# Greedy/Heuristic vs. ML for Community Detection and Influence Maximization

Ogbuchi Chidiebere[1] and Forero Laura[2]

1. CentraleSupélec

Repository of the project: https://github.com/Lala341/CommunityMaximizationGraphs.git.

## 1 Datasets

The goal of this project is to examine and contrast greedy/heuristic methods with ML methods for addressing community detection and influence maximization problems using real data from SNAP datasets. Specifically, two social networks, Facebook and English Twitch, were employed for analysis.

### 1.1 Facebook

The Facebook dataset consists of anonymized 'circles' or 'friends lists' collected from survey participants via a Facebook app. It includes node features, circles, and ego networks. The data has been anonymized by replacing Facebook-internal IDs and obscuring feature interpretations. Despite this, users' shared affiliations can be identified without revealing individual details. The dataset contains 4039 nodes and 88234 edges, with an average clustering coefficient of 0.6055 and a diameter of 8.

Table 1.1: Dataset Statistics

| Statistic | Value |
|---|---|
| Nodes | 4039 |
| Edges | 88234 |
| Nodes in largest WCC | 4039 (1.000) |
| Edges in largest WCC | 88234 (1.000) |
| Nodes in largest SCC | 4039 (1.000) |
| Edges in largest SCC | 88234 (1.000) |
| Average clustering coefficient | 0.6055 |
| Number of triangles | 1612010 |
| Fraction of closed triangles | 0.2647 |
| Diameter (longest shortest path) | 8 |
| 90-percentile effective diameter | 4.7 |

## 1.2 Twitch

The Twitch dataset comprises user-user networks of gamers who stream in English. Nodes represent individual users, while links denote mutual friendships between them. Vertex features are derived from games played, liked, location, and streaming habits. These networks were collected in May 2018 and are used for node classification and transfer learning tasks. The supervised task involves binary node classification to predict whether a streamer uses explicit language. The dataset contains 7,126 nodes and 35,324 edges, with a density of 0.002 and transitivity of 0.042. Additionally, tasks such as transfer learning, link prediction, community detection, and network visualization are possible with this dataset.

Table 1.2: Dataset Statistics

| **Statistic** | **EN** |
|---|---|
| Nodes | 7,126 |
| Edges | 35,324 |
| Density | 0.002 |
| Transitivity | 0.042 |

# 2 Community Detection

Community detection in complex networks has become a fundamental aspect of network analysis, revealing hidden structures and patterns within intricate systems. Networks, such as social networks, biological networks, and technological networks, often exhibit modular organization where nodes form tightly-knit groups or communities. Understanding these communities is crucial for various applications, ranging from targeted marketing strategies to identifying functional modules in biological systems. In this report, we delve into the significance of community detection and explore three widely-used methods: Clique Percolation Method, Louvain Method, and Girvan-Newman Method.

## 2.1 Methods of Community Detection

In this delivery we explore Traditional greedy/heuristic and ML approaches

### 2.1.1 Methods of Community Detection

**Clique Percolation Method (palla):** The Clique Percolation Method (CPM) is a community detection algorithm that focuses on identifying overlapping communities. It defines k-cliques, which are subsets of nodes where each node is directly connected to every other node in the subset. Overlapping communities are then formed by percolating through these k-cliques. The algorithm uses the concept of k-clique percolation coefficient (P(k)), representing the fraction of shared nodes between different k-cliques, to determine the strength of community membership. Since there was no python library for it, we implement ours following the algorithm.‘

$$P(k) = \frac{|V_1 \cap V_2|}{\min(|V_1|, |V_2|)}$$

**Custom Clique Percolation Method (palla overlapping):** Similar to above Clique Percolation Algortihm, we implement our own custom method using the same algorithm but nodes do not overlap in the community.

**Louvain:** The Louvain Method is a modularity-based algorithm that optimizes the modularity of a partitioned network. Modularity ($Q$) is a metric that measures the quality of a community structure within a network. The Louvain Method iteratively optimizes modularity by moving nodes between communities, aiming to maximize the overall modularity of the network.

$$Q = \frac{1}{2m} \sum_{ij} \left[ A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where $A_{ij}$ is the edge weight between nodes $i$ and $j$, $k_i$ and $k_j$ are the degrees of nodes $i$ and $j$, $m$ is the total edge weight in the network, $\delta(c_i, c_j)$ is the Kronecker delta function, and $c_i$ and $c_j$ are the community assignments of nodes $i$ and $j$, respectively.

**Girvan-Newman Method:** The Girvan-Newman Method employs edge betweenness centrality to identify communities by iteratively removing edges with the highest betweenness centrality. This process gradually breaks the network into distinct communities, revealing the hierarchical structure of the network.

$$g_{ij} = \sum_{st} \frac{\sigma_{st}(i, j)}{\sigma_{st}}$$

where $g_{ij}$ is the betweenness centrality of edge $(i, j)$, $\sigma_{st}$ is the total number of shortest paths from node $s$ to node $t$, and $\sigma_{st}(i, j)$ is the number of those paths passing through edge $(i, j)$.

### 2.1.2 ML Approach

**Node2Vec:** This is a method for learning continuous representations for nodes in any (possibly weighted) graph. It is based on the idea of preserving network neighborhoods of nodes in the graph. It uses a flexible notion of a node's network neighborhood and is designed to learn embeddings that respect both local and global network structure. Node2Vec first generates a random walk sequence, and then applies the Skip-Gram model to generate node embeddings.

**GAT (Graph Attention Networks):** This is a type of neural network designed specifically for graph data. GAT introduces the attention mechanism as a substitute for the static graph convolution operation. It allows for a larger degree of freedom by assigning different importances to different nodes within a neighborhood. This allows it to capture more complex patterns. GAT can be applied directly on graph-structured data without needing to convert the graph into a sequence or a grid, preserving the graph's topological properties.

To determine the quality of the embeddings, we perform
**Cosine Similarity:** Given two vectors $A$ and $B$, the cosine similarity ($cos\_sim$) is calculated as:

$$cos\_sim(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

The range of cosine similarity is [-1, 1], where 1 indicates identical vectors, 0 means the vectors are orthogonal, and -1 implies opposite directions.

**Euclidean Distance:** - **Formula:** Given two vectors $A$ and $B$ with $n$ dimensions, the Euclidean distance ($euclidean\_dist$) is calculated as:

$$euclidean\_dist(A, B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

The range of Euclidean distance is [0, +inf], with 0 indicating identical vectors.

**Manhattan Distance:** Given two vectors $A$ and $B$ with $n$ dimensions, the Manhattan distance ($manhattan\_dist$) is calculated as:

$$manhattan\_dist(A, B) = \sum_{i=1}^{n} |A_i - B_i|$$

The range of Manhattan distance is [0, +inf].

After the embeddings we perform **Kmeans, Spectral and Agglomerative clustering models** to generate the communities. Further we use ML methods to determine the quality of the clusters created. Namely:

**Silhouette Score:** Given a data point $i$, its silhouette score ($S(i)$) is calculated as:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- $a(i)$ is the average distance from the $i$-th point to other points in the same cluster, and $b(i)$ is the average distance from the $i$-th point to points in the nearest cluster that the $i$-th point is not a part of.

**Inertia (within-cluster sum of squares):** For a set of clusters $C_i$, the inertia ($I$) is given by:

$$I = \sum_{i=1}^{k} \sum_{j \in C_i} \|x_j - \mu_i\|^2$$

- $k$ is the number of clusters, $x_j$ is a data point, $\mu_i$ is the centroid of cluster $C_i$.

**Davies-Bouldin Score (db):** Given clusters $C_i$ and $C_j$ with centroids $\mu_i$ and $\mu_j$, and the average intra-cluster distance $d(C_i, C_j)$, the Davies-Bouldin Score ($DB$) is calculated as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left( \frac{\text{intra-cluster distance}(\mu_i) + \text{intra-cluster distance}(\mu_j)}{d(\mu_i, \mu_j)} \right)$$

**Calinski-Harabasz Score (ch):** Given the between-cluster variance $B(k)$ and within-cluster variance $W(k)$ for $k$ clusters, the Calinski-Harabasz Score ($CH$) is given by:

$$CH = \frac{B(k)}{(k-1)} \times \frac{N-k}{W(k)}$$

- $N$ is the total number of data points.

## 2.2 Metrics

### 2.2.1 Network Graph Metrics

We initially analyze the structure of the graph using the following metrics.

**Basic Description of Graph:** Indicates the number of nodes and edges.

**The Average Degree:** expressed as:

$$\text{Average Degree} = \frac{\sum_{i=1}^{n} \text{Degree}(v_i)}{n}$$

where:

- $n$ is the number of nodes in the graph,
- $v_i$ represents each node in the graph, and
- $\text{Degree}(v_i)$ is the degree of the $i$-th node, i.e., the number of edges incident to $v_i$.

**Density:** Density measures the proportion of edges in a graph relative to the total possible edges.

$$\text{Density} = \frac{2 \times \text{Number of Edges}}{\text{Number of Nodes} \times (\text{Number of Nodes} - 1)}$$

**Clustering Coefficients:** Clustering coefficient measures the extent to which nodes in a graph tend to cluster together.

$$\text{Clustering Coefficient of Node } v_i = \frac{\text{Number of Actual Edges between Neighbors of } v_i}{\text{Number of Possible Edges between Neighbors of } v_i}$$

The clustering coefficient for the entire graph is often expressed as the average of individual clustering coefficients.

**Average Clustering Coefficient:** It is the average of the clustering coefficients for all nodes in the graph.

$$\text{Average Clustering Coefficient} = \frac{\sum_{i=1}^{n} \text{Clustering Coefficient of Node } v_i}{n}$$

These measures provide insights into the structural characteristics of a graph, such as how interconnected its nodes are.

### 2.2.2 Community Graph Metrics

In addition to the Network graph metrics we implement additional metrics community cluster using. Specifically:

**Traditional Greedy/Heuristic method**

**Number of Communities:** - This represents the count of distinct communities or groups within a network. - Count the number of identified communities in the network.

**Average Community Size:** - It provides the average size of communities in the network. -

$$\text{Average Community Size} = \frac{\text{Total Number of Nodes}}{\text{Number of Communities}}$$

**Community Details:** - This may include information about the nodes within each community.
**Top Level Communities:** - These are the higher-level communities or groups that might encompass sub-communities.
**Last Level Communities:** - These are the lowest-level communities or sub-communities within the network.
**Modularities:** - Modularity measures the degree to which a network can be divided into non-overlapping communities. -

$$\text{Modularity} = \sum_{i=1}^{n} \left[ \frac{e_{ii}}{m} - \left( \frac{a_i}{2m} \right)^2 \right]$$

where: - $e_{ii}$ is the number of edges within community $i$, - $m$ is the total number of edges in the network, - $a_i$ is the sum of the degrees of nodes in community $i$, - $n$ is the number of communities.
**Edge Graph Cuts:** - It measures the number of edges that need to be removed to break a graph into disconnected components.
**Inter-Community Edge Graph Cuts:** - Similar to edge graph cuts, but focusing on edges connecting different communities.
**Conductances:** - Conductance measures the "tightness" of a community, indicating how well-separated it is from the rest of the network. - **Formula (for a community $i$):**

$$\text{Conductance}(i) = \frac{\text{Number of Edges Leaving Community } i}{\min(\text{Volume}(i), \text{Volume}(\text{Complement of } i))}$$

where: - $\text{Volume}(i)$ is the sum of degrees of nodes in community $i$.

Whereas in addition to Density, Modularity and other metrics, we performed additional metrics for the **ML approach**. They are as follows:

1. **Accuracy:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2. **F1 Score:**

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3. **Precision:**

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

4. **Recall:**

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

5. **Adjusted Rand Score:**

$$\text{Adjusted Rand Score} = \frac{\text{Index} - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}$$

6. **Adjusted Mutual Information Score:**

$$\text{Adjusted Mutual Information Score} = \frac{\text{Mutual Information} - \text{Expected Mutual Information}}{\text{Maximal Mutual Information} - \text{Expected Mutual Information}}$$

7. **Homogeneity Score:**

$$\text{Homogeneity Score} = 1 - \frac{H(C|K)}{H(C)}$$

where $H(C|K)$ is the conditional entropy of the class given the cluster assignments, and $H(C)$ is the entropy of the class labels.

## 2.3 Results

### 2.3.1 Facebook Data

**Traditional approach**

Due to complexity of the facebook data, we initially select only a sample of the dataset as in table 2.1

Result of the selected subgraph are as follows in table 2.2

**key:** nc: not calculated, n/a: not available

The results indicate Louvain as the best model as it had the best modularity and performed better than the others with less time taken. Thus, we run the full facebook dataset with louvain algorithm.

**Machine learning approach**

We implement Node2Vec ML approach and generate community using Kmeans, Spectral and Agglomerative methods. It took 1999.89 secs to generate the embeddings with parameters: dimensions=64, walklength=30, num walks=200, and workers=4. We check the quality of the embedding generated and then perform the clustering.

We generate community of clusters k=5 using the Kmeans, Spectral and Agglomerate method as aforementioned. The results are displayed in the figures as follows.

Table 2.5 shows the 3 methods generating slightly similar results.

### 2.3.2 Twitch Data

**Traditional approach**

Since Louvain produced the best result for the Facebook dataset, we implement just the algorithm for the Twitch dataset. The table 2.6 are the thereof.

**Machine Learning approach**

We maintain same parameter as defined earlier in the facebook experiment for Node2Vec to generate embedding which took 3702.43secs. Also, we try the GAT (Graph Attention Networks) using tensors and parameters of epochs=200 lr=0.01, weight_decay=5e-4 and zero grad optimizer to generate the embedding in 2908.43 secs.

This time we only use the K-means on the Node2Vec embedding since the others produced slightly similar results.

The twitch community is seen in figure 2.5.

## 2.4 Discussion:

**Node2Vec:**

1. **Algorithm Type:** Node2Vec is an embedding-based approach that learns node representations in a continuous vector space based on network structure.

| Description | Value |
|---|---|
| Number of Nodes | 201 |
| Number of Edges | 997 |
| Average Degree | 9.92 |
| Density | 0.05 |
| Average Clustering Coefficient | 0.657 |

Table 2.1: Facebook: Selected Subgraph Metrics

| Analysis | Palla (k = 4) | Palla (k = 4, overlapping) | Louvain | Girvan Newman (num_levels 4) |
|---|---|---|---|---|
| Number of Communities | 196 | 17628 | 8 | 5 |
| Average Community Size | 2 | 4 | 25 | 40 |
| Maximum Community Size | 155 | 4 | 70 | 197 |
| Average Conductance | 8.89 | 46.59 | 3.32 | 0.80 |
| Maximum Conductance | 44.00 | 78.50 | 7.13 | 1.00 |
| Average Density | 0.00 | 1.00 | 0.39 | nc |
| Maximum Density | 0.08 | 1.00 | 0.79 | nc |
| Average Edge Graph Cuts | 9.25 | 186.36 | 73.00 | 1.60 |
| Maximum Edge Graph Cuts | 72.00 | 314.00 | 177.00 | 4.00 |
| Modularity | n/a | nc | 0.453 | 0.000 |
| Overlapping Nodes | 0 | 141 | n/a | n/a |
| Maximum Pairwise Cut | 44.00 | 15.00 | 55.00 | 1.00 |
| Average Pairwise Cut | 0.13 | 7.00 | 10.43 | 0.40 |
| Time Taken (secs) | 6.54 | 3779.74 | 0.85 | 2.01 |

Table 2.2: Facebook: Traditional Community Detection Metrics

| **Analysis** | **Louvain** |
|---|---|
| Total nodes | 4039 |
| Total edges | 88234 |
| Average degree in network | 43.691 |
| Density of network | 0.011 |
| Average clustering coefficient | 0.606 |
| **Communities** | |
| Number of communities | 16 |
| Average community size | 252 |
| Maximum Community Size | 548 |
| Average conductance | 1.82 |
| Maximum conductance | 3.823 |
| Average density | 0.30 |
| Maximum density | 0.867 |
| Average edge graph cuts | 433.75 |
| Maximum edge graph cuts | 1270 |
| Modularity | 0.835 |
| Maximum Pairwise Cut | 891 |
| Average Pairwise Cut | 28.92 |
| Time taken (secs) | 55.49 |

Table 2.3: Facebook: Louvain Community Detection Metrics

| | **Kmeans vs Spectral** | **Kmeans vs Agglo** | **Agglo vs Spectral** |
|---|---|---|---|
| **Accuracy** | 0.992 | 0.99 | 0.991 |
| **F1** | 0.99 | 0.987 | 0.986 |
| **Precision** | 0.99 | 0.983 | 0.983 |
| **Recall** | 0.989 | 0.991 | 0.991 |
| **Adj_Rand_Score** | 0.982 | 0.975 | 0.981 |
| **Adj_Mut_Info_Score** | 0.97 | 0.962 | 0.97 |
| **Hom_Score** | 0.97 | 0.964 | 0.972 |

Table 2.4: Facebook Node2Vec: Comparison of Pairwise Clustering Algorithms

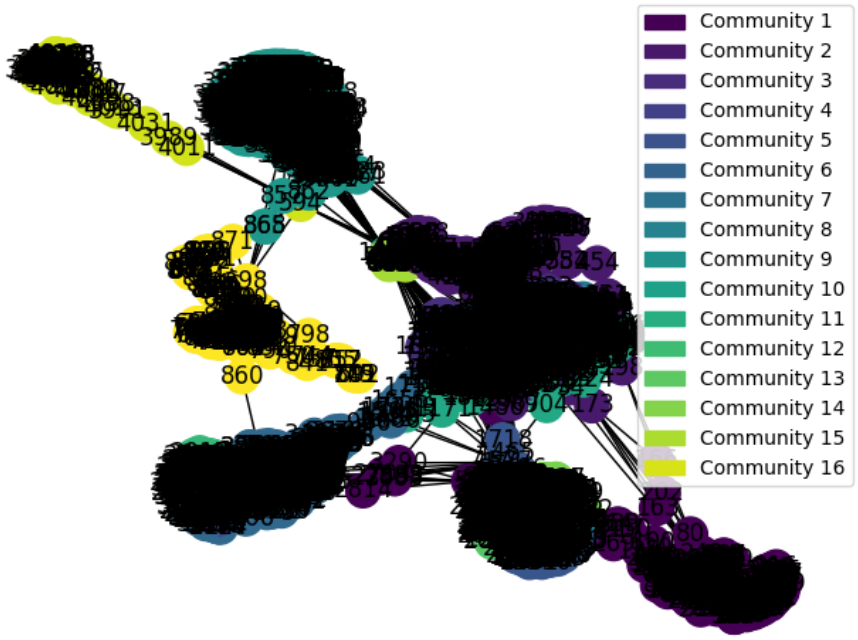


Figure 2.1: Louvain communities for Facebook full Network

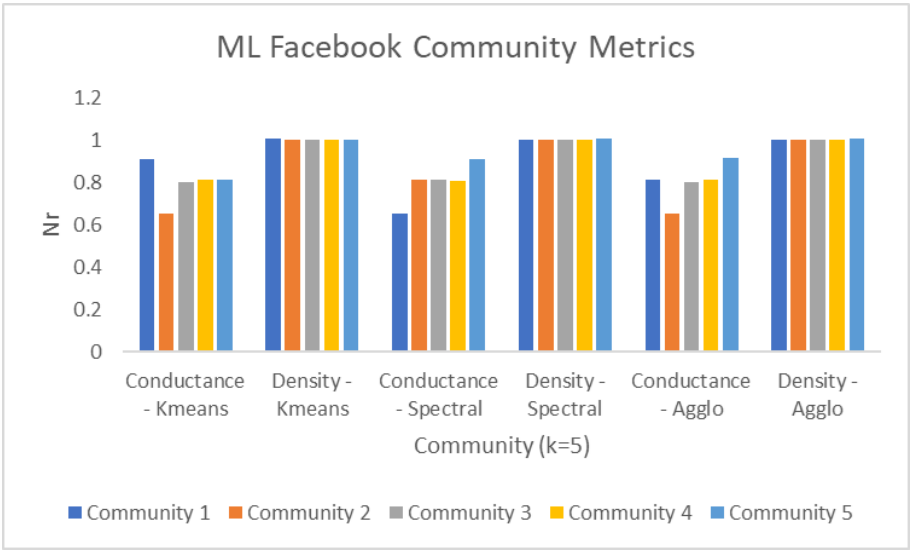


Figure 2.2: Facebook: Node2vec Metrics of Community

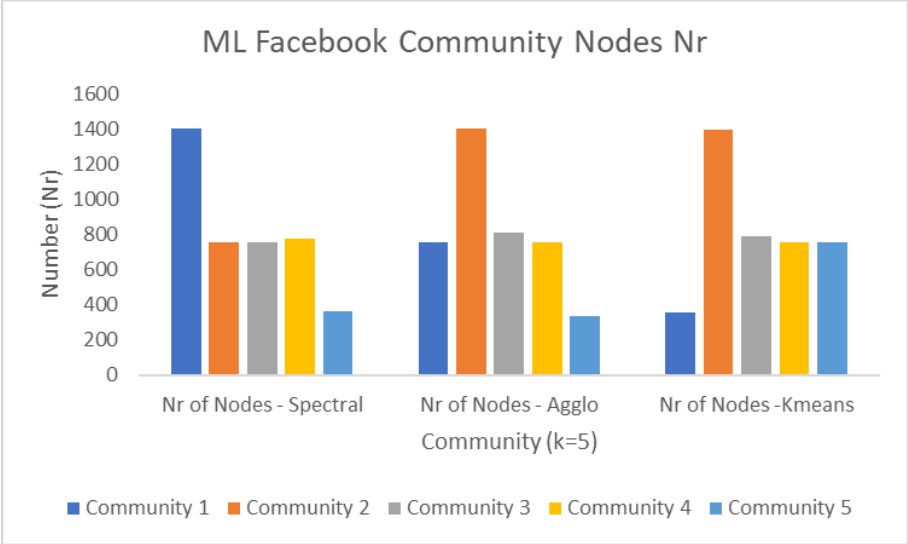


Figure 2.3: Facebook: Node2vec Size of Community

| | Kmeans | Spectral | Agglome |
|---|---|---|---|
| **Silhouette** | 0.251 | 0.25 | 0.248 |
| **Inertia** | 20851.586 | n/a | n/a |
| **Davies-Bouldin (db)** | 1.646 | 1.653 | 1.649 |
| **Calinski-Harabasz (ch)** | 627.244 | 622.832 | 618.122 |
| **Modularity** | 0.153 | 0.152 | 0.152 |
| **Time taken (secs)** | 110.9113214 | 121.1529546 | 102.7454536 |

Table 2.5: Facebook: Node2Vec Clustering Performance Metrics

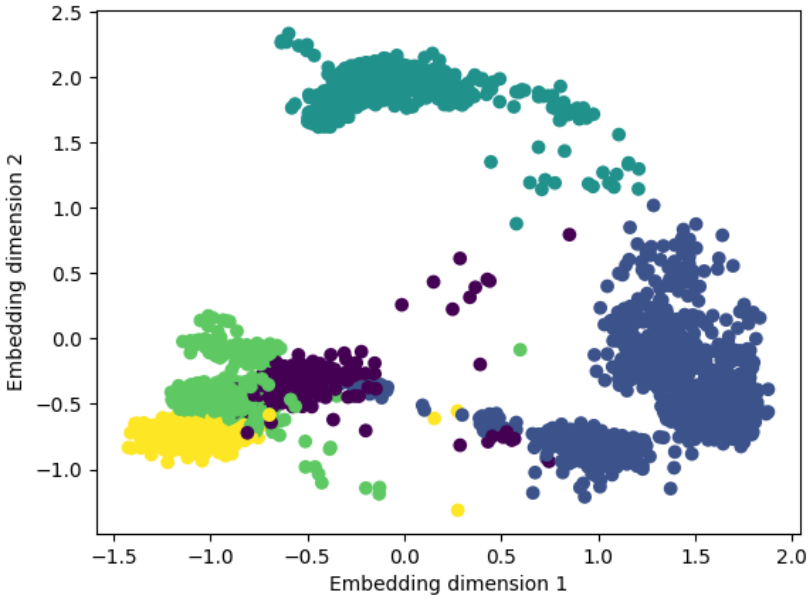


Figure 2.4: Facebook: Node2Vec Kmeans clustering

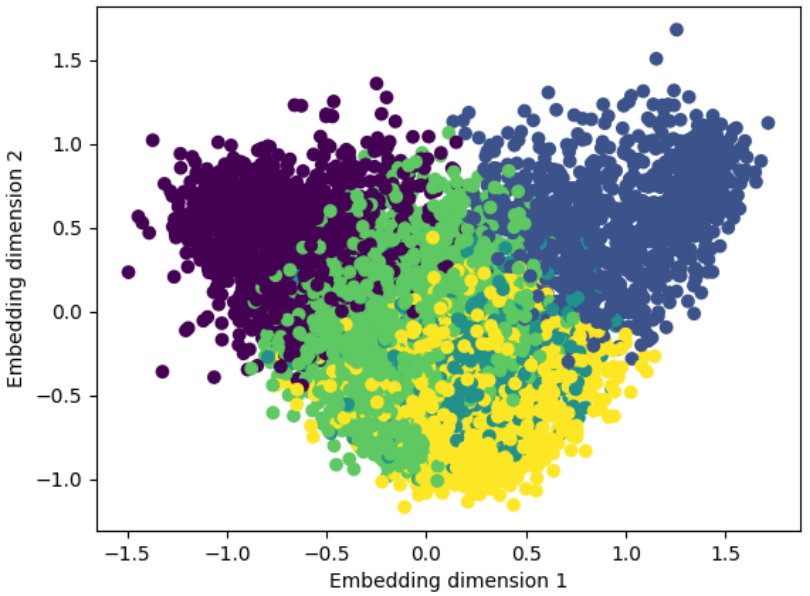


Figure 2.5: Twitch: Node2Vec Kmeans clustering

| **Analysis** | **Louvain** |
|---|---|
| Total nodes | 7126 |
| Total edges | 691 |
| Average degree in network | 2.564 |
| Density of network | 0.005 |
| Average clustering coefficient | 0.073 |
| **Communities** | |
| Number of communities | 60 |
| Average community size | 9 |
| Maximum Community Size | 52 |
| Average conductance | 0.127901639 |
| Maximum conductance | 1.182 |
| Average density | 0.707311475 |
| Maximum density | 1 |
| Average edge graph cuts | 3.803278689 |
| Maximum edge graph cuts | 39 |
| Modularity | 0.761 |
| Maximum Pairwise Cut | 8 |
| Average Pairwise Cut | 0.06 |
| Time taken (secs) | 4.87 |

Table 2.6: Twitch: Louvain Community Detection Metrics

| **Community** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **Conductance** | 0.778 | 0.863 | 0.893 | 0.668 | 0.798 |
| **Density** | 1.001 | 1.002 | 1.003 | 1.001 | 1.001 |
| **Nr of Nodes** | 1584 | 974 | 761 | 2369 | 1443 |

Table 2.7: Twitch: Community Metrics

2. **Flexibility:** More flexible as it can capture complex structural patterns and adapt to various network characteristics.

3. **Handling of Heterogeneous Graphs:** Suitable for heterogeneous graphs as it can learn embeddings for different types of nodes.

4. **Scalability:** Scalable to large networks due to its ability to generate node embeddings efficiently.

**Traditional Methods:**

1. **Algorithm Type:** CPM, Louvain, Girvan-Newman are modular/partitioning methods that identify communities based on different criteria.

2. **Interpretability:** More interpretable in terms of the criteria used for community detection, such as maximizing modularity or minimizing edge betweenness.

3. **Handling of Heterogeneous Graphs:** Typically designed for homogeneous graphs, although modifications can be made for some level of heterogeneity.

4. **Scalability:** May face scalability issues for very large networks, especially Girvan-Newman in its exact form.

**Trade-offs:**

- **Node2Vec:**
  - *Pros:* Flexible, applicable to various network types, captures complex patterns.
  - *Cons:* May lack interpretability, especially when understanding the learned embeddings.
- **Traditional Methods:**
  - *Pros:* Often more interpretable, well-established in community detection literature.
  - *Cons:* May struggle with large or heterogeneous networks like twitch, and results might heavily depend on specific algorithm parameters. Also computational expensive compared to embeddings.

**Use Cases:**

- **Node2Vec:**
  - Well-suited for scenarios where capturing node similarities and structural patterns is crucial.
  - Applicable when dealing with large-scale networks and heterogeneous graphs.
- **Traditional Methods:**
  - Preferred when clear interpretability and understanding of community structures are crucial.
  - Suitable for smaller to medium-sized networks where computational constraints are less of an issue.

**Conclusion:** The choice between Node2Vec and traditional methods depends on the characteristics of the network, the desired level of interpretability, and computational considerations. In practice, a combination of approaches or selecting based on the specific properties of the network may be beneficial.

# 3 Influence Maximization

Influence maximization involves identifying a small group of nodes, known as seed nodes, within a social network, with the goal of maximizing the spread of influence.

## 3.1 Susceptible-Infected-Recovered (SIR) Model

The Susceptible-Infected-Recovered (SIR) model is a mathematical framework used to simulate the spread of infectious diseases within a population. In this model, individuals are classified into three categories: susceptible (S), infected (I), and recovered (R). Initially, a portion of the population is susceptible to the disease, while a smaller subset is infected. As the infected individuals come into contact with susceptible ones, the disease spreads, transitioning them from susceptible to infected. Over time, infected individuals recover from the illness and become immune, moving them into the recovered category. The dynamics of the SIR model are governed by a set of differential equations that describe the rates of transmission, recovery, and removal from the infected population. The SIR model has been widely used in epidemiology to study the dynamics of infectious diseases and inform public health policies and interventions. **wilson1945law**

## 3.2 Linear Threshold Model

The Linear Threshold Model (LTM) is a fundamental concept in social network analysis that explores how influence spreads through interconnected nodes in a network. In this model, each node in the network has a random threshold value, typically drawn from a uniform distribution between 0 and 1. When a node receives input from its neighbors, it becomes active if the weighted sum of the inputs from its neighbors exceeds its threshold value. The weights on the edges represent the influence each neighbor has on the node, and they are constrained such that the sum of the weights from all neighbors of a node does not exceed 1. This model captures the idea that individuals may require varying levels of influence from their social connections before adopting a behavior or idea. The Linear Threshold Model provides valuable insights into how behaviors or information can propagate through social networks and has applications in various fields such as marketing, sociology, and epidemiology. **influence_maximization**

In the Linear Threshold Model, we have the following setup:

- A node $v$ has a random threshold $\theta_v \sim U[0, 1]$.
- A node $v$ is influenced by each neighbor $w$ according to a weight $b_{v,w}$, such that

$$\sum_{w \text{ neighbor of } v} b_{v,w} \leq 1.$$

- A node $v$ becomes active when at least $\theta_v$ fraction of its neighbors are active. That is

$$\sum_{w \text{ active neighbor of } v} b_{v,w} \geq \theta_v.$$

## 3.3 Independent Cascade Model

The Independent Cascade Model (ICM) is a popular way to study how ideas or behaviors spread in social networks. It works by simulating how nodes in a network influence each other. Each node has a chance to

influence its neighbors, and this influence spreads through the network based on probabilities assigned to each connection. The process continues until no new nodes are activated. The ICM helps us understand how information spreads in networks and has many real-world applications, like in marketing and public health. **influence_maximization**

In this model, we model the influences (activation) of nodes based on probabilities in a directed graph:

- Given a directed finite graph $G = (V, E)$.
- Given a node set $S$ starts with a new behavior (e.g., adopted new product and we say they are active).
- Each edge $(v, w)$ has a probability $p_{v,w}$.
- If node $v$ becomes active, it gets one chance to make $w$ active with probability $p_{v,w}$.
- Activation spreads through the network.

**Note:**

- Each edge fires only once.
- If $u$ and $v$ are both active and link to $w$, it does not matter which tries to activate $w$ first.

## 3.4 Methodology

In the analysis of influence maximization using traditional methods, three models were examined: the Susceptible-Infected-Recovered (SIR) Model, along with two diffusion models, namely the Independent Cascade (IC) Model and the Linear Threshold (LT) Model. To further investigate influence maximization, both greedy and the CELF model **leskovec2007cost** were employed.

In the case of methods utilizing machine learning (ML), a node2vec embedding was first applied to the network data. Subsequently, a random forest regression model was trained to predict the probability of influence for each node. Since the analyzed networks lack direct influence information, degree centrality was used as a reference metric. By training this model, we can predict influence probabilities, rank nodes accordingly, and select seed nodes. These selected seed nodes are then used in the Independent Cascade model to calculate their influence within the network. The metrics used were:

1. **Degree centrality:** Nodes are ranked based on their degree centrality, which measures the number of connections a node has in the network. The top-ranked nodes are selected as seed nodes.
2. **Pagerank:** Nodes are ranked according to their Pagerank scores, which assess the importance of a node based on the structure of the network and the importance of its neighbors. The nodes with the highest Pagerank scores are chosen as seed nodes.
3. **HITS:** The HITS algorithm computes two scores for each node, namely hub and authority scores. The hub score measures a node's connectivity to other high-authority nodes, while the authority score reflects a node's importance based on its connections to high-hub nodes. The nodes with the highest hub and authority scores are selected as seed nodes.
4. **k-core:** Nodes are assigned a core number representing the maximum subgraph in which each node has at least k connections. Nodes are ranked based on their core numbers, and the top-ranked nodes are chosen as seed nodes.

5. **Neighborhood coreness:** This metric calculates the sum of the degrees of a node's neighbors, representing the node's influence within its local neighborhood. Nodes are ranked based on their neighborhood coreness, and the top-ranked nodes are selected as seed nodes.

6. **Number of infected nodes:** The total number of nodes infected by the diffusion process is also considered as a metric to evaluate the effectiveness of the seed nodes selected by each method.

Each of these metrics provides insights into the network structure and node importance, helping identify potential influencers for maximizing influence in the network.

## 3.5 Result and experiments

### 3.5.1 Facebook

Firstly, in the analysis of influence using traditional methods, three models were explored: the Susceptible-Infected-Recovered (SIR) model, alongside two diffusion models, namely the Independent Cascade (IC) Model and the Linear Threshold Model (LT). Due to the computational complexity associated with executing these models, which falls under the realm of NP-problems, a decision was made to initially choose one of the models and subsequently apply the greedy and CELF strategies. To select the model, a random seed configuration is obtained.

In the analysis, we utilized different models for influence maximization. For the Susceptible-Infected-Recovered (SIR) model, we determined a gamma value of 0.1 and calculated a beta value of 0.21 using eigenvector centrality, which proved to be the most effective metric for node evaluation. This beta value represents the probability of infection for the SIR model. In contrast, the Independent Cascade model assigns a probability of infection to each edge, with a threshold set at 0.5. As for the Linear Threshold Model (LTM), we initially set a threshold of 0.1. In LTM, a node's activation depends on the active status of its neighbors and the percentage of neighbors who have made the same choice. While this threshold-based approach initially resulted in slow convergence, it eventually led to a rapid increase in activity levels as influence spread throughout the densely connected graph.

The analysis highlighted the Independent Cascade (IC) model's superiority in terms of convergence time, making it the standout performer among the models examined (see Figure 3.1). This model, using randomly selected seed nodes from the Facebook dataset, offers significant time savings for influence maximization analysis.

Due to the efficient convergence observed in the Independent Cascade (IC) model, it was chosen as the diffusion model to examine the influence of seed sets using various centrality measures (see Figure 3.2). Top 10 nodes were selected as seed sets based on each centrality measure. While all metrics displayed distinct performance trends, pagerank emerged as the standout performer, yielding the highest number of infected nodes, reaching a maximum of 3800. Following pagerank, degree centrality and neighborhood coreness (nb) metrics also demonstrated notable effectiveness, with -core benefiting as well.

Continuing the analysis, we employed both a greedy and CELF algorithm to compare their complexity and execution time (see Figure 3.3). Notably, the greedy algorithm required a total of 5 hours for execution, with its time increasing linearly. In contrast, the CELF algorithm was notably more efficient, completing the task in just 2 hours to identify the seed group.

Ultimately, both algorithms converged to the same seed group and thus achieved the same number of infected nodes by the 5th iteration (see Figure 3.4). However, the CELF algorithm demonstrated notable efficiency, being 60% faster compared to the greedy algorithm.

Furthermore, the metrics were scrutinized alongside the CELF results (see Figure 3.5). It was observed that, for this seed group, the metrics yielded similar outcomes, albeit varying across different seed sizes.

| Metric | Value |
|---|---|
| Silhouette | 0.028 |
| Inertia | 47634.023 |
| Davies-Bouldin (db) | 4.717 |
| Calinski-Harabasz (ch) | 167.379 |
| Number of communities | 5 |
| Average community size | 1426 |
| Maximum Community Size | 2369 |
| Average density | 1.0016 |
| Maximum density | 1.003 |
| Average conductance | 0.8 |
| Maximum conductance | 0.893 |
| Modularity | 0.082 |
| Time taken (secs) | 332.77 |

Table 2.8: Twitch: Kmeans Clustering Metrics

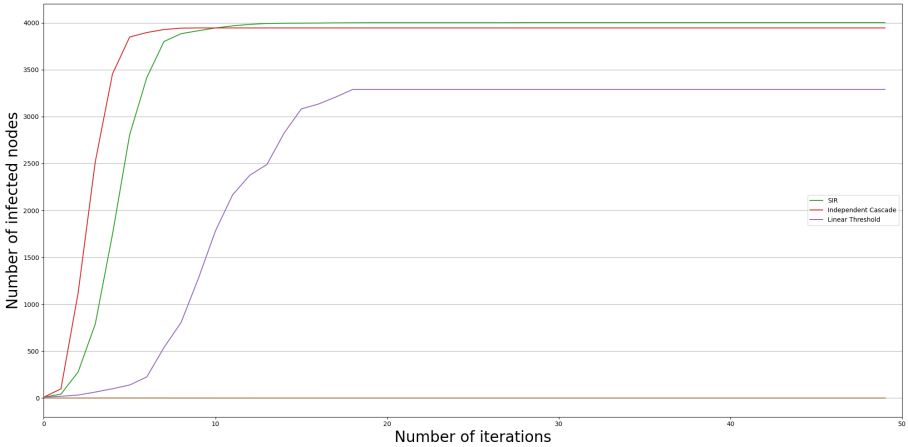


Figure 3.1: Comparison of Convergence Time

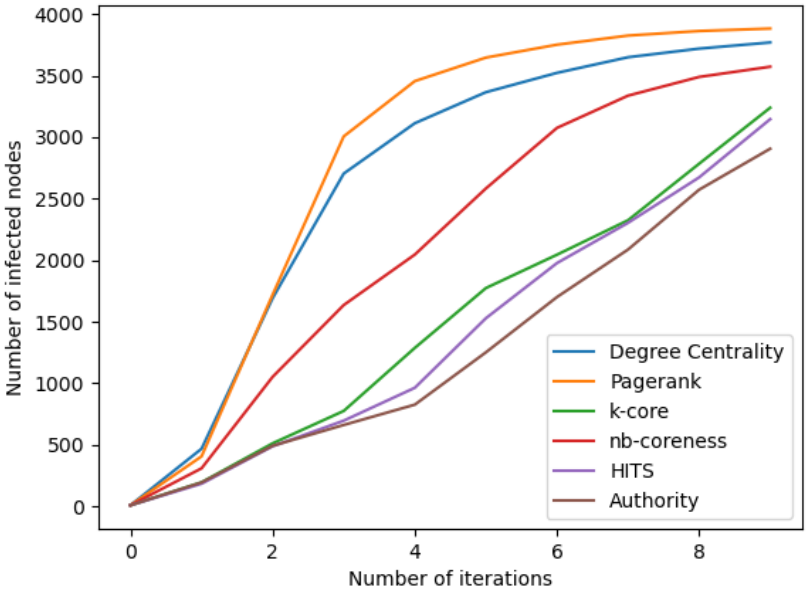


Figure 3.2: Comparison of Seed Set Influence

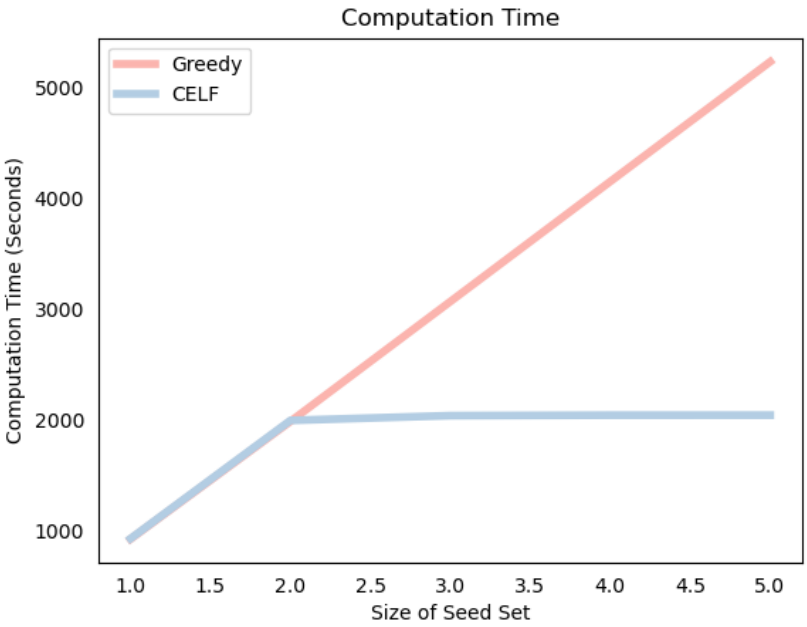


Figure 3.3: Comparison of Computational Time

Notably, the degree metric initially performed poorly for seed groups of size 5, but its behavior aligned with other metrics thereafter. Among some noteworthy observations, both Authority and CELF metrics exhibited higher maximums in the number of infected nodes across various seed sizes.

The ML approach, utilizing node2vec embedding and a random forest regressor for seed set selection in influence maximization, achieved outstanding results (see Table 3.1). The regression model demonstrated exceptional performance with a validation $R^2$ score close to 1, indicating a high degree of fit between predicted and actual values. The low mean squared error (MSE) and root mean squared error (RMSE) further underscored the accuracy and precision of the model in predicting node influence probabilities. Overall, these results demonstrate the effectiveness of the ML approach in accurately selecting seed nodes for influence maximization.

In the ML-based approach, promising results were achieved within an execution time of 12.35 minutes, leading to a total of 3955 infected nodes, which accounts for 97% of the total number of users in the network (see Figure 3.6). These outcomes are considered impressive, especially considering that node selection is based on embedding and the central degree metric.

The table presents the results of three different approaches for influence maximization on the Facebook network (see Table 3.2). In the traditional approach using the greedy algorithm, 3929 nodes were infected after 5 hours of execution time. Similarly, the CELF algorithm achieved the same number of infected nodes but in only 2 hours, demonstrating a significant improvement in efficiency. On the other hand, the ML approach, which utilized node2vec embedding and a random forest regressor, resulted in 3955 infected nodes within just 12 minutes. This approach outperformed both traditional methods in terms of execution time and achieved a slightly higher number of infected nodes. These results indicate that while the traditional methods are effective, the ML approach offers a faster and potentially more accurate solution for influence maximization in the Facebook network. The combination of node2vec embedding and a random forest regressor proved to be particularly efficient in selecting seed nodes, resulting in a higher number of infected nodes in a significantly shorter time frame.

### 3.5.2 Twitch

The experiments conducted on Twitch aimed to assess the efficiency of traditional influence maximization models, similar to the Facebook analysis. Three models were scrutinized: the Susceptible-Infected-Recovered (SIR) model, the Independent Cascade (IC) Model, and the Linear Threshold Model (LTM). Due to the computational complexity associated with these models, one was selected initially, followed by the application of greedy and CELF strategies. In the SIR model, a gamma value of 0.1 and a beta value of 0.21, derived using eigenvector centrality, proved effective for node evaluation. The Independent Cascade model assigns a probability of infection to each edge, with a threshold of 0.5, while the LTM uses a threshold of 0.1, activating nodes based on their neighbors' actions. Notably, LTM performed the worst, failing to reach the same number of infected nodes after 10 iterations compared to other models. The IC model offered significant time savings in influence maximization analysis (see Figure 3.7).

Given the efficient convergence observed in the Independent Cascade (IC) model, it was chosen as the diffusion model to examine the influence of seed sets using various centrality measures (see Figure 3.8). Top 10 nodes were selected as seed sets based on each centrality measure. While all metrics displayed distinct performance trends, k-nodes exhibited the poorest performance in this network. However, it gradually recovered in the later iterations, ultimately achieving results similar to those of the other metrics.

Continuing the analysis, we utilized both greedy and CELF algorithms to assess their computational complexity and execution time (see Figure 3.9). We observed similar results to those obtained with the Facebook dataset regarding computational time. Specifically, the greedy algorithm took a total of 5

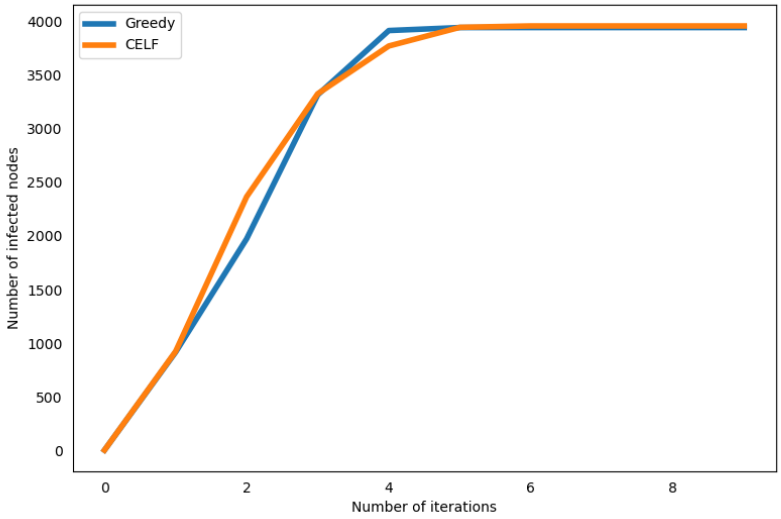


Figure 3.4: Comparison of Algorithm Efficiency

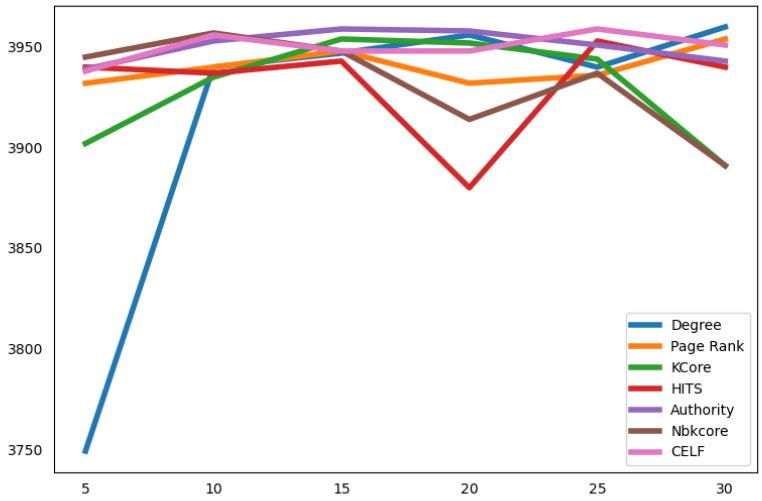


Figure 3.5: Comparison of Seed Set Metrics

| Metric | Value |
|---|---|
| Validation $R^2$ Score | 0.9994783461721873 |
| Validation MAE | $1.9591112157287075 \times 10^{-5}$ |
| Validation MSE | $8.5670530627927 \times 10^{-8}$ |
| Validation RMSE | 0.00029269528630971665 |

Table 3.1: Metrics for the ML Model Execution

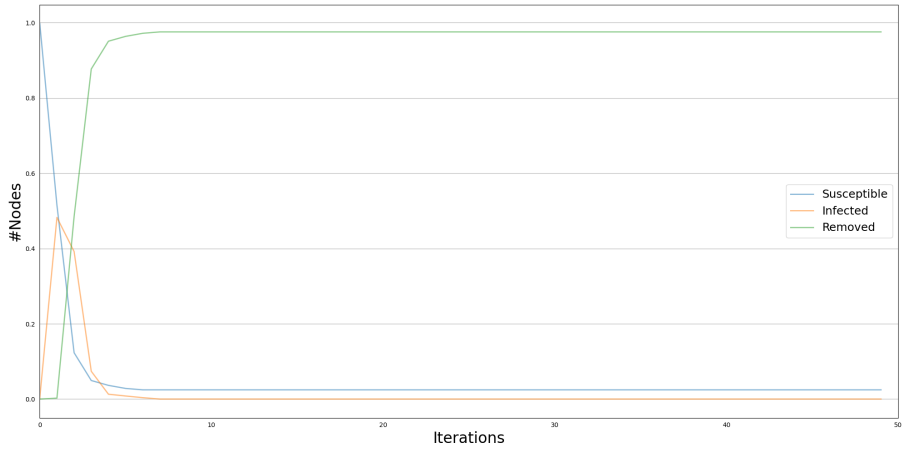


Figure 3.6: Comparison of ML Model Results

| Model Type | Infected Nodes | Execution Time | Seed Set |
|---|---|---|---|
| Traditional Approach (Greedy) | 3929 | 5 hours | [0, 107, 3980, 686, 348] |
| Traditional Approach (CELF) | 3929 | 2 hours | [0, 107, 3980, 686, 348] |
| ML Approach | 3955 | 12 minutes | [107, 1684, 1912, 3437, 0] |

Table 3.2: Comparison of Model Results

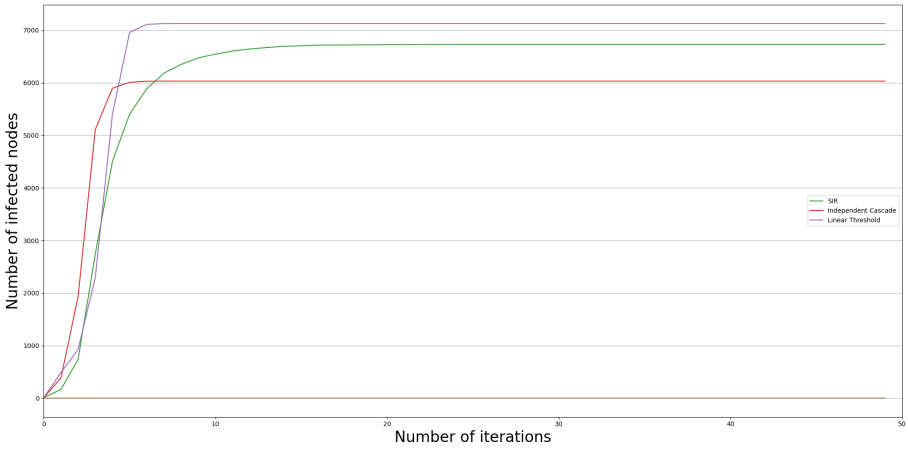


Figure 3.7: Comparison of Convergence Time

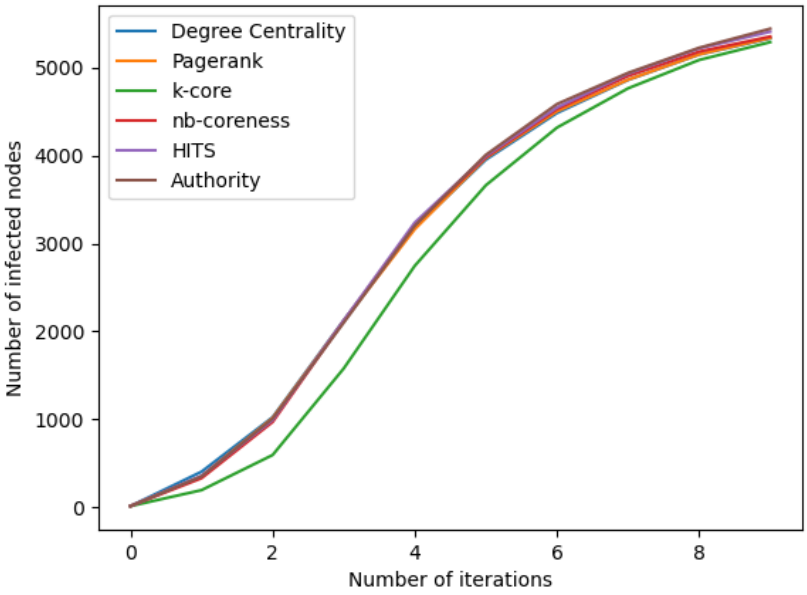


Figure 3.8: Comparison of Seed Set Influence

hours to execute, with its execution time increasing linearly. In contrast, the CELF algorithm proved significantly more efficient, completing the task in just 2 hours to identify the seed group.

Ultimately, both algorithms converged to the same seed group and thus achieved the same number of infected nodes by the 5th iteration (see Figure 3.10). However, the CELF algorithm demonstrated notable efficiency, being 60% faster compared to the greedy algorithm.

The ML approach, which utilized node2vec embedding and a random forest regressor for seed set selection in influence maximization, achieved outstanding results (see Table 3.1). The regression model demonstrated exceptional performance with a validation $R^2$ score close to 1, indicating a high degree of fit between predicted and actual values. The low mean squared error (MSE) and root mean squared error (RMSE) further underscored the accuracy and precision of the model in predicting node influence probabilities. Overall, these results demonstrate the effectiveness of the ML approach in accurately selecting seed nodes for influence maximization.

In the ML-based approach, promising results were achieved within an execution time of 12.35 minutes, leading to a total of 6098 infected nodes, which accounts for 85% of the total number of users in the network (see Figure 3.11). These outcomes are considered impressive, especially considering that node selection is based on embedding and the central degree metric.

The results from the influence maximization analysis on the Twitch network provide valuable insights into the performance of different approaches. As shown in Table 3.4, all three models achieved a high number of infected nodes, indicating their effectiveness in spreading influence through the network. The traditional approach using the greedy algorithm and the CELF algorithm resulted in a similar number of infected nodes, with the greedy algorithm requiring more execution time compared to CELF. On the other hand, the ML approach demonstrated superior efficiency, achieving slightly higher performance in terms of infected nodes within a significantly shorter execution time. These results suggest that the ML approach, leveraging node2vec embedding and a random forest regressor, offers a promising solution for influence maximization in the Twitch network, outperforming traditional methods in terms of both effectiveness and efficiency.

## 3.6 Conclusions

The experiments conducted on the Facebook and Twitch networks aimed to compare and contrast traditional influence maximization approaches with a machine learning (ML) approach.

### 3.6.1 Traditional Approaches

In the case of traditional approaches, namely the greedy algorithm and the Cost-Effective Lazy Forward (CELF) algorithm, both networks exhibited similar trends. The traditional approaches were effective in infecting a large number of nodes, with the CELF algorithm demonstrating superior efficiency in terms of execution time compared to the greedy algorithm. This efficiency was particularly evident in the Twitch network, where CELF achieved comparable results to the greedy algorithm in significantly less time.

### 3.6.2 Machine Learning Approach

The ML approach, utilizing node2vec embedding and a random forest regressor, showcased remarkable performance in both networks. In the Facebook network, the ML approach outperformed traditional methods in terms of execution time and achieved a slightly higher number of infected nodes. Similarly,

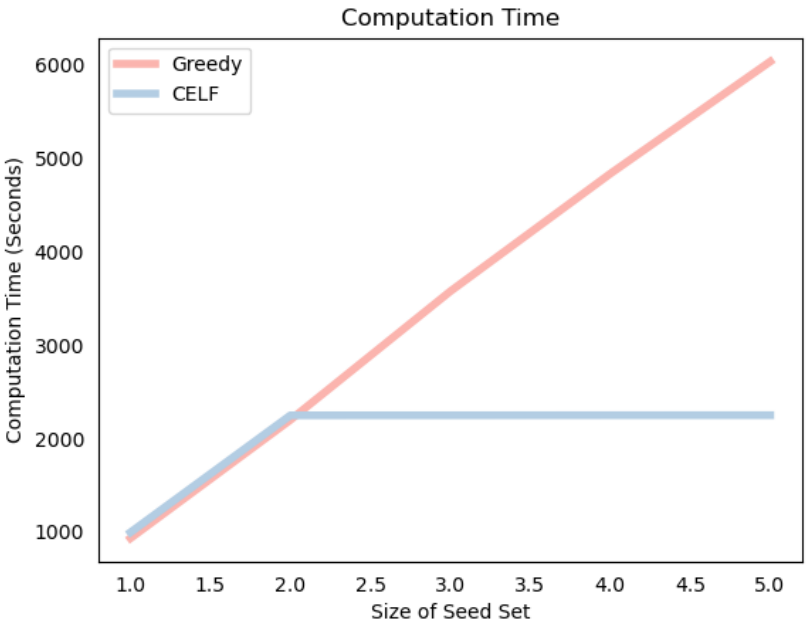


Figure 3.9: Comparison of Computational Time

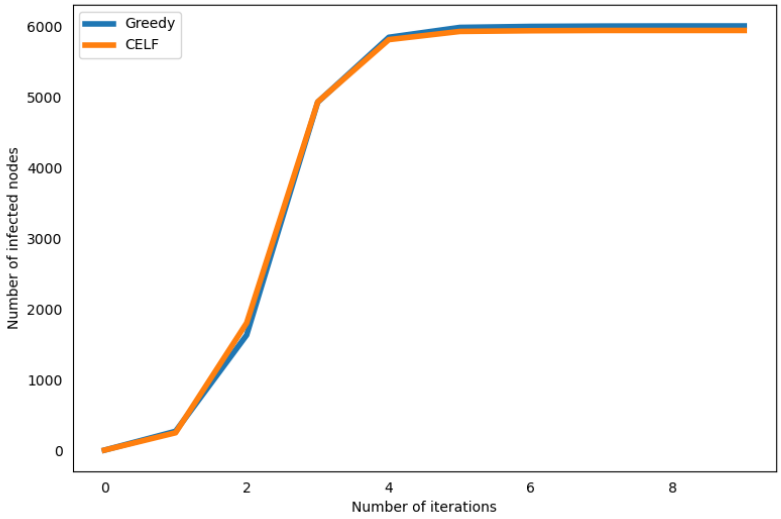


Figure 3.10: Comparison of Algorithm Efficiency

| Metric | Value |
|---|---|
| Validation $R^2$ Score | 0.9505082259923263 |
| Validation MAE | $3.207401392682365 \times 10^{-5}$ |
| Validation MSE | $6.802642834541335 \times 10^{-7}$ |
| Validation RMSE | 0.0008247813549384671 |

Table 3.3: Metrics for the ML model execution

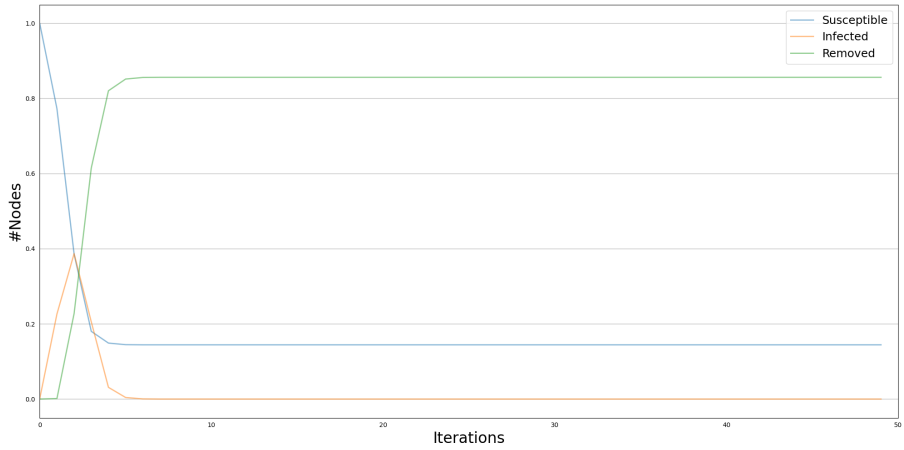


Figure 3.11: Comparison of ML Model Results

| Model Type | Infected Nodes | Execution Time | Seed Set |
|---|---|---|---|
| Traditional Approach (Greedy) | 6010 | 5 hours | [166, 24, 150, 80, 6323] |
| Traditional Approach (CELF) | 6008 | 2 hours | [166, 24, 150, 80, 10] |
| ML Approach | 6011 | 12 minutes | [1773, 4949, 3401, 6136, 166] |

Table 3.4: Comparison of Model Results

in the Twitch network, the ML approach achieved higher performance in terms of infected nodes within a significantly shorter execution time compared to traditional methods.

### 3.6.3 Insights and Implications

**Dataset Characteristics:** The characteristics of the datasets played a crucial role in determining the effectiveness of different approaches. For instance, the Facebook dataset, with its dense and well-connected structure, favored the ML approach due to its ability to capture complex relationships. On the other hand, the Twitch network, characterized by sparse connections, highlighted the efficiency of traditional methods, particularly the CELF algorithm.

**Metric Selection:** The choice of metrics for evaluating node importance and seed selection significantly influenced the performance of the approaches. Metrics such as degree centrality, Pagerank, and neighborhood coreness proved to be effective in both networks, while others, such as k-core, exhibited varying performance depending on the dataset.

**Execution Time vs. Effectiveness:** The trade-off between execution time and effectiveness was evident across all approaches. While traditional methods may achieve comparable results, they often require longer execution times compared to ML approaches. However, the ML approach may require additional preprocessing and parameter tuning. Moreover, the ML approach employed makes a stringent assumption that the probability of influence is solely determined by the embedding and the degree centrality measure of the nodes. This assumption could potentially constrain its applicability in diverse types of social networks.

**Potential Applications:** The results have implications for various applications, including social network analysis, marketing strategies, and recommendation systems. The ML approach offers a more scalable and efficient solution for influence maximization tasks, particularly in large-scale networks.