$ is initialized to the node's input features. A generic message passing layer can be expressed as:

$$m_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\phi^{(k)}(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}) \mid u \in \mathcal{N}(v)\right\}\right),$$

followed by:

$$h_v^{(k)} = \psi^{(k)}(h_v^{(k-1)}, m_v^{(k)}),$$

where $\mathcal{N}(v)$ denotes the set of neighbors of node v, $\phi^{(k)}$ is a learnable message function, $\psi^{(k)}$ is an update function, and e_{uv} represents optional edge features.

The message function $\phi^{(k)}$ computes a message from each neighboring node u to the target node v, potentially incorporating edge features e_{uv}. The aggregation function $\text{AGGREGATE}^{(k)}$ combines these messages into a single aggregated message $m_v^{(k)}$. Finally, the update function $\psi^{(k)}$ combines the node's previous representation $h_v^{(k-1)}$ with the aggregated message to produce the new representation $h_v^{(k)}$.

A crucial requirement for the aggregation operator is permutation invariance: since the neighbors of a node form an unordered set, the aggregation function must not depend on their ordering. Formally, for any permutation π of the neighbor set, we require:

$$\text{AGGREGATE}(\{m_1, m_2, \ldots, m_k\}) = \text{AGGREGATE}(\{m_{\pi(1)}, m_{\pi(2)}, \ldots, m_{\pi(k)}\}).$$

Common choices for permutation-invariant aggregation include:

- **Summation**: $\text{AGGREGATE}(\{m_u\}) = \sum_{u \in \mathcal{N}(v)} m_u$
- **Mean**: $\text{AGGREGATE}(\{m_u\}) = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} m_u$
- **Max pooling**: $\text{AGGREGATE}(\{m_u\}) = \max_{u \in \mathcal{N}(v)} m_u$

Each choice has different properties. Summation preserves the expressiveness needed to match the WL test, while mean pooling normalizes by neighborhood size, which can be beneficial when node degrees vary widely. Max pooling captures the most salient features but may lose information about the distribution of neighbor representations.

Prominent GNN Architectures

Several influential GNN architectures can be understood as instantiations of the MPNN framework with specific choices of message, aggregation, and update functions.

Graph Convolutional Networks (GCN) [54], represent one of the most widely adopted GNN architectures. GCN performs a localized first-order approximation of spectral graph convolutions, resulting in the layer-wise propagation rule:

$$h_v^{(k)} = \sigma\left(W^{(k)} \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\deg(v) \cdot \deg(u)}} h_u^{(k-1)}\right),$$

where $W^{(k)}$ is a learnable weight matrix and σ is a non-linear activation function (typically ReLU). The normalization factor $\frac{1}{\sqrt{\deg(v)\cdot\deg(u)}}$ ensures that node representations remain in a similar scale regardless of neighborhood size.

GraphSAGE (Graph Sample and AggregatE) [43], introduces a sampling-based approach to improve scalability. Rather than aggregating over all neighbors, GraphSAGE samples a fixed-size subset of neighbors at each layer. The framework supports various aggregation functions, including mean, LSTM-based aggregation, and pooling-based aggregation. The update rule can be written as:

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \text{CONCAT}(h_v^{(k-1)}, \text{AGGREGATE}(\{h_u^{(k-1)}, u \in \mathcal{N}(v)\}))\right).$$

Graph Attention Networks (GAT) [99], introduced by Veličković et al., incorporate attention mechanisms to weigh the importance of different neighbors adaptively. Rather than treating all neighbors equally or using fixed normalization, GAT computes attention coefficients:

$$\alpha_{vu} = \frac{\exp(\text{LeakyReLU}(a^T[Wh_v \| Wh_u]))}{\sum_{u' \in \mathcal{N}(v)} \exp(\text{LeakyReLU}(a^T[Wh_v \| Wh_{u'}]))},$$

where a is a learnable attention vector and $\|$ denotes concatenation. The node representation is then updated as:

$$h_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} W^{(k)} h_u^{(k-1)}\right).$$

GAT can be viewed as a differentiable, task-specific mechanism for determining which neighbors are most relevant for each node.

Graph Isomorphism Networks (GIN) [106], were designed to maximize expressive power within the MPNN framework. GIN uses summation for aggregation and incorporates a learnable parameter ϵ to weight the node's own representation:

$$h_v^{(k)} = \text{MLP}^{(k)}\left((1+\epsilon^{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right),$$

where $\text{MLP}^{(k)}$ is a multi-layer perceptron. This design ensures that GIN can distinguish any graphs that the WL test can distinguish.

Multi-Layer Propagation and Receptive Fields

By stacking multiple message passing layers, a GNN enables information to propagate across increasingly larger neighborhoods. After k layers, the representation of a node depends on the features and structure of nodes up to k hops away. This effective receptive field grows exponentially with depth, allowing the model to capture higher-order relational dependencies.

However, this design also introduces challenges. **Oversmoothing** refers to the phenomenon where, as the number of layers increases, node representations become increasingly similar to each other, eventually converging to indistinguishable values. This occurs because repeated averaging of neighbor features causes nodes to lose their distinctive information. Oversmoothing typically becomes problematic after 3-5 layers in many real-world graphs, limiting the depth of GNN architectures.

Oversquashing [4] is another fundamental limitation, where information from distant nodes must be compressed into fixed-size embeddings as it propagates through the network. Long-range dependencies may be lost or distorted due to this information bottleneck. Recent work has proposed architectural modifications (such as residual connections, jumping knowledge networks, or graph rewiring techniques) to mitigate these issues.

The appropriate depth for a GNN depends on both the graph structure and the task. In densely connected graphs with small diameter, shallow networks may suffice. In graphs with complex, long-range dependencies (such as large social networks or biological pathways), deeper architectures or alternative designs may be necessary.

2.2.4 Learning Tasks

GNNs can be applied to a diverse range of learning tasks, which are typically categorized by the level of granularity at which predictions are made.

Node-Level Tasks : In node classification, the goal is to predict a label for each node based on its features and graph context. This is one of the most common GNN applications. Examples include predicting user interests in social networks, classifying research papers into topics in citation networks, or identifying protein functions in biological interaction networks. Node regression extends this to continuous prediction targets, such as estimating property values in real estate networks or predicting gene expression levels.

After K message passing layers, node representations $h_v^{(K)}$ are fed into a task-specific output layer (typically a linear classifier or regressor):

$$\hat{y}_v = \text{softmax}(W_{\text{out}} h_v^{(K)} + b_{\text{out}}).$$

Training typically uses semi-supervised learning, where labels are available for a subset of nodes, and the GNN must generalize to unlabeled nodes by leveraging graph structure and feature similarity.

Edge-Level Tasks : Link prediction aims to infer the presence or properties of edges between nodes. This is formulated as a binary classification problem (edge exists or not) or as a ranking problem (ranking candidate edges by likelihood). Applications include recommending friends in social networks, predicting protein-protein interactions, or completing knowledge graphs.

A common approach combines the representations of two nodes (e.g., through concatenation, element-wise product, or inner product) and applies a scoring function:

$$s(v, u) = \sigma(W_{\text{link}}[h_v^{(K)} \| h_u^{(K)}]),$$

or simply:

$$s(v, u) = \sigma(h_v^{(K)T} h_u^{(K)}).$$

Edge classification extends this to predicting edge types or properties, which is particularly relevant in multi-relational settings.

Graph-Level Tasks : In graph classification, the objective is to predict a label for an entire graph. This requires aggregating node representations into a single graph-level representation. Common aggregation schemes include global mean/max/sum pooling:

$$h_G = \text{READOUT}(\{h_v^{(K)} \mid v \in V\}),$$

where READOUT is a permutation-invariant function. More sophisticated hierarchical pooling methods [41] (such as DiffPool [109], TopK pooling, MinCutPool [10] ox MacCutPool [1]) learn to coarsen the graph progressively, creating hierarchical representations.

Applications of graph classification include molecular property prediction (e.g., toxicity, solubility), program analysis (classifying programs represented as abstract syntax trees or control flow graphs), and social network analysis (classifying communities or groups).

The versatility of GNNs across these tasks highlights the generality and flexibility of the message passing framework.

2.3 From Homogeneous to Heterogeneous Graphs

Most early GNN models were developed under the assumption of homogeneous graphs, where all nodes and edges are of a single type. While this assumption simplifies modeling and implementation, it is often unrealistic in practice. Many real-world datasets involve multiple types of entities and relations, each with distinct semantics and characteristics.

A **heterogeneous graph** (or **heterogeneous information network**) is defined as a graph $G = (V, E)$ with node type mapping $\phi : V \to \mathcal{A}$ and edge type mapping $\psi : E \to \mathcal{R}$, where $\mathcal{A}$ is a set of node types and $\mathcal{R}$ is a set of edge types (relations), with $|\mathcal{A}| + |\mathcal{R}| > 2$. This formalization allows different nodes and edges to have different attributes, semantics, and roles.

For example, in an academic network, node types might include $\mathcal{A} = \{\text{Author}, \text{Paper}, \text{Venue}\}$, and relation types might include $\mathcal{R} = \{\text{writes}, \text{cites}, \text{published-in}\}$. Each relation connects specific node types: "writes" connects Authors to Papers, "cites" connects Papers to Papers, and "published-in" connects Papers to Venues.

Multi-relational graphs arise naturally in many domains:

- **Knowledge Graphs**: Systems like DBpedia, YAGO, or Freebase represent entities (people, places, organizations) and relations (born-in, located-in, works-for) as typed graphs. These graphs often contain millions of entities and thousands of relation

types.

- **Relational Databases**: A relational database can be interpreted as a multi-relational graph where each table corresponds to a node type and foreign key relationships define typed edges between records. For instance, a hospital database might have tables for Patients, Doctors, Diagnoses, and Treatments, with relationships linking these entities.

- **Social Networks**: Modern social platforms support multiple interaction types (friend, follow, message, like, share), multiple entity types (users, posts, groups, events), and rich relational structures. Understanding these networks requires distinguishing between different kinds of connections.

- **Biological Networks**: Systems biology often models complex interactions between genes, proteins, pathways, and diseases. A gene regulatory network might include gene-gene interactions (regulation), protein-protein interactions (binding), and gene-disease associations, each carrying different biological meanings.

2.3.1 Challenges in Heterogeneous Graph Learning

Modeling Heterogeneous graphs introduces several challenges beyond those encountered in homogeneous settings.

First, **different relations require different transformations**: Information propagated along a "co-author" edge in a citation network should be processed differently from information propagated along a "cites" edge. Simply treating all edges uniformly can obscure important patterns and reduce predictive performance.

Second, **relational structure can be highly imbalanced**: Some relation types may be very common (e.g., "follows" in a social network), while others are rare (e.g., "blocks"). Some node types may have many instances, while others have few. This imbalance can lead to biased learning and requires careful handling during training.

Third, **expressiveness and interpretability trade-offs** become more complex. In homogeneous graphs, a node's k-hop neighborhood provides a natural notion of context. In heterogeneous graphs, many different relational paths (sequences of edge types) can connect two nodes, and not all paths are equally meaningful. Identifying which relational patterns (often called meta-paths [25] or relational schemas) are most informative for a given task is a key challenge.

Finally, **scalability** becomes critical when the number of node types and relation types is large. Maintaining separate parameters for each relation can lead to overparameterization, increasing memory requirements and the risk of overfitting.

2.3.2 Relational Graph Neural Network

Relational Graph Convolutional Networks (R-GCNs) [84], were among the first GNN architectures explicitly designed for learning on multi-relational graphs. R-GCN extends the standard GCN framework by associating each relation type with a separate transformation, allowing messages passed along different relations to be processed differently.

In R-GCN, the representation of a node at layer k is updated by aggregating relation-specific messages from its neighbors:

$$h_v^{(k)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_v^r}\frac{1}{c_{v,r}}W_r^{(k)}h_u^{(k-1)} + W_0^{(k)}h_v^{(k-1)}\right),$$

where:

- $\mathcal{N}_v^r$ denotes the set of neighbors of node v connected via relation r,
- $W_r^{(k)}$ is a relation-specific weight matrix,
- $W_0^{(k)}$ is a self-connection weight matrix,
- $c_{v,r}$ is a normalization constant (e.g., $c_{v,r} = |\mathcal{N}_v^r|$),
- σ is a non-linear activation function.

This formulation preserves relational semantics: information from different relation types is transformed independently before being aggregated. The self-connection term $W_0^{(k)}h_v^{(k-1)}$ ensures that the node retains information from its previous representation.

2.3.3 Applications and Extensions

R-GCN has been successfully applied to tasks such as entity classification in knowledge graphs and link prediction in multi-relational networks. Its relational message passing paradigm has inspired numerous extensions and variations.

Some notable extensions include: -

CompGCN [98] (Composition-based GCN), which learns composition operators to combine entity and relation embeddings -

RGAT [17] (Relational Graph Attention Network), which incorporates attention mechanisms to weight different relations and neighbors adaptively -

Heterogeneous Graph Transformer (HGT) [47], which applies transformer-style attention mechanisms to heterogeneous graphs, allowing dynamic, context-dependent aggregation across node and edge types

Despite these advances, challenges remain. Many relational GNNs, including R-GCN, operate as black-box models: they achieve strong predictive performance but do not inherently explain which relational patterns drive their predictions. Understanding why a particular node received a certain label, or which relational paths were most influential, requires post-hoc analysis or auxiliary explanation techniques.

Moreover, the presence of **missing or incomplete features**, a common issue in real-world heterogeneous and multi-relational datasets—can degrade performance. Different node types may have different feature schemas, and certain attributes may be missing for subsets of nodes. Handling such missingness robustly requires careful modeling choices, which are explored in later chapters of this thesis.

In summary, relational GNNs extend the message passing framework to handle multiple node and edge types, enabling learning on complex, heterogeneous graphs. R-GCN, as a foundational model in this space, demonstrates both the promise and the challenges of relational graph learning. The methods developed in subsequent chapters build upon this foundation, focusing on interpretability and robustness in complex relational settings.

2.4 Meta-paths in Heterogeneous Graphs

In heterogeneous graphs, entities can be connected through a variety of relation types, each carrying distinct semantic meaning. As a result, the local neighborhood of a node may contain information that is heterogeneous not only in content but also in relational structure. Simply aggregating information across all relations, as done by many relational GNN architectures, can therefore dilute task-relevant signals and obscure meaningful relational patterns.

Meta-paths [91] have emerged as a principled abstraction to capture higher-order semantic relationships in such graphs. Intuitively, a meta-path specifies a sequence of relation types that defines a semantic pattern of interaction between entities. By constraining information flow to follow specific relational schemas, meta-paths act as structured filters over the graph, enabling models to focus on semantically meaningful connections while ignoring irrelevant relational noise.

2.4.1 Definition and Intuition

Formally, given a multi-relational graph with relation set $\mathcal{R}$, a *meta-path* [91, 25] is defined as an ordered sequence of relations

$$\text{mp} = r_1 \to r_2 \to \cdots \to r_L, \qquad r_i \in \mathcal{R}.$$

A meta-path thus describes a composite relation obtained by chaining together multiple atomic relations. Two nodes are said to be connected by a meta-path if there exists a path in the graph whose edge types follow the specified relation sequence.

Crucially, meta-paths are defined at the level of relation types rather than individual nodes. This makes them reusable semantic templates that generalize across different parts of the graph. For example, in an academic network, the meta-path `Author-Paper-Venue-Paper-Author` captures the notion of authors publishing in the same venue, regardless of the specific papers or venues involved.

From a modeling perspective, meta-paths provide a way to encode domain knowledge about which relational patterns are likely to be informative for a given task. At the same time, they offer a mechanism to control the receptive field of a model, restricting aggregation to paths that respect specific semantic constraints.

2.4.2 Why Meta-paths Are Useful

The utility of meta-paths stems from several key observations. First, in many real-world multi-relational graphs, only a small subset of relations and relation compositions are relevant for a specific predictive task [84, 102]. Aggregating indiscriminately over all relations may introduce noise and degrade performance, especially when the number of relation types is large.

Second, meta-paths provide an explicit representation of higher-order dependencies. While standard message passing aggregates information over k-hop neighborhoods without distinguishing how nodes are reached, meta-paths differentiate between distinct relational routes connecting the same nodes. This distinction is particularly important in heterogeneous graphs, where different paths may correspond to fundamentally different semantics.

Third, meta-paths offer an interpretable abstraction. Because they are expressed as sequences of relation types, meta-paths can often be understood directly in domain terms. This makes them a natural candidate for explaining model predictions at a structural level, linking outcomes to concrete relational patterns in the data.

2.4.3 Meta-paths in Existing Approaches

Meta-paths have been widely used in the literature on heterogeneous information networks and knowledge graphs [84, 102, 19, 37, 102, 47, 113]. Early approaches relied on manually specified meta-paths, provided by domain experts, to guide feature extraction or define similarity measures between nodes. While effective in constrained settings, this strategy does not scale and requires substantial prior knowledge.

Subsequent methods sought to alleviate this dependence by learning soft combinations of relations, for instance by assigning trainable weights to different edge types or by learning adjacency transformations that implicitly encode meta-paths. Although these approaches reduce the need for explicit enumeration, they typically struggle when the number of relations is large and often lack a clear semantic interpretation.

More recently, attention has shifted toward methods that aim to automatically discover informative meta-paths from data. These approaches frame meta-path selection as a learning problem, searching over the space of possible relation sequences and retaining those that are most predictive for the task at hand. However, many existing solutions either ignore node features during meta-path discovery or assume that the informativeness of a meta-path depends solely on its existence, limiting their applicability in richer relational settings.

2.5 Learning with Missing Data in Graph-Structured Domains

Missing or incomplete data is a pervasive characteristic of real-world relational datasets [31]. In particular, relational databases, which naturally give rise to multi-relational graphs [79] frequently contain missing values due to heterogeneous data sources, evolving schemas, optional attributes, data entry errors, or privacy-related constraints. As a consequence, several entities may exhibit missingness patterns, and the absence of information is often structured rather than random.

When relational databases are represented as graphs, missingness typically manifests at the level of node attributes, while the relational structure induced by primary and foreign keys is largely preserved. This setting is common in enterprise databases, biomedical records, and knowledge-driven systems, where relationships between entities are well-defined but attribute completeness varies significantly across tables or entity types [24, 46]. As a result, learning algorithms must often operate on graphs with reliable relational structure but partially observed node features.

While classical machine learning has developed a rich theoretical framework for learning with missing data, most graph learning approaches implicitly assume complete node features. Understanding how missingness interacts with graph structure and message passing is therefore essential for designing robust and principled GNNs.

2.5.1 Missing Data: Classical Perspective

Let $X \in \mathbb{R}^{n \times d}$ denote the complete data matrix of node features. Let $M \in \{0,1\}^{n \times d}$ be a binary missingness mask, where $M_{ij} = 1$ indicates that the feature value X_{ij} is observed and $M_{ij} = 0$ indicates that it is missing.

We denote by $X^{\text{obs}} = \{X_{ij} : M_{ij} = 1\}$ the set of observed feature values and by $X^{\text{mis}} = \{X_{ij} : M_{ij} = 0\}$ the set of missing entries of X.

The relationship between the data and the missingness mechanism is commonly described through the conditional distribution $p(M \mid X)$, which specifies how missing values arise. Based on this dependency, three standard missingness mechanisms are distinguished [81].

Missing Completely at Random (MCAR) Missingness is said to be completely at random if it is independent of both observed and unobserved data:

$$p(M \mid X^{\text{obs}}, X^{\text{mis}}) = p(M).$$

Under MCAR, the observed data constitutes an unbiased subsample of the full dataset. While statistical efficiency may be reduced, learning procedures remain valid without explicitly modeling the missingness mechanism.

Missing at Random (MAR) Missingness is at random if it depends on the observed data but not on the missing values themselves:

$$p(M \mid X^{\text{obs}}, X^{\text{mis}}) = p(M \mid X^{\text{obs}}).$$

In this setting, conditioning on the observed data suffices to account for the missingness process. Many classical imputation and likelihood-based methods rely on this assumption, either explicitly or implicitly.

Missing Not at Random (MNAR) Missingness is not at random when it depends directly on the unobserved values:

$$p(M \mid X^{\text{obs}}, X^{\text{mis}}) \neq p(M \mid X^{\text{obs}}).$$

This represents the most challenging scenario, as the missingness mechanism is entangled with the unknown data itself. Ignoring this dependency generally leads to biased estimates and invalid inference.

These definitions originate from the i.i.d. setting. In relational and graph-structured domains, however, dependencies between entities may induce correlations between missingness patterns across nodes, complicating the direct application of classical assumptions.

2.5.2 Missingness in Graphs

In graph-structured data, missingness can take several forms. Node features may be partially observed [66], entire attributes may be unavailable for certain node types, or features may be systematically missing for specific subsets of nodes. In multi-relational settings, different node types often follow different attribute schemas, leading to heterogeneous and structured missingness patterns.

A key characteristic of missingness in graphs is its non-local impact. Because GNNs propagate information through message passing, the absence of features at one node can influence the representations of many others. Missing data can therefore propagate through the network, affecting predictions far beyond the nodes where information is originally absent.

Moreover, the missingness mechanism may be correlated with the relational structure itself. For example, attributes may only be recorded for entities participating in specific relations, or highly connected nodes may be more thoroughly observed than peripheral ones. Such dependencies violate classical assumptions and complicate the design of learning algorithms for graph-structured data.

2.5.3 Limitations of Existing GNN Approaches

Most existing graph neural network architectures are not explicitly designed to handle missing node features. In practice, common strategies include zero-filling missing values, mean or median imputation, or masking missing entries. While these approaches are straightforward to implement, they implicitly impose strong assumptions on the data

and the missingness mechanism.

Deterministic imputation methods treat missing values as fixed constants, effectively collapsing uncertainty into arbitrary placeholders. Masking strategies preserve information about which values are missing, but typically rely on the model to learn how to handle incomplete inputs without an explicit probabilistic interpretation. As a result, these approaches may introduce biases or degrade performance, particularly when missingness is structured or correlated with the target variable.

In multi-relational graphs, these limitations are further amplified. Different node types may exhibit vastly different missingness patterns, and naive imputation can distort relational signals or introduce spurious correlations. Consequently, existing GNN approaches may struggle to provide reliable predictions in the presence of incomplete and heterogeneous feature information.

Chapter 3

Meta-Path Graph Neural Network Learning

This chapter presents a learning framework for identifying informative meta-paths in multi-relational graphs and for integrating them into graph neural network architectures. It constitutes the first core methodological contribution of this thesis and builds directly upon the concepts of multi-relational graphs and meta-paths introduced in Chapter 2.

Existing multi-relational graph neural networks adopt one of two main strategies to handle relation heterogeneity: either reducing the problem to low-level parameter learning [84, 113], or relying on handcrafted meta-paths [19, 37, 102]. However, the former approach faces scalability issues in the presence of many relations, while the latter requires substantial domain expertise to identify relevant meta-paths.

In this chapter, we introduce a principled approach to learning meta-paths directly from data and to constructing meta-path-aware graph neural networks that are accurate and scalable. The key element of the proposed method is a scoring function that measures the potential informativeness of relations in the incremental construction of a meta-path. This enables the learning of a Meta-Path Graph Neural Network (MP-GNN) that leverages a small number of informative meta-paths while retaining node-specific feature information during aggregation.

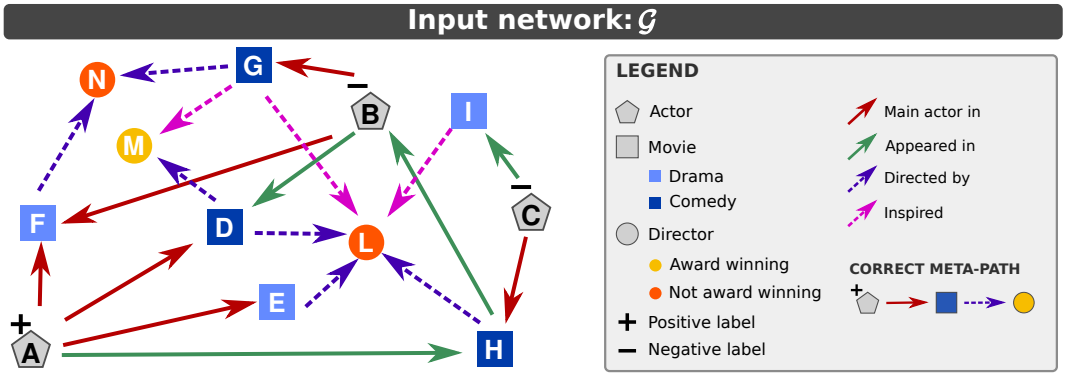


Figure 3.1: A toy node classification problem. Node shapes indicate types, while node attributes and edge types are encoded as colors. The task consist in labelling actor nodes (pentagons, which do not have attributes). An *Actor* is labelled as positive if involved as main actor in a drama directed by and award winning director.

3.1 MP-GNN learning

The goal of our approach is to learn relevant meta-paths that can serve as predictive features for the node classification task. [1] Differently from the approaches that use all relations at the same time by weighting each edge type contribution, we focus on finding the relevant chains of relations (i.e., meta-paths) beneficial for making accurate predictions. Note that, differently from what happens in purely relational settings [87, 71, 112, 59, 60, 75], we assume here that the informativeness of a meta-path also depends on the features of the nodes that are being traversed (which include the node type, but also node attributes and potentially pre-computed node embeddings). Our approach accounts for this aspect in mining relevant meta-paths. Meta-paths are constructed in a greedy, incremental approach using the idea of relational information gain [63] to score candidate extensions of an already constructed meta-path prefix. Consider the toy node classification task in Figure 3.1. To incrementally build the correct meta-path (bottom right in the legend), one has to realize that "Main actor in" is a better candidate than "Appeared in". Intuitively, our scoring function does this by assigning weights (i.e., pseudo-labels) to nodes reached by a candidate relation in such a way that the label of the target node can be inferred by propagating the pseudo-label of the neighbour. Figure 3.2 shows an example of weight assignment for the "Main actor in" and "Appeared in" relations, indicating a higher score for the former. However, these pseudo-labels only hint at the potential informativeness of the relation. Indeed, being

[1] We focused on node classification in this paper, but the approach can be adapted to deal with graph classification tasks, e.g., by considering a supernode connected to all graph nodes as the target.

a main actor in a movie is not enough to qualify as an award winning actor, even in the toy example of Figure 3.1. The movie should be a drama (node feature), and be directed by an award winning director. Whether this potential informativeness actually materializes is determined in the following steps, where the pseudo-labels become new prediction targets for the next extension of the meta-path under construction. Details of this method are described in Section 3.1.1.

Once a candidate meta-path has been extracted, it is used to build a MP-GNN in which each layer corresponds to a relation on the meta-path. Section 3.1.2 presents a formalization of this architecture, and shows how to extend it to account for multiple meta-paths. Finally, these ingredients are combined into an overall algorithm to jointly learn a meta-path and a corresponding MP-GNN. For the sake of readability, the algorithm is presented for the single meta-path case, but its extension to multiple meta-paths using a beam search is straightforward (we employed a beam size equal to three in the experiments). Note that this algorithm is designed to identify existential meta-path features, i.e., cases where the existence of an instance of a meta-path is informative for the class label. Adaptations and extensions where counts or proportions of meta-path realizations are the relevant feature are subject of future work.

3.1.1 Scoring function

The goal of the scoring function is that of providing a measure of informativeness of a relation towards predicting node labels. We start discussing the first iteration, i.e., identifying the first relation in the meta-path, and then show how the function can be adapted to deal with meta-path extension.

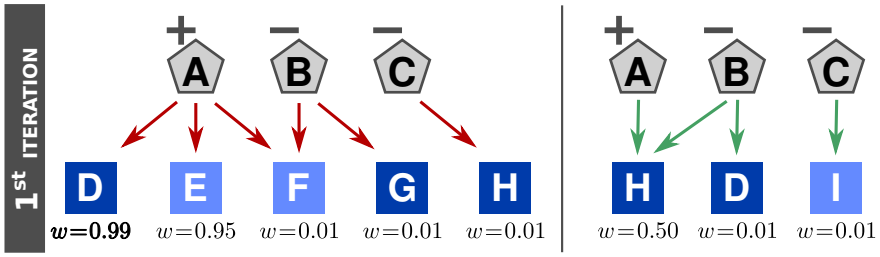


Figure 3.2: **First iteration:** the scoring function assigns a high score to the red ("Main actor in") relation (left panel) by giving large weights to movies D and E, that are only connected to a positive node, and small weights to the other movie nodes. On the other hand, the green ("Appeared in") relation (right panel) has low score, as no weight assignment can jointly explain the positive label of the A node and the negative label of the B node.

In the first iteration, the scoring function takes as input a list of nodes together with their target labels. Under the previously introduced existential quantification assumption, a candidate relation r is informative for the label of a node i if at least one of the neighbors $\mathcal{N}_i^r$ of i according to r belongs to the ground-truth meta-path, and i has the right features (remember that the label is assumed to depend on the combination of the meta-path and the features of the nodes being traversed). This can be formalized as follows:

$$\tilde{y}_i^r = \Theta^T h_i^{(0)} \cdot \max_{j \in \mathcal{N}_i^r} w_j \tag{3.1}$$

Here $h_i^{(0)} \in \mathbb{R}^d$ denotes the feature vector of node i, and $\Theta \in \mathbb{R}^d$ is a learnable weight vector accounting for the contribution of the node features. The term $\Theta^T h_i^{(0)}$ therefore represents a scalar score associated with node i. The variable w_j is a learnable node weight that is set (close) to 1 if node j is predicted as belonging to the ground-truth meta-path, and (close to) zero otherwise.The score of r is computed by minimizing the MSE between the predicted and ground truth node labels over Θ and $\mathbf{w}$:

$$s_r = \min_{\Theta, \mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} (\tilde{y}_i^r - y_i)^2 \tag{3.2}$$

The relation with the minimum score is selected as the first relation of the meta-path.

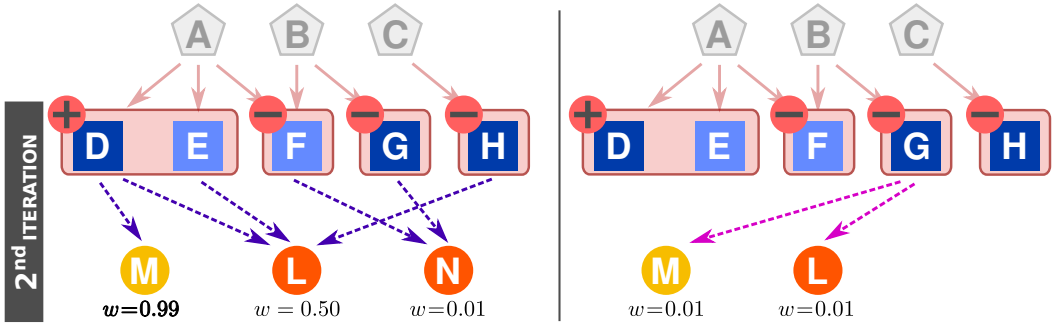


Figure 3.3: **Second iteration**: The scoring function assigns a high score to the purple ("Directed by") relation (left panel) by assigning a large weight to the M director, which is only one connected to a positive bag, and small weights to the other directors. On the other hand, the "pink" (Inspired) relation (right panel) gets a low score as no weight assignment is compatible with the positive bag.

To explain how the scoring of the following relations in the meta-path works, it is important to remember that the weights $\mathbf{w}$ represent a tentative assignment to neighbours as belonging or not-belonging to the ground-truth meta-path (i.e., their *potential informativeness*). Multiple potential assignments can be minimizers of Eq. 3.2. In the

left panel of Figure 3.2, where relation r_1 (red) is being scored, one minimizer of Eq. 3.2 requires $w_E = 1$ (to account for the positive label of node A) and $W_F = 0$ (to account for the negative label of node B). On the other hand, (0,1), (1,0) and (1,1) are all valid assignments to the (W_D, W_E) pair. Indeed, the only constraint that the positive label of A enforces is that the bag (W_D, W_E) contains at least one node with value 1, as happens in multi-instance classification settings [5]. We thus generate labelled bags of nodes for the following iteration(s) of meta-path construction, that will play the role of the node labels y in the initial iteration. Positive bags are computed as follows:

$$B^+(i) = \{j \in \mathcal{N}_i^r \mid \nexists k : j \in \mathcal{N}_k^r \wedge y_k = -1\} \tag{3.3}$$

where i is a positive-labelled node ($y_i = +1$). Negative bags, on the other hand, are singletons, i.e., given a negatively-labelled node j, we create a negative bag $B^-(k) = \{k\}$ for each of its neighbors $k \in \mathcal{N}_j^r$. The informativeness of the new relation s (as extension of relation r) can now be computed in terms of its potential in predicting bag labels:

$$\tilde{y}^s_{B(i)} = \max_{j \in B(i)} \left(\Theta^T h_j^{(0)} \cdot \max_{k \in \mathcal{N}_j^s} w_k \right) \tag{3.4}$$

and obtained minimizing MSE at the bag-label level. See Figure 3.3 for a graphical representation of the components involved.

Once the next relation is selected, the procedure could in principle continue, by further expanding positive bags with a procedure analogous to Eq. 3.3, where i is itself replaced with a bag of nodes. However, this procedures ends up diluting information too much, so that the informativeness of relations becomes difficult to predict. We rather assign a positive label to a node within a bag if it is used to predict the positive label of the bag (Eq. 3.1) at least once out of M restarts with randomly initialized weights. See the Appendix for the details.

3.1.2 MP-GNN

In the single meta-path MP-GNN, a meta-path $mp = r_1, ..., r_L$ induces a multi-relational GNN with L layers, that we denote by MP-GNN(mp). The first layer is associated to the last relation of the meta-path r_L, and so on until the final layer which is associated

with r_1. The message passing update is formalized as follows:

$$h_i^{(l+1)} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{j \in \mathcal{N}_i^{r_{L-l+1}}} \frac{1}{|\mathcal{N}_i^{r_{L-l+1}}|} W^{(l)} h_j^{(l)}\right) \tag{3.5}$$

where l ranges from 1 to L.

The definition can be generalized to deal with multiple meta-paths by concatenating embeddings coming from the different meta-paths:

$$h_i^{(l)} = \Big\|_{k=1}^{K} h_{(i,k)}^{(l)} \tag{3.6}$$

where K is the number of meta-paths, $h_{(i,k)}^{(l)}$ is the embedding of node i according to meta-path k computed using Eq. 3.5 and $\|$ is the concatenation operator.

It is worth mentioning here that while this definition of MP-GNN is a straightforward adaptation of the `RGCN` architecture to deal with learned meta-paths, more complex architectures involving pre-defined meta-paths could in principle be employed [37, 102, 62]. We opted for this simple choice in the paper so as to best single out the contribution of the scoring function in determining the performance of the resulting architecture.

3.1.3 Overall algorithm

The overall algorithm for learning MP-GNN is outlined in Algorithm 2. The algorithm takes as inputs a heterogeneous graph G, a set of candidate relations $\mathcal{R}$, a set of node-label pairs *labels* and a hyper-parameter L_{MAX} indicating the maximal length of candidate meta-paths. The algorithm repeatedly call the scoring function (Eq. 3.2) to score candidate relations and keeps track of the best scoring one. It then builds an MP-GNN with the current (partial) meta-path and trains it to predict node labels, using F_1 score (computed on a validation set, omitted in the algorithm for the sake of compactness) as final meta-path evaluation metric. Note that this is the only "real" measure of meta-path quality, as the one computed by the scoring function is still a "potential" informativeness, that only fully materializes when the meta-path is embedded into an MP-GNN. The algorithm keeps track of the highest F_1 meta-path prefix so far, and proceeds by generating labelled bags as described in Section 3.1.1 for the next round of relation scoring.

As previously mentioned, the algorithm is presented for the sake of simplicity in the single meta-path case. However, the actual implementation performs beam search on the space of meta-paths, retaining the K top-scoring ones according to Eq. 3.2

and concatenating their embeddings into the MP-GNN as per Eq. 3.6. Notice that in evaluating the resulting MP-GNN, meta-paths not contributing to increasing F_1 are discarded, so as to retain only the informative meta-paths in the final architecture.

Algorithm 1 LEARNMP-GNN algorithm. G is a heterogeneous graph, $\mathcal{R}$ is the set of possible relations, *labels* is the initial set of node-label pairs and L_{MAX} is the maximal meta-path length

procedure LEARNMP-GNN(G, $\mathcal{R}, labels, L_{MAX}$)
  Initialize $mp^* \leftarrow [\,]$, $mp \leftarrow [\,]$, $F_1^* \leftarrow 0$, $target \leftarrow labels$
  while $|mp| < L_{MAX}$ **do**
    for $r \in \mathcal{R}$ **do**
      $s_r \leftarrow$ SCORE-RELATION$(G, target, r)$ ▷ Equation 3.2
    end for
    $r^* \leftarrow$ best scoring relation
    $mp \leftarrow mp \cup \{r^*\}$
    $gnn \leftarrow$ TRAIN(MP-GNN$(mp), G, labels)$
    $F_1 \leftarrow$ TEST(gnn)
    if $F_1 > F_1^*$ **then**
      $mp^* \leftarrow mp,\ F_1^* \leftarrow F_1$
    end if
    $target \leftarrow$ GENERATE-BAGS$(target, r^*)$ ▷ Section 3.1.1
  end while
  return mp^*
end procedure

3.2 Experimental results

Our experimental evaluation aims to answer the following research questions:

Q1 Can `MP-GNN` recover the meta-path that generates the node labels in the synthetic datasets as the number of candidate relations increases?

Q2 Is `MP-GNN` competitive with existing approaches in real-world datasets with few relations?

Q3 Does `MP-GNN` outperform existing approaches in real-world datasets with many relations?

We compared `MP-GNN` with existing solutions that: 1) do not require to pre-specify relevant meta-paths, 2) can handle (possibly high-dimensional) node features. Given these requirements, we identified the following competitors:

- `RGCN` [84], a generalization of the GCN architecture to the multi-relational case, that employs a different matrix of parameters for each edge type.

- `GTN` [113] can convert an input graph into different meta-path graphs for specific tasks and learn node representations within these graphs.

- `FastGTN` [113], an efficient variant of `GTN` that avoids adjacency matrices multiplication for graph transformation.

- `R-HGNN` [111], employs a different convolution for each edge type. Finally combines different embeddings with a cross-relation message passing.

- `HGN` [65], utilizes GAT as backbone to design an extremely simple model.

We implemented `MP-GNN` using Pytorch Geometric [78], while the code of the competitors was taken from their respective papers. For `MP-GNN` we used Adam optimizer with a learning rate of 0.01. We set the maximum meta-path length $L_{MAX} = 4$ and the beam size $K = 3$. We used an 70/20/10 split between train, validation and test in all cases, with model selection performed on the validation set for all methods. We employed F1-macro score on the test set as evaluation metric to account for the unbalancing present in many of the datasets.

In the following we report the experimental setting and the results we obtained in addressing each of the research questions under investigation. The statistics of the datasets used in the experiments are reported in the Appendix.

Q1: `MP-GNN` consistently identifies the correct meta-path In order to answer the first research question, we designed a controlled setting where the correct meta-path is known, and experiments can be run for an increasing number of candidate relations. We generated synthetic datasets where nodes are typed A or B. The underlying graph structure is generated using an Erdős–Rényi model, where edges are sampled independently with fixed probability. Node types and relation types are then assigned randomly to nodes and edges, respectively. The number of relations $|\mathcal{R}|$ varies in $\{4, 8, 10, 14\}$, and the number of relations that can connect more than one pair of node types (e.g., $A \xrightarrow{r_1} B$ and $A \xrightarrow{r_1} A$). The ground truth meta-path consists of a (valid) sequence of relations and nodes of a given type (e.g., $x \xrightarrow{r_1} A \xrightarrow{r_3} B$, with x being a node of arbitrary type). Nodes are labelled as positive if found to be starting points of a ground-truth meta-path, and negative otherwise. We generated labelled datasets using ground-truth

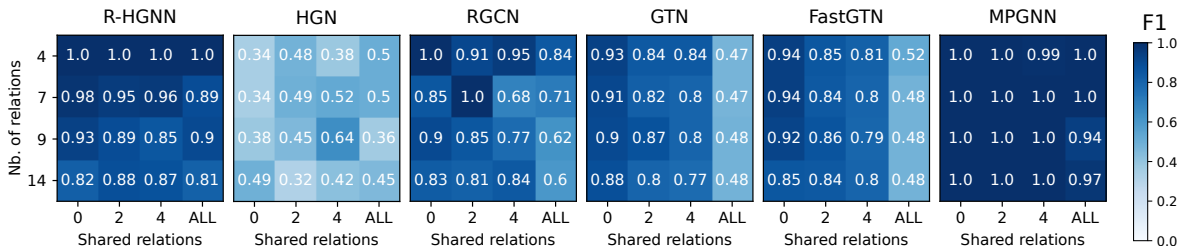


Figure 3.4: Synthetic setting: F1-score (the darker the better) as a function of the overall number of relations (rows) and the number of shared relations (columns).

meta-paths of different lenghts $L \in \{2, 3, 4\}$. Details of the different settings can be found in the Appendix (Figure A.1).

Figure 3.4 shows the F_1 score for each model when varying the overall number of relations and the number of shared relations, for a ground-truth meta-path of length three. Darker cells correspond to higher F_1 value. Results show that the performance of existing multi-relational GNN approaches is severely affected by the relational complexity of the graph, with `RGCN` and `R-HGNN` being more sensible to the overall number of candidate relations and `GTN` and `FastGTN` having bigger problems with the number of shared relations, while `HGN` has poor performance in all settings, likely due to its lack of an explicit modelling of relation types. Conversely, `MP-GNN` consistently achieves optimal or quasi-optimal performance in all settings. Whenever $F_1 = 1$, `MP-GNN` manages to perfectly recover the ground-truth meta-path, while values smaller than one are due to spurious relations being added at the end of the meta-path (which however have a limited impact on predictive performance).

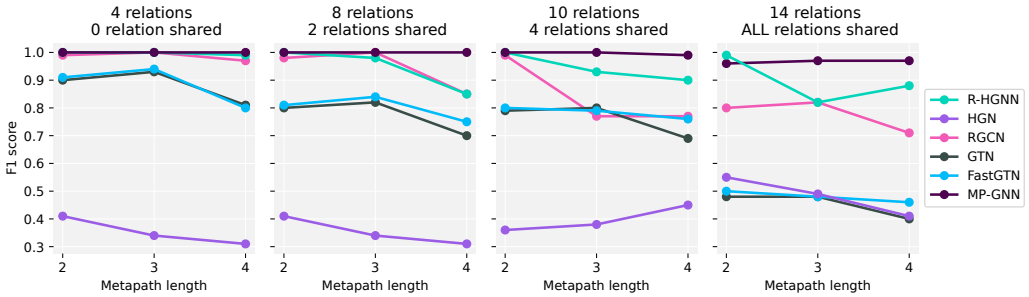


Figure 3.5: Synthetic setting: F1-score as a function of the ground-truth meta-path length, for an increasing complexity of the search space.

Figure 3.5 shows results when increasing the relational complexity of the network *and* the length of the meta-path characterizing the positive class. Each setting corresponds to an entry in the main diagonal of Figure 3.4, where we additionally varied the length of the meta-path from 2 to 4. Results show that `GTN`, `FastGTN` and `HGN` struggle in most

settings, while RGCN and R-HGNN are competitive in the simplest settings (few relations and/or short meta-paths) but its performance quickly degrade when the size of the search space increases. Again, MP-GNN consistently achieves excellent performance in all settings, almost always perfectly recovering the ground-truth meta-path.

Q2: MP-GNN achieves state-of-the-art results on real-world datasets with few relations The second set of experiments focuses on popular real-world benchmarks for multi-relational GNNs. In all cases the task is multi-class classification at the node level. We quickly summarize the characteristics of the benchmarks in the following: **IMDB**: a dataset extracted from the popular Internet Movie Database. It contains 3 types of nodes (movies (M), directors (D) and actors (A)) and uses the genres of movies as labels. **DBLP**: citation network where nodes are of paper (P), author (A) or conference (C) type, connected by edge types PA, AP, PC, CP, and the task is predicting the research area of authors. **ACM**: again a citation network, similar to the one of DBLP with conference nodes replaced by subject (S) nodes (and edge types replaced accordingly).

Table 3.1: Few-relations datasets. **(Top)**: F_1 scores, mean and std computed over five runs. Best results highlighted in bold. **(Bottom)**: learnt meta-paths for MP-GNN and GTN/FastGTN (which learn the very same meta-paths). Other baselines not reported as they do not explicitly extract meta-paths.

Model	DBLP	IMDB	ACM
R-HGNN	0.86(±0.04)	**0.64**(±0.01)	0.9(±0.01)
HGN	**0.94**(±0.01)	0.63(±0.02)	0.92(±0.02)
RGCN	0.91(±0.01)	0.6(±0.01)	0.9(±0.02)
GTN	0.9(±0.01)	0.62(±0.01)	0.91(±0.01)
FastGTN	0.92(±0.00)	0.63(±0.01)	**0.93**(±0.00)
MP-GNN	**0.94**(±0.01)	**0.64**(±0.01)	**0.93**(±0.00)
GTN/ FastGTN	APCPA, APAPA, APA	MAM, MDM, MDMDM	PAP, PSP
MP-GNN	APCPA, APAPA	MAM, MDM	PAP, PSP

Table 3.1 (top) shows the F_1 scores achieves by the different methods. As expected, all approaches achieve similar results, which are consistent with the ones observed in previous work [113]. Indeed, the number of relations is very limited (three for IMDB, four for DBLP and ACM) and, most importantly, no relations are shared among different node pair types, substantially restricting the number of candidate meta-paths. Still, MP-GNN achieves slightly better results, most likely thanks to its ability to select a minimal set of meta-paths, as shown in Table 3.1 (bottom).

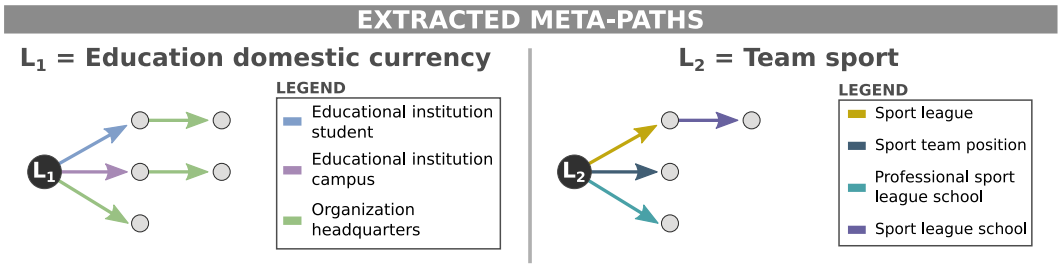


Figure 3.6: Examples of learned meta-paths on two node classification tasks

Q3: `MP-GNN` substantially outperforms competitors on real-world datasets with many relations The last set of experiments aims to evaluate `MP-GNN` in a complex real-world setting characterized by a large set of relations, as typical of general-purpose knowledge graphs. We thus designed a set of node-classification tasks over **FB15K-237** [94], which is a large knowledge graph derived from Freebase. Each entity in the graph is associated with a text description, that we transformed into a bag-of-words representation of length 100 (retaining the most frequent words in the dataset). We identified as target labels all many-to-one relations that have from 2 to 20 possible destination types (to avoid having classes with too few examples). Examples include gender, event type and a number of currency-related relations. See the Appendix for the statistics of the datasets.

Table 3.2 reports F_1 scores for the different methods. `GTN` and `FastGTN` have serious difficulties in learning reasonable models in all cases. Indeed, the unbalancing in the class distribution, combined with the large pool of candidate relations to learn from, drives them to boil down to majority class prediction in most cases. Despite the better performance of `RGCN`, `HGN`, and `R-HGNN`, they still exhibit substantially lower F1-scores compared to `MP-GNN`. Notably `MP-GNN` is surpassed only by `RGCN` and `R-HGNN` in the "event" and "team sport" classification tasks, respectively. Figure 3.6 shows some examples of extracted meta-paths for two different classification tasks, namely predicting the currency of domestic tuition fees in educational institutions and predicting the sport a team is playing. In the former case, extracted meta-paths lead to the headquarters of the organization delivering the educational program, which clearly correlate with the currency being used. In the latter case, meta-paths include the league where the team is playing, which again carries information about the sport being played. Note that in both cases, node features are crucial in leveraging meta-path information, as there are not enough examples to generalize via, e.g., specific headquarter or sport league name. Indeed, an ablation experiment excluding node feature information (the typical setting of meta-path mining approaches [87, 71, 59]), shows that none of the methods manages

to learn any sensible meta-path, always boiling down to learning majority class prediction rules (see Appendix A.3). For the same reasons, plain meta-path mining fails to extract sensible meta-paths, resulting in poor performance (see Appendix A.4 for the results using the popular PRA meta-path miner [59]).

Table 3.2: Many-relations dataset: F1-scores for the different node classification tasks on the FB15K-237 dataset. Results with standard deviations can be found in Table A.5 in Appendix. See Table A.1 in the Appendix for the meaning of the label acronyms.

Label	R-HGNN	HGN	RGCN	GTN	FastGTN	MP-GNN
PNC	0.72	0.68	0.74	0.33	0.33	**0.83**
EDC	0.6	0.75	0.71	0.12	0.12	**0.96**
EIC	0.63	0.65	0.73	0.12	0.12	**0.8**
ELC	0.47	0.74	0.72	0.12	0.15	**0.78**
FBC	0.45	0.48	0.42	0.14	0.14	**0.61**
GNC	0.8	0.74	0.82	0.19	0.19	**0.90**
OC	0.67	0.73	0.78	0.14	0.14	**0.93**
G	0.81	0.64	0.8	0.44	0.44	**0.84**
TS	**0.67**	0.53	0.62	0.09	0.09	0.63
E	0.89	0.8	**0.98**	0.07	0.07	0.96

Finally, to assess the computational efficiency of MP-GNN, we conducted a running time comparison, detailed in Appendix A.5. Results show that our approach is comparable with that of the competitors on the synthetic and few relation (IMDB, DBLP, ACM) datasets. On the freebase tasks, which have a larger set of candidate relations, our approach is more expensive than (most) competitors, but these have substantially lower performance in terms of F1, with the fastest approaches (GTN and FastGTN) completely failing to learn anything sensible.

3.3 Conclusion

In this work we introduced a novel approach inspired by information theoretic principles to effectively learn meta-paths and meta-path based multi-relational GNNs in settings characterized by a large number of candidate relations combined with arbitrarily rich node features. Our experimental evaluation confirms the potential of the approach in recovering correct (in synthetic tasks) and informative (in real-world tasks) meta-paths despite the large number of candidate relations, a setting where existing multi-relational GNNs struggle to learn meaningful models.

Future work includes generalizing the approach to account for counts or propor-

tions of meta-path realizations as relevant features, as well as more complex relational structures like meta-trees.

Chapter 4

A Self-Explainable GNN for Heterogeneous Graphs

This chapter extends the Meta-Path Graph Neural Network (`MP-GNN`)[34] framework introduced in Chapter 3, addressing limitations that arise when meta-path learning is applied to relational deep learning settings. While `MP-GNN` provides an effective and interpretable mechanism for identifying informative relational chains, it relies on an existential quantification assumption, whereby a single occurrence of a meta-path is sufficient to support a prediction. As illustrated in Figure 4.1, this assumption prevents the model from distinguishing between entities connected by few versus many instances of the same meta-path, and from exploiting aggregate information derived from relational patterns.

In relational databases and tabular-to-graph transformations, predictive targets often depend on statistics computed over multiple relational instances rather than on isolated relational witnesses. To overcome this limitation, the model introduced in this chapter generalizes `MP-GNN` by replacing existential aggregation with learnable, task-dependent statistics over meta-path occurrences, while preserving the interpretability and relational inductive biases of the original framework.

4.1 Definitions

This section presents some definitions that will be utilized in the rest of the paper.

Definition 1 (*Relational Database*)**.** A **relational database** $(\mathcal{T}, \mathcal{L})$ consists of a collection of tables $\mathcal{T} = T_1, ...T_n$ and links between these tables $\mathcal{L} \subseteq T \times T$. Each table is a set $T = \{e_1, ..., e_n\}$ where the elements $e \in T$ are referred to as rows or entities. Each

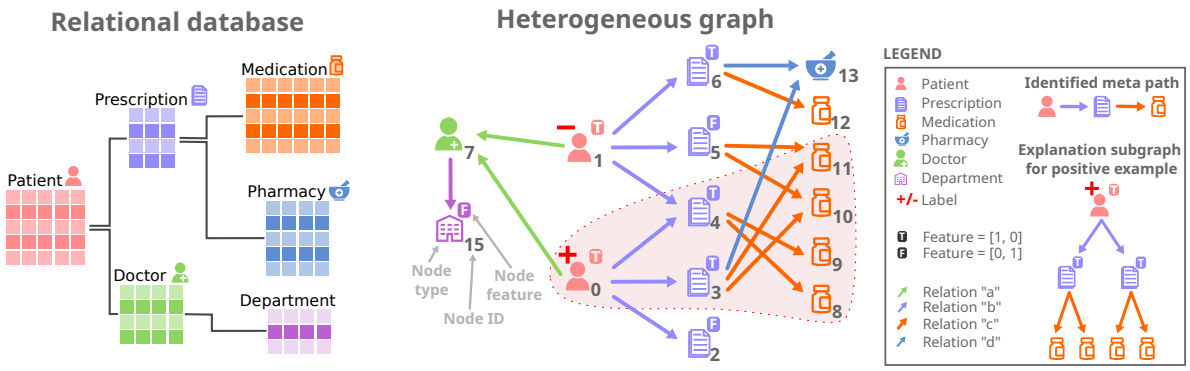


Figure 4.1: **Left**: Relational database schema for a medical domain. **Right**: Heterogeneous graph representation of (part of) the database. The highlighted subgraph shows a prototypical counts-of-counts pattern characterising positive patients, namely having at least two exempt prescriptions (represented by node feature T), each containing at least two medications. Existing heterogeneous GNNs struggle with these patterns as they need to learn a separate weight matrix for each edge type in the graph, while `MPS-GNN` is capable of learning the relevant meta path without any direct user supervision.

entity is a tuple $e = (\mathcal{P}_e, \mathcal{K}_e, a_e)$ where $\mathcal{P}_e$ is the **Primary Key** that uniquely identifies the entity e; $\mathcal{K}_e$ is the set of **Foreign Keys** corresponding to a primary key in other tables, thus connecting the tables; a_e corresponds to the **Attributes** of the entity e.

Definition 2 (*Heterogeneous graph*)**.** A **heterogeneous graph** is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X_\mathcal{V})$ where $\mathcal{V}$ is the set of nodes or entities, $\mathcal{E}$ is the set of directed edges (graphs induced by relational databases will be inherently directed) and $X_\mathcal{V}$ is a matrix of node attributes (with x_v being the attribute vector of node v). Each edge is represented as a triple (u, r, v), indicating that nodes u and v are connected via relation r (written as $u \xrightarrow{r} v$). We indicate the set of relations in the graph as $\mathcal{R}$.

For a node v and a relation r we denote with $\mathcal{N}_v^r$ the set of nodes that can be reached from v by following relation r. We refer to this set as r-neighbors.

From relational database to heterogeneous graph A relational database can be interpreted as an heterogeneous graph where row e becomes node v; attributes a_e become node features x_v; links $\mathcal{L}$ between entries of two tables are identified by the pair of primary $\mathcal{P}$ and foreign $\mathcal{K}$ keys in the two tables. Each pair of connected tables, originates a relation in the graph, specified by r

4.2 Methodology

For now we restrict attention to binary node classification problems. Our main problem is to construct meta-paths $\xrightarrow{r_1}\xrightarrow{r_2}\ldots\xrightarrow{r_L}$ that are predictive features for the class label. When considering a meta-path as a feature, we are thinking of possible numerical features that can be defined by collecting and aggregating information that is found along all occurrences of the meta-path in a concrete data graph, such as the count-of-count feature illustrated in Figure 4.1. We construct meta-paths following a strategy that is

- *Greedy*: a partially constructed meta-path $\xrightarrow{r_1}\xrightarrow{r_2}\ldots\xrightarrow{r_i}$ is extended by a next relation $\xrightarrow{r_{i+1}}$ without lookahead for possible completions $\xrightarrow{r_{i+2}}\ldots\xrightarrow{r_L}$. Similar as in [34], we try to estimate the *potential informativeness* of nodes reached by $\xrightarrow{r_{i+1}}$ by learned weights associated with the nodes. These weights represent putative features that can either be directly materialized as functions of the nodes' attributes, or that can be constructed as features of meta-path extensions starting at the node. An illustrative example of the meta-path construction process is provided in Section 3.1.1.

- *Local*: the meta-path construction step $\xrightarrow{r_i}\xrightarrow{r_{i+1}}$ is performed based only on local consideration of the nodes reached by r_i, and their r_{i+1} successors. The already constructed meta-path prefix $\xrightarrow{r_1}\xrightarrow{r_2}\ldots\xrightarrow{r_i}$ plays no explicit role in this step. We realize this locality by defining for each step a surrogate classification task for the nodes reached by r_i. The problem of extending the constructed meta-path prefix then translates into the problem of finding the first relation for a relevant meta-path solving the surrogate problem. The surrogate problems take the form

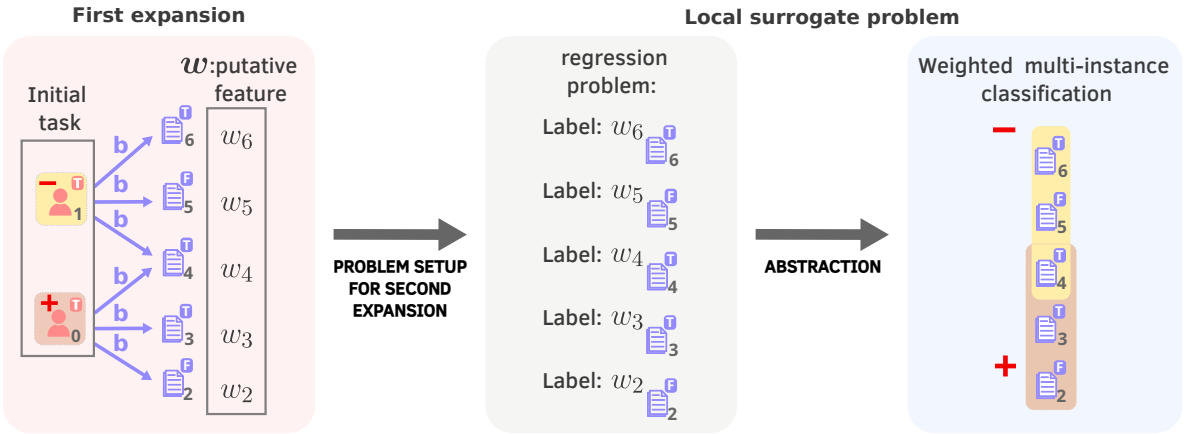


Figure 4.2: Outline of greedy local meta-path construction

of *weighted multi-instance classification* tasks.

Figure4.2 illustrates the high-level principles of our approach. Given an initial (binary) node classification task, a first relation is identified that could solve the task with the help of a putative node feature (weight) $\boldsymbol{w}$ on the successor nodes. Then a surrogate classification task is set up whose target is to materialize the putative feature as a feature computable from the data. This surrogate task takes the form of a weighted multi-instance classification task, which can be seen as an abstraction of a direct regression problem with target $\boldsymbol{w}$ (see Section 4.2.2).

Note that both the greedy and local properties mirror core principles of growing decision trees, which are built incrementally, adding one relation at a time based on solving local classification sub-tasks. The improvements in time complexity with respect to an algorithm that looks at every possible relations in the meta-path construction are detailed in section 4.2.2. In Section 4.2.1, we describe the *weighted multi-instance classification* task and its application in our scenario. Section 4.2.2 outlines the methodology for constructing meta-paths by scoring graph relations, including the examples based on Figure 4.1. In Section 4.2.3, we detail the GNN used in our setup, and Section 4.2.4 presents the complete framework of the proposed approach.

4.2.1 Weighted multi-instance classification

We consider a variant of multi-instance classification, where each instance consists of a bag $\mathcal{B}$ of nodes with a class label in $\{+, -\}$, and each node $v \in \mathcal{B}$ is assigned a weight $\alpha(v, \mathcal{B})$. We denote with $\mathcal{S}^+, \mathcal{S}^-$ (training) sets of positively and negatively labeled bags, respectively. The intention is to interpret the label of the bag as a function of element-level features, and that $\alpha(v, \mathcal{B}) \in \mathbb{R}$ represents a weight of the contribution of v's feature value to the class label of $\mathcal{B}$. This weight can be positive or negative, and its absolute value can be interpreted as a measure for the importance of node v in bag $\mathcal{B}$ (which may differ for different bags that v is an element of). See [36] for related generalizations of the standard multi-instance learning setting.

Our goal is to predict the bag label via discriminant functions of the form

$$F(\mathcal{B}) = \sum_{v \in \mathcal{B}} \alpha(v, \mathcal{B}) f(v), \tag{4.1}$$

where $f(v) \in \mathbb{R}$ is a learnable scalar node feature function. Specifically, we consider

functions that are parameterized by a relation r, and are of the form

$$f(v, r, \Theta, \boldsymbol{w}) = \begin{cases} \Theta^T x_v & \text{if } \mathcal{N}_v^r = \emptyset \\ \Theta^T x_v \sum_{u \in \mathcal{N}_v^r} w_u & \text{if } \mathcal{N}_v^r \neq \emptyset \end{cases} \tag{4.2}$$

where x_v denotes the attribute vector of v, Θ is a trainable parameter vector, and $\boldsymbol{w}$ is a vector of trainable weights $w_u \in [0, 1]$ assigned to v's r-neighbors. The node feature function is thus computed as a combination of v's own attributes and the putative feature w_u of its successors. When the node attributes x_v are not informative for solving the multi-instance classification task, then parameters Θ can be learned that make $\Theta^T x_v$ constant for all v, and thus this part of the node feature function becomes irrelevant (a suitable Θ exists e.g. under the mild assumption that x_v contains at least one categorical attribute [1] in a one-hot encoding: then $\Theta^T x_v = 1$ for Θ that is set to 1 for all entries corresponding to the one-hot encoding of the categorical attribute, and zero everywhere else). The factor $\sum_{u \in \mathcal{N}_v^r} w_u$ captures a dependence of $f(v)$ on the r-neighborhood of v. When the r-neighborhood is non-informative because neither the number nor the identity of nodes $u \in \mathcal{N}_v^r$ has discriminative value, then the $\sum_{u \in \mathcal{N}_v^r} w_u$ factor can be made irrelevant by learning constant weights w_u.

Ideally, the discriminant function separates the classes in the sense that for any pair $\mathcal{B}^+ \in \mathcal{S}^+, \mathcal{B}^- \in \mathcal{S}^-$ we have $F(\mathcal{B}^+) > F(\mathcal{B}^-)$. We note that this problem would be trivially solvable e.g. in the case where every positive bag contains a node v that has an r-successor, which is not also an r-successor of some node in a negative bag. Then assigning a weight of 1 to all such r-successor nodes, and a weight of 0 to all other nodes, would separate the classes. In reality, the complex connectivity of relations will preclude such simple solutions, and perfect separation in general. We therefore use as our learning objective the relaxed loss function

$$L(r, \Theta, \boldsymbol{w}) = \sum_{\mathcal{B}^+ \in \mathcal{S}^+, \mathcal{B}^- \in \mathcal{S}^-} \sigma\Big(F(\mathcal{B}^-, r, \Theta, \boldsymbol{w}) - F(\mathcal{B}^+, r, \Theta, \boldsymbol{w})\Big), \tag{4.3}$$

where σ is the sigmoid function. In practice, given that the number of terms in the sum is quadratic in the number of training examples, we approximate (4.3) by a random sample of positive and negative bags.

[1] In a heterogeneous graph built from a relational database, this would be the table the node belongs to. If node representations are simply embeddings, one can achieve the same goal by concatenating a constant feature to the embedding.

4.2.2 Relation Scoring

The initial weighted multi-instance classification problem for our meta-path construction process is defined by letting each positive (negative) target node v form a one-element bag $\{v\}$ with the corresponding label, and weight $\alpha(v, \{v\}) = 1$. Denote with $\mathcal{S}_1^+, \mathcal{S}_1^-$ the sets of all initial positive and negative bags, respectively. At all iterations we select the relation that minimizes the loss (also referred to as the scoring function)

$$L(r) = \min_{\Theta, \boldsymbol{w}} L(r, \Theta, \boldsymbol{w}). \tag{4.4}$$

If all candidate relations fail to minimize the loss, i.e., optimizing $\Theta, \boldsymbol{w}$ does not lead to substantially lower loss than using random parameters (i.e., does not improve by at least 30%), then the meta-path construction terminates and the current meta-path is returned. Otherwise, the current meta-path is extended with the minimal loss relation r. At this point, the problem becomes capturing the putative node features $\boldsymbol{w}$ by actual features.

A possible approach would be to set this up as a node regression task with target values w_u. However, due to the often very large set of alternative optimal solutions $\boldsymbol{w}$ in the minimization (4.4), this would lead to a too restrictive task. Our goal is to approximate the whole space of possible regression tasks defined by alternative $\boldsymbol{w}$ as a single weighted multi-instance classification task. For this, let r_i denote the relation found to minimize (4.4) in iteration i with optimal parameters Θ_i. For each positive bag $\mathcal{B}^+ \in \mathcal{S}_i^+$ define the new bag

$$\mathcal{B}_{new}^+ = \cup_{v \in \mathcal{B}^+} \mathcal{N}_v^{r_i} \tag{4.5}$$

containing the r_i-children of the nodes in $\mathcal{B}^+$. Similarly for negative bags. For $u \in \mathcal{B}_{new}^+$ define the node weight by the following sum over all nodes $v \in \mathcal{B}^+$ that have u as an r_i-neighbor:

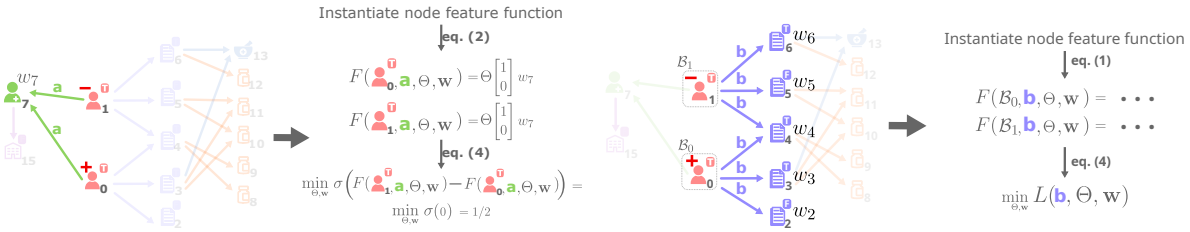
$$\alpha(u, \mathcal{B}_{new}^+) = \sum_{\{v \in \mathcal{B}^+ \,:\, u \in \mathcal{N}_v^{r_i}\}} \Theta_i^T x_v \alpha(v, \mathcal{B}^+). \tag{4.6}$$

This definition of the weights enables to essentially ignore in the setup of the multi-instance classification task for iteration $i+1$ those nodes u that are r_i-successors only of nodes v that did not play a major role in solving the task of the previous iteration – either because of a low absolute value of $\Theta_i^T x_v$, or because $\alpha(v, \mathcal{B}_i^+)$ already had a low absolute value.

The sets of all $\mathcal{B}_{new}^+ (\mathcal{B}_{new}^-)$ form the new training sets $\mathcal{S}_{i+1}^+ (\mathcal{S}_{i+1}^-)$.

Toy Example: To provide a clear understanding of the proposed approach, we illustrate it with the toy example of Figure 4.1, where patients are positive if, and only if, they have at least two exempt prescriptions, each containing at least two medications.

Initially, the approach evaluates the *potential informativeness* of two relations originating from patients: relation a and relation b. Figure 4.3a illustrates the scoring process for relation **a**. The first step involves instantiating the node feature functions for patient nodes 0 and 1 by applying Equation 4.2. Since both nodes share the same unique neighbor, the loss function 4.4 reduces to the constant $\frac{1}{2}$, which cannot be minimized by Θ and $\boldsymbol{w}$.



(a) Scoring relation a (b) Scoring relation b

Figure 4.3: Scoring the first two relations

Similarly, relation b is scored. The loss function 4.4 now becomes

$$\sigma(\Theta^T x_1(w_4 + w_5 + w_6) - \Theta^T x_0(w_2 + w_3 + w_4)),$$

where $x_0 = x_1 = (1, 0)^T$ is the attribute vector representing the T value in a one-hot encoding. This loss can be brought arbitrarily close to zero, e.g. by $w_6 \to 0$, $w_5 \to 0$, $w_3 \to 1$, $w_2 \to 1$, and $\Theta = (Z, 0)^T$ with $Z \gg 0$. Consequently, relation b scores higher than relation a and is selected for further extension. Note that the low loss of relation b is entirely due to its potential informativeness: the b neighborhoods of the nodes 0 and 1 are completely isomorphic, and therefore the meta path consisting of b alone provides no discriminative information.

To extend the meta path prefix b we set up the weighted multi-instance classification task for the second iteration. This gives us a positive bag $\mathcal{B}_2$ containing nodes $\{2, 3, 4\}$ and a negative bag $\mathcal{B}_3$ containing nodes $\{4, 5, 6\}$. Node weights for the bags are computed according to Equation (4.6), and here simply yield uniform weights $\alpha(k, \mathcal{B}_2) = Z$ for all $k \in \{2, 3, 4\}$, and $\alpha(k, \mathcal{B}_3) = Z$ for all $k \in \{4, 5, 6\}$. Note that node 4 has separate weights for the two bags it is part of.

For solving the new classification task we score the candidate relations c and d. The right part of Figure 4.4 illustrates the scoring of c. Due to the indistinguishable

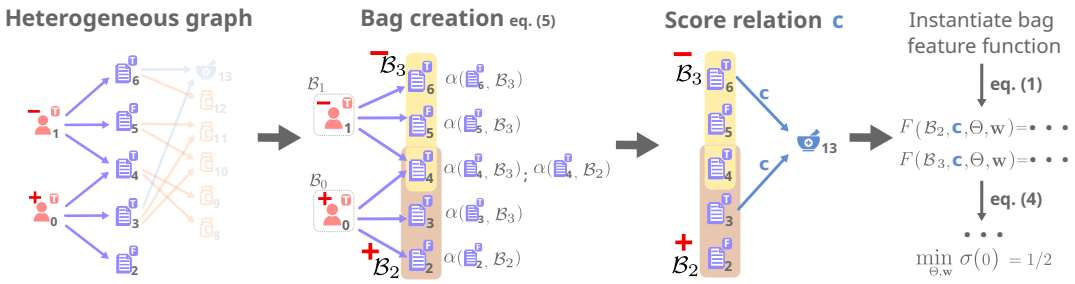


Figure 4.4: Bag generation and scoring of relation c.

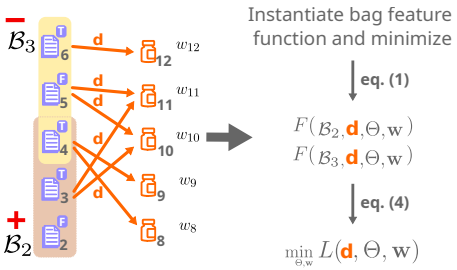


Figure 4.5: Scoring relation d.

structure of the c-neighborhood for bags $\mathcal{B}_2$ and $\mathcal{B}_3$ the loss function 4.4 now reduces again to the constant $1/2$, as in the scoring of relation a.

For scoring the relation d we obtain as the loss function 4.4

$$\sigma\left(Z\Theta^T\begin{bmatrix}1\\0\end{bmatrix}(w_{12}-w_{10}-w_{11})+Z\Theta^T\begin{bmatrix}0\\1\end{bmatrix}(w_{10}+w_{11}-1)\right)$$

(see details in Appendix B.1). This loss can be brought arbitrarily close to zero, e.g. by $w_{10}, w_{11} \to 1$, $w_{12} \to 0$ and $\Theta = (Z', 0)^T$ with $Z' \gg 0$. Thus, relation d is selected to extend the meta-path. The constructed meta-path $\xrightarrow{b}\xrightarrow{d}$ now is sufficient to discriminate between positive and negative examples. This is not directly visible at this step in the meta-path construction process, but will be found by training an `MPS-GNN` 2 on this meta-path, as detailed in sections 4.2.3 and 4.2.4. Check B.10 for a similar example with more complex node features

Time complexity analysis

The purpose of the scoring function is to iteratively construct the meta-path that best classifies the target nodes, avoiding the need to explore all possible paths. A naive search algorithm that simply tests all possible paths would have a polynomial complexity of $O(|\mathcal{R}|^L)$, where $|\mathcal{R}|$ is the number of relations in the graph, and L is the maximum length

of the meta-path we aim to find. By leveraging the scoring function, the complexity is reduced to linear $O(|\mathcal{R}| \cdot L)$. This improvement is achieved because, at each step, one relationship is added to the meta-path under construction, and the subsequent search builds upon it without reconsidering the other relationships scored in the same iteration.

4.2.3 MPS-GNN

Similarly to MP-GNN, in the MPS-GNN framework a meta-path $r_1, ..., r_L$ defines a multi-relational GNN with L layers. In this setup, each layer of the network corresponds to a specific relation in the meta-path: the initial layer is linked to the final relation r_L, progressing sequentially until the last layer, which corresponds to r_1. Our forward model then takes the form:

$$h_v^{(l+1)} = \sigma\Big(W_0^{(l)} h_v^{(l)} + W_{neigh}^{(l)} \sum_{u \in \mathcal{N}_v^{r_{L-l}}} h_u^{(l)} + W_1^{(l)} h_v^{(0)}\Big) \tag{4.7}$$

where $\mathcal{N}_v^{r_{L-l}}$ are neighbours of node v under relation r_{L-l}, $h_v^{(l)}$ is the embedding of node v in layer l, $h_v^{(0)} = x_v$ is the feature vector of node v, while $W_0^{(l)}$, $W_{neigh}^{(l)}$ and $W_1^{(l)}$ are learnable parameter vectors. Note that the latter term $W_1^{(l)} h_v^{(0)}$, which is missing the original MP-GNN [34], represents a skip connection between the input and the $l+1$ layer. This allows the network to access the node attributes at each layer, which is essential for enabling the MPS-GNN to capture node features corresponding to the $\Theta \cdot x_v$ terms in the meta-path construction as shown in the ablation study in B.2). As in [34], the definition can be generalized to multiple meta-paths by concatenating the embeddings obtained from each of them using $h_v^{(l+1)} = \Big\|_{k=1}^{K} h_{(v,k)}^{(l+1)}$ where K is the number of meta-paths, $h_{(v,k)}^{(l+1)}$ is the embedding of node v according to meta-path k and $\|$ is the concatenation operator.

4.2.4 The overall algorithm

Algorithm 2 outlines the whole MPS-GNN procedure for the single meta-path case (in practice, a K beam search is used and multiple meta-paths are learned). The algorithm takes as input a graph G, the set of available relations $\mathcal{R}$, an initial set of binary node labels $\mathcal{Y}$, a maximal meta-path length L_{MAX} and a stopping criteria value η. It initializes the targets $\mathcal{S}$ with a set of singletons (one per labeled node) and their alpha values (collectively indicated by $\mathcal{A}$) to 1.

Algorithm 2 MPS-GNN Learning

procedure learnMPS-GNN($G, \mathcal{R}, \mathcal{Y}, L_{MAX}, \eta$)
  $mp^* \leftarrow [\,]$, $F_1^* \leftarrow 0$, $\mathcal{S} \leftarrow \mathcal{Y}$, $\mathcal{A} \leftarrow 1$
  while $|mp| < L_{MAX}$ **do**
    $r^* \leftarrow \arg\min_r L(r)$ ▷ Eq. 4.4
    if $\min_{r \in \mathcal{R}} L(r) \geq \eta L_{\text{init}}(r)$ **then**
      return mp^*
    end if
    $mp \leftarrow mp, r^*$
    $gnn \leftarrow$ TRAIN(`MPS-GNN`$(mp), G, \mathcal{Y})$
    $F_1 \leftarrow$ TEST(gnn)
    if $F_1 > F_1^*$ **then**
      $mp^* \leftarrow mp,\ F_1^* \leftarrow F_1$
    end if
    $\mathcal{A}, \mathcal{S} \leftarrow$ NEW-TARGETS$(\mathcal{S}, r^*)$ ▷ Eqs. 4.5, 4.6
  end while
  return mp^*
end procedure

At each iteration, the scoring function identifies the relation minimizing Eq. 4.4 and appends it to the meta-path mp. If no relation improves the loss by a factor η, the algorithm stops and returns the meta-path mp^* constructed so far. The algorithm then evaluates mp by training `MPS-GNN` on node labels, and tests it using the F_1 score (computed on a validation set, omitted for brevity), which reflects the meta-path's performance *when embedded* in an `MPS-GNN`, as opposed to its potential informativeness measured by the scoring function. For training `MPS-GNN`, node embeddings are updated using Eq. 4.7.

The algorithm keeps track of the best meta-path prefix together to its F_1 score, and creates new target bags and α values as specified in Equations 4.5 and 4.6 for the next relation scoring round. The algorithm ends when the maximum meta-path length L_{MAX} is reached The algorithm is described for simplicity in the context of a single meta-path and without the additional stopping criteria described in section 4.2.2. In practice, however, the implementation employs a beam search over the meta-path space, selecting the top K best-scoring meta-paths concatenating their embedding for the final node representation as detailed at the end of section 4.2.3. MPS-GNN scales *linearly* in the number of relations and nodes, thanks to its incremental construction of meta-paths. The value chosen for L_{MAX} in the experiment was 4, while η was set to 0.7. The method is however quite robust to variations in η, as the identified meta-paths remained unchanged for a wide range of values in our experiments.

4.2.5 MPS-GNN is a self-explainable model

Self-explainable GNNs [51, 21] are a class of GNN models that aims to achieve explainability by-design. At a high level, these models can be seen as composed of two modules: a *detector* that extracts a class-discriminative subgraph, and a *classifier* that outputs a prediction based on the extracted subgraph. By relying on meta-paths for its predictions, MPS-GNN is a self-explainable GNN model. The scoring function serves as the detector, identifying relevant meta-paths, while the network built using them acts as the classifier. By construction, the network can only access the meta-path induced subgraph, making it strictly sufficient by construction (no changes outside the meta-path induced graph affect the prediction). An analysis of the faithfulness of MPS-GNN's explanations is provided in Section 5.5. While most approaches for GNN explainability focus on the topological aspect, by identifying (hard or soft) subgraphs as explanations, determining which node features contribute to the prediction is also relevant from an interpretability perspective. While interpretability at the node feature level could be encouraged by introducing sparsifying norms for the learnable parameter vectors in Eq. 4.7, guaranteeing a fully transparent processing of node features in the layerwise node embedding computation is beyond the scope of current GNN-based architectures.

4.2.6 Comparison with MP-GNN

The novelty of our approach compared to MP-GNN lies in our model's ability to learn meta-paths that are relevant to the target node class, not merely based on their existence but on statistical measures related to their occurrences. In practice the key difference between MPS-GNN and MP-GNN lies in the way in which relations are scored. Specifically MP-GNN predicts the class label y of a node v, in the first iteration, with $\tilde{y}_v^r = \Theta^T x_v \max_{u \in \mathcal{N}_v^r} w_u$ where the max aggregation of the neighbours is used, indicating that a candidate relation r is informative for the label of a node v if at least one of the neighbors $\mathcal{N}_v^r$ of v according to r belongs to the ground-truth meta-path, and v has the right features. To leverage meta-path occurrences, MPS-GNN employs a sum aggregation strategy, as detailed in Eq. 4.2, enabling the counting of their occurrences.

In subsequent iterations, MP-GNN also employs a bag creation process, where positive bags are formed by including neighbors of positive nodes that are predicted to be positive at least once across multiple prediction procedures. This formulation ensures that the node responsible for the positive label, and thus aligned with the correct meta-path, remains in a positive bag. Unlike this approach, however, we must ensure that all neighbor nodes involved in multiple occurrences of the *correct* meta-path are included

in a positive bag. As detailed in eq. 4.5, the neighbors of positive nodes are therefore placed in a positive bag without the need of any additional prediction step.

The enhancements introduced in `MPS-GNN`, compared to `MP-GNN`, enable it to handle predictions over relational databases where the class label may depend on statistical measures derived from meta-path occurrences.

4.3 Experimental Results

Our experimental evaluation seeks to address the following research questions:

Q1 Can `MPS-GNN` recover the correct meta-path when increasing the setting complexity?

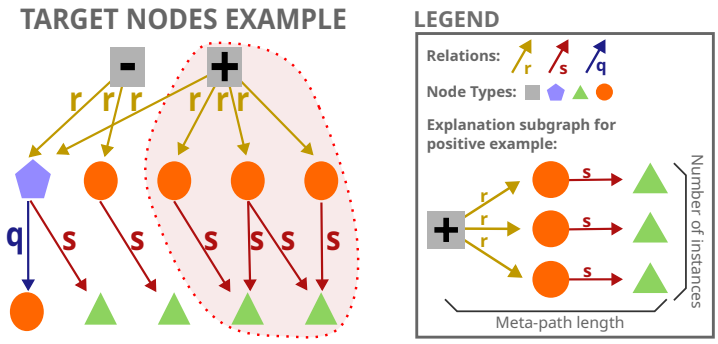
Q2 Does `MPS-GNN` outperform existing approaches in tasks over real world relational databases?

Q3 Is `MPS-GNN` self-explainable?

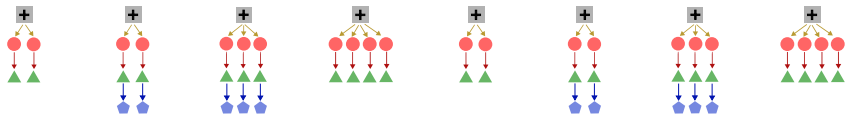
We compared `MPS-GNN` with approaches that don't require predefined meta-paths, handle numerous relations, and incorporate node features in learning. The identified competitors include: `MLP`, to test the sufficiency of target node features alone; `GCN` [53], a baseline non-relational model; `RGCN` [84], extending `GCN` for multi-relational graphs, with distinct parameters for each edge type; `HGN` [47], a heterogeneous GNN model extending `GAT` for multiple relations; `GTN` [113], which transforms input graphs into different meta-path graphs where node representations are learned; `Fast-GTN` [113], an optimized `GTN` variant; `R-HGNN` [111], a relation-aware GNN using cross-relation message passing; and `MP-GNN` [34], the original meta-path GNN supporting only existentially quantified meta-paths.

We implemented our model using PyTorch Geometric, and used the competitors' code from their respective papers for comparison. For training `MPS-GNN`, we used a 70/20/10 split for training, validation, and testing, respectively, and reported the test results for the model selected based on its validation performance. For the sake of comparison with [34], we set the maximum meta-path length to 4 and the beam size to 3. We employed F_1 as evaluation metric to account for the unbalancing in many of the datasets. Hyperparameters of competitors and `MPS-GNN` can be found in Table B.6 in the Appendix.

Q1: `MPS-GNN` consistently identifies the correct meta-path in count based synthetic scenarios In order to address the first research question, we designed a sequence of synthetic node classification scenarios where the correct structure to be learnt is known. In each scenario, a node is labelled as positive if it is the starting point of at least c occurrences of a given meta-path of length l, and negative otherwise. Crucially, existential quantification of meta-paths (as modelled by `MP-GNN` [34]) is insufficient here, as nodes which are starting points of less than c meta-path occurrences are labelled as negatives. We designed scenarios of increasing complexity by changing the length of the ground-truth meta-path l, the number of occurrences c, and the overall number of relations r in the dataset. See Figure 4.6 for the statistics of the different scenarios (left), and for a prototypical example for $l = 2$ and $c = 3$ (right).



Correct Pattern



	S1	S2	S3	S4	S5	S6	S7	S8
$\|\mathcal{R}\|$	5	5	5	5	10	10	10	10
c	2	2	3	4	2	2	3	4
l	2	3	3	2	2	3	3	2

Figure 4.6: **(Top)** Sample scenario. Nodes are labeled as positive if and only if they are the starting point of at least $c = 3$ instances of the $l = 2$ meta-path "*grey node* $\xrightarrow{r}$ *orange node* $\xrightarrow{s}$ *green node*". **(Bottom)** Statistics of synthetic datasets, with $|\mathcal{R}|$ total number of relations, c number of (correct) meta-path instances in positive nodes, and l meta-path length. On top of the table, the explanation subgraph for each dataset.

Table 4.1 shows the F_1 score of each model for an increasing complexity of the classification scenario. Results clearly show that this experimental setting is challenging for existing solutions. While the poor performance of `MLP`, which completely ignores the topological structure, and `GCN`, which ignores the difference between relations, are

expected, solutions specifically conceived for heterogeneous networks also struggle with these datasets. Models like `R-HGNN`, `HGN`, `GTN`, and `Fast-GTN`, despite accounting for different relations in the graph, are affected by both the imbalance between positive and negative target nodes and the limited number of instances of neighbors of a certain type. `RGCN` and `MP-GNN` achieve better performance but are still sub-optimal. The former, like other relational methods, takes into account the diversity of relations in the graph but still uses all of them, thus struggling to single out the relevant portion of the graph. `MP-GNN`, on the other hand, suffers from its existential quantification assumption, and fails to find the correct meta-path in all scenarios. Conversely, `MPS-GNN` manages to achieve nearly-optimal performance in all scenarios, substantially outperforming all existing strategies[2]. Note that the lookahead capabilities of the scoring function are crucial to the effectiveness of `MPS-GNN`. Appendix B.7 shows how replacing the scoring function with a simple greedy approach leads to failure in learning the correct meta-path. These results allow us to answer the first research question in the affirmative.

Table 4.1: F_1 metric with standard deviations for synthetic datasets

	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
`MLP`	0.46(±0.00)	0.44(±0.00)	0.48(±0.00)	0.47(±0.00)	0.44(±0.00)	0.51(±0.00)	0.45(±0.00)	0.47(±0.00)
`GCN`	0.46(±0.00)	0.46(±0.02)	0.48(±0.03)	0.52(±0.05)	0.44(±0.00)	0.48(±0.00)	0.46(±0.00)	0.48(±0.00)
`RGCN`	0.78(±0.02)	0.87(±0.03)	0.86(±0.03)	0.81(±0.02)	0.86(±0.03)	0.77(±0.01)	0.91(±0.00)	0.79(±0.01)
`R-HGNN`	0.50(±0.00)	0.44(±0.03)	0.47(±0.01)	0.47(±0.04)	0.53(±0.00)	0.48(±0.01)	0.46(±0.02)	0.48(±0.02)
`HGN`	0.45(±0.00)	0.46(±0.00)	0.50(±0.03)	0.46(±0.00)	0.46(±0.00)	0.48(±0.00)	0.45(±0.03)	0.40(±0.12)
`GTN`	0.46(±0.00)	0.52(±0.00)	0.49(±0.00)	0.48(±0.00)	0.44(±0.00)	0.47(±0.00)	0.49(±0.00)	0.47(±0.00)
`Fast-GTN`	0.46(±0.00)	0.48(±0.00)	0.51(±0.00)	0.49(±0.00)	0.44(±0.00)	0.46(±0.00)	0.53(±0.00)	0.47(±0.00)
`MP-GNN`	0.84(±0.09)	0.82(±0.13)	0.85(±0.10)	0.95(±0.02)	0.89(±0.06)	0.79(±0.03)	0.84(±0.06)	0.71(±0.21)
`MPS-GNN`	**0.98**(±0.00)	**0.98**(±0.01)	**0.99**(±0.10)	**0.98**(±0.00)	**0.99**(±0.00)	**0.93**(±0.10)	**0.94**(±0.00)	**0.95**(±0.00)

Q2: `MPS-GNN` surpasses competitors in real world databases, learning relevant meta-paths Our approach is particularly useful for predictive tasks in relational databases with multiple tables, where features for a target entity may involve statistics from related tables. To address the second research question, we thus focused on three relational databases with many tables: **EICU**, a medical database with 31 tables, where we predict patient stay duration in the eICU, modeled as binary node classification by thresholding duration at 20 hours to achieve two balanced classes.; **MONDIAL**, a geographic database where the task is predicting whether a country's religion is Christian; and **ErgastF1**, containing Formula 1 data, where the task is

[2]The residual error for `MPS-GNN` is due to the fact that despite relying on the correct meta-path, it occasionally leverages spurious instances where the relation sequence is correct but (some of) the node features are not.

Table 4.2: F_1-score with standard deviations of our method and competitors on real-world datasets.

	EICU	**MONDIAL**	**ErgastF1**
MLP	0.53(±0.02)	0.52(±0.00)	0.50(±0.00)
GCN	0.89(±0.00)	0.60(±0.02)	0.50(±0.01)
RGCN	0.70(±0.00)	0.53(±0.08)	0.57(±0.01)
R-HGNN	0.61(±0.00)	0.61(±0.01)	0.72(±0.02)
HGN	0.75(±0.00)	0.72(±0.01)	0.70(±0.04)
GTN	0.56(±0.02)	0.38(±0.01)	0.60(±0.01)
Fast-GTN	0.46(±0.03)	0.39(±0.04)	0.60(±0.03)
MP-GNN	0.87(±0.02)	0.36(±0.06)	0.71(±0.01)
MPS-GNN	**0.92**(±0.01)	**0.74**(±0.01)	**0.83**(±0.02)

predicting the winner of a race in a binary classification task where target nodes are represented by a combination of race and pilot. The databases were transformed into graphs as explained in Section 4.1; for disconnected components, we enhanced connectivity by clustering rows of auxiliary tables. Additional details for the datasets and the procedure are in the Appendix B.3.

Results Table 4.2 presents the F_1 scores of MPS-GNN and its competitors across three real-world databases, averaged over 5 runs with different seeds. The poor performance of MLP clearly indicates that using target node features only is insufficient for classification. Plain GCN, which treats the graph as homogeneous, performs well only on the **EICU** dataset, where node degree differences exist between positive and negative nodes. Heterogeneous GNN methods also struggle with these datasets, especially **MONDIAL**, where most approaches fail to outperform a simple MLP, and only one (HGN) manages to substantially outperform the non-heterogeneous baseline (GCN). MP-GNN does not provide the performance boost that was observed when applied to knowledge graphs [34], confirming our intuition that existential quantification of meta-path is insufficient when dealing with relational databases. On the other hand, MPS-GNN manages to substantially outperform all competitors, thanks to its ability to identify meta-paths that are *informative thanks to the statistics that can be computed over their realizations*, as shown in the following. It is worth noting that this result is achieved with one/two orders of magnitude fewer parameters than the runner-ups, namely HGN and R-HGNN. See Table B.7 in the Appendix for the details. Additionally, Table B.8 in the Appendix shows that MPS-GNN has competitive execution times with respect to other heterogeneous GNN approaches, thanks to its ability to focus training on relevant

meta-path induced subgraphs.

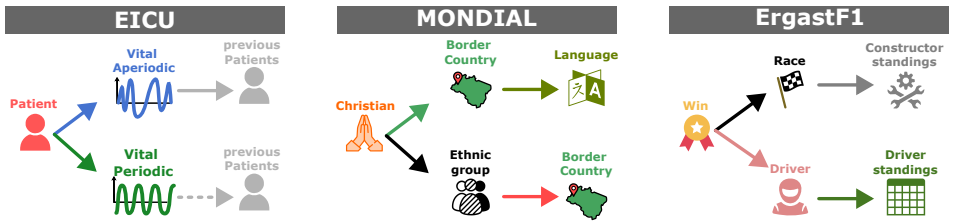


Figure 4.7: Extracted meta-paths for the three real world datasets.

Identified Meta-Paths Figure 4.7 shows the meta-paths extracted by `MPS-GNN` in the three real world datasets, which clearly convey relevant features for the respective task. For **EICU** (left), meta-paths correlate the patient's length of stay (predictive task) with information on patients with similar periodic (top) and aperiodic (bottom) vital signs. For **MONDIAL** (middle), Christianity is predicted collecting information about the language of border countries (top), and the ethnic group of the country and its neighbouring countries. Finally, in **ErgastF1** the winner is predicted via meta-paths collecting information about the constructor (top) and driver (top) standings.

Finally, in Appendix B.8 we present an experimental evaluation where `MPS-GNN` is adapted to deal with temporal databases and tasks [79], showing how it outperforms its competitors also in this context. Summing up, these results enable us to confidently answer Q2 in the affirmative.

Q3: `MPS-GNN` is a self-explainable method To address the third research question, we assessed the faithfulness of the extracted meta-paths. The meta-paths identified by the model from the graph are evaluated based on the complementary metrics of sufficiency and necessity. High sufficiency implies that changing the complement to the explanation (leaving the explanation unchanged) should not affect the model's output. High necessity implies that altering the explanation itself (leaving the complement unchanged) should result in a change in the model's output. It is easy to show that our approach is inherently sufficient. Indeed, the computational graph of `MPS-GNN` consists solely of the subgraph containing the occurrences of the identified meta-paths. Necessity, on the other hand, is calculated as a distance metric, measuring the difference in predicted probabilities between the original predictions and those obtained after masking parts of the explanation (i.e. deleting some instances of meta-paths). Defined as $\text{NEC} = \frac{1}{N}\sum_{v=1}^{N}(p_v(\mathcal{G}) - p_v(\mathcal{G}'))$ where v is a target node, $\mathcal{G}$ is the original graph and p_v denotes the probability associated with the predicted class. $\mathcal{G}'$ is obtained by

removing certain meta-path occurrences (i.e. randomly removing some branches of the identified meta-paths) and $p_v(\mathcal{G}')$ is the probability associated at the class predicted with p_v. Since `MPS-GNN` utilizes just the explanation for making prediction one should expect that removing some part of the explanation, has as effect a decrease in prediction F_1.

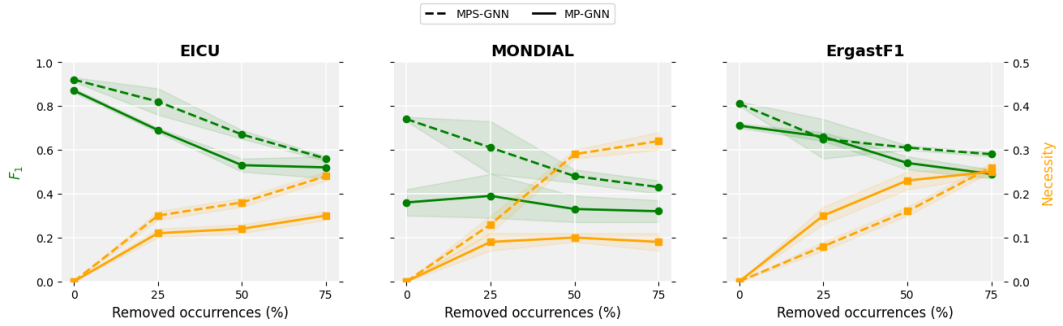


Figure 4.8: Changes to F_1 and posterior probability difference (necessity) when removing 25%, 50%, and 75% of the learned meta-path occurrences for the real-world datasets with `MP-GNN`, dashed line, and `MP-GNN`, solid line.

Figure 4.8 illustrates the effects of removing 25%, 50%, and 75% of the meta-path occurrences in terms of changes in F_1 and necessity between original and modified graphs [6]. In addition to the results for `MPS-GNN` (dashed line), the figure includes the results for `MP-GNN` (solid line), which is also a self-explainable GNN model according to the reasoning in Section 4.2.5. In all datasets, there's a noticeable decline in F_1 performance for `MP-GNN` and a steep increase in probability difference, suggesting that the learned meta-paths are also necessary. These results clearly indicate the faithfulness of the explanations of `MPS-GNN`. `MP-GNN` has a similar behaviour, albeit with lower F_1 with respect to `MPS-GNN` because of its lower expressivity. The only exception is the **MONDIAL** dataset, where `MP-GNN` fails to learn any relevant pattern, resulting in a very low F_1 score that remains approximately constant when removing meta-path occurrences.

4.4 Conclusion

We introduced a novel approach to identify relevant meta-paths of relations for node classification tasks in heterogeneous graphs with a potentially large number of different relations, notably graphs derived from relational databases. Compared to earlier work, our approach does not require user supervision and learns meta-paths for predictive

features defined by aggregate statistics over meta-path occurrences. The explainability of our method is particularly beneficial for sensitive domains like medical or financial data, where it helps address fairness concerns by providing insights into predictions. Experiments demonstrate advantages in accuracy and explainability.

Limitations and future works At this stage, our method is tailored for binary node classification but can be extended to multiclass classification using standard one-vs-all strategies. However, this approach is inefficient, as it requires repeating the entire process for each class. The underlying principles of our local greedy meta-path construction based on scoring potential informativeness of relations directly applies also in the multiclass case. What would be needed in a multiclass adaptation of the approach is to replace the scalar functions (4.1),(4.2), and scalar node weights w_u, with vector-valued versions, and to modify the loss function (4.3) accordingly. Similarly, node regression is currently not supported and would require a modification to the way relations are scored. Another limitation of the scoring function is its reliance on a well-connected graph structure; when target nodes have neighbors that are not connected to other target nodes, the scoring function requires a preprocessing step to create supernodes within the neighborhood, as implemented in certain real-world scenarios.
An interesting direction for future work involves incorporating temporal information into the process of learning optimal meta-paths. By doing so, the model would be capable of accounting for the temporal dimension, enabling it to better capture time-dependent relationships and dynamics within the graph.

Chapter 5

Learning with Missing Node Features: Benchmarks and GNNmim

Relational databases are a primary source of real-world graph-structured data and are well known to exhibit systematic patterns of missing features. Recent benchmarks for relational deep learning, such as RelBench [79], highlight missing node features as a defining characteristic of realistic relational learning scenarios, underscoring the practical relevance of learning under incomplete information.

When relational databases are represented as graphs, this missingness typically manifests at the level of node attributes, while the relational structure induced by primary and foreign keys remains largely intact. Handling missing features in such settings is therefore a fundamental challenge for graph neural networks. However, addressing missingness in multi-relational graphs introduces an additional layer of complexity, as different node types may follow different feature schemas and missingness patterns, and relational dependencies may interact with the missingness mechanism itself.

For this reason, before tackling missingness in fully multi-relational settings, it is essential to study its effects in the simpler yet nontrivial case of homogeneous graphs. Homogeneous graphs provide a controlled setting in which the interaction between graph structure, message passing, and missing node features can be isolated and systematically analyzed. Despite their apparent simplicity, recent empirical evidence suggests that even in homogeneous graphs, missing features can lead to unexpected behaviors and performance degradation in standard GNN architectures.

In this chapter, we study learning with missing node features in homogeneous graph neural networks as a first step toward understanding missingness in relational graph learning. Rather than introducing new imputation schemes, we systematically analyze

how standard GNN architectures behave when node features are partially observed and missingness is handled through commonly adopted heuristics, such as zero-filling or feature masking. Through controlled experimental settings and theoretical considerations, we identify several non-trivial behaviors that arise from the interaction between graph structure, message passing, and incomplete feature information. In particular, we show that graph connectivity and neighborhood aggregation can implicitly compensate for missing features in some regimes, while in others they amplify spurious correlations or lead to degenerate representations. Based on this analysis, we evaluate simple yet principled modeling choices that make the treatment of missing features explicit within the message passing process, without introducing additional supervision or complex inference procedures. Extensive empirical results demonstrate that explicitly accounting for missingness leads to more stable training dynamics and improved robustness across different missingness patterns.

The insights obtained in this homogeneous setting provide a clear characterization of the limitations of existing GNN practices under incomplete information and establish a foundation for extending missingness-aware graph learning to multi-relational settings, which we identify as a natural direction for future work.

5.1 Introduction

Learning with missing features is a pervasive and often unavoidable challenge in many real-world machine learning applications, including healthcare [15, 72], IoT sensor networks [30, 77, 2], and recommender systems [67, 45, 68]. As GNNs are increasingly applied in such domains, the problem of missing node features has received growing attention in the recent literature [96, 114, 80, 42, 92, 27, 95].

A wide range of approaches has been proposed, ranging from simple imputation strategies [110] to architectures that jointly impute missing values and perform prediction during training [42]. However, despite this growing body of work, relatively little attention has been paid to the validity of the evaluation protocols used to assess robustness to missing features. In particular, two aspects remain largely underexplored: the choice of datasets adopted for evaluation, and the missingness mechanisms used to generate incomplete features.

Regarding the former, most existing studies rely on standard node classification benchmarks with extremely sparse feature representations, such as bag-of-words encodings [108]. In such settings, the vast majority of feature entries are already zero, raising the question of whether robustness to missing features can be meaningfully assessed

when informative signal is inherently limited. In this chapter, we provide theoretical and empirical evidence showing that, in highly sparse feature regimes, additional missingness has a limited effect on predictive performance except at extreme missing rates, calling into question the adequacy of these benchmarks for studying feature missingness.

A second major limitation concerns the missingness mechanisms considered during evaluation. Most prior work focuses exclusively on Missing Completely At Random (MCAR) mechanisms [81, 64], where feature deletion is independent of the data. In practice, however, missingness is often related to feature values or prediction targets [18, 44, 55], corresponding to Missing Not At Random (MNAR) settings [81]. Moreover, existing evaluation protocols typically assume that the missingness mechanism is identical at training and test time, an assumption that is frequently violated in real-world deployment scenarios.

To address these limitations, this chapter adopts a more principled evaluation perspective. We consider datasets with dense, semantically meaningful node features and design evaluation protocols that include representative instances of MCAR and MNAR mechanisms, as well as train–test distribution shifts in the missingness process. Within this framework, we analyze the behavior of standard GNN architectures under incomplete feature information and study simple modeling choices that explicitly expose missingness to the message passing process.

5.2 Learning from Incomplete Graph Data

We consider an attributed graph $G = (V, E, \mathbf{X}, \mathbf{Y})$, where $V = \{1, \ldots, n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges represented by the adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the node feature matrix with entry X_{ij} denoting feature j of node i, and $\mathbf{Y} \in \mathcal{Y}^n$ is the vector of node labels.

When data is incomplete, some entries of $\mathbf{X}$ are unobserved. Let $\mathbf{M} \in \{0,1\}^{n \times d}$ be the missingness indicator matrix that has $M_{ij} = 1$ if x_{ij} is missing and 0 otherwise. In our setting, the missingness indicator matrix $\mathbf{M}$ is directly and deterministically constructed from the observed dataset. Missing values are explicitly marked in the raw data, so the mask $\mathbf{M}$ is uniquely defined and contains no uncertainty. Let $\mathbf{X}^{obs}$ be the elements of $\mathbf{X}$ for which $M_{ij} = 0$, and $\mathbf{X}^{miss}$ the elements for which $M_{ij} = 1$. The observed data from which we learn then can be written as $\mathbf{X}^{obs}, \mathbf{Y}, \mathbf{M}$. We note that we here make the assumption that $\mathbf{Y}$ is fully observed in the (training) data, and that there is no uncertainty about the graph structure E. The distribution of the data then

can be parameterized as

$$P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{X}^{obs}, \mathbf{Y}, \mathbf{M}) = \int_{\mathbf{X}^{miss}} P_{\boldsymbol{\theta}}(\mathbf{X}) P_{\boldsymbol{\gamma}}(\mathbf{Y}|\mathbf{X}) P_{\boldsymbol{\lambda}}(\mathbf{M}|\mathbf{X}, \mathbf{Y}), \tag{5.1}$$

where $\mathbf{X} = \mathbf{X}^{obs} \cup \mathbf{X}^{miss}$, $P_{\boldsymbol{\theta}}$ is the node feature distribution, $P_{\boldsymbol{\gamma}}$ is the conditional label distribution, and $P_{\boldsymbol{\lambda}}$ represents the *missingness mechanism*. Though not explicitly reflected in the notation, all these distributions will usually depend on the underlying graph structure, which will typically induce dependencies among the rows of $\mathbf{X}$, and among the elements of $\mathbf{Y}$.

A GNN for node classification with complete feature data is a model $P_{\boldsymbol{\gamma}}(\mathbf{Y}|\mathbf{X})$ with $\boldsymbol{\gamma}$ the weights of the GNN. For classification with incomplete data we need to learn the conditional model

$$P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{Y}|\mathbf{X}^{obs}, \mathbf{M}) = \int_{\mathbf{X}^{miss}} P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{M}) P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{X}^{miss}|\mathbf{X}^{obs}, \boldsymbol{M}). \tag{5.2}$$

The classical *missing (completely) at random (M(C)AR)* assumptions [81] simplify this problem. The original M(C)AR assumptions have been formulated in the context of estimating the parameter of a generative distribution. It has been observed that more specialized variations of the original definitions can be more pertinent in the context of classification [23, 38]. In the following we give formulations of M(C)AR for classification that provide the foundations for our theoretical analysis.

Definition 3. The joint distribution $P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}$ is *feature*-MAR, if

$$P_{\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{M}|\mathbf{X}^{miss}, \mathbf{X}^{obs}) = P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{M}|\mathbf{X}^{obs}). \tag{5.3}$$

It is *label*-MAR if

$$P_{\boldsymbol{\lambda}}(\mathbf{M}|\boldsymbol{X}, \boldsymbol{Y}) = P_{\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{M}|\boldsymbol{X}). \tag{5.4}$$

The distribution is MCAR, if

$$P_{\boldsymbol{\lambda}}(\mathbf{M}|\boldsymbol{X}, \boldsymbol{Y}) = P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{M}). \tag{5.5}$$

In (5.3)-(5.5) all probability functions are indexed with the parameters they actually depend on. Note, for example, that the conditional of $\mathbf{M}$ given $\mathbf{X}$ requires marginalization over $\mathbf{Y}$, and thereby also depends on the parameter $\boldsymbol{\gamma}$. MCAR implies both feature- and label-MAR.

The simplest realization of an MCAR mechanism is *uniform missingness (U-MCAR)*

in which entries of $\mathbf{X}$ are independently missing with a fixed missingness probability μ. This can be generalized by defining a missingness probability matrix $\boldsymbol{\mu} \in [0,1]^{n \times d}$ specifying potentially different missingness probabilities for different entries of $\mathbf{X}$.

MAR assumptions allow us to eliminate the missingness model $P_{\boldsymbol{\lambda}}$ from (5.2). The following proposition states this classical *ignorability* result in a version most suitable in our context.

Theorem 1. If $P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}$ is feature-MAR and label-MAR, then (5.2) simplifies to

$$\int_{\mathbf{X}^{miss}} P_{\boldsymbol{\gamma}}(\boldsymbol{Y}|\boldsymbol{X}) P_{\boldsymbol{\theta}}(\mathbf{X}^{miss}|\mathbf{X}^{obs}). \tag{5.6}$$

Intuition. Under feature-MAR and label-MAR, the missingness pattern carries no predictive information. The learning problem reduces to the usual classification task with imputed features, meaning that methods explicitly modeling the missingness mask do not gain theoretical advantage in this regime.

The proof is straightforward by rewriting the two factors on the right of (5.2) using Bayes's rule, and plugging in (5.3) and (5.4). Formulation (5.6) still poses two major challenges: it requires a feature distribution model $P_{\boldsymbol{\theta}}$ when in reality we only are interested in the conditional model $P_{\boldsymbol{\gamma}}$, and the integration over $\mathbf{X}^{miss}$ is usually intractable [48]. The simplest approach to address these problems is to approximate the integral (5.6) by evaluating $P_{\boldsymbol{\gamma}}(\boldsymbol{Y}|\boldsymbol{X})$ at a single imputed value $\mathbf{X} = \mathit{impute}(\mathbf{X}^{miss})$ [82]. This does not require an explicit model for $P_{\boldsymbol{\theta}}$, but relies on the implicit assumption that the imputed value $\mathit{impute}(\mathbf{X}^{miss})$ has high probability under $P_{\boldsymbol{\theta}}$. A simple example is *mean-imputation*, in which missing values of a given feature are filled with the mean of that feature; we will refer to this approach combined with a standard GNN as `GNNmi` [110]. In addition, we also consider *zero-imputation*, where missing entries are replaced with zeros (`GNNzero`), and *median-imputation*, where they are filled with the feature median (`GNNmedian`). Similarly, `PCFI` [96] does not require an explicit model for $P_{\boldsymbol{\theta}}$; it introduces a confidence-guided imputation scheme where pseudo-confidence is derived from the shortest-path distance to observed features, and combines channel-wise diffusion with inter-channel propagation to recover a single estimate of $\mathbf{X}$. `GOODIE` [114] approximates the integral in (5.6) using a combination of label propagation and `FP` [80], which propagates features by minimizing a Dirichlet energy function, whereas `FairAC` [42] does so by aggregating, via an attention mechanism, the representations from neighbors of nodes with missing features.

Other methods explicitly model $P_{\boldsymbol{\theta}}$. The `GCNmf` approach of [92] introduces a model

of $P_{\boldsymbol{\theta}}$ in the form of a mixture of Gaussians, and approximates (5.6) by $P_{\boldsymbol{\gamma}}(\boldsymbol{Y}, |, \mathbb{E}_{\theta}[\mathbf{L}1 \mid \mathbf{X}^{obs}])$, where $\mathbb{E}_{\theta}[\mathbf{L}_1 \mid \mathbf{X}^{obs}]$ is the expected activation at the first layer of the GNN defining $P_{\boldsymbol{\gamma}}$. Finally, `GSPN` [27] explicitly models $P_{\boldsymbol{\theta}}$ with graph-induced sum–product networks, so missing features are handled by exact marginalization.

An alternative to all these approaches that work entirely with models $P_{\boldsymbol{\theta}}, P_{\boldsymbol{\gamma}}$ for the (complete) data distribution is to include the missingness mechanism explicitly in a model $P_{\boldsymbol{\gamma}^+}(\boldsymbol{Y}|\mathbf{X}^{obs}, \boldsymbol{M})$, that directly captures the left side of (5.2). We here write $\boldsymbol{\gamma}^+$ for the parameters of the model to emphasize that it can be structurally similar to a model $P_{\boldsymbol{\gamma}}(\boldsymbol{Y}|\boldsymbol{X})$, but different in that it has the missingness matrix $\boldsymbol{M}$ as an explicit extra input.

This modeling strategy, often referred to as the Missing Indicator Method (MIM), has been studied in the context of supervised learning with missing features [97], but, to the best of our knowledge, it has not been explored in the context of graph machine learning. In this work, we propose a GNN-based instantiation of the MIM framework, which we call `GNNmim`. In `GNNmim`, we implement $P_{\boldsymbol{\gamma}^+}$ as a GNN, we construct the matrix *zero-pad*$(\mathbf{X}^{obs})$ in which missing values are filled in by zeros, and use the concatenation *zero-pad*$(\mathbf{X}^{obs})[i,:]||\boldsymbol{M}[i,:]$ as the feature vector for node i in an otherwise standard GNN architecture[1]. `GNNmim` does not rely on any MAR assumptions, and thereby can be expected to perform more robustly than other approaches under different missingness mechanisms. As our experiments in Section 5.5 show, this simple yet principled strategy yields robust performance across a wide variety of missingness scenarios. In Appendix C.9, we provide additional analyses where the missing-feature mask is applied not only to zero imputation but also to the existing models presented in this section.

5.3 Are we evaluating GNNs for missing features on the right data?

A rigorous evaluation of GNNs under feature missingness requires not only well-designed models, but also datasets that are suitable for the problem at hand. Recent work in the graph learning community has emphasized the importance of dataset suitability in benchmarking [9, 22]. In the context of learning with missing node features, dataset suitability is even more critical. Models designed to handle missingness should be

[1]We deliberately here say "zero-padding" rather than "zero-imputation". The latter would imply that we view the zeros as somehow reasonable stand-ins for the true unobserved values. We view the zeros as arbitrary placeholders. Ideally, the trained model will learn to ignore these values when the corresponding missingness indicator is 1.

tested on datasets where the presence of missing features meaningfully affects model performance and where reasoning under missingness is necessary and non-trivial.

The current standard practice in the literature is to evaluate state-of-the-art methods on a set of widely-used benchmarks for node-level tasks, namely, Cora, Citeseer, Pubmed, AmazonComputers, and AmazonPhoto. In these datasets, node features are constructed as follows: Cora, Citeseer and Pubmed use binary bag-of-words features, while AmazonComputers and AmazonPhoto use TF-IDF vectors [3]. These feature matrices are typically very sparse, which we quantify using the notion of *feature sparsity*, formally defined as below:

Definition 4 (Feature Sparsity)**.** Given a node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, the *feature sparsity* is defined as the proportion of zero entries: $s(\mathbf{X}) = \frac{1}{nd} \sum_{i=1}^{n} \sum_{j=1}^{d} \mathbf{1}[X_{ij} = 0]$, where $\mathbf{1}[\cdot]$ denotes the indicator function.

Table 5.1: Feature sparsity across benchmarks and custom datasets.

Dataset	**#Features**	**Sparsity ↓**	**Type of features**
Cora	1433	0.9873	BoW (binary)
Citeseer	3703	0.9915	BoW (binary)
Pubmed	500	0.8998	BoW (binary)
Synthetic	5	0.0000	Gaussian
Air	7	0.1615	Raw
Electric	5	0.2000	Raw
Tadpole	15	0.0000	Raw

The sparsity values of the benchmark datasets are reported in Table 5.1 (first three rows). All datasets exhibit substantial sparsity, with more than 50% of features being zero across all the datasets, with Citeseer reaching an extreme sparsity level of approximately 99%. This raises a crucial question: does it make sense to evaluate models designed to handle missing features on datasets where the feature representations are already extremely sparse? In such sparse settings, a high probability of missingness is needed to induce a meaningful information loss. Otherwise, the observed model performance under missingness may reflect artifacts of the dataset rather than the robustness of the method. We formalize this observation in the following theorem.

Theorem 2. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathcal{Y}^n$ be random variables, $\mathbf{M} \in \{0,1\}^{n \times d}$ be a missingness mask and $\mathbf{X}^{obs}$ denotes the observed (incomplete) data. We encode the pair $(\mathbf{X}^{obs}, \mathbf{M})$ with the random variable $\tilde{\mathbf{X}}$ with

$$\tilde{X}_{ij} \;=\; \begin{cases} X_{ij}, & M_{ij} = 0, \\ ?, & M_{ij} = 1. \end{cases}$$

Let the change in the information be defined as $\Delta := I(\mathbf{Y}; \tilde{\mathbf{X}}) - I(\mathbf{Y}; \mathbf{X})$, where $I(\cdot;\cdot)$ denotes the mutual information, i.e., a measure of the statistical dependence between two random variables. Then,

1. If the missingness is label-MAR, then $\Delta \;\leq\; 0$.

2. If $\mathbf{X} \in \{0,1\}^{n \times d}$ and the missingness is U-MCAR with missingness probability μ, and $s(\mathbf{X})$ is the sample sparsity as in Definition 4, then

$$-\, nd\, \mu\, h_2\big(\mathbb{E}[s(\mathbf{X})]\big) \;\leq\; \Delta \;\leq\; 0,$$

   where $h_2(u) = -u \log u - (1-u)\log(1-u)$.

Intuition. When node features are extremely sparse (e.g., BoW/TF-IDF), the information loss induced by missingness is provably negligible unless missingness is extremely high. The inequality above formalizes this observation: masking entries cannot increase the information that features provide about the labels (hence $\Delta \leq 0$), and the possible loss of information is bounded by a term that depends on the number of feature entries nd, the missingness probability μ, and the entropy of the feature sparsity. As a result, existing sparse benchmarks inherently make all methods appear robust, preventing meaningful comparison.

The proof can be found in Appendix C.1. Theorem 2 demonstrates that when feature sparsity is high, a very large amount of missingness is required to produce a meaningful loss of information.

This confirms that such benchmarks do not meaningfully differentiate between approaches, casting doubt on their suitability for evaluating GNNs under feature missingness. As a consequence, we argue for the use of datasets where missingness poses a real challenge. In particular, we introduce a set of four alternative datasets, one new synthetic and three real-world. More details about the datasets are reported in Appendix C.3.

(1) A synthetic dataset tailored to controlled missingness. We construct a dataset based on a Barabási–Albert graph topology, where node features are sampled from a Gaussian distribution. Node labels are assigned using a fixed two-layer GCN applied to the full, complete features, ensuring that a GNN model has the capacity to achieve high classification accuracy in the absence of missingness. This controlled setting provides a testbed for isolating the effects of missingness under varying sparsity, while maintaining a well-defined ground truth.

(2) Real-world datasets with semantically meaningful features. We also advocate for the use of real datasets in which node features correspond to raw, observable properties: 1) **Air** [115], a sensor network dataset from IoT applications, where node features correspond to environmental measurements and node labels indicate sensor status categories; 2) **Electric** [13, 7], a dataset of interconnected electrical sensors, with real-valued measurements as features and operational condition classification as the target task; 3) **Tadpole** [116], a medical graph dataset derived from the TADPOLE challenge, where each node represents a patient, node features include clinical and imaging biomarkers, and the goal is to predict diagnostic labels.

Table 5.2: Evaluation of P1 (feature-structure separability) and P2 (feature-structure complementarity) on our custom datasets. Each cell reports the KS statistic and associated p-value for separability under six perturbation settings. $\gamma_{1,1}$ indicates the feature-structure complementarity. Datasets satisfying each property (as per [22]) are marked with ✓.

Dataset	Empty Feat.	Random Feat.	Complete Feat.	Empty Graph	Random Graph	Complete Graph	$\gamma_{1,1}$	P1	P2
Synthetic	1.00 (8.80e-62)	1.00 (8.80e-62)	1.00 (1.93e-14)	1.00 (1.03e-17)	1.00 (8.80e-62)	1.00 (8.80e-62)	0.62	✓	✓
Air	1.00 (8.80e-62)	1.00 (8.80e-62)	1.00 (8.80e-62)	0.67 (1.53e-30)	1.00 (8.80e-62)	1.00 (8.80e-62)	0.68	✓	✓
Electric	1.00 (8.80e-62)	1.00 (8.80e-62)	1.00 (8.80e-62)	0.98 (1.90e-57)	1.00 (8.80e-62)	1.00 (8.80e-62)	0.69	✓	✓
Tadpole	1.00 (8.80e-62)	0.90 (5.31e-44)	0.61 (4.22e-18)	0.77 (1.53e-30)	1.00 (8.80e-62)	1.00 (8.80e-62)	0.64	✓	✓

Both the synthetic and real-world datasets exhibit low feature sparsity (Table 5.1), a necessary condition for studying missingness. However, sparsity alone is not sufficient: suitable datasets must also ensure that both features and structure are task-informative and interact non-trivially. We assess this using the RINGS framework [22], which measures performance separability via KS statistics under perturbations (e.g., removing all edges or replacing features with noise), and features-topology complementarity via the normalized Gromov–Wasserstein distance $\gamma_{1,1}$ between the structural and feature-induced metric spaces, i.e., a measure of how similar the pairwise distance structures induced by the graph topology and by the node features are (values above 0.5 are considered satisfactory). As shown in Table 5.2, all proposed datasets satisfy both mode complementarity and performance separability. Combined with their low feature

sparsity, these properties make the datasets more suitable than traditional benchmarks for evaluating robustness to incomplete node attributes.

While the real-world datasets we introduce have moderate numbers of nodes and features (Table C.1), they satisfy the three key requirements for evaluating robustness to missing node attributes: (i) dense, semantically meaningful, low-dimensional features; (ii) non-trivial predictive signal under complete information; and (iii) complementary and separable contributions of features and structure. To the best of our knowledge, no existing large-scale graph datasets simultaneously meet all these criteria. This limitation is structural to current benchmarks and has been noted in recent work [9]. Importantly, the effect of missingness on model performance does not depend on graph size: in Appendix C.5 we replicate our experiments on a larger variant of the SYNTHETIC dataset (both in number of nodes and features) and observe trends fully consistent with those reported in the main analysis.

5.4 Beyond Uniform Missingness

Dataset suitability is only one dimension of the evaluation problem. A second, equally important factor is the choice of the missingness mechanism under which models are tested. In the literature, nearly all prior works adopt a masking scheme based on *U-MCAR* mechanism. In other works [92, 96], a different variant is used where entire feature vectors of randomly selected nodes are masked. We denote this as ***Structural MCAR (S-MCAR)***. These two settings have become the default evaluation standards in the context of graph learning. We argue that more challenging and realistic missing data patterns need to be considered for a more informative evaluation of different methods' capabilities. We first introduce a more challenging MCAR mechanism:

Label–Dependent MCAR (LD-MCAR). Missingness here is applied at the feature (column) level, assigning higher missingness probability to features X_j that are more informative for the label, as measured by the mutual information $I(X_j; Y)$.

Then, each entry X_{ij} is masked independently with probability $P(M_{ij} = 1) = \rho \cdot I(X_j; Y)$, where $\rho \in [0, 1]$ is a scaling factor selected to achieve the overall desired expected missingness rate across the dataset. Importantly, this mechanism is still MCAR: the probability that a specific entry is missing does not depend on the actual value of the feature or the label, but only on the mutual information of the feature column and the label.

Outside of graph learning, authors have also emphasized the importance of MAR and MNAR mechanisms that reflect more realistically the kinds of missingness encountered in real-world applications[38, 73, 49, 97]. In many practical scenarios, missing features are indeed related to their values or to the prediction target. For instance, a patient might be less likely to report their weight if it is above a certain threshold. This corresponds to a Missing Not At Random (MNAR) mechanism [81]. Testing GNN models exclusively under MCAR conditions thus fails to capture the challenge of more realistic settings. We therefore propose two different MNAR scenarios:

Feature-Dependant MNAR (FD-MNAR). In this mechanism the probability of missingness depends on the value of the feature itself. In particular, we assume that extreme feature values, e.g., high quantiles, are more likely to be missing, as often observed in real-world settings such as healthcare, where abnormal values may be withheld. Formally, for each feature column j, let $q_j^{(\tau)}$ denote the τ-quantile of the observed values. We define the missingness probability for entry X_{ij} as:

$$P(M_{ij} = 1) = \begin{cases} \mu^{\text{hi}} & \text{if } X_{ij} \geq q_j^{(\tau)}, \\ \mu^{\text{lo}} & \text{otherwise,} \end{cases}$$

with $\mu^{\text{hi}} > \mu^{\text{lo}}$ and both chosen selected to match a desired overall missingness rate.

Class–Dependent MNAR (CD-MNAR). In this mechanism, features whose values are informative for the label, are more likely to be omitted. For example, in medical datasets, patients may be less likely to disclose whether they smoke, a feature strongly associated with the label indicating a history of heart attack. To identify such dependencies, we train a decision tree classifier in a one-vs-rest setting, using the observed features to predict class membership. For each class $c \in \{1, \ldots, C\}$, we extract decision paths that lead to leaf nodes predicting c. These paths define a set of feature-value conditions that contribute to the prediction of class c, which we denote as $\mathcal{R}_c$. Let $\texttt{Cond}_c(j, X_{ij})$ be a predicate that evaluates to true if the value of feature j for node i satisfies at least one condition in $\mathcal{R}_c$. Then, the missingness probability is defined as:

$$P(M_{ij} = 1 \mid Y_i = c) = \begin{cases} \mu^{\text{hi}} & \text{if } \texttt{Cond}_c(j, X_{ij}) = \text{true}, \\ \mu^{\text{lo}} & \text{otherwise,} \end{cases}$$

where $\mu^{\text{hi}} > \mu^{\text{lo}}$, and both are selected to meet a target overall missingness rate.

In almost all existing experimental studies the missingness mechanism is the same in training and test data. An exception is [23], where two types of test data are considered: data that underlies the same missingness as the training data, and complete data. We consider a possible distribution shift in $P_{\boldsymbol{\lambda}}(\boldsymbol{M}|\boldsymbol{X},\boldsymbol{Y})$ to be an important concern for two reasons: first, it represents a realistic scenario in practical applications. For instance, training data may consist of historical records collected over time, which may contain missing features due to manual entry or outdated systems. In contrast, test data are collected in real time with modern infrastructure, and all feature values are available. This results in a shift from incomplete to complete data between training and testing.

The second reason for considering distribution shifts in $P_{\boldsymbol{\lambda}}$ is to assess a possible weakness of `GNNmim`: as a model of the form $P_{\boldsymbol{\gamma}^+}(\boldsymbol{Y}|\mathbf{X}^{obs},\boldsymbol{M})$ it explicitly incorporates a model of the missingness mechanism, and thereby could be expected to be less robust under missingness distribution shifts than models that are based on MAR assumptions and (5.6) (which would be expected to be robust as long as the mechanism is feature and label MAR in both training and test data). We therefore define two evaluation regimes (R1 and R2) with and without a shift in the missingness process. Let $\mu_{\text{tr}}(\mathbf{M} \mid \mathbf{X}, \mathbf{Y})$ and $\mu_{\text{te}}(\mathbf{M} \mid \mathbf{X}, \mathbf{Y})$ denote the missingness distributions in training and testing, respectively.

R1: *i.i.d. missingness* (no shift). The same missingness mechanism (*U-MCAR*, *S-MCAR*, *LD-MCAR*, *FD-MNAR*, *CD-MNAR*) and rate are applied to training and test data, i.e., $\mu_{\text{tr}} = \mu_{\text{te}}$.

R2: *missingness distribution shift* (train $\neq$ test). In this setting, we evaluate combinations of a training missingness mechanism $M_{\text{tr}} \in \{\textit{FD-MNAR}, \textit{CD-MNAR}\}$ with missingness probability $\mu_{\text{tr}} = 50\%$, and a test missingness mechanism $M_{\text{te}} =$ *U-MCAR* with missingness probability $\mu_{\text{te}} \in \{0\%, 25\%, 50\%\}$.

5.5 Experimental Results

We conduct experiments on node classification task using the datasets introduced in Section 5.3 and the more realistic missingness protocols described in Section 5.4. We compare a range of GNN-based models specifically designed to handle missing features described in Section 5.2, namely `GNNzero`, `GNNmedian`, `GNNmi`, `GCNmf`, `GOODIE`, `GSPN`, `PCFI`, `FP`, and `FairAC` as well as our proposed method, `GNNmim`. Following the evaluation protocol adopted by these competitors, we perform all main experiments in a transductive setting, where the full graph structure is available during training

and testing, and the task is to predict labels for nodes that belong to the same graph observed during training. However, we note that `GNNmim` can also be applied in an inductive scenario, where the model is trained on one set of nodes and evaluated on previously unseen nodes (possibly belonging to a different graph). For completeness, in Appendix C.8 we report additional experiments conducted under this inductive setting. For all the experiments, we decide to treat the specific GNN layer type in `GNNmim` as a hyperparameter. Full implementation details and hyperparameter settings are provided in Appendix C.4. The code is provided in the supplementary material. The experiments are designed to answer the following research questions:

- **Q1**: Do the datasets of Section 5.3 provide new and complementary insights regarding the robustness of GNNs under varying rates of missing features?
- **Q2:** How robust are different models for handling incomplete features to different types of missingness?
- **Q3**: Do different models maintain their performance under distribution shifts in missingness between training and test sets?

Q1: To assess the impact of the dataset on evaluating robustness under different missingness rates, we compute the F1 score for each model as a function of the missingness rate μ. Figure 5.1 reports these curves under *Structural MCAR (S-MCAR)* under R1 regimes (see Section 5.4) for both the standard benchmarks (Cora, Citeseer, Pubmed) and the datasets we propose (Electric, Air, Tadpole, and Synthetic). Results for other missingness mechanisms lead to equal conclusions and are included in Appendix C.2.

On Cora, Citeseer, Pubmed, all models appear robust, as their F1 score remains high across a wide range of μ, and only drops at very high missingness rates (85-90%). In contrast, on our proposed datasets, performance drops much earlier, often already at low missingness rates. On Tadpole, the degradation is less pronounced at low μ overall; however, two models, `GOODIE` and `GSPN`, notably diverge from the rest, showing much weaker performance even with limited missingness.

These results show that evaluating robustness solely on traditional benchmarks may lead to overly optimistic conclusions on the robustness of the methods. To properly assess the behavior of GNNs under different missing rates, it is essential to use more challenging datasets.

Q2: To assess robustness across mechanisms, we compute the area under the F1–missingness curve (AUC) for each dataset, model, and missingness mechanism under

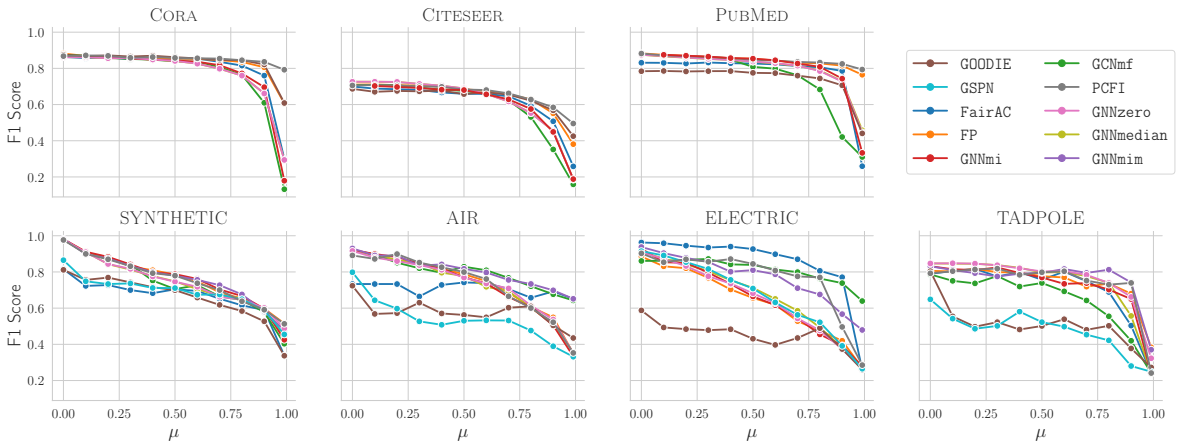


Figure 5.1: Mean F1-score across 5 runs as a function of the missingness probability μ on the proposed datasets and established benchmarks. Each panel reports the performance of all models on a specific dataset under the ***S-MCAR*** setting. The complete tables for all missingness mechanisms are provided in Appendix C.2.

R1 regimes (complete F1 results by model, dataset, missingness rate, and mechanism are reported in Appendix C.6).

Figure 5.2 reports the AUC scores as heatmaps, where lighter colors indicate better model performance for each mechanism within each dataset. We observe that many existing methods exhibit strong sensitivity to the missingness type. For example, `FairAC` performs well under *S-MCAR* settings on ELECTRIC (0.870 AUC, ranking first among all the models), but its performance degrades significantly under *FD-MNAR* on SYNTHETIC (0.641, ranking second-last). Similarly, `GOODIE` ranks highest on SYNTHETIC with uniform missingness (0.771), yet drops to 0.587 under *CD-MNAR*.

These results confirm that performance under *U-MCAR* is not predictive of robustness under more realistic *FD-MNAR* scenarios. This calls into question the validity of evaluations based only on uniform or structure-based missingness. Our proposed method, `GNNmim`, exhibits consistently high AUC across all missingness types and datasets. These results suggest that broad robustness to diverse and realistic missingness mechanisms is achievable, even with lightweight models that do not rely on any MAR assumptions.

This behavior can be partly explained by the assumptions underlying several competing methods. Approaches such as `GCNmf` and `FairAC` implicitly rely on variants of the MAR assumption, either by modeling the feature distribution $P_{\boldsymbol{\theta}}$ or by reconstructing missing values through neighbor aggregation. When the missingness mechanism is MNAR, these assumptions are violated, and the imputation process may introduce

systematic biases in the reconstructed features.

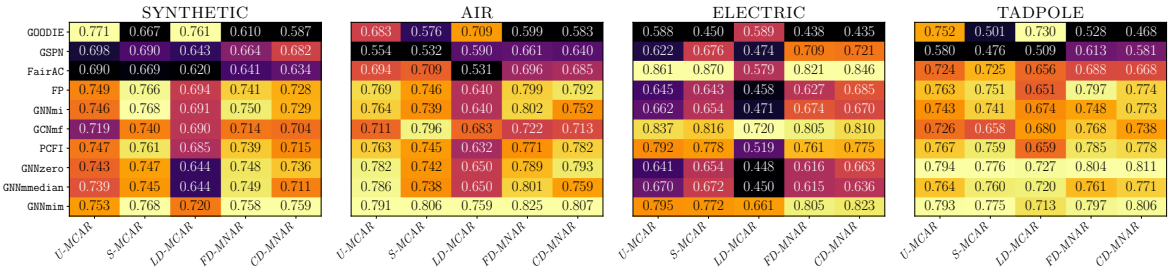


Figure 5.2: Column-normalized heatmaps showing the AUC (area under the F1 vs. missingness rate μ curve) for each model, dataset, and missingness mechanism. Higher values (lighter colors) indicate better overall robustness across increasing levels of missingness.

Q3: To evaluate model robustness under distribution shifts in missingness, we compute the F1 score (mean $\pm$ standard deviation over 5 runs) for each dataset, model, and shift configuration of the R2 regime (Section 5.4). Full results are in Appendix C.7; Figure 5.3 shows a representative subset of the best-performing models from Q2 (`GNNmim`, `GNNmi`, `GCNmf`, `FP`, `PCFI`), trained on *FD-MNAR* with $\mu_{\text{tr}} = 50\%$ and tested on *U-MCAR* with $\mu_{\text{te}} \in \{0\%, 25\%, 50\%\}$. Similar results hold for other models and for the case where the training missing mechanisms is *CD-MNAR* (Appendix C.7).

Each panel shows one dataset, with F1 on the x-axis, models on the y-axis, and color indicating μ_{te} (yellow 0%, blue 25%, green 50%). Dots show mean F1, horizontal lines the standard deviation, and the red vertical bar marks the results obtained in the regime R1 with *FD-MNAR* mechanism on both training and test and $\mu_{\text{tr}} = \mu_{\text{te}} = 50\%$. We observe two findings.

1. Distribution shift generalization is challenging: in almost all cases, performance under R2 test conditions *U-MCAR* 25% is lower than in the i.i.d. R1 setting, despite the test missingness being less severe. This is visible when the blue dot ($\mu_{\text{te}} = 25\%$) lies to the left of the red vertical bar ($\mu_{\text{tr}} = \mu_{\text{te}} = 50\%$). This shows that distribution shifts in missingness create a harder generalization challenge that is not explained solely by missingness severity. The effect is also dataset-dependent, further reinforcing the need to evaluate robustness under these shifts and under different datasets.

2. `GNNmim` is competitive with respect to other models even under R2 conditions. Across datasets and levels of test missingness, `GNNmim` tends to achieve the highest F1 scores

(i.e., yellow, blue, and green dots are consistently farther to the right). In spite of its potential vulnerability in the R2 setting, `GNNmim` is seen to maintain its advantage over the alternative approaches.

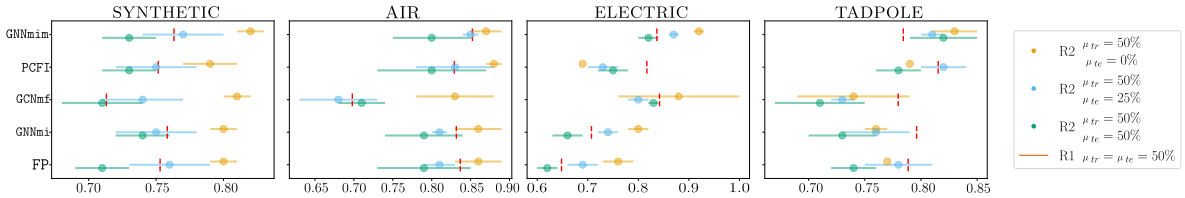


Figure 5.3: F1 scores (mean ± std over 5 runs) under distribution shifts in missingness between training and test data. All models are trained with *FD-MNAR* missingness at 50%. Each panel corresponds to a dataset; each row to a model. Colored dots represent test-time F1 under *U-MCAR* with varying missingness rates: yellow = 0%, blue = 25%, green = 50%. Vertical red lines indicate the F1 achieved in the i.i.d. setting (*FD-MNAR* 50% at both train and test).

5.6 Conclusion and Future Work

We revisited the problem of learning GNNs under missing node features, highlighting fundamental limitations of current evaluation protocols, namely the reliance on benchmarks with sparse features and oversimplified missingness mechanisms. To address these issues, we introduced new datasets with dense, informative features and more realistic missingness patterns that go beyond MCAR, and proposed `GNNmim`, a simple yet effective method that explicitly models missingness through the missing-indicator approach. Our experiments show that `GNNmim` is competitive with respect to more complex architectures across diverse datasets, missingness types, and train–test shifts. This work calls for a shift towards more realistic evaluation settings and demonstrates that lightweight yet principled strategies can achieve strong robustness in challenging missing-feature scenarios.

As a direction for future work, our study underscores the need for larger and more diverse benchmarks specifically designed for missing features, aligning with recent calls for better datasets in graph learning [9], and reveals that there remains substantial room for developing models that are robust to diverse rates and types of missingness. Another promising direction concerns the development of more realistic MNAR mechanisms, potentially incorporating graph-specific dependencies where missingness is influenced by structural properties of the graph itself. Designing richer, structurally grounded

MNAR processes would allow for more faithful stress-testing of models in settings that better reflect more complex patterns.

Chapter 6

Conclusions

This thesis has explored interpretable and robust learning methods for graph-structured data, with a particular focus on multi-relational graphs and learning under incomplete feature information. The proposed approaches address several fundamental challenges in relational graph learning, spanning relational modeling, interpretability, and robustness to imperfect data. At the same time, the results presented in this work naturally give rise to limitations and open questions that point toward promising directions for future research.

6.1 Limitations

The interpretability achieved by the proposed models is primarily structural in nature. By grounding predictions in relational patterns and meta-paths, the models provide insight into which combinations of relations are relevant for a given task. However, this form of interpretability does not capture all aspects of model behavior. In particular, interactions between node attributes and relational structure are not explicitly disentangled, and feature-level contributions remain implicit. As a consequence, the explanations produced by the models should be understood as high-level relational rationales rather than complete or causal explanations of individual predictions.

Another limitation concerns assumptions on the availability and reliability of the graph structure. Several contributions in this thesis assume that the relational structure is fully observed and accurately reflects the underlying data-generating process. While this assumption is reasonable in curated relational databases and knowledge graphs, it may not hold in scenarios where edges are noisy, uncertain, or partially missing. The interaction between structural uncertainty and relational learning is not explicitly

addressed in this work, which may limit the applicability of the proposed methods in less controlled environments.

Finally, the experimental evaluation presented in this thesis focuses predominantly on node-level prediction tasks. Although the proposed approaches are conceptually applicable to edge-level and graph-level settings, these scenarios have not been explored in depth. As a result, the empirical evidence provided does not fully characterize the behavior of the models across the full spectrum of graph learning tasks.

6.2 Future Directions

A natural direction for future research is to relax the assumption of fully observed graph structure and to study models that can jointly reason about uncertain or incomplete relational information. Integrating relational learning with probabilistic modeling of graph structure could improve robustness in settings where both edges and node attributes are unreliable or missing.

Another promising avenue concerns the development of richer forms of interpretability. While this thesis focuses on structural explanations based on relational patterns, future work could explore complementary perspectives, such as feature-aware relational explanations, counterfactual reasoning over graph structure, or hybrid approaches combining symbolic and neural representations. These directions could further enhance transparency and trust in graph-based decision systems.

Extending robust learning methods to heterogeneous and multi-relational graphs also represents an important future challenge. The study of missing node features in this thesis is deliberately grounded in homogeneous graph settings to isolate fundamental effects. Generalizing these insights to heterogeneous graphs, where missingness interacts with multiple node types, relation schemas, and feature spaces, remains an open and non-trivial problem.

Finally, deeper theoretical analysis of relational and robust graph neural networks constitutes a key direction for future work. Formalizing conditions under which relational structure can compensate for incomplete or unreliable feature information, and characterizing identifiability and generalization properties in multi-relational settings, would provide valuable theoretical foundations for the practical deployment of graph learning models.

Closing Remarks

In conclusion, this thesis contributes to the development of graph learning methods that are both interpretable and robust, addressing key limitations of existing approaches in relational settings. By bridging relational modeling, explanation, and learning under incomplete information, this work lays a principled foundation for future research on graph neural networks operating in complex real-world domains.

Bibliography

[1] Carlo Abate and Filippo Maria Bianchi. Maxcutpool: differentiable feature-aware maxcut for pooling in graph neural networks. *arXiv preprint arXiv:2409.05100*, 2024.

[2] Benjamin Agbo, Hussain Al-Aqrabi, Richard Hill, and Tariq Alsboui. Missing data imputation in the internet of things sensor networks. *Future Internet*, 14(5):143, 2022.

[3] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[4] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[5] Jaume Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, 2013.

[6] Steve Azzolin, Antonio Longa, Stefano Teso, and Andrea Passerini. Reconsidering faithfulness in regular, self-explainable and domain invariant GNNs. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] Jongoh Baek and Adam B Birchfield. A tuning method for exciters and governors in realistic synthetic grids with dynamics. In *2023 North American Power Symposium (NAPS)*, pages 1–6. IEEE, 2023.

[8] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

[9] Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M Bronstein, Mathias Niepert, Bryan Perozzi, et al. Position: Graph learning will lose relevance due to poor benchmarks. *arXiv preprint arXiv:2502.14546*, 2025.

[10] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut pooling in graph neural networks, 2020.

[11] Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. *Advances in neural information processing systems*, 36:71603–71618, 2023.

[12] Bahareh Bina, Oliver Schulte, Branden Crawford, Zhensong Qian, and Yi Xiong. Simple decision forests for multi-relational classification. *Decision Support Systems*, 54(3):1269–1279, 2013.

[13] Adam B Birchfield, Ti Xu, Kathleen M Gegner, Komal S Shetye, and Thomas J Overbye. Grid structural characteristics as validation criteria for synthetic networks. *IEEE Transactions on power systems*, 32(4):3258–3265, 2016.

[14] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[15] Carlijn IR Braem, Utku S Yavuz, Hermie J Hermens, and Peter H Veltink. Missing data statistics provide causal insights into data loss in diabetes health monitoring by wearable sensors. *Sensors*, 24(5):1526, 2024.

[16] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009.

[17] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

[18] Giulia Carreras, Guido Miccinesi, Andrew Wilcock, Nancy Preston, Daan Nieboer, Luc Deliens, Mogensm Groenvold, Urska Lunder, Agnes van der Heide, Michela Baccini, et al. Missing not at random in end of life care studies: multiple imputation and sensitivity analysis on data from the action study. *BMC medical research methodology*, 21(1):13, 2021.

[19] Yaomin Chang, Chuan Chen, Weibo Hu, Zibin Zheng, Xiaocong Zhou, and Shouzhi Chen. Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning. *Knowledge-Based Systems*, 235:107611, 2022.

[20] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[21] Marc Christiansen, Lea Villadsen, Zhiqiang Zhong, Stefano Teso, and Davide Mottin. How faithful are self-explainable gnns? *arXiv preprint arXiv:2308.15096*, 2023.

[22] Corinna Coupette, Jeremy Wayland, Emily Simons, and Bastian Rieck. No metric to rule them all: Toward principled evaluations of graph-learning datasets. *arXiv preprint arXiv:2502.02379*, 2025.

[23] Yufeng Ding and Jeffrey S Simonoff. An investigation of missing data methods for classification trees applied to binary response data. *Journal of Machine Learning Research*, 11(1), 2010.

[24] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014.

[25] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

[26] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[27] Federico Errica and Mathias Niepert. Tractable probabilistic graph representation learning with graph-induced sum-product networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[29] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165, 2014.

[30] Rahmat Nur Faizin, Mardhani Riasetiawan, and Ahmad Ashari. A review of missing sensor data imputation methods. In *2019 5th International Conference on Science and Technology (ICST)*, volume 1, pages 1–6. IEEE, 2019.

[31] Wenfei Fan and Floris Geerts. *Foundations of data quality management*. Morgan & Claypool Publishers, 2012.

[32] Francesco Ferrini, Veronica Lachi, Antonio Longa, Bruno Lepri, Matono Akiyoshi, Andrea Passerini, Xin Liu, and Manfred Jaeger. Rethinking gnns and missing features: Challenges, evaluation and a robust solution, 2026.

[33] Francesco Ferrini, Veronica Lachi, Antonio Longa, Bruno Lepri, Andrea Passerini, Xin Liu, and Manfred Jaeger. Beyond sparse benchmarks: Evaluating GNNs with realistic missing features. In *New Perspectives in Graph Machine Learning*, 2025.

[34] Francesco Ferrini, Antonio Longa, Andrea Passerini, and Manfred Jaeger. Meta-path learning for multi-relational graph neural networks. In Soledad Villar and Benjamin Chamberlain, editors, *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pages 2:1–2:17. PMLR, 27–30 Nov 2024.

[35] Francesco Ferrini, Antonio Longa, Andrea Passerini, and Manfred Jaeger. A self-explainable heterogeneous GNN for relational deep learning. *Transactions on Machine Learning Research*, 2025.

[36] James Foulds and Eibe Frank. A review of multi-instance learning assumptions. *The knowledge engineering review*, 25(1):1–25, 2010.

[37] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, pages 2331–2341, 2020.

[38] Amirata Ghorbani and James Y Zou. Embedding for informative missingness: Deep learning with incomplete data. In *2018 56th Annual Allerton Conference*

on Communication, Control, and Computing (Allerton), pages 437–445. IEEE, 2018.

[39] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. Pmlr, 2017.

[40] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[41] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE transactions on neural networks and learning systems*, 35(2):2708–2718, 2022.

[42] Dongliang Guo, Zhixuan Chu, and Sheng Li. Fair attribute completion on graph with missing attributes. *arXiv preprint arXiv:2302.12977*, 2023.

[43] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[44] Audinga-Dea Hazewinkel, Jack Bowden, Kaitlin H Wade, Tom Palmer, Nicola J Wiles, and Kate Tilling. Sensitivity to missing not at random dropout in clinical trials: Use and interpretation of the trimmed means estimator. *Statistics in Medicine*, 41(8):1462–1481, 2022.

[45] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[46] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

[47] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[48] Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. How to deal with missing data in supervised deep learning? In *10th International Conference on Learning Representations*, 2022.

[49] Manfred Jaeger. The aim and em algorithms for learning from coarse data. *Journal of Machine Learning Research*, 23(62):1–55, 2022.

[50] A. Johnson, T. Pollard, O. Badawi, and J. Raffa. eicu collaborative research database demo (version 2.0.1), 2021.

[51] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A survey on explainability of graph neural networks. *arXiv preprint arXiv:2306.01958*, 2023.

[52] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[53] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[54] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[55] Juho Kopra, Tommi Härkänen, Hanna Tolonen, and Juha Karvanen. Correcting for non-ignorable missingness in smoking trends. *Stat*, 4(1):1–14, 2015.

[56] Gueorgi Kossinets and Duncan J Watts. Empirical analysis of an evolving social network. *science*, 311(5757):88–90, 2006.

[57] Veronica Lachi, Francesco Ferrini, Antonio Longa, Bruno Lepri, and Andrea Passerini. A simple and expressive graph neural network based method for structural link representation. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024.

[58] Veronica Lachi, Francesco Ferrini, Antonio Longa, Bruno Lepri, Andrea Passerini, and Manfred Jaeger. Bridging theory and practice in link representation with graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[59] Ni Lao and William Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81:53–67, 10 2010.

[60] Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[61] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[62] Yi Li, Yilun Jin, Guojie Song, Zihao Zhu, Chuan Shi, and Yiming Wang. Graphmse: Efficient meta-path selection in semantically aligned feature space for graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4206–4214, May 2021.

[63] Marco Lippi, Manfred Jaeger, Paolo Frasconi, and Andrea Passerini. Relational information gain. *Machine Learning*, 83(2):219–239, 2011.

[64] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2019.

[65] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1150–1160, New York, NY, USA, 2021. Association for Computing Machinery.

[66] Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi. Graph-based forecasting with missing data through spatiotemporal downsampling. *arXiv preprint arXiv:2402.10634*, 2024.

[67] Benjamin M Marlin and Richard S Zemel. Collaborative filtering and the missing at random assumption. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 267–275, 2009.

[68] Benjamin M Marlin, Richard S Zemel, Sam T Roweis, and Malcolm Slaney. Recommender systems, missing data and statistical model estimation. In *IJCAI*

proceedings-international joint conference on artificial intelligence, volume 22, page 2686, 2011.

[69] Wolfgang May. Information extraction and integration with FLORID: The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999. Available from http://dbis.informatik.uni-goettingen.de/Mondial.

[70] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[71] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th international conference on world wide web*, pages 754–764, 2015.

[72] Eugenij Moiseevich Mirkes, Timothy J Coats, Jeremy Levesley, and Alexander N Gorban. Handling missing data in large healthcare dataset: A case study of unknown trauma outcomes. *Computers in biology and medicine*, 75:203–216, 2016.

[73] Karthika Mohan and Judea Pearl. Graphical models for processing missing data. *Journal of the American Statistical Association*, 116(534):1023–1037, 2021.

[74] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[75] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China, July 2015. Association for Computational Linguistics.

[76] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[77] Nwamaka U Okafor and Declan T Delaney. Missing data imputation on iot sensor networks: Implications for on-site sensor calibration. *IEEE Sensors journal*, 21(20):22833–22845, 2021.

[78] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[79] Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, et al. Relbench: A benchmark for deep learning on relational databases. *Advances in Neural Information Processing Systems*, 37:21330–21341, 2024.

[80] Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. In *Learning on graphs conference*, pages 11–1. PMLR, 2022.

[81] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[82] Donald B Rubin. An overview of multiple imputation. In *Proceedings of the survey research methods section of the American statistical association*, volume 79, page 84, 1988.

[83] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[84] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[85] Oliver Schulte, Bahareh Bina, Branden Crawford, Derek Bingham, and Yi Xiong. A hierarchy of independence assumptions for multi-relational bayes net classifiers. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 150–159. IEEE, 2013.

[86] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[87] Baoxu Shi and Tim Weninger. Mining interesting meta-paths from complex heterogeneous information networks. In *2014 IEEE International Conference on Data Mining Workshop*, pages 488–495, 2014.

[88] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008.

[89] Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS computational biology*, 1(4):e42, 2005.

[90] Ulrich Stelzl, Uwe Worm, Maciej Lalowski, Christian Haenig, Felix H Brembeck, Heike Goehler, Martin Stroedicke, Martina Zenkner, Anke Schoenherr, Susanne Koeppen, et al. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968, 2005.

[91] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[92] Hibiki Taguchi, Xin Liu, and Tsuyoshi Murata. Graph convolutional networks for graphs containing missing features. *Future Generation Computer Systems*, 117:155–168, 2021.

[93] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Knowledge Discovery and Data Mining*, 2009.

[94] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[95] Daeho Um, Sunoh Kim, Jiwoong Park, Jongin Lim, Seong Jin Ahn, and Seulki Park. Propagate and inject: Revisiting propagation-based feature imputation for graphs with partially observed features. In *Forty-second International Conference on Machine Learning*, 2025.

[96] Daeho Um, Jiwoong Park, Seulki Park, and Jin young Choi. Confidence-based feature imputation for graphs with partially known features. In *The Eleventh International Conference on Learning Representations*, 2023.

[97] Mike Van Ness, Tomas M Bosschieter, Roberto Halpin-Gregorio, and Madeleine Udell. The missing indicator method: From low to high dimensions. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5004–5015, 2023.

[98] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.

[99] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[100] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[101] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[102] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[103] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[104] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*, 2013.

[105] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[106] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[107] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[108] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *CoRR*, abs/1603.08861, 2016.

[109] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[110] Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec. Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems*, 33:19075–19087, 2020.

[111] Le Yu, Leilei Sun, Bowen Du, Chuanren Liu, Weifeng Lv, and Hui Xiong. Heterogeneous graph representation learning with relation awareness. *CoRR*, abs/2105.11122, 2021.

[112] Xiao Yu, Yizhou Sun, Brandon Norick, Tiancheng Mao, and Jiawei Han. User guided entity similarity search using meta-path selection in heterogeneous information networks. *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012.

[113] Seongjun Yun, Minbyul Jeong, Sungdong Yoo, Seunghun Lee, Sean S Yi, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks: Learning meta-path graphs to improve gnns. *Neural Networks*, 153:104–119, 2022.

[114] Sukwon Yun, Xin Liu, Yunhak Oh, Junseok Lee, Tianlong Chen, Tsuyoshi Murata, and Chanyoung Park. Oldie but goodie: Re-illuminating label propagation on graphs with partially observed features, 2024.

[115] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2267–2276, 2015.

[116] Qikui Zhu, Bo Du, and Pingkun Yan. Multi-hop convolutions on weighted graphs. *arXiv preprint arXiv:1911.04978*, 2019.

Appendix A

Supplementary material for chapter 3

A.1 Weight initialization for the scoring function

Given a relation r to be evaluated for potential informativeness, node weights are initialized with values that are consistent with the assumption that r determines the target values to be predicted. As discussed in Section 3.1.1, if a node has a negative label, its neighbours according to relation r cannot be part of a ground-truth meta-path, thus their weight should be close to zero. On the other hand, if all the nodes that are connected to a given node via relation r have a positive label, than that given node could be part of a ground-truth meta-path. We thus initialized node weights are follows:

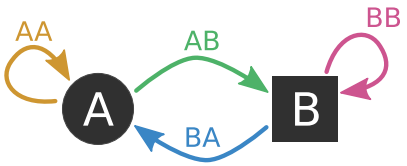
$$w[i] = \min_{j\,:\,i \in \mathcal{N}_j^r} L[j] + \epsilon \tag{A.1}$$

where $j\ :\ i \in \mathcal{N}_j^r$ states that node j is connected to i via relation r, $L[j]$ is the label of node j and $\epsilon \in [-0.3, 0.3]$ is a random value that helps breaking up symmetries during the weight learning process.

A.2 Re-label nodes inside bags

As discussed in Section 3.1.1, from the second iteration onwards targets are associated to bags rather than individual nodes. Adapting the procedure in Eq 3.3 replacing node i with a bag of nodes ends up diluting information too much, so that the informativeness of relations becomes difficult to predict. On the other hand, directly using the weights learned using 3.2 typically leads to an underestimation of the set of positive nodes, as it is sufficient that one node in the set of neighbours of a positive node i has a large weight

for Equation 3.1 to compute the correct label for i. Empirically, we found that running the weight learning procedure M times, with different random initializations of weights as discussed in Section A.1, and retaining for each node its maximum weight across runs, strikes a better balance between precision and coverage in positive node prediction, and thus a better estimate of the informativeness of the corresponding relation. We set $M = 10$ in all our experiments.



Shared relations

Nb. Relations	0	2	4	ALL
4	AA = 0 AB = 1 BA = 2 BB = 3	AA = 0,1 AB = 1 BA = 2,3 BB = 2	AA = 0,3 AB = 1,2 BA = 2,3 BB = 0,1	AA = 0-3 AB = 0-3 BA = 0-3 BB = 0-3
8	AA = 0,1 AB = 2,3 BA = 4,5 BB = 6,7	AA = 0,7 AB = 1,2 BA = 2,3,5 BB = 3,4	AA = 0,1,5 AB = 1,2,7 BA = 4,5,6 BB = 0,7	AA = 0-7 AB = 0-7 BA = 0-7 BB = 0-7
10	AA = 0,1,2 AB = 3,4 BA = 5,6,7 BB = 8,9	AA = 0,1,2 AB = 0,3,4 BA = 5,6,7 BB = 2,8,9	AA = 0,1,2,7 AB = 0,3,4 BA = 5,6,7 BB = 2,3,8,9	AA = 0-9 AB = 0-9 BA = 0-9 BB = 0-9
14	AA = 0,1,2 AB = 3,4,5 BA = 6,7,8,9 BB = 10,11,12,13	AA = 0,1,2,9 AB = 3,4,5,10 BA = 6,7,8,9 BB = 10,11,12,13	AA = 0,1,2,8,9 AB = 3,4,5,10 BA = 6,7,8,9,11 BB = 10,11,12,13	AA = 0-13 AB = 0-13 BA = 0-13 BB = 0-13

Figure A.1: **Synthetic experimental setting.** Each cell is a different experimental setting. A and B are different node types, while $AB = 2, 3$ indicates a setting in which A nodes can be connected to B nodes via relation 2 or 3. Moving from top to bottom we have settings with a large set of relations, while moving from left to right we have settings where relations are progressively less "pure", i.e., the same relation can connect different node-type pairs (e.g., in the second column of the first row there are 2 shared relations: relation 1 that can connect A to A but also A to B, and relation 2 that can connect B to A but also B to B). The last column indicates settings in which any relation is valid between any node-pair type.

Table A.1: **Real-world datasets statistics.** PNC: Person Net Currency (currency associated with an individual's compensation), EDC: Education Domestic Currency (currency associated with domestic tuition fees in educational institutions), EIC: Education International Currency (currency associated with international tuition fees in educational institutions), ELC: Education Local Currency (currency associated with local tuition fees in educational institutions), FBC: Film Budget Currency (currency associated with film budget), GNC: Gdp Nominal Currency (currency associated with GDP nominal value), OC: Organization Currency (currency associated with organization), G: Gender (gender of a person), TS: Team Sport (sport of a specific team), E: Event (links recurring events to their repetitions).

Dataset	# Nodes	# Relations	# Edges	# Features	# Classes	# Target nodes
DBLP	18405	4	67496	334	4	4057
ACM	8994	4	25922	1902	3	3025
IMDB	12772	3	37288	1256	3	2939
FB15K-237 (PNC)	14951	236	306711	100	3	583
FB15K-237 (EDC)	14951	236	308201	100	6	481
FB15K-237 (EIC)	14951	236	305559	100	6	119
FB15K-237 (ELC)	14951	236	307017	100	6	361
FB15K-237 (FBC)	14951	236	315131	100	7	1471
FB15K-237 (GNC)	14951	236	305827	100	3	194
FB15K-237 (OC)	14951	236	307611	100	6	460
FB15K-237 (G)	14951	236	305557	100	2	4530
FB15K-237 (TS)	14951	236	453554	100	9	523
FB15K-237 (E)	14951	236	310742	100	9	134

A.3 Learning meta-path without node features

In this section, we extract meta-paths using the scoring function, excluding the employment of node features. Following this, we apply the identified meta-paths to train `MP-GNN`. A brief analysis of Table A.2 clearly illustrates the crucial impact of node features in extracting valuable meta-paths. The only exception is observed in the case of synthetic dataset 1, which is unsurprising given its inherent simplicity.

Table A.2: `MP-GNN` with and without considering node features in the scoring function.

Dataset	With NF	Without NF
Synthetic 1	**1**(±0.00)	**1**(±0.00)
Synthetic 2	**1**(±0.00)	0.79(±0.05)
Synthetic 3	**1**(±0.00)	0.48(±0.03)
Synthetic 4	**0.97**(±0.02)	0.49(±0.05)
IMDB	**0.64**(±0.01)	0.55(±0.03)
DBLP	**0.94**(±0.01)	0.79(±0.02)
ACM	**0.93**(±0.00)	0.83(±0.03)
PNC (FB15K)	**0.83**(±0.01)	0.49(±0.03)
EDC (FB15K)	**0.96**(±0.01)	0.55(±0.04)
TS (FB15K)	**0.63**(±0.03)	0.52(±0.01)

A.4 Comparison with meta-path miner

The scoring function we propose incrementally builds relevant meta-paths. In this section, we replace the meta-paths identified by the scoring function with those obtained through conventional mining methods. In particular, we rank meta-paths using the well-known path-ranking-algorithm (PRA)[59]. Subsequently, we choose the top three most relevant meta-paths with lengths of 2, 3, and 4. These selected meta-paths are then used to train the `MP-GNN`. In Table A.3, you can observe the macro-F1 scores for three distinct Freebase datasets: PNC, EDC, and TS. It's important to note that we did not conduct this experiment with IMBD, DBLP, and ACM due to their limited number of relations (3 and 4). Table A.3 clearly indicates that the meta-paths identified by the scoring function exhibit significantly higher predictive capabilities compared to those obtained through PRA.

Table A.3: Macro-F1 scores for three distinct Freebase datasets

		PNC	EDC	TS
PRA + `MP-GNN`	meta-path length = 2	0.49	0.14	0.20
	meta-path length = 3	0.49	0.14	0.11
	meta-path length = 4	0.49	0.14	0.11
Scoring Function + `MP-GNN`		**0.83**	**0.96**	**0.63**

A.5 Execution Time

In Table A.4, we provide information on the training time for each model on individual datasets, along with the corresponding F1 scores enclosed in parentheses. Additionally, we present the average execution time and F1 score across the three benchmark datasets (IMBD, DBLP, and ACM), as well as for Freebase and synthetic datasets. All the models achieve a similar average accuracy on the three benchmark datasets, with `RGCN` displaying the shortest execution time and `GTN` the longest. Despite being the second slowest model in terms of execution time, our model achieves the highest average F1 score. It is important to emphasize that our model is designed to learn meaningful meta-paths in networks with a multitude of relation types, whereas the three benchmark datasets only have a limited number of relation types. In the Freebase network, both Fast`GTN` and `GTN` boast the shortest execution times, yet their F1 scores are significantly lower, rendering them ineligible for further consideration. Despite Simple-HGN being the quickest model, its F1 score falls significantly short, averaging 15 points lower than `MP-GNN`. On the other hand, `RGCN` is the second fastest model, but once more, its F1 score lags behind our approach by an average of 9 points. Lastly, concerning the synthetic datasets, our model ranks third in terms of execution time, with Simple-HGN excluded from the analysis due to its poor F1 scores. It's worth highlighting that the fastest model, `FastGTN`, and the second fastest, `RGCN`, both lag behind `MP-GNN` by 23 and 17 points, respectively. In general, our approach is consistently neither the slowest nor the fastest; however, it consistently attains the highest average F1 score in all scenarios.

Table A.4: Execution time in seconds, with the F1 score in parentheses.

	R-HGNN	HGN	RGCN	GTN	FastGTN	MP-GNN
IMDB	680 (0.64)	740 (0.63)	650 (0.60)	1500 (0.62)	910 (0.63)	1000 (0.64)
DBLP	720 (0.86)	870 (0.94)	780 (0.91)	1630 (0.90)	990 (0.92)	1180 (0.94)
ACM	940 (0.90)	750 (0.92)	870 (0.90)	1420 (0.91)	870 (0.93)	960 (0.93)
Mean	780 (0.80)	786 (0.83)	767 (0.80)	1517 (0.81)	923 (0.83)	1046 (0.84)
PNC	7230 (0.72)	560 (0.68)	1830 (0.74)	180 (0.33)	150 (0.33)	2870 (0.83)
EDC	7356 (0.60)	670 (0.75)	1540 (0.71)	190 (0.12)	130 (0.12)	3220 (0.96)
EIC	7020 (0.63)	460 (0.65)	2040 (0.73)	180 (0.12)	110 (0.12)	2480 (0.80)
ELC	820 (0.47)	760 (0.74)	1380 (0.72)	180 (0.15)	130 (0.15)	3010 (0.78)
FBC	5900 (0.45)	410 (0.48)	1190 (0.42)	190 (0.14)	100 (0.14)	2880 (0.60)
GNC	8230 (0.80)	670 (0.74)	1680 (0.82)	175 (0.19)	100 (0.19)	3220 (0.90)
OC	3790 (0.67)	670 (0.73)	1970 (0.78)	185 (0.14)	120 (0.14)	2980 (0.93)
G	5980 (0.81)	450 (0.64)	2010 (0.80)	140 (0.44)	120 (0.44)	2990 (0.84)
TS	5000 (0.67)	410 (0.53)	1995 (0.62)	200 (0.09)	120 (0.09)	3155 (0.63)
E	6790 (0.89)	690 (0.80)	2005 (0.98)	170 (0.07)	140 (0.07)	3040 (0.96)
Mean	5812 (0.67)	575 (0.67)	1764 (0.73)	179 (0.18)	122 (0.18)	2984 (0.82)
Synt 1	320 (1.00)	50 (034)	200 (1.00)	230 (0.93)	210 (0.94)	245 (1.00)
Synt 2	310 (1.00)	66 (0.48)	240 (0.91)	210 (0.84)	180 (0.85)	300 (1.00)
Synt 3	350 (1.00)	68 (0.38)	300 (0.95)	310 (0.84)	280 (0.81)	345 (0.99)
Synt 4	410 (1.00)	78 (0.50)	390 (0.84)	480 (0.47)	320 (0.52)	430 (1.00)
Synt 5	450 (0.98)	67 (0.34)	400 (0.85)	400 (0.91)	360 (0.94)	380 (1.00)
Synt 6	430 (0.95)	120 (0.49)	390 (1.00)	460 (0.82)	400 (0.84)	450 (1.00)
Synt 7	450 (0.96)	94 (0.52)	450 (0.68)	500 (0.8)	490 (0.8)	480 (1.00)
Synt 8	520 (0.89)	128 (0.50)	460 (0.71)	720 (0.47)	450 (0.48)	440 (1.00)
Synt 9	560 (0.93)	135 (0.38)	425 (0.90)	530 (0.90)	430 (0.92)	510 (1.00)
Synt 10	590 (0.89)	90 (0.45)	580 (0.85)	600 (0.87)	520 (0.86)	540 (1.00)
Synt 11	630 (0.85)	200 (0.64)	495 (0.77)	640 (0.8)	510 (0.79)	500 (0.94)
Synt 12	630 (0.90)	130 (0.36)	500 (0.62)	605 (0.48)	525 (0.48)	560 (1.00)
Synt 13	610 (0.82)	175 (0.49)	565 (0.83)	670 (0.88)	570 (0.85)	580 (1.00)
Synt 14	650 (0.88)	200 (0.32)	595 (0.81)	650 (0.8)	600 (0.84)	560 (1.00)
Synt 15	600 (0.87)	185 (0.42)	650 (0.84)	710 (0.77)	680 (0.8)	590 (1.00)
Synt 16	660 (0.81)	200 (0.45)	700 (0.6)	720 (0.48)	705 (0.48)	605 (0.97)
Mean	523 (0.92)	129 (0.44)	476 (0.82)	547 (0.75)	468 (0.76)	484 (0.99)

A.6 FB15K-237 Results with standard deviations

Table A.5: Many-relations dataset: F1-scores for the different node classification tasks on the FB15K-237 dataset. Mean and standard deviation computed over five runs with different random seeds. Best results are highlighted in bold. See Table A.1 in the Appendix for the meaning of the label acronyms.

Label	R-HGNN	HGN	RGCN	GTN	FastGTN	MP-GNN
PNC	0.72(±0.04)	0.68(±0.05)	0.74(±0.03)	0.33(±0.00)	0.33(±0.00)	**0.83**(±0.01)
EDC	0.6(±0.05)	0.75(±0.02)	0.71(±0.02)	0.12(±0.00)	0.12(±0.00)	**0.96** (±0.01)
EIC	0.63(±0.06)	0.65(±0.04)	0.73(±0.01)	0.12(±0.00)	0.12(±0.00)	**0.8**(±0.03)
ELC	0.47(±0.07)	0.74(±0.01)	0.72(±0.02)	0.12(±0.00)	0.15(±0.00)	**0.78**(±0.02)
FBC	0.45(±0.04)	0.48(±0.02)	0.42(±0.00)	0.14(±0.00)	0.14(±0.00)	**0.61**(±0.01)
GNC	0.8(±0.05)	0.74(±0.02)	0.82(±0.03)	0.19(±0.00)	0.19(±0.00)	**0.9**(±0.00)
OC	0.67(±0.02)	0.73(±0.04)	0.78(±0.01)	0.14(±0.00)	0.14(±0.00)	**0.93**(±0.01)
G	0.81(±0.01)	0.64 (±0.01)	0.8(±0.01)	0.44(±0.00)	0.44(±0.00)	**0.84**(±0.04)
TS	**0.67**(±0.04)	0.53(±0.04)	0.62(±0.01)	0.09(±0.00)	0.09(±0.00)	0.63(±0.03)
E	0.89(±0.02)	0.8(±0.03)	**0.98**(±0.00)	0.07(±0.00)	0.07(±0.00)	0.96(±0.00)

Appendix B

Supplementary material for chapter 4

B.1 Second iteration loss computation

In this Section we show the loss computation and minimization for relation "d" of the example in Figure 4.5. Note that Z refers to the learned parameters related to the features of the target nodes in the previous iteration with relation b (Figure 4.3b) and $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$ represent the features of *prescription* nodes.

$$
\begin{aligned}
Z &\gg 0 \\
F(\mathcal{B}_2, d) &= Z\Theta^T \begin{bmatrix}0\\1\end{bmatrix} + 2Z\Theta^T \begin{bmatrix}1\\0\end{bmatrix} (w_8 + w_9) + Z\Theta^T \begin{bmatrix}1\\0\end{bmatrix} (w_{10} + w_{11}) \\
F(\mathcal{B}_3, d) &= 2Z\Theta^T \begin{bmatrix}1\\0\end{bmatrix} (w_8 + w_9) + Z\Theta^T \begin{bmatrix}0\\1\end{bmatrix} (w_{10} + w_{11}) + Z\Theta^T \begin{bmatrix}1\\0\end{bmatrix} (w_{12}) \\
L(d) &= \min_{\Theta, w} \sigma \left(F(\mathcal{B}_3, d) - F(\mathcal{B}_2, d) \right) \\
&= \min_{\Theta, w} \sigma \left(Z\Theta^T \begin{bmatrix}1\\0\end{bmatrix} (2w_8 + 2w_9 + w_{12} - 2w_8 - 2w_9 - w_{10} - w_{11}) \right. \\
&\quad \left. + Z\Theta^T \begin{bmatrix}0\\1\end{bmatrix} (w_{10} + w_{11} - 1) \right) \\
&= \min_{\Theta, w} \sigma \left(Z\Theta^T \begin{bmatrix}1\\0\end{bmatrix} (w_{12} - w_{10} - w_{11}) + Z\Theta^T \begin{bmatrix}0\\1\end{bmatrix} (w_{10} + w_{11} - 1) \right) \\
\Theta^T \begin{bmatrix}1\\0\end{bmatrix} &\gg 0; \quad \Theta^T \begin{bmatrix}0\\1\end{bmatrix} = 0; \quad w_{12} = 0; \quad w_{10}, w_{11} = 1
\end{aligned}
$$

B.2 Ablation study

In this section, we highlight the importance of the skip connection $W_1^{(l)} h_v^{(0)}$ considered in equation 4.7. Table B.1 and B.2 demonstrate respectively the significance of skip connections in the synthetic and real-world settings. Without considering the initial target node features, the F_1 score drops significantly, underscoring the critical role these features play.

Table B.1: F_1 metric with standard deviations for synthetic datasets with `MPS-GNN` and `MPS-GNN` without using skip connection.

	S1	S2	S3	S4	S5	S6	S7	S8
`MPS-GNN`	**0.98**(±0.00)	**0.98**(±0.01)	**0.99**(±0.10)	**0.98**(±0.00)	**0.99**(±0.00)	**0.93**(±0.10)	**0.94**(±0.00)	**0.95**(±0.00)
$\texttt{MPS-GNN}_{\text{no\_skip}}$	0.91(±0.01)	0.93(±0.01)	0.89(±0.02)	0.91(±0.01)	0.88(±0.00)	0.87(±0.02)	0.91(±0.01)	0.85(±0.03)

Table B.2: F_1 metric with standard deviations for real-world datasets with `MPS-GNN` and `MPS-GNN` without using skip connection.

	EICU	**MONDIAL**	**ErgastF1**
`MPS-GNN`	**0.92**(±0.01)	**0.74**(±0.01)	**0.83**(±0.02)
$\texttt{MPS-GNN}_{\text{no\_skip}}$	0.85(±0.02)	0.71(±0.01)	0.80(±0.01)

B.3 Real world setting

In our real-world scenario, we utilized three relational databases, which are detailed below. To convert these databases into heterogeneous graphs, we applied transformations to the attribute columns: categorical attributes were transformed using one-hot encoding, and numerical attributes were normalized to the range $[0, 1]$.

To improve connectivity between target nodes, particularly when the transformation from a relational database to a graph results in each target node (or row in the target table) becoming a separate connected component, we employed simple clustering techniques on the rows of other tables based on their features.

Below, we provide detailed information about the databases and describe the clustering methods used when applicable.

EICU Medical database with 31 tables (node types)[1] from [50]. The task is predicting the duration of a patient's stay in the eICU after admission, modeled as binary node

[1]https://eicu-crd.mit.edu

classification by thresholding duration at 20 hours to achieve two balanced classes. To create clusters of nodes, where each cluster is represented by a single new node that replaces all the nodes within that cluster, we utilized a categorical attribute for each table that is best suited for clustering the specific table.

Table B.4 provides details about the clustering process applied to the nodes of the **EICU** database. For each table, the initial number of rows and the resulting number of clusters (representing the final number of nodes for that type) are shown. The "Clustering Feature" column specifies the column used for creating clusters; if not specified, this indicates the absence of categorical features, and the DBSCAN algorithm is used instead.

MONDIAL Database [2] containing data from multiple geographical web data sources [69]. We predict the religion of a country as Christian (positive) with 114 instances vs. all other religions with 71 instances. In this dataset, clustering of tables is done using DBSCAN [28] clustering algorithm.

Table B.5 shows the resulting number of clusters for each table of the original database. Clustering is computed using DBSCAN algorithm.

ErgastF1 Database [3] containing Formula 1 races from the 1950 season to the present day. It contains detailed information including lap times, pit stop durations, and qualifying performance for all races up to 2017. The objective is to predict the winner of a race using the data available before the race starts, such as the list of participants and qualifying times, while the actual lap times during the race are not yet available.

Table B.3: Setting of real-world datasets. $|\mathcal{T}|$ and $|\mathcal{R}|$ refers respectively to the total number of tables in the original database and the number of relations in the graph used by the models. *Rows* is the sum of all the rows of each specific database.

Database	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	*Rows*
EICU	31	87	457325320
MONDIAL	40	45	21497
ErgastF1	14	33	544056

[2] https://relational-data.org/dataset/Mondial
[3] https://relational-data.org/dataset/ErgastF1

Table B.4: Tables from the **EICU** dataset. 'Clustering Feature' refers to the feature used to group the rows in each table. If not present, it means that the specific table does not have any feature for that purpose, so the DBSCAN algorithm is employed. *Clusters* indicates the final number of nodes after the clustering step.

Table name	Clustering Feature	*Rows*	*Clusters*
admissiondrug	drughiclseqno	7417	578
admissiondx	admitdxname	7578	268
allergy	drughiclseqno	2475	251
apacheapsvar	-	2205	200
apachepatientresult	-	3676	200
apachepredvar	-	2205	200
careplancareprovider	specialty	5627	40
careplaneol	specialty	5627	40
careplangeneral	cplgroup	3314	28
careplangoal	cplcategory	3633	9
careplaninfectiousdisease	cplcategory	112	11
customlab	labothername	30	19
diagnosis	diagnosisstring	24978	110
hospital	region	186	4
infusiondrug	drugname	38256	257
intakeoutput	celllabel	100466	740
lab	labname	434660	147
medication	drughiclseqno	75604	1027
microlab	organism	342	16
note	notepath	24758	360
nurseassessment	cellattributepath	91589	81
nursecare	cellattributepath	42080	19
nursecharting	nursingchartcelltypevalname	1477163	49
pasthistory	pasthistorypath	12109	190
physicalexam	physicalexampath	84058	310
respiratorycare	currenthistoryseqnum	5436	243
respiratorycharting	respchartvaluelabel	5436	243
treatment	treatmentstring	38290	414
vitalaperiodic	-	274088	200
vitalperiodic	-	1634960	200

B.4 Details of competitors' architectures

In Table B.6, we present the hyperparameters for all competitors and `MPS-GNN` on the real-world datasets. For the synthetic cases, the only difference lies in the # **layers**, which is adjusted to match the length of the correct meta-paths for the competitors. For the losses **CE** is cross-entropy and **nll** is negative log likelihood.

Table B.5: Tables from the **MONDIAL** dataset. *Clusters* indicates the final number of clusters after applying DBSCAN algorithm on the features of the specific table.

Table name	*Rows*	*Clusters*
economy	234	5
ethnicGroup	540	65
language	144	20
politics	238	25
population	238	4
encompasses	242	2
province	1450	18
organization	153	15
continent	5	5
city	3111	93
river	218	24
sea	34	17
desert	63	6
lake	130	16
mountain	241	40

Table B.6: Hyperparameters of competitors and `MPS-GNN` for the real-world datasets. The optimizer is omitted from the table as it is Adam for all models. **lr** denotes the learning rate, **wd** represents the weight decay, and **Patience** indicates the early stopping patience (if applicable).

Hyperparameters	**# layers**	**Embedding dim.**	**lr**	**wd**	**# epochs**	**Patience**	**Loss**
`MLP`	2	32	0.01	0.0005	500	50	nll
`GCN`	2	32	0.01	0.0005	500	50	nll
`RGCN`	2	32	0.01	0.0005	500	50	nll
`R-HGNN`	2	64	0.001	0	200	50	CE
`HGN`	2	64	0.0001	0.0005	300	30	nll
`GTN`	2	64	0.01	0.001	200	-	CE
`Fast-GTN`	2	64	0.01	0.001	200	-	CE
`MP-GNN`	2	32	0.01	0.0005	500	50	nll
`MPS-GNN`	2	32	0.01	0.0005	500	50	nll

B.5 Number of parameters

In Table B.7, we present the total number of parameters required for evaluating the various models. In the synthetic setting, when comparing with the only two models that yield satisfactory results, we observe that our approach has a similar number of parameters as `RGCN` (when the total number of relations in the graphs is limited) and `MP-GNN`. `MP-GNN`, which also considers only a subset of graph relations like our method, is designed to have a lower number of parameters; however, it still falls short of matching `MPS-GNN`'s performance.

In the real-world setting, among the models that achieve decent results, `GCN` exhibits

the lowest number of parameters on the **EICU** dataset. However, among the relational methods, MPS-GNN emerges as the most efficient. On the **MONDIAL** dataset, the two leading competitors, HGN and R-HGNN, despite achieving lower F_1 scores, utilize all edge types and consequently require a significantly larger number of parameters.

Finally, on the **ErgastF1** dataset, although MP-GNN outperforms other methods in terms of parameter efficiency by considering different meta-paths, it results in a considerably lower F_1 score. In contrast, HGN and R-HGNN exhibit an exponential increase in the number of parameters.

Table B.7: Number of parameters for each model across synthetic and real-world datasets.

	MLP	GCN	RGCN	R-HGNN	HGN	GTN	Fast-GTN	MP-GNN	MPS-GNN
S1	194	690	3730	525996	10927	866	126902	1346	3618
S2	194	690	6770	787796	42314	946	126942	1346	3618
S3	194	690	3730	525996	42314	866	126902	1346	3618
S4	194	690	6770	787796	74125	946	126942	1346	3618
S5	194	690	3730	525996	74125	866	126902	1346	6786
S6	194	690	6770	787796	74125	946	126942	1346	6786
S7	194	690	3730	525996	74125	866	126902	1346	6786
S8	194	690	6770	787796	74125	946	126942	8834	6786
EICU	1346	3506	400690	47496672	611785	32024	110408	24898	12066
MONDIAL	2144	90546	3709106	88396572	1142962	180554	1539457	183138	234050
ErgastF1	4356	198142	5422432	396543021	439021	2542354	11325242	23413	29538

B.6 Execution times

In Table B.8, we present the training times for each model across the individual datasets. In the synthetic settings, we observe that among models achieving a significant F_1 score (RGCN, MP-GNN, and MPS-GNN), MPS-GNN typically demonstrates the shortest execution time. We would like to highlight that our model is specifically designed to learn meaningful meta-paths in networks with many relation types, whereas the synthetic datasets are limited in the number of relation types. In the **EICU** database, while the MLP model achieves the shortest execution time, it performs poorly in terms of F_1. Among the models with notable results, MPS-GNN exhibits the best execution time. In the **MONDIAL** database, all models have relatively low execution times due to the small graph size, as shown in Table B.3. However, MPS-GNN still achieves the best F_1 score. For the **ErgastF1** dataset, while MLP again has the lowest execution time, its final accuracy is poor. In contrast, MPS-GNN is comparable to MP-GNN and R-HGNN in terms of execution time but surpasses them by 12 and 11 percentage points in F_1 score, respectively. Overall, our approach is consistently neither the quickest nor the slowest,

yet it reliably achieves the highest average F1 score across all settings.

Table B.8: Training times, in seconds, for each model across synthetic and real-world datasets.

	MLP	GCN	RGCN	R-HGNN	HGN	GTN	Fast-GTN	MP-GNN	MPS-GNN
S1	66	660	1094	898	363	388	315	234	322
S2	67	660	2197	674	375	570	812	1461	245
S3	73	675	536	612	380	260	456	657	356
S4	72	483	2142	551	373	697	369	986	457
S5	69	420	445	575	360	875	845	158	467
S6	69	620	111	616	377	567	467	453	321
S7	72	677	285	519	371	834	442	587	449
S8	74	777	167	680	373	878	765	1502	490
EICU	342	5882	4355	4842	3210	10324	682	5787	1273
MONDIAL	156	125	132	131	265	220	119	120	134
ErgastF1	543	954	1491	2456	2245	3015	2945	2280	2421

B.7 Scoring Function Lookahead Illustration

We report an additional experiment demonstrating the lookahead capabilities implemented in the scoring function of our method, and its advantage over a simplistic greedy approach. We construct a synthetic dataset of a multi-relational graph with three relations and the ground truth metapath, r_1, r_2 for the target. We compare our approach against a simple greedy one, in which one always extends the metapath with the relation for which a trained `MPS-GNN` achieves maximal F_1 score. The results are presented in Table B.9. In the first iteration of the scoring function, relation r_1 achieves the lowest loss in our scoring function and would therefore be chosen as the first relation of the metapath. Looking only at the immediate benefit of the relations in terms of the accuracy achieved by a corresponding `MPS-GNN`, however, r_2 would be selected as the best relation. The column r_2-Extensions shows the F_1 scores of all possible length 2 metapaths starting with r_2. Comparing with the F_1 score of the ground truth metapath we find that, indeed, starting the metapath with r_2 is a suboptimal choice, and that our scoring function correctly identifies the most informative relation to start the metapath with, even though this informativeness only is materialized after extension of the metapath with r_2.

Table B.9: Comparison: our metapath construction vs. simple greedy alternative.

	Iteration 1			**r_2- Extensions**			**Ground truth**
Meta-paths	r_1	r_2	r_3	r_2, r_1	r_2, r_2	r_2, r_3	r_1, r_2
Score	**0.001**	45	56				
F_1	0.79	**0.82**	0.69	0.83	0.82	0.85	**0.99**

B.8 Temporal experiment

Recently, a novel benchmark, rel-bench [79], has been introduced. This benchmark consists of multiple relational databases represented as temporal graphs. Additionally, the authors propose `RDL`, a temporal-aware relational message-passing model. It's important to note that the released temporal graphs are not directly compatible with static models. Therefore, in this section, we reconstruct one dataset from the rel-bench repository in a static format and apply our method to this static representation of the graph (setting in table B.10). The tasks are: (1) **dnf**, predicting whether a driver will fail to finish a race within the next month, and (2) **top3**, predicting if a driver will place in the top 3. To handle these temporal tasks, we treated each instance of a node at different timestamps as distinct nodes with separate label predictions. Table B.11 reports the F_1-scores for `MPS-GNN` and the baselines. Our approach outperforms all competitors, including `RDL` which performs lower due to overpredicting the majority class. Note that `RDL` cannot be straightforwardly applied to our other datasets, being designed for temporal datasets. In Section B.8, we provide the necessity calculations for these datasets.

Table B.10: Setting of temporal real-world dataset.

Database	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	*Rows*
rel-f1	9	26	97606

Table B.11: F_1-score with standard deviations of our method and competitors on two temporal datasets.

	rel-f1-dnf	**rel-f1-top3**
MLP	0.48(±0.00)	0.48(±0.00)
GCN	0.57(±0.01)	0.52(±0.02)
RGCN	0.44(±0.02)	0.54(±0.02)
R-HGNN	0.60(±0.01)	0.63(±0.02)
HGN	0.61(±0.02)	0.61(±0.01)
GTN	0.41(±0.02)	0.45(±0.01)
Fast-GTN	0.51(±0.01)	0.50(±0.02)
MP-GNN	0.54(±0.02)	0.52(±0.02)
RDL	0.58(±0.03)	0.53(±0.7)
MPS-GNN	**0.62**(±0.02)	**0.65** (±0.01)

Necessity calculation in temporal tasks

In this section, we present the necessity calculation for the temporal datasets. Specifically, as detailed in Section 5.5, Table B.12 and figureB.1 demonstrates that removing certain identified meta-paths results in decreased performance (F_1) and an increased necessity value. This finding confirms that the learned meta-paths are essential for the prediction task in these datasets as well.

Table B.12: Changes to F_1 and necessity when removing 25%, 50%, and 75% of the learned meta-path occurrences for the real-world temporal tasks **rel-f1-dnf** and **rel-f1-top3**.

	F_1				**Necessity**			
Removed (%)	**0**	**25**	**50**	**75**	**0**	**25**	**50**	**75**
rel-f1-dnf	0.63	0.56	0.49	0.48	0	0.12	0.22	0.31
rel-f1-top3	0.65	0.62	0.55	0.50	0	0.19	0.34	0.34

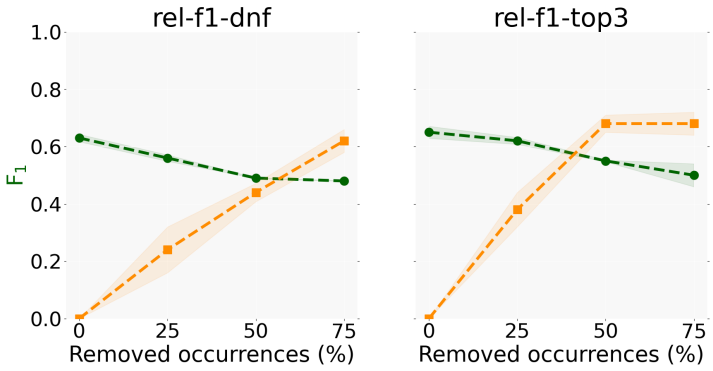


Figure B.1: Changes to F_1 and posterior probability difference (necessity) when removing 25%, 50%, and 75% of the learned meta-path occurrences for the real-world temporal tasks **rel-f1-dnf** and **rel-f1-top3**.

B.9 Non-GNN models on real-world databases

For the **MONDIAL** and **ErgastF1** databases, non-GNN methods have shown competitive performance in the past. For example, [85, 12] report F_1 scores of 0.78 and 0.77 on **MONDIAL**, 0.4 and 0.3 points higher than `MPS-GNN`. However, these results are achieved on a simplified version of the database with only 12 tables, requiring manual feature selection. In contrast, `MPS-GNN` is applied directly to the raw input data across all 40 tables. The non-GNN methods use Multi-relational Bayes Net Classifiers and Simple Decision Forests, where reducing the number of tables and relations aids performance. By comparison, `MPS-GNN` is designed to handle scenarios with a large number of relations effectively.

B.10 Toy example with more complex features

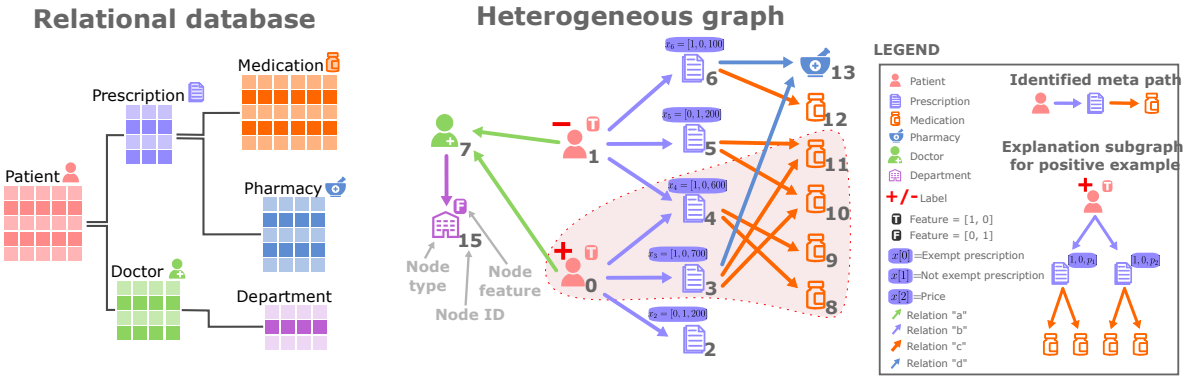


Figure B.2: Toy example similar to Figure 4.1 with more complex features on prescription nodes. As depicted in the legend (**right**), the first two positions of the feature vector determine an exempt prescription ($[1, 0]$) or a not exempt prescription ($[0, 1]$). The last value determine the price of the prescription in dollar. The highlighted subgraph shows a prototypical counts-of-counts pattern characterising positive patients, namely having exempt prescriptions where the total cost is more then 100 dollars, each containing at least two medications

In Figure B.2, we present a scenario similar to the one depicted in Figure 4.1. The key difference lies in the prescription nodes, which now have more complex feature representations. Specifically, the last value of the feature vector corresponds to the price of each prescription.

The first iteration of the scoring function follows the same process as in the previous toy example, as illustrated in Figures 4.3a and 4.3b. The most notable aspect, however, is the second iteration, where the target nodes are prescription nodes, similar to Figure 4.5, where the model evaluates relation d and c. Here, we provide the computations involved in minimizing the loss. Notably, only relation d allows the loss to be minimized to 0. Therefore, relation d will be the one included in the identified meta-path

Minimization relation c

$$
\begin{aligned}
& Z \gg 0 \\
F(\mathcal{B}_2, c) &= 2Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 60 \end{bmatrix} + Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 70 \end{bmatrix} (w_{13}) + Z\Theta^T \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix} \\
F(\mathcal{B}_3, c) &= 2Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 60 \end{bmatrix} + Z\Theta^T \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix} + Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 70 \end{bmatrix} (w_{13}) \\
L(c) &= \min_{\Theta, w} \sigma(F(\mathcal{B}_3, c) - F(\mathcal{B}_2, c)) \\
&= \min_{\Theta, w} \sigma(0) = \frac{1}{2}
\end{aligned}
\tag{B.1}
$$

Minimization relation d

$$
\begin{aligned}
& Z \gg 0 \\
F(\mathcal{B}_2, d) &= 2Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 60 \end{bmatrix} (w_8 + w_9) + Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 70 \end{bmatrix} (w_{10} + w_{11}) \\
&\quad + Z\Theta^T \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix} \\
F(\mathcal{B}_3, d) &= 2Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 60 \end{bmatrix} (w_8 + w_9) + Z\Theta^T \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix} (w_{10} + w_{11}) \\
&\quad + Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 70 \end{bmatrix} (w_{12}) \\
L(d) &= \min_{\Theta, w} \sigma\Big(Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 70 \end{bmatrix} (w_{12}) \\
&\quad + Z\Theta^T \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix} (w_{10} + w_{11} - 1) - Z\Theta^T \begin{bmatrix} 1 \\ 0 \\ 70 \end{bmatrix} (w_{10} + w_{11}) \Big) \\
& \Theta^T \begin{bmatrix} 0 \\ 1 \\ 20 \end{bmatrix} \gg 0, \qquad w_{10}, w_{11}, w_{12} = 0
\end{aligned}
\tag{B.2}
$$

Appendix C

Supplementary material for chapter 5

C.1 Proofs

Theorem 1. If $P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}$ is feature-MAR and label-MAR, then (5.2) simplifies to

$$\int_{\mathbf{X}^{miss}} P_{\boldsymbol{\gamma}}(\boldsymbol{Y}|\boldsymbol{X}) P_{\boldsymbol{\theta}}(\mathbf{X}^{miss}|\mathbf{X}^{obs}). \tag{5.6}$$

Proof.

$$P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{M}) = P_{\boldsymbol{\lambda}}(\boldsymbol{M}|\boldsymbol{X},\boldsymbol{Y}) \frac{P_{\boldsymbol{\gamma}}(\boldsymbol{Y}|\boldsymbol{X})}{P_{\boldsymbol{\gamma},\boldsymbol{\lambda}}(\boldsymbol{M}|\boldsymbol{X})} \overset{(5.4)}{=} P_{\boldsymbol{\gamma}}(\boldsymbol{Y}|\boldsymbol{X})$$

$$P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\mathbf{X}^{miss}|\mathbf{X}^{obs},\boldsymbol{M}) = P_{\boldsymbol{\gamma},\boldsymbol{\lambda}}(\boldsymbol{M}|\mathbf{X}^{obs},\mathbf{X}^{miss}) \frac{P_{\boldsymbol{\theta}}(\mathbf{X}^{miss}|\mathbf{X}^{obs})}{P_{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}}(\boldsymbol{M}|\mathbf{X}^{obs})} \overset{(5.3)}{=} P_{\boldsymbol{\theta}}(\mathbf{X}^{miss}|\mathbf{X}^{obs})$$

□

Theorem 2. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{Y} \in \mathcal{Y}^n$ be random variables, $\mathbf{M} \in \{0,1\}^{n\times d}$ be a missingness mask and $\mathbf{X}^{obs}$ denotes the observed (incomplete) data. We encode the pair $(\mathbf{X}^{obs}, \mathbf{M})$ with the random variable $\tilde{\mathbf{X}}$ with

$$\tilde{X}_{ij} = \begin{cases} X_{ij}, & M_{ij} = 0, \\ ?, & M_{ij} = 1. \end{cases}$$

Let the change in the information be defined as $\Delta := I(\mathbf{Y};\tilde{\mathbf{X}}) - I(\mathbf{Y};\mathbf{X})$, where $I(\cdot;\cdot)$ denotes the mutual information, i.e., a measure of the statistical dependence between two random variables. Then,

1. If the missingness is label-MAR, then $\Delta \leq 0$.

2. If $\mathbf{X} \in \{0,1\}^{n\times d}$ and the missingness is U-MCAR with missingness probability μ, and $s(\mathbf{X})$ is the sample sparsity as in Definition 4, then

$$-\, nd\,\mu\, h_2\big(\mathbb{E}[s(\mathbf{X})]\big) \;\le\; \Delta \;\le\; 0,$$

where $h_2(u) = -u\log u - (1-u)\log(1-u)$.

Proof. By construction $\tilde{\mathbf{X}} = g(\mathbf{X}, \mathbf{M})$ for some measurable g. Thus $(\mathbf{Y}) \to (\mathbf{X}, \mathbf{M}) \to \tilde{\mathbf{X}}$ is a Markov chain, and the data–processing inequality implies

$$I(\mathbf{Y}; \tilde{\mathbf{X}}) \;\le\; I(\mathbf{Y}; \mathbf{X}, \mathbf{M}). \tag{C.1}$$

Moreover, for any three random elements (A, B, C) we have the chain–rule identities

$$I(A; B, C) = I(A; C) + I(A; B \mid C). \tag{C.2}$$

(1) Label-MAR $\Delta \le 0$. Assume label-MAR: $\mathbb{P}(\mathbf{M} \mid \mathbf{X}, \mathbf{Y}) = \mathbb{P}(\mathbf{M} \mid \mathbf{X})$, which is equivalent to $\mathbf{Y} \perp \mathbf{M} \mid \mathbf{X}$. Applying (C.2) with $(A, B, C) = (\mathbf{Y}, \mathbf{X}, \mathbf{M})$,

$$I(\mathbf{Y}; \mathbf{X}, \mathbf{M}) \;=\; I(\mathbf{Y}; \mathbf{X}) + I(\mathbf{Y}; \mathbf{M} \mid \mathbf{X}).$$

Under label-MAR, $I(\mathbf{Y}; \mathbf{M} \mid \mathbf{X}) = 0$, hence

$$I(\mathbf{Y}; \mathbf{X}, \mathbf{M}) \;=\; I(\mathbf{Y}; \mathbf{X}). \tag{C.3}$$

Combining (C.1) and (C.3) yields

$$I(\mathbf{Y}; \tilde{\mathbf{X}}) \;\le\; I(\mathbf{Y}; \mathbf{X}) \quad\Longleftrightarrow\quad \Delta \;=\; I(\mathbf{Y}; \tilde{\mathbf{X}}) - I(\mathbf{Y}; \mathbf{X}) \;\le\; 0.$$

(2) Two-sided bound under uniform MCAR and α-β sparsity. Assume uniform MCAR: $M_{ij} \sim \text{Bernoulli}(1-\mu)$ independently of $(\mathbf{X}, \mathbf{Y})$ and i.i.d. across (i, j), and that $\mathbb{P}\big(s(\mathbf{X}) \ge \alpha\big) \ge \beta$, where $s(\mathbf{X}) = \frac{1}{nd}\sum_{i,j} \mathbb{I}\{X_{ij} = 0\}$.

Upper side. MCAR implies label-MAR, so by part (1): $\Delta \le 0$.

Lower side. We start from the chain–rule identity applied to $(A, B, C) = (\mathbf{Y}, \mathbf{X}, \tilde{\mathbf{X}})$:

$$I(\mathbf{Y}; \mathbf{X}, \tilde{\mathbf{X}}) = I(\mathbf{Y}; \tilde{\mathbf{X}}) + I(\mathbf{Y}; \mathbf{X} \mid \tilde{\mathbf{X}}) = I(\mathbf{Y}; \mathbf{X}) + I(\mathbf{Y}; \tilde{\mathbf{X}} \mid \mathbf{X}).$$

Rearranging gives

$$-\Delta \;=\; I(\mathbf{Y};\mathbf{X}) - I(\mathbf{Y};\tilde{\mathbf{X}}) \;=\; I(\mathbf{Y};\mathbf{X} \mid \tilde{\mathbf{X}}) - I(\mathbf{Y};\tilde{\mathbf{X}} \mid \mathbf{X}). \tag{C.4}$$

The second term on the right is nonnegative, hence

$$-\Delta \;\le\; I(\mathbf{Y};\mathbf{X} \mid \tilde{\mathbf{X}}). \tag{C.5}$$

Using the bound $I(U;V \mid W) \le H(V \mid W)$, we get

$$-\Delta \;\le\; H(\mathbf{X} \mid \tilde{\mathbf{X}}). \tag{C.6}$$

Index the matrix entries by a total order $\prec$ on pairs (i,j) and apply the chain rule:

$$H(\mathbf{X} \mid \tilde{\mathbf{X}}) = \sum_{(i,j)} H\big(X_{ij} \,\big|\, \tilde{\mathbf{X}},\, \{X_{kl} : (k,l) \prec (i,j)\}\big).$$

Since conditioning reduces entropy,

$$H(\mathbf{X} \mid \tilde{\mathbf{X}}) \;\le\; \sum_{i,j} H\big(X_{ij} \mid \tilde{X}_{ij}\big). \tag{C.7}$$

Fix (i,j) and denote $\pi_{ij} = \Pr[X_{ij} = 1]$. Under uniform MCAR,

$$\Pr[\tilde{X}_{ij} = ?] = \mu, \qquad \Pr[\tilde{X}_{ij} = x] = (1-\mu)\Pr[X_{ij} = x], \quad x \in \{0,1\}.$$

Hence: (i) if $\tilde{X}_{ij} \in \{0,1\}$ then X_{ij} is revealed, so $H(X_{ij} \mid \tilde{X}_{ij} \in \{0,1\}) = 0$; (ii) if $\tilde{X}_{ij} = ?$, then $\Pr[X_{ij} = 1 \mid \tilde{X}_{ij} = ?] = \pi_{ij}$ and $H(X_{ij} \mid \tilde{X}_{ij} = ?) = h_2(\pi_{ij})$. Averaging over $\tilde{X}_{ij}$ gives

$$H(X_{ij} \mid \tilde{X}_{ij}) \;=\; \mu\, h_2(\pi_{ij}). \tag{C.8}$$

Combining (C.7) and (C.8):

$$H(\mathbf{X} \mid \tilde{\mathbf{X}}) \;\le\; \sum_{i,j} \mu\, h_2(\pi_{ij}) \;=\; nd\,\mu \cdot \frac{1}{nd} \sum_{i,j} h_2(\pi_{ij}) \;\le\; nd\,\mu \cdot h_2\!\left(\frac{1}{nd}\sum_{i,j} \pi_{ij}\right),$$

since h_2 is concave. Note that

$$\frac{1}{nd}\sum_{i,j} \pi_{ij} = \frac{1}{nd}\sum_{i,j} \Pr[X_{ij} = 1] = \mathbb{E}\left[\frac{1}{nd}\sum_{i,j} \mathbb{I}\{X_{ij} = 1\}\right] = 1 - \mathbb{E}[s(\mathbf{X})].$$

Using the symmetry $h_2(u) = h_2(1-u)$, we conclude

$$H(\mathbf{X} \mid \tilde{\mathbf{X}}) \;\le\; nd\,\mu \cdot h_2\big(\mathbb{E}[s(\mathbf{X})]\big).$$

Combining with $-\Delta \le H(\mathbf{X} \mid \tilde{\mathbf{X}})$ gives

$$-\,nd\,\mu\, h_2\big(\mathbb{E}[s(\mathbf{X})]\big) \;\le\; \Delta \;\le\; 0.$$

This concludes the proof. □

C.2 Additional Results on Benchmarks and Proposed Datasets

Table C.1: Dataset statistics and feature sparsity. Classic benchmarks (CORA, CITESEER, PUBMED) exhibit extremely sparse bag-of-words features, while our proposed datasets (SYNTHETIC, AIR, ELECTRIC, TADPOLE) provide less sparse representations.

Dataset	**#Nodes**	**#Features**	**Sparsity ↓**	**Type of features**
CORA	2708	1433	0.9873	BoW (binary)
CITESEER	3327	3703	0.9915	BoW (binary)
PUBMED	19717	500	0.8998	BoW (binary)
SYNTHETIC	1000	5	0.0000	Gaussian
AIR	430	7	0.1615	Raw
ELECTRIC	2000	5	0.2000	Raw
TADPOLE	555	15	0.0000	Raw

This section presents the full plots of the results under the R1 regime introduced in Section 5.4.

Figure C.1 shows the complete set of results across all datasets, whose statistics are summarized in Table C.1. The top three rows correspond to the classic benchmarks (CORA, CITESEER, PUBMED). Consistently with Proposition 2, models maintain nearly constant F1 scores up to extremely high missingness levels ($\sim 90\%$), confirming that these benchmarks are of limited value for evaluating robustness to missing features.

The bottom four rows correspond to our proposed datasets (SYNTHETIC, AIR, ELECTRIC, TADPOLE). In these cases, performance degrades much earlier and more severely, highlighting the higher realism and difficulty of our benchmarks.

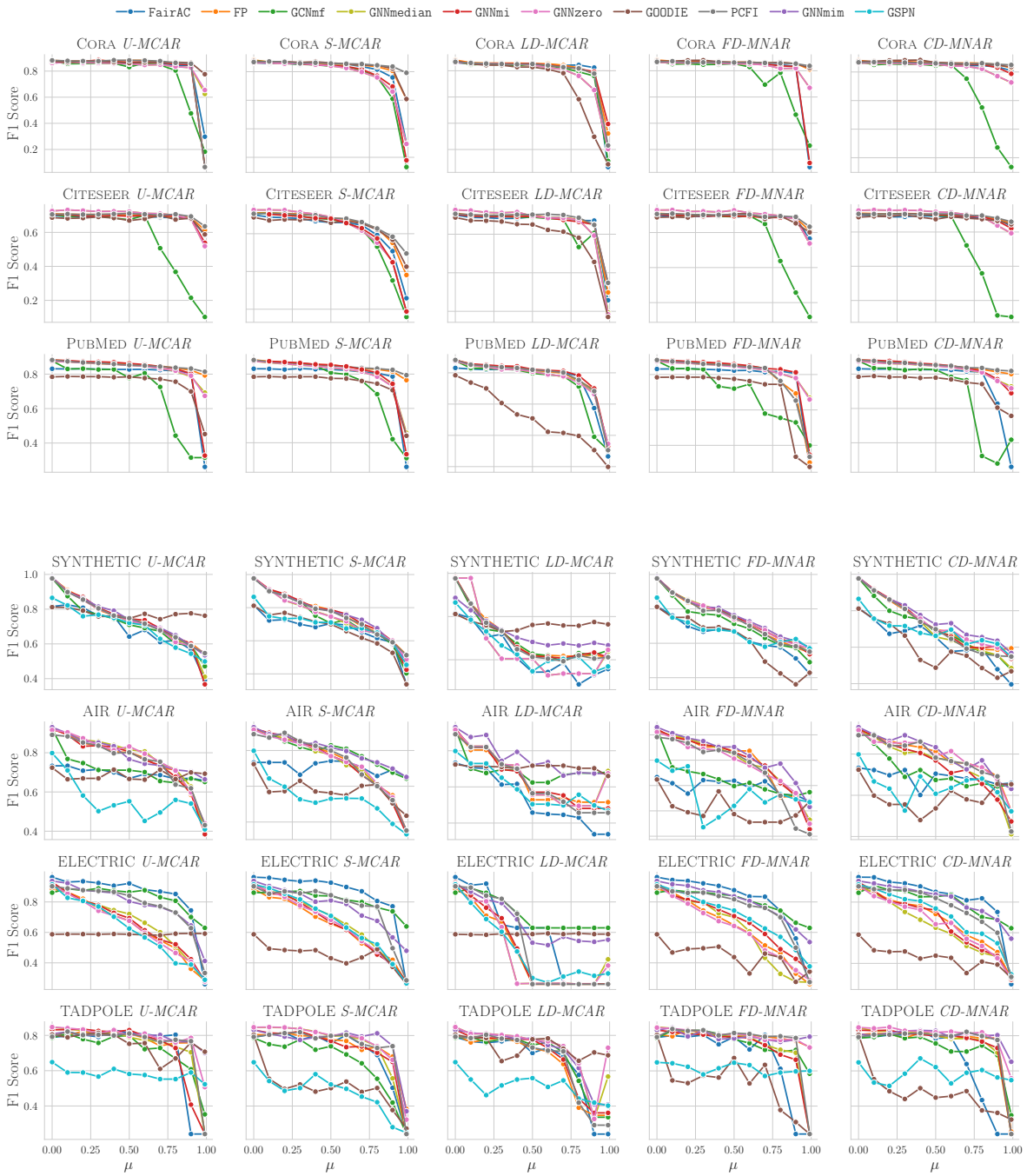


Figure C.1: F1 score as a function of feature missingness (μ) for both classic benchmarks (top three rows) and our proposed datasets (bottom four rows), under the mechanisms described in Section 5.4. Classic benchmarks show almost no degradation until extremely high μ, while the proposed datasets reveal model weaknesses at more realistic missingness levels. Tables for numeric results are in App. C.6

C.3 More challenging datasets

In Section 5.3, we introduced the synthetic and real-world datasets employed in our experiments. We now provide additional details on their construction and characteristics.

Synthetic Synthetic dataset based on a Barabási–Albert graph topology. Each node is associated with five real-valued features sampled from a Gaussian distribution. Node labels are generated deterministically by applying a fixed two-layer GCN with hard-coded weights to the complete feature matrix. This construction ensures that the ground-truth labeling function is fully expressible by a GNN, allowing models to achieve near-perfect accuracy in the absence of missingness. The resulting task is a binary node classification problem, with classes separated according to structured feature combinations defined by the fixed GCN. This controlled setup provides a principled testbed to isolate and analyze the effects of different missingness mechanisms, while preserving a well-defined ground truth.

Air Dataset [115] built from a network of air quality monitoring stations deployed in an urban area. Each node corresponds to a station and is associated with a set of environmental measurements. The node features include both air pollutant concentrations (`CO`, $\texttt{NO}_2$, $\texttt{PM}_{10}$, $\texttt{O}_3$, $\texttt{SO}_2$) and meteorological variables (`temperature`, `humidity`, `wind speed`, `wind direction`). Edges are constructed based on the geographical distance between stations, with two nodes connected if their distance is below a given threshold. The target variable is derived from the $\texttt{PM}_{2.5}$ concentration, which is discretized into three balanced categories (low, medium, high) according to the distribution of observed values. This formulation allows us to frame the problem as a semi-supervised node classification task with three classes.

Electric Dataset [13, 7] derived from a large-scale model of the Texas power grid. Nodes correspond to buses in the electrical network, each enriched with both structural and operational attributes. The node features include identifiers (`area`, `zone`), electrical measurements (`voltage magnitude`, `voltage angle`), and a topological property (`betweenness centrality`). Edges are constructed directly from the transmission lines specified in the raw grid data, connecting pairs of buses. The classification target is the nominal voltage level of each bus (`base kV`), which we discretize into three categories: low voltage (<100 kV), medium voltage (100–200 kV), and high voltage (>200 kV). This setup results in a three-class node classification problem reflecting operational conditions across the grid.

Tadpole The Tadpoledataset [116] originates from the TADPOLE challenge, which provides longitudinal clinical and imaging data for patients at risk of developing Alzheimer's disease. In our graph formulation, each node corresponds to a patient and is associated

with a set of features encompassing clinical scores, cerebrospinal fluid (CSF) biomarkers, and neuroimaging measures such as MRI- and PET-derived variables. Since the original dataset does not provide graph connectivity, we construct edges using a k-nearest neighbors approach over the most informative biomarkers, so that patients with similar profiles are connected. The target variable is the diagnostic label, categorized into three classes (cognitively normal, mild cognitive impairment, Alzheimer's disease). This results in a semi-supervised node classification problem where the goal is to predict the diagnostic status of patients based on multimodal biomedical features and patient similarity structure.

Table C.1 reports, for each dataset, the number of nodes, number of features, feature sparsity, and the type of features. While the number of nodes and features may seem small compared to standard benchmark graph datasets, we emphasize that using real features (as in Air, Electric, and Tadpole) is more realistic in the context of feature missingness. In fact, it is not meaningful to study missingness on pre-computed embeddings, since embeddings are typically high-dimensional representations mapped to wide feature spaces and are not expected to exhibit missingness in practice.

C.4 Experimental Details

All baseline and competitor methods are implemented using the official code released in their respective repositories, following the recommended training protocols and hyperparameter settings. For `GNNmi` and `GNNmim`, we adopt a standard GNN architecture where the convolutional layer type (Table C.2), the number of layers (1-3), the learning rate (10^{-4}-10^{-2}), and the weight decay (10^{-5}-10^{-3}) are tuned via grid search on the validation set. All models are trained on the same data splits with early stopping to ensure a fair comparison.

Table C.2: Best GNN encoder selected within `GNNmim` for each dataset and missingness mechanism.

Dataset	*U-MCAR*	*S-MCAR*	*LD-MCAR*	*FD-MNAR*	*CD-MNAR*
Synthetic	`GCN`	`GCN`	`GraphSAGE`	`GCN`	`GCN`
Air	`GraphSAGE`	`GraphSAGE`	`GraphSAGE`	`GraphSAGE`	`GraphSAGE`
Electric	`GIN`	`GIN`	`GraphSAGE`	`GIN`	`GIN`
Tadpole	`GCN`	`GraphSAGE`	`GraphSAGE`	`GraphSAGE`	`GCN`

C.5 Scaling the Synthetic Dataset

In this section, we analyze what happens when either the number of features or the number of nodes in the synthetic dataset is increased. To this end, we constructed three additional synthetic datasets (SYNTHETIC2, SYNTHETIC3, SYNTHETIC4) following the same design principles as SYNTHETIC. Table C.3 reports their statistics.

As shown in Figure C.2, the behavior of the models in this larger-scale setting is consistent with the one observed in our original setup. In this case, we experimented with the *uniform random missingness* mechanism, and we observe a monotonic decrease in performance for all models as the missingness rate μ increases. This confirms that dataset size does not affect the overall trend of performance degradation under feature missingness.

To further support this point, we also report the runtime and GPU memory consumption of all models on both the main synthetic dataset (SYNTHETIC) and its larger-scale counterpart (SYNTHETIC3), which features an increased number of features. As shown in Table C.4, the runtime and memory requirements remain substantially stable across datasets, with negligible variations between models. This behavior confirms that our approach scales efficiently with the dataset size, as it only involves a standard GNN architecture augmented with a simple MIM mask concatenated to the input features, introducing minimal computational overhead.

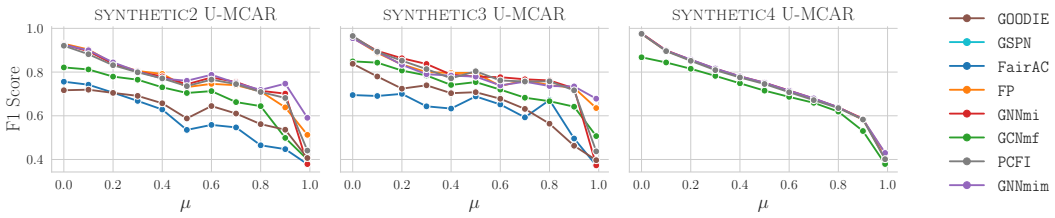


Figure C.2: F1 score as a function of feature missingness (μ) for additional synthetic datasets generated with the same procedure as SYNTHETIC, but with either an increased number of nodes or features. For SYNTHETIC4, the `FairAC`model is not reported since training exceeded the 12-hour time limit, while `GOODIE` is excluded due to out-of-memory errors.

C.6 Complete Result Tables – R1 Regime

Table C.3: Datasets information.

Dataset	**#Nodes**	**#Features**	**Sparsity ↓**	**Type of features**
SYNTHETIC	1000	5	0.0000	Gaussian
SYNTHETIC2	1000	20	0.0000	Gaussian
SYNTHETIC3	1000	50	0.0000	Gaussian
SYNTHETIC4	50000	5	0.0000	Gaussian

Table C.4: Runtime and GPU peak memory consumption for the main synthetic dataset (SYNTHETIC) and the scaled version (SYNTHETIC3). Each value corresponds to the average across all missingness levels under the UMCAR mechanism.

Model	SYNTHETIC		SYNTHETIC4	
	Runtime [s] ↓	GPU Mem [GB] ↓	Runtime [s] ↓	GPU Mem [GB] ↓
`GNNmi`	1.7	0.03	5.3	0.78
`GNNzero`	1.6	0.03	5.0	0.77
`GNNmedian`	1.6	0.03	5.0	0.77
`GNNmim`	1.8	0.03	6.3	0.77
`GCNmf`	4.5	0.02	28.0	0.53
`FP`	1.5	0.02	5.3	0.77
`PCFI`	1.8	0.02	5.2	0.77
`FairAC`	3.9	0.04	–	–
`GSPN`	55.0	0.03	150.0	0.84
`GOODIE`	2.3	0.06	–	–

Table C.5: F1 scores for CORA under mechanism *U-MCAR* and varying μ (`GSPN`is not reported as it is not designed for categorical features).

μ	`GOODIE`	`FairAC`	`FP`	`GNNmi`	`GCNmf`	`PCFI`	`GNNzero`	`GNNmedian`
0.00	0.875 (± 0.00)	0.863 (± 0.01)	0.882 (± 0.00)	0.873 (± 0.00)	0.875 (± 0.00)	**0.882 (± 0.00)**	0.862 (± 0.02)	0.862 (± 0.02)
0.10	0.867 (± 0.00)	0.866 (± 0.00)	0.877 (± 0.00)	0.876 (± 0.00)	0.856 (± 0.00)	**0.878 (± 0.00)**	0.868 (± 0.01)	0.868 (± 0.01)
0.20	0.875 (± 0.00)	0.862 (± 0.00)	**0.878 (± 0.00)**	0.873 (± 0.00)	0.858 (± 0.00)	0.877 (± 0.00)	0.864 (± 0.02)	0.864 (± 0.02)
0.30	0.873 (± 0.00)	0.865 (± 0.00)	0.881 (± 0.00)	**0.885 (± 0.00)**	0.860 (± 0.00)	0.876 (± 0.00)	0.863 (± 0.01)	0.863 (± 0.01)
0.40	0.869 (± 0.00)	0.857 (± 0.00)	0.878 (± 0.00)	0.873 (± 0.00)	0.860 (± 0.00)	**0.884 (± 0.00)**	0.860 (± 0.02)	0.860 (± 0.02)
0.50	0.861 (± 0.00)	0.856 (± 0.00)	**0.882 (± 0.00)**	0.867 (± 0.00)	0.831 (± 0.00)	0.882 (± 0.00)	0.856 (± 0.01)	0.856 (± 0.01)
0.60	0.866 (± 0.00)	0.847 (± 0.00)	**0.882 (± 0.00)**	0.871 (± 0.00)	0.862 (± 0.00)	0.881 (± 0.00)	0.847 (± 0.01)	0.847 (± 0.01)
0.70	0.866 (± 0.00)	0.858 (± 0.00)	0.869 (± 0.00)	0.865 (± 0.00)	0.847 (± 0.00)	**0.877 (± 0.00)**	0.849 (± 0.01)	0.849 (± 0.01)
0.80	**0.868 (± 0.00)**	0.843 (± 0.00)	0.864 (± 0.00)	0.854 (± 0.00)	0.805 (± 0.00)	0.863 (± 0.00)	0.835 (± 0.01)	0.835 (± 0.01)
0.90	**0.864 (± 0.00)**	0.845 (± 0.00)	0.860 (± 0.00)	0.848 (± 0.00)	0.476 (± 0.00)	0.856 (± 0.00)	0.826 (± 0.00)	0.826 (± 0.00)
0.99	**0.776 (± 0.00)**	0.298 (± 0.00)	0.066 (± 0.00)	0.066 (± 0.00)	0.183 (± 0.00)	0.065 (± 0.00)	0.655 (± 0.03)	0.625 (± 0.02)

Table C.6: F1 scores for CORA under mechanism *S-MCAR* and varying μ (`GSPN`is not reported as it is not designed for categorical features).

μ	`GOODIE`	`FairAC`	`FP`	`GNNmi`	`GCNmf`	`PCFI`	`GNNzero`	`GNNmedian`
0.00	0.875 (± 0.00)	0.863 (± 0.01)	**0.882 (± 0.00)**	0.872 (± 0.00)	0.875 (± 0.00)	0.868 (± 0.00)	0.862 (± 0.02)	0.862 (± 0.02)
0.10	0.868 (± 0.00)	0.857 (± 0.00)	0.869 (± 0.00)	0.862 (± 0.00)	0.869 (± 0.00)	**0.872 (± 0.00)**	0.862 (± 0.02)	0.862 (± 0.02)
0.20	**0.872 (± 0.00)**	0.860 (± 0.00)	0.863 (± 0.00)	0.863 (± 0.00)	0.858 (± 0.00)	0.869 (± 0.00)	0.856 (± 0.02)	0.856 (± 0.02)
0.30	**0.865 (± 0.00)**	0.850 (± 0.00)	0.854 (± 0.00)	0.855 (± 0.00)	0.852 (± 0.00)	0.858 (± 0.00)	0.857 (± 0.02)	0.857 (± 0.02)
0.40	**0.870 (± 0.00)**	0.857 (± 0.00)	0.859 (± 0.00)	0.848 (± 0.00)	0.848 (± 0.00)	0.862 (± 0.00)	0.849 (± 0.02)	0.849 (± 0.02)
0.50	**0.862 (± 0.00)**	0.854 (± 0.00)	0.854 (± 0.00)	0.844 (± 0.00)	0.839 (± 0.00)	0.858 (± 0.00)	0.841 (± 0.01)	0.841 (± 0.01)
0.60	0.855 (± 0.00)	0.854 (± 0.00)	0.853 (± 0.00)	0.837 (± 0.00)	0.837 (± 0.00)	**0.856 (± 0.00)**	0.826 (± 0.01)	0.826 (± 0.01)
0.70	0.847 (± 0.00)	0.836 (± 0.00)	0.845 (± 0.00)	0.817 (± 0.00)	0.807 (± 0.00)	**0.854 (± 0.00)**	0.798 (± 0.02)	0.798 (± 0.02)
0.80	**0.845 (± 0.00)**	0.815 (± 0.00)	0.836 (± 0.00)	0.772 (± 0.00)	0.764 (± 0.00)	0.845 (± 0.00)	0.760 (± 0.02)	0.760 (± 0.02)
0.90	0.822 (± 0.00)	0.760 (± 0.00)	0.806 (± 0.00)	0.696 (± 0.00)	0.610 (± 0.00)	**0.836 (± 0.00)**	0.661 (± 0.02)	0.661 (± 0.02)
0.99	0.609 (± 0.00)	0.300 (± 0.00)	0.606 (± 0.00)	0.179 (± 0.00)	0.132 (± 0.00)	**0.792 (± 0.00)**	0.294 (± 0.05)	0.294 (± 0.05)

Table C.7: F1 scores for CORA under mechanism *CD-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.875 (± 0.00)	0.863 (± 0.01)	**0.882 (± 0.00)**	0.873 (± 0.00)	0.875 (± 0.00)	0.868 (± 0.00)	0.862 (± 0.02)	0.862 (± 0.02)
0.10	0.852 (± 0.00)	0.851 (± 0.00)	**0.862 (± 0.00)**	0.857 (± 0.00)	0.846 (± 0.00)	0.860 (± 0.00)	0.858 (± 0.02)	0.858 (± 0.02)
0.20	0.843 (± 0.00)	0.854 (± 0.00)	**0.859 (± 0.00)**	0.854 (± 0.00)	0.850 (± 0.00)	0.855 (± 0.00)	0.854 (± 0.02)	0.854 (± 0.02)
0.30	0.843 (± 0.00)	0.856 (± 0.00)	**0.859 (± 0.00)**	0.855 (± 0.00)	0.846 (± 0.00)	0.852 (± 0.00)	0.853 (± 0.02)	0.853 (± 0.02)
0.40	0.828 (± 0.00)	0.854 (± 0.00)	**0.858 (± 0.00)**	0.853 (± 0.00)	0.838 (± 0.00)	0.849 (± 0.00)	0.849 (± 0.02)	0.849 (± 0.02)
0.50	0.828 (± 0.00)	0.854 (± 0.00)	0.855 (± 0.00)	**0.855 (± 0.00)**	0.848 (± 0.00)	0.852 (± 0.00)	0.844 (± 0.02)	0.844 (± 0.02)
0.60	0.812 (± 0.00)	0.847 (± 0.00)	**0.853 (± 0.00)**	0.844 (± 0.00)	0.837 (± 0.00)	0.841 (± 0.00)	0.825 (± 0.02)	0.825 (± 0.02)
0.70	0.782 (± 0.00)	0.841 (± 0.00)	**0.842 (± 0.00)**	0.831 (± 0.00)	0.822 (± 0.00)	0.827 (± 0.00)	0.810 (± 0.02)	0.810 (± 0.02)
0.80	0.584 (± 0.00)	**0.844 (± 0.00)**	0.822 (± 0.00)	0.815 (± 0.00)	0.792 (± 0.00)	0.818 (± 0.00)	0.761 (± 0.01)	0.761 (± 0.01)
0.90	0.297 (± 0.00)	**0.824 (± 0.00)**	0.777 (± 0.00)	0.793 (± 0.00)	0.760 (± 0.00)	0.778 (± 0.00)	0.653 (± 0.02)	0.654 (± 0.02)
0.99	0.088 (± 0.00)	0.066 (± 0.00)	0.322 (± 0.00)	**0.395 (± 0.00)**	0.113 (± 0.00)	0.231 (± 0.00)	0.204 (± 0.03)	0.204 (± 0.03)

Table C.8: F1 scores for CORA under mechanism *FD-MNAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.875 (± 0.00)	0.863 (± 0.01)	**0.882 (± 0.00)**	0.873 (± 0.00)	0.875 (± 0.00)	0.868 (± 0.00)	0.864 (± 0.02)	0.864 (± 0.02)
0.10	0.872 (± 0.01)	0.862 (± 0.01)	**0.873 (± 0.01)**	0.868 (± 0.01)	0.851 (± 0.01)	0.873 (± 0.00)	0.862 (± 0.02)	0.862 (± 0.02)
0.20	**0.879 (± 0.00)**	0.870 (± 0.01)	0.874 (± 0.00)	0.865 (± 0.01)	0.853 (± 0.01)	0.863 (± 0.01)	0.858 (± 0.01)	0.858 (± 0.01)
0.30	**0.880 (± 0.00)**	0.864 (± 0.01)	0.869 (± 0.00)	0.867 (± 0.01)	0.847 (± 0.01)	0.864 (± 0.01)	0.864 (± 0.01)	0.864 (± 0.01)
0.40	**0.869 (± 0.01)**	0.855 (± 0.01)	0.864 (± 0.01)	0.856 (± 0.01)	0.849 (± 0.00)	0.866 (± 0.01)	0.858 (± 0.02)	0.858 (± 0.02)
0.50	0.865 (± 0.01)	0.860 (± 0.01)	**0.866 (± 0.01)**	0.859 (± 0.01)	0.854 (± 0.01)	0.863 (± 0.01)	0.854 (± 0.02)	0.854 (± 0.02)
0.60	**0.866 (± 0.01)**	0.853 (± 0.01)	0.865 (± 0.01)	0.863 (± 0.01)	0.829 (± 0.02)	0.864 (± 0.01)	0.851 (± 0.01)	0.851 (± 0.01)
0.70	0.859 (± 0.01)	0.847 (± 0.00)	**0.862 (± 0.01)**	0.853 (± 0.00)	0.695 (± 0.14)	0.860 (± 0.00)	0.846 (± 0.01)	0.846 (± 0.01)
0.80	**0.865 (± 0.01)**	0.845 (± 0.01)	0.861 (± 0.01)	0.837 (± 0.00)	0.785 (± 0.05)	0.857 (± 0.01)	0.817 (± 0.02)	0.817 (± 0.02)
0.90	0.854 (± 0.01)	0.833 (± 0.01)	**0.855 (± 0.00)**	0.833 (± 0.00)	0.465 (± 0.21)	0.854 (± 0.01)	0.819 (± 0.01)	0.819 (± 0.01)
0.99	0.822 (± 0.01)	0.066 (± 0.00)	0.810 (± 0.02)	0.098 (± 0.01)	0.230 (± 0.05)	**0.837 (± 0.02)**	0.670 (± 0.02)	0.670 (± 0.02)

Table C.9: F1 scores for CORA under mechanism *CD-MNAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.875 (± 0.00)	0.863 (± 0.01)	**0.882 (± 0.00)**	0.873 (± 0.00)	0.875 (± 0.00)	0.868 (± 0.00)	0.863 (± 0.02)	0.863 (± 0.02)
0.10	**0.875 (± 0.00)**	0.864 (± 0.01)	0.870 (± 0.01)	0.862 (± 0.01)	0.850 (± 0.00)	0.869 (± 0.01)	0.863 (± 0.02)	0.863 (± 0.02)
0.20	**0.881 (± 0.01)**	0.865 (± 0.00)	0.874 (± 0.01)	0.868 (± 0.01)	0.856 (± 0.01)	0.869 (± 0.01)	0.860 (± 0.02)	0.860 (± 0.02)
0.30	**0.882 (± 0.00)**	0.858 (± 0.00)	0.873 (± 0.00)	0.871 (± 0.01)	0.854 (± 0.00)	0.866 (± 0.01)	0.860 (± 0.02)	0.860 (± 0.02)
0.40	**0.884 (± 0.01)**	0.862 (± 0.01)	0.870 (± 0.00)	0.864 (± 0.00)	0.853 (± 0.01)	0.865 (± 0.01)	0.853 (± 0.02)	0.853 (± 0.02)
0.50	0.867 (± 0.01)	0.852 (± 0.01)	**0.867 (± 0.00)**	0.861 (± 0.00)	0.844 (± 0.02)	0.861 (± 0.01)	0.855 (± 0.02)	0.855 (± 0.02)
0.60	**0.864 (± 0.00)**	0.847 (± 0.00)	0.860 (± 0.01)	0.856 (± 0.01)	0.849 (± 0.00)	0.857 (± 0.00)	0.842 (± 0.02)	0.842 (± 0.02)
0.70	0.860 (± 0.01)	0.845 (± 0.01)	**0.864 (± 0.01)**	0.852 (± 0.01)	0.753 (± 0.12)	0.856 (± 0.01)	0.840 (± 0.02)	0.840 (± 0.02)
0.80	0.853 (± 0.01)	0.844 (± 0.02)	**0.862 (± 0.01)**	0.852 (± 0.01)	0.551 (± 0.10)	0.861 (± 0.01)	0.822 (± 0.03)	0.822 (± 0.03)
0.90	0.848 (± 0.01)	0.835 (± 0.01)	0.852 (± 0.00)	0.831 (± 0.01)	0.271 (± 0.23)	**0.855 (± 0.01)**	0.771 (± 0.03)	0.771 (± 0.03)
0.99	0.836 (± 0.01)	0.810 (± 0.01)	0.828 (± 0.01)	0.788 (± 0.02)	0.135 (± 0.05)	**0.849 (± 0.01)**	0.727 (± 0.04)	0.725 (± 0.03)

Table C.10: F1 scores for CITESEER under mechanism *U-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.687 (± 0.00)	0.700 (± 0.00)	0.710 (± 0.02)	0.704 (± 0.02)	0.707 (± 0.00)	0.706 (± 0.02)	**0.726 (± 0.02)**	0.726 (± 0.02)
0.10	0.682 (± 0.00)	0.693 (± 0.00)	0.707 (± 0.00)	0.705 (± 0.00)	0.692 (± 0.00)	0.708 (± 0.00)	**0.732 (± 0.02)**	0.732 (± 0.02)
0.20	0.684 (± 0.00)	0.693 (± 0.00)	0.706 (± 0.00)	0.695 (± 0.00)	0.698 (± 0.00)	0.705 (± 0.00)	**0.728 (± 0.02)**	0.728 (± 0.02)
0.30	0.691 (± 0.00)	0.691 (± 0.00)	0.705 (± 0.00)	0.696 (± 0.00)	0.697 (± 0.00)	0.706 (± 0.00)	**0.723 (± 0.03)**	0.723 (± 0.03)
0.40	0.685 (± 0.00)	0.700 (± 0.00)	0.706 (± 0.00)	0.698 (± 0.00)	0.684 (± 0.00)	0.708 (± 0.00)	**0.724 (± 0.02)**	0.724 (± 0.02)
0.50	0.669 (± 0.00)	0.697 (± 0.00)	0.702 (± 0.00)	0.695 (± 0.00)	0.675 (± 0.00)	0.711 (± 0.00)	**0.722 (± 0.02)**	0.722 (± 0.02)
0.60	0.680 (± 0.00)	0.695 (± 0.00)	0.697 (± 0.00)	0.699 (± 0.00)	0.700 (± 0.00)	0.707 (± 0.00)	**0.712 (± 0.02)**	0.712 (± 0.02)
0.70	0.699 (± 0.00)	0.688 (± 0.00)	0.694 (± 0.00)	0.700 (± 0.00)	0.507 (± 0.00)	0.701 (± 0.00)	**0.710 (± 0.02)**	0.710 (± 0.02)
0.80	0.675 (± 0.00)	0.687 (± 0.00)	0.694 (± 0.00)	0.696 (± 0.00)	0.368 (± 0.00)	**0.707 (± 0.00)**	0.701 (± 0.01)	0.701 (± 0.01)
0.90	0.684 (± 0.00)	0.680 (± 0.00)	0.686 (± 0.00)	0.680 (± 0.00)	0.215 (± 0.00)	**0.694 (± 0.00)**	0.678 (± 0.02)	0.678 (± 0.02)
0.99	0.588 (± 0.00)	0.584 (± 0.00)	0.613 (± 0.00)	0.539 (± 0.00)	0.102 (± 0.00)	**0.636 (± 0.00)**	0.519 (± 0.03)	0.519 (± 0.03)

Table C.11: F1 scores for CITESEER under mechanism *S-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.687 (± 0.00)	0.700 (± 0.00)	0.710 (± 0.02)	-	0.707 (± 0.00)	0.706 (± 0.02)	**0.726 (± 0.02)**	0.726 (± 0.02)
0.10	0.670 (± 0.00)	0.688 (± 0.00)	0.711 (± 0.00)	0.703 (± 0.00)	0.708 (± 0.00)	0.708 (± 0.00)	**0.726 (± 0.03)**	0.726 (± 0.03)
0.20	0.675 (± 0.00)	0.685 (± 0.00)	0.707 (± 0.00)	0.697 (± 0.00)	0.707 (± 0.00)	0.706 (± 0.00)	**0.725 (± 0.03)**	0.725 (± 0.03)
0.30	0.673 (± 0.00)	0.681 (± 0.00)	0.705 (± 0.00)	0.692 (± 0.00)	0.693 (± 0.00)	0.701 (± 0.00)	**0.714 (± 0.02)**	0.714 (± 0.02)
0.40	0.677 (± 0.00)	0.667 (± 0.00)	0.698 (± 0.00)	0.682 (± 0.00)	0.682 (± 0.00)	0.698 (± 0.00)	**0.704 (± 0.03)**	0.704 (± 0.03)
0.50	0.658 (± 0.00)	0.659 (± 0.00)	0.685 (± 0.00)	0.680 (± 0.00)	0.676 (± 0.00)	0.683 (± 0.00)	**0.689 (± 0.03)**	0.689 (± 0.03)
0.60	0.667 (± 0.00)	0.659 (± 0.00)	0.676 (± 0.00)	0.656 (± 0.00)	0.659 (± 0.00)	**0.680 (± 0.00)**	0.659 (± 0.02)	0.659 (± 0.02)
0.70	0.655 (± 0.00)	0.646 (± 0.00)	0.656 (± 0.00)	0.629 (± 0.00)	0.624 (± 0.00)	**0.662 (± 0.00)**	0.617 (± 0.02)	0.617 (± 0.02)
0.80	0.621 (± 0.00)	0.593 (± 0.00)	**0.629 (± 0.00)**	0.575 (± 0.00)	0.531 (± 0.00)	0.628 (± 0.00)	0.553 (± 0.03)	0.553 (± 0.03)
0.90	0.568 (± 0.00)	0.508 (± 0.00)	0.552 (± 0.00)	0.449 (± 0.00)	0.352 (± 0.00)	**0.584 (± 0.00)**	0.455 (± 0.03)	0.455 (± 0.03)
0.99	0.425 (± 0.00)	0.258 (± 0.00)	0.381 (± 0.00)	0.188 (± 0.00)	0.159 (± 0.00)	**0.495 (± 0.00)**	0.186 (± 0.01)	0.186 (± 0.01)

Table C.12: F1 scores for CITESEER under mechanism *CD-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.687 (± 0.00)	0.700 (± 0.00)	0.710 (± 0.02)	0.704 (± 0.02)	0.707 (± 0.00)	0.706 (± 0.02)	**0.726 (± 0.02)**	0.726 (± 0.02)
0.10	0.671 (± 0.00)	0.687 (± 0.00)	0.698 (± 0.00)	0.694 (± 0.00)	0.693 (± 0.00)	0.702 (± 0.00)	**0.723 (± 0.02)**	0.723 (± 0.02)
0.20	0.670 (± 0.00)	0.686 (± 0.00)	0.699 (± 0.00)	0.691 (± 0.00)	0.696 (± 0.00)	0.698 (± 0.00)	**0.713 (± 0.02)**	0.713 (± 0.02)
0.30	0.666 (± 0.00)	0.682 (± 0.00)	0.697 (± 0.00)	0.691 (± 0.00)	0.694 (± 0.00)	0.699 (± 0.00)	**0.711 (± 0.03)**	0.711 (± 0.03)
0.40	0.652 (± 0.00)	0.683 (± 0.00)	0.698 (± 0.00)	0.691 (± 0.00)	0.688 (± 0.00)	0.701 (± 0.00)	**0.715 (± 0.02)**	0.715 (± 0.02)
0.50	0.650 (± 0.00)	0.690 (± 0.00)	0.699 (± 0.00)	0.693 (± 0.00)	0.688 (± 0.00)	**0.702 (± 0.00)**	0.694 (± 0.02)	0.694 (± 0.02)
0.60	0.622 (± 0.00)	0.686 (± 0.00)	0.685 (± 0.00)	0.685 (± 0.00)	0.681 (± 0.00)	**0.704 (± 0.00)**	0.684 (± 0.02)	0.684 (± 0.02)
0.70	0.613 (± 0.00)	0.687 (± 0.00)	0.686 (± 0.00)	0.674 (± 0.00)	0.677 (± 0.00)	**0.700 (± 0.00)**	0.685 (± 0.03)	0.685 (± 0.03)
0.80	0.582 (± 0.00)	0.671 (± 0.00)	0.677 (± 0.00)	0.664 (± 0.00)	0.534 (± 0.00)	**0.686 (± 0.00)**	0.674 (± 0.02)	0.674 (± 0.02)
0.90	0.456 (± 0.00)	**0.671 (± 0.00)**	0.650 (± 0.00)	0.650 (± 0.00)	0.607 (± 0.00)	0.648 (± 0.00)	0.593 (± 0.02)	0.593 (± 0.02)
0.99	0.171 (± 0.00)	0.257 (± 0.00)	0.298 (± 0.00)	0.346 (± 0.00)	0.195 (± 0.00)	**0.348 (± 0.00)**	0.184 (± 0.02)	0.194 (± 0.03)

Table C.13: F1 scores for CITESEER under mechanism *FD-MNAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.687 (± 0.00)	0.700 (± 0.00)	0.710 (± 0.02)	0.704 (± 0.02)	0.707 (± 0.00)	0.706 (± 0.02)	**0.728 (± 0.02)**	0.728 (± 0.02)
0.10	0.689 (± 0.03)	0.691 (± 0.03)	0.706 (± 0.02)	0.699 (± 0.02)	0.699 (± 0.02)	0.708 (± 0.03)	**0.729 (± 0.02)**	0.729 (± 0.02)
0.20	0.686 (± 0.02)	0.698 (± 0.03)	0.703 (± 0.02)	0.697 (± 0.02)	0.696 (± 0.02)	0.704 (± 0.02)	**0.720 (± 0.02)**	0.720 (± 0.02)
0.30	0.701 (± 0.04)	0.690 (± 0.03)	0.701 (± 0.03)	0.693 (± 0.02)	0.704 (± 0.02)	0.700 (± 0.03)	**0.721 (± 0.03)**	0.721 (± 0.03)
0.40	0.696 (± 0.04)	0.699 (± 0.04)	0.695 (± 0.02)	0.695 (± 0.02)	0.692 (± 0.03)	0.701 (± 0.03)	**0.717 (± 0.02)**	0.717 (± 0.02)
0.50	0.707 (± 0.03)	0.688 (± 0.04)	0.698 (± 0.03)	0.693 (± 0.03)	0.690 (± 0.02)	0.702 (± 0.03)	**0.727 (± 0.02)**	0.727 (± 0.02)
0.60	0.708 (± 0.02)	0.694 (± 0.03)	0.691 (± 0.03)	0.693 (± 0.03)	0.696 (± 0.02)	0.702 (± 0.03)	**0.712 (± 0.03)**	0.712 (± 0.03)
0.70	0.678 (± 0.04)	0.688 (± 0.03)	0.688 (± 0.03)	0.686 (± 0.02)	0.649 (± 0.03)	0.690 (± 0.04)	**0.705 (± 0.02)**	0.705 (± 0.02)
0.80	0.695 (± 0.03)	0.689 (± 0.04)	0.689 (± 0.02)	0.685 (± 0.02)	0.437 (± 0.27)	0.694 (± 0.03)	**0.696 (± 0.03)**	0.696 (± 0.03)
0.90	0.653 (± 0.03)	0.681 (± 0.04)	0.682 (± 0.02)	0.687 (± 0.03)	0.257 (± 0.17)	**0.689 (± 0.02)**	0.676 (± 0.02)	0.676 (± 0.02)
0.99	0.601 (± 0.01)	0.566 (± 0.01)	0.611 (± 0.01)	0.535 (± 0.02)	0.118 (± 0.04)	**0.633 (± 0.01)**	0.538 (± 0.03)	0.538 (± 0.03)

Table C.14: F1 scores for CITESEER under mechanism *CD-MNAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.687 (± 0.00)	0.700 (± 0.05)	0.710 (± 0.02)	0.704 (± 0.02)	0.707 (± 0.00)	0.706 (± 0.02)	**0.726 (± 0.02)**	0.726 (± 0.02)
0.10	0.692 (± 0.04)	0.696 (± 0.04)	0.708 (± 0.02)	0.705 (± 0.02)	0.702 (± 0.03)	0.705 (± 0.02)	**0.729 (± 0.02)**	0.729 (± 0.02)
0.20	0.690 (± 0.04)	0.689 (± 0.04)	0.703 (± 0.03)	0.702 (± 0.02)	0.705 (± 0.02)	0.704 (± 0.02)	**0.727 (± 0.02)**	0.727 (± 0.02)
0.30	0.700 (± 0.02)	0.689 (± 0.04)	0.708 (± 0.03)	0.706 (± 0.02)	0.708 (± 0.02)	0.705 (± 0.02)	**0.728 (± 0.02)**	0.728 (± 0.02)
0.40	0.687 (± 0.04)	0.695 (± 0.04)	0.707 (± 0.03)	0.704 (± 0.02)	0.703 (± 0.03)	0.704 (± 0.03)	**0.725 (± 0.02)**	0.725 (± 0.02)
0.50	0.675 (± 0.03)	0.692 (± 0.03)	0.699 (± 0.03)	0.700 (± 0.03)	0.697 (± 0.02)	0.706 (± 0.03)	**0.718 (± 0.02)**	0.718 (± 0.02)
0.60	0.689 (± 0.03)	0.689 (± 0.03)	0.702 (± 0.03)	0.699 (± 0.03)	0.693 (± 0.03)	0.706 (± 0.03)	**0.714 (± 0.02)**	0.714 (± 0.02)
0.70	0.681 (± 0.03)	0.685 (± 0.03)	0.692 (± 0.03)	0.691 (± 0.03)	0.522 (± 0.20)	0.696 (± 0.03)	**0.702 (± 0.03)**	0.702 (± 0.03)
0.80	0.676 (± 0.05)	0.685 (± 0.03)	0.690 (± 0.03)	0.689 (± 0.02)	0.359 (± 0.15)	**0.696 (± 0.04)**	0.689 (± 0.03)	0.689 (± 0.03)
0.90	0.665 (± 0.02)	**0.681 (± 0.03)**	0.677 (± 0.03)	0.666 (± 0.03)	0.113 (± 0.06)	0.681 (± 0.03)	0.638 (± 0.02)	0.638 (± 0.02)
0.99	0.645 (± 0.03)	0.631 (± 0.02)	0.652 (± 0.02)	0.621 (± 0.02)	0.104 (± 0.06)	**0.660 (± 0.02)**	0.593 (± 0.03)	0.592 (± 0.03)

Table C.15: F1 scores for PubMed under mechanism *U-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.784 (± 0.01)	0.831 (± 0.00)	**0.883 (± 0.00)**	0.881 (± 0.00)	0.877 (± 0.00)	0.882 (± 0.00)	0.875 (± 0.00)	0.875 (± 0.00)
0.10	0.787 (± 0.00)	0.830 (± 0.00)	0.877 (± 0.00)	**0.879 (± 0.00)**	0.830 (± 0.00)	0.874 (± 0.00)	0.871 (± 0.00)	0.871 (± 0.00)
0.20	0.786 (± 0.00)	0.831 (± 0.00)	0.868 (± 0.00)	**0.873 (± 0.00)**	0.832 (± 0.00)	0.868 (± 0.00)	0.866 (± 0.00)	0.866 (± 0.00)
0.30	0.785 (± 0.00)	0.830 (± 0.00)	0.870 (± 0.00)	**0.872 (± 0.00)**	0.827 (± 0.00)	0.864 (± 0.00)	0.862 (± 0.00)	0.860 (± 0.00)
0.40	0.782 (± 0.00)	0.828 (± 0.00)	0.861 (± 0.00)	**0.869 (± 0.00)**	0.828 (± 0.00)	0.858 (± 0.00)	0.857 (± 0.01)	0.857 (± 0.00)
0.50	0.784 (± 0.00)	0.827 (± 0.00)	0.856 (± 0.00)	**0.862 (± 0.00)**	0.778 (± 0.00)	0.852 (± 0.00)	0.851 (± 0.01)	0.852 (± 0.00)
0.60	0.777 (± 0.00)	0.828 (± 0.00)	0.851 (± 0.00)	**0.855 (± 0.00)**	0.805 (± 0.00)	0.849 (± 0.00)	0.846 (± 0.00)	0.845 (± 0.00)
0.70	0.772 (± 0.00)	0.824 (± 0.00)	**0.847 (± 0.00)**	0.845 (± 0.00)	0.726 (± 0.00)	0.844 (± 0.00)	0.834 (± 0.01)	0.835 (± 0.01)
0.80	0.756 (± 0.00)	0.819 (± 0.00)	0.836 (± 0.00)	0.832 (± 0.00)	0.443 (± 0.00)	**0.837 (± 0.00)**	0.820 (± 0.00)	0.816 (± 0.00)
0.90	0.700 (± 0.00)	0.806 (± 0.00)	0.822 (± 0.00)	0.803 (± 0.00)	0.315 (± 0.00)	**0.832 (± 0.00)**	0.791 (± 0.01)	0.786 (± 0.01)
0.99	0.452 (± 0.00)	0.262 (± 0.00)	0.793 (± 0.00)	0.327 (± 0.00)	0.315 (± 0.00)	**0.814 (± 0.00)**	0.674 (± 0.02)	0.693 (± 0.01)

Table C.16: F1 scores for PubMed under mechanism *S-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.784 (± 0.01)	0.831 (± 0.00)	**0.883 (± 0.00)**	-	0.877 (± 0.00)	0.882 (± 0.00)	0.875 (± 0.00)	0.875 (± 0.00)
0.10	0.786 (± 0.00)	0.831 (± 0.00)	0.875 (± 0.00)	**0.875 (± 0.00)**	0.870 (± 0.00)	0.871 (± 0.00)	0.868 (± 0.01)	0.866 (± 0.01)
0.20	0.783 (± 0.00)	0.827 (± 0.00)	0.869 (± 0.00)	**0.870 (± 0.00)**	0.861 (± 0.00)	0.867 (± 0.00)	0.860 (± 0.01)	0.859 (± 0.01)
0.30	0.785 (± 0.00)	0.832 (± 0.00)	0.863 (± 0.00)	**0.865 (± 0.00)**	0.861 (± 0.00)	0.863 (± 0.00)	0.853 (± 0.01)	0.852 (± 0.00)
0.40	0.785 (± 0.00)	0.828 (± 0.00)	0.856 (± 0.00)	**0.857 (± 0.00)**	0.848 (± 0.00)	0.856 (± 0.00)	0.846 (± 0.01)	0.847 (± 0.01)
0.50	0.775 (± 0.00)	0.827 (± 0.00)	0.853 (± 0.00)	**0.854 (± 0.00)**	0.808 (± 0.00)	0.848 (± 0.00)	0.838 (± 0.00)	0.837 (± 0.00)
0.60	0.774 (± 0.00)	0.822 (± 0.00)	0.843 (± 0.00)	**0.845 (± 0.00)**	0.798 (± 0.00)	0.843 (± 0.00)	0.829 (± 0.00)	0.827 (± 0.00)
0.70	0.760 (± 0.00)	0.813 (± 0.00)	0.832 (± 0.00)	0.827 (± 0.00)	0.762 (± 0.00)	**0.836 (± 0.00)**	0.815 (± 0.00)	0.814 (± 0.00)
0.80	0.744 (± 0.00)	0.806 (± 0.00)	0.828 (± 0.00)	0.808 (± 0.00)	0.683 (± 0.00)	**0.832 (± 0.00)**	0.785 (± 0.01)	0.788 (± 0.01)
0.90	0.706 (± 0.00)	0.786 (± 0.00)	0.815 (± 0.00)	0.743 (± 0.00)	0.421 (± 0.00)	**0.825 (± 0.00)**	0.727 (± 0.01)	0.729 (± 0.00)
0.99	0.441 (± 0.00)	0.259 (± 0.00)	0.765 (± 0.00)	0.333 (± 0.00)	0.310 (± 0.00)	**0.794 (± 0.00)**	0.446 (± 0.03)	0.458 (± 0.02)

Table C.17: F1 scores for PubMed under mechanism *CD-MCAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.784 (± 0.01)	0.831 (± 0.00)	**0.883 (± 0.00)**	0.881 (± 0.00)	0.877 (± 0.00)	0.882 (± 0.00)	0.875 (± 0.00)	0.876 (± 0.00)
0.10	0.738 (± 0.00)	0.824 (± 0.00)	0.855 (± 0.00)	**0.857 (± 0.00)**	0.830 (± 0.00)	0.852 (± 0.00)	0.848 (± 0.00)	0.846 (± 0.00)
0.20	0.700 (± 0.00)	0.820 (± 0.00)	0.845 (± 0.00)	**0.851 (± 0.00)**	0.828 (± 0.00)	0.844 (± 0.00)	0.837 (± 0.00)	0.836 (± 0.00)
0.30	0.607 (± 0.00)	0.823 (± 0.00)	0.843 (± 0.00)	**0.844 (± 0.00)**	0.823 (± 0.00)	0.836 (± 0.00)	0.822 (± 0.00)	0.822 (± 0.00)
0.40	0.534 (± 0.00)	0.821 (± 0.00)	0.834 (± 0.00)	**0.842 (± 0.00)**	0.818 (± 0.00)	0.830 (± 0.00)	0.821 (± 0.01)	0.821 (± 0.01)
0.50	0.509 (± 0.00)	0.814 (± 0.00)	0.818 (± 0.00)	**0.823 (± 0.00)**	0.797 (± 0.00)	0.820 (± 0.00)	0.808 (± 0.01)	0.806 (± 0.01)
0.60	0.422 (± 0.00)	0.812 (± 0.00)	0.808 (± 0.00)	**0.816 (± 0.00)**	0.787 (± 0.00)	0.812 (± 0.00)	0.790 (± 0.00)	0.793 (± 0.01)
0.70	0.415 (± 0.00)	0.802 (± 0.00)	0.797 (± 0.00)	**0.811 (± 0.00)**	0.779 (± 0.00)	0.801 (± 0.00)	0.778 (± 0.01)	0.774 (± 0.01)
0.80	0.396 (± 0.00)	0.779 (± 0.00)	0.749 (± 0.00)	**0.783 (± 0.00)**	0.713 (± 0.00)	0.754 (± 0.00)	0.738 (± 0.01)	0.749 (± 0.02)
0.90	0.306 (± 0.00)	0.574 (± 0.00)	0.693 (± 0.00)	**0.700 (± 0.00)**	0.391 (± 0.00)	0.683 (± 0.00)	0.664 (± 0.01)	0.667 (± 0.02)
0.99	0.198 (± 0.00)	0.266 (± 0.00)	0.303 (± 0.00)	0.330 (± 0.00)	0.306 (± 0.00)	0.305 (± 0.00)	**0.346 (± 0.02)**	0.345 (± 0.02)

Table C.18: F1 scores for PubMed under mechanism *FD-MNAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.784 (± 0.01)	0.831 (± 0.00)	**0.883 (± 0.00)**	0.881 (± 0.00)	0.877 (± 0.00)	0.882 (± 0.00)	0.875 (± 0.00)	0.874 (± 0.00)
0.10	0.785 (± 0.02)	0.832 (± 0.00)	0.876 (± 0.01)	**0.880 (± 0.01)**	0.834 (± 0.00)	0.874 (± 0.01)	0.867 (± 0.01)	0.868 (± 0.00)
0.20	0.785 (± 0.02)	0.834 (± 0.00)	0.869 (± 0.00)	**0.875 (± 0.00)**	0.832 (± 0.00)	0.869 (± 0.01)	0.864 (± 0.01)	0.864 (± 0.00)
0.30	0.785 (± 0.02)	0.830 (± 0.00)	0.865 (± 0.00)	**0.870 (± 0.00)**	0.829 (± 0.00)	0.860 (± 0.00)	0.858 (± 0.00)	0.858 (± 0.01)
0.40	0.780 (± 0.01)	0.827 (± 0.00)	0.860 (± 0.00)	**0.866 (± 0.00)**	0.733 (± 0.11)	0.856 (± 0.00)	0.853 (± 0.01)	0.854 (± 0.00)
0.50	0.775 (± 0.02)	0.822 (± 0.00)	0.853 (± 0.00)	**0.859 (± 0.00)**	0.720 (± 0.12)	0.850 (± 0.00)	0.844 (± 0.01)	0.846 (± 0.00)
0.60	0.763 (± 0.02)	0.824 (± 0.01)	0.847 (± 0.01)	**0.850 (± 0.00)**	0.746 (± 0.04)	0.842 (± 0.00)	0.836 (± 0.01)	0.836 (± 0.00)
0.70	0.745 (± 0.03)	0.813 (± 0.00)	0.836 (± 0.00)	0.834 (± 0.00)	0.579 (± 0.25)	**0.837 (± 0.00)**	0.827 (± 0.00)	0.826 (± 0.00)
0.80	0.745 (± 0.03)	0.819 (± 0.00)	0.759 (± 0.04)	**0.829 (± 0.00)**	0.555 (± 0.14)	0.764 (± 0.00)	0.805 (± 0.01)	0.805 (± 0.01)
0.90	0.336 (± 0.01)	0.806 (± 0.00)	0.693 (± 0.01)	**0.812 (± 0.00)**	0.529 (± 0.13)	0.653 (± 0.00)	0.780 (± 0.01)	0.777 (± 0.01)
0.99	0.278 (± 0.01)	0.282 (± 0.01)	0.303 (± 0.05)	0.347 (± 0.00)	0.399 (± 0.33)	0.335 (± 0.01)	0.659 (± 0.02)	**0.669 (± 0.02)**

Table C.19: F1 scores for PubMed under mechanism *CD-MNAR* and varying μ (GSPNis not reported as it is not designed for categorical features).

μ	GOODIE	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian
0.00	0.784 (± 0.01)	0.831 (± 0.00)	**0.883 (± 0.00)**	0.881 (± 0.00)	0.877 (± 0.00)	0.882 (± 0.00)	0.874 (± 0.00)	0.875 (± 0.00)
0.10	0.789 (± 0.02)	0.829 (± 0.00)	0.878 (± 0.00)	**0.880 (± 0.00)**	0.835 (± 0.00)	0.877 (± 0.00)	0.866 (± 0.01)	0.869 (± 0.00)
0.20	0.783 (± 0.01)	0.830 (± 0.00)	0.870 (± 0.00)	**0.876 (± 0.00)**	0.834 (± 0.00)	0.867 (± 0.01)	0.862 (± 0.00)	0.861 (± 0.00)
0.30	0.783 (± 0.02)	0.828 (± 0.00)	0.863 (± 0.00)	**0.871 (± 0.00)**	0.823 (± 0.00)	0.866 (± 0.00)	0.860 (± 0.00)	0.859 (± 0.00)
0.40	0.777 (± 0.02)	0.826 (± 0.00)	0.858 (± 0.00)	**0.863 (± 0.00)**	0.830 (± 0.00)	0.857 (± 0.01)	0.854 (± 0.00)	0.852 (± 0.00)
0.50	0.779 (± 0.01)	0.825 (± 0.00)	0.853 (± 0.00)	**0.858 (± 0.00)**	0.826 (± 0.00)	0.853 (± 0.00)	0.847 (± 0.00)	0.849 (± 0.00)
0.60	0.769 (± 0.02)	0.824 (± 0.00)	0.847 (± 0.01)	**0.848 (± 0.01)**	0.784 (± 0.04)	0.848 (± 0.00)	0.840 (± 0.01)	0.840 (± 0.00)
0.70	0.752 (± 0.03)	0.816 (± 0.00)	0.837 (± 0.00)	0.835 (± 0.00)	0.765 (± 0.02)	**0.837 (± 0.00)**	0.827 (± 0.00)	0.825 (± 0.00)
0.80	0.742 (± 0.03)	0.813 (± 0.00)	0.828 (± 0.00)	0.817 (± 0.00)	0.323 (± 0.10)	**0.836 (± 0.00)**	0.810 (± 0.01)	0.809 (± 0.00)
0.90	0.605 (± 0.13)	0.628 (± 0.24)	0.812 (± 0.00)	0.770 (± 0.00)	0.280 (± 0.05)	**0.823 (± 0.00)**	0.760 (± 0.01)	0.763 (± 0.01)
0.99	0.557 (± 0.14)	0.260 (± 0.00)	0.800 (± 0.00)	0.689 (± 0.01)	0.418 (± 0.04)	**0.818 (± 0.00)**	0.717 (± 0.01)	0.728 (± 0.02)

Table C.20: F1 scores for SYNTHETIC under mechanism *U-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.812 (± 0.00)	0.865 (± 0.00)	0.815 (± 0.00)	0.980 (± 0.00)	0.982 (± 0.00)	0.978 (± 0.00)	0.977 (± 0.00)	0.978 (± 0.01)	0.978 (± 0.01)	**0.983 (± 0.01)**
0.10	0.810 (± 0.00)	0.822 (± 0.00)	0.825 (± 0.00)	**0.910 (± 0.00)**	0.902 (± 0.00)	0.875 (± 0.00)	0.898 (± 0.00)	0.902 (± 0.02)	0.903 (± 0.02)	0.901 (± 0.00)
0.20	0.792 (± 0.00)	0.759 (± 0.00)	0.808 (± 0.00)	0.863 (± 0.00)	**0.870 (± 0.00)**	0.790 (± 0.00)	0.855 (± 0.00)	0.853 (± 0.02)	0.853 (± 0.02)	0.861 (± 0.00)
0.30	0.758 (± 0.00)	0.768 (± 0.00)	0.762 (± 0.00)	0.795 (± 0.00)	0.808 (± 0.00)	0.770 (± 0.00)	0.805 (± 0.00)	0.800 (± 0.03)	0.801 (± 0.03)	**0.815 (± 0.00)**
0.40	0.758 (± 0.00)	0.749 (± 0.00)	0.759 (± 0.00)	0.764 (± 0.00)	0.771 (± 0.00)	0.745 (± 0.00)	0.763 (± 0.00)	0.766 (± 0.02)	0.766 (± 0.02)	**0.791 (± 0.00)**
0.50	0.747 (± 0.00)	0.721 (± 0.00)	0.642 (± 0.00)	0.745 (± 0.00)	0.745 (± 0.00)	0.710 (± 0.00)	**0.748 (± 0.00)**	0.732 (± 0.04)	0.730 (± 0.04)	0.739 (± 0.00)
0.60	**0.773 (± 0.00)**	0.708 (± 0.00)	0.680 (± 0.00)	0.720 (± 0.00)	0.737 (± 0.00)	0.692 (± 0.00)	0.717 (± 0.00)	0.714 (± 0.04)	0.710 (± 0.04)	0.714 (± 0.00)
0.70	**0.742 (± 0.00)**	0.629 (± 0.00)	0.611 (± 0.00)	0.683 (± 0.00)	0.689 (± 0.00)	0.673 (± 0.00)	0.678 (± 0.00)	0.687 (± 0.03)	0.693 (± 0.03)	0.693 (± 0.00)
0.80	**0.771 (± 0.00)**	0.579 (± 0.00)	0.621 (± 0.00)	0.632 (± 0.00)	0.638 (± 0.00)	0.601 (± 0.00)	0.638 (± 0.00)	0.610 (± 0.05)	0.621 (± 0.05)	0.649 (± 0.00)
0.90	**0.776 (± 0.00)**	0.544 (± 0.00)	0.567 (± 0.00)	0.605 (± 0.00)	0.602 (± 0.00)	0.592 (± 0.00)	0.588 (± 0.00)	0.589 (± 0.04)	0.599 (± 0.04)	0.590 (± 0.00)
0.99	**0.762 (± 0.00)**	0.499 (± 0.00)	0.391 (± 0.00)	0.542 (± 0.00)	0.367 (± 0.00)	0.471 (± 0.00)	0.547 (± 0.00)	0.548 (± 0.04)	0.411 (± 0.07)	0.535 (± 0.00)

Table C.21: F1 scores for SYNTHETIC under mechanism *S-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.812 (± 0.00)	0.865 (± 0.00)	0.815 (± 0.00)	0.980 (± 0.00)	0.982 (± 0.00)	0.978 (± 0.00)	0.977 (± 0.00)	0.978 (± 0.01)	0.978 (± 0.01)	**0.983 (± 0.01)**
0.10	0.756 (± 0.00)	0.748 (± 0.00)	0.723 (± 0.00)	0.903 (± 0.00)	**0.912 (± 0.00)**	0.903 (± 0.00)	0.900 (± 0.00)	0.909 (± 0.01)	0.911 (± 0.01)	0.898 (± 0.00)
0.20	0.769 (± 0.00)	0.733 (± 0.00)	0.727 (± 0.00)	**0.883 (± 0.00)**	0.883 (± 0.00)	0.872 (± 0.00)	0.870 (± 0.00)	0.844 (± 0.02)	0.843 (± 0.02)	0.875 (± 0.00)
0.30	0.742 (± 0.00)	0.737 (± 0.00)	0.700 (± 0.00)	0.830 (± 0.00)	**0.842 (± 0.00)**	0.841 (± 0.00)	0.831 (± 0.00)	0.817 (± 0.02)	0.813 (± 0.01)	0.833 (± 0.00)
0.40	0.716 (± 0.00)	0.712 (± 0.00)	0.683 (± 0.00)	**0.810 (± 0.00)**	0.798 (± 0.00)	0.752 (± 0.00)	0.793 (± 0.00)	0.775 (± 0.02)	0.777 (± 0.02)	0.799 (± 0.00)
0.50	0.700 (± 0.00)	0.711 (± 0.00)	0.704 (± 0.00)	0.785 (± 0.00)	**0.788 (± 0.00)**	0.705 (± 0.00)	0.780 (± 0.00)	0.746 (± 0.02)	0.748 (± 0.02)	0.779 (± 0.00)
0.60	0.658 (± 0.00)	0.674 (± 0.00)	0.695 (± 0.00)	0.747 (± 0.00)	**0.761 (± 0.00)**	0.726 (± 0.00)	0.738 (± 0.00)	0.718 (± 0.03)	0.705 (± 0.04)	0.756 (± 0.00)
0.70	0.618 (± 0.00)	0.675 (± 0.00)	0.652 (± 0.00)	0.687 (± 0.00)	0.703 (± 0.00)	0.665 (± 0.00)	0.700 (± 0.00)	0.663 (± 0.03)	0.667 (± 0.02)	**0.727 (± 0.00)**
0.80	0.584 (± 0.00)	0.649 (± 0.00)	0.616 (± 0.00)	0.653 (± 0.00)	0.667 (± 0.00)	0.645 (± 0.00)	0.638 (± 0.00)	0.647 (± 0.05)	0.656 (± 0.04)	**0.676 (± 0.00)**
0.90	0.527 (± 0.00)	0.588 (± 0.00)	0.589 (± 0.00)	0.597 (± 0.00)	0.597 (± 0.00)	0.578 (± 0.00)	0.591 (± 0.00)	**0.601 (± 0.02)**	0.593 (± 0.02)	0.582 (± 0.00)
0.99	0.337 (± 0.00)	0.455 (± 0.00)	0.338 (± 0.00)	**0.515 (± 0.00)**	0.425 (± 0.00)	0.403 (± 0.00)	0.513 (± 0.00)	0.488 (± 0.02)	0.444 (± 0.05)	0.477 (± 0.00)

Table C.22: F1 scores for SYNTHETIC under mechanism *CD-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.812 (± 0.00)	0.865 (± 0.00)	0.815 (± 0.00)	0.980 (± 0.00)	**0.982 (± 0.00)**	0.978 (± 0.00)	0.977 (± 0.00)	0.978 (± 0.01)	0.978 (± 0.01)	0.886 (± 0.00)
0.10	0.778 (± 0.00)	0.785 (± 0.00)	0.792 (± 0.00)	0.860 (± 0.00)	0.857 (± 0.00)	0.845 (± 0.00)	0.860 (± 0.00)	**0.978 (± 0.01)**	0.978 (± 0.01)	0.829 (± 0.00)
0.20	0.760 (± 0.00)	0.731 (± 0.00)	0.705 (± 0.00)	**0.788 (± 0.00)**	0.770 (± 0.00)	0.741 (± 0.00)	0.772 (± 0.00)	0.699 (± 0.02)	0.699 (± 0.02)	0.780 (± 0.00)
0.30	0.730 (± 0.00)	0.666 (± 0.00)	0.718 (± 0.00)	0.736 (± 0.00)	0.733 (± 0.00)	0.730 (± 0.00)	0.734 (± 0.00)	0.605 (± 0.03)	0.605 (± 0.03)	**0.738 (± 0.00)**
0.40	**0.736 (± 0.00)**	0.625 (± 0.00)	0.607 (± 0.00)	0.661 (± 0.00)	0.659 (± 0.00)	0.673 (± 0.00)	0.649 (± 0.00)	0.605 (± 0.03)	0.605 (± 0.03)	0.703 (± 0.00)
0.50	**0.761 (± 0.00)**	0.547 (± 0.00)	0.542 (± 0.00)	0.619 (± 0.00)	0.618 (± 0.00)	0.628 (± 0.00)	0.613 (± 0.00)	0.605 (± 0.03)	0.605 (± 0.03)	0.682 (± 0.00)
0.60	**0.768 (± 0.00)**	0.594 (± 0.00)	0.543 (± 0.00)	0.621 (± 0.00)	0.613 (± 0.00)	0.619 (± 0.00)	0.605 (± 0.00)	0.528 (± 0.03)	0.528 (± 0.03)	0.667 (± 0.00)
0.70	**0.759 (± 0.00)**	0.603 (± 0.00)	0.586 (± 0.00)	0.617 (± 0.00)	0.607 (± 0.00)	0.591 (± 0.00)	0.594 (± 0.00)	0.536 (± 0.03)	0.536 (± 0.03)	0.675 (± 0.00)
0.80	**0.758 (± 0.00)**	0.613 (± 0.00)	0.486 (± 0.00)	0.617 (± 0.00)	0.622 (± 0.00)	0.631 (± 0.00)	0.620 (± 0.00)	0.536 (± 0.03)	0.536 (± 0.03)	0.666 (± 0.00)
0.90	**0.775 (± 0.00)**	0.544 (± 0.00)	0.529 (± 0.00)	0.623 (± 0.00)	0.633 (± 0.00)	0.623 (± 0.00)	0.606 (± 0.00)	0.535 (± 0.02)	0.536 (± 0.03)	0.678 (± 0.00)
0.99	**0.764 (± 0.00)**	0.569 (± 0.00)	0.557 (± 0.00)	0.609 (± 0.00)	0.611 (± 0.00)	0.643 (± 0.00)	0.612 (± 0.00)	0.646 (± 0.03)	0.638 (± 0.03)	0.667 (± 0.00)

Table C.23: F1 scores for SYNTHETIC under mechanism *FD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.812 (± 0.00)	0.865 (± 0.00)	0.815 (± 0.00)	0.980 (± 0.00)	0.982 (± 0.00)	0.978 (± 0.00)	0.977 (± 0.00)	0.976 (± 0.01)	0.976 (± 0.01)	**0.983 (± 0.01)**
0.10	0.751 (± 0.05)	0.750 (± 0.03)	0.761 (± 0.02)	0.893 (± 0.01)	**0.900 (± 0.02)**	0.878 (± 0.02)	0.895 (± 0.01)	0.891 (± 0.02)	0.894 (± 0.02)	0.895 (± 0.01)
0.20	0.750 (± 0.03)	0.721 (± 0.01)	0.699 (± 0.04)	0.836 (± 0.02)	0.845 (± 0.02)	0.785 (± 0.04)	0.847 (± 0.02)	0.849 (± 0.03)	**0.854 (± 0.02)**	0.843 (± 0.04)
0.30	0.691 (± 0.04)	0.678 (± 0.02)	0.667 (± 0.03)	0.810 (± 0.01)	0.812 (± 0.01)	0.771 (± 0.03)	0.789 (± 0.01)	0.819 (± 0.02)	**0.821 (± 0.01)**	0.812 (± 0.01)
0.40	0.693 (± 0.03)	0.678 (± 0.03)	0.682 (± 0.03)	0.791 (± 0.02)	0.798 (± 0.00)	0.763 (± 0.02)	0.791 (± 0.00)	0.785 (± 0.02)	0.793 (± 0.02)	**0.806 (± 0.01)**
0.50	0.673 (± 0.04)	0.668 (± 0.01)	0.676 (± 0.03)	0.753 (± 0.01)	0.758 (± 0.02)	0.713 (± 0.03)	0.752 (± 0.01)	0.741 (± 0.02)	0.737 (± 0.02)	**0.763 (± 0.01)**
0.60	0.620 (± 0.02)	0.608 (± 0.02)	0.610 (± 0.02)	0.708 (± 0.01)	0.715 (± 0.00)	0.685 (± 0.02)	0.702 (± 0.02)	0.714 (± 0.01)	0.719 (± 0.01)	**0.727 (± 0.01)**
0.70	0.494 (± 0.07)	0.580 (± 0.06)	0.588 (± 0.02)	0.651 (± 0.03)	0.670 (± 0.04)	0.631 (± 0.03)	0.653 (± 0.04)	0.676 (± 0.02)	0.673 (± 0.03)	**0.688 (± 0.02)**
0.80	0.425 (± 0.07)	0.607 (± 0.04)	0.577 (± 0.01)	0.611 (± 0.01)	0.627 (± 0.02)	0.589 (± 0.03)	0.596 (± 0.01)	0.619 (± 0.01)	0.624 (± 0.01)	**0.639 (± 0.02)**
0.90	0.362 (± 0.02)	**0.625 (± 0.02)**	0.512 (± 0.05)	0.575 (± 0.02)	0.595 (± 0.02)	0.573 (± 0.02)	0.582 (± 0.01)	0.594 (± 0.04)	0.601 (± 0.02)	0.612 (± 0.00)
0.99	0.429 (± 0.13)	0.570 (± 0.02)	0.423 (± 0.11)	0.547 (± 0.02)	0.536 (± 0.01)	0.490 (± 0.05)	0.551 (± 0.01)	0.569 (± 0.03)	0.545 (± 0.04)	**0.576 (± 0.02)**

Table C.24: F1 scores for SYNTHETIC under mechanism *CD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.812 (± 0.00)	0.865 (± 0.00)	0.815 (± 0.00)	0.980 (± 0.00)	0.982 (± 0.00)	0.978 (± 0.00)	0.977 (± 0.00)	0.978 (± 0.01)	0.978 (± 0.01)	**0.983 (± 0.01)**
0.10	0.756 (± 0.04)	0.757 (± 0.02)	0.752 (± 0.02)	0.913 (± 0.02)	**0.918 (± 0.02)**	0.882 (± 0.02)	0.912 (± 0.01)	0.912 (± 0.02)	0.912 (± 0.02)	0.913 (± 0.02)
0.20	0.730 (± 0.05)	0.718 (± 0.02)	0.674 (± 0.05)	0.856 (± 0.03)	**0.868 (± 0.03)**	0.800 (± 0.04)	0.861 (± 0.04)	0.864 (± 0.02)	0.865 (± 0.02)	0.865 (± 0.03)
0.30	0.663 (± 0.05)	0.716 (± 0.02)	0.689 (± 0.03)	0.803 (± 0.02)	0.820 (± 0.02)	0.768 (± 0.03)	0.810 (± 0.03)	0.807 (± 0.02)	0.804 (± 0.02)	**0.830 (± 0.03)**
0.40	0.530 (± 0.16)	0.678 (± 0.01)	0.718 (± 0.03)	0.744 (± 0.01)	0.749 (± 0.00)	0.753 (± 0.01)	0.739 (± 0.03)	0.756 (± 0.01)	0.742 (± 0.01)	**0.776 (± 0.01)**
0.50	0.487 (± 0.12)	0.662 (± 0.03)	0.655 (± 0.04)	0.697 (± 0.03)	0.695 (± 0.03)	0.683 (± 0.04)	0.699 (± 0.04)	0.689 (± 0.03)	0.657 (± 0.02)	**0.725 (± 0.01)**
0.60	0.575 (± 0.06)	0.696 (± 0.03)	0.577 (± 0.02)	0.683 (± 0.03)	0.658 (± 0.03)	0.666 (± 0.02)	0.645 (± 0.03)	0.694 (± 0.04)	0.638 (± 0.03)	**0.731 (± 0.03)**
0.70	0.553 (± 0.03)	0.616 (± 0.03)	0.583 (± 0.02)	0.613 (± 0.02)	0.600 (± 0.04)	0.617 (± 0.04)	0.592 (± 0.05)	0.642 (± 0.03)	0.603 (± 0.04)	**0.668 (± 0.01)**
0.80	0.486 (± 0.06)	0.638 (± 0.03)	0.592 (± 0.03)	0.588 (± 0.02)	0.596 (± 0.03)	0.570 (± 0.02)	0.563 (± 0.03)	0.618 (± 0.02)	0.580 (± 0.04)	**0.655 (± 0.02)**
0.90	0.432 (± 0.08)	0.618 (± 0.05)	0.479 (± 0.10)	0.586 (± 0.04)	0.607 (± 0.03)	0.556 (± 0.03)	0.553 (± 0.01)	0.598 (± 0.03)	0.557 (± 0.04)	**0.635 (± 0.04)**
0.99	0.468 (± 0.03)	0.545 (± 0.06)	0.396 (± 0.08)	**0.594 (± 0.01)**	0.537 (± 0.01)	0.475 (± 0.06)	0.549 (± 0.03)	0.550 (± 0.03)	0.485 (± 0.06)	0.568 (± 0.01)

Table C.25: F1 scores for AIR under mechanism *U-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.724 (± 0.00)	0.798 (± 0.02)	0.733 (± 0.00)	0.918 (± 0.00)	0.922 (± 0.01)	0.922 (± 0.00)	0.891 (± 0.00)	0.916 (± 0.02)	0.916 (± 0.02)	**0.930 (± 0.00)**
0.10	0.665 (± 0.00)	0.710 (± 0.00)	0.733 (± 0.00)	0.895 (± 0.00)	0.891 (± 0.00)	0.768 (± 0.00)	0.883 (± 0.00)	**0.904 (± 0.03)**	0.902 (± 0.03)	0.899 (± 0.00)
0.20	0.669 (± 0.00)	0.582 (± 0.00)	0.709 (± 0.00)	0.848 (± 0.00)	0.833 (± 0.00)	0.747 (± 0.00)	0.852 (± 0.00)	**0.874 (± 0.03)**	0.865 (± 0.03)	0.859 (± 0.00)
0.30	0.669 (± 0.00)	0.502 (± 0.00)	0.715 (± 0.00)	0.836 (± 0.00)	0.837 (± 0.00)	0.712 (± 0.00)	0.836 (± 0.00)	0.837 (± 0.04)	**0.857 (± 0.03)**	0.852 (± 0.00)
0.40	0.714 (± 0.00)	0.532 (± 0.00)	0.700 (± 0.00)	0.805 (± 0.00)	0.829 (± 0.00)	0.712 (± 0.00)	0.797 (± 0.00)	0.813 (± 0.02)	**0.839 (± 0.02)**	0.833 (± 0.00)
0.50	0.666 (± 0.00)	0.553 (± 0.00)	0.669 (± 0.00)	0.801 (± 0.00)	0.805 (± 0.00)	0.711 (± 0.00)	0.802 (± 0.00)	**0.832 (± 0.04)**	0.815 (± 0.03)	0.767 (± 0.00)
0.60	0.663 (± 0.00)	0.452 (± 0.00)	0.691 (± 0.00)	0.775 (± 0.00)	0.762 (± 0.00)	0.701 (± 0.00)	0.767 (± 0.00)	0.795 (± 0.04)	**0.807 (± 0.06)**	0.744 (± 0.00)
0.70	0.714 (± 0.00)	0.495 (± 0.00)	0.686 (± 0.00)	0.724 (± 0.00)	0.736 (± 0.00)	0.656 (± 0.00)	**0.754 (± 0.00)**	0.753 (± 0.07)	0.746 (± 0.05)	0.736 (± 0.00)
0.80	0.666 (± 0.00)	0.559 (± 0.00)	0.667 (± 0.00)	0.712 (± 0.00)	0.677 (± 0.00)	0.647 (± 0.00)	0.637 (± 0.00)	0.709 (± 0.03)	**0.715 (± 0.03)**	0.713 (± 0.00)
0.90	0.700 (± 0.00)	0.541 (± 0.00)	0.670 (± 0.00)	0.585 (± 0.00)	0.593 (± 0.00)	0.669 (± 0.00)	0.619 (± 0.00)	0.598 (± 0.06)	0.628 (± 0.04)	**0.705 (± 0.00)**
0.99	**0.693 (± 0.00)**	0.409 (± 0.00)	0.658 (± 0.00)	0.436 (± 0.00)	0.384 (± 0.00)	0.651 (± 0.00)	0.431 (± 0.00)	0.440 (± 0.05)	0.397 (± 0.04)	0.664 (± 0.00)

Table C.26: F1 scores for AIR under mechanism *S-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.724 (± 0.00)	0.798 (± 0.02)	0.733 (± 0.00)	0.918 (± 0.00)	0.922 (± 0.01)	0.922 (± 0.00)	0.891 (± 0.00)	0.916 (± 0.02)	0.916 (± 0.02)	**0.930 (± 0.00)**
0.10	0.568 (± 0.00)	0.644 (± 0.00)	0.733 (± 0.00)	0.891 (± 0.00)	0.899 (± 0.00)	0.895 (± 0.00)	0.872 (± 0.00)	0.879 (± 0.02)	**0.900 (± 0.02)**	0.891 (± 0.00)
0.20	0.573 (± 0.00)	0.597 (± 0.00)	0.733 (± 0.00)	0.860 (± 0.00)	0.883 (± 0.00)	0.851 (± 0.00)	**0.899 (± 0.00)**	0.860 (± 0.03)	0.865 (± 0.03)	0.890 (± 0.00)
0.30	0.630 (± 0.00)	0.527 (± 0.00)	0.665 (± 0.00)	0.850 (± 0.00)	0.847 (± 0.00)	0.820 (± 0.00)	0.852 (± 0.00)	0.838 (± 0.04)	**0.853 (± 0.03)**	0.835 (± 0.00)
0.40	0.571 (± 0.00)	0.508 (± 0.00)	0.728 (± 0.00)	0.819 (± 0.00)	0.819 (± 0.00)	0.795 (± 0.00)	0.826 (± 0.00)	0.812 (± 0.03)	0.796 (± 0.04)	**0.842 (± 0.00)**
0.50	0.562 (± 0.00)	0.530 (± 0.00)	0.742 (± 0.00)	0.787 (± 0.00)	0.770 (± 0.00)	**0.829 (± 0.00)**	0.799 (± 0.00)	0.769 (± 0.03)	0.778 (± 0.03)	0.817 (± 0.00)
0.60	0.549 (± 0.00)	0.532 (± 0.00)	0.739 (± 0.00)	0.750 (± 0.00)	0.737 (± 0.00)	**0.809 (± 0.00)**	0.761 (± 0.00)	0.736 (± 0.06)	0.718 (± 0.04)	0.797 (± 0.00)
0.70	0.603 (± 0.00)	0.532 (± 0.00)	0.706 (± 0.00)	0.686 (± 0.00)	0.661 (± 0.00)	**0.767 (± 0.00)**	0.666 (± 0.00)	0.709 (± 0.05)	0.693 (± 0.03)	0.756 (± 0.00)
0.80	0.610 (± 0.00)	0.476 (± 0.00)	0.657 (± 0.00)	0.607 (± 0.00)	0.605 (± 0.00)	0.721 (± 0.00)	0.601 (± 0.00)	0.614 (± 0.04)	0.603 (± 0.04)	**0.734 (± 0.00)**
0.90	0.504 (± 0.00)	0.389 (± 0.00)	0.692 (± 0.00)	0.549 (± 0.00)	0.505 (± 0.00)	0.677 (± 0.00)	0.522 (± 0.00)	0.537 (± 0.03)	0.511 (± 0.02)	**0.699 (± 0.00)**
0.99	0.435 (± 0.00)	0.332 (± 0.00)	**0.652 (± 0.00)**	0.350 (± 0.00)	0.333 (± 0.00)	0.643 (± 0.00)	0.353 (± 0.00)	0.351 (± 0.01)	0.354 (± 0.01)	0.652 (± 0.00)

Table C.27: F1 scores for AIR under mechanism *CD-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.724 (± 0.00)	0.798 (± 0.02)	0.733 (± 0.00)	0.918 (± 0.00)	0.922 (± 0.01)	0.922 (± 0.00)	0.891 (± 0.00)	0.916 (± 0.02)	0.916 (± 0.02)	**0.930 (± 0.00)**
0.10	0.714 (± 0.00)	0.730 (± 0.00)	0.706 (± 0.00)	0.804 (± 0.00)	0.819 (± 0.00)	0.700 (± 0.00)	0.820 (± 0.00)	0.825 (± 0.05)	0.825 (± 0.05)	**0.876 (± 0.00)**
0.20	0.714 (± 0.00)	0.730 (± 0.00)	0.703 (± 0.00)	0.804 (± 0.00)	0.819 (± 0.00)	0.677 (± 0.00)	0.820 (± 0.00)	0.825 (± 0.05)	0.825 (± 0.05)	**0.887 (± 0.00)**
0.30	0.710 (± 0.00)	0.651 (± 0.00)	0.613 (± 0.00)	0.721 (± 0.00)	0.697 (± 0.00)	0.696 (± 0.00)	0.726 (± 0.00)	0.725 (± 0.07)	0.725 (± 0.07)	**0.744 (± 0.00)**
0.40	0.701 (± 0.00)	0.587 (± 0.00)	0.617 (± 0.00)	0.717 (± 0.00)	0.687 (± 0.00)	0.691 (± 0.00)	0.701 (± 0.00)	0.719 (± 0.05)	0.719 (± 0.05)	**0.794 (± 0.00)**
0.50	0.717 (± 0.00)	0.504 (± 0.00)	0.458 (± 0.00)	0.528 (± 0.00)	0.571 (± 0.00)	0.625 (± 0.00)	0.564 (± 0.00)	0.556 (± 0.08)	0.556 (± 0.08)	**0.722 (± 0.00)**
0.60	0.717 (± 0.00)	0.504 (± 0.00)	0.450 (± 0.00)	0.528 (± 0.00)	0.571 (± 0.00)	0.625 (± 0.00)	0.564 (± 0.00)	0.556 (± 0.08)	0.556 (± 0.08)	**0.737 (± 0.00)**
0.70	**0.717 (± 0.00)**	0.498 (± 0.00)	0.446 (± 0.00)	0.540 (± 0.00)	0.553 (± 0.00)	0.668 (± 0.00)	0.518 (± 0.00)	0.498 (± 0.04)	0.498 (± 0.04)	0.662 (± 0.00)
0.80	**0.703 (± 0.00)**	0.557 (± 0.00)	0.430 (± 0.00)	0.515 (± 0.00)	0.481 (± 0.00)	0.676 (± 0.00)	0.457 (± 0.00)	0.495 (± 0.05)	0.495 (± 0.05)	0.680 (± 0.00)
0.90	**0.703 (± 0.00)**	0.498 (± 0.00)	0.338 (± 0.00)	0.515 (± 0.00)	0.481 (± 0.00)	0.676 (± 0.00)	0.457 (± 0.00)	0.495 (± 0.05)	0.495 (± 0.05)	0.674 (± 0.00)
0.99	0.660 (± 0.00)	0.468 (± 0.00)	0.338 (± 0.00)	0.515 (± 0.00)	0.481 (± 0.00)	0.682 (± 0.00)	0.457 (± 0.00)	0.675 (± 0.05)	**0.688 (± 0.05)**	0.673 (± 0.00)

Table C.28: F1 scores for AIR under mechanism *FD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.724 (± 0.00)	0.798 (± 0.02)	0.733 (± 0.00)	0.918 (± 0.00)	0.922 (± 0.01)	0.922 (± 0.00)	0.891 (± 0.00)	0.911 (± 0.03)	0.914 (± 0.02)	**0.930 (± 0.00)**
0.10	0.618 (± 0.10)	0.758 (± 0.05)	0.709 (± 0.03)	0.895 (± 0.01)	0.891 (± 0.04)	0.772 (± 0.02)	0.883 (± 0.03)	0.890 (± 0.03)	0.897 (± 0.03)	**0.906 (± 0.02)**
0.20	0.595 (± 0.10)	0.776 (± 0.05)	0.668 (± 0.08)	0.883 (± 0.03)	0.879 (± 0.01)	0.756 (± 0.03)	0.867 (± 0.02)	0.852 (± 0.02)	**0.888 (± 0.02)**	0.887 (± 0.01)
0.30	0.580 (± 0.12)	0.536 (± 0.15)	0.721 (± 0.01)	0.852 (± 0.03)	0.859 (± 0.01)	0.745 (± 0.03)	0.833 (± 0.02)	0.845 (± 0.02)	0.864 (± 0.03)	**0.875 (± 0.01)**
0.40	0.677 (± 0.03)	0.575 (± 0.09)	0.716 (± 0.02)	0.852 (± 0.02)	**0.855 (± 0.03)**	0.725 (± 0.02)	0.840 (± 0.04)	0.839 (± 0.02)	0.848 (± 0.04)	0.852 (± 0.02)
0.50	0.587 (± 0.13)	0.620 (± 0.08)	0.719 (± 0.02)	0.837 (± 0.01)	0.832 (± 0.03)	0.698 (± 0.04)	0.829 (± 0.04)	0.806 (± 0.01)	0.822 (± 0.05)	**0.852 (± 0.03)**
0.60	0.556 (± 0.16)	0.686 (± 0.05)	0.692 (± 0.02)	**0.837 (± 0.02)**	0.808 (± 0.06)	0.711 (± 0.03)	0.793 (± 0.02)	0.780 (± 0.04)	0.783 (± 0.05)	0.817 (± 0.03)
0.70	0.556 (± 0.16)	0.634 (± 0.02)	0.717 (± 0.02)	0.769 (± 0.03)	**0.779 (± 0.05)**	0.685 (± 0.01)	0.750 (± 0.04)	0.745 (± 0.05)	0.771 (± 0.05)	0.770 (± 0.03)
0.80	0.556 (± 0.16)	0.665 (± 0.02)	0.665 (± 0.03)	0.654 (± 0.05)	0.709 (± 0.03)	0.667 (± 0.03)	0.660 (± 0.08)	0.718 (± 0.07)	0.719 (± 0.04)	**0.786 (± 0.02)**
0.90	0.582 (± 0.09)	0.645 (± 0.04)	0.662 (± 0.01)	0.658 (± 0.05)	0.661 (± 0.02)	0.659 (± 0.03)	0.530 (± 0.05)	0.670 (± 0.06)	0.655 (± 0.05)	**0.710 (± 0.05)**
0.99	0.638 (± 0.05)	0.635 (± 0.02)	0.637 (± 0.04)	0.557 (± 0.04)	0.528 (± 0.03)	**0.674 (± 0.02)**	0.508 (± 0.07)	0.549 (± 0.06)	0.565 (± 0.04)	0.616 (± 0.05)

Table C.29: F1 scores for AIR under mechanism *CD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.724 (± 0.00)	0.798 (± 0.02)	0.733 (± 0.00)	0.918 (± 0.00)	0.922 (± 0.01)	0.922 (± 0.00)	0.891 (± 0.00)	0.916 (± 0.02)	0.916 (± 0.02)	**0.930 (± 0.00)**
0.10	0.598 (± 0.11)	0.667 (± 0.02)	0.722 (± 0.02)	0.888 (± 0.05)	0.887 (± 0.04)	0.851 (± 0.01)	0.891 (± 0.03)	0.860 (± 0.04)	0.883 (± 0.04)	**0.895 (± 0.04)**
0.20	0.556 (± 0.16)	0.632 (± 0.19)	0.697 (± 0.02)	**0.864 (± 0.02)**	0.848 (± 0.06)	0.778 (± 0.04)	0.841 (± 0.01)	0.853 (± 0.04)	0.836 (± 0.04)	0.864 (± 0.01)
0.30	0.556 (± 0.16)	0.526 (± 0.13)	0.722 (± 0.03)	0.845 (± 0.01)	0.825 (± 0.04)	0.689 (± 0.02)	0.841 (± 0.03)	0.855 (± 0.02)	0.806 (± 0.04)	**0.891 (± 0.04)**
0.40	0.480 (± 0.16)	0.691 (± 0.14)	0.601 (± 0.12)	0.833 (± 0.02)	0.805 (± 0.03)	0.722 (± 0.02)	0.860 (± 0.03)	0.856 (± 0.02)	0.811 (± 0.02)	**0.860 (± 0.03)**
0.50	0.536 (± 0.16)	0.607 (± 0.09)	0.705 (± 0.02)	0.813 (± 0.02)	0.769 (± 0.04)	0.674 (± 0.01)	0.783 (± 0.04)	0.790 (± 0.05)	0.777 (± 0.05)	**0.833 (± 0.03)**
0.60	0.622 (± 0.06)	0.636 (± 0.04)	0.694 (± 0.01)	0.758 (± 0.05)	0.708 (± 0.07)	0.681 (± 0.01)	0.766 (± 0.06)	**0.814 (± 0.03)**	0.774 (± 0.07)	0.766 (± 0.06)
0.70	0.580 (± 0.10)	0.672 (± 0.07)	0.681 (± 0.01)	**0.757 (± 0.03)**	0.724 (± 0.04)	0.644 (± 0.02)	0.753 (± 0.05)	0.755 (± 0.06)	0.720 (± 0.02)	0.726 (± 0.05)
0.80	0.563 (± 0.12)	0.681 (± 0.05)	0.676 (± 0.01)	0.733 (± 0.02)	0.655 (± 0.02)	0.658 (± 0.02)	0.712 (± 0.01)	0.735 (± 0.05)	0.686 (± 0.06)	**0.769 (± 0.03)**
0.90	0.655 (± 0.03)	0.615 (± 0.04)	0.653 (± 0.01)	**0.693 (± 0.04)**	0.579 (± 0.04)	0.643 (± 0.04)	0.692 (± 0.06)	0.678 (± 0.03)	0.613 (± 0.04)	0.668 (± 0.02)
0.99	0.654 (± 0.03)	0.522 (± 0.04)	**0.660 (± 0.05)**	0.524 (± 0.07)	0.473 (± 0.05)	0.650 (± 0.06)	0.424 (± 0.06)	0.523 (± 0.06)	0.411 (± 0.03)	0.631 (± 0.07)

Table C.30: F1 scores for ELECTRIC under mechanism *U-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.588 (± 0.00)	0.915 (± 0.00)	**0.963 (± 0.01)**	0.885 (± 0.00)	0.929 (± 0.00)	0.861 (± 0.00)	0.903 (± 0.00)	0.912 (± 0.01)	0.909 (± 0.01)	0.938 (± 0.01)
0.10	0.589 (± 0.00)	0.827 (± 0.00)	**0.931 (± 0.00)**	0.865 (± 0.00)	0.864 (± 0.00)	0.887 (± 0.00)	0.889 (± 0.00)	0.855 (± 0.03)	0.854 (± 0.02)	0.923 (± 0.00)
0.20	0.589 (± 0.00)	0.806 (± 0.00)	**0.935 (± 0.00)**	0.821 (± 0.00)	0.807 (± 0.00)	0.876 (± 0.00)	0.877 (± 0.00)	0.805 (± 0.03)	0.807 (± 0.03)	0.877 (± 0.00)
0.30	0.588 (± 0.00)	0.770 (± 0.00)	**0.924 (± 0.00)**	0.758 (± 0.00)	0.780 (± 0.00)	0.889 (± 0.00)	0.872 (± 0.00)	0.742 (± 0.03)	0.781 (± 0.04)	0.868 (± 0.00)
0.40	0.590 (± 0.00)	0.703 (± 0.00)	**0.906 (± 0.00)**	0.711 (± 0.00)	0.728 (± 0.00)	0.874 (± 0.00)	0.865 (± 0.00)	0.710 (± 0.02)	0.746 (± 0.04)	0.859 (± 0.00)
0.50	0.587 (± 0.00)	0.626 (± 0.00)	**0.922 (± 0.00)**	0.676 (± 0.00)	0.693 (± 0.00)	0.864 (± 0.00)	0.841 (± 0.00)	0.676 (± 0.03)	0.721 (± 0.04)	0.804 (± 0.00)
0.60	0.584 (± 0.00)	0.567 (± 0.00)	**0.881 (± 0.00)**	0.598 (± 0.00)	0.614 (± 0.00)	0.877 (± 0.00)	0.793 (± 0.00)	0.597 (± 0.04)	0.663 (± 0.06)	0.779 (± 0.00)
0.70	0.582 (± 0.00)	0.506 (± 0.00)	**0.868 (± 0.00)**	0.548 (± 0.00)	0.553 (± 0.00)	0.831 (± 0.00)	0.771 (± 0.00)	0.528 (± 0.02)	0.601 (± 0.06)	0.766 (± 0.00)
0.80	0.592 (± 0.00)	0.397 (± 0.00)	**0.852 (± 0.00)**	0.496 (± 0.00)	0.522 (± 0.00)	0.807 (± 0.00)	0.730 (± 0.00)	0.465 (± 0.03)	0.509 (± 0.06)	0.728 (± 0.00)
0.90	0.593 (± 0.00)	0.389 (± 0.00)	**0.744 (± 0.00)**	0.361 (± 0.00)	0.423 (± 0.00)	0.701 (± 0.00)	0.628 (± 0.00)	0.407 (± 0.04)	0.395 (± 0.02)	0.646 (± 0.00)
0.99	0.592 (± 0.00)	0.289 (± 0.00)	0.260 (± 0.00)	0.285 (± 0.00)	0.282 (± 0.00)	**0.630 (± 0.00)**	0.333 (± 0.00)	0.278 (± 0.01)	0.276 (± 0.01)	0.412 (± 0.00)

Table C.31: F1 scores for ELECTRIC under mechanism *S-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.588 (± 0.00)	0.915 (± 0.00)	**0.963 (± 0.01)**	0.885 (± 0.00)	0.929 (± 0.00)	0.861 (± 0.00)	0.903 (± 0.00)	0.909 (± 0.01)	0.912 (± 0.01)	0.938 (± 0.01)
0.10	0.493 (± 0.00)	0.891 (± 0.00)	**0.959 (± 0.00)**	0.831 (± 0.00)	0.853 (± 0.00)	0.862 (± 0.00)	0.854 (± 0.00)	0.872 (± 0.01)	0.873 (± 0.02)	0.904 (± 0.00)
0.20	0.484 (± 0.00)	0.855 (± 0.00)	**0.945 (± 0.00)**	0.821 (± 0.00)	0.851 (± 0.00)	0.867 (± 0.00)	0.870 (± 0.00)	0.833 (± 0.01)	0.842 (± 0.03)	0.878 (± 0.00)
0.30	0.478 (± 0.00)	0.816 (± 0.00)	**0.935 (± 0.00)**	0.768 (± 0.00)	0.796 (± 0.00)	0.872 (± 0.00)	0.856 (± 0.00)	0.776 (± 0.02)	0.805 (± 0.02)	0.855 (± 0.00)
0.40	0.483 (± 0.00)	0.756 (± 0.00)	**0.940 (± 0.00)**	0.703 (± 0.00)	0.734 (± 0.00)	0.842 (± 0.00)	0.871 (± 0.00)	0.736 (± 0.03)	0.754 (± 0.01)	0.801 (± 0.00)
0.50	0.431 (± 0.00)	0.708 (± 0.00)	**0.926 (± 0.00)**	0.656 (± 0.00)	0.665 (± 0.00)	0.839 (± 0.00)	0.844 (± 0.00)	0.682 (± 0.02)	0.712 (± 0.01)	0.810 (± 0.00)
0.60	0.397 (± 0.00)	0.632 (± 0.00)	**0.898 (± 0.00)**	0.619 (± 0.00)	0.617 (± 0.00)	0.813 (± 0.00)	0.808 (± 0.00)	0.627 (± 0.03)	0.651 (± 0.01)	0.787 (± 0.00)
0.70	0.435 (± 0.00)	0.563 (± 0.00)	**0.870 (± 0.00)**	0.528 (± 0.00)	0.545 (± 0.00)	0.799 (± 0.00)	0.776 (± 0.00)	0.543 (± 0.04)	0.586 (± 0.05)	0.711 (± 0.00)
0.80	0.490 (± 0.00)	0.522 (± 0.00)	**0.806 (± 0.00)**	0.475 (± 0.00)	0.455 (± 0.00)	0.764 (± 0.00)	0.770 (± 0.00)	0.477 (± 0.03)	0.493 (± 0.02)	0.676 (± 0.00)
0.90	0.374 (± 0.00)	0.392 (± 0.00)	**0.771 (± 0.00)**	0.420 (± 0.00)	0.394 (± 0.00)	0.738 (± 0.00)	0.496 (± 0.00)	0.374 (± 0.03)	0.381 (± 0.03)	0.567 (± 0.00)
0.99	0.260 (± 0.00)	0.265 (± 0.00)	0.260 (± 0.00)	0.269 (± 0.00)	0.277 (± 0.00)	**0.639 (± 0.00)**	0.285 (± 0.00)	0.267 (± 0.01)	0.267 (± 0.01)	0.479 (± 0.00)

Table C.32: F1 scores for ELECTRIC under mechanism *CD-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.588 (± 0.00)	0.915 (± 0.00)	**0.963 (± 0.01)**	0.885 (± 0.00)	0.929 (± 0.00)	0.861 (± 0.00)	0.903 (± 0.00)	0.908 (± 0.01)	0.911 (± 0.01)	0.920 (± 0.00)
0.10	0.585 (± 0.00)	0.794 (± 0.00)	**0.910 (± 0.00)**	0.828 (± 0.00)	0.843 (± 0.00)	0.890 (± 0.00)	0.894 (± 0.00)	0.804 (± 0.03)	0.804 (± 0.03)	0.867 (± 0.00)
0.20	0.584 (± 0.00)	0.687 (± 0.00)	**0.920 (± 0.00)**	0.710 (± 0.00)	0.762 (± 0.00)	0.860 (± 0.00)	0.842 (± 0.00)	0.804 (± 0.03)	0.805 (± 0.03)	0.815 (± 0.00)
0.30	0.591 (± 0.00)	0.604 (± 0.00)	0.650 (± 0.00)	0.672 (± 0.00)	0.693 (± 0.00)	0.815 (± 0.00)	**0.820 (± 0.00)**	0.635 (± 0.01)	0.635 (± 0.01)	0.793 (± 0.00)
0.40	0.587 (± 0.00)	0.475 (± 0.00)	0.630 (± 0.00)	0.475 (± 0.00)	0.494 (± 0.00)	**0.729 (± 0.00)**	0.723 (± 0.00)	0.263 (± 0.01)	0.263 (± 0.01)	0.685 (± 0.00)
0.50	0.589 (± 0.00)	0.301 (± 0.00)	**0.630 (± 0.00)**	0.260 (± 0.00)	0.260 (± 0.00)	0.630 (± 0.00)	0.260 (± 0.00)	0.265 (± 0.01)	0.265 (± 0.01)	0.532 (± 0.00)
0.60	0.593 (± 0.00)	0.271 (± 0.00)	**0.630 (± 0.00)**	0.260 (± 0.00)	0.260 (± 0.00)	0.630 (± 0.00)	0.260 (± 0.00)	0.265 (± 0.01)	0.265 (± 0.01)	0.517 (± 0.00)
0.70	0.589 (± 0.00)	0.310 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	**0.629 (± 0.00)**	0.260 (± 0.00)	0.267 (± 0.01)	0.267 (± 0.01)	0.571 (± 0.00)
0.80	0.593 (± 0.00)	0.343 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	0.263 (± 0.00)	**0.630 (± 0.00)**	0.260 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	0.544 (± 0.00)
0.90	0.589 (± 0.00)	0.315 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	0.263 (± 0.00)	**0.630 (± 0.00)**	0.260 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	0.538 (± 0.00)
0.99	0.589 (± 0.00)	0.330 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	0.260 (± 0.00)	**0.630 (± 0.00)**	0.260 (± 0.00)	0.382 (± 0.01)	0.423 (± 0.02)	0.552 (± 0.00)

Table C.33: F1 scores for ELECTRIC under mechanism *FD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.588 (± 0.00)	0.915 (± 0.00)	**0.963 (± 0.01)**	0.885 (± 0.00)	0.929 (± 0.00)	0.861 (± 0.00)	0.903 (± 0.00)	0.911 (± 0.01)	0.913 (± 0.01)	0.938 (± 0.01)
0.10	0.468 (± 0.15)	0.879 (± 0.01)	**0.944 (± 0.02)**	0.862 (± 0.03)	0.844 (± 0.02)	0.878 (± 0.03)	0.870 (± 0.04)	0.840 (± 0.03)	0.851 (± 0.02)	0.916 (± 0.01)
0.20	0.491 (± 0.13)	0.850 (± 0.01)	**0.938 (± 0.01)**	0.808 (± 0.02)	0.813 (± 0.02)	0.867 (± 0.01)	0.859 (± 0.02)	0.789 (± 0.02)	0.802 (± 0.03)	0.906 (± 0.00)
0.30	0.496 (± 0.13)	0.800 (± 0.02)	**0.922 (± 0.03)**	0.744 (± 0.02)	0.793 (± 0.01)	0.864 (± 0.03)	0.861 (± 0.01)	0.727 (± 0.02)	0.798 (± 0.03)	0.877 (± 0.01)
0.40	0.506 (± 0.12)	0.772 (± 0.04)	**0.906 (± 0.03)**	0.701 (± 0.03)	0.751 (± 0.03)	0.850 (± 0.02)	0.839 (± 0.01)	0.674 (± 0.02)	0.726 (± 0.03)	0.864 (± 0.01)
0.50	0.438 (± 0.12)	0.743 (± 0.01)	**0.877 (± 0.01)**	0.648 (± 0.03)	0.707 (± 0.02)	0.842 (± 0.02)	0.817 (± 0.03)	0.642 (± 0.05)	0.699 (± 0.08)	0.837 (± 0.02)
0.60	0.331 (± 0.05)	0.688 (± 0.02)	**0.836 (± 0.03)**	0.594 (± 0.02)	0.663 (± 0.01)	0.807 (± 0.05)	0.775 (± 0.02)	0.590 (± 0.03)	0.607 (± 0.03)	0.806 (± 0.01)
0.70	0.461 (± 0.14)	0.626 (± 0.01)	**0.834 (± 0.04)**	0.514 (± 0.04)	0.590 (± 0.02)	0.776 (± 0.02)	0.761 (± 0.02)	0.482 (± 0.03)	0.433 (± 0.09)	0.760 (± 0.02)
0.80	0.435 (± 0.12)	0.570 (± 0.02)	0.742 (± 0.06)	0.463 (± 0.01)	0.490 (± 0.04)	**0.743 (± 0.04)**	0.700 (± 0.01)	0.436 (± 0.01)	0.328 (± 0.04)	0.707 (± 0.01)
0.90	0.275 (± 0.01)	0.484 (± 0.01)	0.560 (± 0.22)	0.330 (± 0.06)	0.426 (± 0.03)	**0.663 (± 0.03)**	0.500 (± 0.17)	0.352 (± 0.03)	0.276 (± 0.01)	0.620 (± 0.02)
0.99	0.342 (± 0.12)	0.377 (± 0.03)	0.260 (± 0.00)	0.260 (± 0.00)	0.347 (± 0.01)	**0.629 (± 0.00)**	0.274 (± 0.01)	0.286 (± 0.02)	0.273 (± 0.01)	0.537 (± 0.08)

Table C.34: F1 scores for ELECTRIC under mechanism *CD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.588 (± 0.00)	0.915 (± 0.00)	**0.963 (± 0.01)**	0.885 (± 0.00)	0.929 (± 0.00)	0.861 (± 0.00)	0.903 (± 0.00)	0.908 (± 0.01)	0.908 (± 0.01)	0.938 (± 0.01)
0.10	0.486 (± 0.12)	0.888 (± 0.01)	**0.962 (± 0.01)**	0.869 (± 0.01)	0.874 (± 0.01)	0.908 (± 0.02)	0.885 (± 0.03)	0.839 (± 0.03)	0.860 (± 0.02)	0.922 (± 0.00)
0.20	0.476 (± 0.15)	0.851 (± 0.02)	**0.931 (± 0.02)**	0.815 (± 0.03)	0.802 (± 0.01)	0.879 (± 0.01)	0.879 (± 0.01)	0.805 (± 0.03)	0.801 (± 0.00)	0.902 (± 0.03)
0.30	0.478 (± 0.16)	0.819 (± 0.04)	**0.922 (± 0.00)**	0.789 (± 0.03)	0.789 (± 0.01)	0.872 (± 0.01)	0.880 (± 0.01)	0.770 (± 0.05)	0.736 (± 0.01)	0.890 (± 0.00)
0.40	0.431 (± 0.11)	0.807 (± 0.02)	**0.902 (± 0.01)**	0.775 (± 0.01)	0.762 (± 0.01)	0.835 (± 0.02)	0.865 (± 0.02)	0.749 (± 0.03)	0.685 (± 0.05)	0.869 (± 0.02)
0.50	0.450 (± 0.09)	0.758 (± 0.02)	**0.867 (± 0.03)**	0.722 (± 0.02)	0.748 (± 0.01)	0.835 (± 0.03)	0.827 (± 0.03)	0.656 (± 0.03)	0.633 (± 0.04)	0.850 (± 0.02)
0.60	0.436 (± 0.10)	0.706 (± 0.01)	**0.853 (± 0.05)**	0.663 (± 0.02)	0.608 (± 0.01)	0.847 (± 0.03)	0.780 (± 0.02)	0.664 (± 0.03)	0.593 (± 0.04)	0.836 (± 0.02)
0.70	0.337 (± 0.03)	0.604 (± 0.03)	**0.812 (± 0.03)**	0.585 (± 0.03)	0.538 (± 0.03)	0.770 (± 0.09)	0.729 (± 0.01)	0.560 (± 0.02)	0.514 (± 0.02)	0.765 (± 0.01)
0.80	0.411 (± 0.09)	0.594 (± 0.02)	**0.824 (± 0.08)**	0.540 (± 0.01)	0.486 (± 0.01)	0.703 (± 0.04)	0.671 (± 0.01)	0.513 (± 0.01)	0.469 (± 0.02)	0.742 (± 0.02)
0.90	0.392 (± 0.11)	0.531 (± 0.02)	**0.735 (± 0.07)**	0.473 (± 0.03)	0.449 (± 0.02)	0.686 (± 0.06)	0.600 (± 0.04)	0.434 (± 0.02)	0.445 (± 0.04)	0.683 (± 0.02)
0.99	0.304 (± 0.02)	0.329 (± 0.04)	0.264 (± 0.01)	0.303 (± 0.02)	0.294 (± 0.01)	**0.629 (± 0.00)**	0.312 (± 0.04)	0.305 (± 0.02)	0.292 (± 0.02)	0.561 (± 0.02)

Table C.35: F1 scores for TADPOLE under mechanism *U-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.804 (± 0.00)	0.648 (± 0.01)	0.790 (± 0.00)	0.806 (± 0.00)	0.832 (± 0.02)	0.786 (± 0.00)	0.792 (± 0.00)	**0.847 (± 0.03)**	0.847 (± 0.03)	0.809 (± 0.00)
0.10	0.789 (± 0.00)	0.590 (± 0.00)	0.795 (± 0.00)	0.801 (± 0.00)	0.832 (± 0.00)	0.809 (± 0.00)	0.821 (± 0.00)	**0.841 (± 0.03)**	0.837 (± 0.03)	0.820 (± 0.00)
0.20	0.808 (± 0.00)	0.590 (± 0.00)	0.803 (± 0.00)	0.823 (± 0.00)	**0.836 (± 0.00)**	0.779 (± 0.00)	0.802 (± 0.00)	0.833 (± 0.03)	0.827 (± 0.04)	0.799 (± 0.00)
0.30	0.814 (± 0.00)	0.567 (± 0.00)	0.791 (± 0.00)	0.806 (± 0.00)	**0.825 (± 0.00)**	0.757 (± 0.00)	0.803 (± 0.00)	0.811 (± 0.03)	0.813 (± 0.03)	0.802 (± 0.00)
0.40	0.804 (± 0.00)	0.610 (± 0.00)	**0.831 (± 0.00)**	0.800 (± 0.00)	0.820 (± 0.00)	0.794 (± 0.00)	0.799 (± 0.00)	0.830 (± 0.01)	0.819 (± 0.02)	0.805 (± 0.00)
0.50	0.752 (± 0.00)	0.581 (± 0.00)	0.813 (± 0.00)	0.809 (± 0.00)	**0.830 (± 0.00)**	0.799 (± 0.00)	0.810 (± 0.00)	0.797 (± 0.03)	0.790 (± 0.03)	0.814 (± 0.00)
0.60	0.756 (± 0.00)	0.575 (± 0.00)	0.808 (± 0.00)	0.785 (± 0.00)	0.797 (± 0.00)	0.722 (± 0.00)	0.791 (± 0.00)	**0.810 (± 0.05)**	0.771 (± 0.04)	0.799 (± 0.00)
0.70	0.610 (± 0.00)	0.552 (± 0.00)	0.795 (± 0.00)	0.740 (± 0.00)	0.772 (± 0.00)	0.729 (± 0.00)	0.762 (± 0.00)	0.779 (± 0.04)	0.767 (± 0.03)	**0.802 (± 0.00)**
0.80	0.669 (± 0.00)	0.552 (± 0.00)	**0.804 (± 0.00)**	0.757 (± 0.00)	0.728 (± 0.00)	0.669 (± 0.00)	0.775 (± 0.00)	0.760 (± 0.05)	0.736 (± 0.04)	0.764 (± 0.00)
0.90	0.759 (± 0.00)	0.590 (± 0.00)	0.241 (± 0.00)	0.758 (± 0.00)	0.408 (± 0.00)	0.608 (± 0.00)	0.767 (± 0.00)	**0.786 (± 0.02)**	0.704 (± 0.02)	0.763 (± 0.00)
0.99	**0.707 (± 0.00)**	0.523 (± 0.00)	0.241 (± 0.00)	0.241 (± 0.00)	0.241 (± 0.00)	0.353 (± 0.00)	0.241 (± 0.00)	0.507 (± 0.22)	0.241 (± 0.00)	0.700 (± 0.00)

Table C.36: F1 scores for TADPOLE under mechanism *S-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.804 (± 0.00)	0.648 (± 0.01)	0.790 (± 0.00)	0.806 (± 0.00)	0.832 (± 0.02)	0.786 (± 0.00)	0.792 (± 0.00)	**0.847 (± 0.03)**	0.847 (± 0.03)	0.831 (± 0.04)
0.10	0.554 (± 0.00)	0.542 (± 0.00)	0.805 (± 0.00)	0.803 (± 0.00)	0.815 (± 0.00)	0.751 (± 0.00)	0.804 (± 0.00)	**0.848 (± 0.02)**	0.846 (± 0.02)	0.810 (± 0.00)
0.20	0.497 (± 0.00)	0.486 (± 0.00)	0.818 (± 0.00)	0.818 (± 0.00)	0.811 (± 0.00)	0.737 (± 0.00)	0.814 (± 0.00)	**0.846 (± 0.02)**	0.845 (± 0.03)	0.794 (± 0.00)
0.30	0.523 (± 0.00)	0.502 (± 0.00)	0.775 (± 0.00)	0.799 (± 0.00)	0.825 (± 0.00)	0.777 (± 0.00)	0.818 (± 0.00)	0.837 (± 0.02)	**0.838 (± 0.02)**	0.775 (± 0.00)
0.40	0.482 (± 0.00)	0.581 (± 0.00)	0.800 (± 0.00)	0.797 (± 0.00)	0.794 (± 0.00)	0.719 (± 0.00)	0.784 (± 0.00)	0.820 (± 0.03)	**0.823 (± 0.04)**	0.790 (± 0.00)
0.50	0.501 (± 0.00)	0.523 (± 0.00)	0.757 (± 0.00)	0.777 (± 0.00)	0.769 (± 0.00)	0.739 (± 0.00)	0.798 (± 0.00)	**0.803 (± 0.02)**	0.797 (± 0.03)	0.795 (± 0.00)
0.60	0.539 (± 0.00)	0.498 (± 0.00)	0.802 (± 0.00)	0.769 (± 0.00)	0.734 (± 0.00)	0.693 (± 0.00)	0.804 (± 0.00)	0.804 (± 0.05)	0.799 (± 0.04)	**0.816 (± 0.00)**
0.70	0.480 (± 0.00)	0.453 (± 0.00)	0.748 (± 0.00)	0.719 (± 0.00)	0.738 (± 0.00)	0.642 (± 0.00)	0.752 (± 0.00)	0.784 (± 0.03)	0.777 (± 0.05)	**0.795 (± 0.00)**
0.80	0.502 (± 0.00)	0.422 (± 0.00)	0.689 (± 0.00)	0.736 (± 0.00)	0.703 (± 0.00)	0.555 (± 0.00)	0.730 (± 0.00)	0.739 (± 0.02)	0.740 (± 0.06)	**0.812 (± 0.00)**
0.90	0.377 (± 0.00)	0.280 (± 0.00)	0.503 (± 0.00)	0.680 (± 0.00)	0.650 (± 0.00)	0.420 (± 0.00)	0.739 (± 0.00)	0.662 (± 0.07)	0.557 (± 0.06)	**0.742 (± 0.00)**
0.99	0.272 (± 0.00)	0.249 (± 0.00)	0.241 (± 0.00)	**0.384 (± 0.00)**	0.241 (± 0.00)	0.241 (± 0.00)	0.241 (± 0.00)	0.323 (± 0.05)	0.241 (± 0.00)	0.370 (± 0.00)

Table C.37: F1 scores for TADPOLE under mechanism *CD-MCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.804 (± 0.00)	0.648 (± 0.01)	0.790 (± 0.00)	0.806 (± 0.00)	0.832 (± 0.02)	0.786 (± 0.00)	0.792 (± 0.00)	**0.847 (± 0.03)**	0.847 (± 0.03)	0.831 (± 0.04)
0.10	0.786 (± 0.00)	0.550 (± 0.00)	0.765 (± 0.00)	0.760 (± 0.00)	0.793 (± 0.00)	0.789 (± 0.00)	0.785 (± 0.00)	0.809 (± 0.03)	0.809 (± 0.03)	**0.815 (± 0.00)**
0.20	0.785 (± 0.00)	0.462 (± 0.00)	0.758 (± 0.00)	0.777 (± 0.00)	0.786 (± 0.00)	0.763 (± 0.00)	0.804 (± 0.00)	**0.810 (± 0.04)**	0.810 (± 0.04)	0.806 (± 0.00)
0.30	0.654 (± 0.00)	0.517 (± 0.00)	0.766 (± 0.00)	0.788 (± 0.00)	0.784 (± 0.00)	0.779 (± 0.00)	0.782 (± 0.00)	**0.802 (± 0.04)**	0.802 (± 0.04)	0.800 (± 0.00)
0.40	0.685 (± 0.00)	0.550 (± 0.00)	0.780 (± 0.00)	0.764 (± 0.00)	0.780 (± 0.00)	0.779 (± 0.00)	0.774 (± 0.00)	**0.795 (± 0.03)**	0.795 (± 0.03)	0.780 (± 0.00)
0.50	0.778 (± 0.00)	0.558 (± 0.00)	0.700 (± 0.00)	0.728 (± 0.00)	0.776 (± 0.00)	0.746 (± 0.00)	0.731 (± 0.00)	0.773 (± 0.04)	0.773 (± 0.04)	**0.785 (± 0.00)**
0.60	**0.783 (± 0.00)**	0.508 (± 0.00)	0.731 (± 0.00)	0.708 (± 0.00)	0.729 (± 0.00)	0.760 (± 0.00)	0.714 (± 0.00)	0.767 (± 0.03)	0.767 (± 0.03)	0.745 (± 0.00)
0.70	0.725 (± 0.00)	0.545 (± 0.00)	0.684 (± 0.00)	0.638 (± 0.00)	0.663 (± 0.00)	0.704 (± 0.00)	0.710 (± 0.00)	**0.739 (± 0.03)**	0.739 (± 0.03)	0.722 (± 0.00)
0.80	**0.656 (± 0.00)**	0.442 (± 0.00)	0.576 (± 0.00)	0.391 (± 0.00)	0.442 (± 0.00)	0.543 (± 0.00)	0.419 (± 0.00)	0.643 (± 0.04)	0.643 (± 0.04)	0.615 (± 0.00)
0.90	**0.704 (± 0.00)**	0.419 (± 0.00)	0.241 (± 0.00)	0.348 (± 0.00)	0.361 (± 0.00)	0.337 (± 0.00)	0.292 (± 0.00)	0.327 (± 0.03)	0.327 (± 0.03)	0.409 (± 0.00)
0.99	0.687 (± 0.00)	0.402 (± 0.00)	0.241 (± 0.00)	0.348 (± 0.00)	0.361 (± 0.00)	0.337 (± 0.00)	0.292 (± 0.00)	**0.730 (± 0.03)**	0.567 (± 0.17)	0.409 (± 0.00)

Table C.38: F1 scores for TADPOLE under mechanism *FD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.804 (± 0.00)	0.648 (± 0.01)	0.790 (± 0.00)	0.806 (± 0.00)	0.832 (± 0.02)	0.786 (± 0.00)	0.792 (± 0.00)	0.846 (± 0.03)	**0.849 (± 0.03)**	0.831 (± 0.04)
0.10	0.546 (± 0.07)	0.643 (± 0.01)	0.801 (± 0.01)	0.797 (± 0.01)	0.822 (± 0.02)	0.830 (± 0.04)	0.838 (± 0.03)	0.841 (± 0.03)	0.842 (± 0.03)	**0.846 (± 0.04)**
0.20	0.531 (± 0.11)	0.624 (± 0.05)	0.793 (± 0.04)	**0.836 (± 0.01)**	0.810 (± 0.01)	0.832 (± 0.02)	0.827 (± 0.01)	0.832 (± 0.03)	0.817 (± 0.03)	0.796 (± 0.00)
0.30	0.573 (± 0.12)	0.580 (± 0.04)	0.804 (± 0.05)	0.811 (± 0.03)	0.806 (± 0.04)	0.829 (± 0.04)	**0.831 (± 0.02)**	0.827 (± 0.03)	0.802 (± 0.03)	0.828 (± 0.03)
0.40	0.562 (± 0.09)	0.615 (± 0.03)	0.751 (± 0.03)	0.803 (± 0.04)	0.793 (± 0.04)	**0.811 (± 0.02)**	0.802 (± 0.03)	0.806 (± 0.03)	0.803 (± 0.03)	0.781 (± 0.02)
0.50	0.673 (± 0.04)	0.646 (± 0.07)	0.793 (± 0.02)	0.789 (± 0.02)	0.796 (± 0.05)	0.780 (± 0.01)	**0.815 (± 0.03)**	0.809 (± 0.04)	0.805 (± 0.04)	0.784 (± 0.03)
0.60	0.529 (± 0.09)	0.633 (± 0.06)	0.722 (± 0.07)	0.805 (± 0.04)	0.785 (± 0.05)	0.758 (± 0.02)	**0.810 (± 0.03)**	0.803 (± 0.04)	0.792 (± 0.04)	0.795 (± 0.03)
0.70	0.634 (± 0.05)	0.571 (± 0.04)	**0.804 (± 0.03)**	0.795 (± 0.04)	0.746 (± 0.06)	0.720 (± 0.06)	0.795 (± 0.05)	0.776 (± 0.05)	0.748 (± 0.05)	0.780 (± 0.03)
0.80	0.378 (± 0.10)	0.590 (± 0.06)	0.612 (± 0.14)	0.785 (± 0.02)	0.692 (± 0.05)	0.708 (± 0.02)	**0.797 (± 0.03)**	0.776 (± 0.04)	0.720 (± 0.05)	0.765 (± 0.00)
0.90	0.309 (± 0.10)	0.597 (± 0.01)	0.241 (± 0.00)	0.771 (± 0.03)	0.663 (± 0.05)	0.719 (± 0.01)	**0.787 (± 0.02)**	0.779 (± 0.03)	0.703 (± 0.05)	0.777 (± 0.06)
0.99	0.241 (± 0.00)	0.600 (± 0.05)	0.241 (± 0.00)	0.736 (± 0.03)	0.241 (± 0.00)	0.584 (± 0.05)	0.241 (± 0.00)	0.733 (± 0.01)	0.241 (± 0.00)	**0.794 (± 0.04)**

Table C.39: F1 scores for TADPOLE under mechanism *CD-MNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.804 (± 0.00)	0.648 (± 0.01)	0.790 (± 0.00)	0.806 (± 0.00)	0.832 (± 0.02)	0.786 (± 0.00)	0.792 (± 0.00)	**0.847 (± 0.03)**	0.847 (± 0.03)	0.809 (± 0.00)
0.10	0.553 (± 0.06)	0.534 (± 0.09)	0.793 (± 0.05)	0.813 (± 0.03)	0.829 (± 0.04)	0.792 (± 0.03)	0.806 (± 0.03)	**0.842 (± 0.02)**	0.826 (± 0.04)	0.803 (± 0.01)
0.20	0.485 (± 0.06)	0.515 (± 0.04)	0.804 (± 0.03)	0.812 (± 0.03)	0.832 (± 0.03)	0.810 (± 0.02)	0.806 (± 0.02)	**0.849 (± 0.01)**	0.826 (± 0.04)	0.815 (± 0.02)
0.30	0.441 (± 0.02)	0.584 (± 0.06)	0.805 (± 0.03)	0.785 (± 0.02)	0.811 (± 0.03)	0.786 (± 0.02)	0.812 (± 0.02)	**0.828 (± 0.03)**	0.813 (± 0.04)	0.827 (± 0.03)
0.40	0.502 (± 0.07)	0.671 (± 0.03)	0.828 (± 0.01)	0.818 (± 0.03)	0.808 (± 0.02)	0.793 (± 0.02)	0.814 (± 0.03)	0.824 (± 0.02)	0.826 (± 0.03)	**0.830 (± 0.01)**
0.50	0.448 (± 0.02)	0.621 (± 0.04)	0.784 (± 0.02)	0.804 (± 0.04)	0.799 (± 0.04)	0.756 (± 0.03)	0.803 (± 0.05)	0.819 (± 0.02)	0.800 (± 0.04)	**0.828 (± 0.04)**
0.60	0.457 (± 0.01)	0.529 (± 0.03)	0.791 (± 0.01)	0.781 (± 0.03)	0.803 (± 0.03)	0.710 (± 0.07)	0.797 (± 0.03)	**0.823 (± 0.04)**	0.783 (± 0.03)	0.792 (± 0.03)
0.70	0.485 (± 0.07)	0.590 (± 0.09)	0.639 (± 0.29)	0.797 (± 0.05)	0.787 (± 0.04)	0.710 (± 0.05)	**0.822 (± 0.02)**	0.813 (± 0.03)	0.784 (± 0.07)	0.818 (± 0.01)
0.80	0.376 (± 0.10)	0.605 (± 0.04)	0.434 (± 0.27)	0.785 (± 0.05)	0.767 (± 0.09)	0.744 (± 0.01)	0.798 (± 0.05)	**0.819 (± 0.04)**	0.776 (± 0.04)	0.800 (± 0.02)
0.90	0.362 (± 0.09)	0.563 (± 0.03)	0.241 (± 0.00)	0.788 (± 0.01)	0.730 (± 0.08)	0.689 (± 0.05)	0.776 (± 0.06)	0.771 (± 0.06)	0.704 (± 0.05)	**0.803 (± 0.05)**
0.99	0.324 (± 0.12)	0.547 (± 0.08)	0.241 (± 0.00)	0.255 (± 0.02)	0.241 (± 0.00)	0.348 (± 0.05)	0.241 (± 0.00)	0.558 (± 0.15)	0.241 (± 0.00)	**0.652 (± 0.04)**

C.7 Complete Result Tables – R2 Regime

This appendix complements the analysis of Research Question 3 (Section 5.4). It reports the complete set of results for the R2 regime, where training and test data are subject to different missingness mechanisms. We include both numerical tables (F1-score mean ± std over 5 runs) and extended visualizations across all models and datasets.

C.7.1 Numerical Results

Table C.40 reports the full F1-scores for all models, datasets, and shift configurations considered in the R2 regime.

C.7.2 Extended Visualizations

In addition to Figure 5.3 in the main paper, Figures C.3 and C.5 report the full results for all models under both training mechanisms.

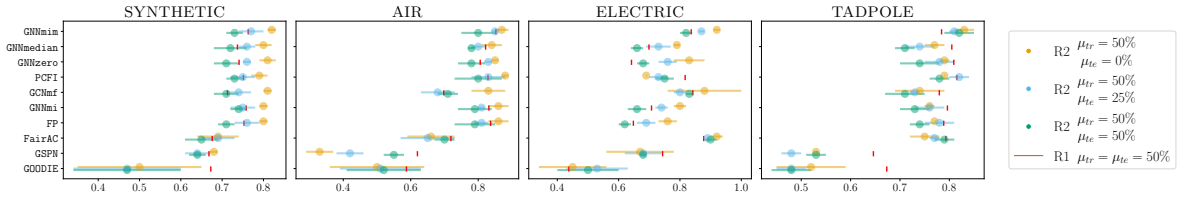


Figure C.3: Full results for all models trained with *FD-MNAR* at $\mu_{\text{tr}} = 50\%$, tested on *U-MCAR* with $\mu_{\text{te}} \in \{0\%, 25\%, 50\%\}$. Each panel corresponds to one dataset; each row to one model. Reported values are mean ± std over 5 runs.

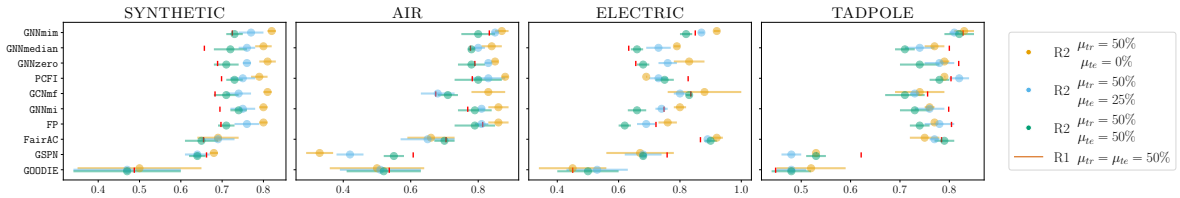


Figure C.4: Full results for all models trained with *CD-MNAR* at $\mu_{\text{tr}} = 50\%$, tested on *U-MCAR* with $\mu_{\text{te}} \in \{0\%, 25\%, 50\%\}$. Same layout as Figure C.3.

Table C.40: F1 (mean ± std over 5 runs). Setup: **R2** missingness distribution shift, where training data are subject to either *FD-MNAR* or *CD-MNAR*, while test data have either no missingness, 25% or 50% of *U-MCAR*

Task	Train mech.	μ Test	GOODIE	GSPN	FairAC	GCNmf	PCFI	FP	GNNmi	GNNzero	GNNmedian	GNNmim
Synthetic	*FD-MNAR*	0	0.50 (± 0.15)	0.68 (± 0.01)	0.69 (± 0.05)	0.81 (± 0.01)	0.79 (± 0.02)	0.80 (± 0.01)	0.80 (± 0.01)	0.81 (± 0.02)	0.80 (± 0.02)	**0.82 (± 0.01)**
	FD-MNAR	0.25	0.47 (± 0.13)	0.64 (± 0.03)	0.69 (± 0.04)	0.74 (± 0.03)	0.75 (± 0.03)	0.76 (± 0.03)	0.75 (± 0.03)	0.76 (± 0.01)	0.76 (± 0.02)	**0.77 (± 0.03)**
	FD-MNAR	0.50	0.47 (± 0.13)	0.64 (± 0.02)	0.65 (± 0.04)	0.71 (± 0.03)	0.73 (± 0.02)	0.71 (± 0.02)	0.74 (± 0.02)	0.71 (± 0.03)	0.72 (± 0.04)	**0.73 (± 0.02)**
	CD-MNAR	0	0.71 (± 0.07)	0.70 (± 0.03)	0.70 (± 0.05)	0.80 (± 0.04)	0.81 (± 0.02)	0.80 (± 0.02)	0.78 (± 0.02)	0.82 (± 0.02)	0.76 (± 0.02)	**0.85 (± 0.04)**
	CD-MNAR	0.25	0.66 (± 0.05)	0.68 (± 0.05)	0.68 (± 0.03)	0.75 (± 0.06)	0.78 (± 0.04)	0.77 (± 0.04)	0.77 (± 0.02)	0.78 (± 0.03)	0.72 (± 0.03)	**0.80 (± 0.03)**
	CD-MNAR	0.50	0.56 (± 0.10)	0.64 (± 0.04)	0.65 (± 0.01)	0.73 (± 0.02)	0.72 (± 0.03)	0.72 (± 0.05)	0.72 (± 0.01)	0.72 (± 0.04)	0.70 (± 0.01)	**0.75 (± 0.03)**
Air	*FD-MNAR*	0	0.50 (± 0.14)	0.33 (± 0.04)	0.66 (± 0.07)	0.83 (± 0.05)	**0.88 (± 0.01)**	0.86 (± 0.03)	0.86 (± 0.03)	0.85 (± 0.01)	0.84 (± 0.03)	0.87 (± 0.02)
	FD-MNAR	0.25	0.51 (± 0.12)	0.42 (± 0.04)	0.65 (± 0.08)	0.68 (± 0.05)	0.83 (± 0.05)	0.81 (± 0.02)	0.81 (± 0.01)	0.83 (± 0.01)	0.80 (± 0.02)	**0.85 (± 0.01)**
	FD-MNAR	0.50	0.52 (± 0.11)	0.55 (± 0.03)	0.70 (± 0.03)	0.71 (± 0.03)	**0.80 (± 0.07)**	0.79 (± 0.06)	0.79 (± 0.05)	0.78 (± 0.04)	0.78 (± 0.01)	0.80 (± 0.05)
	CD-MNAR	0	0.56 (± 0.16)	0.35 (± 0.02)	0.65 (± 0.08)	0.60 (± 0.20)	**0.88 (± 0.01)**	0.71 (± 0.07)	0.86 (± 0.06)	0.83 (± 0.07)	0.82 (± 0.03)	0.85 (± 0.00)
	CD-MNAR	0.25	0.56 (± 0.16)	0.45 (± 0.50)	0.70 (± 0.05)	0.70 (± 0.05)	0.84 (± 0.05)	0.75 (± 0.05)	0.84 (± 0.04)	0.80 (± 0.05)	0.79 (± 0.03)	**0.84 (± 0.06)**
	CD-MNAR	0.50	0.62 (± 0.07)	0.47 (± 0.04)	0.68 (± 0.07)	0.70 (± 0.02)	**0.80 (± 0.05)**	0.72 (± 0.03)	0.76 (± 0.05)	0.76 (± 0.01)	0.74 (± 0.03)	0.76 (± 0.02)
Electric	*FD-MNAR*	0	0.45 (± 0.11)	0.67 (± 0.11)	**0.92 (± 0.02)**	0.88 (± 0.12)	0.69 (± 0.00)	0.76 (± 0.03)	0.80 (± 0.02)	0.83 (± 0.05)	0.79 (± 0.01)	0.92 (± 0.01)
	FD-MNAR	0.25	0.53 (± 0.10)	0.68 (± 0.06)	**0.89 (± 0.00)**	0.80 (± 0.02)	0.73 (± 0.03)	0.69 (± 0.03)	0.74 (± 0.02)	0.76 (± 0.03)	0.73 (± 0.04)	0.87 (± 0.01)
	FD-MNAR	0.50	0.50 (± 0.10)	0.68 (± 0.01)	**0.90 (± 0.02)**	0.83 (± 0.01)	0.75 (± 0.03)	0.62 (± 0.02)	0.66 (± 0.03)	0.68 (± 0.02)	0.66 (± 0.02)	0.82 (± 0.02)
	CD-MNAR	0	0.52 (± 0.10)	0.78 (± 0.04)	0.92 (± 0.02)	0.86 (± 0.01)	0.88 (± 0.01)	0.83 (± 0.05)	0.81 (± 0.01)	0.81 (± 0.01)	0.79 (± 0.02)	**0.94 (± 0.00)**
	CD-MNAR	0.25	0.50 (± 0.10)	0.78 (± 0.01)	**0.88 (± 0.01)**	0.86 (± 0.02)	0.85 (± 0.02)	0.74 (± 0.04)	0.73 (± 0.03)	0.72 (± 0.01)	0.73 (± 0.02)	0.85 (± 0.03)
	CD-MNAR	0.50	0.49 (± 0.12)	0.70 (± 0.02)	**0.87 (± 0.02)**	0.82 (± 0.03)	0.81 (± 0.00)	0.66 (± 0.01)	0.70 (± 0.03)	0.65 (± 0.02)	0.68 (± 0.02)	0.83 (± 0.02)
Tadpole	*FD-MNAR*	0	0.52 (± 0.07)	0.53 (± 0.00)	0.75 (± 0.03)	0.74 (± 0.05)	0.79 (± 0.00)	0.77 (± 0.00)	0.76 (± 0.01)	0.79 (± 0.01)	0.77 (± 0.02)	**0.83 (± 0.02)**
	FD-MNAR	0.25	0.48 (± 0.03)	0.48 (± 0.02)	0.77 (± 0.01)	0.73 (± 0.01)	**0.82 (± 0.02)**	0.78 (± 0.03)	0.76 (± 0.03)	0.78 (± 0.03)	0.74 (± 0.03)	0.81 (± 0.01)
	FD-MNAR	0.50	0.48 (± 0.04)	0.53 (± 0.02)	0.79 (± 0.02)	0.71 (± 0.04)	0.78 (± 0.02)	0.74 (± 0.02)	0.73 (± 0.03)	0.74 (± 0.04)	0.71 (± 0.02)	**0.82 (± 0.03)**
	CD-MNAR	0	0.60 (± 0.02)	0.26 (± 0.02)	0.79 (± 0.05)	0.75 (± 0.04)	**0.80 (± 0.04)**	0.80 (± 0.03)	0.79 (± 0.05)	0.79 (± 0.04)	0.75 (± 0.04)	0.79 (± 0.06)
	CD-MNAR	0.25	0.47 (± 0.09)	0.52 (± 0.02)	0.82 (± 0.05)	0.78 (± 0.01)	**0.80 (± 0.04)**	**0.80 (± 0.04)**	0.77 (± 0.04)	0.78 (± 0.04)	0.73 (± 0.06)	0.75 (± 0.03)
	CD-MNAR	0.50	0.49 (± 0.07)	0.62 (± 0.05)	0.81 (± 0.03)	0.75 (± 0.00)	0.79 (± 0.01)	**0.82 (± 0.02)**	0.76 (± 0.03)	0.76 (± 0.05)	0.73 (± 0.06)	0.74 (± 0.02)

C.8 Inductive Synthetic Setting

In addition to the transductive experiments reported in the main paper, we also ran a set of experiments in an inductive setting to demonstrate that our model, `GNNmim`, is not restricted to transductive scenarios. As shown in Figure C.5, `GNNmim` remains competitive with all other baselines even under this inductive setup.

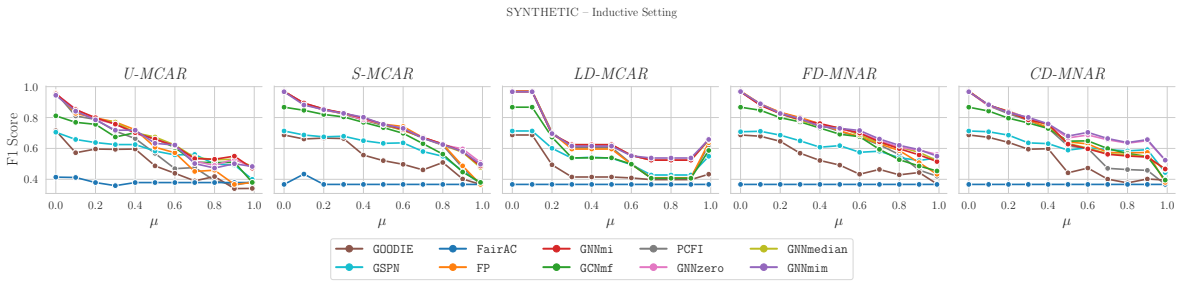


Figure C.5: Performance of `GNNmim` and all competitors in an inductive setting. The synthetic dataset is constructed so that test nodes form a separate graph component and are never connected to training nodes, ensuring that no message can propagate between the two sets during training. Despite this strictly inductive setup, `GNNmim` remains competitive with all baselines.

Table C.41: F1 scores for INDUCTIVE under mechanism *CDMNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.687 (± 0.166)	0.713 (± 0.045)	0.367 (± 0.000)	**0.972 (± 0.011)**	0.968 (± 0.011)	0.867 (± 0.023)	0.970 (± 0.011)	0.968 (± 0.011)	0.968 (± 0.011)	0.967 (± 0.011)
0.10	0.672 (± 0.167)	0.708 (± 0.022)	0.367 (± 0.000)	0.880 (± 0.014)	0.881 (± 0.014)	0.842 (± 0.010)	0.876 (± 0.011)	0.875 (± 0.018)	0.878 (± 0.020)	**0.883 (± 0.020)**
0.20	0.639 (± 0.151)	0.686 (± 0.048)	0.367 (± 0.000)	0.836 (± 0.015)	0.838 (± 0.022)	0.796 (± 0.026)	0.825 (± 0.018)	0.840 (± 0.022)	**0.842 (± 0.020)**	0.832 (± 0.019)
0.30	0.595 (± 0.122)	0.636 (± 0.031)	0.367 (± 0.000)	0.785 (± 0.020)	0.785 (± 0.034)	0.765 (± 0.036)	0.782 (± 0.023)	0.796 (± 0.029)	0.793 (± 0.026)	**0.801 (± 0.020)**
0.40	0.598 (± 0.119)	0.631 (± 0.043)	0.367 (± 0.000)	0.734 (± 0.019)	0.758 (± 0.024)	0.729 (± 0.021)	0.731 (± 0.008)	0.754 (± 0.017)	0.750 (± 0.023)	**0.759 (± 0.017)**
0.50	0.442 (± 0.092)	0.589 (± 0.029)	0.367 (± 0.000)	0.643 (± 0.036)	0.628 (± 0.040)	0.647 (± 0.041)	0.616 (± 0.029)	0.668 (± 0.023)	0.632 (± 0.030)	**0.680 (± 0.018)**
0.60	0.473 (± 0.063)	0.605 (± 0.034)	0.367 (± 0.000)	0.629 (± 0.031)	0.597 (± 0.029)	0.649 (± 0.041)	0.600 (± 0.052)	0.687 (± 0.013)	0.602 (± 0.033)	**0.704 (± 0.021)**
0.70	0.401 (± 0.070)	0.592 (± 0.024)	0.367 (± 0.000)	0.574 (± 0.016)	0.562 (± 0.007)	0.599 (± 0.064)	0.471 (± 0.041)	0.656 (± 0.023)	0.566 (± 0.018)	**0.664 (± 0.027)**
0.80	0.377 (± 0.012)	0.584 (± 0.026)	0.367 (± 0.000)	0.571 (± 0.026)	0.551 (± 0.020)	0.567 (± 0.044)	0.463 (± 0.069)	0.634 (± 0.025)	0.557 (± 0.016)	**0.638 (± 0.028)**
0.90	0.402 (± 0.062)	0.592 (± 0.031)	0.367 (± 0.000)	0.574 (± 0.048)	0.544 (± 0.020)	0.548 (± 0.052)	0.458 (± 0.046)	0.650 (± 0.033)	0.547 (± 0.028)	**0.657 (± 0.020)**
0.99	0.395 (± 0.052)	0.444 (± 0.060)	0.367 (± 0.000)	0.380 (± 0.022)	0.467 (± 0.020)	0.395 (± 0.035)	0.367 (± 0.000)	**0.524 (± 0.045)**	0.464 (± 0.013)	**0.524 (± 0.045)**

Table C.42: F1 scores for INDUCTIVE under mechanism *FDMNAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.687 (± 0.166)	0.708 (± 0.045)	0.367 (± 0.000)	**0.972 (± 0.011)**	0.967 (± 0.011)	0.867 (± 0.022)	0.968 (± 0.013)	0.967 (± 0.011)	0.967 (± 0.011)	0.968 (± 0.011)
0.10	0.679 (± 0.166)	0.711 (± 0.012)	0.367 (± 0.000)	0.888 (± 0.013)	0.879 (± 0.024)	0.847 (± 0.013)	0.885 (± 0.014)	0.882 (± 0.022)	0.886 (± 0.020)	**0.889 (± 0.017)**
0.20	0.646 (± 0.154)	0.686 (± 0.033)	0.367 (± 0.000)	**0.834 (± 0.024)**	0.825 (± 0.024)	0.799 (± 0.016)	0.832 (± 0.026)	0.830 (± 0.022)	0.825 (± 0.025)	0.826 (± 0.028)
0.30	0.569 (± 0.133)	0.649 (± 0.013)	0.367 (± 0.000)	**0.800 (± 0.042)**	0.786 (± 0.034)	0.772 (± 0.028)	0.796 (± 0.025)	0.789 (± 0.036)	0.782 (± 0.032)	0.793 (± 0.036)
0.40	0.522 (± 0.134)	0.608 (± 0.037)	0.367 (± 0.000)	0.759 (± 0.021)	**0.761 (± 0.027)**	0.732 (± 0.032)	0.753 (± 0.026)	0.757 (± 0.032)	0.743 (± 0.028)	0.742 (± 0.032)
0.50	0.492 (± 0.135)	0.618 (± 0.008)	0.367 (± 0.000)	0.714 (± 0.016)	0.731 (± 0.015)	0.692 (± 0.027)	0.710 (± 0.028)	0.724 (± 0.017)	**0.736 (± 0.018)**	0.730 (± 0.019)
0.60	0.433 (± 0.084)	0.575 (± 0.025)	0.367 (± 0.000)	0.675 (± 0.031)	0.699 (± 0.032)	0.676 (± 0.022)	0.674 (± 0.039)	0.702 (± 0.030)	0.687 (± 0.027)	**0.716 (± 0.031)**
0.70	0.464 (± 0.090)	0.582 (± 0.020)	0.367 (± 0.000)	0.630 (± 0.031)	0.643 (± 0.037)	0.594 (± 0.040)	0.623 (± 0.035)	0.651 (± 0.037)	0.635 (± 0.033)	**0.661 (± 0.019)**
0.80	0.429 (± 0.065)	0.540 (± 0.009)	0.367 (± 0.000)	0.586 (± 0.021)	0.598 (± 0.027)	0.527 (± 0.053)	0.560 (± 0.030)	0.607 (± 0.029)	0.609 (± 0.019)	**0.620 (± 0.024)**
0.90	0.444 (± 0.082)	0.522 (± 0.034)	0.367 (± 0.000)	0.508 (± 0.105)	0.558 (± 0.049)	0.486 (± 0.061)	0.460 (± 0.129)	0.589 (± 0.042)	0.575 (± 0.044)	**0.592 (± 0.023)**
0.99	0.370 (± 0.005)	0.538 (± 0.041)	0.367 (± 0.000)	0.433 (± 0.093)	0.515 (± 0.035)	0.454 (± 0.076)	0.420 (± 0.105)	**0.561 (± 0.036)**	0.521 (± 0.040)	0.550 (± 0.040)

Table C.43: F1 scores for INDUCTIVE under mechanism *LDMCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.687 (± 0.166)	0.713 (± 0.045)	0.367 (± 0.000)	**0.972 (± 0.011)**	0.968 (± 0.011)	0.867 (± 0.023)	0.970 (± 0.011)	0.968 (± 0.011)	0.968 (± 0.011)	0.967 (± 0.011)
0.10	0.687 (± 0.166)	0.713 (± 0.045)	0.367 (± 0.000)	**0.972 (± 0.011)**	0.968 (± 0.011)	0.867 (± 0.023)	0.970 (± 0.011)	0.968 (± 0.011)	0.968 (± 0.011)	0.967 (± 0.011)
0.20	0.494 (± 0.117)	0.601 (± 0.039)	0.367 (± 0.000)	0.701 (± 0.023)	0.692 (± 0.031)	0.673 (± 0.036)	**0.705 (± 0.019)**	0.692 (± 0.031)	0.692 (± 0.031)	0.696 (± 0.029)
0.30	0.415 (± 0.076)	0.537 (± 0.032)	0.367 (± 0.000)	0.596 (± 0.010)	**0.624 (± 0.010)**	0.539 (± 0.028)	0.606 (± 0.006)	**0.624 (± 0.010)**	**0.624 (± 0.010)**	0.615 (± 0.011)
0.40	0.415 (± 0.076)	0.543 (± 0.037)	0.367 (± 0.000)	0.596 (± 0.010)	**0.624 (± 0.010)**	0.539 (± 0.028)	0.606 (± 0.006)	**0.624 (± 0.010)**	**0.624 (± 0.010)**	0.615 (± 0.011)
0.50	0.415 (± 0.076)	0.537 (± 0.032)	0.367 (± 0.000)	0.596 (± 0.010)	**0.624 (± 0.010)**	0.539 (± 0.028)	0.606 (± 0.006)	**0.624 (± 0.010)**	**0.624 (± 0.010)**	0.615 (± 0.011)
0.60	0.409 (± 0.053)	0.495 (± 0.044)	0.367 (± 0.000)	0.497 (± 0.015)	**0.555 (± 0.019)**	0.498 (± 0.022)	0.501 (± 0.022)	**0.555 (± 0.019)**	**0.555 (± 0.019)**	0.552 (± 0.027)
0.70	0.398 (± 0.037)	0.428 (± 0.030)	0.367 (± 0.000)	0.410 (± 0.027)	0.524 (± 0.044)	0.407 (± 0.051)	0.407 (± 0.025)	0.524 (± 0.044)	0.524 (± 0.044)	**0.538 (± 0.023)**
0.80	0.398 (± 0.037)	0.428 (± 0.030)	0.367 (± 0.000)	0.410 (± 0.027)	0.524 (± 0.044)	0.407 (± 0.051)	0.407 (± 0.025)	0.524 (± 0.044)	0.524 (± 0.044)	**0.538 (± 0.023)**
0.90	0.398 (± 0.037)	0.428 (± 0.030)	0.367 (± 0.000)	0.410 (± 0.027)	0.524 (± 0.044)	0.407 (± 0.051)	0.407 (± 0.025)	0.524 (± 0.044)	0.524 (± 0.044)	**0.538 (± 0.023)**
0.99	0.433 (± 0.069)	0.549 (± 0.024)	0.367 (± 0.000)	0.637 (± 0.036)	0.659 (± 0.029)	0.587 (± 0.031)	0.623 (± 0.027)	**0.660 (± 0.025)**	0.652 (± 0.025)	0.658 (± 0.023)

Table C.44: F1 scores for INDUCTIVE under mechanism *SMCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.687 (± 0.166)	0.713 (± 0.045)	0.367 (± 0.000)	**0.972 (± 0.011)**	0.968 (± 0.011)	0.867 (± 0.023)	0.970 (± 0.011)	0.968 (± 0.011)	0.968 (± 0.011)	0.967 (± 0.011)
0.10	0.661 (± 0.148)	0.687 (± 0.013)	0.434 (± 0.133)	0.887 (± 0.012)	**0.894 (± 0.016)**	0.847 (± 0.025)	0.891 (± 0.018)	**0.894 (± 0.017)**	0.890 (± 0.021)	0.881 (± 0.018)
0.20	0.667 (± 0.157)	0.675 (± 0.036)	0.367 (± 0.000)	0.850 (± 0.017)	0.855 (± 0.027)	0.820 (± 0.030)	**0.856 (± 0.025)**	0.847 (± 0.018)	0.851 (± 0.027)	0.851 (± 0.028)
0.30	0.664 (± 0.155)	0.679 (± 0.034)	0.367 (± 0.000)	**0.830 (± 0.016)**	0.829 (± 0.032)	0.804 (± 0.028)	0.822 (± 0.025)	0.828 (± 0.032)	0.824 (± 0.034)	0.827 (± 0.038)
0.40	0.557 (± 0.152)	0.650 (± 0.029)	0.367 (± 0.000)	0.785 (± 0.030)	0.796 (± 0.035)	0.769 (± 0.018)	0.785 (± 0.029)	0.785 (± 0.043)	0.790 (± 0.039)	**0.802 (± 0.032)**
0.50	0.521 (± 0.152)	0.633 (± 0.045)	0.367 (± 0.000)	0.757 (± 0.029)	0.758 (± 0.018)	0.735 (± 0.019)	0.748 (± 0.030)	**0.760 (± 0.018)**	0.755 (± 0.019)	0.756 (± 0.009)
0.60	0.497 (± 0.135)	0.636 (± 0.058)	0.367 (± 0.000)	**0.742 (± 0.030)**	0.722 (± 0.034)	0.698 (± 0.021)	0.723 (± 0.038)	0.724 (± 0.039)	0.716 (± 0.031)	0.730 (± 0.027)
0.70	0.461 (± 0.125)	0.580 (± 0.062)	0.367 (± 0.000)	0.670 (± 0.018)	0.671 (± 0.029)	0.631 (± 0.036)	0.666 (± 0.038)	**0.673 (± 0.030)**	0.672 (± 0.028)	0.666 (± 0.035)
0.80	0.509 (± 0.121)	0.549 (± 0.071)	0.367 (± 0.000)	0.628 (± 0.053)	**0.629 (± 0.025)**	0.563 (± 0.070)	0.621 (± 0.044)	0.623 (± 0.013)	0.622 (± 0.025)	0.625 (± 0.037)
0.90	0.402 (± 0.071)	0.455 (± 0.068)	0.367 (± 0.000)	0.487 (± 0.070)	0.580 (± 0.043)	0.447 (± 0.060)	0.474 (± 0.092)	**0.597 (± 0.026)**	0.575 (± 0.039)	0.580 (± 0.027)
0.99	0.367 (± 0.000)	0.372 (± 0.010)	0.367 (± 0.000)	0.367 (± 0.000)	0.486 (± 0.027)	0.380 (± 0.019)	0.367 (± 0.000)	**0.509 (± 0.038)**	0.476 (± 0.024)	0.498 (± 0.031)

Table C.45: F1 scores for INDUCTIVE under mechanism *UMCAR* and varying μ

μ	GOODIE	GSPN	FairAC	FP	GNNmi	GCNmf	PCFI	GNNzero	GNNmedian	GNNmim
0.00	0.715 (± 0.096)	0.705 (± 0.033)	0.414 (± 0.055)	**0.960 (± 0.009)**	0.953 (± 0.006)	0.811 (± 0.030)	**0.960 (± 0.009)**	0.953 (± 0.006)	0.953 (± 0.006)	0.944 (± 0.017)
0.10	0.572 (± 0.137)	0.658 (± 0.031)	0.412 (± 0.057)	0.827 (± 0.050)	0.851 (± 0.043)	0.769 (± 0.112)	0.810 (± 0.034)	**0.855 (± 0.044)**	0.846 (± 0.047)	0.841 (± 0.051)
0.20	0.596 (± 0.165)	0.638 (± 0.025)	0.379 (± 0.000)	0.798 (± 0.033)	**0.799 (± 0.020)**	0.756 (± 0.032)	0.788 (± 0.027)	0.790 (± 0.028)	0.788 (± 0.021)	0.785 (± 0.021)
0.30	0.594 (± 0.145)	0.625 (± 0.014)	0.359 (± 0.040)	**0.771 (± 0.037)**	0.757 (± 0.046)	0.674 (± 0.133)	0.712 (± 0.045)	0.758 (± 0.049)	**0.771 (± 0.042)**	0.718 (± 0.047)
0.40	0.596 (± 0.132)	0.625 (± 0.005)	0.379 (± 0.000)	**0.721 (± 0.055)**	0.702 (± 0.044)	0.702 (± 0.055)	0.664 (± 0.080)	0.697 (± 0.049)	0.701 (± 0.048)	0.718 (± 0.029)
0.50	0.487 (± 0.113)	0.583 (± 0.040)	0.379 (± 0.000)	0.608 (± 0.067)	0.660 (± 0.027)	0.664 (± 0.053)	0.568 (± 0.074)	0.659 (± 0.021)	**0.674 (± 0.022)**	0.633 (± 0.035)
0.60	0.439 (± 0.118)	0.558 (± 0.034)	0.379 (± 0.000)	0.572 (± 0.077)	0.617 (± 0.038)	0.606 (± 0.081)	0.469 (± 0.102)	0.617 (± 0.038)	**0.622 (± 0.039)**	**0.622 (± 0.062)**
0.70	0.390 (± 0.074)	**0.561 (± 0.019)**	0.379 (± 0.000)	0.451 (± 0.092)	0.534 (± 0.073)	0.511 (± 0.095)	0.476 (± 0.118)	0.518 (± 0.076)	0.541 (± 0.089)	0.502 (± 0.092)
0.80	0.418 (± 0.123)	0.499 (± 0.029)	0.379 (± 0.000)	0.459 (± 0.074)	**0.530 (± 0.060)**	0.508 (± 0.088)	0.392 (± 0.087)	0.490 (± 0.059)	0.528 (± 0.044)	0.473 (± 0.052)
0.90	0.340 (± 0.048)	0.493 (± 0.022)	0.379 (± 0.000)	0.367 (± 0.046)	**0.550 (± 0.139)**	0.511 (± 0.082)	0.362 (± 0.041)	0.532 (± 0.134)	0.529 (± 0.131)	0.501 (± 0.122)
0.99	0.341 (± 0.045)	0.400 (± 0.025)	0.379 (± 0.000)	0.379 (± 0.000)	0.472 (± 0.022)	0.380 (± 0.003)	0.384 (± 0.011)	0.476 (± 0.038)	**0.485 (± 0.018)**	0.483 (± 0.033)

C.9 Gain using MIM with competitors

Tables C.46 through C.49 report the performance gain observed when all competitor models described in the main paper are equipped with the MIM mask, mirroring the setup used for `GNNmim`. Consistently, basic imputation methods that replace missing features with a constant, such as `GNNmi`and `GNNmedian`, show a positive and comparable performance increase when supplied with the same mask. This suggests that the improvement comes from the model's ability to selectively ignore the padded or imputed feature values indicated by the mask.

Table C.46: F1 gain from using mask on SYNTHETIC under mechanism *U-MCAR*

μ	FairAC	FP	GCNmf	GNNmedian	GNNmi	GOODIE	GSPN	PCFI	GNNzero
0.00	-0.087	-0.016	-0.145	0.002	0.003	-0.256	-0.094	-0.020	0.005
0.10	-0.094	-0.022	-0.065	0.006	0.005	-0.253	-0.080	-0.004	0.001
0.20	-0.102	-0.013	-0.005	0.002	0.004	-0.215	-0.052	-0.001	0.008
0.30	-0.078	0.002	-0.021	0.012	0.014	-0.198	-0.068	-0.008	0.015
0.40	-0.082	0.008	-0.022	0.012	0.07	-0.223	-0.075	0.006	0.025
0.50	0.011	-0.006	-0.010	0.005	0.09	-0.268	-0.079	-0.018	0.007
0.60	-0.025	-0.004	-0.029	0.004	0.013	-0.346	-0.072	-0.001	0.000
0.70	0.013	0.001	-0.044	0.005	0.004	-0.321	-0.008	0.006	0.006
0.80	-0.070	-0.008	0.009	0.002	0.015	-0.429	0.015	-0.014	0.039
0.90	-0.020	-0.017	-0.011	0.011	0.014	-0.346	0.053	0.001	0.001
0.99	0.052	-0.007	0.056	-0.020	-0.013	-0.422	0.024	-0.011	-0.013

Table C.47: F1 gain from using mask on SYNTHETIC under mechanism *S-MCAR*

μ	FairAC	FP	GCNmf	GNNmedian	GNNmi	GOODIE	GSPN	PCFI	GNNzero
0.00	-0.080	-0.016	-0.145	0.002	0.003	-0.256	-0.091	0.05	0.005
0.10	0.013	0.001	-0.077	0.03	0.04	-0.211	0.005	-0.11	-0.011
0.20	-0.018	-0.039	-0.086	0.003	0.007	-0.245	-0.019	-0.026	0.031
0.30	0.000	-0.026	-0.083	0.006	0.015	-0.234	-0.013	-0.015	0.016
0.40	0.010	-0.034	-0.012	0.002	0.019	-0.185	-0.014	-0.018	0.024
0.50	-0.062	-0.048	0.005	0.006	0.016	-0.207	-0.036	-0.039	0.033
0.60	-0.045	-0.028	-0.038	0.018	0.032	-0.161	0.001	-0.026	0.038
0.70	0.009	-0.007	-0.025	0.011	0.025	-0.153	-0.015	-0.033	0.064
0.80	0.010	-0.011	-0.046	0.011	0.02	-0.136	-0.002	0.004	0.029
0.90	-0.045	0.003	-0.018	0.002	-0.002	-0.071	0.043	-0.000	-0.019
0.99	0.128	-0.024	0.074	0.002	-0.015	0.048	0.033	-0.025	-0.011

Table C.48: F1 gain from using mask on SYNTHETIC under mechanism *LD-MCAR*

μ	FairAC	FP	GCNmf	GNNmedian	GNNmi	GOODIE	GSPN	PCFIGNNzero	
0.00	-0.073	-0.016	-0.145	0.002	0.003	-0.256	-0.094	-0.020	0.005
0.10	-0.047	0.104	-0.012	0.026	0.095	-0.222	-0.014	0.097	-0.08
0.20	-0.105	-0.078	-0.081	0.004	0.075	-0.251	-0.092	-0.067	0.081
0.30	-0.106	-0.119	-0.106	0.015	0.101	-0.331	-0.091	-0.118	0.133
0.40	0.014	-0.044	-0.049	0.015	0.039	-0.337	-0.054	-0.033	0.098
0.50	0.080	-0.002	-0.004	0.015	0.002	-0.362	0.027	0.003	0.077
0.60	-0.079	-0.073	-0.068	0.004	0.081	-0.386	-0.046	-0.069	0.139
0.70	-0.111	-0.084	-0.034	0.001	0.070	-0.423	-0.039	-0.060	0.139
0.80	0.001	-0.084	-0.074	0.001	0.085	-0.422	-0.056	-0.086	0.130
0.90	-0.067	-0.090	-0.066	0.001	0.096	-0.439	0.023	-0.072	0.143
0.99	0.046	0.037	-0.054	0.007	0.014	-0.359	0.025	0.039	0.020

Table C.49: F1 gain from using mask on SYNTHETIC under mechanism *FD-MNAR*

μ	FairAC	FP	GCNmf	GNNmedian	GNNmi	GOODIE	GSPN	PCFIGNNzero	
0.00	-0.080	-0.018	-0.141	0.002	0.003	-0.256	-0.081	-0.018	0.005
0.10	-0.035	-0.006	-0.057	0.007	0.013	-0.216	-0.002	-0.001	0.014
0.20	0.018	0.015	0.024	0.06	0.005	-0.193	-0.012	-0.009	-0.005
0.30	0.021	0.002	-0.005	0.002	0.007	-0.138	0.015	0.016	-0.018
0.40	0.001	-0.007	-0.031	0.006	0.011	-0.186	-0.032	0.003	0.021
0.50	-0.025	-0.011	-0.020	0.008	0.013	-0.208	-0.009	-0.007	0.022
0.60	0.011	0.006	-0.019	0.012	0.008	-0.121	0.030	0.013	0.013
0.70	0.022	0.029	0.004	0.000	0.003	-0.063	0.044	0.013	0.012
0.80	0.010	0.013	-0.010	0.002	0.001	-0.006	-0.017	0.033	0.020
0.90	0.053	0.032	-0.032	0.005	0.011	0.048	-0.023	0.020	0.018
0.99	0.156	0.002	-0.008	0.001	0.010	-0.015	0.006	-0.003	0.007

Table C.50: F1 gain from using mask on SYNTHETIC under mechanism *CD-MNAR*

μ	FairAC	FP	GCNmf	GNNmedian	GNNmi	GOODIE	GSPN	PCFIGNNzero	
0.00	-0.078	-0.016	-0.145	0.002	0.003	-0.256	-0.091	-0.020	0.005
0.10	-0.025	-0.002	-0.060	0.004	0.010	-0.239	-0.019	-0.005	0.001
0.20	0.023	0.006	-0.003	0.004	0.002	-0.202	-0.029	0.004	0.001
0.30	-0.005	0.017	-0.004	0.009	0.007	-0.121	-0.030	-0.006	0.023
0.40	-0.045	0.017	-0.015	0.014	0.017	0.005	-0.024	0.021	0.020
0.50	-0.035	0.010	0.001	0.048	0.010	-0.035	-0.042	0.009	0.036
0.60	0.054	0.036	-0.011	0.019	0.015	-0.111	-0.047	0.073	0.037
0.70	0.038	0.051	0.001	0.025	0.028	-0.064	0.031	0.072	0.026
0.80	0.045	0.046	0.047	0.017	0.011	-0.028	-0.021	0.086	0.037
0.90	0.136	0.033	0.039	0.011	0.021	-0.009	-0.047	0.075	0.037
0.99	0.098	-0.041	0.057	0.017	0.015	-0.050	0.044	0.013	0.018