Identifying Influential Users using Classical Models and Graph Neural Networks

Harshavardana Reddy Kolan Shikha Kumari Amir Hossein Jafari

February 2025

Abstract

Identifying influential users in social networks is a fundamental problem with applications in recommendation systems, content moderation, marketing, and information diffusion. While traditional machine learning models rely on hand-crafted features derived from user activity and engagement metrics, these approaches often overlook the relational dynamics that define influence in a networked environment. Graph Neural Networks (GNNs) provide a more powerful alternative by learning directly from the structure of the social graph, capturing complex interaction patterns that are difficult to encode manually. This work presents a comparative study of classical models and GNNs for influence classification, highlighting the advantages of graph-based learning in understanding user roles and network behavior. Our findings show that while traditional methods can be effective with careful feature engineering, GNNs offer a more robust and scalable solution for modeling influence in complex social systems.

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.

Table 1: Comparison of GNN, Classical Models, and Label Propagation

Aspect	**Graph Neural Networks (GNNs)**	**Classical Machine Learning Models**	**Label Propagation**
Data Dependency	Designed for graph-structured data with node/edge features	Operates on tabular/vectorized data with independent samples	Requires graph structure, uses connectivity but not node features
Feature Utilization	Learns representations by aggregating neighbor features	Uses hand-crafted features or engineered features	Does not use features; only relies on labels and graph structure
Learning Mechanism	End-to-end supervised or semi-supervised learning via message passing	Supervised learning using algorithms like SVM, Logistic Regression, Random Forest	Unsupervised/semi-supervised; propagates labels through edges
Scalability	Scalable with mini-batching and sampling techniques (e.g., GraphSAGE)	Easily scalable for independent data, but not optimized for graphs	Less scalable for large graphs due to iterative propagation
Use Case Scenarios	Node classification, link prediction, graph classification	Classification, regression, clustering on structured datasets	Community detection, semi-supervised node labeling

2 Preliminaries

3 Graph Neural Networks

3.1 Introduction to GNNs

Graph Neural Networks (GNNs) have gained popularity for modeling data with an inherent graph structure. Unlike traditional machine learning models that assume independent and identically distributed data points, GNNs consider relationships between nodes, allowing information to propagate through connected entities.

Unlike conventional machine learning models that rely solely on tabular features, GNNs exploit the topology of the data by iteratively aggregating information from neighboring nodes. This iterative message-passing mechanism enables GNNs to learn meaningful representations that capture both local and global graph structures.

For example, in social networks, GNNs can model influence propagation by aggregating information from a user's immediate neighbors, enabling predictions based on both direct and indirect connections. Similarly, in recommendation systems, GNNs can leverage collaborative filtering by analyzing user-item interactions as a bipartite graph.

Several key variants of GNNs exist, including Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and GraphSAGE, each employing different strategies for node aggregation and message passing. These models have been successfully applied to a wide range of domains, such as fraud detection, molecular chemistry, and natural language processing, showcasing their versatility and effectiveness in structured data environments.

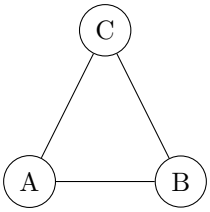
tikz

3.2 Types of Graphs with Figurative Examples

Below are common types of graphs used in graph-based learning, each accompanied by a simple illustration using TikZ.

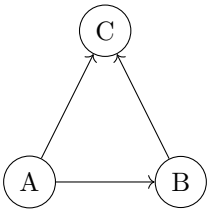
Undirected Graph

An undirected graph has edges that do not have a direction. Each connection between nodes is bidirectional.



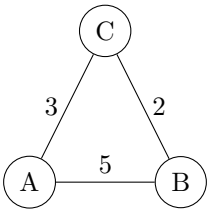
Directed Graph

A directed graph has edges with a direction, indicating a one-way relationship between nodes.



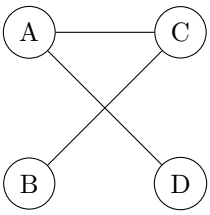
Weighted Graph

A weighted graph assigns weights (or values) to the edges, indicating the strength of the relationships between nodes.



Bipartite Graph

A bipartite graph divides nodes into two sets where each edge connects a node in one set to a node in the other set.



3.3 Graph Types Based on Properties Relevant to GNNs

Graphs can also be categorized based on structural and semantic properties that affect how Graph Neural Networks process and learn from them. Understanding these types is essential for selecting appropriate GNN models and designing message-passing mechanisms.

- **Homogeneous Graphs:** All nodes and edges are of the same type. These graphs are simpler and are typically used in standard GCNs. For example, a user-user interaction network on Twitter.
- **Heterogeneous Graphs:** Contain multiple types of nodes and/or edges. These require specialized heterogeneous GNNs (e.g., R-GCN, HAN) to capture the semantics of different relationships. Stack Overflow, with users, questions, and answers as different node types, is a classic example.
- **Directed Graphs:** Edges have a direction, indicating asymmetric relationships (e.g., who answered whose question). Many GNNs treat edges as bidirectional by adding reverse edges to facilitate message passing in both directions.

- **Undirected Graphs:** Edges are symmetric, with no inherent direction. This simplifies message passing as information flows equally between connected nodes.
- **Dynamic Graphs:** Graph structure or features change over time (e.g., evolving user interactions or transactions). Temporal GNNs (like TGAT or TGN) are used to model such data.
- **Static Graphs:** Graph structure remains fixed during model training. Most classical GNN models operate on static graphs.

3.4 Message Passing in Graph Neural Networks

Message passing is a key concept in Graph Neural Networks (GNNs). It allows nodes to exchange information with their neighbors to update their embeddings. Each node's new embedding depends on its own features, the features of its neighbors, and optionally, the edge features between nodes.

Message passing works by:

- Aggregating information from neighboring nodes.
- Using learnable functions to update node embeddings.
- Repeating this process for multiple layers, allowing nodes to capture information from increasingly distant neighbors.

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

3.4.1 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:

- $\mathbf{x}_i^{(k)} \in R^F$: Embedding of node i at layer k.
- $N(i)$: Neighbors of node i.
- $\mathbf{e}_{j,i} \in R^D$: 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.

3.5 Heterogeneous Graph Representation for Stack Overflow

The GNN model used in our study is based on a heterogeneous graph representation, capturing 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," and "accepted answers."

Three primary node types:

- **Users:** Representing contributors who ask and answer questions.
- **Questions:** Representing queries posted by users.
- **Answers:** Representing 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-loops" to enhance message passing in GNN models.

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

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

3.6 Heterogeneous Graph Representation for AskReddit

To evaluate the generalizability of our models, we also experiment 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:

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

Edges in the graph capture the interactions between these entities:

- `Asks` (User → Post): The user creates a post.
- `Rev_Asks` (Post → User): Reverse edge for bidirectional message passing.
- `Answers` (User → Comment): The user writes a comment.
- `Rev_Answers` (Comment → User): Reverse of the above edge.
- `Has` (Post → Comment): A post contains a comment.
- `Rev_Has` (Comment → Post): Reverse of the above edge.
- `Self_Loop` (User → User): Self-loop on user 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. However, the constructed heterogeneous graph still allows us to model the structural and contextual interactions among users and their contributions.

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

4 Experiments and Results

4.1 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.

4.2 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.

4.3 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 4 (Table 1) presents the performance metrics for the XGBoost classifier, while Table 5 (Table 1) reports the evaluation metrics for the GNN-based model, allowing a comparative analysis between the two approaches.

Table 2: Performance Metrics of the XGBoost Classifier

Metric	Value
F1-score	0.602
Accuracy	0.812
Precision	0.589
Recall	0.646
AUC	0.646

The initial performance of our GNN-based model is also evaluated using the same metrics. Table 5 provides the test results:

AskReddit Dataset

To evaluate the performance of the proposed XGBoost-based classification model for identifying active users on AskReddit dataset, we used several standard metrics. Table 4 presents the performance metrics for the XGBoost classifier, while Table 5 reports the evaluation metrics for the GNN-based model, allowing a comparative analysis between the two approaches.

Table 3: Performance Metrics of the GNN Classifier

Metric	Value
F1-score	0.583
Accuracy	0.883
Precision	0.623
Recall	0.568
AUC	0.568

Table 4: Performance Metrics of the XGBoost Classifier

Metric	Value
F1-score	0.6807
Accuracy	0.813
Precision	0.330
Recall	0.845
AUC	0.895

The initial performance of our GNN-based model is also evaluated using the same metrics. Table 5 provides the test results:

Table 5: Performance Metrics of the GNN Classifier

Metric	Value
F1-score	0.677
Accuracy	0.9122
Precision	0.625
Recall	0.954
AUC	0.954

5 Training Enhancement via Stratified Mini-Batching

During initial experiments, the model was trained using a small, class-balanced subset of the user nodes, created via hard undersampling. While this ensured equal representation from both classes, it significantly reduced the overall training data available, especially from the majority class. This led to poor generalization, as the model lacked sufficient exposure to diverse examples, particularly from the underrepresented class.

To overcome this limitation, we introduced **stratified mini-batching**, a more effective sampling strategy for handling class imbalance in node classification tasks. Instead of downsampling, we preserved the full training set and generated mini-batches in such a way that each batch contains an equal number of examples from each class. This method ensures that every gradient update is computed from a class-balanced view of the data without sacrificing the diversity and volume of the training samples.

We implemented the stratified_input_batches function to construct class-balanced mini-batches, which groups training nodes by class, shuffles each group, and creates batches by selecting an equal number of samples from each class. The function dynamically calculates the maximum number of full batches that can be formed without exhausting the smaller class, ensuring balanced training throughout.

This stratified sampling method resulted in substantial improvements in model performance, particularly in recall, precision, and F1-score. The model was able to learn better class boundaries and handle imbalanced distributions more effectively. Table 6 compares the evaluation metrics before and after applying the stratified mini-batching technique.

These results demonstrate the effectiveness of stratified mini-batching in imbalanced node classification settings. By ensuring each mini-batch contributes equally to the learning process across all classes, the model achieved significantly improved generalization, especially on the minority class. This strategy can be generalized to other graph learning problems facing similar class imbalance challenges.

Table 6: Performance Comparison After Stratified Mini-Batching

Metric	After (Stratified)
Accuracy	0.8160
Precision	0.5955
Recall	0.6552
F1-score	0.6106
AUC	0.6552

6 Conclusion

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. This graph-centric representation allowed our GNN models to capture complex user-content interactions without relying on extensive feature pre-processing.

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.

Across both datasets, GNNs showed competitive, and in some cases superior, generalization capabilities—especially in the Reddit dataset, where they significantly outperformed XGBoost in terms of recall and AUC. These findings suggest that GNNs are well-suited for influence and activity modeling tasks in social networks, particularly when node relationships are diverse and deeply interconnected.

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] A. Vaswani *et al.*, "Attention is All You Need," *arXiv preprint arXiv:1706.02216*, 2017. https://arxiv.org/pdf/1706.02216

[2] 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

[3] H. Chen and D. Kim, "GNN-Based Influence Detection in Q&A Communities," *IEEE Access*, 2024. https://ieeexplore.ieee.org/document/10802916

[4] 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

[5] 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

[6] 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

[7] 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