Advancing Tabular Data Classification with Graph Neural Networks: A Random Forest Proximity Method

Soheila Farokhi*
Department of Computer Science
Utah State University
soheila.farokhi@usu.edu

Haozhe Chen*
Department of Mathematics and Statistics
Utah State University
a02314155@usu.edu

Kevin Moon
Department of Mathematics and Statistics
Utah State University
kevin.moon@usu.edu

Hamid Karimi
Department of Computer Science
Utah State University
hamid.karimi@usu.edu

2024 IEEE International Conference on Big Data (BigData) | 979-8-3503-6248-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/BigData62323.2024.10825972

***Abstract*—Graphs are essential for modeling complex relationships, analyzing networks, and offering versatile representations that capture diverse data structures. Graph Neural Networks (GNNs) excel in processing graph-structured data by leveraging the relational information encoded in graph topology. However, not all types of data possess an *explicit* graph structure, particularly tabular data, which is ubiquitous in the real world. To enable the use of GNNs for tabular data, it is necessary to convert tabular data into graph-structured data. Existing methods for this conversion often lack a generic, straightforward approach with unrestrictive assumptions that can directly apply GNNs to tabular data for downstream tasks. In this paper, we introduce RF-GNN, a novel method that enhances traditional machine learning approaches by transforming tabular data into graph structures and leveraging GNNs. Our approach calculates the similarity between pairs of samples based on Random Forest (RF) proximities, which measure how often the same pair appears in the same terminal nodes of a tree in a Random Forest. This enables the creation of an adjacency matrix for instances of tabular data, allowing the application of GNNs. Extensive experiments on 36 different datasets demonstrate that RF-GNN consistently outperforms traditional machine learning models and recent methods in terms of weighted F1-score. We conduct additional experiments to evaluate the effectiveness of RF-GNN components and settings. The code is available in https://github.com/DSAatUSU/RF-GNN.**

***Index Terms*—Tabular Data, Graph Neural Networks, Random Forest, Graph Representation Learning**

I. Introduction

Graphs are essential for modeling complex relationships, analyzing networks, and offering versatile representations that capture diverse data structures [1]–[6]. Graph Neural Networks (GNNs) [7] are pivotal as they directly process graph-structured data using the relational information encoded in a graph topology. GNNs excel in tasks such as node classification, link prediction, and graph classification across various domains such as social networks [8], [9], biology [10], and education [11]–[13]. Their scalability, efficiency, and versatility make them invaluable for learning from large-scale graph datasets and solving a wide range of real-world problems efficiently and effectively.

However, not all types of data possess an *explicit* graph structure, particularly tabular data which is ubiquitous in the real world. Unlike graph-structured data, which explicitly encodes relationships between entities using nodes and edges, tabular data often lacks such explicit relational information. Instead, tabular datasets are commonly assumed to be independent and identically distributed (i.i.d.), meaning that each observation is drawn from the same underlying distribution and is independent of other observations. Nevertheless, studies have shown that breaking this assumption and creating explicit links (edges) between records (i.e., representing the tabular data in graph form) can bring about many benefits [14]. Among these is the ability to apply the power of GNNs to tabular data for better performance on downstream tasks such as credit loan classification. Now the question is *how can we effectively convert tabular data into a graph structure so we can leverage the power of GNNs?*

In this paper, we introduce RF-GNN, a novel method aimed at improving the performance of traditional machine learning methods by transforming tabular data into graph-structured data and then applying GNNs. Figure 1 shows an overview of our model. Specifically, we propose constructing a Random Forest (RF) proximity matrix for the data. Random Forest proximities offer advantages over similarity measures like Jaccard and cosine similarity, as well as kernel methods like RBF. Random Forests can handle heterogeneous data, capture nonlinear relationships, and perform automatic feature selection, enhancing model robustness and interpretability. Their ensemble nature provides robustness to noise and outliers, making them suitable for real-world datasets. Following the

*Co-first authors and equal contributions.

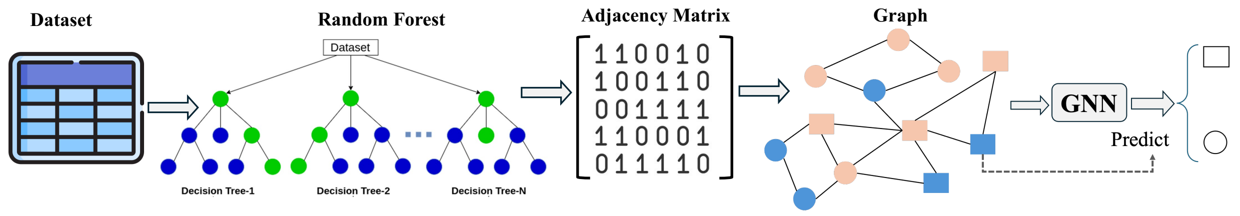


Fig. 1: An overview of the proposed method (RF-GNN). A random forest is first trained on the tabular data. Pairwise proximities are extracted from the random forest and then converted to an adjacency matrix, which is used as input to a GNN.

calculation of proximity values, we apply a threshold to these values to construct an *adjacency matrix*, effectively capturing the relationships between samples and facilitating the generation of pseudo-graphs based on the specified threshold. Through this approach, we investigate the potential of transforming tabular data into structured graphs, enabling the usage of GNNs to enhance machine learning performance.

We present extensive experimental results on 36 different datasets that showcase promising outcomes. These results suggest that converting tabular data into a graph has the potential to augment the efficacy of conventional machine learning algorithms, particularly when an optimal threshold is carefully selected. The integration of spatial relationships and patterns derived from graph representations significantly enhances predictive performance. In summary, our contributions are as follows:

- We introduce a novel, general, and straightforward method based on Random Forest proximities that transforms tabular data into a graph structure for use in a GNN.
- We conduct extensive experiments on 36 benchmark datasets encompassing a wide range of tasks, dataset sizes, and features.
- Our method, RF-GNN, achieves up to a 0.51 improvement in the F1-score over traditional machine learning algorithms. RF-GNN consistently performs on par with or surpasses the current state of the art.

The remainder of this paper is organized as follows: in Section II, we review related research in this area. Section III presents the background, followed by the formal problem statements and notations in Section IV. In Section V, we describe the proposed method in detail. Section VI is dedicated to experiments and discussions. Finally, we conclude the paper in Section VII and suggest future directions.

II. Related Work

With graph representation learning gaining significant popularity in recent years, there have been numerous efforts to represent tabular data in graph form and apply graph embedding techniques for downstream classification or regression tasks. Supervised methods like Deep Graph Learning (DGL) have been explored in several studies [15]–[18]. DGL shows promise by using GNNs to construct and refine the graph topology. These approaches parameterize the adjacency matrix using various models, such as probabilistic models [16], [19], full parameterization models [20], or metric learning models [15], [21]. Further, they optimize both the adjacency matrix and GNN parameters through downstream task optimization. Subsequently, unsupervised DGL methods have emerged, such as the one proposed by Liu et al. [22], which operates without relying on label information, thereby mitigating potential biases in learned edge distributions. However, these approaches prioritize learning improved graphical structures over directly converting tabular data into graphs. IDGL (Iterative Deep Graph Learning) [15] employs a weighted cosine similarity to refine graph learning iteratively, enhancing both the graph structure and node embeddings. While they improve GNN learning by refining noisy graphs and introducing graph structures to non-graphical data, direct conversion of tabular data into graph form is not their focus.

On the other hand, some methods focus on constructing graphs by labels. Rocheteau et al. [23] proposed linking patients with similar diagnoses in a graph to predict outcomes using an LSTM and GNN. However, its reliance on diagnosis outcomes for graph construction limits its generality. Additionally, Li et al. [24] listed numerous methodologies tailored to tabular datasets, some involving direct or indirect graph construction techniques like k-nearest neighbors (NN) and similarity measurement. Yet, model effectiveness crucially hinges on selecting suitable k values and similarity measures. The same issue persists for [25], which uses kNN to construct graphs. In summary, current studies lack a *general* and *straightforward* approach with *nonrestrictive assumptions* that can directly convert tabular data into a graph and apply GNNs for downstream tasks. In addition, only a handful of studies provide public code. Those that do not, are complicated to reproduce, limiting their usability[1].

III. Background

In this section, we review the foundational methodologies that underpin our study.

Random Forests [26] are widely recognized as robust predictors, consisting of an ensemble of binary recursive decision trees. They typically perform well "out of the box" and require little to no tuning. They are adapted for both classification and regression tasks, offer straightforward parallelization, accommodate mixed variable types (continuous and categorical), remain unaffected by monotonic transformations, demonstrate resilience to outliers, scale adeptly to datasets of

[1]See https://github.com/Roytsai27/awesome-GNN4TDL

varying sizes, handle missing values effectively, capture non-linear interactions, and exhibit robustness to noise variables.

Random Forest Proximity: Random forests naturally generate pair-wise proximity measures based on the partitioning space of their constituent decision trees. These proximities, defined by Leo Breiman as the proportion of trees in which observations share the same terminal node [27], encode a supervised similarity measure, leveraging the optimized splitting variable values for the task at hand. Used in various applications such as data visualization [28]–[31], outlier detection [28], [32], and data imputation [33]–[35], random forest proximities extend many unsupervised problems to a supervised context effectively. A more robust proximity metric called Random Forest-Geometry- and Accuracy-Preserving proximities (RF-GAP) has been introduced in [36]. The proximity-weighted sum (regression) or majority vote (classification) using RF-GAP precisely replicates the out-of-bag random forest prediction, capturing the data geometry learned by the model. It outperforms original random forest proximities in tasks like data imputation, outlier detection, and visualization. However, as RF-GAP proximities are computed using weighted sums of training points, the test-test proximity is inherently zero. Hence, we chose to employ the original RF proximity in our study due to this limitation as the test-test proximities are required for our method.

IV. Problem Statement

Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ be the given tabular dataset, where $\mathbf{x}_i \in \mathbb{R}^d$ are the data points in the dataset and $y_i \in \mathbb{N}_0$ are the labels. For each dataset, we use $c \geq 2$ to indicate the number of classes. We use $\mathcal{X}$ to represent all data points, i.e. $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and $\mathcal{Y}$ to represent labels, i.e. $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$. Our objective is to first develop a model $f(\cdot)$ that, given a tabular dataset $\mathcal{D}$, constructs a graph where data points serve as nodes and edges between these nodes indicate that the data points exhibit high similarity. We can write this as:

$$A = f(\mathcal{X}, \mathcal{Y}_{train}), \tag{1}$$

where $\mathcal{X}$ is defined above, $\mathcal{Y}_{train}$ is a subset of labels $\mathcal{Y}$ chosen for training model $f(\cdot)$, and A is the adjacency matrix for the resulting graph. Then we employ a Graph Neural Network (GNN) that, given the above adjacency matrix, node features $\mathcal{X}$, and target labels $\mathcal{Y}_{train}$ learns to predict labels of test data points $\mathcal{Y}_{test}$. The model's predictive function can be mathematically represented as follows:

$$\hat{y_i} = \phi(A, \mathcal{X}, \mathcal{Y}_{train}), \tag{2}$$

where $\hat{y_i} \in \mathbb{R}^c$ is the vector that represents the probability of data point x_i belonging to each possible class, and ϕ represents any GNN model that is designed for node classification.

V. The Proposed Method (RF-GNN)

In this section, we detail RF-GNN, a novel approach that combines the robust feature extraction capabilities of random forests with the dynamic relational modeling strength of GNNs. Figure 2 illustrates our proposed method. RF-GNN first transforms tabular data into a structured graph format, enabling the application of advanced neural network techniques tailored for graph data described in Section V-A. In Section V-B, we elaborate on how the graph-structured data are processed using Graph Convolutional Networks (GCNs).

A. Graph Construction

To leverage graph embedding techniques, we construct a graph that captures the similarity between instances in a tabular dataset using random forest proximities. The resulting proximity matrix quantifies the similarity between pairs of data points based on their co-occurrence in the same terminal node (leaf) across multiple decision trees within a random forest model. High proximity indicates that data points are "close" or "similar" within the forest's decision-making context, providing insights into relationships in the feature space. This method is useful for tasks such as cluster identification and outlier detection [36] and as a foundation for other machine-learning models.

We begin by performing a grid search with 5-fold cross-validation to identify the optimal random forest model trained on the training set. After selecting the best model, we fit it to the training data. Then, we apply the model to the whole dataset, which applies the trees in the forest to all data samples and returns leaf indices. In other words, for each data point $\mathbf{x} \in \mathcal{X}$ and for each tree in the forest, this method returns the index of the leaf $\mathbf{x}$ ends up in without fitting the model to the whole data. The random forest proximity between observations i and j is computed as follows:

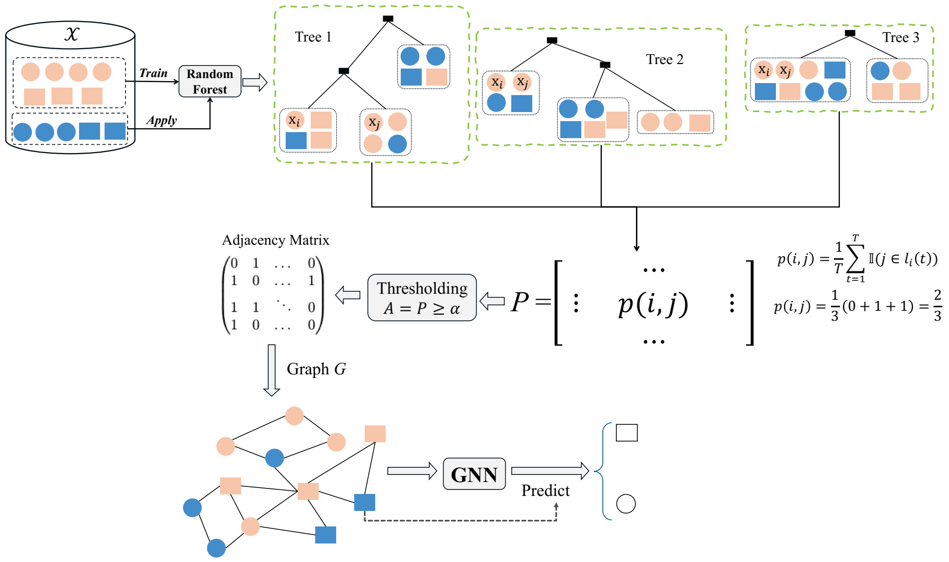
$$p(i, j) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}(j \in l_i(t)), \tag{3}$$

where T represents the number of trees, $l_i(t)$ are the indices of observations in the same leaf as x_i in tree t, and $\mathbb{1}(\cdot)$ is the indicator function. This equation determines the proportion of trees where observations i and j share the same terminal node. For instance, in Figure 2, $p(i, j) = \frac{2}{3}$, indicating x_i and x_j appear in the same leaf nodes in two trees out of the three decision trees (Tree 2 and Tree 3).

Using Equation 3, we calculate the proximity between all pairs of points, resulting in a proximity matrix $P \in \mathbb{R}^{N \times N}$ where $P_{i,j} = p(i, j)$. Since using P directly as the adjacency matrix results in a highly dense (or fully connected) graph, which would negatively impact the performance of GNNs and increase both storage and computational complexity, we select a threshold α to transform the matrix P into a binary adjacency matrix $A \in \mathbb{R}^{\{0,1\}}$:

$$A = P \geq \alpha. \tag{4}$$

Thus, A can serve as the adjacency matrix for an unweighted graph with nodes as data points x_i and edges representing that the random forest proximity exceeds the threshold α. In Section VI-D, we analyze the effect and distribution of this threshold.



$\mathcal{X}$
Train
Apply
Random Forest
Tree 1
Tree 2
Tree 3
Adjacency Matrix
Thresholding $A = P \geq \alpha$
$P =$
$p(i,j)$
$p(i,j) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(j \in l_i(t))$
$p(i,j) = \frac{1}{3}(0+1+1) = \frac{2}{3}$
Graph G
GNN
Predict

Fig. 2: The workflow of our proposed method (RF-GNN), which uses a random forest to learn a graph from tabular data. The graph structure is then inputted to a GNN for final prediction.

B. Graph Neural Network

By transforming tabular data into a graph format, we redefine our initial classification problem as a node classification task. GNNs have proven highly effective in this domain by exploiting both node features and their interconnections within the graph [37]–[39]. In particular, GCNs [37] harness the graph's topology to aggregate and refine features from adjacent nodes. This approach allows nodes to integrate information from their immediate and extended neighborhood, adapting the principles of convolutional neural networks to graph data [40]. As a result, GCNs are proficient at handling various graph-related tasks, including node classification, graph classification, and link prediction.

In the GCN framework, multiple graph convolutional layers are used for neighborhood aggregation. Initially, at the 0-th layer, a node's embedding vector, or latent representation, is set to its feature vector. For instance, for node i corresponding to data point x_i, we define $h_i^{(0)} = x_i$, where $h_i^{(0)}$ represents the embedding of node i at the 0-th layer. In subsequent layers l, within a GCN consisting of L layers, the embeddings of nodes are updated through a weighted average of the embeddings of their neighbors, combined with the node's own previous layer's embedding, followed by the application of a non-linear activation function. This process is mathematically expressed as:

$$h_i^{(l+1)} = \sigma\left(W_l \sum_{j \in N(i)} \frac{h_j^{(l)}}{|N(i)|} + B_l h_i^{(l)} \right), \forall l \in \{0, \dots, L-1\} \tag{5}$$

where σ denotes a non-linear function such as the Rectified Linear Unit (ReLU), and W_l and B_l are learnable parameters. W_l is the weight matrix at layer l applied to the neighbors' embeddings, B_l is the weight matrix applied to the node's own embedding from the previous layer, $N(i)$ represents the set of neighbors of the node i, and $|N(i)|$ indicates the count of these neighbors. After processing through L layers, the final embedding of node i is a k-dimensional vector given by $h_i^{(L)}$.

After applying the GCN, the final embedding of each node or data point is used to predict its label. For this purpose, we use a two-layer Multilayer Perceptron (MLP), known as a fully connected network as well. This MLP takes the final node embedding as input and predicts the probabilities of the node belonging to each class. The mathematical formulation of the MLP operations is given by:

$$a = \mathrm{ReLU}(W_{\text{in}} \cdot h_i^{(L)} + b_a), \tag{6}$$

$$\hat{y}_i = W_{\text{out}} \cdot a + b_o. \tag{7}$$

In Equation 6, W_{in} and b_a are learnable parameters. W_{in} is the weight matrix for the input layer to the hidden layer, b_a is the bias for the hidden layer, and a is the output of the hidden layer. In Equation 7, W_{out} and b_o are learned by the model. W_{out} is the weight matrix connecting the hidden layer to the output layer, b_o is the bias for the output layer, and $\hat{y}_i$ is a c-dimensional vector representing the probability of data point x_i belonging to each class. Finally, a softmax function is applied to $\hat{y}_i$ to determine the final label. We used a cross-entropy loss function to train the model for classification.

VI. Experiments

A. Datasets

We employ datasets available through the OpenML[2] API in our experiments. Most of our datasets belong to OpenML-CC18 benchmark [41], which comprises real-world classification datasets specifically curated for benchmarking machine learning algorithms and is widely recognized and commonly used for tabular data classification. For our analysis, we select

[2]https://www.openml.org/

TABLE I: Dataset Statistics: $\mathbf{N}$ (# of instances), $\mathbf{d}$ (# of features), $\mathbf{d_{cat}}$ (# of categorical features), $\mathbf{c}$ (# of classes).

ID	Name	N	d	$\mathbf{d_{cat}}$	c
902	Sleuth Case 2002	147	6	4	2
1006	Lymphography	148	18	15	2
955	Teaching Assistant Evaluation	151	5	2	2
941	Low Birth Weight	189	9	7	2
1012	Nations' Flags	194	28	26	2
446	Cushing's Syndrome	200	7	1	2
40710	Heart Disease	303	13	8	2
915	Plasma Retinol	315	13	3	2
1167	"pc1_req"	320	8	1	2
40663	calendarDOW	399	32	20	5
475	German Political System	400	5	4	4
1498	South Africa Heart Disease	462	9	1	2
853	Boston Housing Data	506	13	1	2
825	Corrected Boston Housing Data	506	20	3	2
43757	Wisconsin Breast Cancer	569	30	0	2
40981	Australian Credit Approval	690	14	8	2
43942	Annealing	898	38	32	2
40705	Tokyo SGI Server Performance	959	44	2	2
31	German Credit 1	1000	20	13	2
44098	German Credit 2	1000	20	13	2
43255	Student Performance	1000	7	4	2
983	Contraceptive Method Choice	1473	9	7	2
23	Contraceptive Method Choice (Multi)	1473	9	7	3
720	Abalone Age Prediction	4177	8	1	2
1557	Abalone Age Prediction (Multi)	4177	8	1	3
40701	Churn	5000	20	4	2
182	Landsat Satellite	6430	36	0	6
300	Isolated Letter Speech Recognition	7797	617	0	26
1478	Human Activity Recognition	10299	561	0	6
1053	Software Defect Prediction	10885	21	0	2
32	Pen-Based Recog of Handwritten Digits	10992	16	0	10
4534	Phishing Websites	11055	30	30	2
6	Letter Image Recognition	20000	16	0	26
1486	Nomao (Search engine of places)	34465	118	29	2
1461	Portuguese Bank Marketing	45211	16	9	2
1590	Adult (Census Income)	48842	14	8	2

36 datasets, each featuring a mix of numerical and categorical attributes. These datasets are diverse in terms of the number of instances, number of features, number of classes, and their data sources. Table I details the statistics of these datasets. We partitioned the datasets into training and test sets following an 80:20 split ratio.

B. Baselines

We compare our model with the following baseline models:

1) Random Forest (RF): Random Forestis an ensemble learning method that builds multiple decision trees and averages their outcomes to improve accuracy and control overfitting. It is effective for both classification and regression tasks. For our experiments, we employed the Random Forest implementation provided by scikit-learn[3].

2) XGBoost (XGB): XGBoost is a highly efficient and flexible gradient boosting library that enhances the speed and performance of gradient-boosted trees. It is designed for scalability and handles large-scale data effectively. We employed the XGBoost Python package[4] in our experiments.

[3]https://scikit-learn.org/stable/
[4]https://xgboost.readthedocs.io/en/stable/python/

3) LightGBM (LGBM): LightGBM is a fast, distributed gradient boosting framework that uses tree-based learning algorithms. It is optimized for speed and memory efficiency, using advanced techniques like histogram-based splits. In our experiments, we used the LightGBM python package[5].

4) Gradient Boosting (GB): Gradient Boostingconstructs a predictive model through a sequential ensemble of weak models, typically decision trees. It optimizes arbitrary differentiable loss functions, making it versatile for various regression and classification problems. For our experiments, we used the Gradient Boosting implementation available in scikit-learn[6].

5) Multi-layer Perceptron (MLP): MLP is a type of feed-forward artificial neural network with multiple layers, using backpropagation for training. It excels in complex pattern recognition by modeling non-linear relationships between inputs and outputs. For our experiments, we implemented an MLP with 2 hidden layers using the PyTorch library[7].

6) Interaction Network Contextual Embedding (INCE): In addition to the above ML algorithms, we compare RF-GNN with the latest relevant method from the literature. In INCE [42], both categorical and continuous features are projected individually into a common dense latent space. The resulting feature embeddings are then structured into a fully connected graph, augmented with an additional virtual node. Subsequently, a GNN is employed to model the relationships among all nodes, including the original features and the virtual node, thereby enhancing their representations. The enhanced representation of the virtual node is then input into the final classifier or regressor. For this model, we used the authors' implementation available on GitHub[8].

For each model, we employed grid search to find the best hyper-parameters. We repeated each experiment 5 times with different random seeds and recorded the mean and standard deviation of weighted F1-scores. The experiments were conducted on a system with an AMD EPYC 7513 CPU, 4 NVIDIA RTX A4000 GPUs, and 1 TB of RAM. In RF-GNN, we used PyTorch to implement the GCN component and scikit-learn for the random forest. Since RF-GNN requires a proximity threshold, for each dataset, we experimented with 51 evenly spaced proximity thresholds α ranging from 0 to 1 (inclusive). We then picked the best proximity threshold and model hyper-parameters using 5-fold cross-validation on the training set and applied the best model on the test set.

Table II shows the results of our experiments. Based on these results, we make the following observations:

- **RF-GNN** demonstrates outstanding performance, securing the top rank on 16 out of 36 datasets and the second rank on another 10, resulting in the best overall average rank of 2.22. This underscores the efficacy of integrating Random Forest proximity with GNNs, which excellently captures relationships within tabular data, thereby enhancing model performance.

[5]https://lightgbm.readthedocs.io/en/latest/Python-Intro.html
[6]https://scikit-learn.org/stable/
[7]https://pytorch.org/
[8]https://github.com/MatteoSalvatori/INCE

TABLE II: Comparison of the performance of baseline models and our proposed model (RF-GNN) in terms of weighted F1-score. The **best** result in each column is displayed in bold, and the second-best result is underlined.

Dataset	RF	XGB	GB	LGBM	MLP	INCE [42]	RF-GNN
902	0.8269 ± 0.0176	0.7819 ± 0.0153	0.7451 ± 0.0323	0.8125 ± 0.0000	0.8001 ± 0.0211	0.8332 ± 0.0236	$\mathbf{0.8446 \pm 0.0048}$
1006	0.9044 ± 0.0238	0.8707 ± 0.0251	0.8745 ± 0.0531	0.8757 ± 0.0358	0.8734 ± 0.0134	0.8930 ± 0.0154	$\mathbf{0.9472 \pm 0.0208}$
955	0.7874 ± 0.0253	0.7767 ± 0.0254	0.7324 ± 0.0414	0.6056 ± 0.0212	0.5901 ± 0.0607	0.8217 ± 0.0569	$\mathbf{0.8571 \pm 0.0000}$
941	0.7641 ± 0.0314	0.6925 ± 0.0211	0.7560 ± 0.0199	0.7091 ± 0.0264	0.5013 ± 0.0745	$\mathbf{0.8121 \pm 0.0295}$	0.7836 ± 0.0118
1012	0.4211 ± 0.0000	0.4071 ± 0.0248	0.4118 ± 0.0354	0.4261 ± 0.0712	0.4299 ± 0.0243	$\mathbf{0.6861 \pm 0.0193}$	0.6424 ± 0.0121
446	0.8793 ± 0.0118	0.8949 ± 0.0120	0.9052 ± 0.0129	0.9009 ± 0.0147	0.7150 ± 0.1617	$\mathbf{1.0000 \pm 0.0000}$	$\mathbf{1.0000 \pm 0.0000}$
40710	0.7961 ± 0.0088	0.7414 ± 0.0225	0.7648 ± 0.0156	0.7550 ± 0.0103	0.6450 ± 0.0359	$\mathbf{0.8723 \pm 0.0269}$	0.8321 ± 0.0132
915	0.5563 ± 0.0300	0.5856 ± 0.0224	0.5889 ± 0.0139	0.5788 ± 0.0370	0.4474 ± 0.0268	0.5716 ± 0.0182	$\mathbf{0.7293 \pm 0.0025}$
1167	0.3389 ± 0.0669	0.3226 ± 0.0000	0.3460 ± 0.0574	0.2798 ± 0.0267	$\mathbf{0.6352 \pm 0.0237}$	0.6294 ± 0.0817	0.4819 ± 0.0132
40663	$\mathbf{0.6755 \pm 0.0070}$	0.5904 ± 0.0115	0.6730 ± 0.0066	0.6124 ± 0.0060	0.4985 ± 0.0154	0.6335 ± 0.0319	0.6529 ± 0.0100
475	0.3539 ± 0.0169	0.4029 ± 0.0157	0.4372 ± 0.0254	0.3710 ± 0.0268	0.3214 ± 0.0277	0.4140 ± 0.1775	$\mathbf{0.4395 \pm 0.0181}$
1498	0.5314 ± 0.0296	0.5728 ± 0.0191	0.5589 ± 0.0301	0.5733 ± 0.0245	0.6507 ± 0.0328	$\mathbf{0.7525 \pm 0.0146}$	0.6383 ± 0.0282
825	0.8591 ± 0.0059	0.8307 ± 0.0050	0.8413 ± 0.0126	0.8408 ± 0.0081	0.7374 ± 0.0251	0.8682 ± 0.0175	$\mathbf{0.9015 \pm 0.0081}$
853	0.9091 ± 0.0121	0.8986 ± 0.0039	0.9049 ± 0.0147	0.9050 ± 0.0035	0.7496 ± 0.0152	0.8774 ± 0.0182	$\mathbf{0.9152 \pm 0.0095}$
43757	0.9722 ± 0.0000	0.9735 ± 0.0052	0.9630 ± 0.0055	0.9682 ± 0.0056	0.9370 ± 0.0034	$\mathbf{0.9789 \pm 0.0100}$	0.9767 ± 0.0033
40981	0.9163 ± 0.0058	0.9109 ± 0.0099	0.9025 ± 0.0065	0.9103 ± 0.0054	0.8005 ± 0.0225	0.9304 ± 0.0067	$\mathbf{0.9323 \pm 0.0091}$
43942	$\mathbf{1.0000 \pm 0.0000}$	$\mathbf{1.0000 \pm 0.0000}$	0.9969 ± 0.0041	$\mathbf{1.0000 \pm 0.0000}$	$\mathbf{1.0000 \pm 0.0000}$	$\mathbf{1.0000 \pm 0.0000}$	$\mathbf{1.0000 \pm 0.0000}$
40705	0.9454 ± 0.0025	0.9408 ± 0.0042	0.9452 ± 0.0067	0.9455 ± 0.0039	0.7690 ± 0.0114	0.9404 ± 0.0047	$\mathbf{0.9548 \pm 0.0015}$
31	$\mathbf{0.8318 \pm 0.0128}$	0.8309 ± 0.0077	0.8264 ± 0.0096	0.8283 ± 0.0100	0.5765 ± 0.0000	0.7557 ± 0.0215	0.8270 ± 0.0032
43255	$\mathbf{0.8938 \pm 0.0044}$	0.8743 ± 0.0050	0.8850 ± 0.0083	0.8827 ± 0.0043	0.8633 ± 0.0086	0.7912 ± 0.2436	0.8858 ± 0.0066
44098	$\mathbf{0.8318 \pm 0.0128}$	0.8309 ± 0.0077	0.8264 ± 0.0096	0.8283 ± 0.0100	0.5778 ± 0.0027	0.7193 ± 0.0802	0.8287 ± 0.0031
983	0.6409 ± 0.0111	0.6241 ± 0.0041	0.6412 ± 0.0183	0.6313 ± 0.0098	0.6740 ± 0.0211	$\mathbf{0.7288 \pm 0.0078}$	0.6645 ± 0.0087
23	0.5256 ± 0.0113	0.5299 ± 0.0128	0.5209 ± 0.0101	0.5173 ± 0.0073	0.4842 ± 0.0119	0.4841 ± 0.1277	$\mathbf{0.5463 \pm 0.0039}$
720	0.7734 ± 0.0024	0.7735 ± 0.0022	0.7716 ± 0.0044	0.7689 ± 0.0041	0.7277 ± 0.0018	0.7879 ± 0.0058	$\mathbf{0.7965 \pm 0.0019}$
1557	0.6583 ± 0.0041	0.6562 ± 0.0036	0.6524 ± 0.0095	0.6529 ± 0.0085	0.5613 ± 0.0129	$\mathbf{0.6716 \pm 0.0040}$	0.6683 ± 0.0021
40701	0.8700 ± 0.0055	0.8710 ± 0.0042	0.8657 ± 0.0061	0.8831 ± 0.0049	0.7976 ± 0.0044	0.8849 ± 0.0529	$\mathbf{0.8853 \pm 0.0022}$
182	0.9184 ± 0.0020	0.9191 ± 0.0028	$\mathbf{0.9245 \pm 0.0030}$	0.9171 ± 0.0000	0.8285 ± 0.0027	0.8728 ± 0.0082	0.9171 ± 0.0029
300	0.9383 ± 0.0009	0.9517 ± 0.0016	0.9482 ± 0.0004	0.9622 ± 0.0000	0.9387 ± 0.0022	0.6978 ± 0.0461	$\mathbf{0.9636 \pm 0.0018}$
1478	0.9758 ± 0.0026	0.9908 ± 0.0000	0.9903 ± 0.0000	$\mathbf{0.9919 \pm 0.0005}$	0.9440 ± 0.0009	0.9237 ± 0.0738	0.9822 ± 0.0009
1053	0.3291 ± 0.0087	0.2918 ± 0.0000	0.2624 ± 0.0031	0.2423 ± 0.0024	0.7381 ± 0.0070	$\mathbf{0.7704 \pm 0.0017}$	0.7579 ± 0.0034
32	0.9904 ± 0.0005	0.9893 ± 0.0000	0.9918 ± 0.0000	0.9900 ± 0.0003	0.9497 ± 0.0079	0.9801 ± 0.0059	$\mathbf{0.9934 \pm 0.0002}$
4534	$\mathbf{0.9758 \pm 0.0006}$	0.9691 ± 0.0000	0.9695 ± 0.0009	0.9685 ± 0.0010	0.9138 ± 0.0032	0.9687 ± 0.0004	0.9709 ± 0.0011
6	$\mathbf{0.9664 \pm 0.0003}$	0.9636 ± 0.0000	0.9655 ± 0.0001	0.9646 ± 0.0018	0.6900 ± 0.0062	0.7754 ± 0.0409	0.9388 ± 0.0036
1486	0.9793 ± 0.0004	$\mathbf{0.9801 \pm 0.0000}$	0.9783 ± 0.0002	0.9770 ± 0.0006	0.9334 ± 0.0013	0.9545 ± 0.0036	0.9658 ± 0.0005
1461	0.5105 ± 0.0032	0.5586 ± 0.0000	0.5443 ± 0.0000	0.5438 ± 0.0057	0.8471 ± 0.0101	$\mathbf{0.9099 \pm 0.0012}$	0.8916 ± 0.0017
1590	0.6833 ± 0.0015	0.7143 ± 0.0000	0.7185 ± 0.0000	0.7134 ± 0.0017	0.7306 ± 0.0082	$\mathbf{0.8184 \pm 0.0901}$	0.7636 ± 0.0119
# 1st rank	7	2	1	2	2	12	**16**
# 2nd rank	3	5	5	2	2	7	**10**
Avg rank	3.55	4.28	4.22	4.42	5.80	3.50	**2.22**

- **INCE** is the second-best model with an average rank of 3.50, leading in performance on 12 datasets and achieving second place on 7 datasets. Its robust performance highlights the benefits of graph embedding techniques for tabular data classification.
- **Random Forest** emerges as a strong contender, holding a close third position with an average rank of 3.55. It consistently performs well, emphasizing its reliability as a robust method for diverse datasets.
- The superior performance of **RF-GNN** and **INCE** across the majority of the datasets (26 out of 36) illustrates their particular effectiveness in leveraging graph-based approaches to improve predictions in tabular datasets.
- While **RF-GNN** generally outperforms **INCE**, it shows notably higher scores on datasets such as 43255 and 6, indicating significant improvements. Conversely, **INCE** demonstrates better results than **RF-GNN** on datasets such as 1498 and 983, suggesting that the choice between these models may depend on specific dataset characteristics or the nature of the data relationships captured by each model.

C. Proximity Measure Analysis

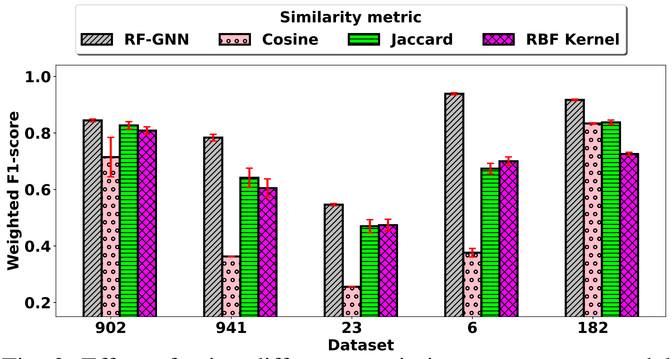


Fig. 3: Effect of using different proximity measures on model performance on 5 different datasets in terms of weighted F1-score. The RF proximity gives the best performance.

In this section, we assess the effectiveness of the proposed RF proximity approach, which is the core of our proposed

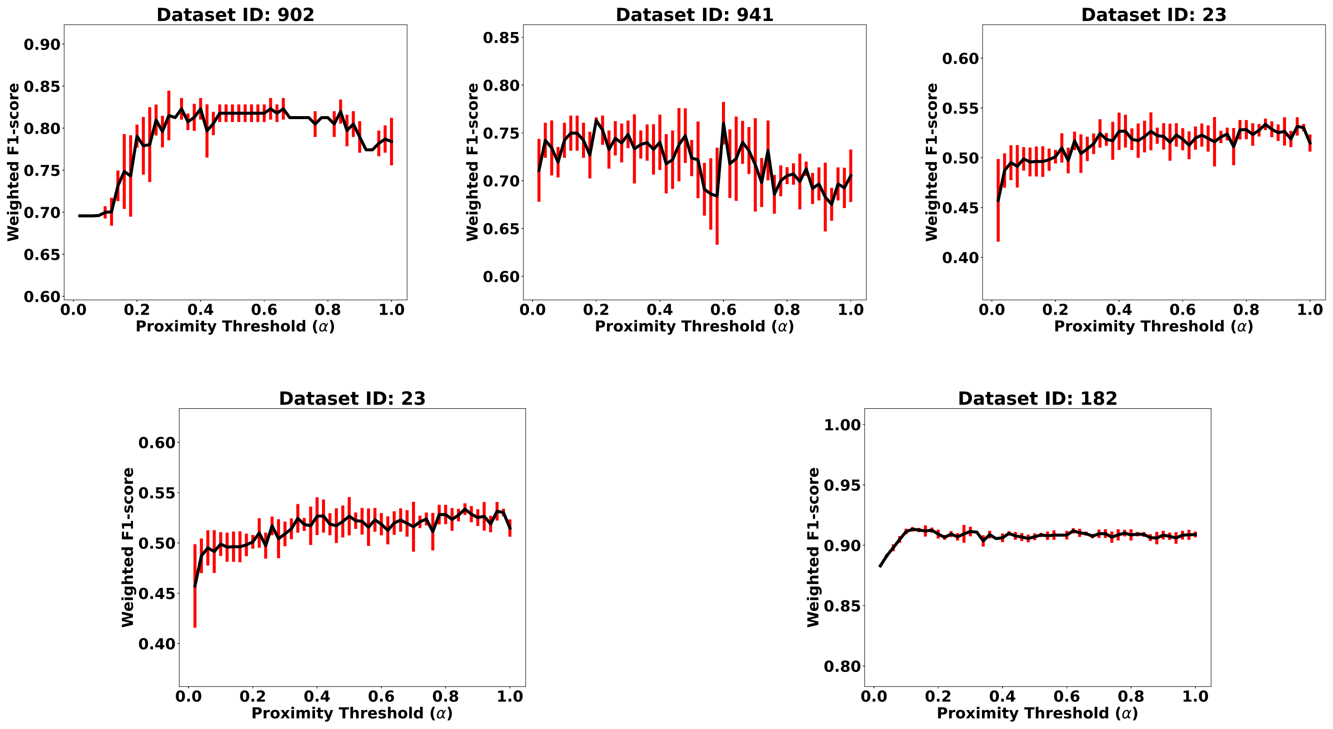


Fig. 4: Sensitivity analysis of GNN performance across varying proximity thresholds α for five datasets. The results show general stability in performance for α values between 0.4 and 0.6, with notable variability in dataset 941, indicating differing sensitivity to threshold settings.

method. As an alternative to the RF proximity, one might be tempted to use readily available similarity measures. To evaluate the efficacy of the RF proximity compared to such measures, we opted to use cosine, Jaccard, and RFB kernel measures where for each pair of samples x_i and x_j, they will yield a similarity scalar between 0 and 1 based on the features of these two samples. The following are mathematical formulations of these similarity measures:

$$Cosine(x_i, x_j) = \frac{x_i \cdot x_j}{||x_i|| \times ||x_j||} \tag{8}$$

where $\cdot$ is the inner product operation, and $||\cdot||$ is the Euclidean norm operation.

$$Jaccard(x_i, x_j) = \frac{\sum_{k=1}^{d} \min(x_{i,k}, x_{j,k})}{\sum_{k=1}^{d} \max(x_{i,k}, x_{j,k})} \tag{9}$$

where $x_{i,k}$ is the k-th element in the d-dimensional vector x_i.

$$RBF(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{10}$$

where $||\cdot||$ is the Euclidean norm operation and γ is a hyper-parameter that determines the impact of each training example on the decision boundary. We set the value of γ to 0.01 in our experiments. We also standardized the RFB value to be in the range $[0, 1]$.

We compare the weighted F1-score of RF-GNN with GNNs trained on graphs constructed using the aforementioned similarity measures across 5 different datasets[9]. We employ the same training scheme, constructing proximity matrices for the dataset via all these similarity measures and then applying a threshold to construct graphs for GNNs. The results are shown in Figure 3. Each experiment was executed 5 times with different random seeds, and the mean and standard deviation of weighted F1-scores are reported. We make the following observations.

- Our experimental results in Figure 3 show that RF-GNN consistently outperforms GNNs trained on graphs constructed from other proximity measures.
- Unlike traditional proximity measures that solely focus on the feature space, RF proximity considers both features and labels, providing a more comprehensive understanding of sample similarity and potentially enhancing performance in tasks where label information is informative or critical.
- The RF proximity, trained within the context of a random forest model, inherently prioritizes the most informative features during the learning process.
- The RF proximity tends to be less sensitive to outliers compared to other similarity measurements due to the ensemble nature of the Random Forest algorithm and the way proximities are calculated within it.

[9]We obtained similar results for other datasets.

D. Threshold Sensitivity Analysis

We conducted a sensitivity analysis to determine the impact of various proximity thresholds α on the performance of the GNN. This analysis was performed across five of the datasets in Table I, using 51 evenly spaced thresholds ranging from 0 to 1. For each threshold, experiments were repeated five times, and both the mean and standard deviation of the weighted F1-scores were recorded. The results are illustrated in Figure 4.

The analysis indicates that, apart from some minor variations between seeds, the performance for datasets generally stabilizes for α values between 0.4 and 0.6. However, dataset 941 shows fluctuating performance at certain thresholds, suggesting variable sensitivity to the proximity threshold setting.

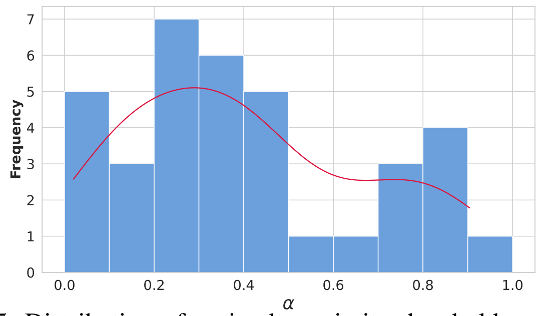


Fig. 5: Distribution of optimal proximity thresholds α across 36 datasets, highlighting a concentration between 0.1 and 0.5. This range suggests that moderate graph density is optimal for GNN performance.

Additionally, we examined the distribution of the optimal proximity thresholds across all 36 datasets by averaging the best-performing thresholds from each of our 5 experiments. The resulting histogram and Kernel Density Estimate (KDE) plot, shown in Figure 5, demonstrate that optimal α values typically range from 0.1 to 0.5. This pattern indicates that moderately dense graph structures, which strike a balance between connectivity and noise, tend to yield the best performance in GNNs.

E. Dataset Attribute Analysis

In this section, we investigate whether dataset attributes affect the performance of RF-GNN and other methods. Specifically, we examine the relationship between the performance of a predictive model and several dataset attributes. We consider the performance of a method as the *dependent variable* and the number of instances, the number of features, the number of classes, and the number of categorical features as *independent variables*. Subsequently, we perform a regression analysis. Table III summarizes the results for all methods. Based on these results, we make the following observations.

- **Negative R-squared Values:** All models resulted in negative R-squared values, indicating that the linear regression models do not fit the data well. This suggests that the dataset attributes do not adequately explain the variability in a model's performance.
- **Mean Squared Error (MSE):** The MSE values are relatively low, but given the negative R-squared values, this does not imply good regression performance. The low MSE might be misleading without considering the R-squared values.
- **Consistent Intercept Across Models:** The intercept (const) is consistently positive and significant across all models, with coefficients ranging from approximately 0.6672 to 0.7914. This indicates that even when all independent variables are zero, the models predict positive values for the dependent variables.
- **#Instances Variable:** The coefficient for #Instances is small and not statistically significant in any of the models, with p-values well above the 0.05 threshold. This suggests that the number of instances does not have a meaningful impact on the performance of the models.
- **#Features Variable:** The #Features variable has positive coefficients in all models but is not statistically significant, except for MLP (with a borderline p-value of 0.055). This indicates that the number of features may have a slight positive influence on model performance, particularly for MLP, but the evidence is not strong.
- **#Classes Variable:** The coefficient for #Classes is generally positive but not significant across all models. This implies that the number of classes does not significantly affect the performance of the models.
- **#Categorical Features Variable:** The #Categorical Features variable also has positive coefficients across all models but is not statistically significant. This suggests that the presence of categorical features does not significantly influence model performance.
- **Significance Levels:** None of the independent variables are statistically significant at the 0.05 level in any of the models. This indicates that none of these features strongly explain the variance in the dependent variables for this particular dataset.
- **Potential Multicollinearity:** The lack of significance for all independent variables across all models could indicate potential multicollinearity issues or that these features are not the key drivers of model performance. Further investigation into feature interactions or alternative features might be necessary.

Overall, the regression analysis indicates that dataset attributes have an insignificant impact on the classification performance of an ML model. Specifically, for RF-GNN, this suggests that our proposed method can consistently perform well across various datasets, regardless of dataset size, number of features, number of classes, and number of categorical features. This further supports the versatility of RF-GNN.

VII. Conclusion and Future Work

This study presented a novel method, RF-GNN, which successfully enhances the performance of traditional machine learning algorithms by converting tabular data into graph structures using Random Forest proximities and then applying Graph Neural Networks. RF-GNN proved to be effective in

TABLE III: Regression analysis of the relationship between the dataset attributes and the performance of a method (weighted F1-score). Overall, the regression analysis suggests that dataset attributes have an insignificant impact.

Method	Dataset Attribute (Variable)	Coefficient	Std. Error	t-value	$P > \|t\|$	95% Confidence Interval
RF	const	0.7031	0.0543	12.9472	0.000	[0.5923, 0.8139]
	#Instances	-0.000001	0.000003	-0.5061	0.616	[-0.000007, 0.000004]
	#Features	0.000276	0.000303	0.9118	0.369	[-0.000342, 0.000894]
	#Classes	0.007963	0.007423	1.0727	0.292	[-0.007176, 0.023102]
	#Categorical Features	0.004202	0.003895	1.0789	0.289	[-0.003742, 0.012146]
	Mean Squared Error (MSE): 0.0325			R-squared (R^2): -0.944		
XGB	const	0.6966	0.0539	12.9301	0.000	[0.5867, 0.8064]
	#Instances	-0.000000	0.000003	-0.1631	0.872	[-0.000006, 0.000005]
	#Features	0.000309	0.000301	1.0281	0.312	[-0.000304, 0.000922]
	#Classes	0.007435	0.007364	1.0098	0.320	[-0.007583, 0.022454]
	#Categorical Features	0.003443	0.003864	0.8910	0.380	[-0.004438, 0.011323]
	Mean Squared Error (MSE): 0.0357			R-squared (R^2): -1.189		
GB	const	0.6966	0.0530	13.1511	0.000	[0.5886, 0.8047]
	#Instances	-0.000001	0.000003	-0.2979	0.768	[-0.000007, 0.000005]
	#Features	0.000294	0.000296	0.9944	0.328	[-0.000309, 0.000897]
	#Classes	0.008033	0.007241	1.1094	0.276	[-0.006735, 0.022800]
	#Categorical Features	0.003991	0.003799	1.0506	0.302	[-0.003757, 0.011740]
	Mean Squared Error (MSE): 0.0334			R-squared (R^2): -1.207		
LGBM	const	0.6831	0.0561	12.1681	0.000	[0.5686, 0.7976]
	#Instances	-0.000001	0.000003	-0.2151	0.831	[-0.000007, 0.000005]
	#Features	0.000325	0.000313	1.0386	0.307	[-0.000313, 0.000964]
	#Classes	0.008159	0.007673	1.0633	0.296	[-0.007491, 0.023809]
	#Categorical Features	0.004250	0.004026	1.0555	0.299	[-0.003962, 0.012462]
	Mean Squared Error (MSE): 0.0334			R-squared (R^2): -1.207		
MLP	const	0.6672	0.0433	15.4256	0.000	[0.5790, 0.7554]
	#Instances	0.000004	0.000002	1.6066	0.118	[-0.000001, 0.000008]
	#Features	0.000482	0.000241	1.9961	0.055	[-0.000010, 0.000974]
	#Classes	-0.001551	0.005912	-0.2624	0.795	[-0.013609, 0.010506]
	#Categorical Features	0.001211	0.003102	0.3904	0.699	[-0.005116, 0.007538]
	Mean Squared Error (MSE): 0.0565			R-squared (R^2): -3.559		
INCE	const	0.7914	0.0408	19.3826	0.000	[0.7121, 0.8706]
	#Instances	0.000002	0.000002	0.7532	0.457	[-0.000002, 0.000007]
	#Features	0.000147	0.000215	0.6837	0.498	[-0.000295, 0.000589]
	#Classes	-0.004842	0.005594	-0.8657	0.392	[-0.015676, 0.005992]
	#Categorical Features	0.001261	0.002787	0.4523	0.654	[-0.004424, 0.006946]
	Mean Squared Error (MSE): 0.0283			R-squared (R^2): -2.879		
RF-GNN	const	0.7797	0.0405	19.2645	0.000	[0.6971, 0.8622]
	#Instances	0.000002	0.000002	0.7357	0.467	[-0.000003, 0.000006]
	#Features	0.000256	0.000226	1.1338	0.266	[-0.000204, 0.000717]
	#Classes	0.002758	0.005532	0.4986	0.622	[-0.008525, 0.014041]
	#Categorical Features	0.001771	0.002903	0.6102	0.546	[-0.004149, 0.007691]
	Mean Squared Error (MSE): 0.0269			R-squared (R^2): -1.207		

capturing complex relational information within tabular data, thus overcoming the limitations of conventional techniques that typically struggle with such tasks. Through extensive experimentation across 36 diverse datasets, RF-GNN demonstrated significant improvements over several strong baselines, including Random Forests, XGBoost, LightGBM, Gradient Boosting, Multi-layer Perceptron, and Interaction Network Contextual Embedding. Our results showed that RF-GNN achieved the best performance on 16 out of the 36 datasets and was among the top-performing models on most others, highlighting its robustness and versatility.

Additionally, the analysis of different proximity measures reaffirmed the superiority of Random Forest proximity in constructing effective graphs for GNNs. The sensitivity analysis of the proximity threshold further emphasized the importance of selecting an optimal threshold to maximize performance, while the dataset attribute analysis indicated that RF-GNN consistently provided strong results across various dataset characteristics.

Overall, RF-GNN offers a general, straightforward approach for transforming tabular data into graph structures, enabling the application of advanced GNN techniques. This method has the potential to significantly enhance the predictive performance of machine learning models on tabular data, making it a valuable contribution to the field of data science and machine learning.

Possible future directions include exploring alternative proximity measures to better capture relationships in tabular data, integrating more sophisticated or hybrid similarity metrics, and developing automated methods for optimizing proximity thresholds across different datasets. Additionally, it is worth studying the impact of dynamic thresholding strategies during training, improving the scalability of RF-GNN for very large datasets, and implementing parallel processing techniques for efficiency. Experimenting with other GNN architectures like Graph Attention Networks (GAT) [38] or GraphSAGE [39] could provide further performance improvements, as could combining RF-GNN with other neural network models like transformers. Extending the RF-GNN method to handle temporal and sequential data, and testing robustness to noisy and incomplete data using noise-resistant methods and data augmentation techniques are also important directions.

References

[1] X. Cheng, C. Yang, Y. Zhao, Y. Wang, H. Karimi, and T. Derr, "A comprehensive analysis of social tie strength: Definitions, prediction methods, and future directions," *arXiv preprint arXiv:2410.19214*, 2024.

[2] K. Kheiri, M. F. A. Khan, T. Derr, and H. Karimi, "An analysis of the dynamics of ties on twitter," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 5809–5817.

[3] H. Karimi, K. T. Knake, and K. A. Frank, "An analysis of diffusion of teacher-curated resources on pinterest." *International Educational Data Mining Society*, 2023.

[4] D. Moore, J. Edwards, H. Karimi, R. Khadka, and P. Bodily, "Temporal abstract syntax trees for understanding student coding thought process," in *2022 Intermountain Engineering, Technology and Computing (IETC)*. IEEE, 2022, pp. 1–6.

[5] S. Javadi, A. Moradan, M. Sorkhpar, K. Zaporojets, D. Mottin, and I. Assent, "Wiki entity summarization benchmark," *arXiv preprint arXiv:2406.08435*, 2024.

[6] H. Boomari and S. Farokhi, "Computing boundary cycle of a pseudo-triangle polygon from its visibility graph," in *Topics in Theoretical Computer Science: Third IFIP WG 1.8 International Conference, TTCS 2020, Tehran, Iran, July 1–2, 2020, Proceedings 3*. Springer, 2020, pp. 61–71.

[7] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[8] H. Karimi, T. Derr, A. Brookhouse, and J. Tang, "Multi-factor congressional vote prediction," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 266–273.

[9] S. Farokhi, A. Azizian Foumani, X. Qi, and H. Karimi, *EDGE-UP: Enhanced Dynamic GNN Ensemble for Unfollow Prediction in Online Social Networks*, ser. Lecture Notes in Computer Science, L. M. Aiello, T. Chakraborty, and S. Gaito, Eds. Springer Cham, 2024, vol. 1.

[10] R. Li, X. Yuan, M. Radfar, P. Marendy, W. Ni, T. J. O'Brien, and P. M. Casillas-Espinosa, "Graph signal processing, graph neural network and graph learning on biological data: a systematic review," *IEEE Reviews in Biomedical Engineering*, vol. 16, pp. 109–135, 2021.

[11] S. Farokhi, A. Yaramal, J. Huang, M. F. A. Khan, X. Qi, and H. Karimi, "Enhancing the performance of automated grade prediction in mooc using graph representation learning," in *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2023, pp. 1–10.

[12] A. Yaramala, S. Farokhi, and H. Karimi, "Navigating the data-rich landscape of online learning: Insights and predictions from assistments," in *Proceedings of the 17th International Conference on Educational Data Mining*, 2024, pp. 321–331.

[13] H. Karimi, T. Derr, J. Huang, and J. Tang, "Online academic course performance prediction using relational graph convolutional neural network." *International Educational Data Mining Society*, 2020.

[14] C.-T. Li, Y.-C. Tsai, and J. C. Liao, "Graph neural networks for tabular data learning," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 3589–3592.

[15] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[16] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," 2019.

[17] Y. Li, W. Jin, H. Xu, and J. Tang, "Deeprobust: A pytorch library for adversarial attacks and defenses," *arXiv preprint arXiv:2005.06149*, 2020.

[18] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, , and L. Wang, "Deep graph structure learning for robust representations: A survey." *arXiv preprint arXiv:2103.03036*, 2021.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations*, 2018.

[20] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, and X. Xie, "Graph structure estimation neural networks," pp. 342–353, 2021.

[21] B. Fatemi, L. E. Asri, , and S. M. Kazemi, "Slaps: Self supervision improves structure learning for graph neural networks." 2021.

[22] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," pp. 1392–1403, 2022.

[23] E. Rocheteau, C. Tong, P. Veličković, N. Lane, and P. Liò, "Predicting patient outcomes with graph representation learning," 2021.

[24] C.-T. Li, Y.-C. Tsai, C.-Y. Chen, and J. C. Liao, "Graph neural networks for tabular data learning: A survey with taxonomy and directions," 2024.

[25] S. Kang, "k-nearest neighbor learning with graph neural networks," *Mathematics 2021, 9(8), 830.*

[26] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[27] L. Breiman and A. Cutler, "Random forests," https://www.stat.berkeley.edu/\protect\begingroup\immediate\write\@unused\def\MessageBreak■\let\protect\edefYoumayprovideadefinitionwith\MessageBreak\DeclareUnicodeCharacter\errhelp\let\def\MessageBreak■(inputenc)\def\errmessagePackageinputencError:Unicodecharacter(U+223C)\MessageBreaknotsetupforusewithLaTeX.■■Seetheinputencpackagedocumentationforexplanation.■TypeH⟨return⟩forimmediatehelp\endgroupbreiman/RandomForests/cchome.htm#prox, accessed on 05/16/2024.

[28] H. Pang, A. Lin, M. Holford, B. E. Enerson, B. Lu, M. P. Lawton, E. Floyd, and H. Zhao, "Pathway analysis using random forests classification and regression," *Bioinformatics*, vol. 22, no. 16, pp. 2028–2036, 2006.

[29] E. J. Finehout, Z. Franck, L. H. Choe, N. Relkin, and K. H. Lee, "Cerebrospinal fluid proteomic biomarkers for alzheimer's disease," *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, vol. 61, no. 2, pp. 120–129, 2007.

[30] J. S. Rhodes, A. Cutler, G. Wolf, and K. R. Moon, "Random forest-based diffusion information geometry for supervised visualization and data exploration," in *2021 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2021, pp. 331–335.

[31] J. S. Rhodes, A. Aumon, S. Morin, M. Girard, C. Larochelle, E. Brunet-Ratnasingham, A. Pagliuzza, L. Marchitto, W. Zhang, A. Cutler *et al.*, "Gaining biological insights through supervised data visualization," *bioRxiv*, pp. 2023–11, 2023.

[32] N. Nesa, T. Ghosh, and I. Banerjee, "Outlier detection in sensed data using statistical learning models for iot," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[33] A. Pantanowitz and T. Marwala, "Missing data imputation through the use of the random forest algorithm," in *Advances in computational intelligence*. Springer, 2009, pp. 53–62.

[34] M. Kokla, J. Virtanen, M. Kolehmainen, J. Paananen, and K. Hanhineva, "Random forest-based imputation outperforms other methods for imputing lc-ms metabolomics data: a comparative study," *BMC bioinformatics*, vol. 20, pp. 1–11, 2019.

[35] A. D. Shah, J. W. Bartlett, J. Carpenter, O. Nicholas, and H. Hemingway, "Comparison of random forest and parametric imputation models for imputing missing data using mice: a caliber study," *American journal of epidemiology*, vol. 179, no. 6, pp. 764–774, 2014.

[36] J. S. Rhodes, A. Cutler, and K. R. Moon, "Geometry- and accuracy-preserving random forest proximities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–13, 2023.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[38] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[39] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[40] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[41] B. Bischl, G. Casalicchio, M. Feurer, P. Gijsbers, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren, "Openml benchmarking suites," *arXiv preprint arXiv:1708.03731*, 2017.

[42] M. Villaizán-Vallelado, M. Salvatori, B. Carro Martinez, and A. J. Sanchez Esguevillas, "Graph Neural Network contextual embedding for Deep Learning on Tabular Data," *Neural Networks*, 2024. [Online]. Available: https://arxiv.org/pdf/2303.06455.pdf