Reimagining Influence Detection in Social Networks via Graph Neural Networks

Harshavardana Reddy Kolan[1] · Shikha Kumari[1] · Amir Hossein Jafari[1]

harshavardanareddy.kolan@gwu.edu · shikha.kumari@gwu.edu · ajafari@gwu.edu
Data Science, The George Washington University, 2121 I St NW, Washington, DC 20052, USA

Abstract

Influence detection within social networks is a critical challenge for recommendation systems and marketing strategies. Classical machine learning often depends on user activity and engagement metrics as input features, which are usually crafted manually. However, these methods do not capture intricate relationships that are essential in social networks. Graph Neural Networks (GNNs) provide an alternative approach to deal with this problem by learning directly from the structure of the social graph. They overcome the shortcomings of manual encodings through automating the learning of interaction patterns that are typically difficult to define algorithmically. In our study, we analyze both classical and GNN models for influence classification, highlighting the efficiency of the graph-based model for understanding user behavior and inter-network relationships. Our results show that while classical methods can yield reasonably satisfying outputs with careful feature engineering, GNNs offer a more robust solution for representing influence in complex social systems.

Key words: Graph Neural Networks, PyTorch Geometric

1. Introduction

Identifying influential users within online communities is necessary for recommendation, marketing, and information diffusion. Prior research has primarily relied on classical machine-learning models and graph-theoretic methods to rank or classify users based on their network position, activity patterns, or content features. These approaches include PageRank, k-shell decomposition, and even XGBoost, all used on Twitter, Facebook, and Instagram.

However, recent studies point out the gaps in these approaches. Abbasi and Fazl-Ersi (2017) emphasize the value of content, particularly visual features, over network data in platforms like Instagram. Cossu et al. (2015) show that real-world influence is better captured through language modeling than through follower counts or other network metrics. Ferdous and Anwar (2021) suggest a blend of content and structure with temporal patterns and argue for hybrid approaches, while Ishfaq et al. (2024) aim to discover topic-oriented influencers on question-answer forums like Stack Overflow using techniques from rule mining.

Motivated by these findings, this paper reanalyzes the task of classifying influential users on Stack Overflow. Traditional network-based measures may miss domain-specific authority rooted in technical contributions in this domain. Unlike prior work that relies on a classical approach, we explore whether Graph Neural Networks (GNNs) can improve performance by directly learning from the underlying graph structure and node-level features.

To validate our models, we also test their performance on a second dataset from the AskReddit dataset. Although the problem focuses on identifying user roles, active vs. influence, distinct from influencers, the dataset serves to measure the extent of generalizability of the GNN and the XGBoost classifying framework.

In this paper, we explore whether Graph Neural Networks (GNNs) can achieve better performance than classic approaches, such as XGBoost, in discovering influential or active users in various online communities. In line with this goal, our work also adds to the emerging research landscape that investigates deep learning approaches in social network analysis, specifically focusing on user interactions in question-answer-based websites.

2. Graph Neural Networks

2.1. Introduction to GNNs

Graph neural networks (GNNs) have emerged as a popular framework for modeling data with an inherent graph structure. Unlike traditional machine learning models, which rely on the independence and identical distribution of data points, GNNs consider relationships between the nodes, enabling information to propagate throughout connected entities.

Unlike conventional machine learning models, where tabular features are sufficient, GNNs leverage the data's topology by aggregating neighbors' information over multiple iterations.

In social networks, for instance, when it comes to predicting user behavior, GNNs can model the process of influence spread by aggregating the information of a user's immediate neighbors and making predictions based on all direct and indirect connections. GNNs can also benefit from collaborative filtering in recommendation systems by treating user-item interactions as a bipartite graph.

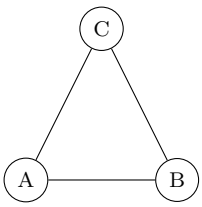
Different forms of GNNs, including Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), also implement individual strategies for aggregation and message-passing among nodes. Four great flavors of GNNs have been successfully applied to various domains, such as fraud detection in social network structure, molecular chemistry, and natural language.

2.2. Types of Graphs with Figurative Examples

Here is a list of the most common graphs used in graph-based learning, each of which has been illustrated.

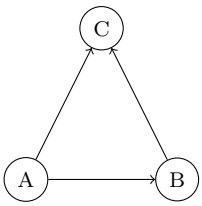
Undirected Graph

Undirected graphs are edges that are not directional. The links (connections) between nodes are bidirectional.



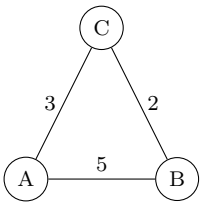
Directed Graph

A directed graph is a graph in which the edges have a direction. It shows the connection from node to node.



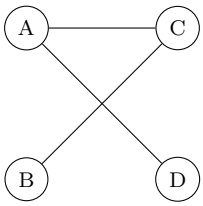
Weighted Graph

In a weighted graph, weights (or values) are assigned to edges, showing how strong the relationships among nodes are.



Bipartite Graph

In a bipartite graph, nodes can be divided into two disjoint sets, i.e., there are two groups of nodes and no edges between them.



2.3. Graph Types Based on Properties Relevant to GNNs

In addition to these fundamental properties, a graph's structural and semantic properties positively influence the way Graph Neural Networks process and learn from it. Familiarity with these types is critical for choosing suitable GNN models and structuring message-passing processes.

- **Homogeneous Graphs:** There is only one type of node and one type of edge. These graphs are less complex and are usually found in regular GCNs, such as Twitter's user-user interaction network.
- **Heterogeneous Graphs:** Have various functions based on nodes and/or edges. They need hetero-GNNs (e.g., R-GCN) to maintain the semantics of multiple relationships. For example, Stack Overflow with users, questions, and answers is a separate node type and a distinct relationship these nodes share as edges.
- **Directed Graphs:** Edges indicate asymmetric relationships (e.g., who answered whose question). Many GNNs handle edges as bidirectional edges by adding reverse edges so that message passing can happen in both directions.
- **Undirected Graphs:** Edges are undirected (or symmetric), meaning no direction is associated with the link, which simplifies message passing since information flows evenly among connected nodes.

2.4. Heterogeneous Graph Representation for Stack Overflow

The GNN model used in our study is based on a heterogeneous graph representation, which captures the intricate relationships between different entities in the Stack Overflow platform. This heterogeneous graph consists of multiple node types—users, questions, and answers—along with directed edges representing interactions such as "asks," "answers,", "has" and "accepted answers." Three primary node types:

- **Users:** Represents contributors who ask and answer questions.
- **Questions:** Represents queries posted by users.
- **Answers:** Represents responses to questions.

Edges in the graph represent interactions between these entities:

- `Asks` (user → question)
- `Answers` (user → answer)
- `Has` (question → answer)
- `Accepted Answer` (question → answer)
- `Reverse` edges to capture bidirectional influence.
- `Self-loop` (user → user) to enhance message passing in GNN models.

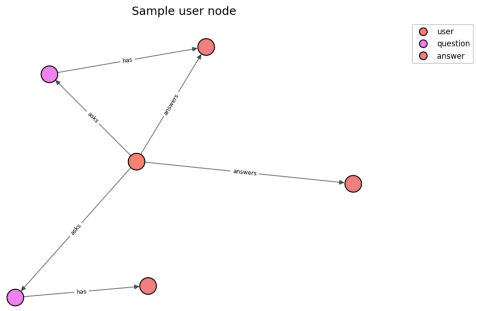


Fig. 1: Ego-graph of the heterogeneous Stack Overflow graph (users, questions, answers, and reverse/self-loop edges).

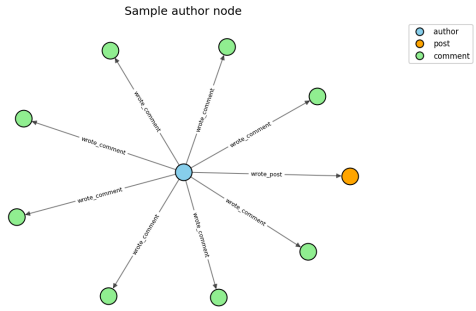


Fig. 2: Ego-graph of the heterogeneous AskReddit graph (authors, posts, comments, and reverse/self-loop edges).

To enhance message passing, we incorporate reverse edges, allowing a bidirectional flow of information between users and content and self-loops on user nodes to reinforce their intrinsic properties. Additionally, we utilize a two-layer GraphSAGE model with a mean-based aggregator, leveraging neighborhood information to enhance feature propagation. The GNN model uses a NeighborLoader, which efficiently samples neighbors for training.

By structuring the Stack Overflow interactions as a heterogeneous graph, our GNN can leverage both direct and indirect user contributions, more effectively capturing influence propagation and contextual relationships than classical models.

2.5. Heterogeneous Graph Representation for AskReddit

To evaluate the generalizability of our models, we also experimented on a second dataset sourced from AskReddit, a subreddit where users initiate discussions through posts and others respond through comments. Similar to Stack Overflow, we represent this dataset as a heterogeneous graph, modeling the interactions among users, posts, and comments. Three primary node types:

- **Author:** Represents individuals who create posts or comments.
- **Posts:** Represents the initial questions or threads started by users.
- **Comments:** Represents replies made to posts.

Edges in the graph capture the interactions between these entities:

- `Wrote post` (author → post): The user creates a post.
- `Wrote comment` (author → comment): The user writes a comment.
- `Has` (post → comment): A post contains a comment.
- `Reverse` edges to capture bidirectional influence.
- `Self_Loop` (author → author): Self-loop on author nodes to preserve node-specific features.

Unlike Stack Overflow, the AskReddit platform does not contain an "accepted answer" signal. Therefore, no edge type is designated to capture authoritative comments. Nevertheless, the resulting heterogeneous graph effectively captures the structural and contextual relationships between users.

We also apply the same GraphSAGE-based heterogeneous GNN model to this graph structure, utilizing neighbor sampling and reverse edges to enable efficient and expressive learning over Reddit-style user interactions.

2.6. Message Passing in Graph Neural Networks

Message passing is a key concept in Graph Neural Networks (GNNs). It enables nodes to communicate with their neighbors to refresh their embeddings. The new embedding of each node is a function of its features, the features of its neighbors, and, optionally, the edge features between nodes. Message passing works by information aggregation from neighboring nodes and updating node embeddings through learnable functions until the nodes can absorb information of a radius n neighbors.

The message-passing framework is beneficial for node classification, link prediction, and graph classification tasks.

2.6.1. Formula for Message Passing

The message-passing process in GNNs is generally expressed as:

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)} \oplus \sum_{j \in N(i)} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

Where:

- $\mathbf{x}_i^{(k)} \in \mathbb{R}^F$: Embedding of node i at layer k.
- $N(i)$: Neighbors of node i.
- $\mathbf{e}_{j,i} \in \mathbb{R}^D$: Optional edge features from node j to node i.
- $\oplus$: Permutation-invariant aggregation function (sum, mean, or max).
- $\phi^{(k)}$: Learnable function, such as a Multi-Layer Perceptron (MLP).
- $\gamma^{(k)}$: Learnable update function, often a non-linear transformation.

3. Data

3.1. Dataset Description

This study uses two real-world datasets—one from Stack Overflow and the other from Reddit—to evaluate the performance of user influence and activity classification models.

Stack Overflow Dataset: Extracted using the Stack Exchange API, this dataset includes user, question, and answer data. It contains 81,801 user records, 118,700 questions, and 29,600 answers. Based on a computed influence score, the top 10% of users are labeled as *influential*. Using the following formula, we calculate the influence score:

$$\text{Influence Score} = \text{Reputation} + 3 \times \text{Gold Badges} + 2 \times \text{Silver Badges} + \text{Bronze Badges} \quad (1)$$

Reddit Dataset: We use the "A Month of AskReddit" dataset from Kaggle, which comprises 369,232 posts and 1,048,576 comments. Authors in the top 10% based on total post and comment counts were labeled as *active*.

Both datasets are inherently imbalanced; this reflects the natural skew in online platforms where a minority of users contribute disproportionately. This imbalance is intrinsic to the problem and makes the datasets suitable for benchmarking classification techniques.

3.2. Data Preprocessing

This section outlines the preprocessing strategies adopted for both GNN and XGBoost models. Each dataset required tailored transformations to suit the structural and tabular nature of the respective learning methods.

Stack Overflow for GNN Relevant fields were extracted from the users, questions, and answers datasets, and columns directly used to compute the target label (e.g., reputation and badge counts) were removed. Each user's influence score was calculated, and the top 10% were labeled as influential. The cleaned data was stored in structured CSV files for use in GNN-based models.

Stack Overflow for XGBoost Data from users, questions, and answers were merged into a unified dataset. Features were aggregated per user, including the number of questions asked, average question/answer scores, and the total number of accepted answers. Missing values—common among users with limited activity—were filled with zeros. The resulting dataset was saved for training with XGBoost.

AskReddit for GNN and Xgboost Preparation: Raw post and comment data were cleaned by removing irrelevant columns and handling missing entries. A user activity score was computed by summing the number of posts and comments per user. The top 10% most active users were labeled accordingly. To reduce computational overhead, we selected approximately the first 80K rows from both the posts and comments datasets for graph construction, while for xgboost model first 75K rows were considered. This graph was then used to train the model, where an equal number of nodes from each class were sampled during training. The details of this stratified sampling approach are discussed later in the paper.

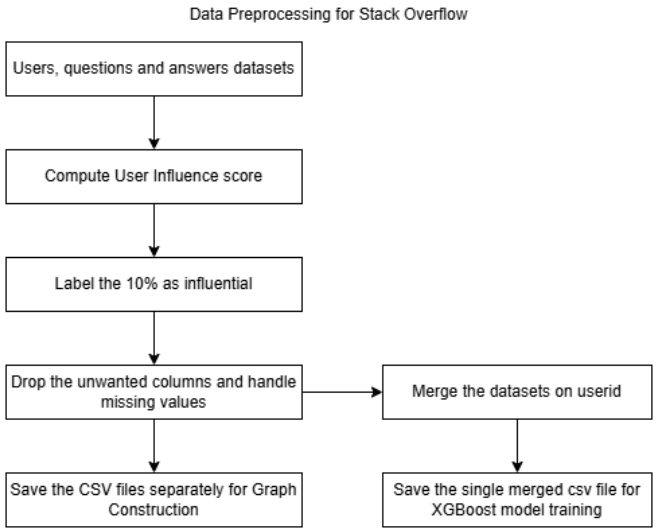


Fig. 3: Data Preprocessing Pipeline for Stack Overflow

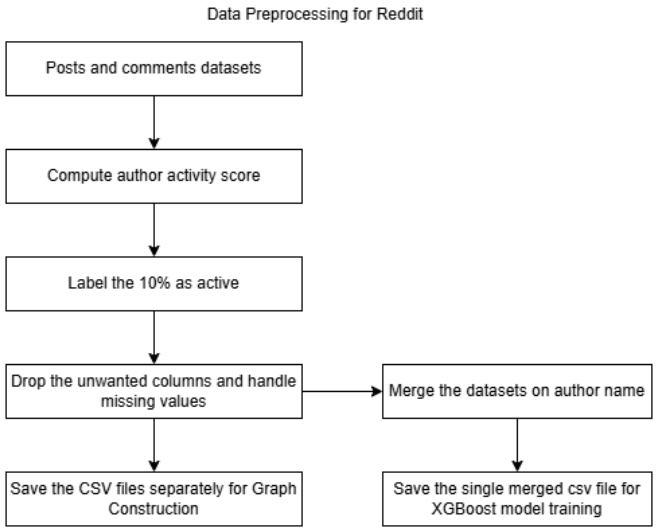


Fig. 4: Data Preprocessing Pipeline for AskReddit

4. Modeling

We examined two different approaches for identifying influential users on online social networks: a classical machine learning model (XGBoost) and a graph-based deep learning model (graph neural network utilizing GraphSAGE). These approaches were carefully tailored to the heterogeneous nature of the Stack Overflow and AskReddit datasets, which feature various node types and directed edges.

4.1. Classical Machine Learning: XGBoost

For our benchmark model, we chose the XGBoost classifier, a popular tree-based gradient boosting algorithm effective for

structured tabular data. The input features were carefully crafted by aggregating users, questions and answers dataset on user id, such as total_questions, avg_question_score, avg_answer score and accepted_answers.

To optimize the performance, we conducted a grid search across various hyperparameter sets as given in the `config.yaml` file. Parameters such as `n_estimators`, `max_depth`, `learning_rate`, `subsample`, `colsample_bytree`, `gamma`, and `scale_pos_weight` were systematically tuned. XGBoost served as a strong benchmark for evaluating the maximum performance of models that rely solely on aggregated and manually crafted features, without leveraging graph structure.

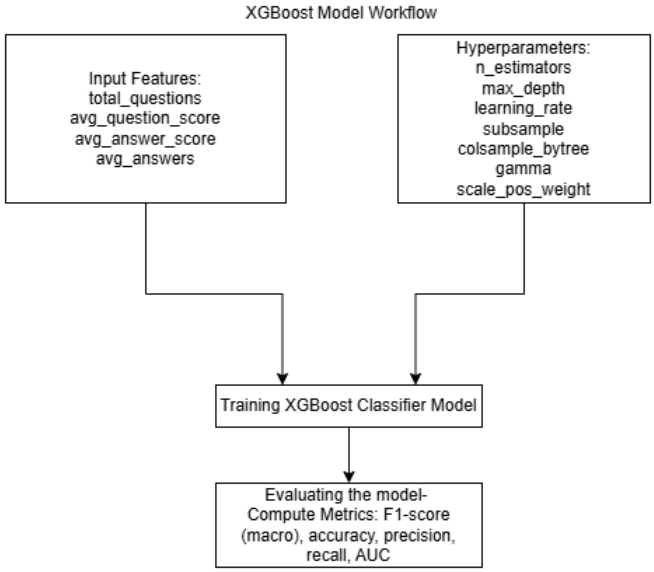


Fig. 5: XGBoost Model Workflow

4.2. Graph-Based Deep Learning: GraphSAGE

To capture both relational and topological dependencies present in our dataset, we developed a Heterogeneous Graph Neural Network utilizing PyTorch Geometric. Our main model is built on the GraphSAGE architecture, a type of GCN, which facilitates inductive learning of node representations through the aggregation of neighbor information. The foundational structure consists of two SAGEConv layers, followed by a fully connected layer for the final classification process.

Considering the diverse nature of our graph comprising elements such as users, questions, answers (in Stack Overflow) as well as authors, posts, comments (in AskReddit), we used `to_hetero()` to make the GraphSAGE model for accommodating various types of nodes and edge relationships.

4.2.1. Graph Convolutional Network (GCN)

Graph Convolutional Networks (GCNs) are a class of neural networks that are specifically designed for graph-structured data. Unlike traditional convolutional neural networks that handle grid-like data, such as images, GCNs extend the concept of convolution to accommodate graphs, allowing for efficient learning from non-Euclidean domains. First presented by Kipf and Welling in 2017, GCNs function by consolidating feature information from the local neighborhood of each node, enabling the model to capture the relational structures inherent in the graph's topology.

The core part of GCNs lies in the message passing mechanism, where each node updates its representation by aggregating features from its immediate neighbors, followed by a learnable transformation and nonlinearity. This iterative process occurs across multiple layers to extract information from neighborhoods that span multiple hops within the graph.

Why We Used GCN

In the context of predicting user influence on social network platforms like Stack Overflow and AskReddit, it is essential to utilize the comprehensive structure details within the graph when modeling user relationships, question-answer interactions, and content hierarchy. GCNs are particularly well-suited for such tasks due to their ability to explore graph connectivity to learn node embeddings that encapsulate both feature and structural contexts.

In contrast to traditional models such as XGBoost, which rely on manually crafted features, GCNs can learn end-to-end representations directly from graph structure and node features, which is crucial for tasks involving multi-relational and multi-type data.

How We Used GCN

In our project, we developed a heterogeneous graph consisting of distinct node types, including users, questions, and answers for Stack Overflow, as well as authors, posts, and comments for AskReddit. The graph also included different edge types such as `asks`, `answers`, `has`, and `accepted_answer`. We used SAGEConv layers from PyTorch Geometric as the base model and implemented the GCN using PyG's `to_hetero()` wrapper to support node and edge-type transformations. During preprocessing, we explicitly added reverse edges and self-loop edges to enable bidirectional message passing. Each node type's input features were processed through two stacked GCN layers followed by a fully connected classification layer. The

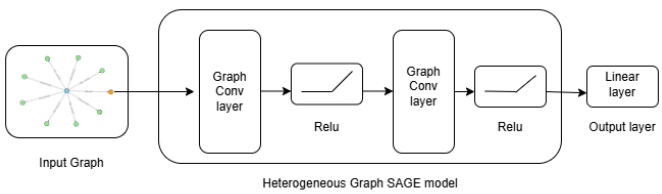


Fig. 6: GCN Model Architecture

variations of each model were determined by parameters such as `hidden_channels`, `aggregation` (sum or mean), `learning rate`, `batch_size`, and `num_neighbors` (number of neighbor hops per layer). These hyperparameters were dynamically retrieved from a `config.yaml` configuration file to ensure reproducibility and facilitate experimentation with different configurations.

We adopted a mini-batch training approach using NeighborLoader, which is addressed in the Experiments and Results section.

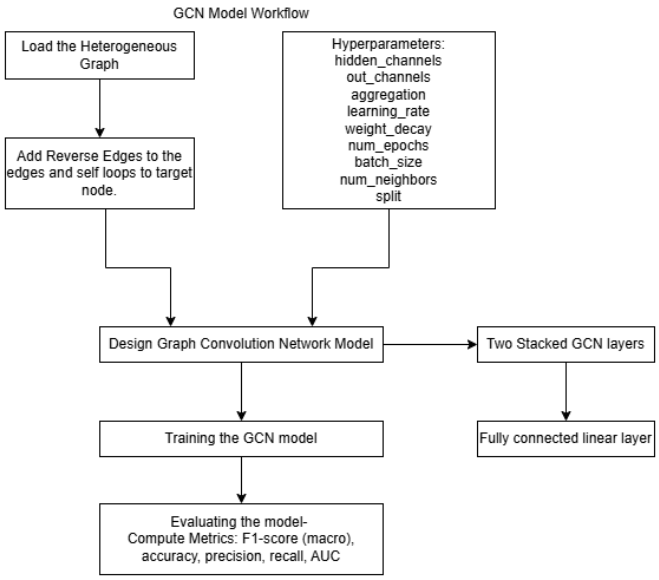


Fig. 7: GCN Model Workflow

Algorithm 1 Stratified Mini-Batching

Input: Target Variable y, train mask, batch size B
Output: Initialize a list of class-balanced mini-batches
$node\_indices \leftarrow$ Extract indices where train mask is true
Initialize $class\_indices \leftarrow$ empty dictionary
for each class c in unique($y[node\_indices]$) **do**
    $idx \leftarrow$ indices in $node\_indices$ where label $= c$
    Shuffle idx
    $class\_indices[c] \leftarrow idx$
end for
$num\_classes \leftarrow$ number of unique classes
$samples\_per\_class \leftarrow B \div num\_classes$
$min\_batches \leftarrow$ minimum complete batches possible across classes
Initialize $batches \leftarrow$ empty list
for $i = 0$ to $min\_batches - 1$ **do**
    $batch \leftarrow$ concatenate: $class\_indices[c][i \cdot samples\_per\_class : (i+1) \cdot samples\_per\_class]$ for all c
    Append $batch$ to $batches$
end for
return $batches$

5. Experiments and Results

5.1. Stratified Mini-Batching

Influential user classification generally suffers from substantial class imbalance; here, only a small proportion of nodes represent the minority (influential) class. To address this issue, we designed a stratified mini-batch strategy tailored for heterogeneous graph data. Instead of applying techniques like random sampling or down-sampling on the datasets, we constructed mini-batches that contain an equal number of nodes from each class, ensuring that each batch reflects a balanced class distribution. This approach prevents class bias during training and enables each gradient update to be computed from a class-balanced sample of the data.

Our methodology first shuffles node indices according to their respective classes, then dynamically computes the maximum number of fully balanced batches that can be formed without exhausting the minority class. These batches are then passed to PyG's NeighborLoader for layer-wise sampling, which preserves neighborhood structure and local connectivity. This method yields two key advantages:

- **Training Stability and Fairness** – Each class contributes equally during training, reducing bias and ensuring fair representation.
- **Improved Generalization** – Particularly for the minority class, we observed significant improvement in recall, precision, and F1-score, as illustrated in `table 1`.

This stratified sampling approach results in establishing a clear class distribution before feeding the model. Importantly, it achieves this while maintaining the diversity of data by keeping the complete training set intact. Due to its efficiency, this approach can be readily extended to other graph-based learning tasks that involve imbalanced node classification, such as fraud detection, rare event modeling, or anomaly detection within social networks.

5.2. Performance Metrics

To evaluate the performance of the proposed models for predicting influential users on both the Stack Overflow and AskReddit datasets, we used several standard metrics. The `table 1` below presents the performance metrics for the XGBoost classifier and GCN model, allowing a comparative analysis between the two approaches. We have used set 1, which is identified as the second set within the config.yaml file.

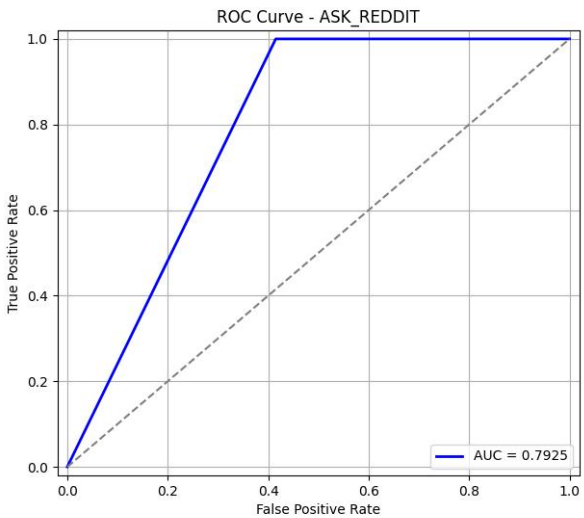


Fig. 8: ROC-AUC curve for Ask Reddit dataset

Table 1. Performance Comparision

Models	Stack Overflow					AskReddit				
	Accuracy	Precision	Recall	F1-score	AUC	Accuracy	Precision	Recall	F1-score	AUC
Xgboost	0.813	0.589	0.646	0.603	0.655	0.701	0.657	0.659	0.658	0.706
GCN	0.813	0.596	0.661	0.612	0.672	0.655	0.664	0.792	0.616	0.792

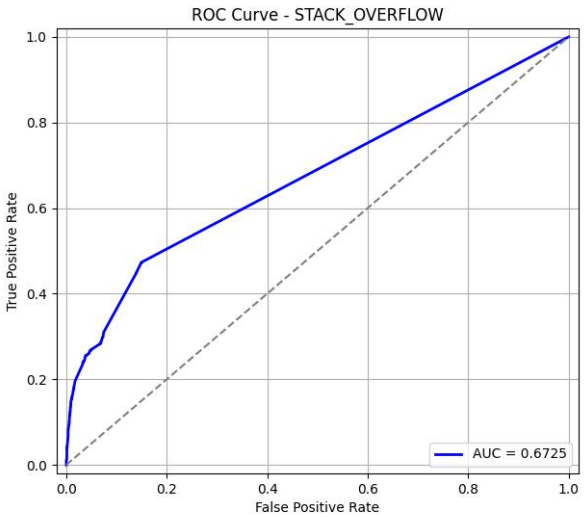


Fig. 9: ROC-AUC curve for Stack Overflow dataset

6. Discussion

Our study of XGBoost and GCN for influential user prediction provides valuable observations regarding their performance across various scenarios within heterogeneous social network datasets, StackOverflow, and AskReddit.

XGBoost performed well on both the datsets Stack Overflow and AskReddit, particularly due to its extensive feature engineering.

On the other hand, GCN model demonstrated an inherent ability to model relational patterns directly from the graph structure. It's performance remained either consistent or superior across both datasets, with noticeable improvements in AUC. This suggests that GNN-based models are better suited for influence modeling in settings where user interactions are more implicit, temporally scattered, or structurally complex.

The advantage of stratified mini-batching also became apparent, especially in addressing class imbalance, where only a minority of users display influential behavior. This helped maintain balanced gradient updates and led to significant improvements in the GCN model learning dynamics as can be seen from the table.

6.1. Limitations and Future Scope

Limitations

A major limitation of current research is the focus on static network properties. In reality, user influence often tends to change over time, and our current static graph-based models may overlook such temporal patterns. Additionally, while GCN model showed significant improvement, they can be heavy on computation cost, particularly for large graphs. This restricts their usage in real-time systems unless efficient algorithms for graph sampling or a scalable architecture are developed.

Another constraint is the lack of content-level features (e.g., post text, tags, comment semantics), which could have further improved the representation learning process for both XGBoost and GNN models.

Future Scope

Future studies could examine the application of attention-based Graph Neural Networks (GNNs), such as Graph Attention Networks (GAT), to enhance both the interpretability and accuracy of the models. Additionally, integrating text embeddings obtained from user posts and comments into the node feature set may increase the model's context-awareness. Moreover, testing over more datasets that have different graph topologies (e.g., Twitter, GitHub) could widen the generalization of the results beyond just the two platforms currently utilized.

Finally, investigating multi-task learning approaches where the model jointly predicts influence, activity level, and community participation may lead to a more deeper insights into use behavior.

7. Conclusion

This study presents a comparative investigation into user influence modeling using traditional and graph-based learning techniques. Our experiments on the Stack Overflow and AskReddit datasets demonstrate that while XGBoost performs well with manual feature engineering, GCNs offer a competitive alternative by directly leveraging the graph structure.

The use of stratified mini-batching was essential to mitigate issues associated with class imbalance and to get better performance of GCN, especially on sparse data. These results underscore the capacity of graph-based models to learn intricate relationship dynamics that are frequently overlooked by classical approaches. Overall, GNNs emerge as a viable medium for influence modeling in social networks, with potential for further gains when extended to temporal, attention-based, or content-aware architectures.

A. Appendix

A.1. Summary of Hyperparameters

A snippet from the `config.yaml` file used for defining hyperparameter sets for both GNN and XGBoost models.

```
gnn:
  sets:
    - hidden_channels: 64
```

```
      out_channels: 2
      aggregation: "mean"
      learning_rate: 0.01
      weight_decay: 1e-4
      num_epochs: 50
      batch_size: 64
      num_neighbors: [10, 10]
      split: "train_rest"
xgboost:
  sets:
    - n_estimators: [100, 200, 300]
      max_depth: [5, 10, 15]
      learning_rate: [0.005, 0.02, 0.1]
      subsample: [0.6, 0.8, 1.0]
      colsample_bytree: [0.6, 0.8, 1.0]
      gamma: [0, 0.05, 0.15]
      scale_pos_weight: "balanced"
```

8. Declarations

Conflict of interest The authors affirm that there are no conflict of interest that could influence the objectivity or integrity of the research reported in this paper. We declare that we have no financial or personal relationships with individuals or organizations that could inappropriately influence our work. Neither have we received any financial support or compensation related to the subject matter of this paper. Additionally, we do not hold any patents or patent applications relevant to the content of the paper nor have we received funding from organizations that may have a vested interest in the outcome of this research. Furthermore, we have no professional or personal affiliations that may be perceived as a conflict of interest in connection with the work presented in this paper. By declaring no conflict of interest, we uphold the principles of integrity and transparency in scientific research, ensuring the credibility and ethical conduct of this study.

References

1. Kaggle Datasets, "A Month of AskReddit," *arXiv preprint arXiv:1706.02216*, 2017. https://www.kaggle.com/datasets/thunderz/a-month-of-askreddit
2. A. Vaswani *et al.*, "Attention is All You Need," *arXiv preprint arXiv:1706.02216*, 2017. https://arxiv.org/pdf/1706.02216
3. T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv preprint arXiv:1403.6652*, 2014. https://arxiv.org/pdf/1403.6652
4. H. Chen and D. Kim, "GNN-Based Influence Detection in Q&A Communities," *IEEE Access*, 2024. https://ieeexplore.ieee.org/document/10802916
5. Y. Li *et al.*, "A Survey on Graph Neural Networks for Link Prediction," *arXiv preprint arXiv:2412.12416*, 2024. https://arxiv.org/pdf/2412.12416
6. W. Zhang and H. Liu, "Comparing Graph Neural Networks and Traditional Models for Social Network Influence," *Journal of Computer and Communications*, vol. 11, no. 7, pp. 1–15, 2023. https://www.scirp.org/pdf/jcc_2023072714441986.pdf
7. M. R. Haque *et al.*, "Influencer Detection in Online Social Networks Using Machine Learning Algorithms: A Survey," *Journal of Intelligent Systems*, 2021. https://www.tandfonline.com/doi/full/10.1080/08839514.2021.2010886
8. A. Jindal, M. Singh, and M. Sharma, "Ranking Influential Nodes in Complex Networks Using Local Structure Information," *2015 Int. Conf. on Computational Intelligence and Communication Networks (CICN)*, pp. 553–558, IEEE, 2015. https://ieeexplore.ieee.org/document/7321240
9. PyG Library. Available at: https://pytorch-geometric.readthedocs.io
10. Torch Library. Available at: https://pytorch.org/docs/stable/index.html
11. Tyler Wallett and Amir Jafari, "A benchmark for graph-based dynamic recommendation systems"https://link.springer.com/article/10.1007/s00521-024-10425-6
12. Sunisha Harish, Chirag Lakhanpal and Amir Hossein Jafari , "Leveraging graph-based learning for credit card fraud detection: a comparative study of classical, deep learning and graph-based approaches"https://link.springer.com/article/10.1007/s00521-024-10397-7
13. Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande and Jure Leskovec , "Strategies for Pre-training Graph Neural Networks"https://arxiv.org/abs/1905.12265
14. Shaked Brody, Uri Alon and Eran Yahav , "How Attentive are Graph Attention Networks?"https://arxiv.org/abs/2105.14491
15. Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang and Yu Sun , "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification"https://arxiv.org/abs/2009.03509
16. Guohao Li, Chenxin Xiong, Ali Thabet and Bernard Ghanem , "DeeperGCN: All You Need to Train Deeper GCNs"https://arxiv.org/abs/2006.07739