Using Categorical Features for Deriving Graphs from Tabular Data

Santiago Tena Hernandez

Supervisors
Prof. Dr. Dr. Lars Schmidt-Thieme
Dr. Maximilian Stubbemann

Table of Contents

Table of Contents

Tabular Data Learning

With tabular data, each row corresponds to a single instance while each column represents a continuous or categorical feature.

Widely used for the training of predictive models:

SVM, Regression Models, GBDT, DNN

	f_1	f_2	…	f_j
x_1			…	
x_2			…	
…	…	…	…	…
x_i			…	

Motivation

Graph Neural Networks are a subtype of deep learning algorithms.

They seem to leverage relationships between data points. Effective in molecular chemistry, social networks and traffic modeling.

They have been effectively used in the context of tabular data learning.

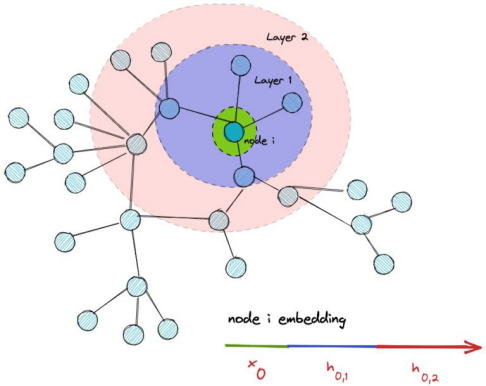


Figure 1: *A high-level representation of a GNN architecture; taken from [Aomar, 2021]*

Research Questions

1. Can tabular data be used to define a graph structure that can be effectively used in a GNN for classification purposes?

2. Are categorical features enough to create meaningful graph representations?

3. How do graphs derived from categorical features differ from those derived from continuous features?

4. How do these GNN models compare to the algorithms commonly used for this kind of task?

Table of Contents

Tabular Data Learning

Let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ denote a tabular dataset with n data instances, where $\mathbf{x}_i = (\mathbf{x}_i^{(\text{num})}, \mathbf{x}_i^{(\text{cat})}) \in \mathcal{X} \subseteq \mathbb{R}^d$ represents the numerical feature $x_{ij}^{(\text{num})}$ and categorical features $x_{ij}^{(\text{cat})}$, and $y_i \in \mathcal{Y}$ denotes the label of data instance i. The total number of features is denoted as d. The dataset can be split into three disjoint subsets, $D = D_{\text{train}} \cup D_{\text{val}} \cup D_{\text{test}}$.

The tabular data learning classification tasks have two types: binary classification $\mathcal{Y} = \{0, 1\}$ and multi-class classification $\mathcal{Y} = \{1, 2, ..., C\}$, where C is the number of class labels.

It is assumed that every input feature vector $\mathbf{x}_i$ in D is sampled i.i.d. from a feature distribution p_X, and the labeled data pairs $(\mathbf{x}_i, y_i)$ in D are drawn from a joint distribution $p_{X,Y}$. When labeled samples from $p_{X,Y}$ are available, we can train a predictive model $f : \mathcal{X} \to \mathcal{Y}$ by minimizing the empirical supervised loss $\sum_{i=1}^{|D_{\text{train}}|} \mathcal{L}(f(\mathbf{x}_i), y_i)$, where $\mathcal{L}$ is a standard supervised loss function, namely cross-entropy for classification.

	x_1	…	x_j	y
1		…		
2		…		
…	…	…	…	…
n		…		

Multilayer Perceptron

Given an input vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ and during forward propagation, each neuron performs a weighted sum of inputs and applies an activation function:

- Let $W^{(l)}$ be the weight matrix for layer l.
- Let $b^{(l)}$ be the bias vector for layer l.
- Let $h^{(l)}$ be the output of layer l before activation.

The neuron computations are:

$$h^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = f(h^{(l)})$$

where:

- $a^{(l-1)}$ is the activations from the previous layer, with $a^{(0)} = \mathbf{x}$.
- $f(\cdot)$ is an activation function, such as *ReLU*, *sigmoid*, or *tanh*.

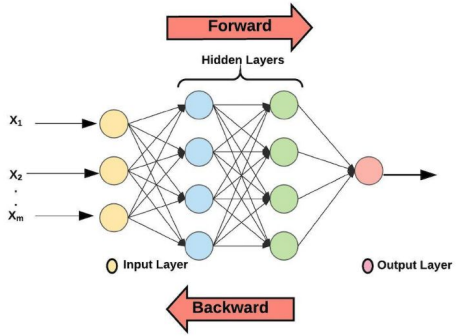


Figure 3: *A depiction of an MLP model; obtained from [Aldosary et al., 2021]*

Multilayer Perceptron

For mulit-class classfication, the output layer uses *softmax*:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

where:

- $z = a^{(L-1)}$.
- C is the number of classes.

The objective function is the Categorical Cross-Entropy Loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log \hat{y}_{ij}$$

where:

- N is the number of samples
- y is the true class label
- $\hat{y}$ is the predicted probability.

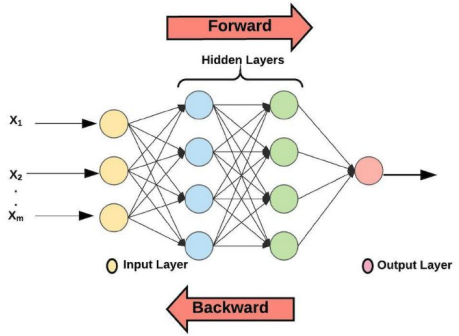


Figure 3: *A depiction of an MLP model; obtained from [Aldosary et al., 2021]*

XGBoost

XGBoost builds an ensemble of decision trees, where each new tree corrects the mistakes of the previous ones by focusing on incorrectly classified instances.

Its objective function consists of two parts:

1. **Loss function** L measures how well the model fits the data.
2. **Regularization term** Ω penalizes overly complex trees.

$$\mathcal{O}(\Theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t)$$

where:

- y_i is the ground truth.
- $\hat{y}_i$ is the predicted output.
- f_t represents the tree added at step t.
- $\Omega(f_t)$ is the regularization term.

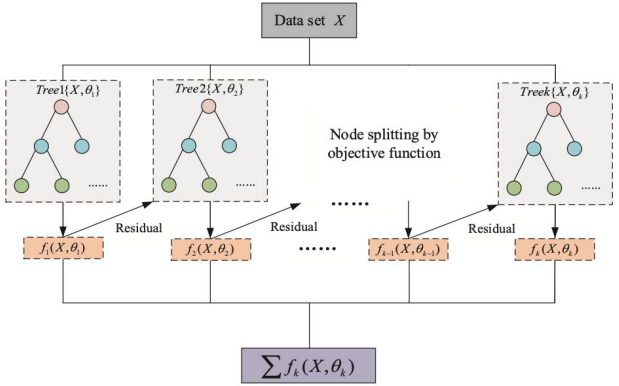


Figure 2: *Gradient boosting representations; obtained from [Guo et al., 2020]*

XGBoost

For each iteration, XGBoost refines its predictions in an iterative manner:

$$\hat{y}_i^{(t+1)} = \hat{y}_i^{(t)} + f_t(x_i)$$

Using a second-order Taylor expansion, the loss function for a new tree is approximated as:

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

where:

- $g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ (first derivative of the loss function)
- $h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$ (second derivative of the loss function)

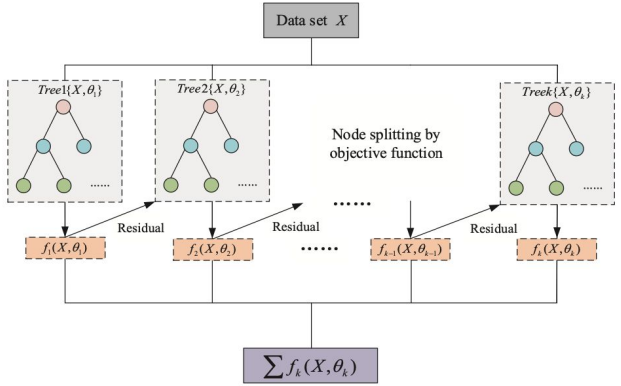


Figure 2: *Gradient boosting representations; obtained from [Guo et al., 2020]*

Graph Construction (SFV)

Consider a dataset with n data points, each represented by a feature vector: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ where each x_i is a feature vector in $\mathbb{R}^d$ if dealing with numerical features or a categorical space. Let F be the set of all features, and let $f \in F$ be a specific feature of interest. The Same Feature Value Graph is defined as an undirected graph:

$$G_{SFV} = (V, E)$$

where:

- $V = X$, each node represents a data instance
- $E = (x_i, x_j) \mid f(x_i) = f(x_j),\ i \neq j$

This means that an edge exists between two nodes if they share the same value for feature f.

Graph Construction (KNN)

K-Nearest Neighbors is a supervised learning algorithm. It is a non-parametric, instance-based learning method that can be used to generate graphs using distance metrics to determine the value of edges connecting their nodes. The most common choice for a distance function is the Eucledian distance given by:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2}$$

where:

- $x = \{x_1, x_2, ..., x_n\}$ is the input point to be classified or predicted.
- $x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$ is a training example.
- n is the number of features.

Graph Construction (KNN)

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a dataset with n data points. The K-Nearest Neighbors Graph is defined as an undirected graph:

$$G_{KNN} = (V, E)$$

where:

- $V = X$, where each node represents a data instance.
- $E = \{(x_i, x_j) \mid x_j \in kNN(x_i)\}$.

This means an edge exists between x_i and x_j if x_j is one of the k nearest neighbors of x_i.

Graph Convolutional Network

Each graph node $v \in V$ has a feature vector x_v, and all node features are represented as a matrix $X \in \mathbb{R}^{N \times F}$ where N is the number of instances and F is the number of input features per node.

The core operation follows this propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$$

where:

- $H^{(l)}$ is the node feature matrix at layer l, with $H^{(0)} = X$ as input.
- $W^{(l)}$ is the trainable weight matrix at layer l.
- $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix, where $A_{ij} = 1$ if nodes i and j are connected, otherwise 0.
- $\tilde{A} = A + I$.
- $\tilde{D}$ is the diagonal degree matrix, where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.
- σ is a non-linearity, such as $ReLU$.

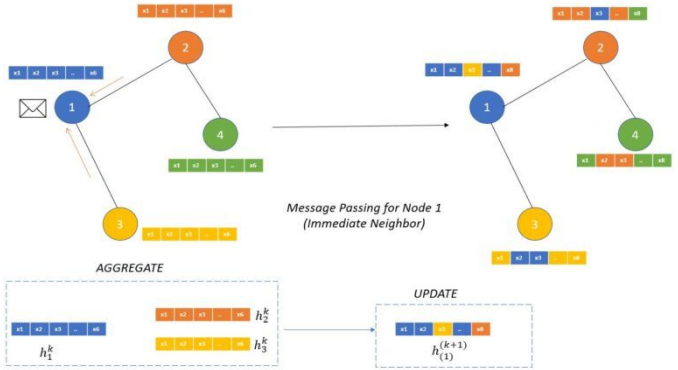


Figure 4: *A representation of GCN message passing; taken from [Sen, 2022]*

F1-Score

The F1-Score is a metric that relies in the harmonic mean of Precision and Recall and is defined as:

$$F = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where:

- **Precision** (Positive Predictive Value) measures how many predicted positives are actually correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall** (Sensitivity) measures how many actual positives are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Table of Contents

Datasets

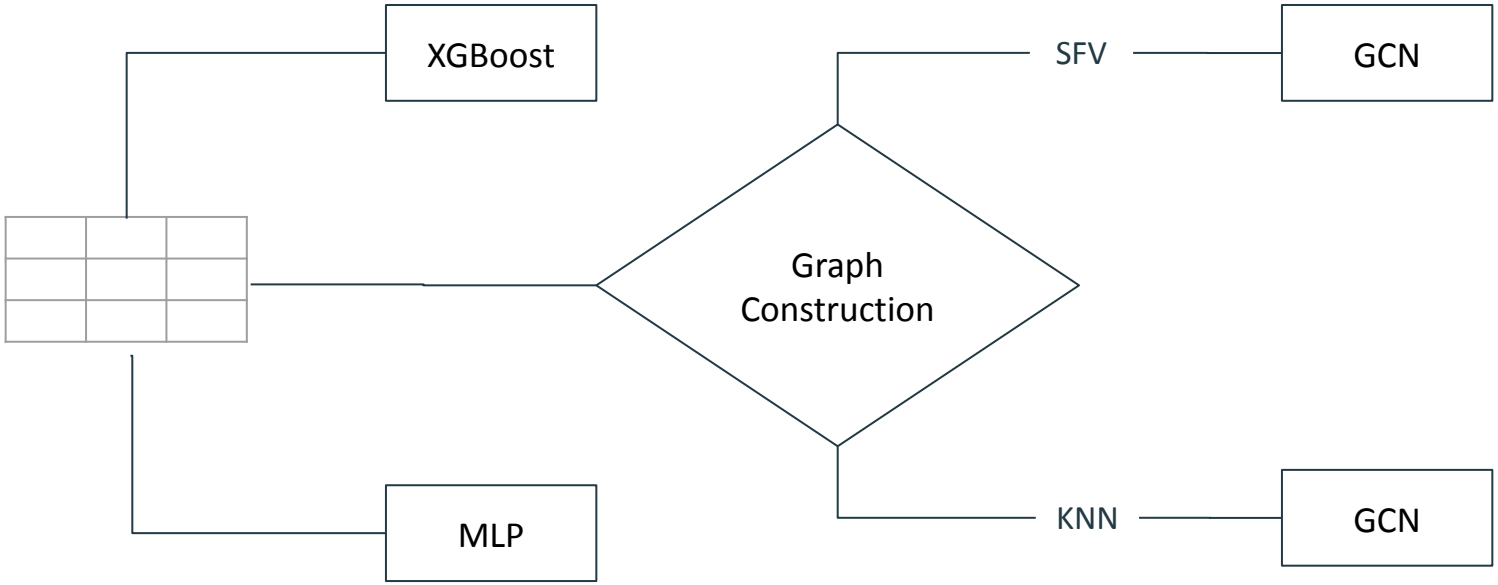
	# samples	# attributes	# categorical	# continuous	# classes
Dry Bean	13611	16	0	16	7
Isolet	7797	617	0	617	26
Musk v2	6598	166	0	166	2
Abalone	4177	8	1	7	28
German Credit Data	1000	20	13	7	2
Car Evaluation	1728	6	6	0	4

Table 1: *Datasets' statistics*

The Dry Bean, Isolet, and Musk v2 datasets were used in a study focused on KNN-based GNN models [Errica, 2024]. The remaining datasets were selected due to their relative content of categorical features.

All datasets were divided into numerical and categorical features, their numerical features were standardized, their categorical features were one-hot encoded, and the target labels were assigned.

Experimental Overview



Hyperparameter Tuning

K-Fold Cross-Validation

Each dataset was divided into 10 fold subsets.

For each fold:

- The fold subset was used as a test set.
- The remainder was used to train the model (90%) training and (10%) validation sets.
- All combinations of parameters were tested. The best hyperparameters combination was determined via the validation F1-score.
- The best performing model was re-trained three times using the full training set and tested on the test set.
- The mean of the three test F1-scores were the result of that fold.
- Repeat the process with the next fold.

Finally, the mean and standard deviation of the F1-scores across all folds was calculated.

Hyperparameter Tuning

Parameter	Values
max depth	4/6/8
alpha	1e-6/1e-2/0.1
lambda	1e-6/1e-2/0.1
eta	0.1/0.2

Table 2: *XGBoost hyperparameters*

The selection of hyperparemeters for the XGBoost Classifier the Tabzilla Benchmark Suite [McElfresh et al., 2024] was used as reference.

Hyperparameter Tuning

Parameter	Values
# units	32/64/128
# layers	1/2/3
learning rate	1e-2/1e-3
epochs	500
weight decay	0/5e-4

Table 3: *MLP and GCN hyperparameters*

The hyperparameters used to train deep learning models were based on previously published work [Errica, 2024].

Table of Contents

KNN Model Selection

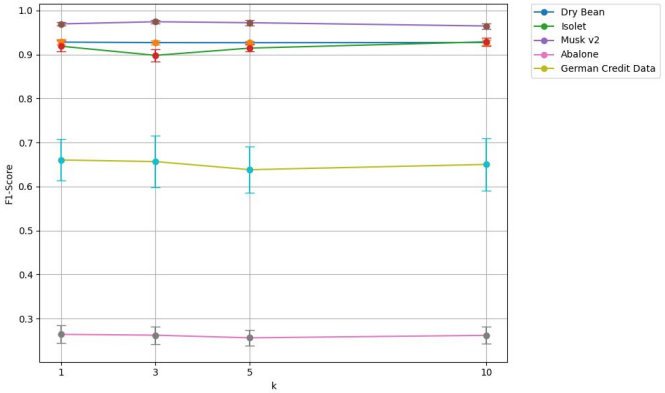


Figure 5: *F1-Scores across k values*

Several KNN graphs with values of 1, 3, 5 and 10 were generated. Each of them was coupled to a GCN and ran a 10-fold cross-validation.

The average F1-scores across the folds were determined, and the resulting F1-score values are plotted in Figure 5.

KNN Model Selection

	k			
	1	3	5	10
Dry Bean	**0.9283** ± 0.0062	0.9272 ± 0.0049	0.9270 ± 0.0033	0.9274 ± 0.0055
Isolet	0.9190 ± 0.0125	0.8982 ± 0.0141	0.9147 ± 0.0073	**0.9289** ± 0.0089
Musk v2	0.9697 ± 0.0045	**0.9746** ± 0.0037	0.9724 ± 0.0060	0.9650 ± 0.0062
Abalone	**0.2641** ± 0.0203	0.2622 ± 0.0200	0.2562 ± 0.0175	0.2618 ± 0.0192
German Credit Data	**0.6604** ± 0.0471	0.6568 ± 0.0582	0.6384 ± 0.0531	0.6503 ± 0.0597

Table 4: *F1-Scores for different k values across datasets. Bold values correspond to the best performing models per dataset*

An unpaired two-samples t-test was carried out between all k-value conditions with each dataset considering 10 sample runs per case and with a threshold of a p-value of 0.05 to consider statistical significance.

The models with the highest F1-score were selected to be used for comparison with the XGBoost and Deep Learning models.

Model Comparison

	XGBoost	MLP	MLP + Dropout	KNN + GCN	SFV + GCN
Dry Bean	**0.9273** ± 0.0067	0.9194 ± 0.0058	**0.9320** ± 0.0084	**0.9283** ± 0.0062	**0.9310** ± 0.0062
Isolet	0.9537 ± 0.0082	0.9527 ± 0.0067	0.9571 ± 0.0109	0.9289 ± 0.0089	**0.9678** ± 0.0051
Musk v2	0.9907 ± 0.0032	**0.9980** ± 0.0018	0.9949 ± 0.0034	0.9746 ± 0.0037	**0.9976** ± 0.0016
Abalone	0.2343 ± 0.0178	0.2043 ± 0.0198	**0.2456** ± 0.0207	**0.2641** ± 0.0203	0.0917 ± 0.0180
German Credit Data	**0.7381** ± 0.0521	**0.7279** ± 0.0581	**0.7329** ± 0.0422	0.6604 ± 0.0471	0.5772 ± 0.0556
Car Evaluation	0.9899 ± 0.0093	**0.9989** ± 0.0023	**0.9985** ± 0.0038	-	0.5774 ± 0.0539

Table 5: *F1-Scores for different models across datasets. Bold values correspond to the best performing models per dataset*

The average F1-scores across the folds were determined, and an unpaired two-samples t-test was performed between all tested models with each dataset considering 10 sample runs per case and with a threshold of a p-value of 0.05 to consider statistical significance.

Model Comparison

	XGBoost	MLP	MLP + Dropout	KNN + GCN	SFV + GCN
Dry Bean	**0.9273** ± 0.0067	0.9194 ± 0.0058	**0.9320** ± 0.0084	**0.9283** ± 0.0062	**0.9310** ± 0.0062
Isolet	0.9537 ± 0.0082	0.9527 ± 0.0067	0.9571 ± 0.0109	0.9289 ± 0.0089	**0.9678** ± 0.0051
Musk v2	0.9907 ± 0.0032	**0.9980** ± 0.0018	0.9949 ± 0.0034	0.9746 ± 0.0037	**0.9976** ± 0.0016
Abalone	0.2343 ± 0.0178	0.2043 ± 0.0198	**0.2456** ± 0.0207	**0.2641** ± 0.0203	0.0917 ± 0.0180
German Credit Data	**0.7381** ± 0.0521	**0.7279** ± 0.0581	**0.7329** ± 0.0422	0.6604 ± 0.0471	0.5772 ± 0.0556
Car Evaluation	0.9899 ± 0.0093	**0.9989** ± 0.0023	**0.9985** ± 0.0038	-	0.5774 ± 0.0539

Table 5: *F1-Scores for different models across datasets. Bold values correspond to the best performing models per dataset*

XGBoost was a top performer model for two datasets. According to published studies [Grinsztajn et al., 2022], tree-based models remain state-of-the-art in medium-sized data of around 10,000 samples.

This superiority is partly explained by the specific features of tabular data. That is, they tend to have irregular patterns in the target function and uninformative features within them.

Model Comparison

	XGBoost	MLP	MLP + Dropout	KNN + GCN	SFV + GCN
Dry Bean	**0.9273** ± 0.0067	0.9194 ± 0.0058	**0.9320** ± 0.0084	**0.9283** ± 0.0062	**0.9310** ± 0.0062
Isolet	0.9537 ± 0.0082	0.9527 ± 0.0067	0.9571 ± 0.0109	0.9289 ± 0.0089	**0.9678** ± 0.0051
Musk v2	0.9907 ± 0.0032	**0.9980** ± 0.0018	0.9949 ± 0.0034	0.9746 ± 0.0037	**0.9976** ± 0.0016
Abalone	0.2343 ± 0.0178	0.2043 ± 0.0198	**0.2456** ± 0.0207	**0.2641** ± 0.0203	0.0917 ± 0.0180
German Credit Data	**0.7381** ± 0.0521	**0.7279** ± 0.0581	**0.7329** ± 0.0422	0.6604 ± 0.0471	0.5772 ± 0.0556
Car Evaluation	0.9899 ± 0.0093	**0.9989** ± 0.0023	**0.9985** ± 0.0038	-	0.5774 ± 0.0539

Table 5: *F1-Scores for different models across datasets. Bold values correspond to the best performing models per dataset*

The MLP and MLP + Dropout models were the highest performers on three and four datasets respectively.

This could be seen consistent with reports of well-regularized plain MLPs significantly outperforming recent state-of-the-art specialized neural network architectures and strong traditional methods such as XGBoost [Kadra et al., 2021].

It is also worth noting that the MLP + Dropout model was a top performer for all datasets that contained categorical features.

Model Comparison

	XGBoost	MLP	MLP + Dropout	KNN + GCN	SFV + GCN
Dry Bean	**0.9273** ± 0.0067	0.9194 ± 0.0058	**0.9320** ± 0.0084	**0.9283** ± 0.0062	**0.9310** ± 0.0062
Isolet	0.9537 ± 0.0082	0.9527 ± 0.0067	0.9571 ± 0.0109	0.9289 ± 0.0089	**0.9678** ± 0.0051
Musk v2	0.9907 ± 0.0032	**0.9980** ± 0.0018	0.9949 ± 0.0034	0.9746 ± 0.0037	**0.9976** ± 0.0016
Abalone	0.2343 ± 0.0178	0.2043 ± 0.0198	**0.2456** ± 0.0207	**0.2641** ± 0.0203	0.0917 ± 0.0180
German Credit Data	**0.7381** ± 0.0521	**0.7279** ± 0.0581	**0.7329** ± 0.0422	0.6604 ± 0.0471	0.5772 ± 0.0556
Car Evaluation	0.9899 ± 0.0093	**0.9989** ± 0.0023	**0.9985** ± 0.0038	-	0.5774 ± 0.0539

Table 5: *F1-Scores for different models across datasets. Bold values correspond to the best performing models per dataset*

The KNN graph construction method coupled to a GCN was a top performer only for the Abalone and Dry Bean datasets. The former having a single categorical feature and the latter having none.

Therefore, the performance of this model did not seem to depend on the number of continuous features but on the dataset for which it was being used

The KNN based model outperformed the SFV-based model when dealing with datasets with categorical features in a statistically significant way.

Model Comparison

	XGBoost	MLP	MLP + Dropout	KNN + GCN	SFV + GCN
Dry Bean	**0.9273** ± 0.0067	0.9194 ± 0.0058	**0.9320** ± 0.0084	**0.9283** ± 0.0062	**0.9310** ± 0.0062
Isolet	0.9537 ± 0.0082	0.9527 ± 0.0067	0.9571 ± 0.0109	0.9289 ± 0.0089	**0.9678** ± 0.0051
Musk v2	0.9907 ± 0.0032	**0.9980** ± 0.0018	0.9949 ± 0.0034	0.9746 ± 0.0037	**0.9976** ± 0.0016
Abalone	0.2343 ± 0.0178	0.2043 ± 0.0198	**0.2456** ± 0.0207	**0.2641** ± 0.0203	0.0917 ± 0.0180
German Credit Data	**0.7381** ± 0.0521	**0.7279** ± 0.0581	**0.7329** ± 0.0422	0.6604 ± 0.0471	0.5772 ± 0.0556
Car Evaluation	0.9899 ± 0.0093	**0.9989** ± 0.0023	**0.9985** ± 0.0038	-	0.5774 ± 0.0539

Table 5: *F1-Scores for different models across datasets. Bold values correspond to the best performing models per dataset*

The SFV graph construction method coupled to a GCN was only a top performer when used on datasets without categorical features.

This can potentially be explained by GCN layers devolving into dropout-regularized fully connected MLP layers where each node only updates its own features without receiving any neighbor information. This happens because a dataset with no categorical features creates an adjacency matrix that is an identity matrix where the normalizing degree matrix plays no role.

The performance of the SFV-based model was unremarkable with datasets that have categorical features.

Table of Contents

Research Questions Revisited

1. Can tabular data be used to define a graph structure that can be effectively used in a GNN for classification purposes?

- Yes, graphs were built using the KNN and SFV approaches. A GCN received them as input.

2. Are categorical features enough to create meaningful graph representations?

- The SFV-based approach produced a viable but not effective model relative to the other models present in this study.

Research Questions Revisited

3. How do graphs derived from categorical features differ from those derived from continuous features?

- The KNN-based approach using categorical features had better performance than the SFV-based approach derived from categorical features.

4. How do these GNN models compare to the algorithms commonly used for this kind of task?

- The SFV-based model was not able to compare favorably with any other model.
- The KNN-based matched the performance of the XGBoost and MLP-based models with some datasets.
- The regularized MLP-based models were the best overall performers within the confines of this study.

Future Work

Meta-features analysis of the data:

- Number of numeric/categorical features
- Skewness of all feature distributions
- Feature correlation
- Ratio of categorical and continuous features

Alternative graph construction methods:

- Thresholding
- Fully connected
- Learning-based
- KNN + SFV hybrid

Thank you for your attention

Appendix - Multilayer Perceptron

Backpropagation is then used to update the weights of the network via gradient descent:

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

where:

- η is the learning rate.

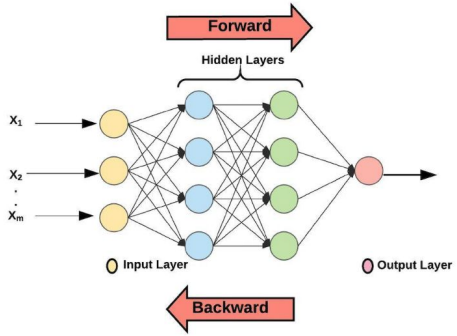


Figure 3: *A depiction of an MLP model; obtained from [Aldosary et al., 2021]*

Appendix - Accuracy

Accuracy measures the proportion of correctly classified instances out of the total instances. Mathematically, it is defined as:

$$\text{Accuracy} = \frac{correct\ classifications}{all\ classifications}$$

Appendix - Accuracy Results

	k			
	1	3	5	10
Dry Bean	0.9283 ± 0.0060	0.9272 ± 0.0048	0.9270 ± 0.0033	0.9274 ± 0.0054
Isolet	0.9189 ± 0.0125	0.8980 ± 0.0141	0.9146 ± 0.0074	0.9288 ± 0.0088
Musk v2	0.9701 ± 0.0043	0.9750 ± 0.0037	0.9730 ± 0.0058	0.9659 ± 0.0060
Abalone	0.2851 ± 0.0223	0.2870 ± 0.0195	0.2803 ± 0.0168	0.2849 ± 0.0177
German Credit Data	0.6800 ± 0.0417	0.6840 ± 0.0568	0.6660 ± 0.0432	0.6850 ± 0.0461

Table A1: *Accuracy scores for different k values across datasets*

	XGBoost	MLP	MLP + Dropout	KNN + GCN	SFV + GCN
Dry Bean	0.9273 ± 0.0065	0.9195 ± 0.0057	0.9322 ± 0.0083	0.9283 ± 0.0060	0.9312 ± 0.0061
Isolet	0.9537 ± 0.0082	0.9525 ± 0.0067	0.9571 ± 0.0110	0.9288 ± 0.0088	0.9677 ± 0.0052
Musk v2	0.9908 ± 0.0031	0.9980 ± 0.0019	0.9949 ± 0.0033	0.9750 ± 0.0037	0.9976 ± 0.0015
Abalone	0.2453 ± 0.0180	0.2071 ± 0.0182	0.2628 ± 0.0211	0.2851 ± 0.0223	0.1927 ± 0.0210
German Credit Data	0.7500 ± 0.0484	0.7320 ± 0.0558	0.7540 ± 0.0347	0.6800 ± 0.0417	0.7000 ± 0.0427
Car Evaluation	0.9896 ± 0.0096	0.9988 ± 0.0023	0.9985 ± 0.0040	-	0.7002 ± 0.0410

Table A2: *Accuracy scores for different models across datasets*

Appendix - F1-Score Study Comparison

	MLP		KNN + GCN	
	F1	ACC	F1	ACC
Dry Bean	$0.78_{\pm 0.04}$	$77.4_{\pm 4.0}$	$0.71_{\pm 0.07}$	$72.6_{\pm 4.9}$
Isolet	$0.95_{\pm 0.01}$	$95.4_{\pm 0.5}$	$0.92_{\pm 0.01}$	$91.7_{\pm 1.4}$
Musk v2	$0.99_{\pm 0.01}$	$99.3_{\pm 0.3}$	$0.94_{\pm 0.02}$	$96.7_{\pm 0.9}$

Table A3: *Experimental results from [Errica, 2024], node classification mean and standard deviation results for an MLP model compared to a KNN graph coupled with a GCN model*

There is a degree of consistency between the results found in this work and their results, particularly with the MLP model using the Isolet and Musk v2 datasets. However, there are differences in the results.

The KNN-based model seems to perform better in this work. A possible explanation might be because the deep learning models are similar but not equivalent in both studies. The same set of hyperparameters were tested in both cases, but differences between the KNN-based models such as the presence of dropout regularization layers are likely to have impacted their performance.

Appendix - Hyperparameter Selection

Dry Bean					
	# hidden units	# layers	learning rate	weight decay	k
MLP	128	3	0.01	0	-
MLP + Dropout	128	3	0.01	0	-
KNN + GCN	128	1	0.01	0	1
SFV + GCN	128	3	0.01	0	-
	max depth	alpha	lambda	eta	
XGBoost	4	0.01	1e-06	0.1	

Table A4: *Most common set of selected hyperparameters for the the Dry Bean dataset*

Isolet					
	# hidden units	# layers	learning rate	weight decay	k
MLP	32	1	0.01	0	-
MLP + Dropout	64	1	0.01	0	-
KNN + GCN	128	1	0.01	0	10
SFV + GCN	128	1	0.01	5e-05	-
	max depth	alpha	lambda	eta	
XGBoost	4	0.1	1e-06	0.2	

Table A5: *Most common set of selected hyperparameters for the the Isolet dataset*

Appendix - Hyperparameter Selection

Musk v2					
	# hidden units	# layers	learning rate	weight decay	k
MLP	32	1	0.01	0	-
MLP + Dropout	32	1	0.01	0	-
KNN + GCN	128	2	0.01	0	3
SFV + GCN	128	1	0.01	0	-
	max depth	alpha	lambda	eta	
XGBoost	6	0.01	1e-06	0.2	

Table A6: *Most common set of selected hyperparameters for the Musk v2 dataset*

Abalone					
	# hidden units	# layers	learning rate	weight decay	k
MLP	128	3	0.01	0	-
MLP + Dropout	128	2	0.01	0	-
KNN + GCN	128	3	0.001	0	1
SFV + GCN	32	2	0.01	0	-
	max depth	alpha	lambda	eta	
XGBoost	4	0.01	0.01	0.2	

Table A7: *Most common set of selected hyperparameters for the Abalone dataset*

Appendix - Hyperparameter Selection

German Credit Data					
	# hidden units	# layers	learning rate	weight decay	k
MLP	32	1	0.01	0	-
MLP + Dropout	64	1	0.01	0	-
KNN + GCN	32	1	0.001	5e-05	1
SFV + GCN	32	1	0.01	0	-
	max depth	alpha	lambda	eta	
XGBoost	4	1e-06	0.01	0.1	

Table A8: *Most common set of selected hyperparameters for the German Credit Data dataset*

Car Evaluation				
	# hidden units	# layers	learning rate	weight decay
MLP	32	1	0.01	0
MLP + Dropout	64	1	0.01	0
SFV + GCN	32	1	0.01	0
	max depth	alpha	lambda	eta
XGBoost	4	1e-06	1e-06	0.2

Table A9: *Most common set of selected hyperparameters for the Car Evaluation dataset*

References

[Aldosary et al., 2021] Aldosary, A. M. et al. (2021). Predictive wireless channel modeling of mmwave bands using machine learning. Electronics, 10(24):3114.

[Aomar, 2021] Aomar, A. A. (2021). Over-smoothing issue in graph neural network. https://towardsdatascience.com/over-smoothing-issue-in-graph-neural-network-bddc8fbc2472/.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794

[Errica et al., 2019] Errica, F., Podda, M., Bacciu, D., and Micheli, A. (2019). A fair comparison of graph neural networks for graph classification. arXiv preprint arXiv:1912.09893.

[Errica, 2024] Errica, F. (2024). On class distributions induced by nearest neighbor graphs for node classification of tabular data. Advances in Neural Information Processing Systems, 36.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning, volume 1. MIT Press, Cambridge.

References

[Grinsztajn et al., 2022] Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems, 35:507–520.

[Guo et al., 2020] Guo, R. et al. (2020). Degradation state recognition of piston pump based on iceemdan and xgboost. Applied Sciences, 10(18):6593.

[James et al., 2023] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2023). An Introduction to Statistical Learning: With Applications in Python. Springer International Publishing, Cham.

[Kadra et al., 2021] Kadra, A., Andr´e, M. S., Kr¨amer, A., and B¨ack, T. (2021). Regularization is all you need: Simple neural nets can excel on tabular data. arXiv preprint arXiv:2106.11189.

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

[Liu et al., 2021] Liu, Y. et al. (2021). Content matters: A gnn-based model combined with text semantics for social network cascade prediction. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, Cham. Springer International Publishing.

References

[Li et al., 2024] Li, C.-T. et al. (2024). Graph neural networks for tabular data learning: A survey with taxonomy and directions. arXiv preprint arXiv:2401.02143.

[Lu and Uddin, 2021] Lu, H. and Uddin, S. (2021). A weighted patient network-based framework for predicting chronic diseases using graph neural networks. Scientific Reports, 11(1):1–12.

[McElfresh et al., 2024] McElfresh, D. et al. (2024). When do neural nets outperform boosted trees on tabular data? Advances in Neural Information Processing Systems, 36.

[Metz, 1978] Metz, C. E. (1978). Basic principles of ROC analysis. Seminars in Nuclear Medicine, 8(4):283–298.

[Roy et al., 2021] Roy, A. et al. (2021). Sst-gnn: Simplified spatio-temporal traffic forecasting model using graph neural network. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, Cham. Springer International Publishing.

[Scarselli et al., 2008] Scarselli, F. et al. (2008). The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80.

References

[Sen, 2022] Sen, A. (2022). Graph neural network message passing (gcn) – part 1. https://www.aritrasen.com/graph-neural-network-message-passing-gcn-1-1/.

[Sun et al., 2022] Sun, R., Dai, H., and Yu, A. W. (2022). Does gnn pretraining help molecular representation? Advances in Neural Information Processing Systems, 35:12096–12109.

[Taha and Hanbury, 2015] Taha, A. A. and Hanbury, A. (2015). Metrics for evaluating 3d medical image segmentation: Analysis, selection, and tool. BMC Medical Imaging, 15:1–28.

[Ucar et al., 2021] Ucar, T., Hajiramezanali, E., and Edwards, L. (2021). Subtab: Subsetting features of tabular data for self-supervised representation learning. Advances in Neural Information Processing Systems, 34:18853–18865.

[Wang et al., 2024] Wang, B., Cai, B., Sheng, J., and Jiao, W. (2024). AAGCN: a graph convolutional neural network with adaptive feature and topology learning. Scientific Reports, 14(1):10134.