Final Report On Reimagining Influence Detection in Social Networks via Graph Neural Networks

Harshavardana Reddy Kolan Shikha Kumari

February 2025

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

1 Introduction

Identifying influential users in online communities is critical for applications such as information diffusion, content moderation, and targeted engagement. Prior research in this area 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—ranging from PageRank and k-shell decomposition to models like XGBoost—have been widely adopted in platforms such as Twitter, Facebook, and Instagram.

However, recent studies highlight key limitations in existing methods. 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) argue for hybrid methods that integrate content, structure, and temporal patterns, while Ishfaq et al. (2024) propose rule-mining techniques to identify topic-specific influencers in QA forums like Stack Overflow.

Motivated by these findings, this paper revisits the problem of classifying influential users, specifically within the context of Stack Overflow—a platform where traditional network-based measures may miss domain-specific authority rooted in technical contributions. Unlike prior work that leans on classical models, we investigate whether Graph Neural Networks (GNNs) can offer improved performance by directly learning from the underlying graph structure and node-level features.

To validate our models, we also evaluate their performance on a second dataset from AskReddit. While the problem setting differs—focusing on classifying highly active users rather than influencers—the dataset allows us to test the generalizability of both the GNN and XGBoost models.

Our primary objective is to assess whether GNNs outperform classical models like XGBoost in identifying influential or active users across different community platforms. This comparative study aims to contribute to the growing conversation around leveraging deep learning methods for social network analysis, particularly in user-centric applications within QA and discussion forums.

2 Data

2.1 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, 118,701 questions, and 29,601 answers. Based on a computed influence score, the top 10% of users are labeled as *influential*. The influence score is calculated using the following formula:

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

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 are labeled as *active*.

Both datasets are inherently imbalanced, with only 10% of users marked as influential or active. 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.

2.2 Data Preprocessing

Stack Overflow – GNN Preparation: 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 – XGBoost Preparation: 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.

Reddit – GNN 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, and the updated data was saved for GNN modeling.

Reddit – XGBoost Preparation: Cleaned posts and comments were processed to compute per-author features such as average post scores, total comment counts, and posting frequency. These features were grouped by author and merged into a consolidated dataset. Missing values were handled, and the final dataset was saved for use with XGBoost.

2.3 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.

2.4 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.

3 Models Used

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

3.1 XgBoost

3.1.1 Objective:

Used XGBoost to classify users as influential (Stack Overflow) or active (AskReddit).

3.1.2 Data Preparation:

Preprocessing: Removed irrelevant columns `(author, user_id, display_name)` to reduce noise. Splitting: Data split into train and test sets using `train_test_split` with stratification to maintain class balance.

3.1.3 Handling Class Imbalance

Computed `scale_pos_weight` dynamically based on the class distribution in the training set. Helped XGBoost focus more on minority classes during training.

3.1.4 Hyperparameter Tuning

Used RandomizedSearchCV with: - `3fold Cross Validation`
- `f1score` and `AUC` as the evaluation metric (important for imbalanced classification).
- Parameters tuned included `learning_rate, max_depth, n_estimators`, and others.

3.2 GraphSAGE

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. 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.

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.

3.2.1 Objective

Build a Heterogeneous Graph Neural Network (GNN) to classify users: Influential users on Stack Overflow. Active users on AskReddit.

3.2.2 Graph Construction

Created a heterogeneous graph with multiple node and edge types: Stack Overflow nodes: user, question, answer and for AskReddit nodes: author, post, comment

Added reverse edges `(e.g., rev_asks, rev_answers)` and `selfloops` for better message passing.

Ensured all `edge_index` tensors were contiguous (important for `pyglib` and NeighborSampler).

3.2.3 Data Preparation

- Applied RandomNodeSplit: `80% train, 10% validation, 10% test.`

- Maintained class distribution across splits.

- After splitting, saved the preprocessed dataset to `model_artifacts` folder `(dataset_split_dataset_name.pt)` to avoid re-splitting during testing.

3.2.4 GNN Model Architecture

Implemented a GraphSAGE-based model:

- 2 convolution layers (SAGEConv) with ReLU activations.
- Final linear layer for classification.
- Converted the model to heterogeneous using `to_hetero()` from PyG.

3.2.5 Training Strategy

Used stratified mini-batches:

- Ensured equal representation of each class in every batch.
- Trained using NeighborLoader sampling (to scale to large graphs).

Parameters:

- Optimizer: Adam with configurable learning rate and weight decay.
- Loss function: Cross Entropy Loss.

3.3 Stratified Mini-Batching

Influential user classification often suffers from severe class imbalance, with only a small fraction representing the influential class. To address this, we designed a stratified mini-batch strategy that ensures each mini-batch contains an equal number of nodes from each class. Instead of random sampling or downsampling, we shuffled nodes by class and created balanced batches, preserving neighborhood structure through PyG's NeighborLoader. Refer `Algorithm 1`

3.4 Snippets (config.yaml)

Our models were trained on the below configurations:

```
gnn:
  sets:
    - hidden_channels: 64
      out_channels: 2
      aggregation: "mean"
      learning_rate: 0.01
      weight_decay: 1e-4
```

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 $batches$

```
      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"
```

4 Performance Metrics

Stack Overflow Dataset

To evaluate the performance of the proposed XGBoost-based classification model for predicting influential users on Stack Overflow, we used several standard metrics. (Table 1) presents the performance metrics for the XGBoost classifier, while (Table 2) reports the evaluation metrics for the GNN-based model, allowing a comparative analysis between the two approaches.

Table 1: Performance Metrics of the XGBoost Classifier

Metric	**Value**
Accuracy	0.813
Precision	0.589
Recall	0.646
F1-score	0.603
AUC	0.655

The performance of our GNN-based model using stratified is also evaluated using the same metrics.

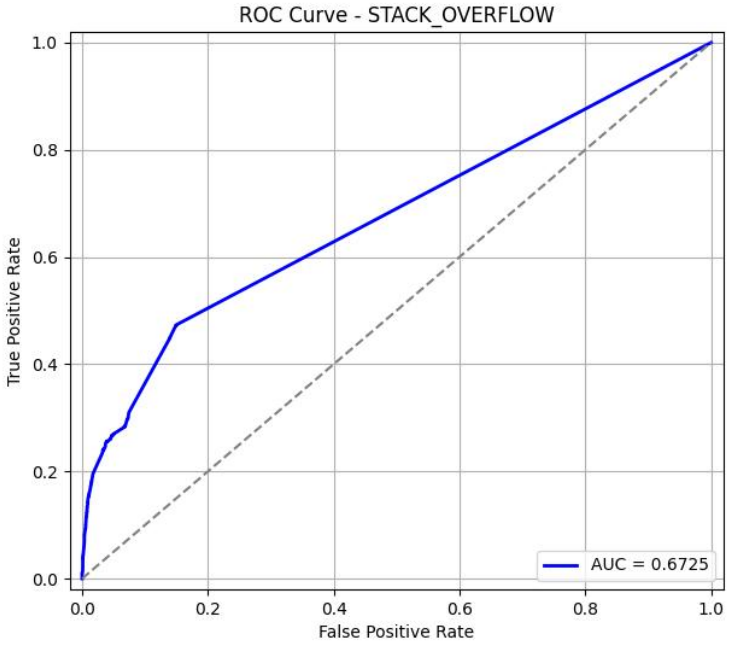

Figure 1: ROC-AUC curve for Stack Overflow dataset

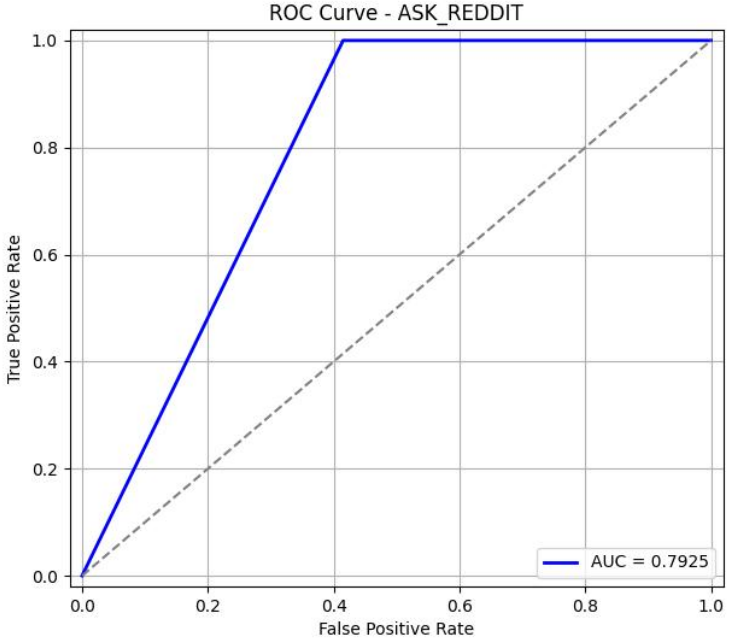

Figure 2: ROC-AUC curve for Ask Reddit dataset

Table 2: Performance Metrics of the GNN Classifier

Metric	Value
Accuracy	0.813
Precision	0.596
Recall	0.661
F1-score	0.612
AUC	0.672

AskReddit Dataset

To evaluate the performance of the proposed XGBoost-based classification model for identifying active users on AskReddit dataset, we used several metrics. Table (3) presents the performance metrics for the XGBoost classifier, while Table (4) reports the evaluation metrics for the GNN-based model, allowing a comparative analysis between the two approaches.

Table 3: Performance Metrics of the XGBoost Classifier

Metric	Value
Accuracy	0.701
Precision	0.657
Recall	0.659
F1-score	0.658
AUC	0.706

The performance of our GNN-based model using stratified sampling is also evaluated using the same metrics. From Figure 2, we can infer that although the AUC is higher for AskReddit, the shape of the ROC curve implies underfitting. The reason for this behaviour might be label imbalance

Table 4: Performance Metrics of the GNN Classifier

Metric	Value
Accuracy	0.655
Precision	0.664
Recall	0.792
F1-score	0.616
AUC	0.792

5 Conclusion

5.1 Observations

In this study, we explored the task of identifying influential or active users across two distinct online platforms—Stack Overflow and AskReddit—using both classical machine learning and graph-based learning approaches. We proposed a comparative evaluation between XGBoost, a high-performing classical model, and Graph Neural Networks (GNNs), which are designed to leverage relational structure inherent in graph data.

Our results demonstrate that while XGBoost can achieve strong performance when feature engineering is thorough, GNNs offer a compelling alternative by learning directly from heterogeneous graph structures.

Furthermore, we addressed the issue of class imbalance through stratified mini-batching, which significantly improved the performance of our GNN model, particularly in recall and F1-score. This enhancement strategy proved especially useful in learning from sparse and skewed datasets where only a small fraction of users exhibit influential behavior.

5.2 Future Work

Future work can explore advanced GNN variants such as temporal GNNs for evolving interactions or attention-based models for capturing importance-weighted message passing. Integrating content features (e.g., text embeddings) with structural information could further enhance performance in real-world applications.

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