Deep Learning for School Dropout Detection: A Comparison of Tabular and Graph-Based Models for Predicting At-Risk Students

Pablo G. Almeida[1], Guilherme A. L. Silva[2], Valéria Santos[1],
Gladston Moreira[1], Pedro Silva[1] and Eduardo Luz[1]
[1]*Computing Department, Federal University of Ouro Preto*, Ouro Preto, Brazil, 35402-136
[2]*Postgraduate Program in Computer Science, Federal University of Ouro Preto*, Ouro Preto, Brazil, 35402-136

***Abstract*—Student dropout is a significant challenge in educational systems worldwide, leading to substantial social and economic costs. Predicting students at risk of dropout allows for timely interventions. While traditional Machine Learning (ML) models operating on tabular data have shown promise, Graph Neural Networks (GNNs) offer a potential advantage by capturing complex relationships inherent in student data if structured as graphs. This paper investigates whether transforming tabular student data into graph structures, primarily using clustering techniques, enhances dropout prediction accuracy. We compare the performance of GNNs (a custom Graph Convolutional Network (GCN) and GraphSAGE) on these generated graphs against established tabular models (Random Forest (RF), XGBoost, and TabNet) using a real-world student dataset. Our experiments explore various graph construction strategies based on different clustering algorithms (K-Means, HDBSCAN) and dimensionality reduction techniques (Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP)). Our findings demonstrate that a specific GNN configuration, GraphSAGE on a graph derived from PCA-KMeans clustering, achieved superior performance, notably improving the macro F1-score by approximately 7 percentage points and accuracy by nearly 2 percentage points over the strongest tabular baseline (XGBoost). However, other GNN configurations and graph construction methods did not consistently surpass tabular models, emphasizing the critical role of the graph generation strategy and GNN architecture selection. This highlights both the potential of GNNs and the challenges in optimally transforming tabular data for graph-based learning in this domain.**

***Index Terms*—Student Dropout Prediction, Graph Neural Networks, Tabular Data, Machine Learning, Educational Data Mining, Clustering.**

I. Introduction

Student dropout represents a critical issue across all educational levels, with particularly alarming rates in higher education in countries like Brazil, where it can reach 57% in public and private institutions [1]. This phenomenon not only curtails individual student potential but also incurs substantial financial losses for educational institutions [1] and wider socio-economic consequences. Early identification of students at risk of dropout is paramount, as it enables institutions to implement targeted interventions and support mechanisms.

[1]https://www.correiobraziliense.com.br/euestudante/ensino-superior/2024/05/6852929-ensino-superior-no-brasil-tem-57-de-evasao-na-rede-publica-e-privada.html

Predicting student dropout typically involves analyzing diverse data, often available in tabular format, encompassing demographic information, academic performance, and engagement metrics. While traditional ML models like RF [2] and XGBoost [3], as well as Deep Learning (DL) models for tabular data like TabNet [4], have achieved considerable success [5], they may not fully capture latent, complex interdependencies between students or student attributes.

GNNs [6] have emerged as powerful tools for learning from graph-structured data, offering the potential to model such relationships more effectively. However, educational data is not inherently graph-structured. Converting tabular data into meaningful graph representations that GNNs can leverage poses a significant challenge [7], [8]. The central research question (RQ1) we address is: *Can transforming tabular student data into graph structures via clustering techniques improve the accuracy of dropout prediction models compared to state-of-the-art tabular methods?* Furthermore (RQ2): *How do different graph construction strategies, based on various clustering and dimensionality reduction combinations, affect GNN performance in this task?*

This paper presents an empirical study to address these questions by proposing and evaluating a comprehensive pipeline. Initially, a real-world student dataset undergoes thorough preprocessing. Subsequently, we employ clustering algorithms, such as K-Means and HDBSCAN, often in conjunction with dimensionality reduction techniques including PCA, UMAP, and t-distributed Stochastic Neighbor Embedding (t-SNE), to group students based on their feature similarity; the quality of these groupings is assessed using the Silhouette Score. Following this, graph representations are constructed from the identified clusters, where nodes correspond to student instances and edges signify co-membership within clusters or proximity in the reduced feature space. These generated graphs then serve as input for training and evaluating GNN models, specifically a custom GCN and GraphSAGE, for the task of predicting student dropout status (Enrolled, Dropout, or Graduate). Finally, the performance of these GNN models is systematically compared against strong baselines that operate directly on the original tabular data, namely RF, XGBoost, and TabNet.

Our experiments, conducted using a publicly available dataset [9], demonstrate that certain configurations of graph neural networks (GNNs) applied to cluster-derived graphs can attain performance levels comparable to, and in one case marginally surpassing, those of traditional tabular models. However, these performance gains are not consistently significant across all employed graph construction techniques. This suggests that simple clustering-based approaches may not always unlock the full potential of GNNs for tabular educational data, and the heterogeneity of such data presents ongoing challenges [8].

The main contributions of this work are:

- A systematic investigation into the efficacy of converting tabular educational data to graph structures using clustering for student dropout prediction.
- An empirical comparison of GNN models against established tabular ML and DL methods on this specific task and dataset.
- An analysis of how different clustering (K-Means, HDBSCAN) and dimensionality reduction (PCA, UMAP) combinations for graph construction impact predictive performance.
- Insights into the challenges and potential limitations of current graph representation learning techniques for heterogeneous tabular educational data.

II. Related Work

The prediction of student dropout has been extensively studied using traditional ML techniques on tabular data. Cambruzzi et al. [5] utilized learning analytics trails, while many other studies have employed models like decision trees, support vector machines, RF [2], and XGBoost [3] with varying degrees of success, often highlighting the importance of feature engineering and handling imbalanced data. More recently, DL architectures for tabular data, such as TabNet [4] and TabTransformer [10], have shown competitive performance by learning feature representations automatically. Our work contrasts by exploring the transformation of tabular data into graphs.

The application of GNNs to tabular data is an emerging research area. Li et al. [6] provides a comprehensive survey, categorizing methods for constructing graphs from tabular data. Common approaches include k-Nearest Neighbors (k-NN) graphs, fully connected graphs with learned edge weights, or graphs derived from domain knowledge. Our focus on explicit clustering as a primary mechanism for graph construction differs from many existing approaches that rely on direct similarity metrics or assume latent graph structures.

Several studies have applied GNNs in educational contexts. Xiaochen et al. [11] introduced a GNN model using structural density-based sampling, leveraging Jaccard coefficient and Pearson correlation to capture similarity between students for performance prediction. While related, their graph construction does not explicitly employ hierarchical clustering. Li et al. [12] proposed a pipeline building multiple graphs with distinct topologies based on similarity metrics and integrating information via attention for student performance prediction. Our work, in contrast, focuses on a direct comparison resulting from cluster-based graph conversion against purely tabular methods.

In [7], authors explore Graph Filtration Learning to create graphs and then filtering them, which is related to the idea of constructing meaningful graph structures. Musa et al. [13] use graph-based approaches for integrating heterogeneous learning analytics data via ontologies. The work proposed in [14] applies network graph analysis to inform learning design, but not for direct dropout prediction. Our contribution lies in the specific investigation of clustering-based conversion for dropout prediction and the direct, comparative evaluation against state-of-the-art tabular models, which is not extensively covered in existing literature for this particular problem formulation.

III. Background

A. Student Dropout Prediction

Student dropout is the premature cessation of studies before course completion. Formally, given a set of students $S = \{s_1, s_2, \ldots, s_N\}$, each student s_i is represented by a feature vector $\mathbf{x}_i \in \mathbb{R}^D$, where D is the number of features (e.g., demographics, academic history, engagement metrics). Each student also has a label $y_i \in \mathcal{Y}$ indicating their status, where $\mathcal{Y} = \{\text{Enrolled, Dropout, Graduate}\}$ in our case. The objective is to learn a predictive function $f : \mathbb{R}^D \to \mathcal{Y}$ that accurately maps student features to their dropout status.

B. Tabular Learning Models

Traditional machine learning models operate directly on these tabular feature vectors $\mathbf{x}_i$. In this work, we will evaluate two traditional machine learning algorithms:

- **RF** [2]: An ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- **XGBoost** [15]: A gradient boosting framework that builds trees sequentially, where each new tree corrects errors made by previously trained trees. It is known for its high performance and regularization capabilities.

We also evaluate a model that is based on Deep Learning. The evaluated architecture is:

- **TabNet** [4]: Architecture designed for tabular data. It uses sequential attention to choose which features to reason from at each decision step, enabling interpretability and efficient learning.

C. Graph Neural Networks (GNNs)

GNNs are a class of neural networks designed to operate directly on graph-structured data [16]. A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (or vertices) and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ can have features $\mathbf{h}_v \in \mathbb{R}^{F_v}$. GNNs typically learns node representations by aggregating information from their neighbors. In this work, we evaluate two main architectures based on graphs:

- **GCN** [17]: A type of GNN that learns node representations by aggregating feature information from their local graph neighborhoods using a spectral approach, simplified to a spatial message-passing scheme. For a layer l, the representation $\mathbf{h}_v^{(l+1)}$ for node v is:

$$\mathbf{h}_v^{(l+1)} = \phi\left(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \frac{1}{\sqrt{\deg(v)\deg(u)}}\mathbf{W}^{(l)}\mathbf{h}_u^{(l)}\right) \quad (1)$$

  where $\mathcal{N}(v)$ is the set of neighbors of node v, $\deg(v)$ is the degree of node v, $\mathbf{W}^{(l)}$ is a trainable weight matrix for layer l, and ϕ is an activation function.
- **GraphSAGE** [18]: An inductive GNN framework that learns functions to generate node embeddings by sampling and aggregating features from a node's local neighborhood. For a node v and layer l:

$$\mathbf{h}_{\mathcal{N}(v)}^{(l+1)} = \text{AGGREGATE}^{(l+1)}\left(\{\mathbf{h}_u^{(l)}, \forall u \in \mathcal{N}(v)\}\right) \quad (2)$$

$$\mathbf{h}_v^{(l+1)} = \phi\left(\mathbf{W}^{(l+1)} \cdot \text{CONCAT}(\mathbf{h}_v^{(l)}, \mathbf{h}_{\mathcal{N}(v)}^{(l+1)})\right) \quad (3)$$

  where AGGREGATE can be a mean, pooling, or LSTM aggregator.

D. Clustering and Dimensionality Reduction

To convert tabular student data into a graph-based representation—with nodes and connections—we first apply dimensionality reduction to project each student instance into a lower-dimensional space. Then, we group similar instances using clustering techniques, enabling the identification of meaningful structures that can later be represented as graph edges.

- **K-Means Clustering (K-Mean)**: An iterative algorithm that partitions N observations into K clusters by minimizing the within-cluster sum of squares. [19]
- **Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)**: A hierarchical, density-based clustering algorithm that extends DBSCAN by converting it into a hierarchical clustering algorithm, then extracting flat clusters based on stability. [20]
- **PCA**: A linear dimensionality reduction technique that transforms data into a new coordinate system such that the greatest variance by some scalar projection of the data lies on the first coordinate (the first principal component), the second greatest variance on the second coordinate, and so on. [21]
- **UMAP**: A non-linear dimensionality reduction technique that constructs a high-dimensional graph representation of the data, then optimizes a low-dimensional graph to be as structurally similar as possible. [22]
- **T-SNE**:A non-linear dimensionality reduction technique that models pairwise similarities between points in high dimensions and maps them to a low-dimensional space, preserving local structure for visualization. [23]

E. Problem Setting and Notation

Let $\mathcal{D}_{tab} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be the initial tabular dataset, where $\mathbf{x}_i \in \mathbb{R}^D$ is the feature vector for student i and $y_i \in \mathcal{Y}$ is their corresponding dropout status. Our primary hypothesis (H1) is that converting $\mathcal{D}_{tab}$ into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ represents students and edges $\mathcal{E}$ encode relationships derived from clustering, allows GNNs to achieve superior predictive performance compared to models operating on $\mathcal{D}_{tab}$.

The graph construction process involves: 1. Optional dimensionality reduction: $\mathbf{x}_i \to \mathbf{x}'_i \in \mathbb{R}^{D'}$, where $D' < D$. 2. Clustering: Assign each student $\mathbf{x}'_i$ (or $\mathbf{x}_i$) to a cluster $c_k \in \{C_1, \ldots, C_K\}$. 3. Edge creation: An edge $(v_i, v_j) \in \mathcal{E}$ exists if students s_i and s_j belong to the same cluster c_k. Variations can include connecting students if they are close in the reduced latent space (e.g., using KD-Trees). Node features for v_i in $\mathcal{G}$ are the original $\mathbf{x}_i$.

The task is then to train a GNN model $f_{GNN}(\mathcal{G}, \mathbf{X})$ to predict y_i for nodes v_i, where $\mathbf{X}$ is the matrix of node features. We compare $f_{GNN}(\mathbf{X})$ against tabular models $f_{tab}(\mathbf{X})$. This directly addresses RQ1 and RQ2.

IV. Experimental Methodology

This section details our approach for converting tabular student data into graph structures and applying GNNs for dropout prediction, along with the setup for baseline tabular models.

A. Data Acquisition and Preprocessing

The primary dataset used in this study is from the "Predict Students' Dropout and Academic Success" dataset [24] [2]. It contains anonymized features related to student demographics, socio-economic background, and academic performance for a higher education institution, with the target variable indicating whether a student is "Enrolled", "Dropout", or "Graduate".

The preprocessing steps starts with handling missing data by checking for null values using `data_df.isnull().sum()`; columns without missing data were retained, though imputation (e.g., by mean, median, or mode) would have been considered if necessary. Subsequently, header inconsistencies were corrected using `data_df.rename(columns={...}, inplace=True)` to standardize names like "Nacionality" to "Nationality" and "Daytime/evening attendance
t" to "Daytime attendance". Categorical feature encoding was then performed by applying One-Hot Encoding to variables such as {`Marital status`, `Application mode`, `Course`, `...`} via `pd.get_dummies()`, and boolean values were converted to 0/1 integers using a `map` function. Following this, continuous numerical features like {`Previous qualification (grade)`, `Admission grade`, `...`} were normalized to the $[0, 1]$ range using `MinMaxScaler()`. The

[2]https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success

multi-class `Target` variable ("Enrolled", "Dropout", "Graduate") was then converted to numeric labels ($\{0 \rightarrow$ Dropout, $1 \rightarrow$ Enrolled, $2 \rightarrow$ Graduate$\}$) with `LabelEncoder()`. Finally, the dataset was split into training, validation, and test sets using `train_test_split(..., stratify=y)` to preserve class proportions, allocating 80% of the data for training and validation (which was further split $75\%/25\%$ respectively) and 20% for the final test set, ensuring stratified splitting throughout.

B. Graph Construction via Clustering

We explored two main strategies for constructing graphs from the tabular student data:

Strategy 1: Silhouette Score Optimized Clustering:

1) **Dimensionality Reduction and Clustering Combinations**: We experimented with various combinations of dimensionality reduction techniques (PCA, UMAP, t-SNE, and PCA followed by UMAP or t-SNE) and clustering algorithms (K-Means and HDBSCAN).
2) **Cluster Validation**: For each combination, the quality of the resulting clusters was evaluated using the Silhouette Score. This metric measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Scores range from -1 to +1, with higher values indicating better-defined clusters.
3) **Optimal Configuration Selection**: The combination yielding the highest Silhouette Score was selected for primary graph construction. In our experiments, this was UMAP (to 10 components) followed by HDBSCAN clustering (with euclidean distance).
4) **Graph Generation**: Nodes represent students. An undirected edge connects two student nodes if they are assigned to the same cluster by the HDBSCAN configuration. Outlier nodes would not form edges based on shared clusters or could be isolated. The original student features (after preprocessing) were used as node features.

Strategy 2: Fixed Configuration for GNN Benchmarking:

1) **Fixed Dimensionality Reduction**: Data was projected into a 2-dimensional space using PCA followed by UMAP. This combination is often used for visualization and can reveal underlying manifold structures.
2) **Graph Generation via Proximity**: A graph was constructed based on proximity in this 2D projected space. Edges were established between nodes (students) if they were among the $k = 5$ nearest neighbors of each other, identified using a KD-Tree. This creates a graph based on local similarity in the embedded space, rather than explicit cluster membership.
3) Node features remained the original preprocessed student features.

The properties of the resulting graphs, such as the number of connected components and graph density, were analyzed.

C. Graph Neural Network Models

We implemented two GNN architectures using PyTorch Geometric [25]:

1. Custom GCN Model: The architecture comprised three GCN layers (`GCNConv`). Each convolutional layer was followed by `BatchNorm1d` for stabilization, a `LeakyReLU` activation function, and `Dropout` for regularization. The output from the final GCN layer was passed through a fully connected layer to produce class predictions.

2. GraphSAGE Model: This model also used three convolutional layers, but of type `SAGEConv`. Similar to the custom GCN, each `SAGEConv` layer was followed by `BatchNorm1d` and `ReLU` activation, and `Dropout`. A final fully connected layer performed the classification.

Training and Optimization: Both GNN models were trained using the Adam optimizer [26] and Cross-Entropy Loss [26]. Hyperparameters (hidden dimension size, learning rate, dropout rate) were optimized using Optuna [27], performing a specified number of trials. Early stopping was employed based on validation loss to prevent overfitting, with a patience of 15 epochs.

D. Tabular Baseline Models

To provide a strong comparative baseline, we trained and evaluated three established models directly on the preprocessed tabular data:

- **RF and XGBoost**: Trained using 5-fold cross-validation. Hyperparameters were optimized via random search over 5 combinations for each model, aiming to maximize performance on the validation folds.
- **TabNet**: Model was configured with several key hyperparameters. These included `n_d` and `n_a` to set the widths of the decision prediction and attention embedding layers, respectively, within its attentive transformers. The number of sequential decision stages was defined by `n_steps`, while `gamma` controlled feature selection sparsity. The architecture also incorporated `n_independent` and `n_shared` Gated Linear Unit (GLU) layers, applied independently to features and shared across features within GLU blocks, respectively. The `mask_type` (e.g., `sparsemax` or `entmax`) dictated the feature selection masking strategy. Training utilized an Adam optimizer, a learning rate scheduler, and early stopping with a 20% validation split, following a public Kaggle implementation[3]. It was trained using an Adam optimizer, a learning rate scheduler, and early stopping based on validation performance (20% validation split). [4]

E. Evaluation Metrics

All models were evaluated on the held-out test set using standard classification metrics. For a multi-class classification problem with classes $C_1, \ldots, C_K$, let TP_k, FP_k, TN_k, and

[3]https://www.kaggle.com/code/ar2197/tabnet-multi-class-and-binary-classification

[4]Configured based on a public Kaggle implementationhttps://www.kaggle.com/code/ar2197/tabnet-multi-class-and-binary-classification

FN_k denote the number of True Positives, False Positives, True Negatives, and False Negatives for class C_k, respectively.

The **Accuracy** measures the overall proportion of correctly classified instances across all classes and is defined as:

$$\text{Accuracy} = \frac{\sum_{k=1}^{K} TP_k}{\text{Total number of instances}} \tag{4}$$

For each class C_k, **Precision**$_k$ quantifies the ability of the classifier not to label a sample as positive if it is negative for that class. It is calculated as:

$$\text{Precision}_k = \frac{TP_k}{TP_k + FP_k} \tag{5}$$

Similarly, **Recall**$_k$ (or Sensitivity) for class C_k measures the ability of the classifier to find all positive samples belonging to that class:

$$\text{Recall}_k = \frac{TP_k}{TP_k + FN_k} \tag{6}$$

The **F1-Score**$_k$ for class C_k is the harmonic mean of Precision$_k$ and Recall$_k$, providing a balance between them:

$$\text{F1-Score}_k = 2 \cdot \frac{\text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \tag{7}$$

To obtain an overall performance measure that accounts for all classes equally, especially given the potential class imbalance identified in Figure 1, we also report macro-averaged versions of Precision, Recall, and F1-Score. Macro-averaging involves calculating the metric independently for each class and then taking the unweighted mean of these per-class scores. For example, Macro F1-Score is:

$$\text{Macro F1-Score} = \frac{1}{K} \sum_{k=1}^{K} \text{F1-Score}_k \tag{8}$$

Macro Precision and Macro Recall are calculated analogously. These macro-averaged metrics are particularly important as they give equal weight to each class, regardless of its frequency.

V. Experimental Setup

A. Dataset

After preprocessing, the dataset consisted of 4424 instances and 250 features. The target variable had three classes: "Dropout", "Graduate", and "Enrolled". As shown in Figure 1, the dataset exhibits class imbalance, with the "Graduate" class being the most frequent, followed by "Dropout", and "Enrolled" being the least frequent. This imbalance was considered during model training and evaluation (focus on macro-averaged metrics).

B. Implementation Details

All experiments were conducted using **Python**, **Optuna** [27] for hyperparameter optimization and **NetworkX** [28] for graph manipulation, along with **PyTorch Geometric**. Hyperparameter tuning for GNNs explored learning rates typically in the range $[10^{-4}, 10^{-1}]$, hidden dimensions from 16 to 256, and dropout rates from 0.1 to 0.5. For tabular models, ranges were set based on common practices for RF and XGBoost.

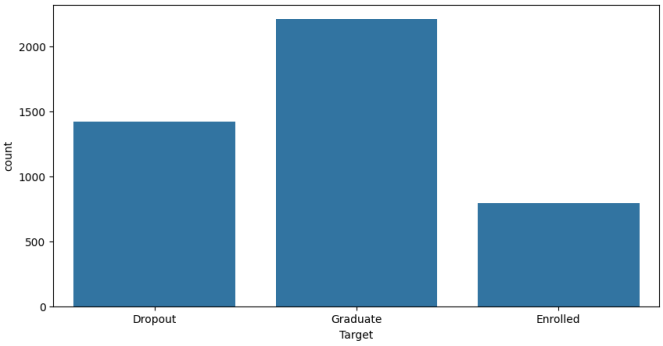

Fig. 1. Distribution of target classes in the student dataset

VI. Results and Discussion

A. Performance of Tabular Baselines

The performance of the traditional ML and DL models on the tabular data is summarized in Table I. RF and XGBoost achieved competitive macro-averaged F1-scores (0.69 for XGBoost, 0.66 for RF) and overall accuracy (0.76). XGBoost demonstrated slightly better precision and recall balance. TabNet, the DL baseline for tabular data, performed slightly lower, with a macro F1-score of 0.64 and accuracy of 0.73. Notably, all tabular models struggled with the Dropout class (class 0), achieving F1-scores between 0.36 (TabNet) and 0.49 (XGBoost). This highlights the difficulty of predicting this specific class, likely due to its minority status or more complex patterns.

TABLE I
Performance of Tabular Models

Model	Class/Avg	Precision	Recall	F1-Score	Accuracy
Random Forest	Dropout (0)	0.57	0.27	0.37	0.76
	Enrolled (1)	0.78	0.77	0.78	
	Graduate (2)	0.78	0.93	0.85	
	Macro Avg	0.71	0.66	0.66	
XGBoost	Dropout (0)	0.52	0.42	0.46	0.76
	Enrolled (1)	0.78	0.75	0.76	
	Graduate (2)	0.81	0.89	0.85	
	Macro Avg	0.70	0.69	0.69	
TabNet	Dropout (0)	0.48	0.29	0.36	0.73
	Enrolled (1)	0.77	0.70	0.73	
	Graduate (2)	0.76	0.91	0.83	
	Macro Avg	0.67	0.63	0.64	

B. Performance of GNN Models

We evaluated the GNN models on graphs constructed using the two proposed strategies.

GNNs on Fixed PCA+UMAP Graph: Table II reports the results for this configuration. GraphSAGE achieved the best overall performance, with an accuracy of 0.7440 and a macro F1-score of 0.7583. In contrast, the custom GCN attained 0.7154 accuracy and a macro F1-score of 0.6884. As shown in Figure 2, GraphSAGE drives its validation loss down very quickly—within the first 20–25 epochs, whereas the custom

GCN only begins to settle around epoch 50. On the accuracy side, GraphSAGE peaks at roughly 0.74 – 0.75 by epoch 25 before exhibiting some oscillation, whereas the GCN gradually climbs and ultimately plateaus around 0.71 after roughly 60 epochs. Thus, GraphSAGE not only converges significantly faster but also achieves a modestly higher maximum validation accuracy compared to the custom GCN

TABLE II
GNN PERFORMANCE ON FIXED PCA+UMAP GRAPH - STRATEGY 1

Model	Accuracy	Precision	Recall	F1-Score
GCN_Gmod	0.7154	0.6941	0.7154	0.6884
SAGE_Gmod	0.7440	0.7880	0.7440	0.7583

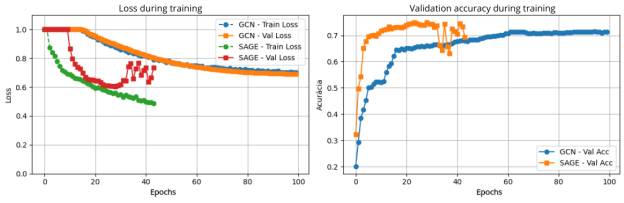

Fig. 2. Training loss and validation accuracy for GNN models on strategy 1 (PCA+UMAP graph)

GNNs on Silhouette-Optimized UMAP (10) + HDBSCAN Graph (*G*)**:** Table III presents results for the graph constructed using UMAP (10) and HDBSCAN (Silhouette Score: 0.8256). Again, GraphSAGE outperformed GCN, achieving an accuracy of 0.7756 and a macro F1-score of 0.7658. The GCN reached 0.6265 accuracy and a 0.5565 F1-score. This GraphSAGE result represents the best performance among all models tested.

TABLE III
GNN PERFORMANCE ON SILHOUETTE-OPTIMIZED UMAP (10) + HDBSCAN GRAPH - STRATEGY 2

Model	Accuracy	Precision	Recall	F1-Score
GCN_G	0.6265	0.6041	0.6265	0.5565
SAGE_G	0.7756	0.7747	0.7756	0.7658

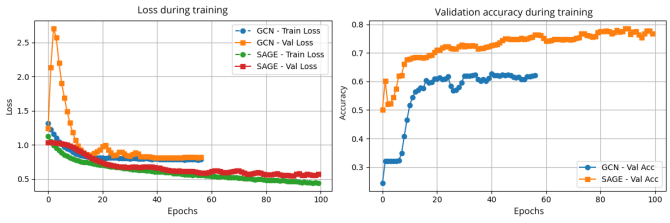

Fig. 3. Training loss and validation accuracy for GNN models on strategy 2 (UMAP(10) + HDBSCAN graph)

C. Comparative Analysis and Discussion

Table IV provides an overall comparison. Among all configurations, the GraphSAGE model trained on the UMAP(10)+HDBSCAN graph achieved the highest macro F1-score (76.58%) and accuracy (77.56%).

TABLE IV
OVERALL MODEL PERFORMANCE COMPARISON

Model	Accuracy	Precision	Recall	F1-Score
Random Forest (ML Tabular)	76.16%	74.15%	76.16%	73.88%
XGBoost (ML Tabular)	76.05%	74.95%	76.05%	75.30%
TabNet (DL Tabular)	73.11%	66.97%	63.28%	64.01%
GCN (UMAP (10) + HDBSCAN)	62.66%	60.41%	62.66%	55.65%
GraphSAGE (UMAP (10) + HDBSCAN)	**77.56%**	77.47%	**77.56%**	**76.58%**
GCN (PCA + UMAP)	71.54%	69.41%	71.54%	68.84%
GraphSAGE (PCA + UMAP)	74.40%	**78.80%**	74.40%	75.83%

Addressing Research Questions:

RQ1 (Can graph conversion improve prediction?): Yes, but only modestly. GraphSAGE trained on the UMAP(10)+HDBSCAN graph achieves the highest validation accuracy (around 0.78) and lowest validation loss (0.8) after convergence, as seen in Figures 4 and 5. This configuration yields a macro F1-score of 76.58% (accuracy 77.56%), roughly 1.3 percentage points higher than the best tabular baseline, XGBoost (75.30% F1, 76.05% accuracy). Other GNN variants—such as GCN on UMAP+HDBSCAN—suffer from slower accuracy growth and higher validation loss (e.g., GCN_G peaks near 0.62 accuracy), while GraphSAGE on PCA+UMAP (SAGE_Gmod) converges to an F1-score of 75.83% (0.71 accuracy). In short, although graph-based models can outperform tabular ones, the gain is relatively small and strongly depends on both architecture and graph construction strategy.

RQ2 (Impact of graph-construction strategy?): The graph-construction method has a decisive impact on GNN performance. UMAP(10)+HDBSCAN produces a graph in which GraphSAGE (SAGE_G) quickly reaches a stable validation accuracy above 0.75 and validation loss below 1.0 by epoch 20, whereas PCA+UMAP (SAGE_Gmod) converges more gradually to around 0.71 accuracy and 0.75 loss by epoch 100 (Figures 4–5). In both embedding strategies, GraphSAGE outperforms our custom GCN: GCN_G (UMAP+HDBSCAN) flattens around 0.62 accuracy and suffers higher loss, and GCN_Gmod (PCA+UMAP) plateaus near 0.70 accuracy with loss 0.9. These dynamics highlight that higher-quality clustering and dimensionality reduction (UMAP + HDBSCAN) yield more discriminative graph structures, and that GraphSAGE's sampling/aggregation mechanism is more robust on these generated graphs compared to our GCN.

VII. CONCLUSION

This study evaluated the conversion of tabular student data into graph representations for dropout prediction using various GNN architectures and graph-construction methods. A GraphSAGE model applied to a graph built via UMAP (10 dimensions) followed by HDBSCAN clustering achieved the highest performance, with an F1-score of 76.58%. This slightly surpasses the best tabular baseline (XGBoost, F1 = 75.30%) and outperforms Random Forest (F1 = 73.88%) and TabNet

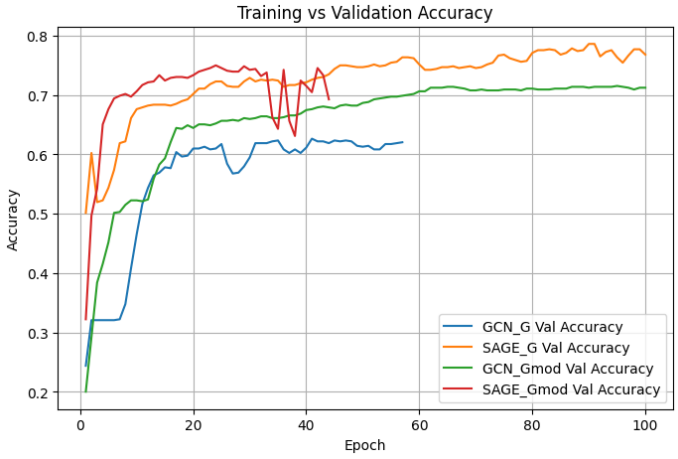

Fig. 4. Training versus Validation Accuracy over 100 Epochs. GCN_G and SAGE_G use UMAP(10) + HDBSCAN; GCN_Gmod and SAGE_Gmod use PCA + UMAP.

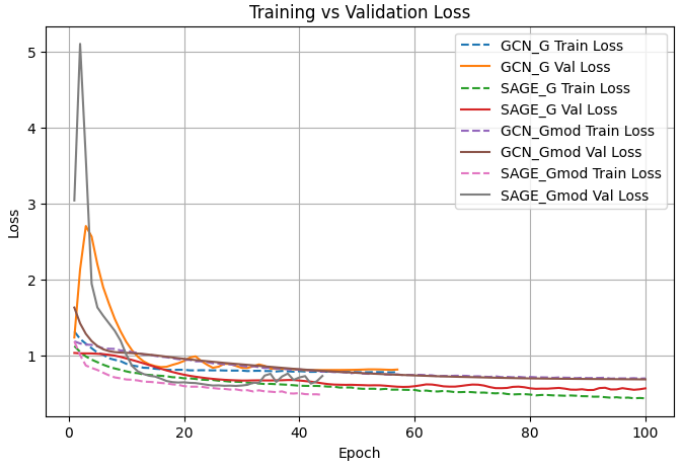

Fig. 5. Training versus Validation Loss over 100 Epochs. GCN_G and SAGE_G use UMAP(10) + HDBSCAN; GCN_Gmod and SAGE_Gmod use PCA + UMAP.

(F1 = 64.01%). Even when using PCA + UMAP for graph construction, GraphSAGE reached an F1 of 75.83%, demonstrating that appropriately engineered graphs allow GNNs to match or exceed classical ML models on tabular features.

In contrast, GCNs performed substantially worse—achieving only 55.65% F1 with UMAP (10) + HDBSCAN and 68.84% F1 with PCA + UMAP—highlighting the importance of both the GNN architecture and the graph-generation strategy. Overall, GraphSAGE with UMAP + HDBSCAN stands out as the most promising approach by delivering balanced precision and recall (77.47% and 77.56%, respectively) and improving macro F1 over XGBoost. Future work should explore more sophisticated graph-construction techniques—such as learnable edges or domain-informed connectivity—together with alternative GNN architectures and pre-training schemes to further enhance dropout-prediction performance.

Acknowledgment

This work was supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, grants 308400/2022-4, 307151/2022-0), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES - grant 001), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG, grant APQ-01647-22). We also thank the Universidade Federal de Ouro Preto (UFOP) for their support.

References

[1] D. B. d. Silva, A. A. d. O. Ferre, P. d. S. Guimarães, R. d. Lima, and I. B. Espindola, "Evasão no ensino superior público do brasil: estudo de caso da universidade de são paulo," *Avaliação: Revista da Avaliação da Educação Superior (Campinas)*, vol. 27, no. 02, pp. 248–259, 2022. 1

[2] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001. 1, 2

[3] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000. 1, 2

[4] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 6679–6687. 1, 2

[5] W. L. Cambruzzi, S. J. Rigo, and J. L. Barbosa, "Dropout prediction and reduction in distance education courses with the learning analytics multitrail approach." *J. Univers. Comput. Sci.*, vol. 21, no. 1, pp. 23–47, 2015. 1, 2

[6] C.-T. Li, Y.-C. Tsai, C.-Y. Chen, and J. C. Liao, "Graph neural networks for tabular data learning: A survey with taxonomy and directions," *arXiv preprint arXiv:2401.02143*, 2024. 1, 2

[7] C. Hofer, F. Graf, B. Rieck, M. Niethammer, and R. Kwitt, "Graph filtration learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4314–4323. [Online]. Available: https://proceedings.mlr.press/v119/hofer20b.html 1, 2

[8] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, "Saint: Improved neural networks for tabular data via row attention and contrastive pre-training," 2021. [Online]. Available: https://arxiv.org/abs/2106.01342 1, 2

[9] V. Realinho, M. Vieira Martins, J. Machado, and L. Baptista, "Predict Students' Dropout and Academic Success," UCI Machine Learning Repository, 2021, DOI: https://doi.org/10.24432/C5MC89. 2

[10] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, "Tabtransformer: Tabular data modeling using contextual embeddings," 2020. [Online]. Available: https://arxiv.org/abs/2012.06678 2

[11] L. Xiaochen, H. Wentao, Z. Zheng, and B. Xiaohan, "Predicting student performance with graph structure density-based graph neural networks," in *Proceedings of 2024 International Conference on Machine Learning and Intelligent Computing*, ser. Proceedings of Machine Learning Research, Z. Nianyin and R. B. Pachori, Eds., vol. 245. PMLR, 26–28 Apr 2024, pp. 25–32. [Online]. Available: https://proceedings.mlr.press/v245/xiaochen24a.html 2

[12] M. Li, X. Wang, Y. Wang, Y. Chen, and Y. Chen, "Study-gnn: A novel pipeline for student performance prediction based on multi-topology graph neural networks," *Sustainability*, vol. 14, p. 7965, 2022. 2

[13] M. H. Musa, S. Salam, S. F. A. Fesol, M. S. Shabarudin, J. F. Rusdi, M. A. Norasikin, and I. Ahmad, "Integrating and retrieving learning analytics data from heterogeneous platforms using ontology alignment: Graph-based approach," *MethodsX*, vol. 14, p. 103092, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215016124005430 2

[14] D. Ifenthaler, D. Gibson, and E. Dobozy, "Informing learning design through analytics: Applying network graph analysis," *Australasian Journal of Educational Technology*, vol. 34, no. 2, Apr. 2018. [Online]. Available: https://ajet.org.au/index.php/AJET/article/view/3767 2

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794. 2

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020. 2
[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016. 3
[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017. 3
[19] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5. University of California press, 1967, pp. 281–298. 3
[20] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2013, pp. 160–172. 3
[21] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933. 3
[22] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426 3
[23] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html 3
[24] M. V. Martins, D. Tolledo, J. Machado, L. M. Baptista, and V. Realinho, "Early prediction of student's performance in higher education: A case study," in *Trends and Applications in Information Systems and Technologies: Volume 1 9*. Springer, 2021, pp. 166–175. 3
[25] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019. 4
[26] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into deep learning*. Cambridge University Press, 2023. 4
[27] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019. [Online]. Available: https://arxiv.org/abs/1907.10902 4, 5
[28] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), Tech. Rep., 2008. 5