# INTERPRETABLE GRAPH NEURAL NETWORKS FOR TABULAR DATA

**Amr Alkhatib, Sofiane Ennadir & Henrik Boström**
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology, Sweden
{alkhat,ennadir,bostromh}@kth.se

**Michalis Vazirgiannis**
DaSciM, LIX, École Polytechnique
Institut Polytechnique de Paris, France
mvazirg@lix.polytechnique.fr

## 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 (Lakkaraju et al., 2017; Goodman & Flaxman, 2017). 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 (Berkson, 1944), 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 (Chen & Guestrin, 2016), Random Forests (Breiman, 2001), and deep learning algorithms (Pintelas et al., 2020; Mori & Uchihira, 2019). Post-hoc explanation techniques, e.g., SHAP (Lundberg & Lee, 2017), LIME (Ribeiro et al., 2016), and Anchors (Ribeiro et al., 2018), have been put forward as tools to explain predictions of the black-box models. However, the explanations provided by such techniques are limited in that they either do not show how exactly the predictions are computed, but merely present feature scores, such as LIME and SHAP, or come with no guarantees on the fidelity, i.e., that the provided explanation agrees with the underlying model (Yeh et al., 2019; Delaunay et al., 2020). As extensively argued in (Rudin, 2019), 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 (Xu et al., 2019). 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 (Guo et al., 2021) is an example of the first approach, where each data point is represented as a node in a graph. In comparison, TabularNet (Du et al., 2021) and Table2Graph (Zhou et al., 2022) 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 (Ivanov & Prokhorenkova, 2021), 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, as well as to an algorithm generating interpretable models; TabNet (Arik & Pfister, 2021)

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. Finally, in the concluding remarks section, we summarize the main conclusions and point out directions for future work.

## 2 Related Work

### 2.1 Self-Explaining Graph Neural Networks

The Self-Explaining GNN (SE-GNN) (Dai & Wang, 2021) uses similarities between nodes to make predictions on the nodes' labels and provide explanations using the most similar K nodes with labels. ProtGNN (Zhang et al., 2022) also computes similarities, but between the input graph and prototypical graph patterns that are learned per class. Cui et al. (2022) proposed a framework to build interpretable GNNs for connectome-based brain disorder analysis that resembles the signal correlation between different brain areas.

The approaches that target explainable graph neural networks provide abstract views of the predictions; the users cannot trace the exact computations, in contrast to when using transparent models, which in principle allows for the inferences to be executed by hand.

### 2.2 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 (Richman & Wüthrich, 2022). TabNet (Arik & Pfister, 2021) 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

## 3.1 INTERPRETABLE GRAPH NEURAL NETWORK

The input to a GNN learning algorithm is a set of graphs denoted by $\mathcal{G} = (V, E, X, \mathcal{W})$, consisting of a set of nodes $V$, a set of edges $E$, a set of node feature vectors $X$, and a set of edge weights $\mathcal{W}$, where $V = \{v_1, \ldots, v_N\}$, $E \subseteq \{(v_i, v_j) | v_i, v_j \in V\}$, $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, and the weight of edge $(v_i, v_j)$ is represented by a scalar value $\delta_{i,j}$ in the set of edge weights $\mathcal{W}$, where $\delta_{i,j} = \mathcal{W}(i, j)$. A GNN algorithm learns a representation vector $\boldsymbol{h}_i$ for each node $v_i$, which is initialized as $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$. The key steps in a GNN for graph classification can be summarized by the following two phases (Xu et al., 2019):

(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. The message passing phase can be formulated as:

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

where $\delta_{i,u}$ is the weight assigned to the edge between node $v_i$ and node $v_u$.

$\boldsymbol{h}_i^{(l)}$ is the hidden representation of the node $v_i$ in the $l$-th layer, $\boldsymbol{w}^{(l+1)}$ represents the learnable parameters, and $\varphi$ 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) (Kipf & Welling, 2017) as shown in equation 2.

$$\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \tag{2}$$

Here $\boldsymbol{D}$ is the degree matrix $\boldsymbol{D}_{ii} = \sum_j A_{ij}$ (Kipf & Welling, 2017).

(b) **Graph Pooling (Readout):** A representation of the whole graph $\mathcal{G}$ is learned using a simple or advanced function (Xu et al., 2019), 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 (Zhang et al., 2018; Ying et al., 2018).

*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 ($\mathcal{R}(\boldsymbol{h}_i^{(l+1)} \in \mathbb{R}^n) = h_i \in \mathbb{R}^1$), and concatenated to obtain the final representation $\boldsymbol{g}$ of the graph $\mathcal{G}$ where a scalar value in $\boldsymbol{g}$ 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, $\boldsymbol{g}$ 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 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:

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

In the case of binary classification, one vector of weights ($\boldsymbol{w}$) 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 illustrated in Figure 1 and outlined in Algorithm 1.

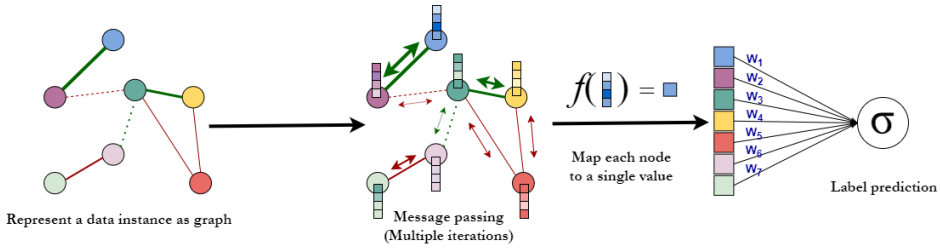



Figure 1: **An overview of our proposed approach.** Each data instance is represented as a graph by embedding the feature values into a higher dimensionality, and the edge between two features (nodes) is the correlation value. Multiple iterations of message passing are then applied. Finally, the learned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

**Algorithm 1:** IGNNet

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

### 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 (Li & Leskovec, 2022). Moreover, a GNN with an injective aggregation scheme can not only distinguish different structures but also map similar structures to similar representations (Xu et al., 2019). 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 Ennadir et al. (2023) 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) (Dasoulas et al., 2021).

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 is bounded to interact with a particular neighborhood, where it maintains correlations with the nodes within that neighborhood. As a result, the aggregated messages could potentially hold significance to the input feature values.

Since the edge weights are the correlation values, they determine the strength and the sign of the messages obtained from a neighborhood, allowing each node to store information not only concerning the original input feature but also features that are correlated with it. 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

### 4.1 Experimental Setup

We propose a general GNN architecture for our empirical investigation.[1] However, it is up to the user to modify the architecture as long as the conditions in subsections 3.1 and 3.3 are satisfied. The hyperparameters, e.g., the number of message-passing layers and the number of units in linear transformations, were found based on a quasi-random search on development sets of the following 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.

In the experiments, 20 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. The Pearson correlation coefficient (Pearson, 1895) is used to estimate the linear relationship between features. In the data

[1] The source code is available at: https://github.com/amrmalkhatib/IGNNet

[2] All the datasets were obtained from `https://www.openml.org`

[3] Detailed information about each dataset is provided in the Appendix C.

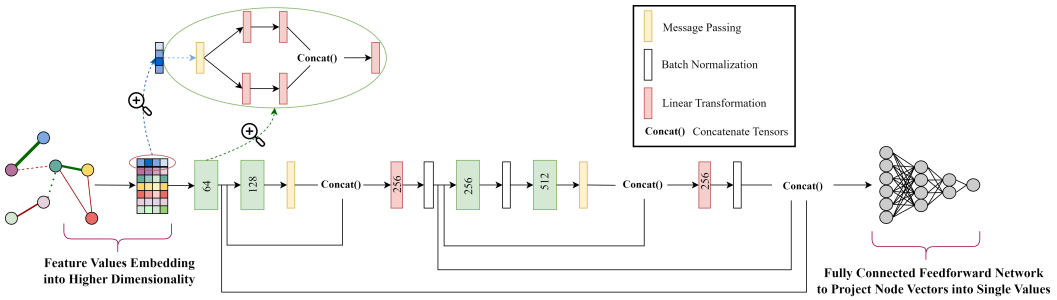



Figure 2: **IGNNet default architecture.** It starts with the embedding layer, a linear transformation from one dimension to 64 dimensions. A Relu activation function follows each message-passing layer and each green block as well. The feedforward network at the end has no activation functions between 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.

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 and TabNet. XGBoost and Random Forests are trained on the combined training and development sets. 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.

### 4.2 Evaluation of Predictive Performance

Detailed results for IGNNet and the three competing algorithms on the 20 datasets are shown in Table 1.The ranking of the four algorithms across 20 datasets, based on their AUC values, reveals IGNNet to exhibit superior performance, claiming the top position, followed by XGBoost. In order to investigate whether the observed differences are statistically significant, the Friedman test (Friedman, 1939) was employed, which did not allow to reject the null hypothesis at the 0.05 level. The results show that IGNNet maintains the performance level of other powerful algorithms for tabular data, e.g., XGBoost and Random Forests.

### 4.3 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 (Covert & Lee, 2021; Jethani et al., 2022), 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.[4] For the feasibility of the experiment, 500 examples are ran-

[4]KernelSHAP experiments were conducted using the following open-source implementation: https://github.com/iancovert/shapley-regression

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

| Dataset | IGNNet | TabNet | Random Forests | XGBoost |
|---|---|---|---|---|
| Abalone | 0.881 | 0.857 | 0.876 | 0.869 |
| Ada Prior | 0.905 | 0.848 | 0.885 | 0.894 |
| Bank 32 nh | 0.887 | 0.881 | 0.876 | 0.874 |
| Covertype | 0.984 | 0.969 | 0.995 | 0.967 |
| Electricity | 0.901 | 0.894 | 0.97 | 0.973 |
| First Order Theorem Proving | 0.776 | 0.495 | 0.854 | 0.858 |
| Helena | 0.875 | 0.884 | 0.855 | 0.875 |
| Heloc | 0.783 | 0.772 | 0.778 | 0.775 |
| Higgs | 0.762 | 0.804 | 0.793 | 0.797 |
| Indian Pines | 0.984 | 0.99 | 0.979 | 0.987 |
| Jannis | 0.856 | 0.867 | 0.861 | 0.872 |
| JM1 | 0.739 | 0.711 | 0.747 | 0.733 |
| Microaggregation2 | 0.778 | 0.752 | 0.768 | 0.781 |
| MC1 | 0.957 | 0.89 | 0.844 | 0.943 |
| Numerai28.6 | 0.526 | 0.52 | 0.519 | 0.514 |
| PC2 | 0.881 | 0.844 | 0.55 | 0.739 |
| Satellite | 0.998 | 0.911 | 0.998 | 0.992 |
| Speed Dating | 0.853 | 0.797 | 0.845 | 0.86 |
| Vehicle sensIT | 0.918 | 0.917 | 0.912 | 0.916 |
| waveform-5000 | 0.965 | 0.933 | 0.959 | 0.957 |

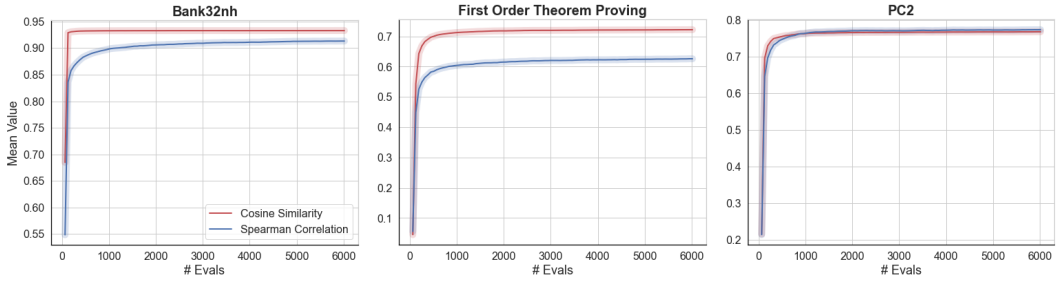



Figure 3: **Comparison of KernelSHAP's approximations and the importance scores obtained from IGNNet.** We measure the similarity of KernelSHAP's approximations to the scores of IGNNet at each iteration of data sampling and evaluation of KernelSHAP. KernelSHAP exhibits improvement in approximating the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

domly 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 (Han et al., 2012), while the Spearman rank-order measures the similarity in ranking the importance scores (Rahnama et al., 2021).

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 20 datasets, as depicted in Figure 3.[5] The consistent convergence to more similar values clearly indicates that IGNNet provides transparent models with feature scores aligned with the true Shapley values.

[5]The complete results of the 20 datasets are provided in Appendix B.

## 5 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, and TabNet, 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 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.

### Acknowledgments

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

## References

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

Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944. ISSN 01621459.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation using linear regression. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3457–3465. PMLR, 13–15 Apr 2021.

Hejie Cui, Wei Dai, Yanqiao Zhu, Xiaoxiao Li, Lifang He, and Carl Yang. Interpretable graph neural networks for connectome-based brain disorder analysis. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII*, pp. 375–385, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-16451-4.

Enyan Dai and Suhang Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 302–311, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469.

George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2456–2466. PMLR, 18–24 Jul 2021.

Julien Delaunay, Luis Galárraga, and Christine Largouët. Improving Anchor-based Explanations. In *CIKM 2020 - 29th ACM International Conference on Information and Knowledge Management*, pp. 3269–3272, Galway / Virtual, Ireland, October 2020. ACM.

Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei 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.

Sofiane Ennadir, Yassine ABBAHADDOU, Michalis Vazirgiannis, and Henrik Boström. A simple and yet fairly effective defense for graph neural networks. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023. URL `https://openreview.net/forum?id=CJgBMut3nC`.

Milton 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.

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

Xiawei Guo, Yuhan Quan, Huan Zhao, Quanming Yao, Yong Li, and Weiwei Tu. Tabgnn: Multiplex graph neural network for tabular data prediction. *CoRR*, abs/2108.09127, 2021.

Jiawei Han, Micheline Kamber, and Jian Pei. 2 - getting to know your data. In Jiawei Han, Micheline Kamber, and Jian Pei (eds.), *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pp. 39–82. Morgan Kaufmann, Boston, third edition edition, 2012. ISBN 978-0-12-381479-1.

Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time shapley value estimation. In *International Conference on Learning Representations*, 2022.

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

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. *CoRR*, abs/1707.01154, 2017.

Pan Li and Jure Leskovec. The expressive power of graph neural networks. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao (eds.), *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 63–98. Springer Singapore, Singapore, 2022.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Toshiki Mori and Naoshi Uchihira. Balancing the trade-off between accuracy and interpretability in software defect prediction. *Empirical Software Engineering*, 24(2):779–825, 2019. ISSN 1573-7616.

Karl Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London Series I*, 58:240–242, January 1895.

Emmanuel Pintelas, Ioannis E. Livieris, and Panagiotis Pintelas. A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability. *Algorithms*, 13(1), 2020. ISSN 1999-4893.

Amir Hossein Akhavan Rahnama, Judith Bütepage, Pierre Geurts, and Henrik Boström. Evaluation of local model-agnostic explanations using ground truth. *CoRR*, abs/2106.02488, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*, pp. 1135–1144, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Ronald Richman and Mario V. Wüthrich. Localglmnet: interpretable deep learning for tabular data. *Scandinavian Actuarial Journal*, 0(0):1–25, 2022.

Cynthia 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.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. On the (in)fidelity and sensitivity of explanations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 4805–4815, Red Hook, NY, USA, 2018. Curran Associates Inc.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

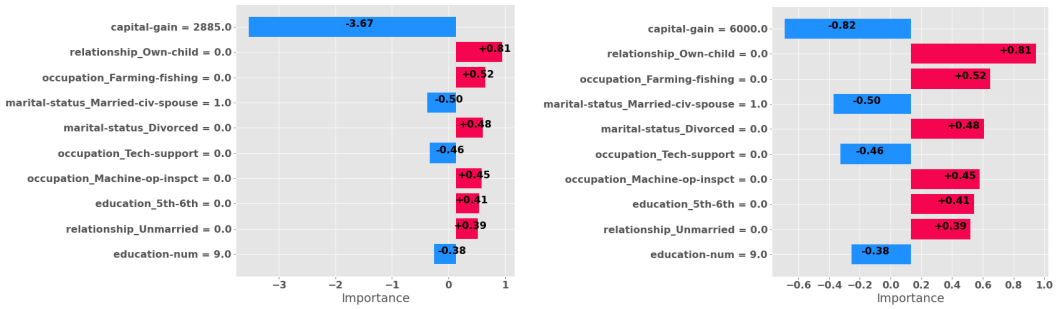
Zaixin Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Protgnn: Towards self-explaining graph neural networks. In *AAAI*, 2022.

Kaixiong Zhou, Zirui Liu, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. Table2graph: Transforming tabular data to unified weighted graph. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 2420–2426. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

## A ILLUSTRATION OF EXPLANATIONS

In this section, we show 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 explainer (Lundberg & Lee, 2017). 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 two examples from the Adult dataset and the Churn dataset. In the following illustrations, 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 $\tau_i$ of feature $x_i$ represents all the weights and the computations applied to the input value, as shown in equation 5.

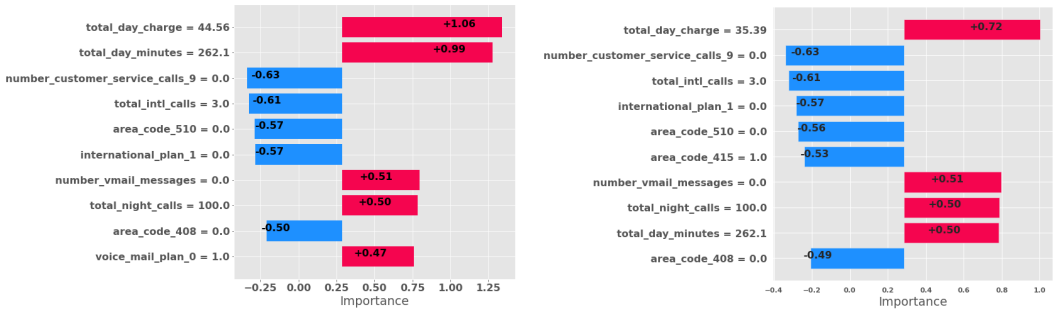
$$\begin{aligned}\tau_i =& \mathbf{w}_i f(\varphi(\mathbf{w}^{(l+1)}(\sum_{u\in\mathcal{N}(i)} \delta_{i,u}\mathbf{h}_u^{(l)} \\ &+ \delta_{i,i}\varphi(\mathbf{w}^{(l)}(\sum_{u\in\mathcal{N}(i)} \delta_{i,u}\mathbf{h}_u^{(l-1)} \\ &+ \delta_{i,i}(...\varphi(\mathbf{w}^{(1)}(\sum_{u\in\mathcal{N}(i)}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




(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.




(a) The original data point.

(b) The data point with a modified total day charge value.

Figure 5: The explanation of a single prediction on Churn dataset.

The first example, from the Adult dataset, shown in Figure 4a. 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,

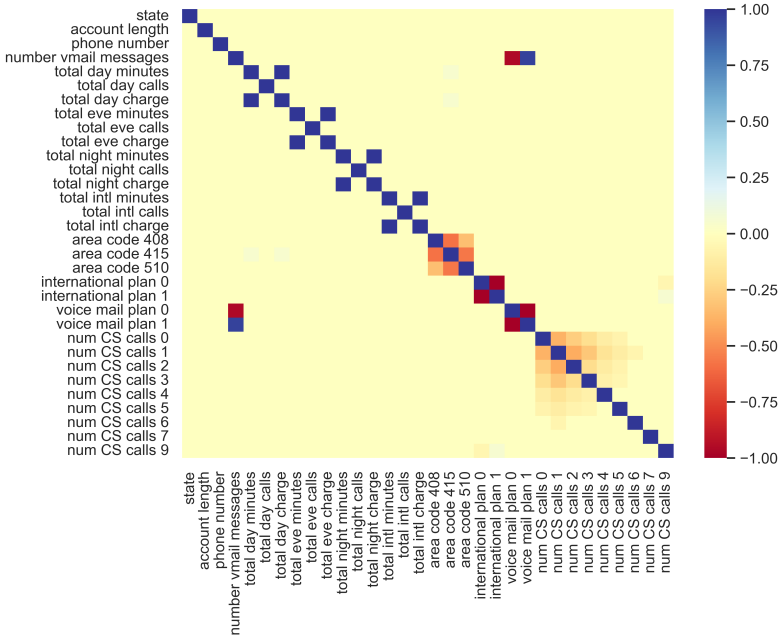



Figure 6: The correlation matrix of the features of the Churn dataset.

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. 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.

The data point, from the Churn dataset, has a positive prediction with a narrow margin (0.565). Consequently, the reduction of the top positively important feature (total day charge), as illustrated in Fig. 5a, may be enough to obtain a negative prediction. The total day charge has a maximum value of 59.76, a minimum of 0.44, a mean of 30.64, and a 9.17 standard deviation in the training set. However, the total day charge is highly correlated with the total day minutes by more than 0.99, as shown in Fig. 6. The high correlation effect is obvious in the outcome when the total day charge value is reduced by one standard deviation, from 44.56 to 35.39, as the scores of both total day charge and total day minutes drop from 1.06 and 0.99 to 0.72 and 0.5, respectively, as shown in Fig. 5b. Moreover, the sum of feature scores and bias falls below 0.5 after the sigmoid function, resulting in a negative prediction with a predicted value of 0.297.

## B Transparency Evaluation

In this section, we demonstrate the detailed results of the explanations evaluation experiment using 20 datasets. The results show a general trend across the 20 datasets where the explanations obtained using KernelSHAP converge to more similar values to the feature scores produced by IGNNet, given more sampled data. The results are displayed in Figure 7 and Figure 8.

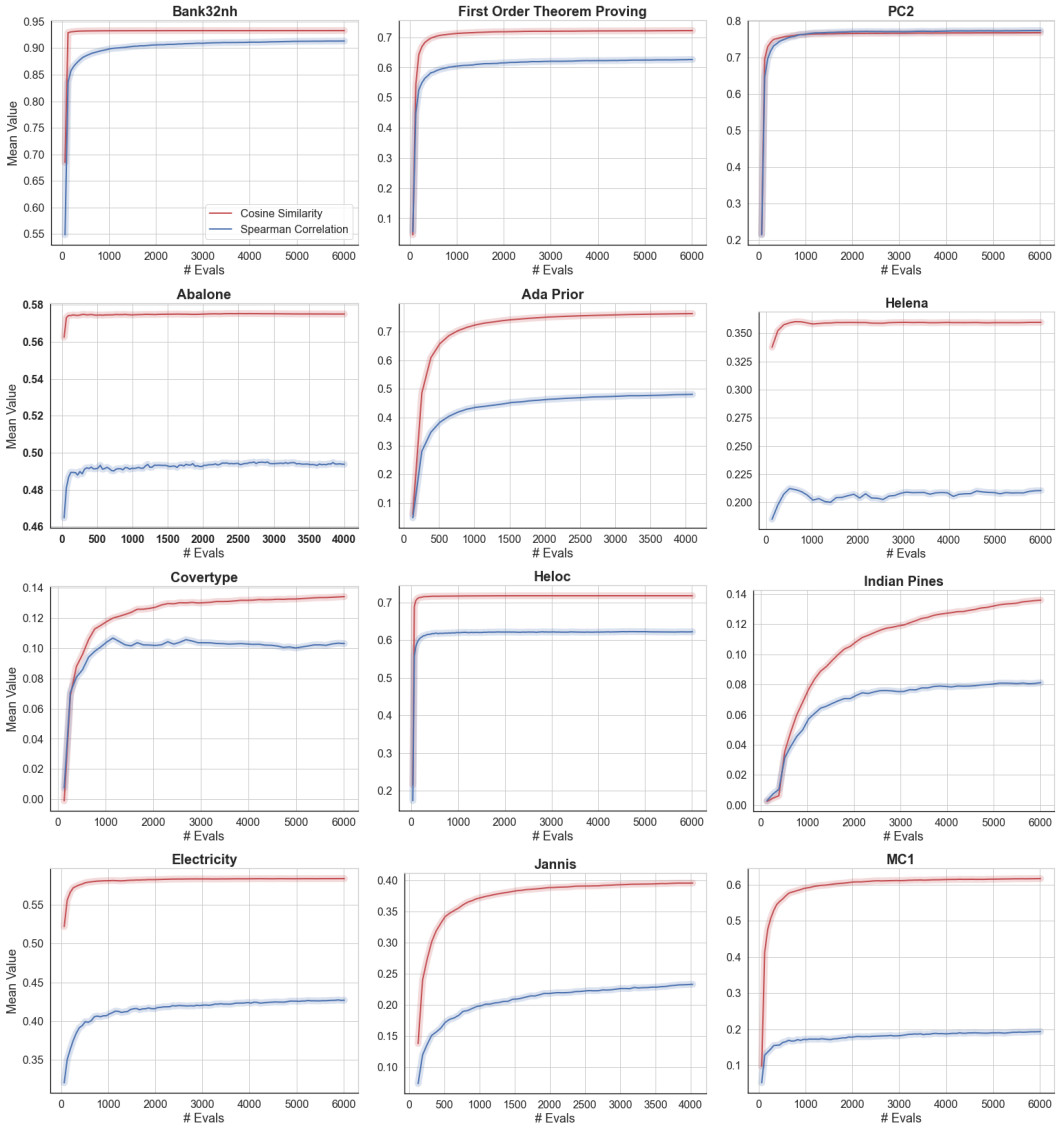



Figure 7: **Comparison of KernelSHAP's approximations and the importance scores obtained from IGNNet.** We measure the similarity of KernelSHAP's approximations to the scores of IGNNet at each iteration of data sampling and evaluation of KernelSHAP. KernelSHAP exhibits improvement in approximating the scores derived from IGNNet with more data sampling.

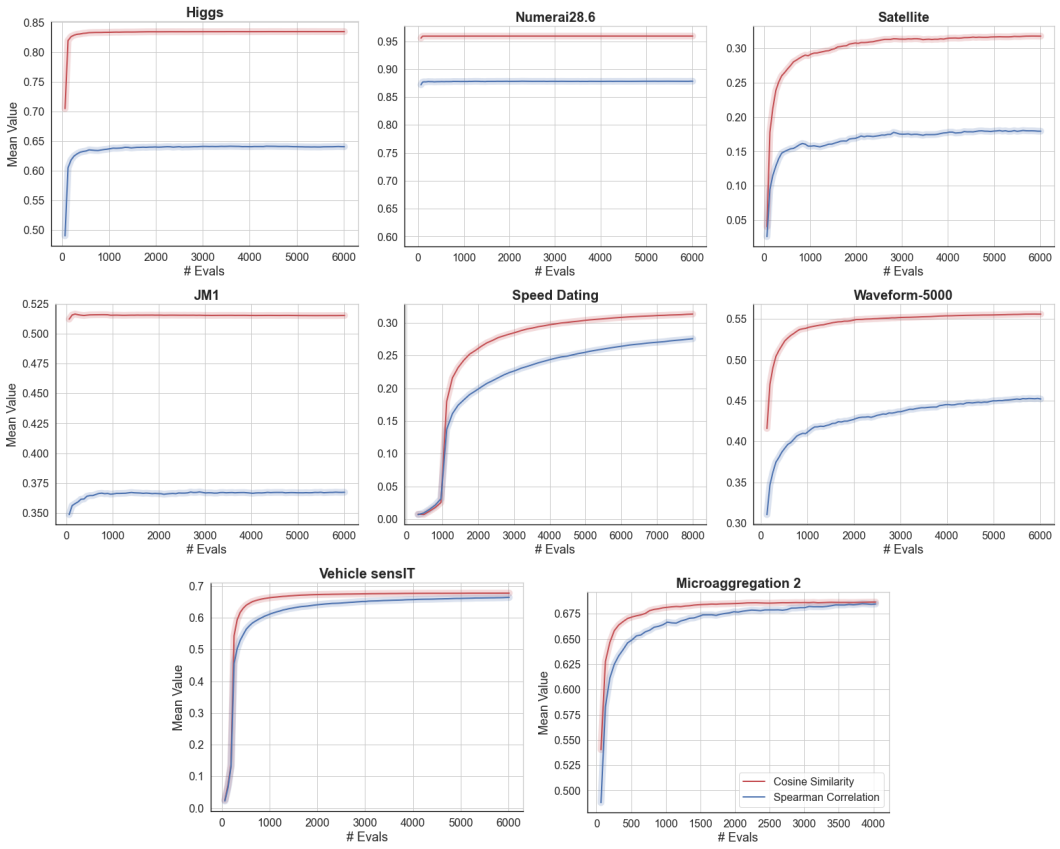



Figure 8: **Comparison of KernelSHAP's approximations and the importance scores obtained from IGNNet.** We measure the similarity of KernelSHAP's approximations to the scores of IGNNet at each iteration of data sampling and evaluation of KernelSHAP. KernelSHAP exhibits improvement in approximating the scores derived from IGNNet with more data sampling.

## C Information about the Used Datasets and Specifications of the Hardware

This subsection provides a summary of the datasets utilized in the experiments. In Table 2, we provide information about the used datasets, including the number of classes (Num of Classes), the number of features, the size of the dataset, the size of the training, validation, and test splits, as well as, the used correlation threshold (Corr. Thr.) of each dataset, the weight on the self-loop (SL Wt.), the number of training epochs, and finally the ID of each dataset on OpenML.

The experiments have been performed in a Python environment on an Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz system with 64.0 GB of RAM, and the GPUs are NVIDIA® GeForce® GTX 1650 Ti with 4 GB GDDR6, and NVIDIA® GeForce® GTX 1080 Ti with 8 GB. All the software and package dependencies are documented with the source code.

Table 2: The dataset information.

| Dataset | Classes | Features | Size | Train. Set | Dev. Set | Test Set | Corr. Thr. | SL Wt. | Epochs | ID |
|---|---|---|---|---|---|---|---|---|---|---|
| Abalone | 2 | 8 | 4177 | 2506 | 836 | 835 | 0.2 | 20 | 220 | 720 |
| Ada Prior | 2 | 14 | 4562 | 2737 | 913 | 912 | 0.2 | 4 | 216 | 1037 |
| Bank 32 nh | 2 | 32 | 8192 | 5734 | 1229 | 1229 | 0.2 | 2 | 540 | 833 |
| Covertype | 7 | 54 | 581012 | 524362 | 27599 | 29051 | 0.2 | 10 | 300 | 1596 |
| Electricity | 2 | 8 | 45312 | 36249 | 4532 | 4531 | 0.2 | 3 | 396 | 151 |
| 1st Order Theorem | 6 | 51 | 6118 | 3915 | 979 | 1224 | 0.2 | 30 | 848 | 1475 |
| Helena | 100 | 27 | 65196 | 41724 | 10432 | 13040 | 0.2 | 10 | 550 | 41169 |
| Heloc | 2 | 22 | 10000 | 7500 | 1250 | 1250 | 0.2 | 20 | 234 | 45023 |
| Higgs | 2 | 28 | 98050 | 88245 | 4903 | 4902 | 0.05 | 4 | 394 | 23512 |
| Indian Pines | 8 | 220 | 9144 | 5852 | 1463 | 1829 | 0.2 | 400 | 394 | 41972 |
| Jannis | 4 | 54 | 83733 | 53588 | 13398 | 16747 | 0.05 | 20 | 300 | 41168 |
| JM1 | 2 | 21 | 10885 | 8708 | 1089 | 1088 | 0.2 | 50 | 187 | 1053 |
| Microaggregation2 | 5 | 20 | 20000 | 12800 | 3200 | 4000 | 0.2 | 15 | 599 | 41671 |
| MC1 | 2 | 38 | 9466 | 7478 | 994 | 994 | 0.2 | 80 | 198 | 1056 |
| Numerai28.6 | 2 | 21 | 96320 | 86688 | 4816 | 4816 | 0.2 | 20 | 36 | 23517 |
| PC2 | 2 | 36 | 5589 | 3353 | 1118 | 1118 | 0.2 | 60 | 37 | 1069 |
| Satellite | 2 | 36 | 5100 | 2805 | 1148 | 1147 | 0.2 | 60 | 287 | 40900 |
| Speed Dating | 2 | 120 | 8378 | 5864 | 1257 | 1257 | 0.2 | 10 | 58 | 40536 |
| Vehicle sensIT | 2 | 100 | 98528 | 88675 | 4927 | 4926 | 0.2 | 10 | 172 | 357 |
| waveform-5000 | 2 | 40 | 5000 | 3000 | 1000 | 1000 | 0.2 | 4 | 97 | 979 |