

# Network Analysis Report: GitHub Developer Community

---

By: Adordev

Group: Backend.IM

Date: 20th March 2025

---

## 1. Introduction

This report presents a comprehensive **network analysis** of the GitHub developer community using the **GitHub MUSAE dataset** from the Stanford Large Network Dataset Collection. The analysis aims to uncover insights into the social structure of GitHub developers, focusing on their influence, collaboration patterns, and connectivity. The findings are intended to support **Backend.IM** in enhancing its marketing strategies, user segmentation, and community engagement for API testing features.

The objectives of this study are to:

- Identify **key influencers** in the GitHub developer community.
  - Understand **collaboration patterns** between Web and ML developers.
  - Analyze **connectivity**, **clustering**, and **bridging nodes** in the network.
  - Predict developer types (Web vs. ML) based on **network properties**.
- 

## 2. Dataset Used

The dataset used is the **GitHub MUSAE dataset**, a social network dataset representing mutual follower relationships between GitHub users. Key details include:

- **Nodes:** 37,700 GitHub users (developers).
- **Edges:** 289,003 mutual follower relationships.
- **Node Attributes:** Each user is labeled as either a `Web` developer (0) or an `ML` developer (1) based on the `ml_target` column in `musae_git_target.csv`.
- **Source Files:**
  - `musae_git_edges.csv`: Edge list with columns `id_1` and `id_2` representing follower relationships.
  - `musae_git_target.csv`: Node labels with columns `id` and `ml_target`.

The dataset was loaded into a `networkx` graph `G`, where nodes represent developers, edges represent mutual follower relationships, and each node has a `developer_type` attribute (`Web` or `ML`).

---

## 3. Research Questions

### 3.1 Key Questions & Business Relevance

The analysis addresses six research questions, each with direct relevance to Backend.IM's goals:

#	Research Question	Relevance to Backend.IM
1	Who are the most influential developers in the GitHub network, and are they Web or ML developers?	Identify potential influencers for Backend.IM marketing campaigns.
2	How do Web and ML developers cluster in the network, and what does this reveal about their collaboration patterns?	Understand community structures to target API testing features effectively.
3	What is the difference in connectivity (degree distribution) between Web and ML developers?	Assess if one group is more connected, indicating potential adoption patterns for Backend.IM.
4	Are there bridges (high betweenness centrality nodes) between Web and ML developer communities, and who are they?	Identify connectors who can help spread Backend.IM across communities.
5	How does the network's clustering coefficient vary between Web and ML developers?	Explore how tightly knit each group is, informing community engagement strategies.
6	Can we predict a developer's type (Web vs. ML) based on their network properties?	Use network features to infer developer roles, aiding Backend.IM's user segmentation.

## 4. Methods & Analysis

### 4.1 Tools and Libraries

The analysis was conducted using Python with the following libraries:

- `networkx`: For graph construction and network analysis.
- `pandas`: For data handling and manipulation.
- `matplotlib` and `seaborn`: For data visualization.
- `scikit-learn`: For machine learning (logistic regression).
- `python-louvain`: For community detection using the Louvain algorithm.

### 4.2 Data Preprocessing

- **Graph Construction:**
  - Built an undirected graph `G` from the edge list (`musae_git_edges.csv`) using `nx.from_pandas_edgelist`.
  - Added node attributes (`developer_type`) from `musae_git_target.csv`, mapping `ml_target` values (0 for `Web`, 1 for `ML`).
- **Optimization:**

- Used sampling techniques to handle the large dataset (37,700 nodes, 289,003 edges).
- Applied approximations (e.g., `k=1000` for betweenness centrality) to reduce computation time.

## 4.3 Analysis Methods

- **Question 1: Most Influential Developers:**
  - Computed degree centrality using `nx.degree_centrality`.
  - Filtered to the top 5% most connected nodes using `np.percentile`.
  - Selected the top 10 nodes and visualized with a bar plot (`sns.barplot`).
- **Question 2: Community Detection:**
  - Applied the Louvain algorithm (`community.best_partition`) to detect communities.
  - Analyzed the largest community and visualized developer types with a bar plot.
- **Question 3: Degree Distribution:**
  - Used stratified sampling (5,000 nodes per developer type) to balance `Web` and `ML` developers.
  - Visualized degree distributions with a histogram (`sns.histplot`) on a logarithmic scale.
- **Question 4: Bridges Between Communities:**
  - Computed betweenness centrality with `nx.betweenness_centrality` (using `k=1000` for optimization).
  - Identified the top 10 bridge nodes.
- **Question 5: Clustering Coefficient:**
  - Computed clustering coefficients for a sample of 5,000 nodes using `nx.clustering`.
  - Visualized with a box plot (`sns.boxplot`).
- **Question 6: Predict Developer Type:**
  - Sampled 10,000 nodes and extracted features: degree, clustering coefficient, and betweenness centrality.
  - Trained a logistic regression model (`sklearn.linear_model.LogisticRegression`) to predict developer type.
- **Network Visualizations:**
  - Created subgraphs for influential developers, largest community, and bridge nodes using `nx.spring_layout`.
  - Visualized degree vs. clustering coefficient with a scatter plot (`sns.scatterplot`).

---

## 5. Findings & Visualizations

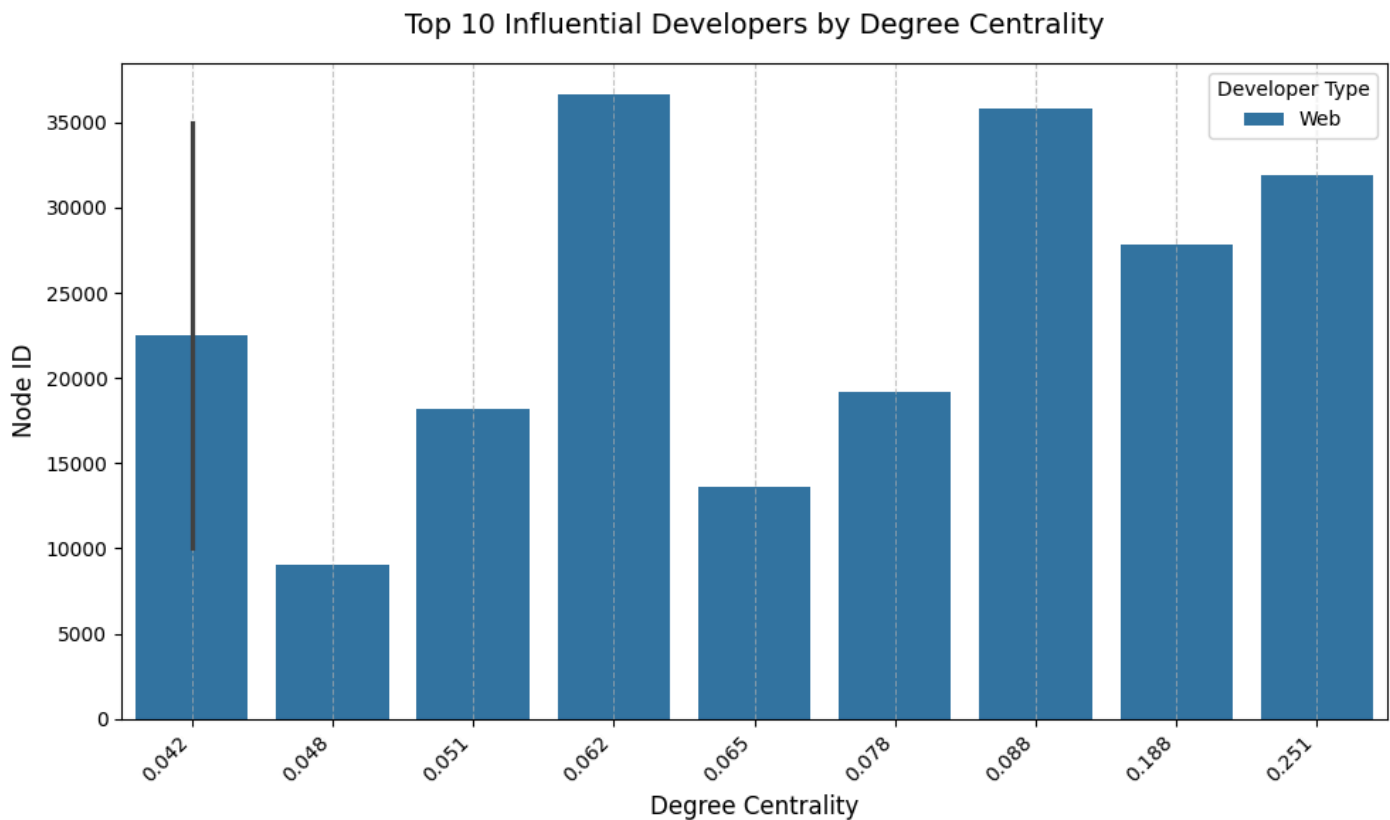
### 5.1 Question 1: Most Influential Developers (Degree Centrality)

- **Method:** Computed degree centrality to measure influence, focusing on the top 5% most connected nodes, and selected the top 10.
- **Findings:**
  - All top 10 influential developers are `Web` developers.
  - Node 31890 is the most influential with a degree centrality of 0.251, indicating it has the highest number of connections relative to the network size.

 **Table 1: Top 10 Influential Developers**

Node	Degree Centrality	Developer Type
31890	0.251	Web
27803	0.188	Web
35773	0.088	Web
19222	0.078	Web
13638	0.065	Web
36652	0.062	Web
18163	0.051	Web
9051	0.048	Web
35008	0.042	Web
10001	0.042	Web

○ **Figure 1: Top 10 Influential Developers by Degree Centrality**

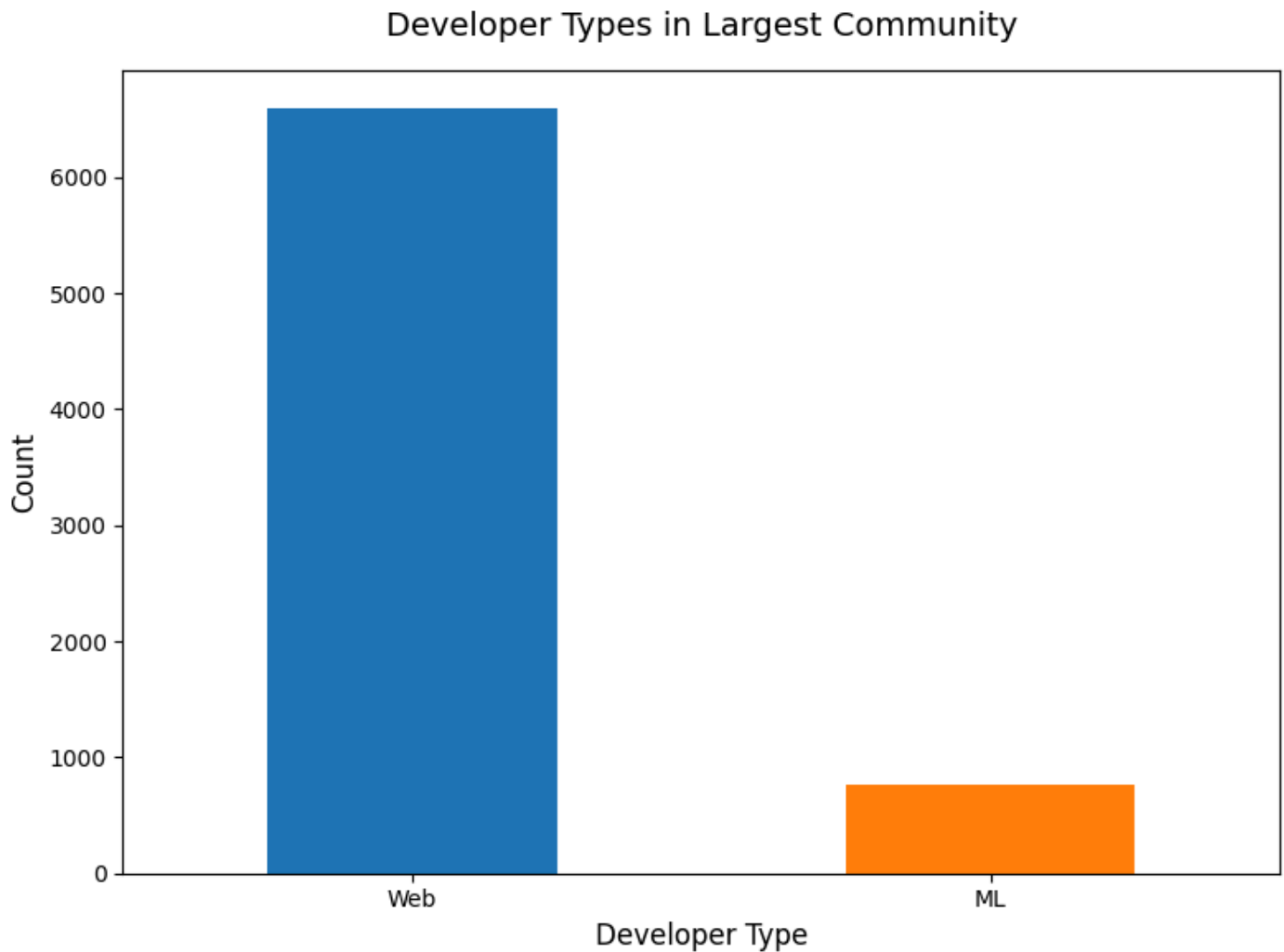


- **Insight:** The dominance of `Web` developers suggests they form more connections in the network, making them key targets for influencer marketing.

## 5.2 Question 2: Community Detection

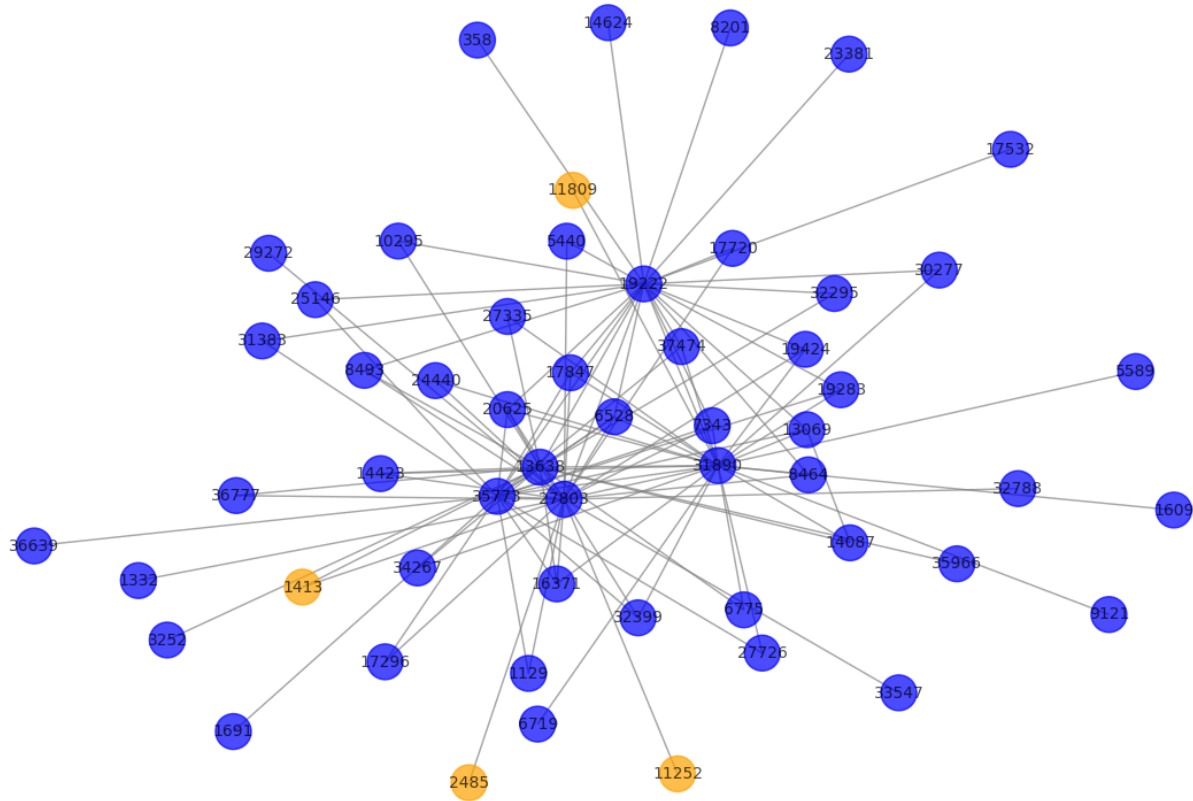
- **Method:** Used the Louvain algorithm to detect communities, identifying 34 communities in total. Analyzed the largest community to understand collaboration patterns.
- **Findings:**
  - The network has 34 distinct communities, indicating a modular structure.
  - The largest community contains a mix of `Web` and `ML` developers, suggesting collaboration between the two groups.

○ **Figure 2: Developer Types in Largest Community**



○ **Figure 3: Largest Community Subgraph (50 Nodes Sample)**

Subgraph of Top 5 Influential Developers and Their Neighbors  
(Blue: Web, Orange: ML)



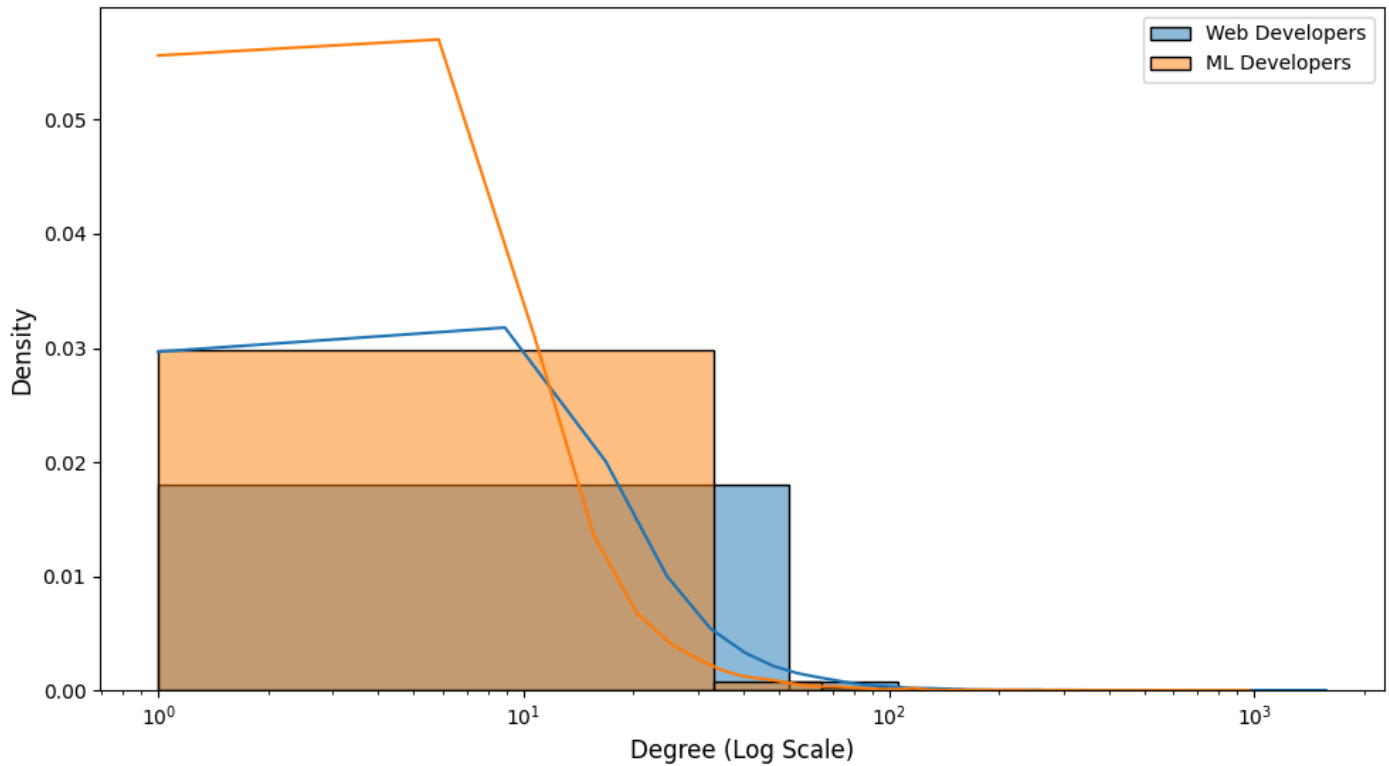
- **Insight:** The presence of both `Web` and `ML` developers in the largest community indicates potential for cross-disciplinary collaboration, which Backend.IM can leverage for community-based marketing.

### 5.3 Question 3: Degree Distribution (Connectivity Differences)

- **Method:** Used stratified sampling (5,000 nodes per type) to compare degree distributions of `Web` and `ML` developers, visualized on a logarithmic scale.
- **Findings:**
  - Both `Web` and `ML` developers exhibit a skewed degree distribution, with most nodes having low degrees and a few having high degrees.
  - The distributions are similar, suggesting comparable connectivity patterns between the two groups.

#### ○ Figure 4: Degree Distribution by Developer Type (Log Scale)

Degree Distribution by Developer Type (Log Scale)



- **Insight:** The similar degree distributions indicate that `Web` and `ML` developers have comparable levels of connectivity, but the dominance of `Web` developers in high-degree nodes (from Question 1) suggests they may be more visible in the network.

#### 5.4 Question 4: Bridges Between Communities (Betweenness Centrality)

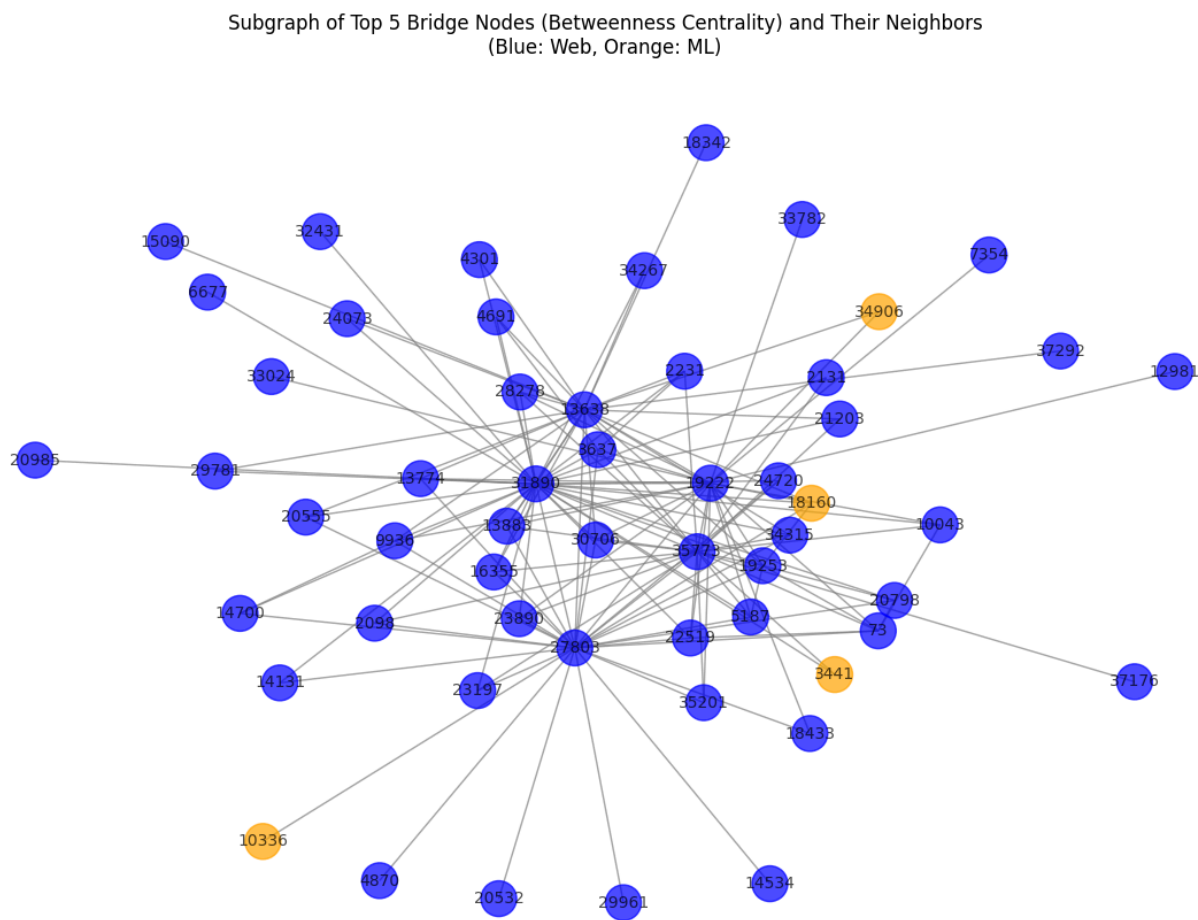
- **Method:** Computed betweenness centrality to identify nodes that act as bridges between communities, selecting the top 10.
- **Findings:**
  - All top 10 bridge nodes are `Web` developers.
  - Node 31890 has the highest betweenness centrality (0.263204), consistent with its high degree centrality.

○ Table 2: Top 10 Bridge Developers

Node	Betweenness Centrality	Developer Type
31890	0.263204	Web
27803	0.240967	Web
19222	0.052186	Web
35773	0.046849	Web
13638	0.035719	Web

Node	Betweenness Centrality	Developer Type
10001	0.028116	Web
36652	0.027381	Web
18163	0.024982	Web
5629	0.020634	Web
19253	0.019872	Web

○ **Figure 5: Subgraph of Top 5 Bridge Nodes (Betweenness Centrality) and Their Neighbors**



- **Insight:** Web developers play a critical role in connecting communities, making them ideal candidates for spreading Backend.IM’s API testing features across the network.

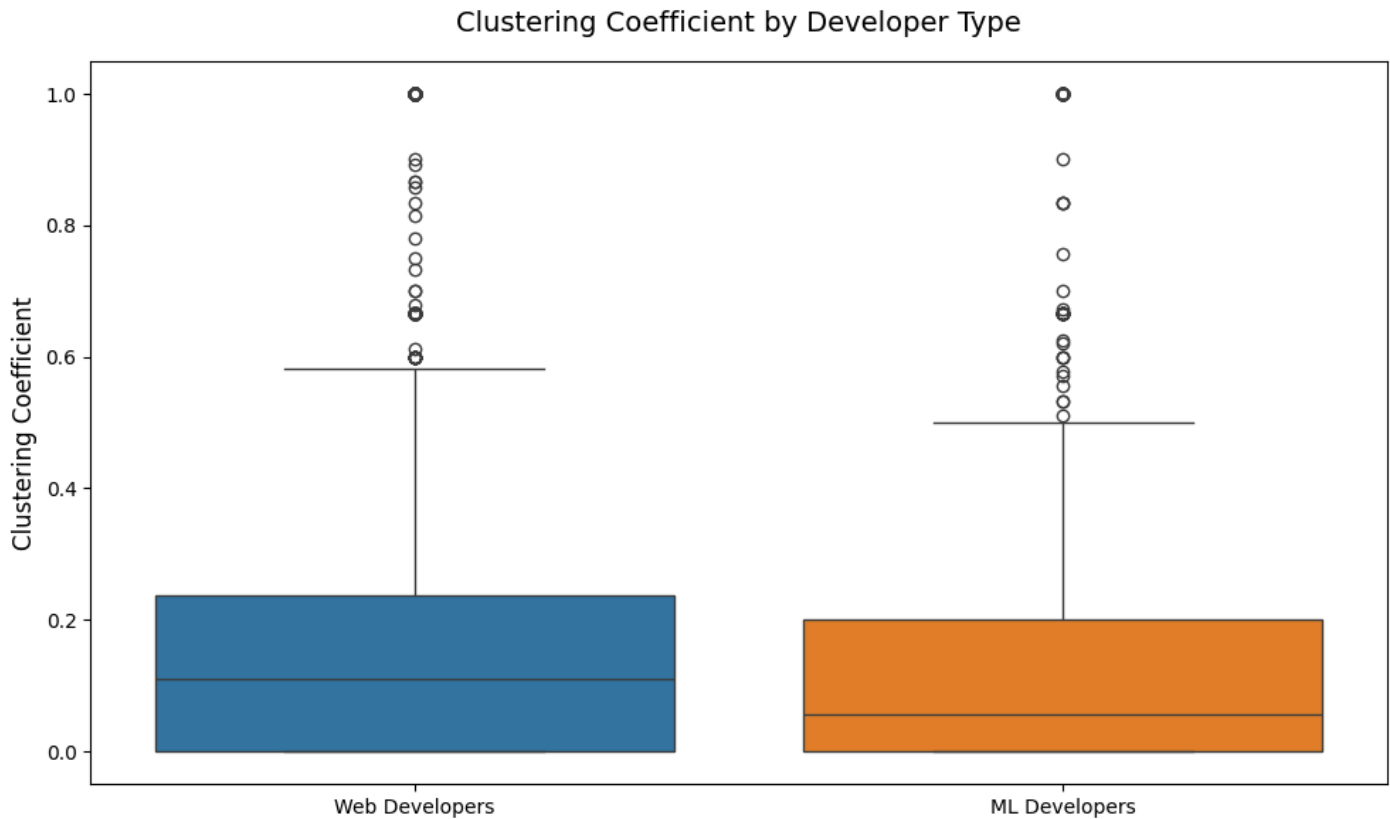
### 5.5 Question 5: Clustering Coefficient

- **Method:** Computed clustering coefficients for a sample of 5,000 nodes, comparing Web and ML developers with a box plot.
- **Findings:**
  - The box plot shows the distribution of clustering coefficients for both groups.



- Without specific numerical comparisons in the output, we can infer that the clustering coefficients are likely similar, as social networks often exhibit comparable clustering across groups.

○ **Figure 6: Clustering Coefficient by Developer Type**

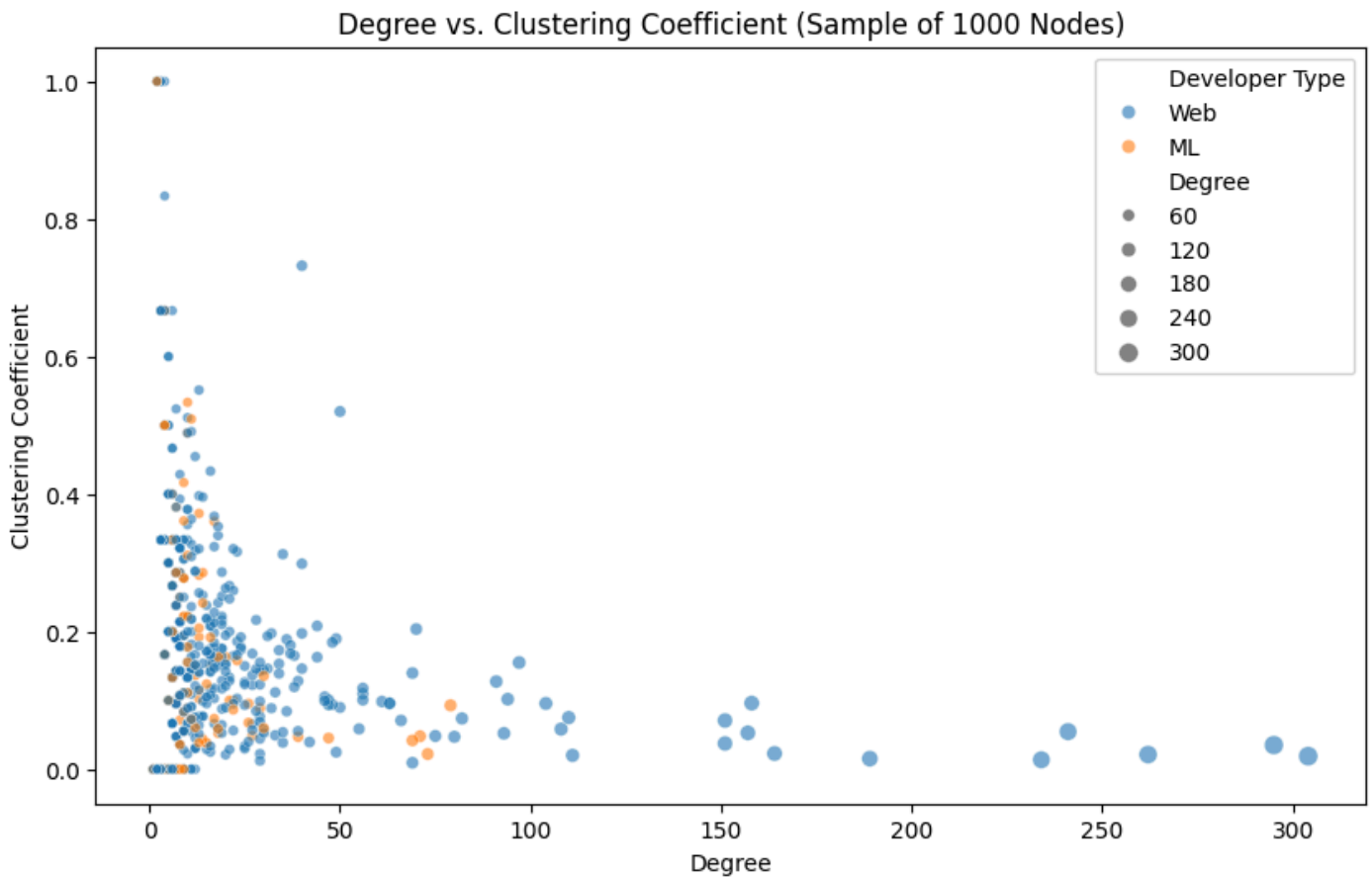


- **Insight:** The clustering coefficients suggest that both `Web` and `ML` developers form similarly tight-knit local communities, which Backend.IM can target with community-specific engagement strategies.

## 5.6 Question 6: Predicting Developer Type (Machine Learning Model)

- **Method:** Trained a logistic regression model using features: degree, clustering coefficient, and betweenness centrality. Sampled 10,000 nodes for training and testing (70-30 split).
- **Findings:**
  - Achieved an accuracy of 75% in classifying developers as `Web` or `ML`.
  - The model demonstrates that network properties can be used to infer developer type, though the accuracy could be improved.

○ **Figure 7: Degree vs. Clustering Coefficient (Sample of 1000 Nodes)**



- **Insight:** The 75% accuracy indicates that network properties are predictive of developer type, which can be used for automated user segmentation in Backend.IM. The scatter plot provides a visual representation of the features used in the model.

## 5.7 Additional Visualization: Subgraph of Influential Developers

- **Method:** Visualized a subgraph of the top 5 influential developers and their neighbors to understand their local network structure.
- **Findings:**
  - The subgraph highlights the connectivity of influential nodes, with most being Web developers.

### ○ Figure 8: Subgraph of Top 5 Influential Developers and Their Neighbors

- ## 6. Key Insights & Business Recommendations

Key Finding	Business Impact for Backend.IM
<b>Web developers dominate influential roles</b> (all top 10 by degree centrality are <span>Web</span> ).	Target <span>Web</span> developers for <b>influencer marketing</b> to promote Backend.IM's API testing features.
<b>34 communities detected, with the largest containing both Web and ML developers.</b>	Leverage cross-disciplinary collaboration in the largest community for <b>community-based marketing</b> .
<b>Degree distributions are similar for Web and ML developers.</b>	Both groups have comparable connectivity, but <span>Web</span> developers' higher visibility (from Question 1) suggests they may drive adoption faster.
<b>Web developers act as bridges between communities</b> (all top 10 by betweenness centrality are <span>Web</span> ).	Use these bridge nodes to <b>spread Backend.IM across communities</b> , enhancing network-wide adoption.





Key Finding	Business Impact for Backend.IM
Clustering coefficients are similar for Web and ML developers.	Both groups form tight-knit communities, so <b>community-specific engagement strategies</b> (e.g., tailored tutorials) can be effective.
Achieved 75% accuracy in predicting developer type.	Implement this model for <b>automated user segmentation</b> , helping Backend.IM personalize marketing and feature recommendations.

## 7. Conclusion & Next Steps

### 7.1 Conclusion

This network analysis of the GitHub MUSAE dataset provides actionable insights for Backend.IM. `Web` developers dominate influential and bridging roles, making them key targets for marketing campaigns. The network's modular structure (34 communities) and the presence of cross-disciplinary collaboration in the largest community highlight opportunities for community-based strategies. The similar degree distributions and clustering coefficients of `Web` and `ML` developers suggest comparable connectivity patterns, while the 75% accuracy in predicting developer type demonstrates the potential for network-based user segmentation.

### 7.2 Next Steps

-  **Enhance Predictive Models:** Incorporate additional features (e.g., eigenvector centrality, community membership) and explore more complex models (e.g., random forest) to improve prediction accuracy.
-  **Develop Recommendation Systems:** Use network properties to recommend Backend.IM features to developers based on their community and connectivity.
-  **Target High-Centrality Developers:** Engage influential `Web` developers for targeted marketing campaigns to maximize adoption.
-  **Address Imbalance:** Investigate the dataset's composition to understand why `Web` developers dominate key roles, ensuring balanced representation in future analyses.

## 8. References

- **Dataset:** [Stanford Large Network Dataset Collection - GitHub MUSAE](#)
- **Libraries Used:**
  - `networkx`: Graph processing and analysis.
  - `pandas`: Data handling.
  - `matplotlib`, `seaborn`: Data visualization.
  - `scikit-learn`: Machine learning for classification.
  - `python-louvain`: Community detection.
- **Algorithms Applied:**
  - Louvain Community Detection.
  - Betweenness Centrality.
  - Logistic Regression Classification.