IT414 - Data Warehousing and Data Mining

Outlier detection using Unsupervised Graph Neural Network

Team 9 Members

➢	Pratham Nayak	- 191IT241
➢	Aprameya Dash	- 191IT209
➢	Sarthak Jain	- 191IT145

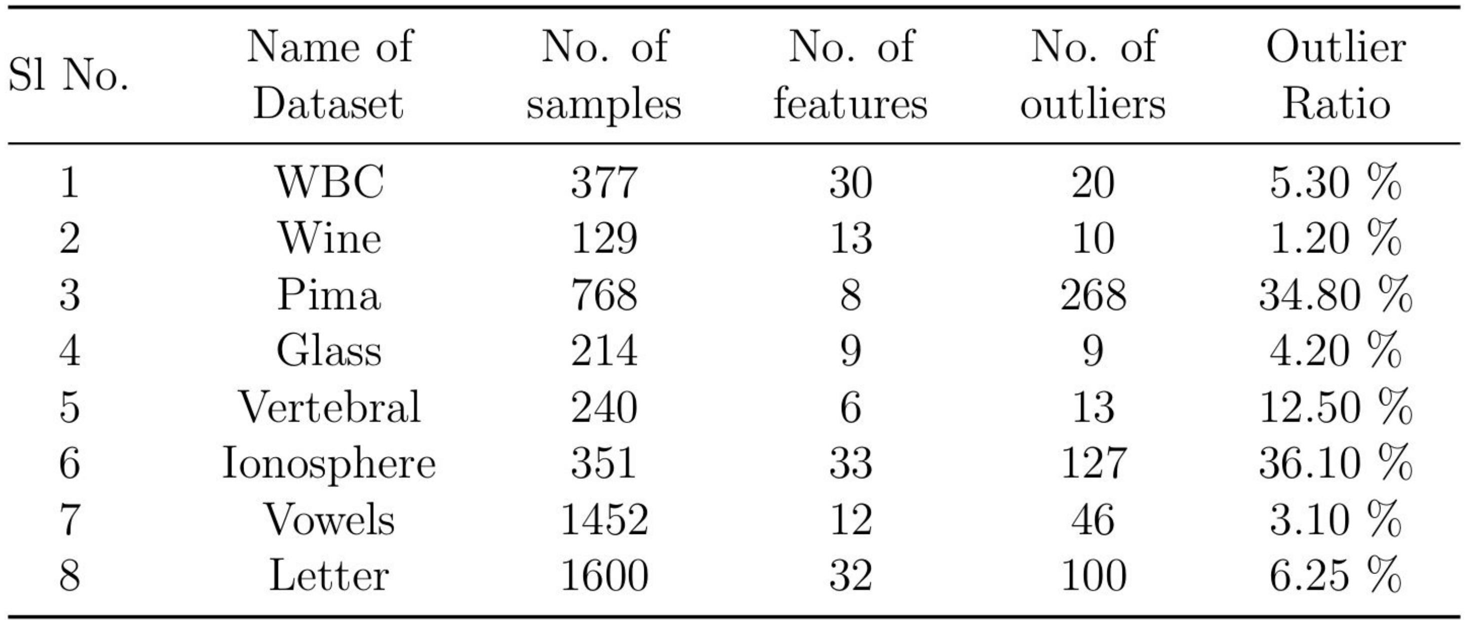
Introduction

- Outlier detection refers to detection of objects that behave abnormally in comparison to other objects. Most existing outlier detection algorithms have difficulty detecting outliers that are mixed within normal object regions or around dense clusters.
- GNNs are neural networks that can be directly applied to graphs, and provide an easy way to do node-level, edge-level, and graph-level prediction tasks.
- GraphSAGE is a framework for inductive representation learning on large graphs. GraphSAGE is used to generate low-dimensional vector representations for nodes, and is especially useful for graphs that have rich node attribute information.
- This project develops a unsupervised learning based technique to perform outlier detection using a modified version of GraphSAGE that incorporates edge weights.

Problem Statement

To identify the outliers using techniques based on unsupervised learning for a given a set of data points.

Datasets



Sl No.	Name of Dataset	No. of samples	No. of features	No. of outliers	Outlier Ratio
1	WBC	377	30	20	5.30 %
2	Wine	129	13	10	1.20 %
3	Pima	768	8	268	34.80 %
4	Glass	214	9	9	4.20 %
5	Vertebral	240	6	13	12.50 %
6	Ionosphere	351	33	127	36.10 %
7	Vowels	1452	12	46	3.10 %
8	Letter	1600	32	100	6.25 %

Table 1: Dataset Description

Metrics

AUC Score

- It is the area under the receiver operating characteristic (ROC) curve.
- In the case of an outlier detection task, an ROC curve is drawn by plotting the FPR on the x-axis and TPR on the y-axis by varying the contamination level from 0 to 1.
- True Positive Rate (TPR) is the fraction of positive samples that are correctly classified.
- False Positive Rate (FPR) is the fraction of negative samples that are incorrectly classified.

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

Methodology

Graph Representation

- The given set of data points is represented using a graph structure by interpreting the points as the vertices of a graph. For each pair of vertices/data points, an edge is added between them.
- These edges are weighted and directed edges with an edge weight that is computed using the cosine similarity between the vector representation of the two data points.

$$similarity_{cosine}(a, b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

Methodology

Graph Sampling

- Sampling of a graph refers to selecting a suitable subgraph that will be used to train the model.
- One way of sampling is by randomly sampling a fixed number of neighbours for each node.
- Another way of sampling is by sampling the neighbours of a node based on some metric/measure. Since we have a weighted graph, the weights of the edges are used to sample the neighbourhood of any given node. Let K be a constant that defines the number of sampled neighbours. Then for each node, the K adjacent nodes with the highest cosine similarity are sampled. Hence after sampling, the complete graph is transformed into a graph where each node has an in-degree of K.

Methodology (Cont..)

GraphSAGE

- Let the neighbourhood of node v be given by N(v) and $h^{(l)}_v$ be the hidden layer representation of node v in the lth layer. The graph convolution operator used for neighbourhood aggregation in the lth GraphSAGE layer when summation-based aggregation is used is given by the following equation.

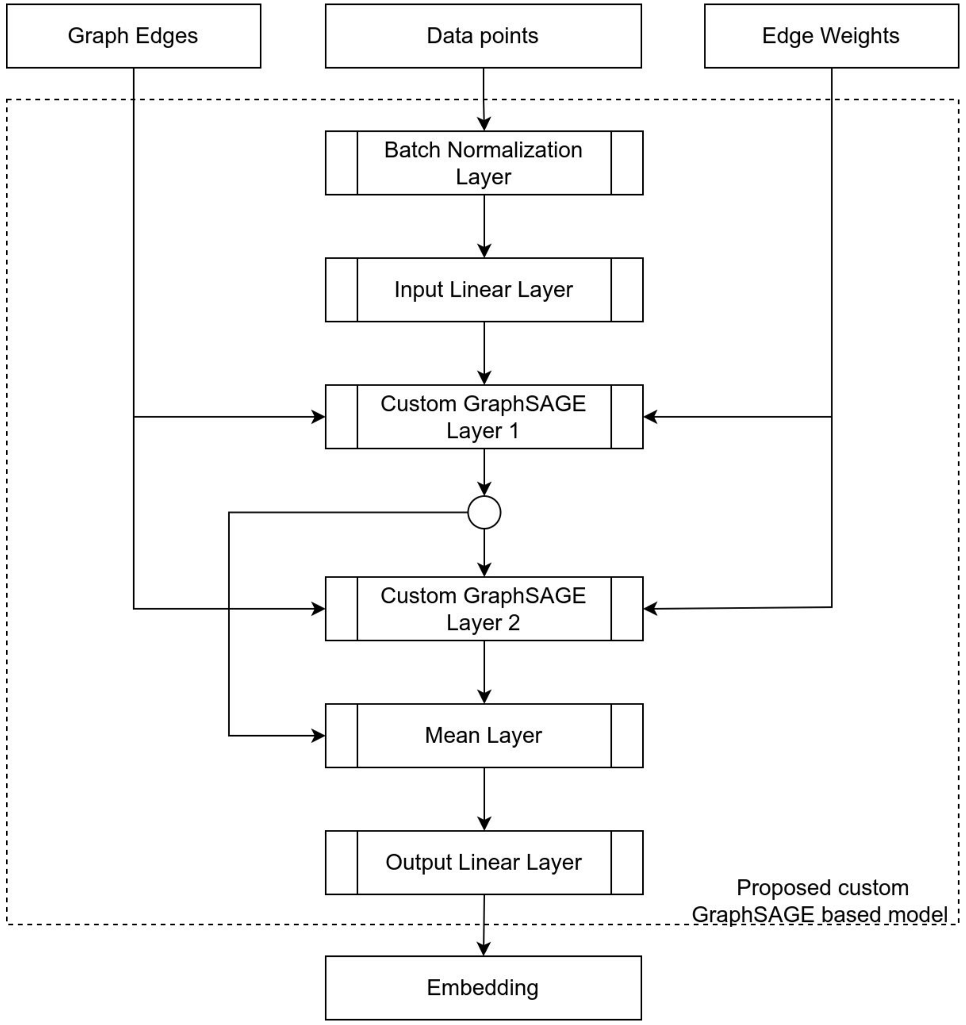
$$h_v^{(l)} = \sigma(W_{self}^{(l)} \cdot h_v^{(l-1)} + W_{neigh}^{(l)} \cdot \sum_{u \in N(v)} h_u^{(l-1)} + b^{(l)})$$

Methodology (Cont..)

Modified GraphSAGE

- Let e_{uv} be the weight of the edge between node u and v, then using the modified equations, the embedding of the node v in the l^{th} layer is given by the following equations.

$$h_v^{(l)} = \sigma(W_{self}^{(l)} \cdot h_v^{(l-1)} + W_{neigh}^{(l)} \cdot \sum_{u \in N(v)} e_{uv} \cdot h_u^{(l-1)} + b^{(l)})$$



$$h_v^{(1)} = \frac{x_v - E[V]}{\sqrt{Var[V]}} * \gamma + \beta$$

$$h_v^{(2)} = \sigma(h_v^{(1)} \cdot W^{(2)} + b^{(2)})$$

$$h_v^{(3)} = \sigma(W_{self}^{(3)} \cdot h_v^{(2)} + W_{neigh}^{(3)} \cdot \sum_{u \in N(v)} e_{uv} \cdot h_u^{(2)} + b^{(3)})$$

$$h_v^{(4)} = \sigma(W_{self}^{(4)} \cdot h_v^{(3)} + W_{neigh}^{(4)} \cdot \sum_{u \in N(v)} e_{uv} \cdot h_u^{(3)} + b^{(4)})$$

$$h_v = \sigma(\frac{(h_v^{(3)} + h_v^{(4)})}{2} \cdot W^{(5)} + b^{(5)})$$

Fig 1: GNN Architecture

Methodology (Cont..)

Unsupervised Learning

- The parameters of the proposed GNN model are trained using an auto encoder-inspired, unsupervised loss function given by:

$$Loss(x_v, h_v) = \sqrt{\sum_{i=1}^{n} |x_i - h_i|^2}$$

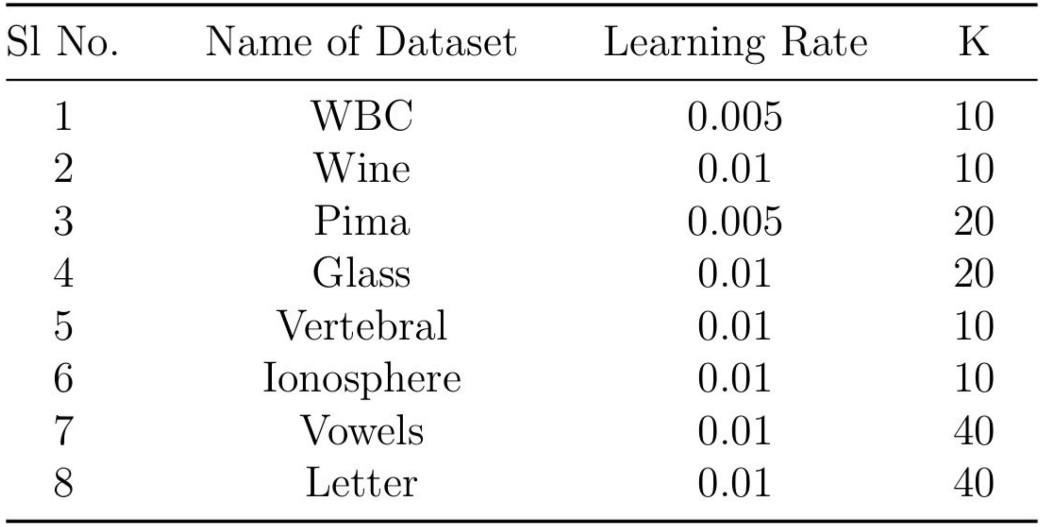
- The loss function essentially tries to minimize the Euclidean distance between the embeddings and the initial data points. Since the number of outliers is small, the model will learn a mapping function that maps the majority of the points (non-outliers) to an embedding that is close to the original data point, whereas the outliers will be mapped further away.

Methodology (Cont..)

Outlier Detection

- For a given set of points, first, the graph is constructed and sampled based on cosine similarity scores between the nodes.
- Then the proposed model is trained on the constructed graph using the unsupervised loss function. The model is then used to generate the embeddings for each of the nodes of the graph (data points).
- The Euclidean distance between the data points and the computed embeddings is calculated, and the **top n** data points with the highest euclidean distance with their embeddings are reported as the outliers.

Experimental Setup

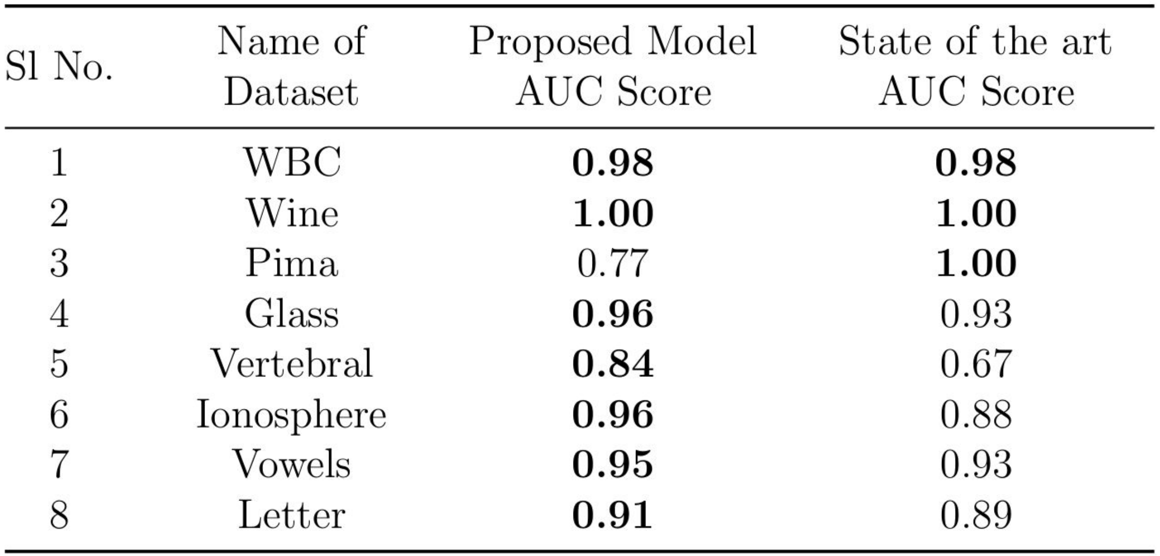


Sl No.	Name of Dataset	Learning Rate	K
1	WBC	0.005	10
2	Wine	0.01	10
3	Pima	0.005	20
4	Glass	0.01	20
5	Vertebral	0.01	10
6	Ionosphere	0.01	10
7	Vowels	0.01	40
8	Letter	0.01	40

Table 2: Hyperparameter Selection

- For each dataset, the hyper parameters, K and learning rate are varied, and the best-performing set of values are noted down.
- Table 2 shows the optimal hyper parameters selected for each dataset.

Results



Sl No.	Name of Dataset	Proposed Model AUC Score	State of the art AUC Score
1	WBC	**0.98**	**0.98**
2	Wine	**1.00**	**1.00**
3	Pima	0.77	**1.00**
4	Glass	**0.96**	0.93
5	Vertebral	**0.84**	0.67
6	Ionosphere	**0.96**	0.88
7	Vowels	**0.95**	0.93
8	Letter	**0.91**	0.89

Table 3: Performance on ODDS datasets

- The proposed model is trained and evaluated on each of the eight datasets selected from the ODDS collection.
- Table 3 shows the comparison of the obtained AUC score with the existing state-of-the-art techniques.
- The comparison reveals that the proposed system performs better on seven of the eight selected datasets.

Results (Cont..)

Sl No.	Name of Dataset	Proposed Model AUC Score ($K \neq 0$)	Proposed Model AUC Score ($K = 0$)
1	WBC	**0.98**	0.97
2	Wine	**1.00**	**1.00**
3	Pima	**0.77**	0.72
4	Glass	**0.96**	0.88
5	Vertebral	**0.84**	0.70
6	Ionosphere	**0.96**	0.95
7	Vowels	**0.95**	0.93
8	Letter	**0.91**	0.85

Table 4: Performance on ODDS datasets without graph structure

- Table 4 shows the comparison with the proposed approach when no neighbourhood information is used.
- This is achieved by setting K = 0. As evident from the results, removing neighbourhood information reduces the performance drastically, thereby emphasizing the importance of graph structure.

Novelty/Contributions

- Most of the existing works on outlier detection do not use graph structure at all, the ones that do use graph structure do not make use of graph neural networks. Our work leverages graph neural network by proposing a novel custom GraphSAGE based model architecture that utilises graph structure to detect outliers.
- Proposed model architecture also includes batch normalization which standardizes the data and increases the generalization capacity of the model and consequently makes outlier detection easier.
- Our proposed methodology outperforms the state of the art in seven of the eight datasets used for evaluation.

THANK YOU