Node-Level Data Valuation on Graphs

Anonymous authors
Paper under double-blind review

Abstract

How much is a node worth? We answer this question using an emerging set of data valuation techniques, where the value of a data point is measured via its marginal contribution when added to the (training) dataset. Data valuation has been primarily studied in the i.i.d. setting, giving rise to methods like influence functions, leave-one-out estimation, data Shapley, and data Banzhaf. We conduct a comprehensive study of data valuation approaches applied to graph-structured models such as graph neural networks in a semi-supervised transductive setting. Since all nodes (labeled and unlabeled) influence both training and inference we construct various scenarios to understand the diverse mechanisms by which nodes can impact learning. We show that the resulting node values can be used to identify (positively and negatively) influential nodes, quantify model brittleness, detect poisoned data, and accurately predict counterfactuals[1].

1 Introduction

When we learn from data, a natural question to ask is *how each data point influences learning*. We can study how a given training point affects the learned weights, the accuracy, or the prediction for a given test point. Beyond offering insights into our models, this question is also practically important, especially for modern machine learning methods that rely on increasingly larger datasets (Zhou et al., 2017). For example, if we gather data from unreliable sources (e.g. the internet) we may want to filter out low quality instances. When we obtain data from data markets, different data providers (e.g. individuals, hospitals) should be equitably compensated for the data that they provide. Data values have been used to quantify train-test leakage, find semantically similar examples (Ilyas et al., 2022), detect mislabeled instances (Wang & Jia, 2023) and for dataset selection (Engstrom et al., 2024).

A classical answer to the data valuation question is influence functions (Cook & Weisberg, 1982; Koh & Liang, 2017). They approximate the effect of removing a data point on the model performance. From linear regression to deep learning, they have been thoroughly studied in various settings, including their limitations (Basu et al., 2021; Bae et al., 2022). Broadly, data valuation encompasses techniques that try to relate some output of a model with the data it is trained on. While there is growing research that explores different notions of value, the vast majority is focused on supervised learning for i.i.d. data.

We conduct the first comprehensive data valuation study for graph-based models in the semi-supervised transductive setting. Only two recent works have explored data valuation for graphs. Chen et al. (2023) derive closed-form estimate for the leave-one-out influence of nodes and edges using SGC (Wu et al., 2019) as a surrogate model. Chi et al. (2025a) propose the precedence-constrained winter value where nodes are grouped in coalitions based on the graph structure. Our focus is not on proposing new notions, but rather rigorously evaluating existing ones. Our results show that both data Banzhaf (Wang & Jia, 2023) and datamodels (Ilyas et al., 2022) significantly outperform other valuation methods, but neither of these notions has been studied for graphs so far. Given the importance and ubiquity of graph-based models such as graph neural networks (Ju et al., 2024), we hope that our work provides a solid foundation for the study of data valuation on graphs.

It is clear that not all data are created equally, e.g. some instances may be noisy or mislabeled. However, even a "high-quality" instance may not be valuable if there are already similar instances in the dataset,

[1]The code is provided as supplementary material.

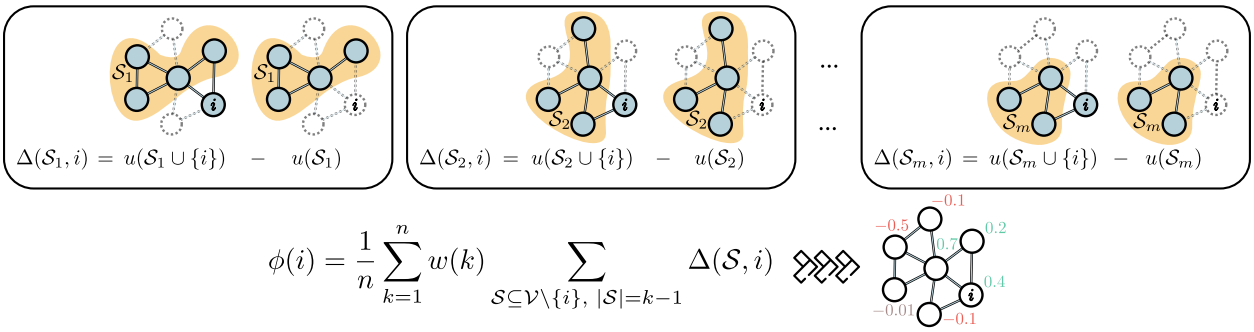

Figure 1: Data valuation assigns importance to a node based on its marginal contribution to the utility across various subsets. When removing the node: if model performance decreases, it is considered informative (green); if performance remains unchanged, the node is irrelevant (grey); if performance improves, the node is misleading (red). Weighting schemes w define different data valuation methods.

indicating that to properly evaluate the worth of an instance we need to carefully consider the appropriate context. Lack of context is why leave-one-out estimation often fails in practice – its focus is on the effect of an instance in isolation. In contrast, recent game-theoretic notions such as data Shapley (Ghorbani & Zou, 2019; Jia et al., 2019b) and other semi-values (Dubey et al., 1981) compute the average marginal contribution of an instance considering all potential subsets of the data. This indeed comes at a significant computational cost, but approximation strategies allow us to compute useful estimates in practice.

The right context is even more important in the graph setting. First, the prediction of a node depends on its neighborhood – nodes are not i.i.d. This suggests that looking at nodes in isolation is likely suboptimal and we should consider different subgraphs. The main idea is illustrated in Fig. 1. To estimate the influence of node i on some utility function $u(\cdot)$ such as the accuracy, we consider the difference in utility $\Delta(\mathcal{S}, i) = u(\mathcal{S} \cup \{i\}) - u(\mathcal{S})$ with and without node i across many subsets $\mathcal{S} \subseteq \mathcal{V}$ of different sizes. These marginal contributions $\Delta(\mathcal{S}, i)$ are merged together using a weighted average to obtain the final node value $\phi(i)$. Different weights recover many of the established data value notions, including data Shapley, beta Shapley, data Banzhaf, and leave-one-out.

Second, all nodes, either labeled and unlabeled, influence both training and inference. We account for this by constructing different variants (§ 3) of each data value notion that aim at disentangling these effects. Unsurprisingly, even though the number of training nodes is a small percentage of all nodes in the sparsely-labeled scenario (Shchur et al., 2018), the top most influential nodes are mostly training nodes. At the same time, we can find unlabeled nodes which have greater influence than many training nodes. This reflects the complex interplay between training and test nodes in graphs.

In addition to influence-based and game-theoretic notions, predictive approaches, such as datamodels (Ilyas et al., 2022) and MLPbV (Wu et al., 2024), construct surrogate (proxy) models that can predict the utility function for any subset without the need to train the model from scratch. They allow us to answer counterfactual questions, i.e. predicting the performance on arbitrary unseen subsets of data. Proxy models are trained on subsets of all possible $\{(\mathcal{S}_i, u(\mathcal{S}_i)\}$ pairs, where $\mathcal{S}_i \subseteq \mathcal{V}$. As we discuss in § 2.1, we can trivially turn any set of data values into a predictive surrogate. Moreover, the resulting data values from predictive and game-theoretic approaches are equivalent for certain configurations (see § 2.2). Thus, this distinction between these types of notions may not be very useful.

Overall, our results show that approaches accounting for subgraphs instead of single-node contribution result in more accurate data values. We demonstrate the usefulness of the data values for various downstream applications: i) *finding highly influential nodes* – pruning nodes with high positive (negative) influence results in a significant drop (rise) in model performance; ii) *spotting brittle predictions* which depend on a small set of support nodes whose removal results in misclassification; iii) *detecting poisoned (mislabeled) data*; iv) *estimating counterfactuals* such as predicting the performance on an arbitrary subset; and v) *visualizations* that provide further insight in the data.

We thoroughly study the problem of data valuation for graphs, including variants that attribute importance to both labeled and unlabeled nodes. We analyse game-theoretic and predictive notions and show that datamodels and data Banzhaf, neither of which was previously considered in the context of graphs, significantly outperform other valuation techniques, including the two recently proposed graph-specific approaches. Our study was computationally expensive (over 2500 compute hours in total) and storage-intensive (more than 10TB of raw data). Nonetheless, we show that one can obtain accurate estimates even with modest computational resources, and we plan to publicly release the raw data.

2 Data valuation

Broadly, data valuation methods try to relate the presence or absence of an instance in the dataset with the resulting change in some user-defined utility function. Traditionally, this is some measure of model performance such as accuracy, where a high positive (negative) value indicates that including the instance in the data improves (deteriorates) performance. For both game-theoretic and predictive approaches the main idea is to consider different subsets of the training set $\mathcal{D}$ and assess the value of missing training samples according to the change in the utility. We denote with $\phi(i) \in \mathbb{R}$ the value assigned to instance i.

Utility. Let $u : 2^{|\mathcal{D}|} \to \mathbb{R}$ be a utility function which maps any subsets $\mathcal{S} \subseteq \mathcal{D}$ to a score indicating the usefulness of the subset. For classification tasks, we commonly have $u(\mathcal{S}) = \mathrm{acc}(\mathcal{A}(\mathcal{S}))$ where $\mathcal{A}$ is a learning algorithm that returns a model $f_{\mathcal{S}}$ trained on $\mathcal{S}$, and acc computes the accuracy on a held-out set. Sometimes we are not interested in the overall performance but rather how the training data influences a specific test point. For example, the utility can be the prediction margin for the test point $\boldsymbol{x}_{\text{test}}$. That is $u(\mathcal{S}) = u(\mathcal{S}, \boldsymbol{x}_{\text{test}}) = \mathrm{margin}(f_{\mathcal{S}}, \boldsymbol{x}_{\text{test}}) = f_{\mathcal{S}}(\boldsymbol{x}_{\text{test}})_{y^*} - \max_{y \neq y^*} f_{\mathcal{S}}(\boldsymbol{x}_{\text{test}})_y$ where $f_{\mathcal{S}}(\cdot)_y$ returns the logit-score or probability for label y, and y^* is the ground-truth label. Any other function of (the outputs of) $f_{\mathcal{S}}$ is valid.

2.1 Categorization of data valuation approaches

Game-theoretic notions. These methods consider different subsets (coalitions) of the training data, treating each data point as a player in a cooperative game. If $u(\mathcal{D})$ represents the outcome of the game, the goal is to fairly distribute the outcome to each data point according to its contribution. This concept is at the core of Shapley values (Shapley, 1953; Ghorbani & Zou, 2019), a well-known framework for assessing the value of data according to the average marginal contribution of adding a data point into all possible subsets, weighted by the number of permutations in which that data point appears.[2] Let $n = |\mathcal{D}|$. The Shapley value of a data point i is given by

$$\phi_{\texttt{SHAP}}(i) = \frac{1}{n} \sum_{k=1}^{n} \binom{n-1}{k-1}^{-1} \sum_{\substack{\mathcal{S} \subseteq \mathcal{D} \setminus \{i\} \\ |\mathcal{S}| = k-1}} \big[u(\mathcal{S} \cup \{i\}) - u(\mathcal{S}) \big] \tag{1}$$

To compute it exactly we need to train $2^{|\mathcal{D}|}$ models, i.e. one for each possible subset. Since this is prohibitively expensive there are various approximation techniques such as Monte Carlo sampling. Shapley values are popular since they uniquely satisfy four axioms that were argued to be necessary to ensure a fair valuation. These are *symmetry*, *linearity*, *null player*, and *efficiency*.

Kwon & Zou (2022) questions the necessity of the *efficiency* axiom, which requires all the values to sum up to the utility on the original dataset, i.e. $\sum_{i \in \mathcal{D}} \phi(i) = u(\mathcal{D})$. Removing this axiom we get the so called *semivalues* (Dubey et al., 1981) which satisfy the other three axioms. As with any axiomatic approach, these axioms can be debated. For the purposes of this paper, we consider them as given. It turns out that all

[2] The initial use of Shapley values in the machine learning community was for feature attribution – determining which features (e.g. pixels) highly influence the prediction. That is, they were used as a post-hoc explainability method. In contrast, Ghorbani & Zou (2019) introduce data Shapley where the goal is to consider the influence of training points instead, which is the setting that we consider in this paper (see also § 5).

semivalues can be written in a canonical form

$$\phi_{\text{semivalue}}(i, w) = \frac{1}{n} \sum_{k=1}^{n} w(k) \sum_{\substack{\mathcal{S} \subseteq \mathcal{D} \setminus \{i\} \\ |\mathcal{S}| = k-1}} \left[u(\mathcal{S} \cup \{i\}) - u(\mathcal{S}) \right] \tag{2}$$

where $w : [n] \to \mathbb{R}$ is a weight function such that $\sum_{k=1}^{n} \binom{n-1}{k-1} w(k) = n$. We can recover Shapley (`SHAP`) with $w(k) = \binom{n-1}{k-1}^{-1}$. Leave-one-out (`LOO`) measures the difference in utility when removing a single data point from the dataset and can be obtained by choosing $w(k) = n\mathbf{1}[k = n]$. Data Banzhaf (`BANZ`) assigns uniform weights $w(k) = \frac{n}{2^{n-1}}$. Weighted Banzhaf (α-`BANZ`) has $w(k) = n\alpha^{k-1}(1-\alpha)^{n-k}$ weights (Li & Yu, 2023).

Predictive notions. Wang et al. (2021) propose to learn a surrogate model on utilities, followed by Ilyas et al. (2022) that introduce the datamodel framework where the idea is to learn a surrogate model $m_{\boldsymbol{\theta}} : \{0,1\}^{|\mathcal{D}|} \to \mathbb{R}$ that can directly predict the utility for any given subset, i.e. $m_{\boldsymbol{\theta}}(\mathcal{S}) \approx u(\mathcal{S})$. The weights $\boldsymbol{\theta}$ of the surrogate are learned from $\{(\mathcal{S}_i, u(\mathcal{S}_i))\}$ where $\mathcal{S}_i$ is encoded as a binary vector $\{0,1\}^{|\mathcal{D}|}$ indicating the presence or absence of an instance. They show that even a simple linear model can approximate the mapping from $\mathcal{S}$ to $u(\mathcal{S})$ well. The data values are given by the weights of the linear model $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{D}|}$, i.e. $\phi_{\texttt{DM}}(i; \boldsymbol{\theta}) = \theta_i$. Wu et al. (2024) propose to instead use a multi-layer perception as a surrogate which can use additional inputs.

Influence functions. The number of possible subsets increases exponentially with the size of the dataset. Moreover, for each subset, we need to train the entire model from scratch. If the dataset is large even computing the leave-one-out error which requires "only" $n = |\mathcal{D}|$ evaluations can be too expensive. In contrast, influence functions (Cook & Weisberg, 1982) aim to approximately compute the change in the model output (or model weights) when removing a training sample. They rely on first and second-order information (gradient, Hessian) to obtain estimates without model retraining. Nevertheless, influence functions perform poorly in non-convex settings as in deep neural networks (Basu et al., 2021; Bae et al., 2022). At best, they perform as well as leave-one-out errors which are suboptimal since they consider instances in isolation. Chen et al. (2023) derive different data values using influence functions for graph neural networks.

From game-theoretic to predictive. Any game-theoretic approach can be made predictive by assuming that the values are the weights of an implicit linear model (without a bias term). This implies that the predicted value of any unseen subset equals the sum of the data values of the nodes in the subset, i.e. $u(\mathcal{S}) \approx \sum_{i \in \mathcal{S}} \phi(i)$ for any $\phi(i)$. This is a reasonable assumption given the linearity axiom of semivalues, and it's explicitly made by datamodel values (Ilyas et al., 2022). To compute both `DM` and α-`BANZ` we sample subsets where each node is included with probability α.

2.2 Properties of data valuation approaches

Connections between different values. If $\alpha = 0.5$ and we use no regularization when fitting $m_{\boldsymbol{\theta}}$ the datamodel values are equivalent to Banzhaf values since the Banzhaf values are the best linear approximation to u in terms of least square loss (Wang & Jia, 2023; Hammer & Holzman, 1992). Lin et al. (2022) prove that if we sample $\alpha \sim \text{Unif}(0,1)$ rather than having it fixed, then the optimal weights of a certain regularized linear model converge relatively quickly to the Shapley values as we increase the number of samples.

Limitations of data values. The utility function can be viewed as a vector in $\mathbb{R}^{2^n}$ which is transformed into a vector of data values $\boldsymbol{\phi} \in \mathbb{R}^n$. This mapping is not injective, so there are different utility functions that yield identical values. A natural question is which utility functions are well approximated by data values. Wang et al. (2024) show that a sufficient condition is for the utility to be a monotonically transformed modular function. In general, data values may be no better than a random baseline for dataset selection. They also generalize the result from Saunshi et al. (2022) which states that the residual error of the unregularized linear approximation equals to the sum of all u's Fourier coefficients of order 2 or higher. Nonetheless, data values work well in practice. This is remarkable since they approximate $u(\mathcal{S})$ which involves training of a model on $\mathcal{S}$ from scratch (often a complicated neural network) and then computing some function of the model outputs.

Robustness and efficiency. In our context the utility function is stochastic due to the randomness in the training process, making the influence of a data point a random variable with an expectation and a

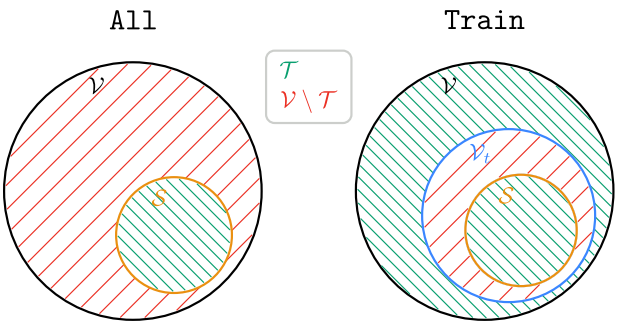

Figure 2: Venn diagram of subsets for the induced subgraph. Downward hatches from left to right indicate the final subset $\mathcal{T}$.

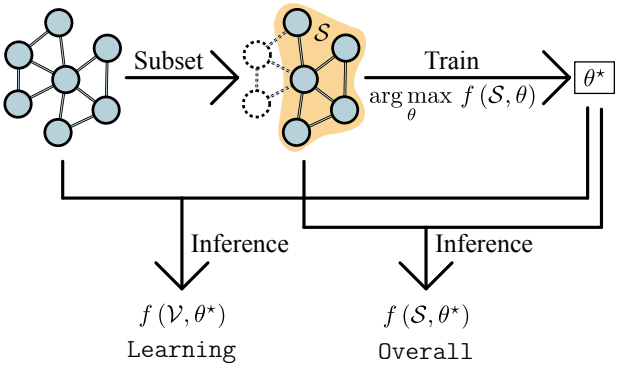

Figure 3: Pipeline for the `learning` vs. `overall` signals: train on subset $\mathcal{S}$, then measure utility either on $\mathcal{S}$ or $\mathcal{V}$.

variance (Nguyen et al., 2024). Thus, using a single sample of $u(\mathcal{S})$ may be unreliable. Wang & Jia (2023) define the safety margin – the largest noise that a semivalue can tolerate without altering the ranking of any distinguishable pair of data points. They prove that the data Banzhaf achieves the largest safety margin among all values. This property can be attributed to the uniform weights since $\phi_{\texttt{BANZ}}(i) = \mathbb{E}_{\mathcal{S}\sim\text{Unif}(2^{n\setminus i})}[u(\mathcal{S}\cup\{i\}) - u(\mathcal{S})]$. The standard Monte Carlo estimator directly samples $\mathcal{S}$ from $\text{Unif}(2^{n\setminus i})$ but needs to be repeated n times once for each i. Wang & Jia (2023) propose an alternative estimator where given samples $\mathcal{M} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ i.i.d. from $\text{Unif}(2^n)$ it computes $\phi_{\texttt{BANZ}}(i) = \frac{1}{|\mathcal{S}_{\in i}|}\sum_{\mathcal{S}\in\mathcal{S}_{\in i}} u(\mathcal{S}) - \frac{1}{|\mathcal{S}_{\notin i}|}\sum_{\mathcal{S}\in\mathcal{S}_{\notin i}} u(\mathcal{S})$ as the difference of the average utility of the sets $\mathcal{S}_{\in i} \subseteq \mathcal{M}$ that contain i, minus the sets $\mathcal{S}_{\notin i} \subseteq \mathcal{M}$ that do not contain it. Now, maximal sample reuse (MSR) is achieved since all evaluations of $u(\cdot)$ are used in the estimation of all $\phi_{\texttt{BANZ}}(i)$ values. The existence of an efficient MSR estimator is a unique to the Banzhaf value among all semivalues. Liu et al. (2024) alos show that the latter can be approximated with a linear regression formulation. Nonetheless, we apply the same idea of Li & Yu (2023) and use α-`BANZ` where instances in the subsets are sampled with a fixed probability α (rather than uniform). In our experiments, α-`BANZ` outperforms `BANZ` and shows strong results across all settings.

3 Data valuation on graphs

Semi-supervised setting. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Let $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ be the matrix of d-dimensional node features, $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ be the adjacency matrix and $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{V}|}$ be the vector of labels. Let $\mathcal{V}_\ell$ and $\mathcal{V}_u$ be the subsets of labeled and unlabeled nodes respectively, and let $\mathcal{V}_\ell$ be further split into training nodes $\mathcal{V}_t$ and validation nodes $\mathcal{V}_v$. Usually, $\mathcal{V}_\ell$ is a small subset of nodes while the majority is in $\mathcal{V}_u$ (Shchur et al., 2018). In a transductive semi-supervised learning setting, the model is exposed to both $\mathcal{V}_\ell$ and $\mathcal{V}_u$ during the training, such that it can propagate label information from the few labeled nodes to the many unlabeled nodes leveraging the underlying graph connectivity and node features.

Graph Neural Networks (GNNs). In each layer k of a GNN, the hidden representation $\boldsymbol{h}_v^{(k)}$ of a node v is the result of aggregating information from its neighborhood $\mathcal{N}(v)$ and its own representation from the previous layer. Many popular GNNs can be succinctly written in matrix notation as $\boldsymbol{H}^{(k)} = \sigma\left(\boldsymbol{S}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$, where σ is the non-linearity, $\boldsymbol{W}^{(k)}$ are trainable parameters, $\boldsymbol{S}$ is the spatial graph convolution operator and $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. For example, in GCN (Kipf & Welling, 2017) we use the degree normalized adjacency matrix $\boldsymbol{S} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$, where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_{|\mathcal{V}|}$ and $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$ is the degree matrix. SGC (Wu et al., 2019) uses the same architecture as GCN without the non-linearity resulting in a model that is linear w.r.t. the weights.

Training nodes vs. All nodes. In the transductive setting, all nodes influence learning. The training nodes do so directly since the loss is computed using their ground-truth labels as supervision signal. The remaining nodes do so indirectly since they influence the hidden representation of the training nodes via the

graph structure. This means that in addition to investigating the value of the training nodes $\mathcal{V}_t$ as in the standard i.i.d. setting, we can also compute the value of all nodes $\mathcal{V}$. We refer to these two settings as `train` and `all` respectively. Let $\mathcal{D}$ be the set of considered nodes, namely either $\mathcal{D} = \mathcal{V}_t$ or $\mathcal{D} = \mathcal{V}$, for the `train` or `all` setting respectively. Recall that to compute data values we consider different subsets $\mathcal{S} \subseteq \mathcal{D}$. We first remove the nodes that are in $\mathcal{D}$ but not in $\mathcal{S}$, and then we train a GNN on the graph induced by the remaining nodes $\mathcal{T} = \mathcal{V} \setminus (\mathcal{D} \setminus \mathcal{S})$. In the `train` setting $\mathcal{T} = \mathcal{V} \setminus (\mathcal{V}_t \setminus \mathcal{S})$ and in the `all` setting $\mathcal{T} = \mathcal{V} \setminus (\mathcal{V} \setminus \mathcal{S}) = \mathcal{S}$. Fig. 2 shows Venn diagram representations of the considered subgraphs in the two settings. Since all nodes influence learning we focus on the `all` setting in the main paper. However, `train` results are in § C.2.

Learning signal vs. Overall signal. After training, we need to evaluate our utility function. We consider the *accuracy* to measure the influence on the final performance, and the *prediction margin* to measure influence w.r.t. any individual node (see § 2). Here we assume access to ground-truth test labels since our goal is to understand how nodes influence learning for different models. This is standard in the data valuation literature. In practical scenarios, one can use validation labels instead. Importantly, after training on the graph induced by $\mathcal{T}$ we can compute the utility in two different ways which provide different and complementary insights. In particular, we can compute the utility on the same induced subgraph $\mathcal{T}$ which captures the `overall` influence of a node since removed nodes are not present during inference. Alternatively, we can compute the utility on the whole graph $\mathcal{V}$ – that is we bring back the removed nodes $\mathcal{V} \setminus \mathcal{T}$ which only captures their influence during learning. These `learning` data values measure the training signal provided by a node, while the `overall` data values measure the influence during training and inference. This is in contrast to the i.i.d. setting where we can only measure the train signal provided by an instance. Fig. 3 depicts the pipelines for computing the values in the two described settings.

4 Experimental results and analysis

Setting. As representative game-theoretic notions, we consider the three well-known semivalues: leave-one-out (`LOO`), data Shapley (`SHAP`), and weighted Banzhaf values (α-`BANZ`), where $\alpha = 0.5$ recovers the original Banzhaf value. As a representative predictive notion, we consider datamodels (`DM`). Finally, we consider the precedence-constrained winter value (`PCW`) as the only data value custom-designed for graphs since the method by Chen et al. (2023) approximates `LOO`, which we already cover. We introduce a sampling improvement to `PCW` which we call `PCWP` (see § D.2.1 for details). We include two more baselines, namely degree (`DEG`) where a node value corresponds to its degree, and randomly assigning node values (`RND`). We test these methods on the largest connected component (LCC) of different citation graphs (`Citeseer`, `CoraML`, `PubMed`, and `CoPhysics`) and co-purchase graphs (`Photo` and `Computers`). We consider three models SGC, GCN and GAT. Details about datasets, models, and training as well as running times are provided in § D. Throughout our analysis, we assume that a node's value depends on both its features and its edges, since we completely remove any node $v \notin \mathcal{S}$.

Key findings. We summarize our results in five key findings, each supported by one or more experiments.

I. Nodes exert positive influence within their class and negative influence across classes.

We study how every node influences every other node using prediction margins as utilities. We obtain a matrix of values $\mathbf{\Phi} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{D}|}$, where a column j of $\mathbf{\Phi}$ shows how node j influences other nodes. A positive value $\mathbf{\Phi}[i, j]$ means that adding node j to the graph increases the margin of node i, while a negative value means that the margin decreases. On Fig. 4, we show a heatmap of the entire matrix $\mathbf{\Phi}$ (`overall` signal), where we also annotate training nodes and highly influential nodes (outliers) with bigger markers. Since the rows/columns are sorted by both class and degree, the block-diagonal pattern with positive diagonal and negative off-diagonal entries, leads to the conclusion stated in Finding I. The results are consistent for different methods, and we observe no qualitative difference between `learning` and `overall` values (§ B.3).

In the average per-class influence plot on Fig. 4 (top right), we summarize $\mathbf{\Phi}$ by computing the average influence between nodes from each pair of classes. The average influence is positive between nodes of the same class and negative between different classes. As expected, including more within-class nodes increases the margin. Interestingly, the size of a community (the number of nodes from the same class) is highly

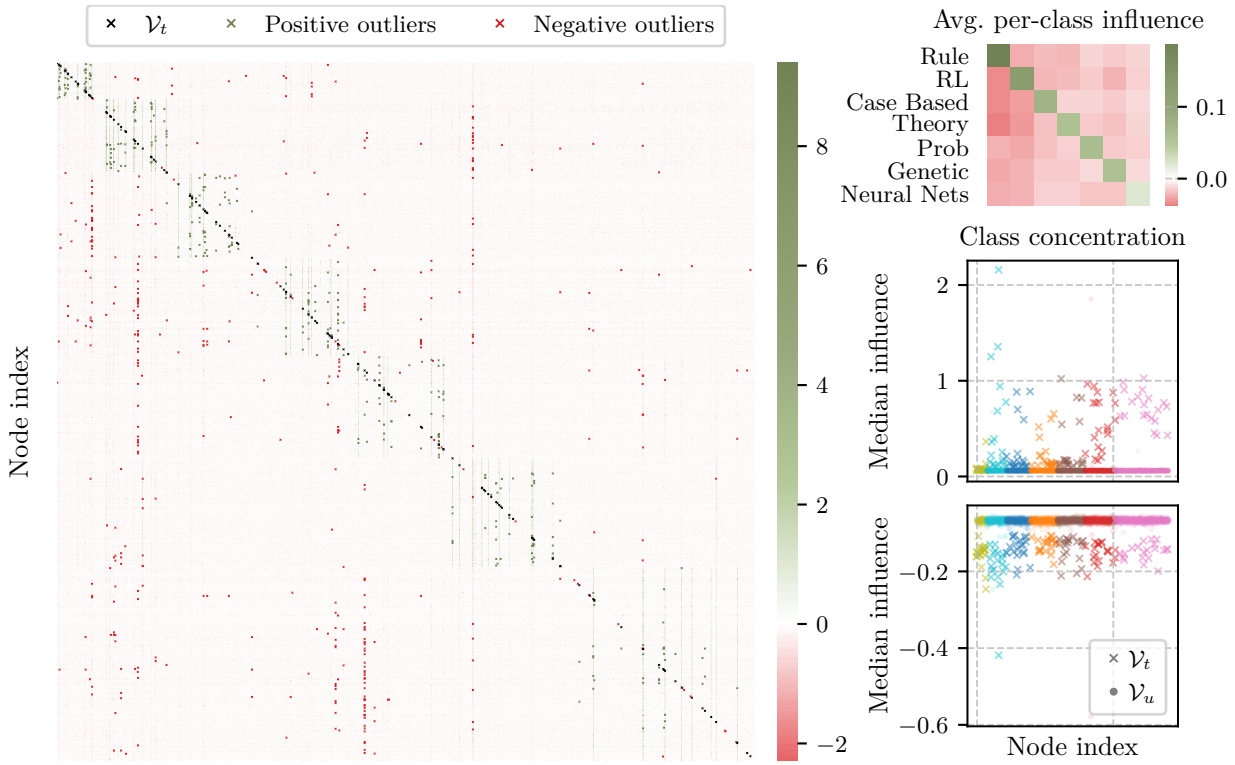

Figure 4: (Left) Heatmap of $\mathbf{\Phi}$, where each cell shows the positive (green) or negative (red) effect of adding node j on node i's margin. (Top right) Average per-class influence confirms positive intra-class and negative inter-class effects. (Bottom right) Median influence plots (nodes colored by class) reveal that a few (training) nodes dominate influence (vertical stripes in $\mathbf{\Phi}$).

correlated with the average within-class influence. Intuitively, adding a node to a larger community has a smaller effect since there is redundancy, i.e. there are many other nodes that serve an equivalent role. This effect is consistent for all datasets (see § B.3). We can also see that the influence is not symmetric, e.g. the nodes from the *Theory* class have a larger influence on the *Rule* class than the other way around. Moreover, the smallest communities have relatively higher negative between-class influence compared to larger communities. This is warrants further study.

II. Influence is concentrated and synchronized within each class.

Concentrated. A small subset of nodes (mostly training nodes) is highly influential. This can be easily seen from the $\mathbf{\Phi}$ heatmap on Fig. 4 where a few columns have large positive and negative values across many rows, visible as vertical strips. In the class concentration subplots on Fig. 4 (right), we show the median influence of a node (median per column of $\mathbf{\Phi}$) and split them in positive in the top plot and negative in the bottom plot. We color the nodes by class and mark them as training (crosses) and non-training (circles). The results confirm that most of the influence (either positive or negative) *concentrates* among training nodes and a few of those have a high median value highlighting it is influential for many nodes. Since we sorted $\mathbf{\Phi}$ by class and degree, we

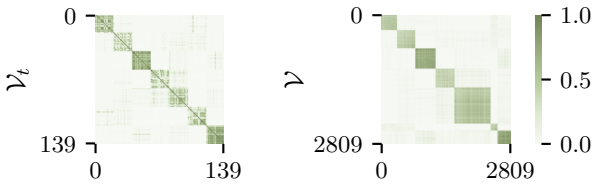

Figure 5: Pairwise R^2 for $\mathbf{\Phi}$ columns (left) shows same-class training nodes ($\mathcal{V}_t$) affect others similarly; for rows (right), same-class nodes ($\mathcal{V}$) also share influence.

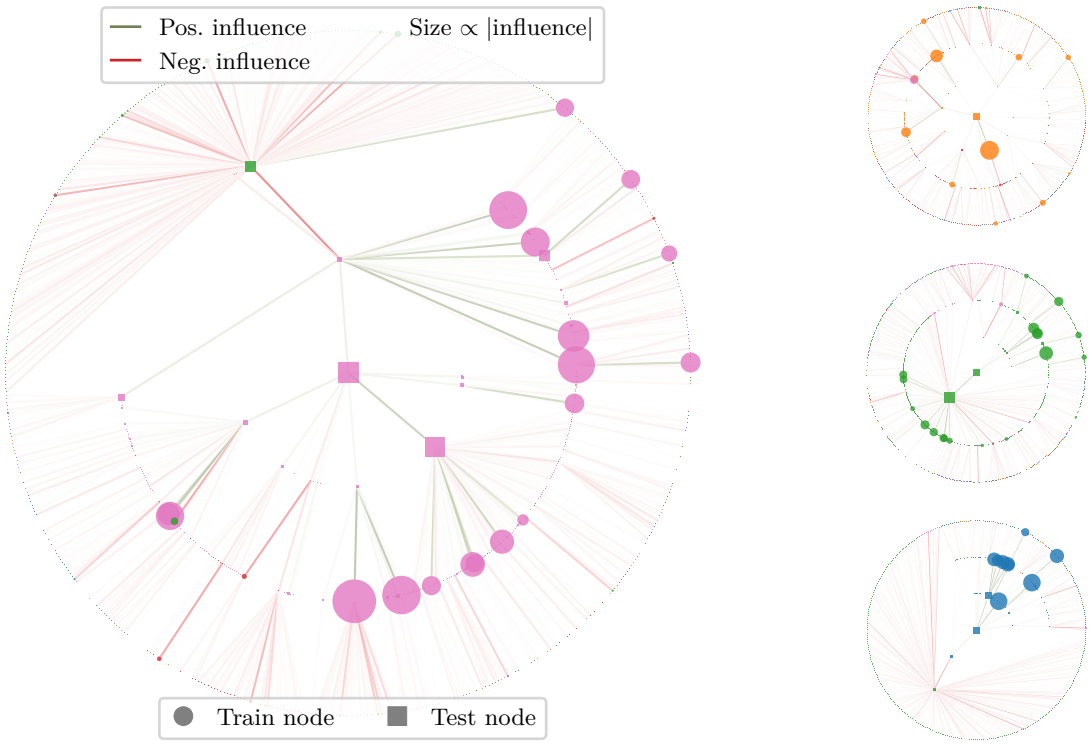

Figure 6: Two-hop breadth-first trees are rooted at selected *test* nodes (center of each radial graph). Nodes are class-colored, sized by their $\mathbf{\Phi}$-based influence on the root, and edges show sign (green for positive, red for negative). Training nodes (circles) are typically most influential, but some test nodes (squares) stand out, often when highly connected to same-class training nodes.

can actually see how the degree plays an important role for the positive influential nodes while less important for the negative ones.

Synchronized. We additionally see high correlation between many pairs of columns. To study this further, we compute the pairwise R^2 coefficient of entries in $\mathbf{\Phi}_{\texttt{DM}}$ (either rows or columns) in the `train` setting. From the block-diagonal structure on Fig. 5 (left) we see that any pair of training nodes i and j from the same class *influence* all other nodes in the same way since the R^2 score between $\mathbf{\Phi}[:, i]$ and $\mathbf{\Phi}[:, j]$ is high. Similarly, on Fig. 5 (right) we also see that any pair of same-class nodes *are being influenced* in the same way by training nodes as show by high R^2 score between $\mathbf{\Phi}[i, :]$ and $\mathbf{\Phi}[j, :]$. Since nodes within each class both exert and receive influence in similar ways, we refer to this effect as synchronization (see § B.4 for more settings).

III. Training nodes are most influential, but a few test nodes stand out.

We look at the most positively and negatively important nodes in the two-hop neighbors (to match with the receptive field of a two-layers GNN) of a selected test node. On Fig. 6 we generate a breadth-first tree from a given test node. We color nodes by their classes, resize them according to the absolute value of their influence w.r.t. the root, and shape them as circles if they are training nodes or squares otherwise. We color edges green if they positively influence the root and red otherwise. We see that the most important (positive) nodes are usually training ones (large circles). However, we can find important test nodes (large squares) which are more relevant than other training nodes. These are usually test nodes connected to training nodes from the same class. The class concentration plots in Fig. 4 also confirm this insight (compare crosses to dots). This emphasizes the importance of looking at both training and unlabeled nodes within the graph

domain. In § B.2 we rank nodes according to their values and show how the approaches estimating the more accurate values, rank few non-training nodes earlier than training ones.

IV. Data values are excellent at detecting brittle and poisoned nodes.

Brittle. The support of a node v is the minimum set of nodes such that if removed, the node v is misclassified. This concept was introduced by Ilyas et al. (2022) for images. Using $\mathbf{\Phi}[v, :]$ we can estimate nodes that have high influence on v, which in turn allows us to estimate the support. We follow Alg. 1 from Ilyas et al. (2022), using different approaches to estimate the top-k nodes and the support in the `all` setting. On Fig. 7 (right) we see that, regardless of the considered signal (`learning` at the top of the right-hand column, and `overall` at the bottom) most nodes are brittle – more than half of the nodes (y-axis) have a support size of 15 or less (x-axis), and by just removing them we can misclassify more than 50% of nodes. Computing the exact support is intractable and each data value gives us an upper bound. Choosing the best upper bound among all values we arrive at the best estimate marked with a black dash line (ensemble guess). Again we see that `DM` and α-`BANZ` are closest to this estimate, and are the only two approaches reaching a reasonable estimate in the `overall` setting where all the other fail. In § C we include results for the `train` setting as well, where we can draw similar conclusions.

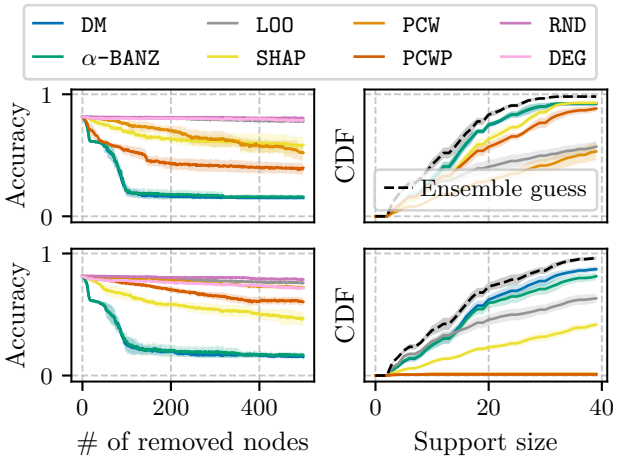

Figure 7: Sampling reuse approaches (`DM` and α-`BANZ`) show more consistent performance in both estimating values (left) and predicting supports (right).

Poisoned. $\mathbf{\Phi}[v, :]$ can also be used to spot corrupted instances. When gathering data from external sources, some instances may be corrupted intentionally (poisoning) or unintentionally (mislabeling). The rank of node values can help detect such instances. Namely, a high data value of a labeled node for itself (self-importance) indicates memorization. When a training node v has a wrong label and $v \in \mathcal{S}$ the model must memorize it to achieve low training loss. If $v \notin \mathcal{S}$ then the model's prediction does not match the wrong label. This leads to a large difference $u(\mathcal{S} \cup \{v\}) - u(\mathcal{S})$, and thus a large value. To test this, we poison 10% of the training data and compute the node values using the margins of the nodes themselves as the utility. We consider the `learning` signals in the `train` setting. In Fig. 9, we show the percentage of poisoned data for `CoraML` appearing in top-k ranked nodes as k increases, with `DM` and α-`BANZ` consistently detecting poisoned data better than the others. We provide more details in § C.4.

The brittleness experiment emphasizes the sensitivity of GNNs to structure perturbation. However, as confirmed by the poisoning experiments, data values can be leveraged as tools, complementary to adversarial robustness, useful for analysing model's behaviors.

V. Sample reuse is a critical feature for data valuation approaches to achieve good performance.

To evaluate the quality of node values, we can again use their rank. Nodes with high positive value should have positive influence on the model, while nodes with high negative value should have the opposite effect. A standard analysis is to then remove (or add) nodes according to the rank and observe the change in performance. Fig. 8 shows the results for removing high-ranking nodes (`all`, `learning` signal, test accuracy as utility) across datasets with a GCN model. On Fig. 7 (left) a GAT model is used instead for computing both `learning` (top) and `overall` (bottom) signals with test margins as utility (see § D.1 for details on how to compute nodes ranking from margins). We expect a steep decline in the performance as we remove the first most important nodes and then a plateau when removing non-influential nodes. We stop after removing

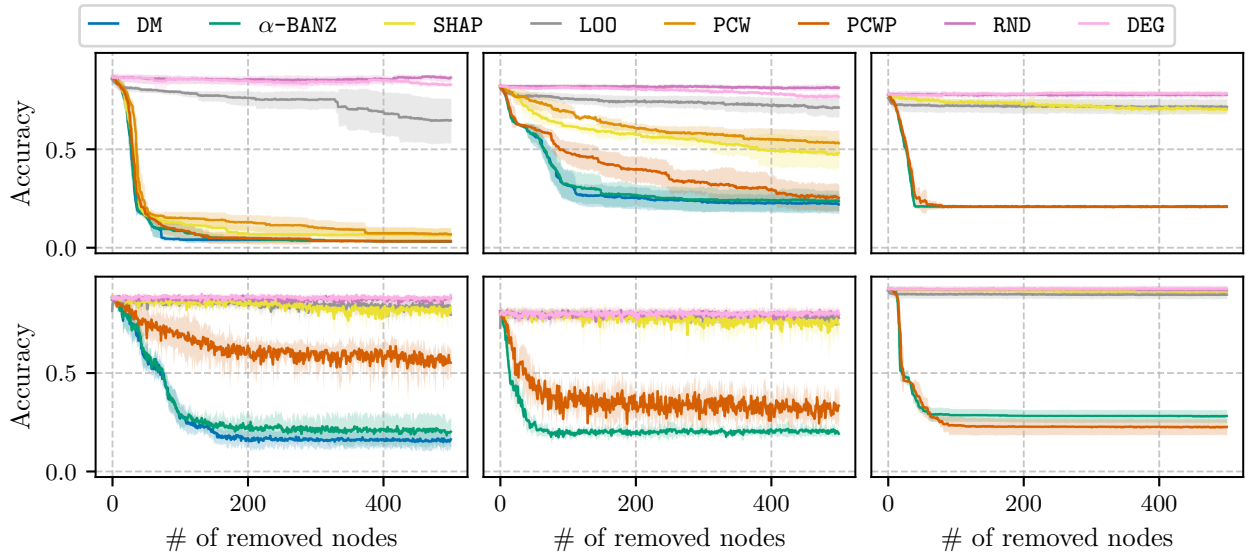

Figure 8: Most influential node removal: `Citeseer`, `CoraML` and `PubMed` (first row), and `Photo`, `Computers` and `CoPhysics` (second row). Pruning high-value nodes leads to a performance (test accuracy) drop, revealing the quality of estimated values. `DM` and α-`BANZ` consistently show the steepest declines across datasets.

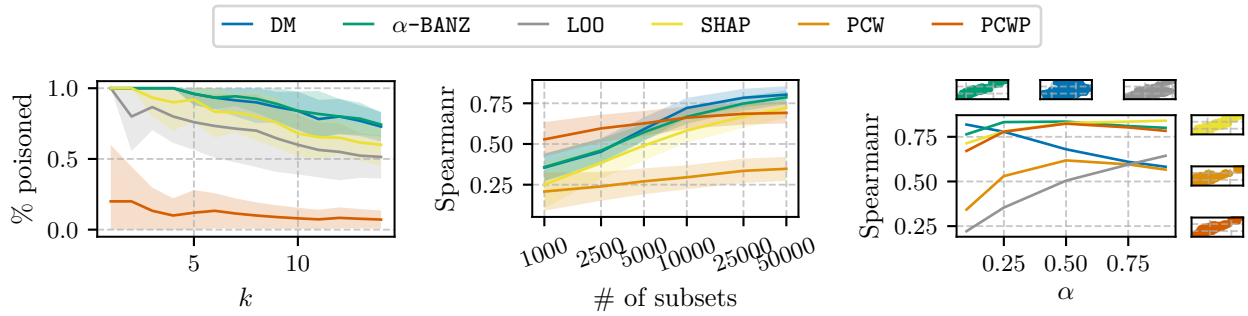

Figure 9: Ratio of poisoned nodes among top-k ranks, with `DM` and α-`BANZ` identifying most of them.

Figure 10: An increasing number of subsets leads to better predictive performance (LDS).

Figure 11: LDS on counterfactual subsets. Insets show predicted vs. true utility correlation at $\alpha = 0.5$.

500 nodes since we are only interested in the initial performance drop. Overall, `DM` and α-`BANZ` consistently show strong results and have the steepest decline in performance. `PCWP` performs worse, but better than `PCW` and `SHAP`. We omit `DM` on some of the large datasets due to memory issues. `LOO` and `SHAP` converge to the simple `RND` and `DEG` baselines on larger graphs. In contrast, `PCWP` seems to work better on larger graphs (e.g. `PubMed` and `CoPhysics`) but still lags behind `DM` and α-`BANZ`. However, the gap between the methods is emphasized in the `overall` setting. Here the valuation methods have "less information" since if node $i \notin \mathcal{S}$ the node is not available during training or inference. In § C we include many additional experiments with all variants, including node addition. The conclusions are similar.

Park et al. (2023) introduce the *linear datamodeling score* (LDS) defined as the Spearman correlation between the predicted utility and the true utility. The idea is that a good (predictive) valuation method should accurately approximate the utility on held-out subsets. Fig. 10 shows the improvement in the LDS as increasing the number of subsets when the held-out set comes from the same distribution (same $\alpha = 0.1$ for both computing and evaluating the data values) for `DM` and α-`BANZ`. Fig. 11 shows instead the LDS performance

when the distribution of the held-out dataset changes while keeping $\alpha = 0.1$ fixed for computing the data values. From the former, we can see how from already just 10 000 subsets `DM` and α-`BANZ` achieve reasonable results. From the latter instead, α-`BANZ` generalizes better across the counterfactual sets, followed by `SHAP` and `PCWP`. On the other hand, `DM` becomes worse for a larger distribution shift (larger α).

Generally, we notice consistent good performance in approaches taking advantage of the sampling reuse (Fig. 7 to Fig. 10). The data value of an instance is updated leveraging information from all the subsets and not only from the ones where the node is included. Indeed, both `DM` and α-`BANZ` take advantage of the sampling reuse trick (we study further the limitations of game-theoretic approaches in the context of graphs in § A).

Discussion. Finding I confirms that nodes tend to reinforce intra-class predictions while suppressing inter-class ones, consistent with the inductive bias of GNNs on homophilic graphs. However, the extent of this influence is shaped by community size: smaller communities exert disproportionately stronger negative cross-class influence, while larger ones exhibit redundancy, diminishing their per-node impact. We find this phenomenon equally compelling for i.i.d. settings and leave its exploration to future work. Moreover, observed asymmetries in cross-class influence (e.g., *Theory* influencing *Rule* more than vice versa in Fig. 4) suggest that connectivity patterns amplify directional influence.

Interestingly, Finding II shows that a handful of training nodes dominate the value spectrum. The synchronization of influence within classes evidenced by high pairwise R^2 scores in Fig. 5, implies that nodes behave as collective units rather than individuals. This redundancy suggests opportunities for potential pruning, but also vulnerabilities: poisoning a single high-value node (e.g., a hub in Fig. 6) can propagate errors across neighbors due to GNNs' aggregation. However, a node value can be used to detect such vulnerabilities (Finding IV).

Finding III challenges the i.i.d. dichotomy between training and test data. Unlabeled nodes, through structural proximity to labeled hubs, act as latent training proxies. This hints to potential data curation strategies: one could prune structurally irrelevant unlabeled nodes without hurting accuracy.

Finally, Finding V reveals that sample reuse is the key to accurate data valuation on graphs as well – Li & Yu (2024) observed this for the i.i.d. setting. Indeed, methods like `DM` and α-`BANZ` outperform approaches ad-hoc for graphs (e.g., `PCWP`) by efficiently repurposing subset evaluations, even without neighborhood constraints. While this remains interesting, it hints to structure-aware MSR techniques benefiting from both adaptive subgraph sampling and statistical sampling reuse.

Together, these findings position node-level data valuation not just as a diagnostic tool but as a lens to rethink GNN design: mitigating brittleness via value-aware robust training, exploiting synchronization for efficient learning, and redefining the role of unlabeled data. Bridging game-theoretic principles with graphs, we hope to spark research in this direction with the goal of developing more interpretable, efficient, and robust GNNs.

5 Related works

In the literature, data valuation has been adopted mainly in an i.i.d. setting to study relations between training data and changes in the model's utility or to support prediction's explanations at inference time. This section presents an overview of the works that are closest to ours.

Data valuation in the i.i.d. setting. The seminal works by Ghorbani & Zou (2019); Jia et al. (2019b) extends Shapley value from feature-level to data point granularity to assess training data importance. As this may result in a computationally expensive procedure, Jia et al. (2021; 2019a) compare the utility of different data attribution methods and propose a fast estimator for Shapley values based on a k-nearest neighbors surrogate for which the exact Shapley values can be efficiently computed (Jia et al., 2019a). A more recent research direction uses linear surrogates to learn a mapping from a subset of the training set to the model's utility (Ilyas et al., 2022; Wang et al., 2021). The surrogate is then used to estimate the utility from newly sampled subsets and the surrogate weights represent the data values. As the number of possible training subsets is exponential w.r.t. the number of training data points, the learning of the surrogate may be time-consuming. In a follow-up work, Park et al. (2023) employ Taylor approximation to linearize a considered deep neural network such that the one-step Newton approximation can be applied in a non-linear setting. They show that in such a way, just a few training subsets are needed to attribute importance to

training data accurately. Ultimately, Wang & Jia (2023) introduce the concept of data Banzhaf for assessing the values of training data and showing its robustness in differentiating data. Except for data Shapley, these approaches have not yet been applied to the graph domain. In our work, we incorporate and compare all these approaches to investigate their behaviour when structural information comes into play. Further, Sundararajan et al. (2020) introduce the Shapley-Taylor interaction index, a generalization of the Shapley value to feature interactions of bounded order. Following this work, Tsai et al. (2023) propose Faith-Shap, a variant satisfying the standard Shapley axioms via a least-squares formulation, while Fumagalli et al. (2023) develop SHAP-IQ, a unified estimator applicable to any cardinal interaction index. Also, Li & Yu (2024) propose an estimator based on maximum sample reuse to approximate any probabilistic value, notably improving convergence for Shapley-like attributions. Although they address the computational bottlenecks of subset sampling, their evaluation is limited to feature attribution. We leave the investigation of interactive indices in the data valuation setting as future work. Most recently, Liu et al. (2024) introduced Kernel Banzhaf, which frames Banzhaf value estimation as a linear least-squares regression problem with a pair subset sampling procedure that considers the complement of each subset, improving the value estimation.

Data valuation for graph-structured data. Graph data valuation relating data to model predictions is still under investigation. As far as we are concerned, only two works try to tackle this direction. Chen et al. (2023) adapt influence functions to a graph-structured model to approximate the leave-one-out training. However, they consider only training nodes in a transductive setting, ignoring in such a way the interactions between unlabeled and labeled nodes. Differently from them, we consider a more comprehensive picture by studying the transductive scenario, looking at the influence of all nodes, whether labeled or not. Instead, Chi et al. (2025a) propose a new coalition sampling based on the Winter value that considers the graph's structure when creating a permutation of nodes to process and estimate each node's contribution. However, this work focuses on the inductive scenario where unlabeled nodes do not play a role during the model training.

Graph data valuation at inference time. When applied at inference time, a data valuation approach assesses the most important input patterns that mostly influence the final model's prediction (feature or node/edge/subgraph attribution). Particularly for graphs, this translates into highlighting the graph structure or node features that cause the prediction for the input. For instance, Duval & Malliaros (2021) compute graph structure and node feature explanations for a single example by constructing a surrogate model on a perturbed graph and computing Shapley values as explanations. Akkas & Azad (2024) explain predictions by computing Shapley values edge-wise and outputting the subgraph with edges from the top-k values. Chhablani et al. (2024) use Banzhaf value in combination with thresholded utility functions on edges to provide counterfactual explanations. Bui et al. (2024) introduce the Myerson-Taylor index which includes structure information inside the Shapley value to identify important motifs for the prediction. Finally, Chi et al. (2025b) introduce a structure-aware Shapley value utility by leveraging the precedence-constrained sampling from `PCW` (Chi et al., 2025a), adapted to the inference setting, and combines it with transferable feature extraction and a Shapley-guided optimization procedure tailored to structure-aware utility learning. Differently from the mentioned works, we employ data valuation to discover relations between nodes involved in the training and the final output of the model. Namely, we do not treat the GNN as a fixed black-box function.

6 Conclusion and further research

We present the first extensive study of data valuation methods for graph-structured data. We introduce different data valuation scenarios, and apply state-of-the-art data valuation approaches that have not yet been investigated in the context of graphs. Our results demonstrate that these methods significantly outperform the latest efforts in graph data valuation across multiple applications. We also show how different utility functions can enable several applications. We summarize our results in five key findings that emphasize properties of different methods. Due to the need to train numerous models on different subsets, our study was computationally intensive, and more effort is needed to develop more efficient data valuation methods for node values (both in terms of resources and time efficiency) in particular for large graphs. Future directions also include exploring alternative proxies for datamodels that account for graph structures and a deeper investigation into settings where nodes are simply unlabeled rather than removed.

References

Selahattin Akkas and Ariful Azad. Gnnshap: Scalable and accurate gnn explanation using shapley values. In *Proceedings of the ACM on Web Conference 2024*, WWW '24, pp. 827–838, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645599. URL https://doi.org/10.1145/3589334.3645599.

Juhan Bae, Nathan Hoyen Ng, Alston Lo, Marzyeh Ghassemi, and Roger Baker Grosse. If influence functions are the answer, then what is the question? In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=hzbguA9zMJ.

Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xHKVVHGDOEk.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Ngoc Bui, Hieu Trung Nguyen, Viet Anh Nguyen, and Rex Ying. Explaining graph neural networks via structure-aware interaction index. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4854–4883. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bui24b.html.

Zizhang Chen, Peizhao Li, Hongfu Liu, and Pengyu Hong. Characterizing the influence of graph elements. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=51GXyzOKOp.

Chirag Chhablani, Sarthak Jain, Akshay Channesh, Ian A. Kash, and Sourav Medya. Game-theoretic counterfactual explanation for graph neural networks. In *Proceedings of the ACM on Web Conference 2024*, WWW '24, pp. 503–514, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645419. URL https://doi.org/10.1145/3589334.3645419.

Hongliang Chi, Wei Jin, Charu C. Aggarwal, and Yao Ma. Precedence-constrained winter value for effective graph data valuation. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=tVRVEOOAyb.

Hongliang Chi, Qiong Wu, Zhengyi Zhou, and Yao Ma. Shapley-guided utility learning for effective graph inference data valuation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=8X74NZpARg.

R.D. Cook and S. Weisberg. *Residuals and Influence in Regression*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1982. ISBN 9780412242809. URL https://books.google.de/books?id=MVSqAAAAIAAJ.

Pradeep Dubey, Abraham Neyman, and Robert James Weber. Value theory without efficiency. *Mathematics of Operations Research*, 6(1):122–128, 1981. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/3689271.

Alexandre Duval and Fragkiskos Malliaros. Graphsvx: Shapley value explanations for graph neural networks. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2021.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. DsDm: Model-aware dataset selection with datamodels. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12491–12526. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/engstrom24a.html.

Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. SHAP-IQ: Unified approximation of any-order shapley interactions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=IEMLNF4gK4.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9525–9587. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ilyas22a.html.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proc. VLDB Endow.*, 12(11):1610–1623, July 2019a. ISSN 2150-8097. doi: 10.14778/3342263.3342637. URL https://doi.org/10.14778/3342263.3342637.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In Kamalika Chaudhuri and Masashi Sugiyama (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1167–1176. PMLR, 16–18 Apr 2019b. URL https://proceedings.mlr.press/v89/jia19a.html.

Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8239–8247, June 2021.

Joblib Development Team. Joblib: running python functions as pipeline jobs. https://joblib.readthedocs.io/, 2020. Software library.

Wei Ju, Siyu Yi, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, Xinwang Liu, Xiao Luo, Philip S. Yu, and Ming Zhang. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges, 2024. URL https://arxiv.org/abs/2403.04468.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/koh17a.html.

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8780–8802. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/kwon22a.html.

Weida Li and Yaoliang Yu. Robust data valuation with weighted banzhaf values. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=u359tNBpxF.

Weida Li and Yaoliang Yu. One sample fits all: Approximating all probabilistic values simultaneously and efficiently. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=AUg9D2VjcF.

Jinkun Lin, Anqi Zhang, Mathias Lécuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. In *International Conference on Machine Learning*, pp. 13468–13504. PMLR, 2022.

Yurong Liu, R Teal Witter, Flip Korn, Tarfah Alrashed, Dimitris Paparas, Christopher Musco, and Juliana Freire. Kernel banzhaf: A fast and robust estimator for banzhaf values. *arXiv preprint arXiv:2410.08336*, 2024.

Elisa Nguyen, Minjoon Seo, and Seong Joon Oh. A bayesian approach to analysing training data attribution in deep learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Mądry. Trak: attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. In *The Eleventh International Conference on Learning Representations*, 2022.

Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, Princeton, 1953.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The shapley taylor interaction index. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9259–9268. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sundararajan20a.html.

Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Faith-shap: The faithful shapley interaction index. *Journal of Machine Learning Research*, 24(94):1–42, 2023. URL http://jmlr.org/papers/v24/22-0202.html.

Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6388–6421. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/wang23e.html.

Jiachen T Wang, Tianji Yang, James Zou, Yongchan Kwon, and Ruoxi Jia. Rethinking data shapley for data selection tasks: Misleads and merits. *arXiv preprint arXiv:2405.03875*, 2024.

Tianhao Wang, Yu Yang, and Ruoxi Jia. Learnability of learning performance and its application to data valuation. *CoRR*, abs/2107.06336, 2021. URL https://arxiv.org/abs/2107.06336.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wu19e.html.

Ou Wu, Weiyao Zhu, and Mengyang Li. Is data valuation learnable and interpretable? 2024.

Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V. Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.01.026. URL https://www.sciencedirect.com/science/article/pii/S0925231217300577.

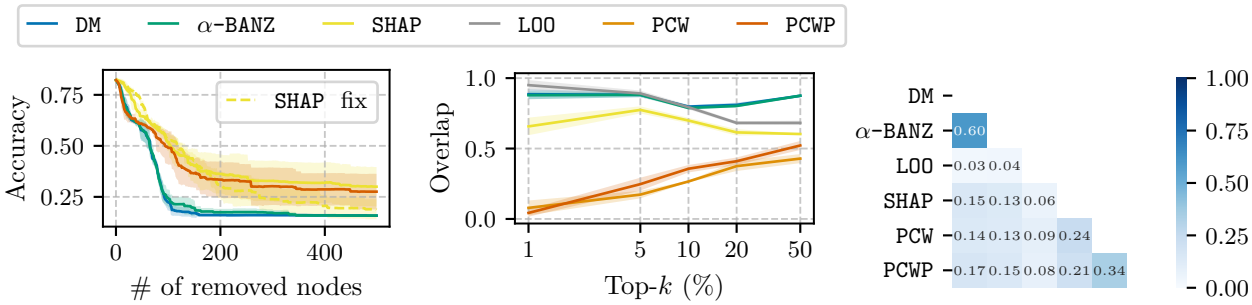

Figure 12: SHAP can be improved by adding 10% more nodes as context in its evaluated subsets.

Figure 13: DM and α-BANZ show consistent high overlap between learning and overall rankings.

Figure 14: Kendall's τ among approaches indicates DM and α-BANZ share a high rank correlation.

A Empirical limitation of game-theoretic approaches

We investigated why game-theoretic methods underperform compared to MSR-based approaches. Our analysis suggests that those methods commonly evaluate subsets containing only a small number of nodes when building up from empty sets via permutations. This yields many degenerate subsets and introduces noise in the final node value estimates. In contrast, randomly sampling larger subsets seems to better capture the underlying graph structure.

To address this, we propose a small modification to SHAP that ensures every evaluated subset includes a bit more context. Specifically, whenever a node is added to the subset during the permutation scan, we also include an extra 10% of the total nodes (in line with $\alpha = 0.1$ in MSR-based approaches). As in the standard data Shapley procedure, only the value of the newly added node is updated; the difference is that the node's influence is now assessed in a richer structural context.

Fig. 12 illustrates the impact of this modification: the dashed line shows that even a modest increase in contextual nodes yields more realistic Shapley values, thereby boosting performance. Determining the optimal fraction of additional context nodes remains an open question and is left as future work.

B Analysis of node values

Focusing on the values computed with SGC as the model on CoraML, we more carefully analyse the computed values in different settings.

B.1 Learning vs. overall rank

Because learning and overall signals can provide complementary insights, we investigate how these two perspectives differ in the rankings they produce. Specifically, for each approach, we compute the overlap in the top k% of node rankings derived from the corresponding $\mathbf{\Phi}$ values. Fig. 13 summarizes these results, illustrating that DM and α-BANZ yield consistently high overlap between the learning and overall rankings. This stability suggests that, even when a node is removed for both training and inference, these methods capture node importance in a way that remains robust across varying levels of information.

By contrast, PCW and PCWP show notable variations in their top k% rankings under different removal conditions. These fluctuations underscore the challenge of maintaining stable value estimates for nodes whose significance may shift when the graph and model context change between training and inference.

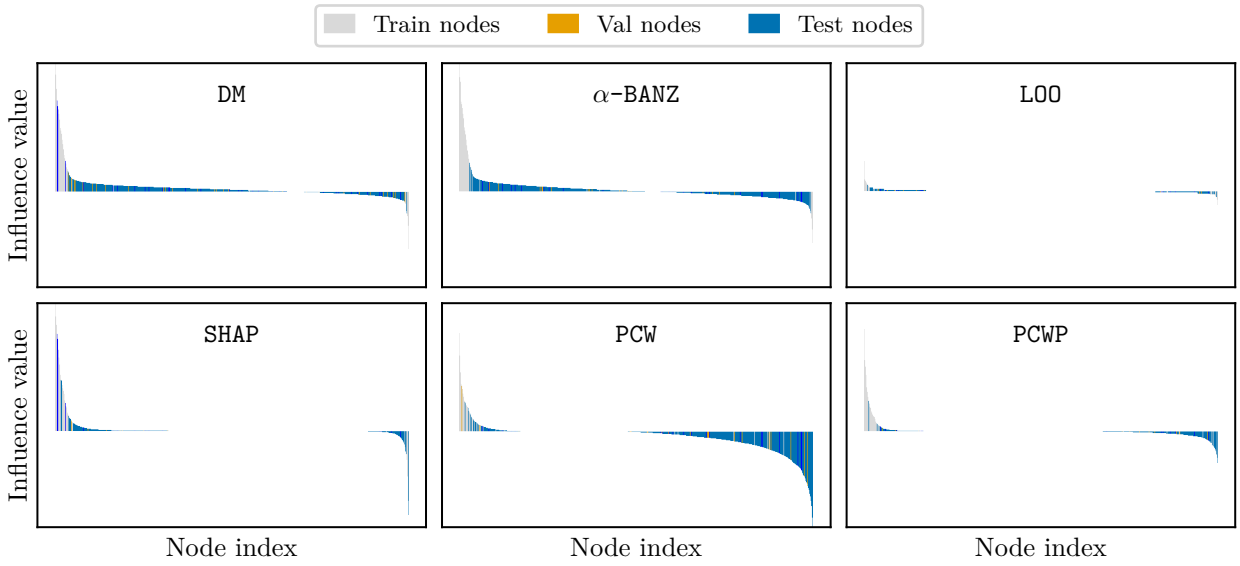

Figure 15: Node rankings by `learning` signal using test accuracy as utility.

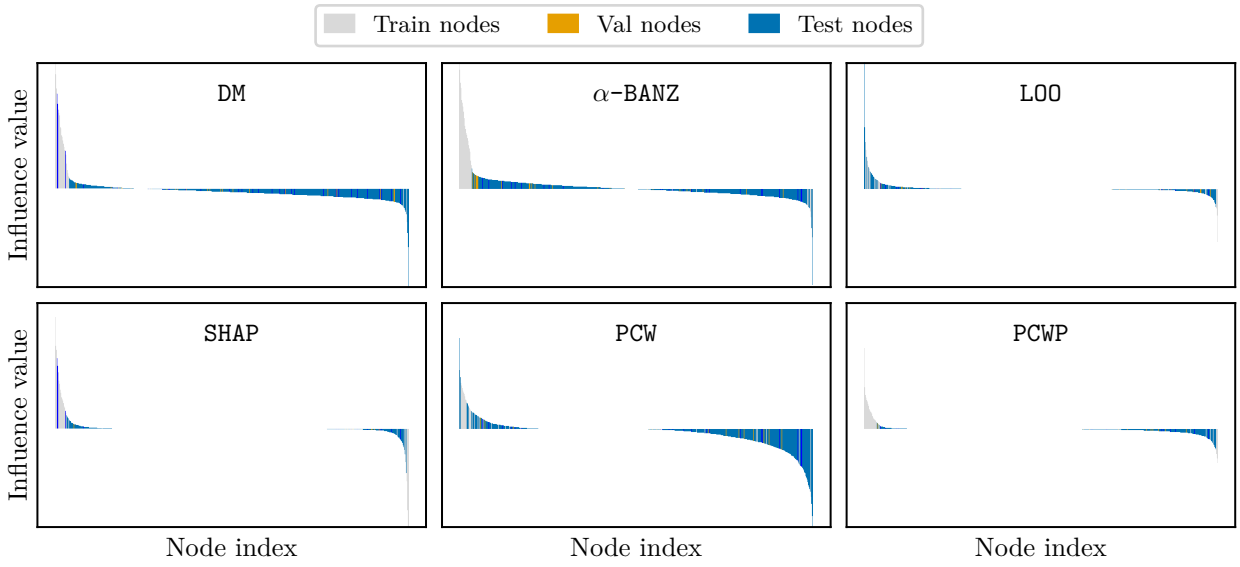

Figure 16: Node rankings by `learning` signal using test margins as utility.

B.2 Rankings of approaches

Because `DM` and α-`BANZ` demonstrate similar performance, we also examine how their node-level rankings compare to those generated by other methods. In Fig. 14, we report Kendall's τ correlations computed from the margin-based $\mathbf{\Phi}$ values, showing that `DM` and α-`BANZ` also exhibit a strong alignment. As expected, `PCW` and `PCWP` show notable correlation too.

Fig. 15 and Fig. 16 illustrate how each method ranks nodes according to the `learning` signal, using test accuracy (Fig. 15) and test margins (Fig. 16) as the utility (see § D.1 for an explanation on how to get influence scores from the per-node margins matrix). We highlight training nodes in gray, validation nodes in orange, and test nodes in blue. Although we use the `all` setting (where nodes can be removed at both

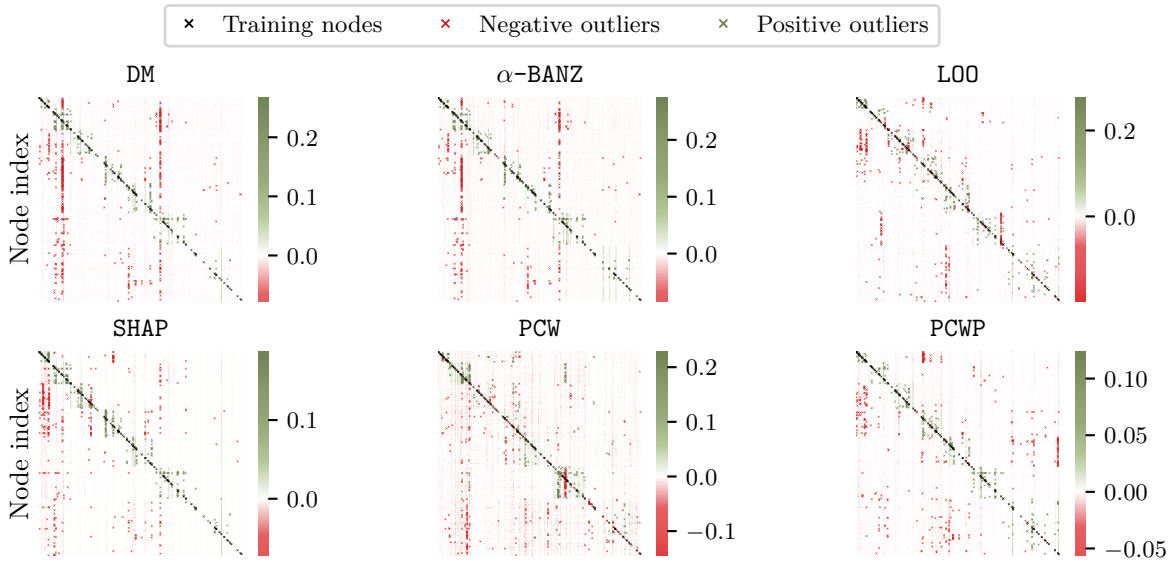

Figure 17: Heatmap of the `learning` signal values $\mathbf{\Phi}$ on `CoraML` for different approaches. Green and red dots represent positive and negative importance respectively. Crosses represent highly impactful nodes (with values larger than the ones indicated in the bars). Black crosses mark the training nodes.

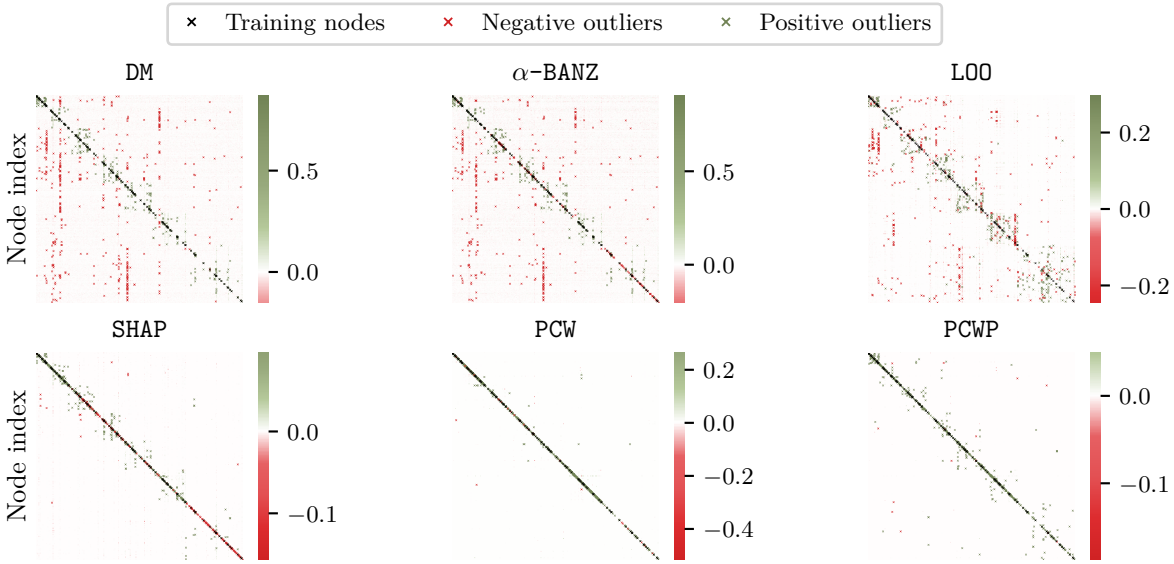

Figure 18: Heatmap of the `overall` signal values $\mathbf{\Phi}$ on `CoraML` for different approaches. Green and red dots represent positive and negative importance respectively. Crosses represent highly impactful nodes (with values larger than the ones indicated in the bars). Black crosses mark the training nodes.

training and inference), the majority of top-ranked nodes remain in the training set. Nevertheless, certain test nodes appear prominently in these rankings, suggesting that their removal adversely impacts model performance. For instance, `DM` often assigns high importance to particular test nodes which is consistent with Finding III. Overall, these results underscore that while most training nodes drive performance, there can be select test nodes that substantially influence model outcomes as well.

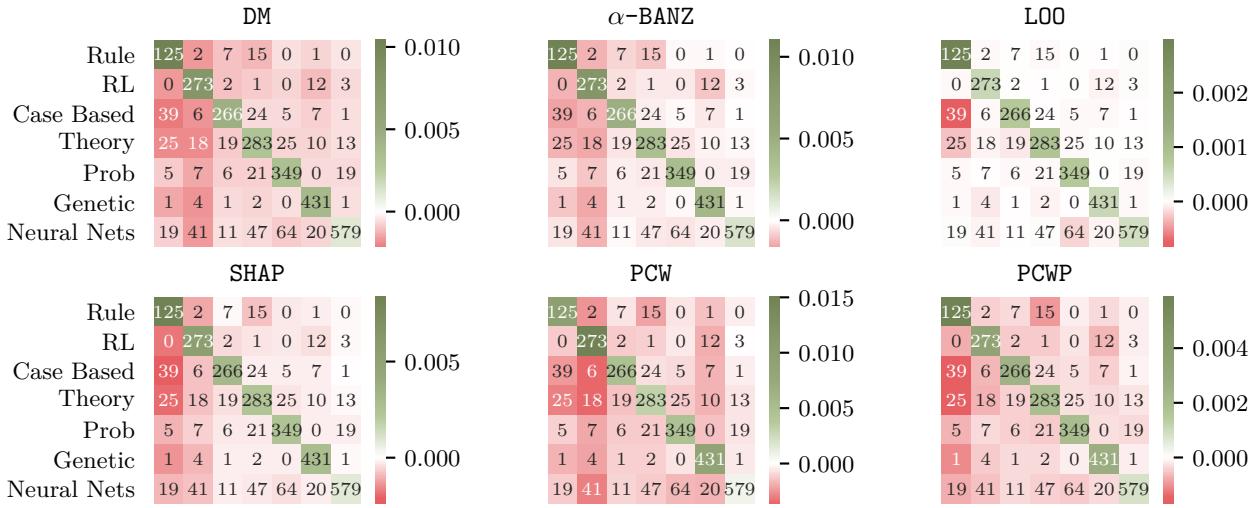

Figure 19: Average class-wise influence across different approaches according to $\mathbf{\Phi}$ capturing the `learning` signal. Classes are sorted by their size. Cell annotations are the confusion matrix values for model predictions.

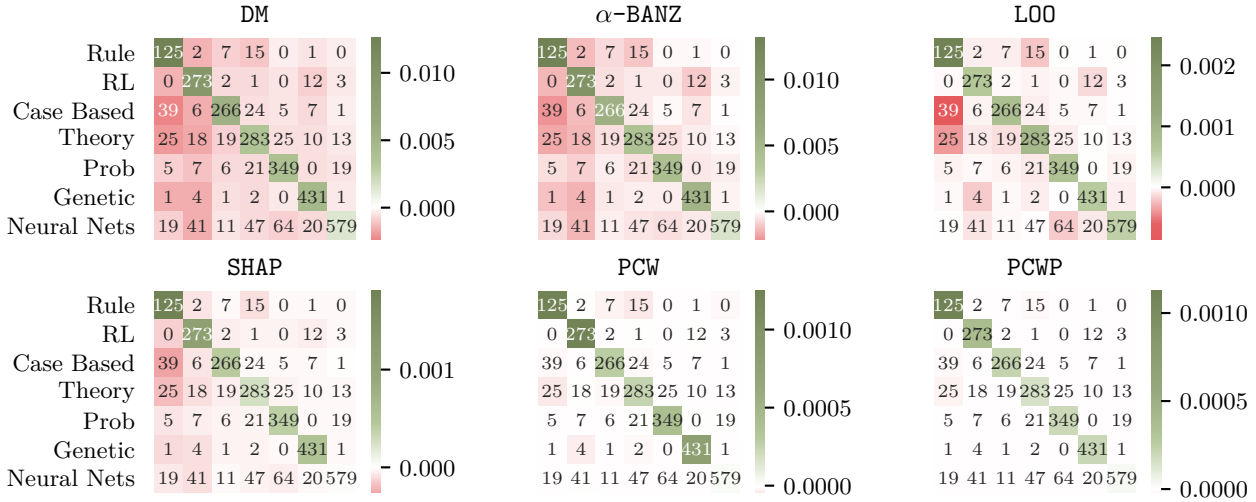

Figure 20: Average class-wise influence across different approaches according to $\mathbf{\Phi}$ capturing the `overall` signal. Classes are sorted by their size. Cell annotations are the confusion matrix values for model predictions.

B.3 Node- and class-level influence

Node influence. To provide a more detailed comparison in support of Finding I, we study the values $\mathbf{\Phi}$ under both `learning` and `overall` signals. Fig. 17 and Fig. 18 show the heatmaps of these values, sorted by class and degree. Positive (green) entries indicate that a node's removal has a beneficial effect on performance, while negative (red) entries show a detrimental effect. Black crosses mark training nodes; their corresponding vertical stripes illustrate how training nodes often exert influence on multiple other nodes supporting also Finding II.

Some methods (e.g., `PCW` and `PCWP`) assign values close to zero under the `overall` signal, mirroring their less reliable performance in the removal experiment. As a result, their final rankings can approximate random noise (see lower plots in Fig. 7).

Figure 21: Average class-wise influence across datasets. Classes are sorted by their size. Adding nodes from smaller communities highly impact the margins positively for nodes from the same class and negatively for nodes from different classes. This effect weakens the larger the community is.

Cluster influence. Fig. 19 and Fig. 20 show the average $\mathbf{\Phi}$ values on a per-class basis for the `learning` and `overall` signals. We overlay the confusion matrix of the model's predictions, allowing us to interpret how higher positive influence (darker green) correlates with a lower rate of false positives. Notably, `PCW` and `PCWP` have difficulty identifying highly negative influence in the `overall` setting, causing negative values to diminish in magnitude.

Fig. 21 shows the average class influence across all the datasets using $\mathbf{\Phi}_{\texttt{BANZ}}$ (`learning` signal). As pointed out in Finding I, smaller communities get the highest value than larger ones. We see how this effect is consistent across all the considered datasets.

Node influence visualization. Finally, to support Finding III, we extend the breadth-first search influence visualization to the `overall` in Fig. 22. As before, node size encodes importance for a given test node, and the color of the incoming edge (green or red) signifies whether the influence is positive or negative. We again observe that training nodes are typically the most influential, although certain test nodes within the same class (or those tightly connected to highly impactful training nodes) can also assume significant roles. In the `overall` signal, more outliers are apparent in terms of node size, indicating that a few nodes can exert an especially large effect on model outcomes.

B.4 Synchronized class influence

Analogously to Fig. 5, we investigate how nodes in the same class tend to exhibit similar influence patterns, both in how they influence others and in how they are influenced themselves (Fig. 23). To quantify these effects, we calculate the pairwise R^2 coefficient of the rows (bottom) and columns (top) of $\mathbf{\Phi}$ for different experimental settings.

On the left-most column of Fig. 23, we focus on the `train` setting, `overall` signal. As in Finding II, we observe that nodes within class tend to influence other nodes in a similar manner (top), resulting in a pronounced clustering effect. Similarly, the row-wise R^2 shows that nodes from the same class are also influenced similarly by other nodes (bottom), aligning with our earlier findings that training nodes can exert a broad impact across the graph.

On the right-most two columns of Fig. 23, we focus on the `all` setting and examine the two different signals. The middle column corresponds to the `learning` signal, while the right column corresponds to the `overall` signal. Similar class-wise clustering patterns are visible in the `learning` plots (particularly for the

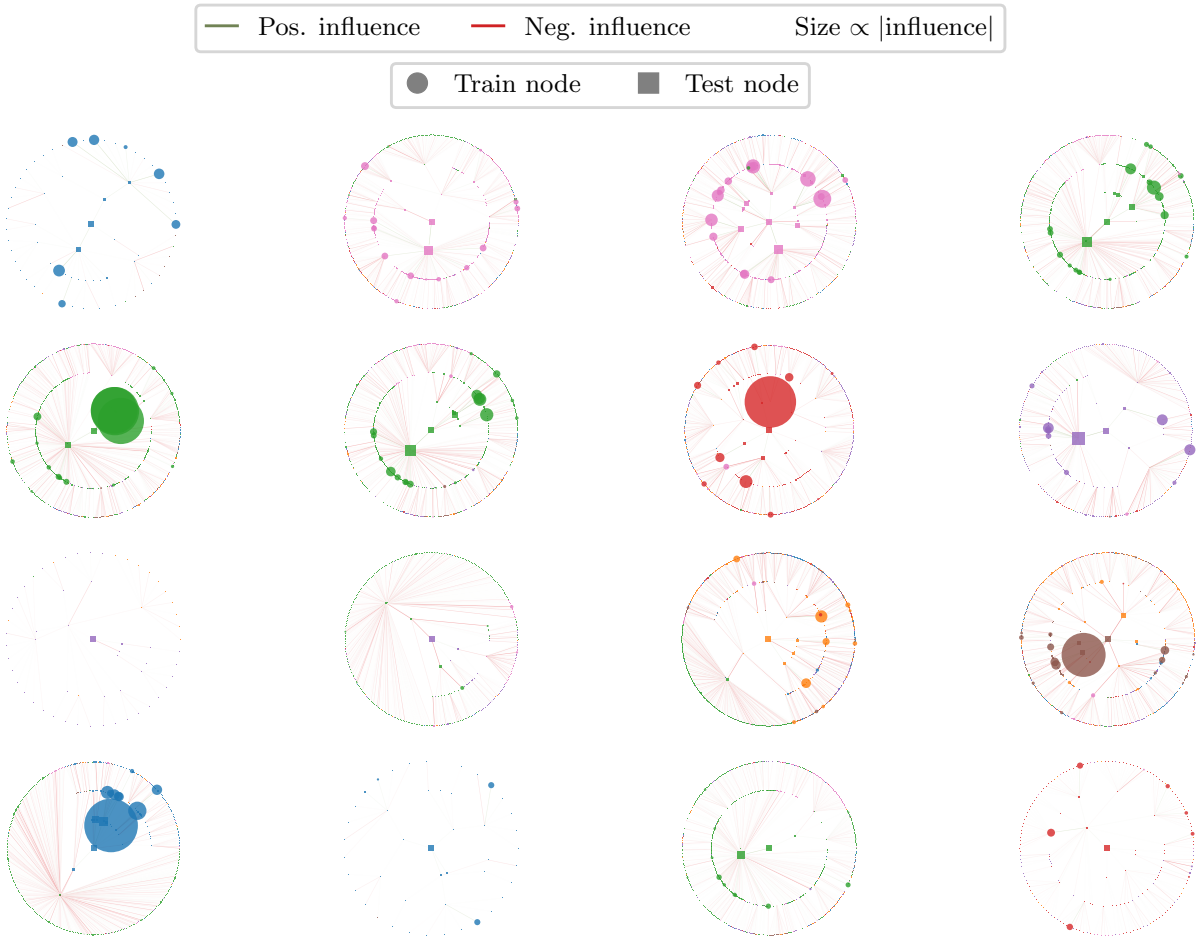

Figure 22: Two-hop breadth-first trees for `overall` $\mathbf{\Phi}$. Nodes are class-colored, sized by their $\mathbf{\Phi}$-based influence on the root, and edges show sign (green for positive, red for negative).

row-wise R^2), but they become weaker under the `overall` signal. We leave this as an interesting direction to investigate more in the future.

Overall, our results highlight a similar influence pattern in how nodes from the same class exert and receive influence to and by others.

C Additional experiments

C.1 Most influential addition

In Fig. 24, we show the results for the most influential node addition experiment. In this experiment, we start with an empty graph and then incrementally add nodes in order of their importance, as determined by each method's computed node values $\mathbf{\Phi}$. We evaluate the predictive performance (test accuracy) of the model after each batch of added nodes. Contrary to the most influential pruning, because we are adding most influential nodes first, we anticipate a rapid initial increase in accuracy, followed by a plateau once most of the crucial nodes have been included. We present results computed according to both `learning` (top) and `overall` (bottom) values. Similarly to the removal experiment, `DM` and α-`BANZ` better assess node values.

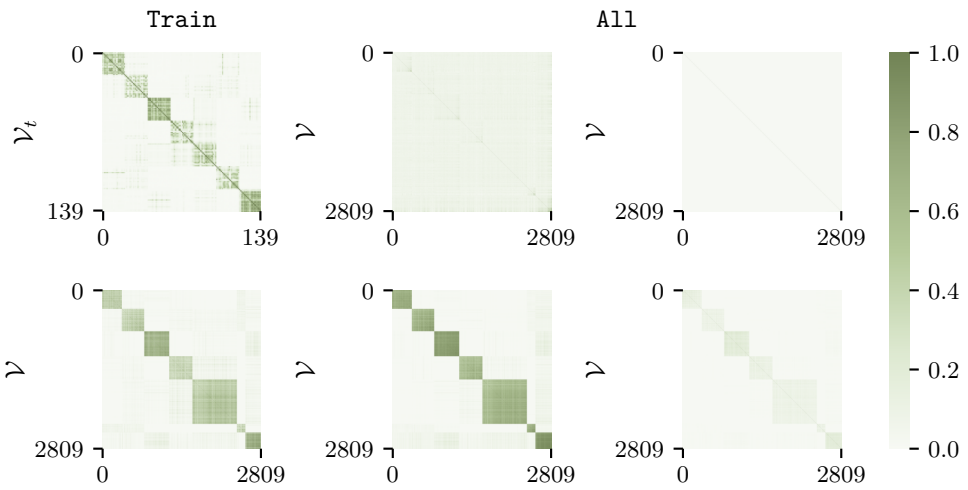

Figure 23: Pairwise R^2 coefficients of $\mathbf{\Phi}$ under different settings. The first row shows the column-wise R^2 measures, and the second row shows the row-wise R^2 measures. The left-most column corresponds to the `Train` setting, `overall` signal. The right-most two columns correspond to the `All` setting, where the middle column represents the `learning` signal and the right column represents the `overall` signal.

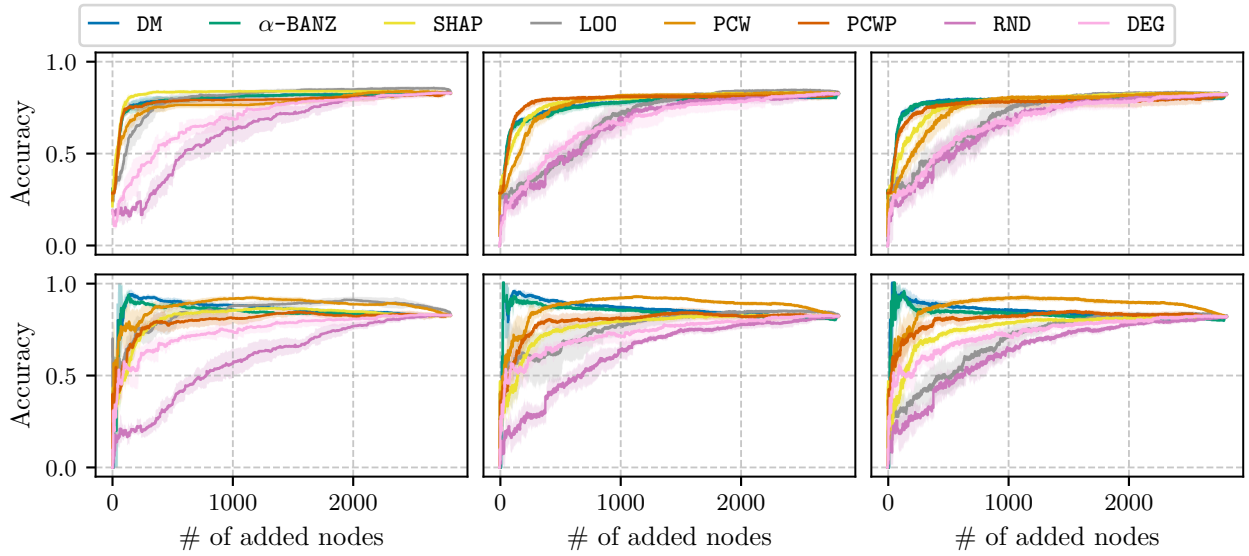

Figure 24: Most influential node addition for `CoraML` across models: SGC (first column), GCN (second column) and GAT (third column). `learning` (top) and `overall` (bottom) signal of $\mathbf{\Phi}$. Adding high-value nodes leads to a performance (test accuracy) raise until reaching a plateau given by keep adding uninfluential nodes.

C.2 Influential nodes in the `train` setting

Here we present the results for the node influence experiment in the `train` setting, namely we attribute importance (and then remove) only training nodes. In Fig. 25, we show how the approaches capture the real value considering both the `learning` (top) and `overall` (bottom) signals, across different models – SGC (first column), GCN (second column) and GAT (third column). Notably, we observe the approaches struggle more in attributing the right values when non-linearities are introduced in the models (compare

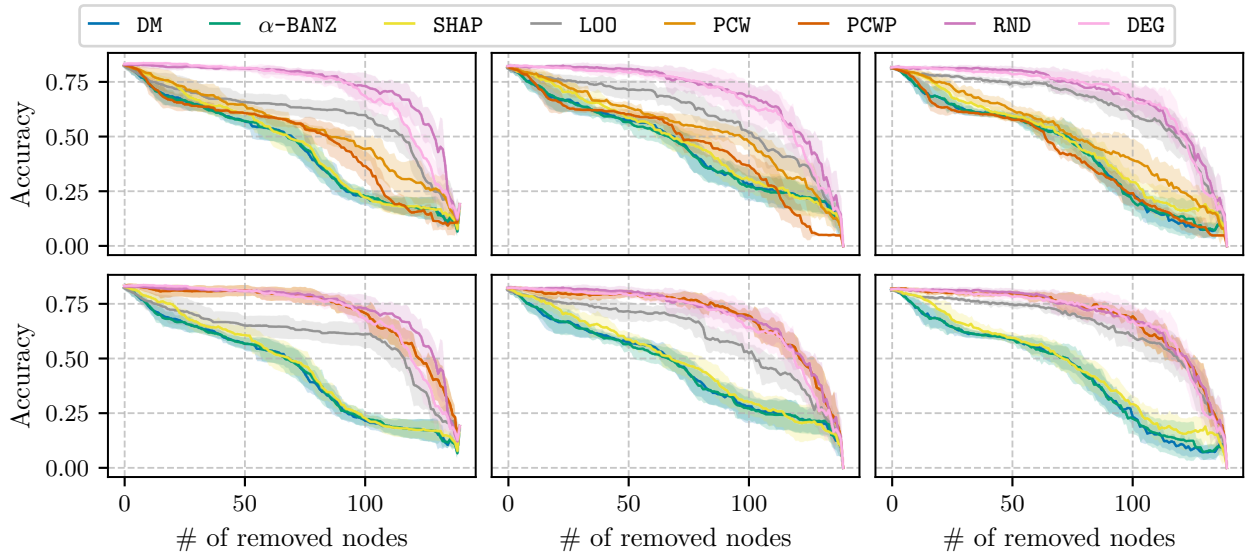

Figure 25: Most influential node removal for `CoraML` across models in the `train` setting: SGC (first column), GCN (second column) and GAT (third column). `learning` signal (top) and `overall` signal (bottom) $\mathbf{\Phi}$.

SGC w.r.t. GCN and GAT). Consistently with previous results, `DM` and α-`BANZ` perform better than the other approaches.

C.3 Brittleness in the `train` setting

We replicate the same experiment used in Finding IV, but now in the `train` setting, to assess whether each approach maintains strong predictive performance. In Fig. 26, we evaluated both the `learning` (top) and `overall` (bottom) signals computed with SGC (left) and GCN (right), and in each scenario, `DM` and α-`BANZ` demonstrated performance closest to the best guess among all methods.

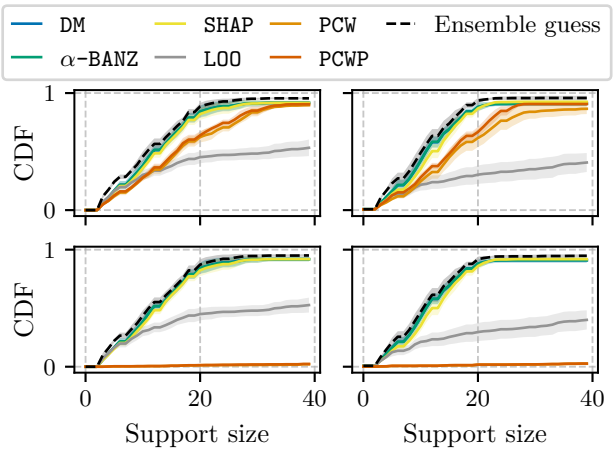

Figure 26: Brittle prediction in the `train` setting for SGC (left) and GCN (right). `DM` and α-`BANZ` consistently approach the best guess regardless of the considered `learning` (top) or `overall` (bottom) signals.

C.4 Memorization ranking

Fig. 27 shows the rankings of the approaches as a result of the memorization experiment. As explained in Finding IV, we poison 10% of the training data and expect a good data validation approach to rank these nodes as the most important for their prediction. We show the results for `CoraML`, where the number of training nodes is 140 resulting in 14 poisoned nodes. We then show only the top 14 nodes ranked by the approaches. The results show that `DM` and α-`BANZ` rank first the poisoned nodes establishing as robust approaches for detecting poisoned or mislabeled data.

In this experiment, we train `DM` by considering all possible node subsets, regardless of whether a particular node is included. Unlike the procedure described in § D, this setup is feasible here because we are no longer focused on maximizing predictive performance

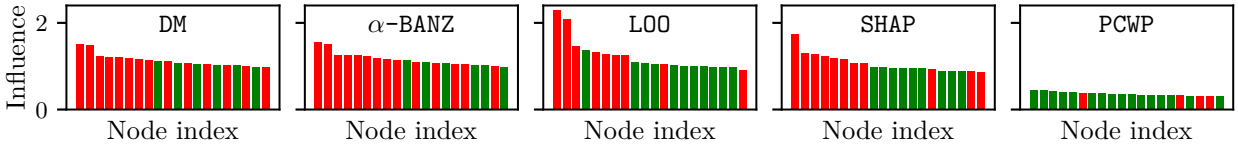

Figure 27: Training nodes rankings according to values computed on a graph with 10% of poisoned (red) nodes.

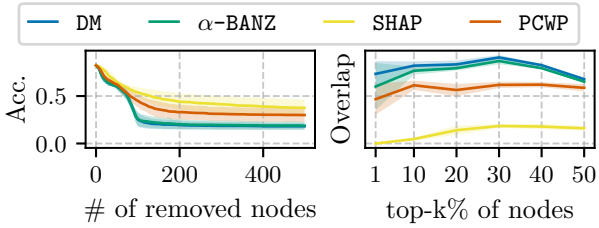

Figure 28: Approaches stability: (left) in predicting node values; (right) in predicting the ranking for a given split across 10 seeds.

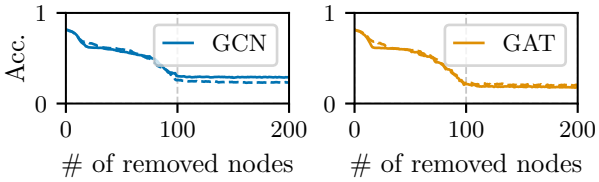

Figure 29: Node removal using data values computed and evaluated by each model (solid) or transferred from SGC (dashed).

(where information leakage would pose a risk). Instead, our primary goal is to evaluate each method's ability to identify problematic data.

C.5 Stability

A crucial aspect of reliable data valuation is consistency – producing similar ranking results across multiple runs of the same experiment. Fig. 28 illustrates the stability of each approach, measured over 10 random seeds for each of the 5 train/val/test splits in `CoraML`. Each seed defines different subsets/permutations of nodes for valuation. We report the average performance of each method (left) and observe that `DM` and `α-BANZ` exhibit the lowest variance across runs, making them the most reliable choices for data valuation on graphs.

We also evaluate ranking stability (right panel) by measuring the percentage of top-k nodes that overlap across different seeds in `CoraML`. Once again, `DM` and `BANZ` demonstrate near-identical rankings across all seeds for a given train/val/test split, underscoring their reliability and consistency.

C.6 Transferability

In Fig. 29, we repeat the node removal experiment (similar to Fig. 8) with two variants: i) Computing data values with a GCN model and evaluating performance on the same GCN (solid line, no transfer). ii) Computing data values with SGC and seeing whether these values transfer to GCN (dashed line) and GAT.

Using `α-BANZ` and margins as the utility metric on `CoraML`, we observe nearly identical performance among these variants, indicating that data values capture intrinsic properties of the data rather than the specifics of each model. This finding further suggests that cheaper models (e.g., SGC) suffice to compute data values that effectively transfer to more complex architectures.

D Further experiment details

Datasets and models. Tab. 1 summarizes the datasets used in our experiments. Following Shchur et al. (2018), we employ stratified sampling to select the training nodes (20 per class) along with an equal number of validation nodes. The remaining nodes serve as the test set. We choose SGC, GCN, and GAT as our architectures, each configured with 2 layers. For GCN and GAT, we use 16 hidden channels, a dropout rate

of 0.5, and a ReLU activation. Specifically for GAT, we adopt the GATv2 variant from Brody et al. (2022), using 8 attention heads.

Table 1: Summary of datasets.

Dataset	Nodes	Edges	Features	Classes	Nodes (LCC)	Edges (LCC)	# Train/Val/Test
CoraML	2 995	16 316	2 879	7	2 810	15 962	140 / 140 / 2 715
Citeseer	4 230	10 674	602	6	1 681	5 804	120 / 120 / 3 990
PubMed	19 717	88 648	500	3	19 717	88 648	60 / 60 / 19 597
Photo	7 650	238 162	745	8	7 487	238 086	160 / 160 / 7 330
Computers	13 752	491 722	767	10	13 381	491 556	200 / 200 / 13 352
CoPhysics	34 493	495 924	8 415	5	34 493	495 924	100 / 100 / 34 293

Training setting. We train GCN and GAT using the Adam optimizer with a learning rate of 0.01 for 3000 epochs. Early stopping is based on the validation loss, with 50 epochs of patience. Results are averaged over 10 runs for each model, except for larger datasets where we use 5 runs. In contrast, SGC leverages a closed-form solution by relaxing the classification objective into a regression problem. Due to the absence of non-linearities, the model weights can be derived as $\boldsymbol{W}^\star = \tilde{\hat{\boldsymbol{X}}}_\ell \boldsymbol{H}$, where $\tilde{\hat{\boldsymbol{X}}} = (\hat{\boldsymbol{X}}^\top \hat{\boldsymbol{X}} + \lambda \boldsymbol{I})^{-1} \hat{\boldsymbol{X}}^\top$ and $\hat{\boldsymbol{X}} = \boldsymbol{S}^2 \boldsymbol{X}$. For all models, we conduct experiments using 5 different train/val/test splits and report average performance. We train each node datamodel with the ridge regression (with cross-validation) implementation from scikit-learn (Pedregosa et al., 2011) using its default hyperparameters.

Running these experiments on CPUs generally proves faster than on GPUs, given the shallow architectures. We parallelize the runs of different models with the joblib library (Joblib Development Team, 2020), leveraging the larger RAM capacity (compared to GPU memory) to execute as many jobs as possible simultaneously. Experiments are performed on a cluster of 136 nodes, each equipped with 2x AMD Epyc 9654 processors (96 cores, 2.4–3.7 GHz) and 768 GB of RAM. For larger datasets, such as `CoPhysics`, we switch to a bigger partition with 8 nodes (same CPU specifications) but 3 TB of RAM to accommodate higher memory requirements.

Running times. We split the computational cost of each method into two components: gathering utility across the selected subsets and computing the final data values. Tab. 2, Tab. 3, and Tab. 4 show the total cost for each approach under 50 000 subsets with $\alpha = 0.1$. Among methods with comparable performance, α-`BANZ` exhibits the best overall runtime. A "-" in the tables indicates that an approach ran out of memory.

Table 2: SGC computation times.

Dataset	DM		α-BANZ		LOO		SHAP		PCWP	
	Train	All	Train	All	Train	All	Train	All	Train	All
Citeseer	7 m 32 s	11 m 38 s	7 m 20 s	5 m 7 s	13 s	29 s	6 m 28 s	3 m 24 s	4 m 4 s	3 m 57 s
CoraML	41 m 30 s	58 m 50 s	41 m 13 s	25 m 57 s	25 s	2 m 54 s	43 m 16 s	46 m 15 s	40 m 35 s	35 m 28 s
PubMed	-	-	-	2 d 4 h 24 m 40 s	29 m 18 s	2 d 6 h 57 m 7 s	2 m 9 s	-	3 d 17 h 35 m 6 s	3 d 17 h 42 m 44 s
Photo	7 h 53 m 49 s	13 h 18 m 7 s	7 h 46 m 41 s	3 h 0 m 23 s	2 m 4 s	1 h 13 m 57 s	8 h 25 m 21 s	17 h 32 m 58 s	7 h 41 m 21 s	7 h 42 m 2 s
Computers	1 d 20 h 18 m 23 s	3 d 16 h 3 m 12 s	1 d 19 h 39 m 17 s	16 h 36 m 34 s	19 m 8 s	11 h 49 m 18 s	1 m 8 s	-	2 d 9 h 30 m 55 s	2 d 9 h 10 m 48 s
CoPhysics	-	-	-	-	3 h 8 m 27 s	-	34 m 25 s	-	-	-

Table 3: GCN computation times.

Dataset	DM		α-BANZ		LOO		SHAP		PCWP	
	Train	All	Train	All	Train	All	Train	All	Train	All
Citeseer	52 m 40 s	16 m 33 s	51 m 6 s	9 m 46 s	39 s	2 m 22 s	53 m 35 s	28 m 2 s	10 m 48 s	10 m 48 s
CoraML	3 h 53 m 11 s	56 m 33 s	3 h 54 m 16 s	24 m 13 s	1 m 7 s	14 m 54 s	3 h 48 m 52 s	39 m 35 s	17 m 39 s	18 m 1 s
PubMed	3 h 5 m 17 s	-	3 h 2 m 40 s	2 h 57 m 21 s	50 s	1 h 28 m 26 s	2 m 4 s	2 h 20 m 5 s	43 m 16 s	44 m 23 s
Photo	10 h 9 m 58 s	14 h 24 m 27 s	10 h 17 m 5 s	34 m 50 s	9 m 39 s	5 h 22 m 32 s	11 h 3 m 58 s	1 h 44 m 26 s	21 m 24 s	21 m 25 s
Computers	1 d 1 h 47 m 17 s	-	1 d 1 h 20 m 49 s	1 h 43 m 28 s	19 m 12 s	18 h 5 m 51 s	1 m 8 s	5 h 22 m 29 s	27 m 38 s	27 m 34 s
CoPhysics	4 d 3 h 57 m 31 s	-	-	1 d 2 h 42 m 20 s	17 m 23 s	-	40 m 10 s	-	-	-

Datamodels training. By default, `DM` excludes any subset that contains a target sample in the training set to prevent information leakage (Ilyas et al., 2022), particularly when evaluating predictive performance. However, this procedure is not always feasible across different graph settings. Specifically, when computing a

Table 4: GAT computation times.

Dataset	DM		α-BANZ		LOO		SHAP		PCWP	
	Train	All	Train	All	Train	All	Train	All	Train	All
Citeseer	3 h 40 m 48 s	25 m 53 s	3 h 32 m 20 s	19 m 36 s	2 m 1 s	10 m 22 s	3 h 28 m 29 s	1 h 0 m 43 s	27 m 57 s	27 m 57 s
CoraML	15 h 15 m 20 s	1 h 17 m 7 s	15 h 2 m 52 s	45 m 14 s	4 m 56 s	1 h 11 m 2 s	16 h 50 m 56 s	1 h 59 m 5 s	44 m 45 s	44 m 45 s
PubMed	17 h 59 m 22 s	-	17 h 46 m 60 s	3 h 22 m 10 s	4 m 32 s	10 h 56 m 16 s	2 m 12 s	9 h 58 m 24 s	52 m 32 s	47 m 53 s
Photo	-	12 h 30 m 27 s	-	1 h 17 m 51 s	44 m 25 s	1 d 5 h 18 m 20 s	4 d 7 h 42 m 0 s	7 h 9 m 39 s	41 m 51 s	41 m 33 s
Computers	-	-	-	3 h 50 m 26 s	2 h 17 m 30 s	-	1 m 3 s	1 d 1 h 51 m 20 s	1 h 14 m 26 s	1 h 14 m 21 s
CoPhysics	-	-	-	1 d 12 h 25 m 59 s	1 h 13 m 2 s	-	38 m 14 s	-	-	-

node's `learning` signal value, we exclude subsets containing that node from the sampled distribution used to train the datamodel[3]. Conversely, when computing the `overall` value, excluding those subsets would leave no data available for training the datamodel, making this approach impractical.

Banzhaf margins utility. In our work, we adopt the MSR implementation to approximate the Banzhaf values:

$$\phi_{\texttt{BANZ}}(i) = \frac{1}{|\mathcal{S}_{\in i}|} \sum_{\mathcal{S} \in \mathcal{S}_{\in i}} u(\mathcal{S}) - \frac{1}{|\mathcal{S}_{\notin i}|} \sum_{\mathcal{S} \in \mathcal{S}_{\notin i}} u(\mathcal{S}). \tag{3}$$

While this is straightforward when each node's utility is defined for every other node, when computing the `overall` signal in the `all` setting, the utilities may be undefined for some nodes. Consequently, the sum of utilities has to be averaged according to considered utilities. Namely, it depends on whether each other node is inside or outside the subsets considered for node i. Concretely, given arrays of margins $\mathbf{\Phi}_{(\mathcal{S})}[i,:]$ for a node i across different subsets $\mathcal{S}$, utilities are undefined for a node j that is not in $\mathcal{S}$. Thus, dividing by $|\mathcal{S}_{\in i}|$ in cases where not all those subsets actually define a margin for j artificially reduces the measured impact of i on j. For this reason, we instead average the influence accounting for the the number of times node j appears in the considered subsets $|\mathcal{S}_{\in i}|$, and analogously for $|\mathcal{S}_{\notin i}|$.

D.1 Nodes ranking - from margins to influence scores

In the node influence experiment, each node needs a single influence score. However, when margins serve as the utility, we obtain an influence value for every other node's prediction, $\mathbf{\Phi}[i,:] \in \mathbb{R}^n$. To derive a final score and, subsequently, a ranking, we average each node's contribution across all other nodes' predictions. Concretely, we form a matrix of influence scores where row i indicates how other nodes influence node i's prediction. We then compute the mean of each column in this matrix to obtain a final score for each node, which we use to produce the ranking.

D.2 Subsets sampling

Because it is infeasible to evaluate utility for all $2^{|\mathcal{D}|}$ subsets, a common approximation strategy is to sample subsets. A straightforward but effective approach is to include each instance independently with probability α, i.e., $\Pr[i \in \mathcal{S}] = \alpha$ for every $i \in \mathcal{D}$. As a result, the subset size follows a Binomial distribution with expected value $\alpha \cdot |\mathcal{D}|$, implicitly capping the average subset size. We adopt this method for both data Banzhaf and datamodel values. As shown in § D.3, α is an important hyperparameter that we tune. To approximate the Shapley value, we randomly sample a permutation of the nodes, then scan them in order until reaching a user-defined truncation threshold. For a fair comparison, each method uses the same total number of subset evaluations.

We sample 50 000 subsets per method. For α-`BANZ` and `DM`, we tested various α values and identified $\alpha = 0.1$ as optimal for `CoraML` (details in § D.3); we then apply this choice to the remaining graphs. For `SHAP`, we adopt the implementation of Ghorbani & Zou (2019), which stops processing permutations after 25% of the nodes have been scanned. Consequently, we set the number of permutations to $\frac{50\,000}{\lceil n \cdot \mathrm{tr} \rceil}$ with $\mathrm{tr} = 0.25$. When computing node values under the `all` setting, permutations may start with nodes outside the training set, causing random predictions to introduce noise. Therefore, we begin updating node values only once we reach the first training node in each permutation and truncate 25% from that point. Similarly, for `PCW`, we choose the number of permutations so that the total number of subsets remains 50 000. In Chi et al. (2025a), the

[3]In the `all` setting, this exclusion also applies to $\mathcal{V}_u$.

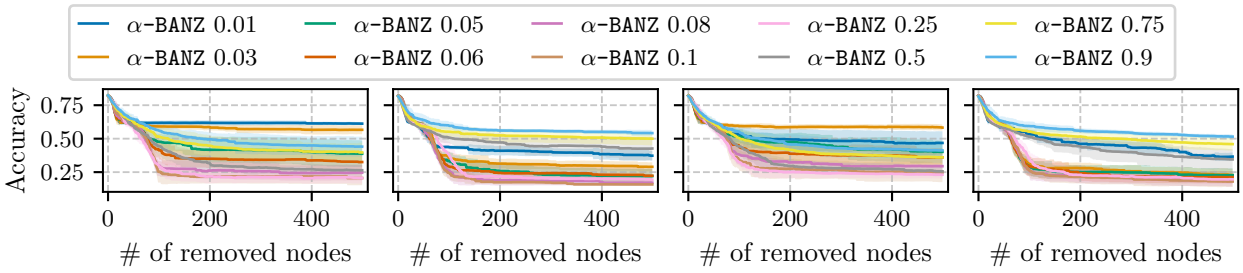

Figure 30: Hyperparameter search for the best α for subsets' sampling. From left to right, subplots report the results for `learning` & test margins utility, `learning` & test accuracy utility, `overall` & test margins utility and `overall` test accuracy utility.

default truncation ratios tr for 1-hop and 2-hop neighbors are 0.5 and 0.7, respectively; we optimize these ratios to obtain `PCWP`, which yields slightly better performance.

D.2.1 Optimized `PCW` – `PCWP`

In `PCW`, we compute marginal contributions for nodes only within the first $(1-\text{tr})$ portion of each node's child sub-trees. We approximate the contributions of the remaining sub-trees as 0, stopping once we reach exactly 50 000 subsets. Because this approach was originally developed for an inductive setting, its default truncation ratios appeared overly tailored to that context. We discovered that optimizing these truncation ratios to maximize the number of considered permutations improved performance in the transductive setting. Consequently, we set both the 1-hop and 2-hop neighbor truncation ratios to 0.99, referring to this variant as `PCWP`.

D.3 Choice of the hyperparameter α

As mentioned in § D.2, α is the probability of including each node from $\mathcal{D}$ in a subset, effectively determining the subset size. For example, in the `train` setting, where $\mathcal{D} = \mathcal{V}_t$, choosing $alpha = 0.1$ yields subsets comprising 10% of $\mathcal{V}_t$ plus all the other nodes. This parameter serves as a hyperparameter for `DM` and α-`BANZ` and can be selected via a grid or random search. Due to the high computational cost of performing such a search on every dataset, we determined the optimal α for `CoraML` and reused it across all other datasets, acknowledging it may be suboptimal in certain cases. Nonetheless, as demonstrated in § 4, this choice of α outperforms other methods on most datasets.

Fig. 30 shows how α-`BANZ` performs in the most influential pruning experiment under various α values used to sample subsets for node-value prediction. We observe that $\alpha = 0.1$ and $\alpha = 0.25$ yield similar estimates. However, because smaller subsets reduce training time for the models, we chose $\alpha = 0.1$ as the best overall setting.