ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240697

Interpretable Graph Neural Networks for Tabular Data

Amr Alkhatib [a], Sofiane Ennadir [a], Henrik Boström [a] and Michalis Vazirgiannis [a,b]

[a]KTH Royal Institute of Technology
Electrum 229, 164 40 Kista, Stockholm, Sweden
[b]DaSciM, LIX, École Polytechnique, Institut Polytechnique de Paris, France.

Abstract. Data in tabular format is frequently occurring in real-world applications. Graph Neural Networks (GNNs) have recently been extended to effectively handle such data, allowing feature interactions to be captured through representation learning. However, these approaches essentially produce black-box models, in the form of deep neural networks, precluding users from following the logic behind the model predictions. We propose an approach, called IGNNet (Interpretable Graph Neural Network for tabular data), which constrains the learning algorithm to produce an interpretable model, where the model shows how the predictions are exactly computed from the original input features. A large-scale empirical investigation is presented, showing that IGNNet is performing on par with state-of-the-art machine-learning algorithms that target tabular data, including XGBoost, Random Forests, and TabNet. At the same time, the results show that the explanations obtained from IGNNet are aligned with the true Shapley values of the features without incurring any additional computational overhead.

1 Introduction

In some application domains, e.g., medicine and law, predictions made by machine learning models need justification for legal and ethical considerations [24, 17]. In addition, users may put trust in such models only with a proper understanding of the reasoning behind the predictions. A direct solution is to use learning algorithms that produce interpretable models, such as logistic regression [5], which provides both local (instance-specific) and global (model-level) explanations for the predictions. However, such algorithms often result in a substantial loss in predictive performance compared to algorithms that generate black-box models, e.g., XGBoost [7], Random Forests [6], and deep learning algorithms [29, 32]. Post-hoc explanation techniques, e.g., SHAP [27], LIME [34], and Anchors [35], have been put forward as tools to explain predictions of the black-box models. However, the explanations provided by such methods are often computationally intensive [2, 22] and lack fidelity guarantees, i.e., there are no guarantees that the provided explanation accurately reflects the underlying model, as studied previously [8, 12, 41]. As extensively argued in [37], there are hence several reasons to consider generating interpretable models in the first place, if trustworthiness is a central concern.

Graph Neural Networks (GNNs) have emerged as a powerful framework for representation learning of graph-structured data [39]. The application of GNNs has been extended to tabular data, where a GNN can be used to learn an enhanced representation for the data points (rows) or to model the interaction between different features (columns). TabGNN [19] is an example of the first approach, where each data point is represented as a node in a graph. In comparison, TabularNet [13] and Table2Graph [46] follow the second approach, where the first uses a Graph Convolutional Network to model the relationships between features, and the second learns a probability adjacency matrix for a unified graph that models the interaction between features of the data points. GNNs can also be combined with other algorithms suited for tabular data, e.g., as in BGNN [21], which combines gradient-boosted decision trees and a GNN in one pipeline, where the GNN addresses the graph structure and the gradient-boosted decision trees handle the heterogeneous features of the tabular data. To the best of our knowledge, all previous approaches to using GNNs for tabular data result in black-box models and they are hence associated with the issues discussed above when applied in contexts with strong requirements on trustworthiness. In this work, we propose a novel GNN approach for tabular data, with the aim to eliminate the need to apply post-hoc explanation techniques without sacrificing predictive performance.

The main contributions of this study are:

- a novel approach, called **I**nterpretable **G**raph **N**eural **Net**work for tabular data (IGNNet), that exploits powerful graph neural network models while still being able to show exactly how the prediction is derived from the input features in a transparent way
- a large-scale empirical investigation evaluating the explanations of IGNNet as well as comparing the predictive performance of IGNNet to state-of-the-art approaches for tabular data; XGBoost, Random Forests, and multi-layer perceptron (MLP), as well as to an algorithm generating interpretable models; TabNet [4]
- an ablation study comparing the performance of the proposed approach to a black-box version, i.e., not constraining the learning algorithm to produce transparent models for the predictions

In the next section, we briefly review related work. In Section 3, we describe the proposed interpretable graph neural network. In Section 4, results from a large-scale empirical investigation are presented and discussed, in which the explanations of the proposed method are evaluated and the performance is compared both to interpretable and powerful black-box models. Section 5 discusses the limitations of the proposed approach. Finally, in the concluding remarks section, we summarize the main conclusions and point out directions for future work.

2 Related Work

In this section, we provide pointers to self-explaining (regular and graph) neural networks, and briefly discuss their relation to model-agnostic explanation techniques and interpretable models. We also provide pointers to work on interpretable deep learning approaches for tabular data.

2.1 Self-Explaining Neural Networks

Several approaches to generating so-called self-explaining neural networks have been introduced in the literature; in addition to generating a prediction model, they all incorporate a component for explaining the predictions. They can be seen as model-specific explanation techniques, in contrast to model-agnostic techniques, such as LIME and SHAP, but sharing the same issues regarding fidelity and lack of detail regarding the exact computation of the predictions.

Approaches in this category include the method in [25], which is an early self-explaining neural network for text classification, the Contextual Explanation Network (CEN) [1], which generates explanations using intermediate graphical models, the Self-explaining Neural Network (SENN) [3], which generalizes linear classifiers to neural networks using a concept autoencoder, and the CBM-AUC [38], which improves the efficiency of the former by replacing the decoder with a discriminator. Some approaches generate explanations in the form of counterfactual examples, e.g., CounterNet [18], and Variational Counter Net (VCNet) [20]. Again, such explanations do not provide detailed information on how the original predictions are computed and how exactly the input features affect the outcome.

2.2 Self-Explaining Graph Neural Networks

The Self-Explaining GNN (SE-GNN) [10] uses similarities between nodes to make predictions on the nodes' labels and provide explanations using the most similar K nodes with labels. ProtGNN [45] also computes similarities, but between the input graph and prototypical graph patterns that are learned per class. Cui et al. [9] proposed a framework to build interpretable GNNs for connectome-based brain disorder analysis that resembles the signal correlation between different brain areas. Feng et al. [15] proposed the KerGNN (Kernel Graph Neural Network), which improves model interpretability using graph filters. The graph filters learned in KerGNNs can uncover local graph structures in a dataset. Xuanyuan et al. [40] proposed scrutinizing individual neurons in a GNN to generate global explanations using neuron-level concepts. CLARUS [28] enhances human understanding of GNN predictions in medicine and facilitates the visualization of patient-specific information.

The self-explainable GNN methods mentioned above are not suited for tabular data, as they are designed for input data that are inherently graphical, such as social networks and molecular structures.

2.3 Interpretable Deep Learning for Tabular Data

In an endeavor to provide an interpretable regression model for tabular data while retaining the performance of deep learning models, and inspired by generalized linear models (GLM), LocalGLMnet was proposed to make the regression parameters of a GLM feature dependent, allowing for quantifying variable importance and also conducting variable selection [36]. TabNet [4] is another interpretable method proposed for tabular data learning, which employs a sequential attention mechanism and learnable masks for selecting a subset of meaningful features to reason from at each decision step. The feature selection is instance-based, i.e., it differs from one instance to another. The feature selection masks can be visualized to highlight important features and show how they are combined. However, it is not obvious how the features are actually used to form the predictions.

3 The Proposed Approach: IGNNet

This section describes the proposed method to produce an interpretable model using a graph neural network. We first outline the details of a GNN for graph classification and then show how it can be constrained to produce interpretable models. Afterward, we show how it can be applied to tabular data. Finally, we show how the proposed approach can maintain both interpretability and high performance.

3.1 Interpretable Graph Neural Network

The input to a GNN learning algorithm is a set of graphs denoted by $\mathcal{G} = (V, E, X, \boldsymbol{A})$, consisting of a set of nodes V, a set of edges E, a set of node feature vectors X, and an adjacency matrix $\boldsymbol{A}$, where $V = \{v_1, \dots, v_N\}$, $E \subseteq \{(v_i, v_j)|v_i, v_j \in V\}$, $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and the weight of edge (v_i, v_j) is represented by a scalar value $\delta_{i,j}$ in the weighted adjacency matrix $\boldsymbol{A}$, where $\delta_{i,j} = \boldsymbol{A}(i,j)$. A GNN algorithm learns a representation vector $\mathbf{h}_i$ for each node v_i, which is initialized as $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. The key steps in a GNN for graph classification can be summarized by the following two phases [39]:

(a) **Message Passing:** Each node passes a message to the neighboring nodes, then aggregates the passed information from the neighbors. Finally, the node representation is updated with the aggregated information. A neural network can also be used to learn some message functions between nodes. The message passing phase can be formulated as:

$$\mathbf{h}_i^{(l+1)} = \varphi\left(\mathbf{w}^{(l)}\left(\delta_{i,i}\mathbf{h}_i^{(l)} + \sum_{u \in \mathcal{N}(i)} \delta_{i,u}\mathbf{h}_u^{(l)}\right)\right) \quad (1)$$

Where $\mathbf{h}_i^{(l)}$ is the hidden representation of the node v_i in the l-th layer, $\mathcal{N}(i)$ is the neighborhood of node v_i, $\mathbf{w}^{(l)}$ represents the learnable parameters, and φ is a non-linearity function.
The adjacency matrix $\boldsymbol{A}$ of size $|V| \times |V|$ contains the edge weights and can be normalized similar to a Graph Convolutional Network (GCN) [23] as shown in (2).

$$\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \quad (2)$$

Here $\boldsymbol{D}$ is the degree matrix $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$ [23].

(b) **Graph Pooling (Readout):** A representation of the whole graph $\mathcal{G}$ is learned using a simple or advanced function [39], e.g, sum, mean, or MLP.

The whole graph representation obtained from the **graph pooling** phase can be submitted to a classifier to predict the class of the graph, which can be trained in an end-to-end architecture [44, 42].

The pooling function can be designed to provide an interpretable graph classification layer. Thus, the final hidden representation of each node is mapped to a single value, for instance, through a neural network layer or dot product ($\mathcal{R}(\mathbf{h}_i^{(l+1)} \in \mathbb{R}^n) = h_i \in \mathbb{R}^1$), and concatenated to obtain the final representation $\mathbf{g}$ of the graph $\mathcal{G}$ where a scalar value in $\mathbf{g}$ corresponds to a node in the graph. Consequently, if a set of weights is applied to classify the graph, we can trace the contribution of each node to the predicted outcome, i.e., the user can find out which nodes contributed to the predicted class. For example, **g** can be used directly as follows:

$$\hat{y} = \text{link}\left(\sum_{i=1}^{n} w_i g_i\right) \tag{3}$$

where w_i is the weight (in vector **w**) assigned to node v_i represented in g_i. The link function is applied to accommodate a valid range of outputs, e.g., the sigmoid function for binary and softmax for multi-class classification. This is equivalent to:

$$\hat{y} = \text{link}\left(\sum_{i=1}^{n} w_i \mathcal{R}(\mathbf{h}_i^{(l+1)})\right) \tag{4}$$

In the case of binary classification, one vector of weights (**w**) is applied, and for multiple classes, each class has a separate vector of weights.

3.2 Representing Tabular Data Points as Graphs

The proposed readout function in the previous subsection allows for determining the contribution of each node in a prediction, if a white-box classification layer is used for the latter. Therefore, we propose representing each data instance as a graph where *the features are the nodes* of that graph and *the linear correlation between features are the edge weights*, as we assume that not all features are completely independent. The initial representation of a node is a vector of one dimension, and the value is just the feature value, which can be embedded into a higher dimensionality. The idea is outlined in Algorithm 1 and illustrated in Figure 1.

Algorithm 1: IGNNet

```
Data: a set of graphs $\mathbb{G}$ and labels $\mathbb{Y}$
Result: Model parameters $\theta$
Initialize $\theta$
for number of training iterations do
    $\mathcal{L} \leftarrow 0$
    for each $\mathcal{G}_j \in \mathbb{G}$ do
        $\boldsymbol{H}_j^{(0)} \leftarrow \mathcal{G}_j$
        for each layer $l \in$ messagePassing layers do
            $\boldsymbol{H}_j^{(l+1)} \leftarrow$messagePassing($\boldsymbol{H}_j^{(l)}$)
        end
        $\mathbf{g}_j \leftarrow$readout($\boldsymbol{H}_j^{(l+1)}$)
        $\hat{y}_j \leftarrow$predict($\mathbf{g}_j$)
        $\mathcal{L} \leftarrow \mathcal{L}$+loss($\hat{y}_j, y_j \in \mathbb{Y}$)
    end
    Compute gradients $\nabla_\theta \mathcal{L}$
    Update $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$
end
```

In order to make a prediction for a test instance, the data point has to be converted to a graph using the same procedure for building the input graphs to IGNNet, and for which graph node representations are obtained using a GNN with parameters θ. Finally, the output layer is used to form the prediction.

3.3 How can IGNNet achieve high performance while maintaining interpretability?

An expressive GNN can potentially capture complex patterns and dependencies in the graph, allowing nodes to be mapped to distinct representations based on their characteristics and relationships [26]. Moreover, a GNN with an injective aggregation scheme can not only distinguish different structures but also map similar structures to similar representations [39]. Therefore, if the tabular data are properly presented as graphs, GNNs with the aforementioned expressive capacities can model relationships and interactions between features, and consequently approximate complex non-linear mappings from inputs to predictions. On top of that, it has been shown by [14] that GCNs based on 1-Lipschitz continuous activation functions can be improved in stability and robustness with Lipschitz normalization and continuity analysis; similar findings have also been demonstrated on graph attention networks (GAT) [11]. This property is of particular importance when the application domain endures adversarial attacks or incomplete tabular data.

The proposed readout function in subsection 3.1 can produce an interpretable output layer. However, it does not guarantee the interpretability of the whole GNN without message-passing layers that consistently maintain relevant representations of the input features. Accordingly, we constrain the message-passing layer to produce interpretable models using the following conditions:

1. Each feature is represented in a distinct node throughout the consecutive layers.
2. Each node has a highly weighted self-loop which conveys the main message of the node.
3. Each node is bounded to interact with a particular neighborhood, where it maintains correlations with the nodes within that neighborhood.

The message-passing operation is based on the linear relationships between nodes. The strength of a passed message is directly proportional to the linear relationship between the two nodes, and the sign of the correlation value determines the signs of the messages. In addition, the highly weighted self-loops maintain the main messages carried by the nodes and prevent them from fading away through multiple message-passing layers. As a result, the aggregated messages could potentially hold significance to the input feature values. The proposed graph pooling function, combined with the constrained message-passing layers that keep representative information about the input features, allows tracking each feature's contribution at the output layer and also through the message-passing layers all the way to the input features.

4 Empirical Investigation

This section evaluates both the explanations and predictive performance of IGNNet. We begin by outlining the experimental setup, then by evaluating the explanations produced by IGNNet, and lastly, we benchmark the predictive performance.

4.1 Experimental Setup

A GNN consists of one or more message-passing layers, and each layer can have a different design, e.g., different activation functions and batch normalization, which is the intra-layer design level [43].

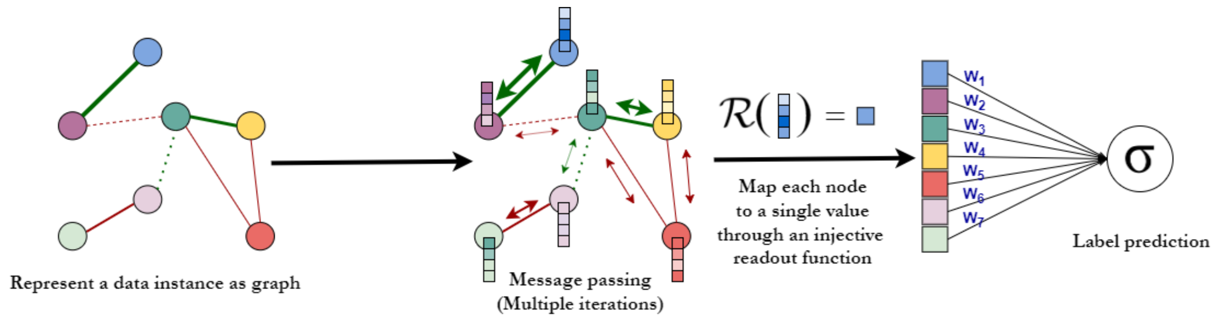


Figure 1: **An overview of our proposed approach.** Each data instance is represented as a graph by embedding the feature values into a higher dimensionality, and the edge between two features (nodes) is the correlation value. Multiple iterations of message passing are then applied. Finally, the learned node representation is projected into a single value, and a whole graph representation is obtained by concatenating the projected values.

There is also the inter-layer design level, which involves, for instance, how the message-passing layers are organized into a neural network and if skip connections are added between layers [43]. While the intra-layer and inter-layer designs can vary based on the nature of the prediction task, we propose a general architecture for our empirical investigation.[1] However, it is up to the user to modify the architecture of the GNN. The number of message-passing layers, number of units in linear transformations, and other hyperparameters were found based on a quasi-random search and evaluation on development sets of the following three datasets: Churn, Electricity, and Higgs. We have six message-passing layers in the proposed architecture, each with a Relu activation function. Multiple learnable weights are also applied to the nodes' representation, followed by a Relu function. Besides three batch normalization layers, four skip connections are added as illustrated in Figure 2. After all the GNN layers, we use a feedforward neural network (FNN) to map the multidimensional representation of each node into a single value. In the FNN, we do not include any activation functions in order to keep the mapping linear, but a sigmoid function is applied after the final layer to obtain a value between 0 and 1 for each node. The FNN is composed of 8 layers with the following numbers of units (128, 64, 32, 16, 8, 4, 2, 1) and 3 batch normalization layers after the second, fourth, and sixth hidden layers. After the FNN, the nodes' final values are concatenated to form a representation of the whole graph (data instance). Finally, the weights that are output are used to make predictions. The GNN is trained end-to-end, starting from the embeddings layer and ending with the class prediction.

We also provide an opaque variant of IGNNet (OGNNet, opaque graph neural net for tabular data), where the FNN, along with the output layer, is replaced by an MLP of one hidden layer with 1024 hidden units and a Relu activation function. All the learned node representations just before the FNN are concatenated and passed to the MLP for class prediction. The OGNNet is introduced to determine, by an ablation study, how much predictive performance we may lose by squashing the learned multidimensional representation of the nodes into scalar values and applying a white box classifier instead of a black box.

In the experiments, 35 publicly available datasets are used.[2] Each dataset is split into training, development, and test sets. The development set is used for overfitting detection and early stopping of the training process, the training set is used to train the model, and the test set is used to evaluate the model.[3] For a fair comparison, all the compared learning algorithms are trained without hyperparameters tuning using the default settings on each dataset. In cases where the learning algorithm does not employ the development set to measure performance progress for early stopping, the development and training subsets are combined into a joint training set. The adjacency matrix uses the correlation values computed on the training data split. The weight on edge from the node to itself (self-loop) is a user-adjustable hyperparameter, constitutes between 70% to 90% (on average) of the weighted summation to keep a strong message per node that does not fade out with multiple layers of message-passing. Weak correlation values are excluded from the graph, so if the absolute correlation value is below 0.2, the edge is removed unless no correlation values are above 0.2; in case of the latter, the procedure is repeated using a reduced threshold of 0.05.[4] The Pearson correlation coefficient [31] is used to estimate the linear relationship between features. In the data preprocessing step, the categorical features are binarized using one-hot encoding, and all the feature values are normalized using min-max normalization (the max and min values are computed on the training split). The normalization keeps the feature values between 0 and 1, which is essential for the IGNNet to have one scale for all nodes.

The following algorithms are also evaluated in the experiments: XGBoost, Random Forests, MLP and TabNet. XGBoost and Random Forests are trained on the combined training and development sets. The MLP has two layers of 1024 units with Relu activation function and is trained on the combined training and development sets with early stopping and 0.1 validation fraction. TabNet is trained with early stopping after 20 consecutive epochs without improvement on the development set, and the best model is used in the evaluation.

For imbalanced binary classification datasets, we randomly oversample the minority class in the training set to align the size with the majority class. All the compared algorithms are trained using the oversampled training data. While for multi-class datasets, no oversampling is conducted.

The area under the ROC curve (AUC) is used to measure the predictive performance, as it is not affected by the decision threshold. For the multi-class datasets, weighted AUC is calculated, i.e., the AUC is computed for each class against the rest and weighted by the support.

[1] The source code is available here: https://github.com/amrmalkhatib/ignnet
[2] All the datasets were obtained from https://openml.org

[3] Detailed information about each dataset is provided in the appendix: https://arxiv.org/abs/2308.08945
[4] The appendix includes an ablation study on how various hyperparameter preferences affect predictive performance.

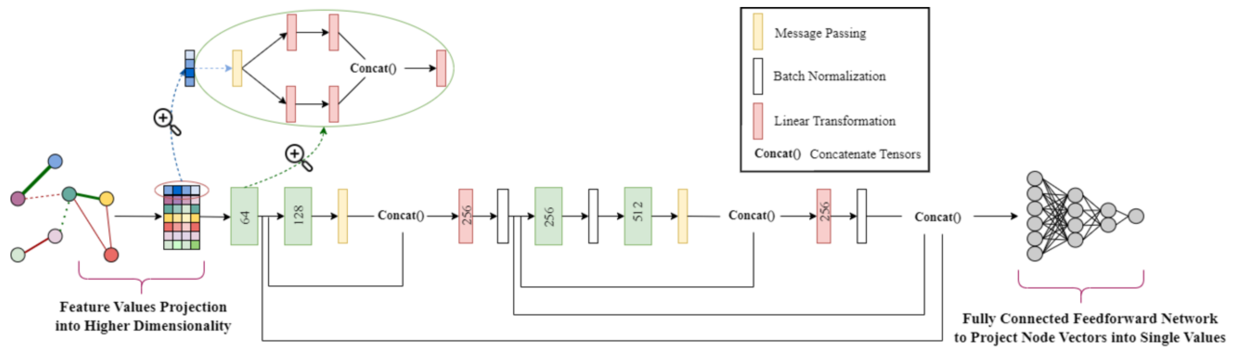

Message Passing
Batch Normalization
Linear Transformation
Concat() Concatenate Tensors
Concat()
64
128
256
512
Feature Values Projection into Higher Dimensionality
Fully Connected Feedforward Network to Project Node Vectors into Single Values

Figure 2: **IGNNet default architecture.** It starts with a layer to project the features into higher dimensionality, a linear transformation from one dimension to 64 dimensions. A Relu activation function follows each message-passing layer and each green block as well. The feedforward network at the end has no activation functions between layers to ensure a linear transformation into a single value. A sigmoid activation function follows the feedforward network to obtain the final value for each feature between 0 and 1.

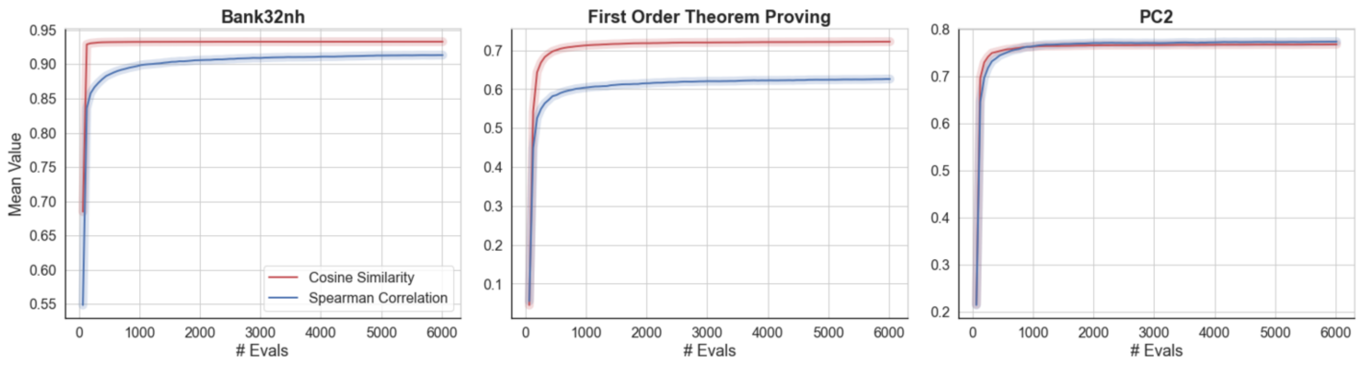


Figure 3: **Comparison of KernelSHAP's approximations and the importance scores obtained from IGNNet.** We measure the similarity of KernelSHAP's approximations to the scores of IGNNet at each iteration of data sampling and evaluation of KernelSHAP. KernelSHAP exhibits improvement in approximating the scores derived from IGNNet with more data sampling.

4.2 Evaluation of Explanations

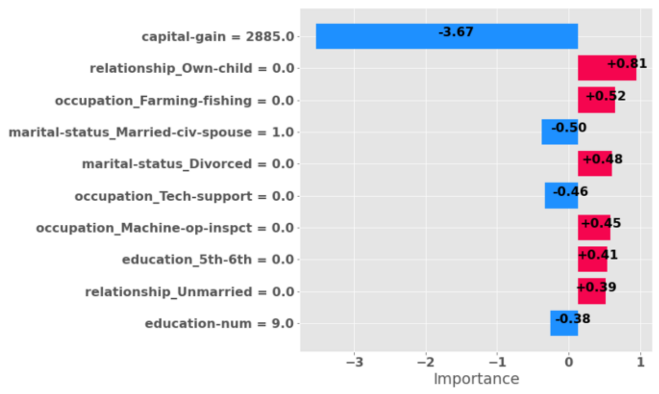
The feature scores produced by IGNNet should ideally reflect the contribution of each feature toward the predicted outcome and, therefore, they should be equivalent to the true Shapley values. As it has been shown that KernelSHAP converges to the true Shapley values when provided with an infinite number of samples [8, 22], it is anticipated that the explanations generated by KernelSHAP will progressively converge to more similar values to the scores of IGNNet as the sampling process continues. This convergence arises from KernelSHAP moving towards the true values, while the scores of IGNNet are expected to align with these true values. To examine this conjecture, we explain IGNNet using KernelSHAP and measure the similarity between KernelSHAP's explanations and IGNNet's scores following each iteration of data sampling and KernelSHAP evaluation. For the feasibility of the experiment, 500 examples are randomly selected from the test set of each dataset to be explained. The cosine similarity and Spearman rank-order correlation are used to quantify the similarity between explanations. The cosine similarity measures the similarity in the orientation, while the Spearman rank-order measures the similarity in ranking the importance scores [33].

The results demonstrate a general trend wherein KernelSHAP's explanations converge to more similar values to IGNNet's scores across various data instances and the 35 datasets, as depicted in Figure 3. The consistent convergence to more similar values clearly indicates that IGNNet provides transparent models with feature scores aligned with the true Shapley values.[5]

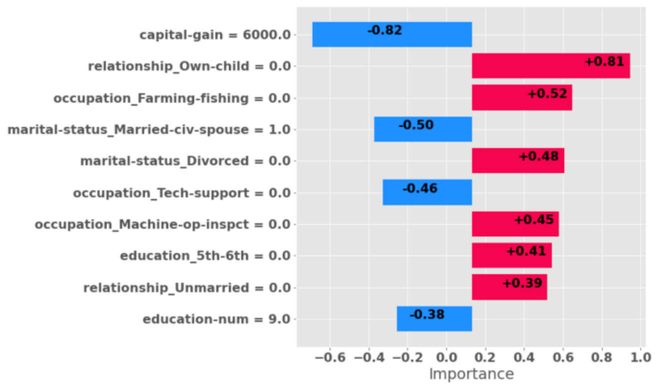
4.3 Illustration of Explanations

In this section, we show, using a toy example, how the computed feature scores by IGNNet can be used to understand the feature contributions toward a specific prediction. Note that this is done in exactly the same way as how one would interpret the predictions of a logistic regression model or the feature importance scores generated by the SHAP [27] or LIME [34] explainer. As the computed feature scores reveal exactly how IGNNet formed the prediction, the user can directly see which features have the greatest impact on the final prediction, and possibly also how they may be modified to affect the outcome. To demonstrate this, we present the feature scores for predictions made by IGNNet using an example from the Adult dataset.[6] In the following illustration, we display the feature scores centered around the bias value, which, when summed with the bias, will produce the exact outcome of IGNNet if the sigmoid function is applied. The scores are sorted according to their absolute values, and only the top 10 features are plotted for ease of presentation. A displayed score τ_i of feature x_i represents all the weights and the computations applied to the input value, as shown in equation 5.

[5] The complete results of the 35 datasets are provided in the appendix.
[6] We provide another example from the Churn dataset in the appendix.



(a) The original data point.



(b) The data point with a modified capital gain value.

Figure 4: Explanation to a single prediction on Adult dataset.

$$
\begin{aligned}
\tau_i =& w_i \mathcal{R}(\varphi(\mathbf{w}^{(l)}(\sum_{u \in \mathcal{N}(i)} \delta_{i,u} \mathbf{h}_u^{(l)} \\
&+ \delta_{i,i} \varphi(\mathbf{w}^{(l-1)}(\sum_{u \in \mathcal{N}(i)} \delta_{i,u} \mathbf{h}_u^{(l-1)} \\
&+ \delta_{i,i}(\ldots \varphi(\mathbf{w}^{(0)}(\sum_{u \in \mathcal{N}(i)} \delta_{i,u} \mathbf{x}_u + \delta_{i,i} \mathbf{x}_i)\ldots))))))
\end{aligned}
\tag{5}
$$

IGNNet predicted the negative class ($\leq$ 50K) with a narrow margin (0.495). The explanation shows that a single feature (capital-gain=2885) has the highest contribution compared to any other feature value. In the training data, the capital-gain has a maximum value of 99999.0, a minimum value of 0, a mean value of 1068.36, and a 7423.08 standard deviation. Therefore, the capital-gain is relatively low for the selected data instance. To test if the explanation reflects the actual reasoning of IGNNet, we raise the capital-gain value by a smaller value than the standard deviation to be 6000 while leaving the remaining feature values constant, and it turns out to be enough to alter the prediction to a positive ($>$ 50K) with 0.944 as the predicted value. We can also see that the negative score of the capital-gain feature went from -3.67 in the original instance (4a) to -0.82 in the modified instance, as shown in Figure 4b. So the user can adjust the value of an important feature as much as needed to alter the prediction.

4.4 *Evaluation of Predictive Performance*

Detailed results for IGNNet and the five competing algorithms on the 35 datasets are shown in Table 1. The ranking of the six algorithms across 35 datasets, based on their AUC values, reveals OGNNet to exhibit superior performance, claiming the top position, closely followed by IGNNet and XGBoost. In order to investigate whether the observed differences are statistically significant, the Friedman test [16] was employed, which indeed allowed to reject the null hypothesis, i.e., reject that there is no difference in predictive performance, as measured by the AUC, at the 0.05 level. The result of subsequently applying the post-hoc Nemenyi test [30] to determine what pairwise differences are significant, again at the 0.05 level, is summarized in Figure 5. However, the result shows no specific significant pairwise differences between any of the compared algorithms. Furthermore, the results show that using IGNNet instead of the black-box variant, OGNNet, does not significantly reduce the predictive performance while maintaining performance at the level of other powerful algorithms for tabular data, e.g., XGBoost and Random Forests.

4.5 *Computational Cost*

The computational cost varies based on the architecture of the graph neural network, which can be altered and determined based on the predictive task and the dataset. The cost relative to the conventional graph neural networks remains the same, i.e., the cost does not increase with the proposed approach. The computational cost also depends on the number of features in the dataset. Therefore, the user can decide on the suitable architecture and the acceptable computational cost.

5 Limitations

The proposed approach targets tabular data primarily consisting of numerical features. However, datasets with substantial categorical features impose two challenges. Firstly, Pearson correlation is inadequate in handling correlation between categorical and numerical features or between two categorical features. Secondly, we use one-hot encoding for categorical features, which can result in an exponential growth in dimensionality when dealing with nominal features with numerous categories. Another limitation arises when input features lack correlation, resulting in a null graph (a completely disconnected graph).

6 Concluding Remarks

We have proposed IGNNet, an algorithm for tabular data classification, which exploits graph neural networks to produce transparent models. In contrast to post-hoc explanation techniques, IGNNet does not approximate or require costly computations, but provides the explanation while computing the prediction, and where the explanation prescribes exactly how the prediction is computed.

We have presented results from a large-scale empirical investigation, in which IGNNet was evaluated with respect to explainability and predictive performance. IGNNet was shown to generate explanations with feature scores aligned with the Shapley values without further computational cost. IGNNet was also shown to achieve a similar predictive performance as XGBoost, Random Forests, TabNet, and MLP, which are all well-known for their ability to generate high-performing models.

One direction for future research is to explore approaches to model feature interactions in the adjacency matrix that go beyond linear correlations. Understanding how such non-linear interactions between

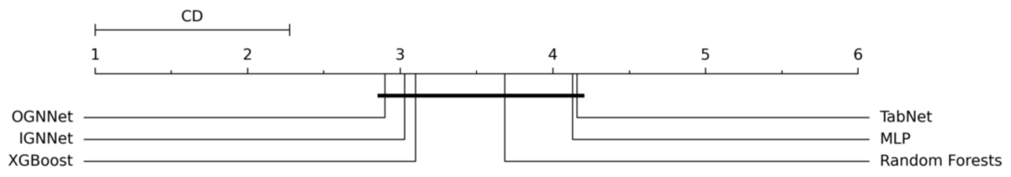


Figure 5: **The average rank of the compared classifiers on the 35 datasets with respect to the AUC** (a lower rank is better), where the critical difference (CD) represents the largest difference that is not statistically significant.

Table 1: The AUC of IGNNet, OGNNet, MLP, Random Forests, and XGBoost. The best-performing model is colored in blue, and the second best-performing is colored in light blue.

Dataset	IGNNet	OGNNet	TabNet	MLP	Random Forests	XGBoost
Abalone	0.881	0.883	0.857	0.877	0.876	0.869
Ada Prior	0.905	0.888	0.848	0.877	0.885	0.894
Adult	0.917	0.915	0.919	0.881	0.907	0.931
Bank 32 nh	0.887	0.887	0.881	0.859	0.876	0.874
Covertype	0.984	0.988	0.969	0.861	0.995	0.967
Credit Card Fraud	0.987	0.966	0.969	0.913	0.914	0.975
Delta Ailerons	0.977	0.977	0.974	0.977	0.978	0.977
Electricity	0.901	0.929	0.894	0.928	0.97	0.973
Elevators	0.951	0.948	0.95	0.95	0.913	0.943
HPC Job Scheduling	0.908	0.921	0.775	0.908	0.952	0.955
Fars	0.956	0.958	0.954	0.955	0.949	0.962
1st Order Theorem Proving	0.776	0.808	0.495	0.805	0.854	0.858
Helena	0.875	0.889	0.884	0.897	0.855	0.875
Heloc	0.783	0.787	0.772	0.783	0.778	0.775
Higgs	0.762	0.785	0.804	0.774	0.793	0.797
Indian Pines	0.984	0.992	0.99	0.973	0.979	0.987
Jannis	0.856	0.857	0.867	0.861	0.861	0.872
JM1	0.739	0.725	0.711	0.728	0.747	0.733
LHC Identify Jets	0.941	0.941	0.944	0.861	0.935	0.941
Madelon	0.906	0.718	0.501	0.668	0.79	0.891
Magic Telescope	0.907	0.921	0.927	0.929	0.934	0.928
MC1	0.957	0.904	0.89	0.853	0.844	0.943
Mozilla4	0.954	0.961	0.971	0.963	0.988	0.99
Microaggregation2	0.778	0.792	0.752	0.782	0.768	0.781
Numerai28.6	0.526	0.534	0.52	0.518	0.519	0.514
Otto Group Product	0.96	0.971	0.968	0.972	0.973	0.974
PC2	0.881	0.815	0.844	0.571	0.55	0.739
Phonemes	0.922	0.96	0.898	0.93	0.964	0.953
Pollen	0.492	0.508	0.509	0.496	0.489	0.467
Satellite	0.998	0.993	0.911	0.992	0.998	0.992
Scene	0.994	0.989	0.986	0.992	0.983	0.982
Speed Dating	0.853	0.835	0.797	0.822	0.845	0.86
Telco Customer Churn	0.858	0.845	0.841	0.783	0.84	0.843
Vehicle sensIT	0.918	0.918	0.917	0.914	0.912	0.916
Waveform-5000	0.965	0.962	0.933	0.965	0.959	0.957

features may impact the model's interpretability could be an intriguing area of exploration. A second direction is to investigate alternative encoders for categorical features rather than relying on one-hot encoding. It would also be interesting to extend IGNNet to handle non-tabular datasets, including images and text, which would require entirely different approaches to representing each data point as a graph. Another important direction for future work is to use IGNNet for studying possible adversarial attacks on a predictive model. Finally, an important direction would be to complement the empirical evaluation with user-grounded evaluations, e.g., measuring to what extent certain tasks could be more effectively and efficiently solved when the users are provided with a transparent model that shows how the prediction has been computed from the input.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] M. Al-Shedivat, A. Dubey, and E. Xing. Contextual explanation networks. *J. Mach. Learn. Res.*, 21(1), 2022.

[2] A. Alkhatib, H. Bostrom, S. Ennadir, and U. Johansson. Approximating score-based explanation techniques using conformal regression. In *Proceedings of the 12th Symposium on Conformal and Probabilistic Prediction with Applications*, volume 204, pages 450–469, 2023.

[3] D. Alvarez Melis and T. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] S. O. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (8):6679–6687, 2021.

[5] J. Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[7] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[8] I. Covert and S.-I. Lee. Improving kernelshap: Practical shapley value estimation using linear regression. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 3457–3465. PMLR, 13–15 Apr 2021.

[9] H. Cui, W. Dai, Y. Zhu, X. Li, L. He, and C. Yang. Interpretable graph neural networks for connectome-based brain disorder analysis. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022: 25th International Conference*, page 375–385, 2022.

[10] E. Dai and S. Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 302–311, New York, NY, USA, 2021. Association for Computing Machinery.

[11] G. Dasoulas, K. Scaman, and A. Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2456–2466. PMLR, 18–24 Jul 2021.

[12] J. Delaunay, L. Galárraga, and C. Largouët. Improving Anchor-based Explanations. In *CIKM 2020 - 29th ACM International Conference on Information and Knowledge Management*, pages 3269–3272, Galway / Virtual, Ireland, Oct. 2020. ACM.

[13] L. Du, F. Gao, X. Chen, R. Jia, J. Wang, J. Zhang, S. Han, and D. Zhang. Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'21)*, 2021.

[14] S. Ennadir, Y. Abbahaddou, J. F. Lutzeyer, M. Vazirgiannis, and H. Boström. A simple and yet fairly effective defense for graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21063–21071, Mar. 2024.

[15] A. Feng, C. You, S. Wang, and L. Tassiulas. Kergnns: Interpretable graph neural networks with graph kernels. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6614–6622, Jun. 2022.

[16] M. Friedman. A correction: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 34(205):109–109, 1939.

[17] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3): 50–57, 2017.

[18] H. Guo, T. Nguyen, and A. Yadav. Counternet: End-to-end training of counterfactual aware predictions. In *ICML 2021 Workshop on Algorithmic Recourse*, 2021.

[19] X. Guo, Y. Quan, H. Zhao, Q. Yao, Y. Li, and W. Tu. Tabgnn: Multiplex graph neural network for tabular data prediction. *CoRR*, abs/2108.09127, 2021.

[20] V. Guyomard, F. Fessant, T. Guyet, T. Bouadi, and A. Termier. Vcnet: A self-explaining model for realistic counterfactual generation. In *Machine Learning and Knowledge Discovery in Databases, ECML PKDD*, 2022.

[21] S. Ivanov and L. Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[22] N. Jethani, M. Sudarshan, I. C. Covert, S.-I. Lee, and R. Ranganath. FastSHAP: Real-time shapley value estimation. In *International Conference on Learning Representations*, 2022.

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

[24] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Interpretable & explorable approximations of black box models. *CoRR*, abs/1707.01154, 2017.

[25] T. Lei, R. Barzilay, and T. Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[26] P. Li and J. Leskovec. The expressive power of graph neural networks. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 63–98. Springer Singapore, Singapore, 2022.

[27] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[28] J. M. Metsch, A. Saranti, A. Angerschmid, B. Pfeifer, V. Klemt, A. Holzinger, and A.-C. Hauschild. Clarus: An interactive explainable ai platform for manual counterfactuals in graph neural networks. *Journal of Biomedical Informatics*, 150:104600, 2024.

[29] T. Mori and N. Uchihira. Balancing the trade-off between accuracy and interpretability in software defect prediction. *Empirical Software Engineering*, 24(2):779–825, 2019.

[30] P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

[31] K. Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London Series I*, 58:240–242, Jan. 1895.

[32] E. Pintelas, I. E. Livieris, and P. Pintelas. A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability. *Algorithms*, 13(1), 2020.

[33] A. H. A. Rahnama, J. Bütepage, P. Geurts, and H. Boström. Evaluation of local model-agnostic explanations using ground truth. *CoRR*, abs/2106.02488, 2021.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[35] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *The 32nd Conference on Artificial Intelligence (AAAI)*, 2018.

[36] R. Richman and M. V. Wüthrich. Localglmnet: interpretable deep learning for tabular data. *Scandinavian Actuarial Journal*, 0(0):1–25, 2022.

[37] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[38] Y. Sawada and K. Nakamura. Concept bottleneck model with additional unsupervised concepts. *IEEE Access*, 10:41758–41765, 2022.

[39] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[40] H. Xuanyuan, P. Barbiero, D. Georgiev, L. C. Magister, and P. Liò. Global concept-based interpretability for graph neural networks via neuron analysis. *Proceedings of AAAI Conference on Artificial Intelligence*, 37(9):10675–10683, Jun. 2023.

[41] C.-K. Yeh, C.-Y. Hsieh, A. Suggala, D. I. Inouye, and P. K. Ravikumar. On the (in)fidelity and sensitivity of explanations. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[42] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4805–4815, Red Hook, NY, USA, 2018. Curran Associates Inc.

[43] J. You, R. Ying, and J. Leskovec. Design space for graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[44] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence and 13th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018.

[45] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C.-K. Lee. Protgnn: Towards self-explaining graph neural networks. In *AAAI*, 2022.

[46] K. Zhou, Z. Liu, R. Chen, L. Li, S.-H. Choi, and X. Hu. Table2graph: Transforming tabular data to unified weighted graph. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 2420–2426, 7 2022.