

# UNIVERSITÀ DEGLI STUDI DI MILANO

FACULTY OF POLITICAL, ECONOMIC AND SOCIAL SCIENCES

MASTER'S DEGREE COURSE IN DATA SCIENCE AND ECONOMICS


# Graph-based Recommendation System Leveraging Community and Influence Features


Academic Year: 2022/2023
Thesis of: Amr Rashad
Advisor: Prof. Sabrina Tiziana Gaito
Co-Advisor: Ric. Matteo Zignani

# Abstract

The aim of this thesis is to explore the development and implementation of a novel recommendation system using graph-based methodologies within the dynamic landscape of online communities. The research focuses on leveraging community detection algorithms and influence features to enhance the accuracy and relevance of recommendations within diverse online environments. The study first gathers data from popular subreddits, extracting posts, user interactions and population. The project further explores the application of two established methods for community detection: the Louvain and Greedy algorithms. It analyzes their effectiveness in identifying significant communities within the collected data. Additionally, the study employs the Implicit Alternating Least Squares (ALS) technique to provide personalized recommendations within distinct user communities. Results showcase the enhancement of recommendation accuracy by incorporating community structures and user influence. This research contributes to the field of recommendation systems by exploring the integration of community-centric features in a graph-based framework, offering valuable insights into the improvement of recommendation accuracy and relevance within online communities.

# Table of Contents

# 1. Introduction:

Online platforms nowadays contain a variety of content, spanning from social media interactions to news articles and product recommendations. As these digital platforms continue to evolve, it becomes increasingly challenging to deliver personalized, relevant content to individual users. In response to this challenge, recommendation systems have emerged as a crucial mechanism, aiming to use the large amount of information available and deliver tailored suggestions that match the preferences and behaviors of users. However, as online platforms continue to evolve, the available content undergoes exponential growth, creating unique challenge in delivering personalized, relevant material to individual users. This continuous evolution resulted in the need for more advanced and precise recommendation systems, aiming to predict user preferences accurately. The main goal of implementing such systems is to have a more engaging and personalized user experience.

Recommendation systems operate through distinct phases, working thoroughly to provide users with personalized content that aligns with their preferences. Such systems involve a set of phases including data collection and processing, pattern identification and modelling, and the third one recommendation generation. In the first phase, data is gathered and collected, and is then processed, often employing several algorithms to convert raw data into meaningful patterns and user-item interactions. In the second phase, processed data is analyzed with the aim of identifying patterns and relationships, techniques such as collaborative filtering (CF), content-based filtering and more advanced methods are employed with the aim of modelling user preferences and item characteristics. In the third and final phase, the system generates recommendations based on the patterns and models identified. These recommendations are tailored to the individual user by predicting the likelihood of a user's preference for an item. It's worth noting that even though recommendation systems share a common structure, variations exist across different applications.

The primary aim of this project is to exhibit how integrating a recommendation system within a community and utilizing influential features can notably improve its accuracy, especially through the use of implicit ALS methods. To accomplish this, a user graph will be created, forming a network, with the identification of communities and influencers, and the calculation of implicit scores. This comprehensive understanding will be pivotal in demonstrating the practical application of recommendation systems within diverse online communities.

## 2. Literature review:

The advancement of any field is contingent upon understanding the wealth of knowledge that has already been accumulated. This section serves as the starting point for this research. To solve specific problems effectively, it's important to learn from past methods and approaches used by others facing similar challenges. This not only helps in building on past findings but also offers insights into potential methodologies. This section explores various research papers to establish a strong base for the thesis.

Deshmukh & Ingle [1], discuss the different types of recommendation systems, explaining their respective contexts for optimal utilization and providing comprehensive insights into their operational mechanisms. For instance, it discusses CF recommender systems, highlighting their popularity and widespread implementation as a prevalent technique in recommendation systems. The paper also examines how integrating community detection techniques with recommender systems leads to more tailored recommendations specifically tailored to individual users within the same community.

The study conducted by Shokrzadeh et al. [2] investigates the formation of communities built upon similar tags and examines the generation of recommendations based on these tag-based communities. The type of recommender system implemented in the study is a CF recommendation system.

The work conducted by Gasparetti, Micarelli, and Sansonetti [3] discusses community-based recommendation systems, more specifically it explores various methods and considerations involved in detecting communities within social networks, considering different user behaviors, interactions, and the nature of the connections for recommendation systems and network analysis. An important aspect addressed in this work involves addressing the challenges of the cold-start problem and data sparsity by utilizing community detection algorithms and model-based approaches, such as collaborative filtering (CF). Additionally, the paper delves into the application of dimensionality reduction, especially in scenarios with a high volume of users and items. The presence of several missing values in such cases can hinder traditional CF-based methods, leading to potential limitations or failures in the recommendation process.

These research papers provide positive insights into the fusion of community detection algorithms and recommendation systems, showcasing how this integration could result in more tailored and precise recommendations for individual users. Therefore, this thesis project aims to build on these previous works and explore how including user influence could further enhance the precision of the recommendations.

# 3. Methodology:

## 3.1. Dataset Description and Characteristics:

The dataset in which this thesis project is going to be based on is a reddit dataset. The dataset was collected via Reddit's API and was stored in a GitHub repository in order to make a straightforward process for data access and data loading. The dataset includes information from the top 100 subreddits. Each subreddit's content is saved in a separate file. In each file, there will be details about the top 1000 posts from the past year. Each post includes these details:

- Subreddit Name
- Subreddit Population
- Post Title
- User ID who made the post
- List of user IDs who interacted with the post
- Number of upvotes
- Date
- Time

## 3.2. Data Preprocessing:

Some preparation was needed to create a structured table for making the user interaction graph later. First, a new 'Users List' column is made by putting together the 'User ID' and 'Users Who Commented' details into a single list. If the user who posted also commented, only one unique ID goes into the list. Some columns like 'Downvotes', 'User ID', and 'Users Who Commented' were removed.

Due to the dataset's large size, especially the number of user IDs, only active user IDs, those listed at least 40 times in a subreddit, are kept in the 'Users List' column. Different trials were done to find the best threshold for a manageable dataset that can be processed on a single machine. The starting threshold was 100 and was gradually lowered by 10. The optimal threshold that resulted in a good dataset and graph size was found to be 40.

## 3.3. Graph Creation:

In order to perform community detection, a network graph is necessary. Various methods exist for creating a network graph, but in this case, the network structure adopted links users if they have participated in a common subreddit. For instance, if user A and user B interacted in the subreddit X, then both users are going to be linked together. Hence, the nodes of such graph are going to be the user IDs.

It’s worth noting that the network graph created is undirected graph and the library used to create such graph is called NetworkX, a well-known Python library, designed for creating, manipulating, and analyzing complex network structures and graphs.

The graph, with a network order of 8,587 nodes and a network size of 1,375,357 edges, indicates the extensive scale of connections within the network.; however, the graph is composed of 5 connected components. A connected component in a network is a part of the network where every member is connected to every other member through at least one path. It's important to note that within the component, there are no paths connecting any node in the component to any node outside of the component. Additionally, there’s no possibility of adding any other node to the component, as by doing so the connected nature of the original component won’t be maintained. Such commitment to connectedness and exclusivity characterizes the maximality of connected components. Typically, there is one large strongly connected component and a selection of small ones, which is the case in this graph, as the first component is considered to be the largest one with 8,513 nodes and 1,373,559 edges. Consequently, the other connected components were discarded and only the first one was retained. Figure 1 displays an example of a network graph with multiple connected components, while Figure 2 showcases the network graph of this project.

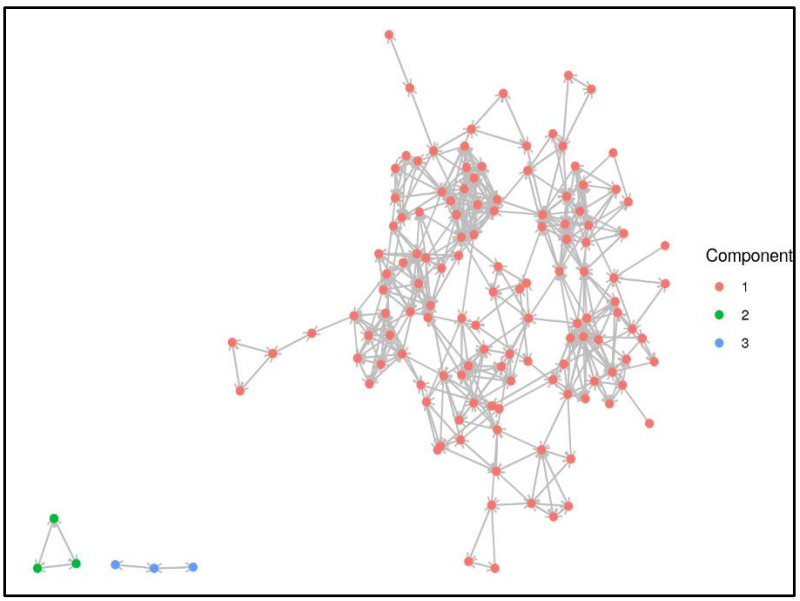


*Figure 1: Network graph containing 3 connected components*

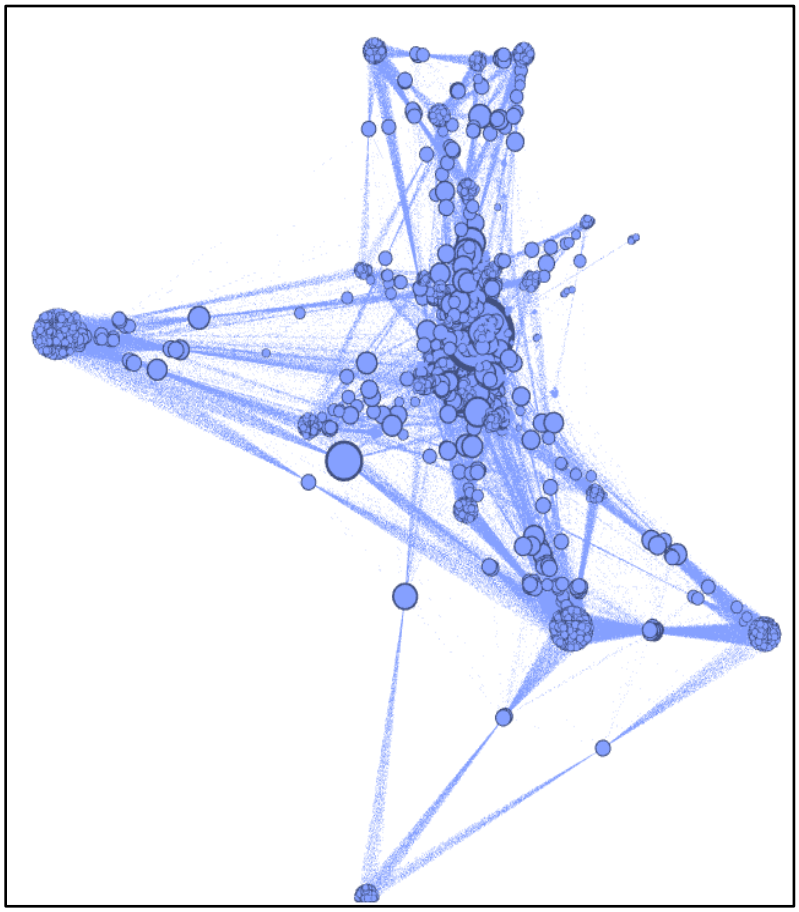

*Figure 2: Reddit Users Network Graph*

# 4. Network Analysis:

## 4.1. Graph Detailed Analysis:

The aim of this section is to present the study's comprehensive analysis of the graph formed by user interactions within the subreddit community. Various statistical measures have been conducted such as node degree distribution, network density, transitivity, average shortest path, and several other measures. The analysis of these parameters serves as a fundamental step in order to understand the characteristics of the network.

The node degree signifies the number of links connected to a specific node. In this graph, several statistical measures define the distribution of these degrees. The average degree is around 323, with a median value of 283 and a standard deviation of approximately 232. This variation in degrees indicates a range from a minimum of 1 to a maximum of 2024, highlighting the extensive connections present within the network.

The next measure that was explored is network density, as the aim is to understand if the network has few or many links given the number of nodes. Network density $\Delta$ is the ratio of the number of links L to the number of possible links in a network with N nodes.

$$\Delta = \frac{L}{N(N-1)/2} = \frac{2L}{N(N-1)}$$

The network density of the study's graph is around 0.038, which suggests that approximately 3.8% of all possible connections between the nodes in the graph are actual connections. In other words, the network is not heavily interconnected, indicating a relatively sparse relationship between its nodes. This is an expected value as typically social networks often exhibit sparse connections.

Furthermore, the analysis extends to assess the transitivity of the network. Transitivity measures the extent of indirect connections between nodes in the network, for example, if A knows B and B knows C, then there is a higher probability that A knows C. There are two types of transitivity. Perfect transitivity only occurs when the graph is fully connected, but achieving this is a theoretical concept rather than a practical reality within social networks due to their inherent complexity. Partial transitivity, however, refers to the presence of transitive

relationships within subsets or parts of a network while not universally spanning the entire network.

$$Transitivity = \frac{3 * number\ of\ triangles}{number\ of\ connected\ triples}$$

In this project's graph, the network transitivity is around 0.89, indicating a very high degree of transitivity as it's close to 1. In other words, a high transitivity score in this context suggests a high probability that users who share a connection with a common friend also tend to be connected to each other.

Subsequently, the average shortest path of the network is analyzed, which is a measure used in network analysis that gives the average length of the shortest path between all pairs of nodes in a network. It shows how, on average, nodes are connected in terms of the fewest number of edges between them. The shortest path length between two nodes is represented by d (i, j), where i and j are two nodes.

$$Average\ Shortest\ Path = \frac{\sum_{i \neq j} d(i,j)}{N(N-1)}$$

The average shortest path in this study's network is around 2.59, meaning that it would take around 2.59 steps to go from node A to node B. In essence, any two nodes are connected by a path of roughly 2.59 edges.

Another analysis was conducted is plotting the Empirical Complementary Cumulative Distribution Function (ECCDF) plot for the degrees of the nodes in the network. This plot is used to visualize the distribution of degrees and how many nodes have a degree greater than or equal to a given value. ECCDF is a statistical measure often utilized to analyze the distribution of data within a dataset. Plotting the ECCDF in a log-log scale provides insights into the tail behavior of the distribution, emphasizing the frequencies of larger values within the dataset. Figure 3 demonstrates the distribution of degrees among these reddit users.

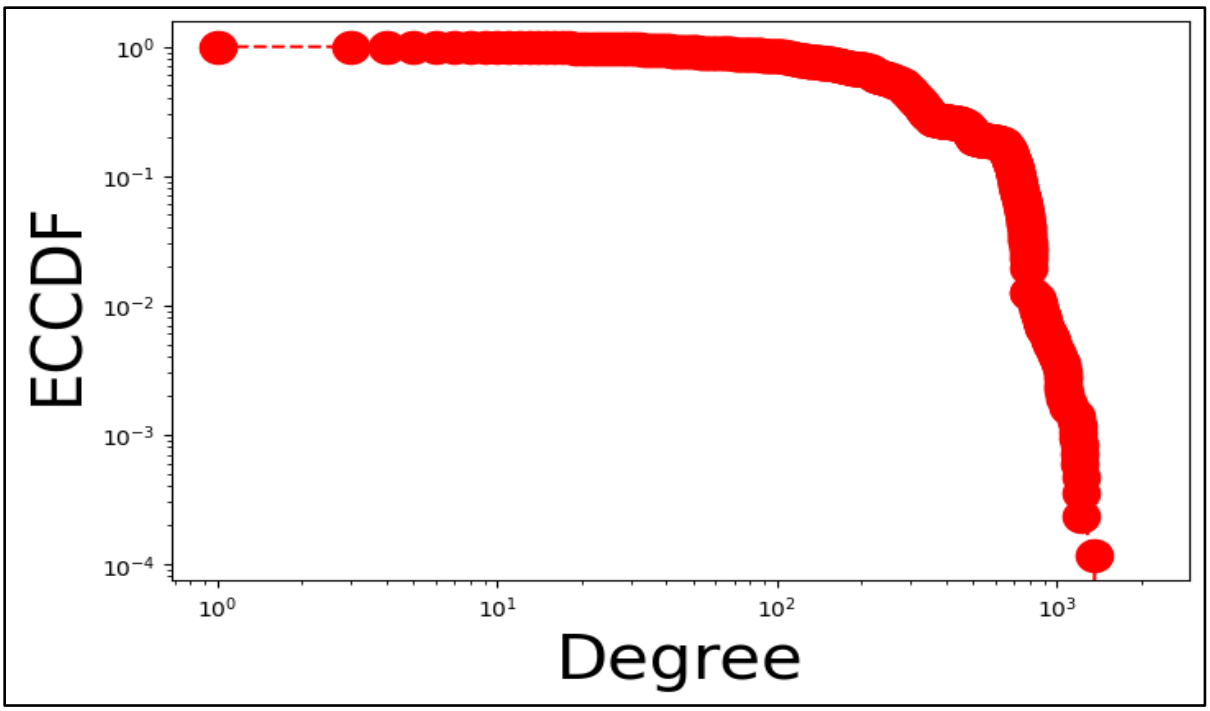


*Figure 3: ECCDF Plot*

The log-log scale ECCDF plot (Figure 3) implies that the more the number of degree increases, the less is the number of nodes who have such amount of links, meaning that it highlights the distribution's tail behavior, emphasizing the decreasing frequency of nodes possessing higher values.

Another key measure that was conducted in this study was calculating the 95$^{th}$ percentile, which means identifying the minimum number of degree that the 5% percent of the nodes have, such nodes are called high degree nodes or hubs, which indicates that they have a significant number of connections to other nodes within the graph. Concerning this graph, the 95$^{th}$ percentile value is 766, which implies that 5% of the nodes in the graph have a degree equal to or greater than 766.

The final measure that was computed for this section was the average clustering coefficient, which measures the degree to which nodes in a graph tend to cluster or form connected groups with each other. It is the mean value of the clustering coefficient over all nodes in the graph. Regarding this graph, the average clustering coefficient is around 0.92, indicating that the nodes in the graph form tightly interconnected groups or clusters, with a large number of closed triangles or cliques within the network.

## 4.2.Centrality Measures:

Centrality measures within a network play a crucial role in identifying the relative importance of nodes within the structure. Three fundamental centrality measures often utilized are degree centrality, betweenness centrality and eigenvector centrality. These three measures have been in fact analyzed in this study.

Degree centrality in undirected network ranks nodes with more connections higher in terms of centrality. $d_i$ is the degree for vertex $v_i$. The following formula shows how the degree centrality is computed.

$$C_d(v_i) = d_i$$

Betweenness centrality is the measure of the extent to which a node lies on paths between other nodes, meaning how much a node falls between others. Nodes with a high betweenness centrality have control over the information flowing in the network.

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

Betweenness centrality is defined as the formula given above, where $\sigma_{st}(v_i)$ represents the number of those shortest paths that pass through a specific node $v_i$, and $\sigma_{st}$ represents the number of shortest paths between nodes s and t, also known as information pathways. It's worth noting that the path from s to t is different from the path from t to s even in undirected network.

Eigenvector centrality in undirected network can defined as an extension of the degree centrality, the key point is that it tries to generalize degree centrality by incorporating the importance of neighbors. For instance, not all friends are equivalent, therefore having more friends dos not necessarily guarantee that user A is more important, however having more important friends provides a stronger signal. The standard formula of eigenvector is given below, where $C_e$ is the eigenvector centrality scores as a vector, A is the adjacency matrix and $\lambda$ is the eigenvalue.

$$\lambda C_e = A C_e$$

Figures 4, 5, and 6 display the node rankings based on their degree centrality, betweenness centrality, and eigenvector centrality, respectively. In these figures, the shade of green represents the centrality score of the nodes: the darker the green color, the higher the centrality score.

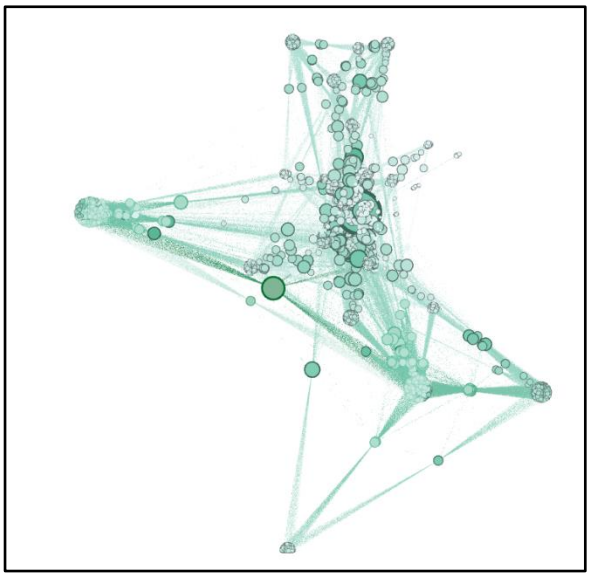

*Figure 4: Graph Ranked by Degree Centrality*

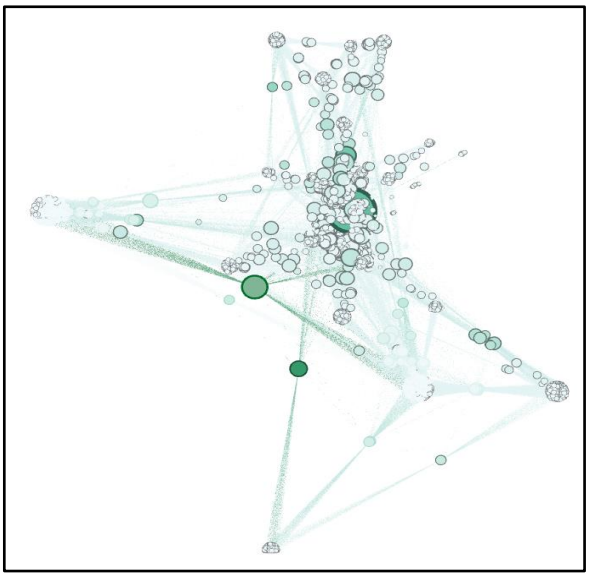

*Figure 5: Graph Ranked by Betweenness Centrality*

*Figure 6: Graph Ranked by Eigenvector centrality*

## 4.3. Community Detection Techniques and Evaluation:

Community detection refers to the identification of natural groupings or communities within complex networks. In other words, it's focusing on discovering implicit communities by the analysis of the network structure. Usually, a community represents a locally dense and interconnected subgraph within a network, with more links internally compared to those linking with the rest of the network. Two types of communities exist within network analysis. Disjoint communities which refer to non-overlapping partitions in a network, where nodes belong to one and only one community. Nodes within a disjoint community don't share membership with any other community, resulting in distinct, separate groups. On the other hand, overlapping communities involve nodes that can exist in more than one community simultaneously, this would mean that nodes are allowed to have multiple memberships, which would lead to blurring the lines between boundaries of these groups and creating a more interconnected network structure. Figure 7 demonstrates the formation of disjoint and overlapping communities within a network.

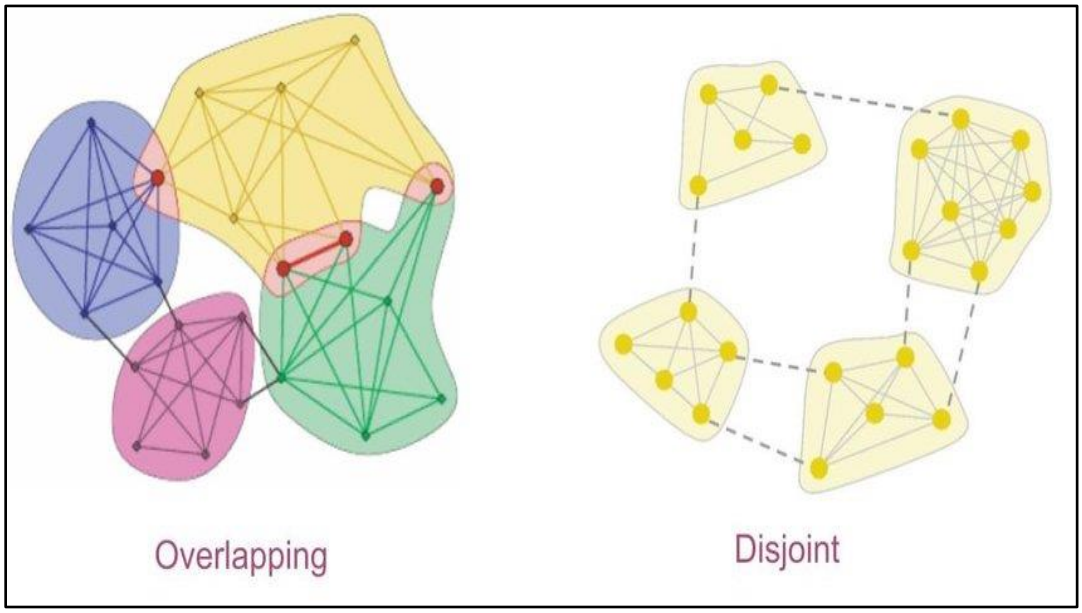


*Figure 7: Overlapping V Disjoint Communities*

Community detection aims to identify the best groupings within a network, and to assess these groups, their quality needs measurement. Modularity helps in this. It looks at the links within the identified groups and compares them to an expected random link distribution. A higher modularity score suggests a better community structure. It's a metric used to perform

community detection by attempting to find the maximum modularity value and corresponding network subdivision into communities. Modularity Q is calculated through the application of the following formula:

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - P_{ij})\,\delta(C_i, C_j)$$

Where:

- $A_{ij}$: Adjacency matrix of the network itself
- $P_{ij}$: The probability of having a link between node I and node j in a random network
- m is the total number of edges
- $C_i$ and $C_j$ are the communities to which nodes i and j belong.
- $\delta(C_i, C_j)$ is the delta function and equals 1 if $C_i = C_j$ and 0 otherwise.

Figure 8 exhibits the various types of partitions that can be obtained when calculating the modularity.

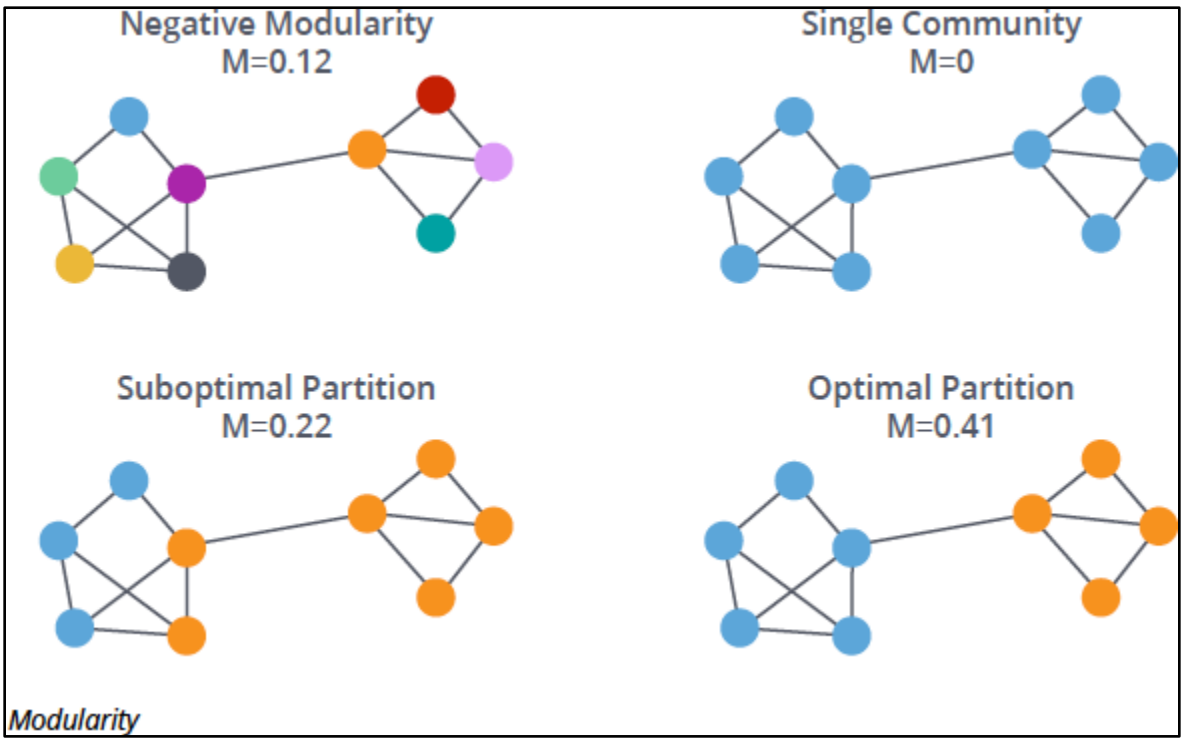


*Figure 8: Modularity-based Network Partitions*

There are various methods for performing modularity maximization, two of the most notable being the Greedy Algorithm and the Louvain Algorithm. The Greedy algorithm is a basic and commonly used technique, it optimizes modularity by successfully joining pairs of communities that lead to the maximum increase in modularity, in other words it would only join two nodes if such join would increase the partition's modularity. The algorithm can be broken down into 6 steps which are the following:

1. Initialization: Initially, each node is considered as an individual community.
2. Pairwise Examination: The algorithm inspects pairs of communities that are connected by at least one edge. It computes the change in modularity that would result from merging these community pairs.
3. Optimization: Among these pairs, the algorithm identifies the community pair where merging them would generate the largest increase in modularity.
4. Community Merge: Merging the identified community pair that maximizes modularity improvement.
5. Iterative Process: The steps of examining, optimizing, and merging are repeated iteratively until all nodes coalesce into a single community.
6. Selection of Optimal Partition: Throughout the process, the algorithm records the modularity score for each step. Finally, it selects the partition at which the modularity is maximal.

However, the greedy algorithm might have limitations similar to those associated with algorithms based on maximal modularity, such as resolution limit which is failing to detect smaller communities within larger ones, there is also the issue of scalability, where for larger networks, the computational demands can be high. Maximizing modularity involves considerable computation, and for substantial networks, it can become computationally expensive and time-consuming.

On the other hand, Louvain algorithm's aim is to find communities in large networks, this algorithm is scalable because it has computational complexity which is linear to the number of links O(L). It allows the identification of communities in networks with millions of nodes. The Louvain algorithm is divided into two phases that are repeated iteratively. These phases are the following:

- Phase 1: Modularity Maximization by Local Moving: In this phase, the algorithm initially assigns each node to its own community. It then iteratively goes through each node and checks if the modularity can be increased by moving the node to the community of one of its neighbors. This local optimization process is repeated until no further modularity increase is possible.
- Phase 2: Agglomerative Hierarchical Clustering: The second phase of the Louvain algorithm involves the construction of a new network, where the communities detected in the first phase are represented as nodes. The edges in the new network are constructed based on the connections between communities. This process is repeated iteratively, again optimizing the modularity to obtain a hierarchical clustering.
- Once the second phase is completed, phases 1 and 2 are repeated, calling their combination a “Pass”. The number of communities decreases with each pass. The passes are repeated until there are no more changes and maximum modularity is achieved.

The following table (Figure 9) exhibits the number of communities and modularity scores obtained when applying both algorithms on this study's graph. Figures 10 and 11 display the distribution size of the communities obtained after applying Louvain's algorithm.

| | **Number of Communities** | **Modularity Score** |
|---|---|---|
| *Greedy Algorithm* | 22 | 0.807 |
| *Louvain Algorithm* | 20 | 0.834 |

*Figure 9: Community Detection Results: Number of Communities and Modularity Scores.*

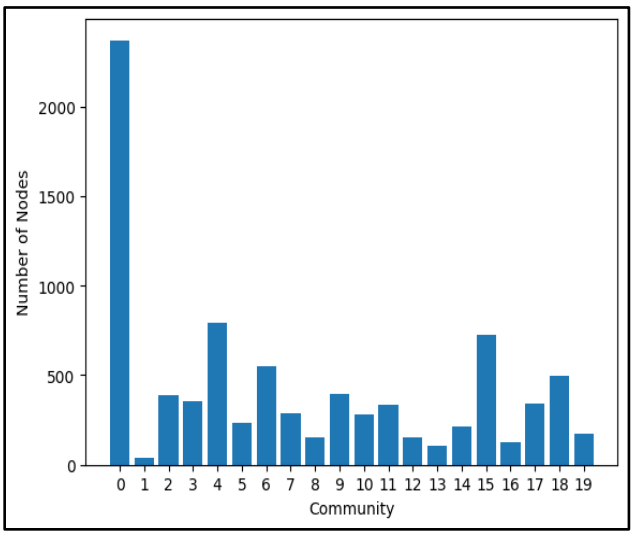


*Figure 10: Community Size Distribution Post Louvain's Algorithm Application*

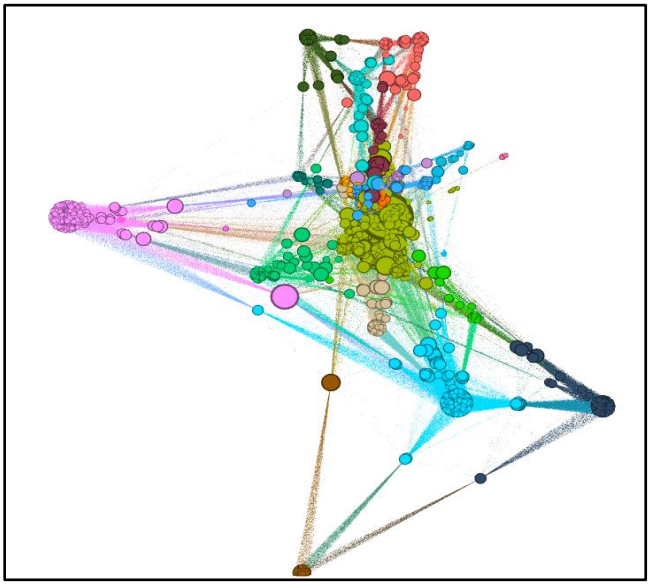

*Figure 11: Different Communities in The Network Graph*

## 5. PageRank and Influence Analysis:

This section focuses on assessing the importance of nodes in the network using the PageRank algorithm and further explores how this influence might extend to communities within the network. Each node is assigned a PageRank score, and this score can be used as an indication of the importance of the node within the entire network. The PageRank algorithm can be broken down into 6 steps which are the following:

1. Graph Representation: Nodes and their connections are initialized within a graph to establish network relationships.
2. Random Walking/Surfing: PageRank begins with a simulation where a random surfer navigates nodes by choosing random links.
3. Taxation/Teleport: A damping factor, usually around 0.85 (default value in the NetworkX python package), avoids dead-end nodes or endless loops by redistributing users.
4. PageRank Calculation: Node scores are iteratively updated based on the damping factor and the sum of neighboring node scores.
5. Iterative Process & Convergence: The power method updates the PageRank vector until the scores converge based on a defined threshold.
6. Importance Interpretation: Higher PageRank scores denote nodes with more influence due to connections to other influential nodes.

Regarding this project's graph, the PageRank scores have been computed, nodes have two PageRank scores, one represents the importance of the node within the network and the other represents the importance of the node within its own community. Figures 12 and 13 display the node rankings based on their importance within the network and within their own community respectively. In these figures, the shade of blue represents the PageRank score of the nodes: the darker the blue color, the higher the PageRank score.

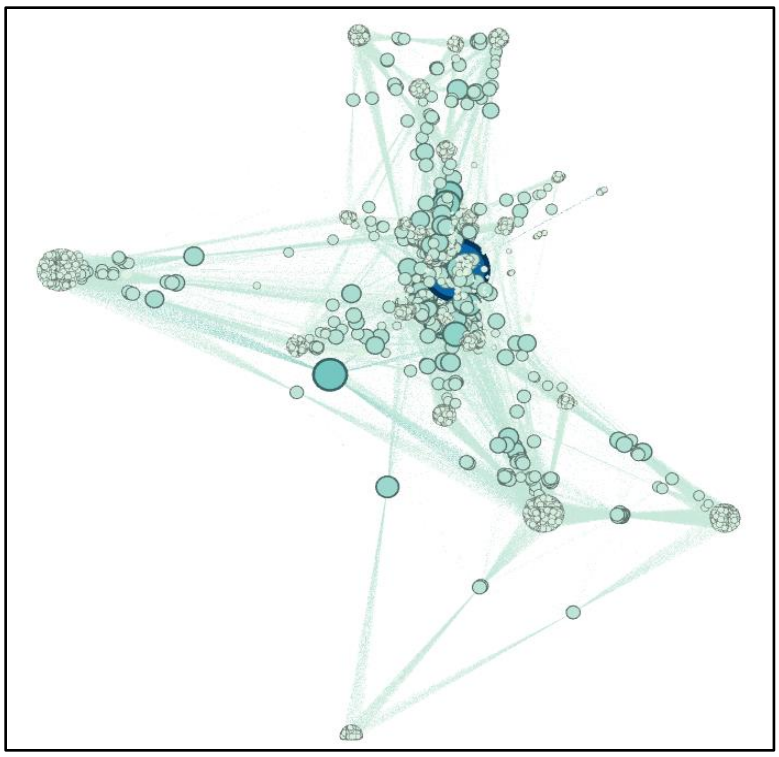

*Figure 12: Node Rankings by Network PageRank Scores*

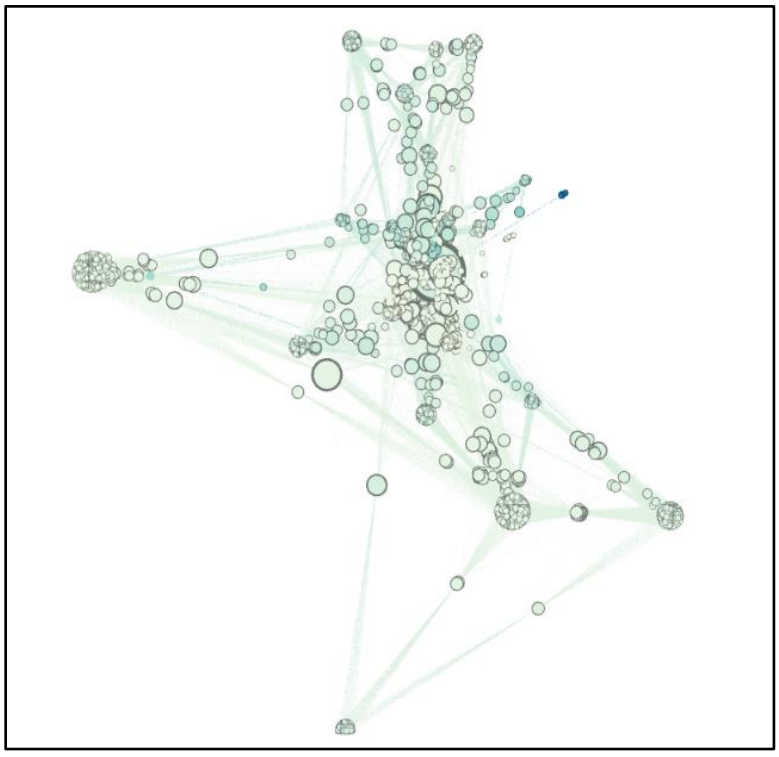

*Figure 13: Node Rankings by Community PageRank Scores*

# 6. Data Preparation and Integration:

The aim of this section is to explain how the graph created is translated into a dataframe that will be used for the recommendation system. It encompasses the processes involved in acquiring, refining, and combining various datasets to create a unified data structure suitable for the implementation of the CF recommendation system. After creating the final dataframe, a new column is added which represents the subreddit category. For instance, the category of subreddit 'dataisbeautiful' is Science. This column will be beneficial when calculating the implicit scores for the CF recommendation system. Figures 14 and 15 exhibit the distribution of row counts, where row represents a subreddit post record, in each community and the distribution of unique subreddits in each community, respectively.

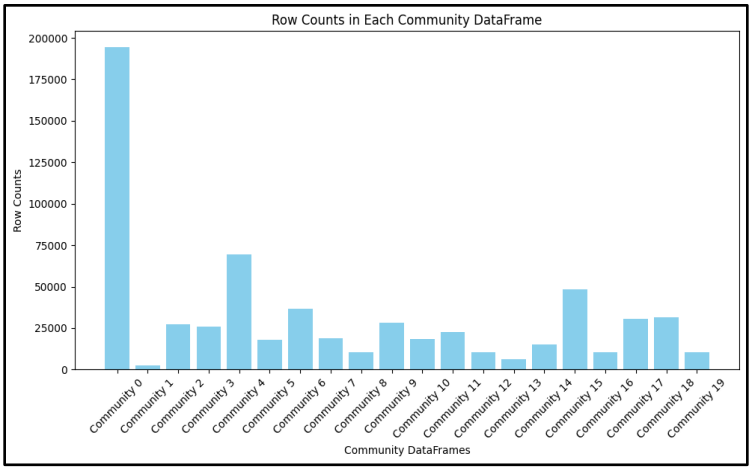


*Figure 14:Distribution of Row Counts per Community*

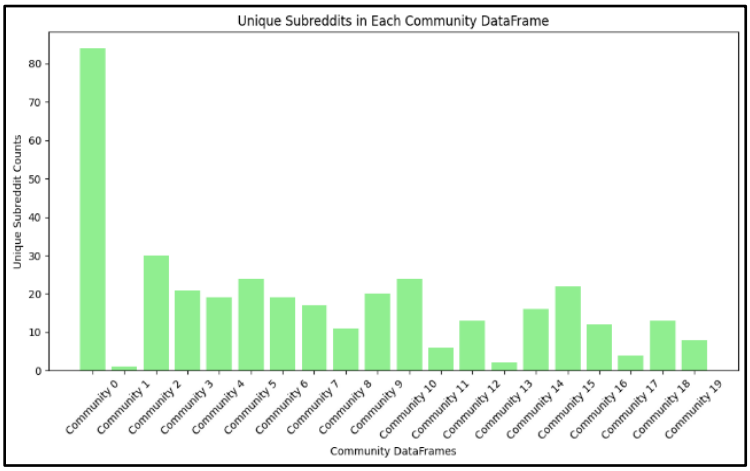


*Figure 15: Distribution of Unique Subreddits per Community*

# 7. Key Steps for ALS Implementation:

## 7.1. Understanding ALS:

Alternating Least Squares (ALS) is a machine learning algorithm used in collaborative filtering-based recommendation systems. It operates by optimizing a matrix factorization problem by alternating between optimizing user factors and item factors. The aim of matrix factorization involves breaking down a large matrix of user-item interactions into two lower dimensional matrices of latent factors, which capture hidden patterns and relationships with the data. The term latent factor refers to user preferences or item characteristics. The first lower dimensional matrix represents the users and the other represents items. This process enables the algorithm to make recommendations by filling in the missing values in the matrix based on users' preferences and items' characteristics. Figure 16 demonstrates the matrix factorization process, while the formula below exhibits matrix factorization mathematically.

$$R \approx U \cdot I^T$$

- R represents the original user-item interaction matrix, while U represents users and I represents items.
- U is a matrix of dimensions m * k, where m is the number of users and k is the number of latent factors.
- $I^T$ is the transpose matrix of dimensions n * k, where n is the number of items.

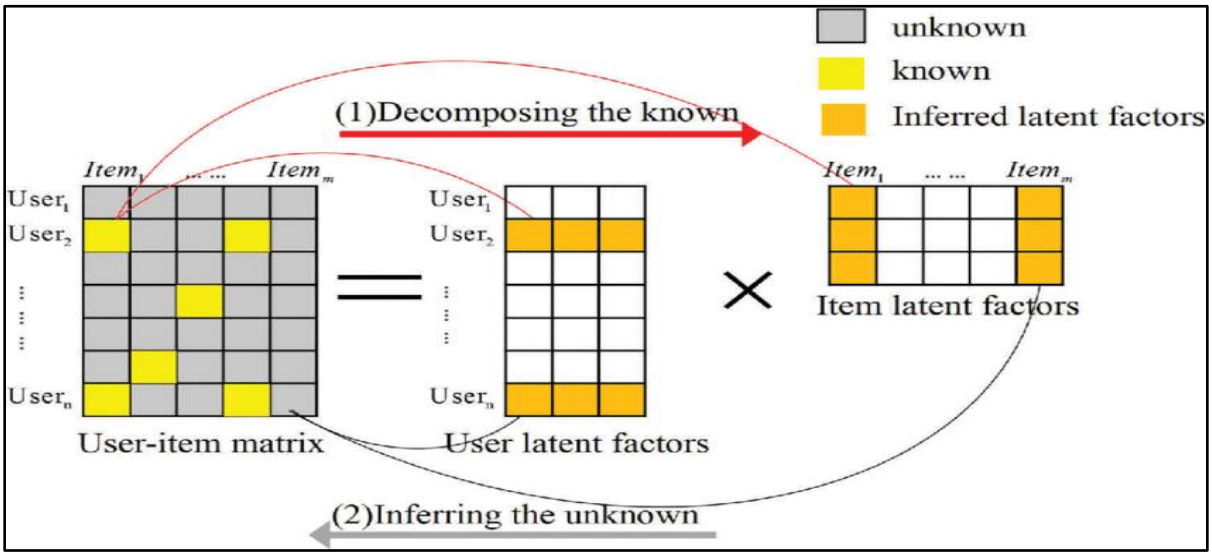


*Figure 16: Matrix Factorization*

The ALS algorithm employs a procedure of alternating optimization between two lower-dimensional matrices: one for users and the other for items. It aims to minimize the difference between the product of these matrices and the original user-item interaction matrix. This process iterates by optimizing one matrix while keeping the other fixed, then switching roles to optimize the second matrix while the first remains constant. The algorithm aims to converge at a point where the difference between the product of the matrices and the original interaction matrix is minimized. ALS is commonly applied in Apache Spark ML and is tailored for addressing large-scale collaborative filtering problems.

It's worth noting that ALS implements L2 regularization as opposed to other algorithms that use L1 regularization. L2 regularization, also known as Ridge regularization, is a technique used in machine learning with the aim of preventing overfitting by adding a penalty term to the loss function. The regularization term is proportional to the sum of the squared values of the coefficients multiplied by the regularization parameter, denoted as λ. Hence, the optimization objective is not only to minimize the error between the predicted value and the actual value, but also aims to minimize the magnitude of the coefficients. L2 regularization term is $\lambda \sum_{j=1}^{p} \theta_j^2$, where:

- λ is the regularization parameter (a hyperparameter).
- $p$ is the number of features in the model.
- $\theta_j$ are the model coefficients.

The key difference between L2 (Ridge) and L1 (Lasso) regularization is that the L1 tends to produce sparse coefficients by pushing less important features to zero, in other words eliminating some features. While L2 tends to shrink the coefficients of correlated features, making them closer to zero but not exactly zero. The addition of the regularization term encourages the model to select smaller coefficients, in order to prevent fitting the training data too closely and making the model more robust by avoiding overfitting.

The addition of L2 regularization result in ALS preventing the learned latent factors from becoming too large, reducing the chances of overfitting the model to the training data and promoting better generalization to unseen data or users.

## 7.2. Types of ALS:

ALS is primarily categorized into two types based on the approach for matrix factorization:

- Explicit Feedback ALS: This type is utilized when the dataset includes direct and numerical feedback provided by users, such as ratings given to items. ALS processes this explicit feedback to factorize the user-item interaction matrix.
- Implicit Feedback ALS: This type is implemented when the dataset does not contain any explicit feedback, but instead it's only employed when implicit feedback is calculated at first and made available in the dataset. Implicit feedback includes user behaviors such as views, purchase history or any form of engagement that does not contain explicit rating.

## 7.3. ALS Hyperparameters:

In the ALS algorithm, there are three hyperparameters that can be tuned in order to optimize the model. They are essential for controlling the behavior and performance of the ALS algorithm. Tuning them correctly can indeed impact significantly the model's accuracy and ability to generalize to unseen data. These three hyperparameters are the following:

- Rank: Number of latent factors to be used in the matrix factorization model. A higher rank might provide more modeling power but could potentially lead to overfitting.
- Iterations: Number of iterations the algorithm performs in order to optimize the factorization matrices.
- Regularization Parameter: It controls the strength of the regularization term to prevent overfitting. A higher value imposes stronger regularization.

## 7.4. Application of ALS:

The type of ALS that is going to be implemented in this study is implicit feedback ALS as there is no explicit ratings provided in this project's dataset. It will be implemented on each community separately and on the whole dataset. The aim of this is to make a comparison between both outputs and determine if the integration of ALS and community detection can lead to more personalized recommendations for the individual user. The only community in which the ALS will be applied is community 0, the reason is it contains an appropriate number of records and 80 unique subreddits.

The code for the implementation of ALS recommendation system follows Object-Oriented Programming (OOP) techniques in Python. In the code, different classes are defined, and each class encapsulates relevant functionalities and methods:

- DataLoader
- DataPreprocessor
- ImplicitScoreCalculator
- ScoreNormalizer
- TrainTestSplit
- ALSCustom
- Recommender
- AccuracyPlots

These classes, detailed in the upcoming sections, follow the Object-Oriented Programming (OOP) paradigm. They encapsulate specific functionalities, enhancing modularity, reusability, and readability. This approach contributes to a more structured and organized ALS recommendation system development.

## 7.5.Preprocessing Variables for Implicit Score Calculation:

Before utilizing ALS, the calculation of the implicit score is a crucial initial step. This process requires preliminary data preprocessing to create specific variables that are essential for computing the implicit score, which serves as a key input for the ALS models. These variables are the following:

- Interactions: Represents the normalized count of interactions between user ID and Subreddit pairs within the provided dataframe. This variable signifies the level of interaction or engagement of each user with specific subreddits.
- Average Population: Indicates the normalized average population for each subreddit. This value represents the mean 'Population' attribute for posts within a subreddit.
- Average Upvotes: Represents the normalized average upvotes for each subreddit. It is derived from the mean number of upvotes received for posts within a particular subreddit.
- Subreddit Popularity: Indicates the normalized popularity of each subreddit within the dataframe. This score is computed from the count of unique user IDs associated with each subreddit.
- Category Popularity: Represents the normalized popularity of each category in the provided dataframe. It is calculated based on the count of unique user IDs within each category.

## 7.6. Implicit Score Calculation:

In this section, the focus revolves around the computation and calculation of implicit scores that are pivotal in preparing the data for the application of the Implicit ALS algorithm. The implicit score is computed using the following formula:

$$Partial\ Score = I * CP * SP * AVG\ Population * AVG\ Upvotes$$

$$Implicit\ Score = (1 - \alpha) * (Partial\ Score) + \alpha * PR$$

Where:

- I is the normalized interactions score
- CP is the normalized category popularity score
- SP is the normalized subreddit popularity score
- AVG Population is the normalized average population score
- AVG Upvotes is the normalized average subreddit upvotes
- PR is the normalized PageRank score, using the min-max scaling technique, (either the community PageRank score or the PageRank score regarding the entire network)
- $\alpha$ is the parameter that controls the balance between the partial score and the PageRank scores. Higher $\alpha$ gives more weight to the PageRank scores. It's worth noting that this variable has been experimented with in order to see if $\alpha$ has an effect on the accuracy of the recommendations or not. The $\alpha$ values that have been tried are 0.2, 0.4, 0.6 and 0.8.

After computing the implicit score, the outliers have been handled. This was done by first determining the Lower Quartile ($25^{th}$ percentile), the Upper Quartile ($75^{th}$ percentile) and the Interquartile Range (IQR), which is the difference between the upper and lower quartiles. The implicit score is then adjusted by transforming values that fall below the Lower Bound to the Lower Bound value and those above the Upper Bound to the Upper Bound value. This process helps cap extreme values (outliers) that significantly deviate from the majority of data points, ensuring they fall within a reasonable range defined by the quartiles and IQR.

## 7.7. Implicit Score Normalization:

The implicit score are normalized using the Z-scores, which are measures of how many standard deviations a data point is from the mean. The reason behind normalizing the implicit scores is to enhance the ALS model's performance and robustness during the training process.

## 7.8. Additional Data Preprocessing:

It's worth noting that some columns have been dropped in order to get a final dataframe containing user interactions with unique subreddits, in other words duplicate records have been dropped. The columns that have been dropped are the following: Upvotes, Date and Post title. The following figures are representations of the initial dataframe and the final dataframe that is ready for the next phase.

| | Subreddit | Population | Upvotes | User_ID | Community_Label | PageRank | PR_Community | Category | Subreddit_Popularity | Category_Popularity | Average_Upvotes | Average_Population | Interactions |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | worldnews | 32145109 | 29941 | Bay1Bri | 0 | 0.000159 | 0.168775 | World News | 0.392749 | 1.000000 | 0.473722 | 0.596681 | 0.081847 |
| 1 | pcmasterrace | 7997200 | 19702 | chocotripchip | 0 | 0.000223 | 0.144485 | Technology | 0.015106 | 0.583472 | 0.314021 | 0.060301 | 0.035677 |
| 2 | explainlikeimfive | 22391279 | 792 | Soranic | 0 | 0.000104 | 0.105242 | Learning and Education | 0.311178 | 0.810089 | 0.046137 | 0.380027 | 0.041973 |
| 3 | Jokes | 26192558 | 16467 | ReubenZWeiner | 0 | 0.000253 | 0.179758 | Humor | 0.015106 | 0.887240 | 0.140143 | 0.464462 | 0.067156 |
| 4 | EarthPorn | 23374821 | 3421 | toastibot | 0 | 0.000032 | 0.017686 | Outdoors and Nature | 0.012085 | 0.115727 | 0.123469 | 0.401873 | 1.000000 |

*Figure 17: Initial Dataframe*

| | Subreddit | Population | User_ID | Community_Label | PageRank | PR_Community | Category | Subreddit_Popularity | Category_Popularity | Average_Upvotes | Average_Population | Interactions | Implicit_Score | Normalized_Implicit_Scor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | worldnews | 32145109 | Bay1Bri | 0 | 0.000159 | 0.168775 | World News | 0.392749 | 1.000000 | 0.473722 | 0.596681 | 0.081847 | 0.041024 | 0.6544 |
| 1 | pcmasterrace | 7997200 | chocotripchip | 0 | 0.000223 | 0.144485 | Technology | 0.015106 | 0.583472 | 0.314021 | 0.060301 | 0.035677 | 0.026902 | 0.0048 |
| 2 | explainlikeimfive | 22391279 | Soranic | 0 | 0.000104 | 0.105242 | Learning and Education | 0.311178 | 0.810089 | 0.046137 | 0.380027 | 0.041973 | 0.021197 | -0.4080 |
| 3 | Jokes | 26192558 | ReubenZWeiner | 0 | 0.000253 | 0.179758 | Humor | 0.015106 | 0.887240 | 0.140143 | 0.464462 | 0.067156 | 0.035098 | 0.3851 |
| 4 | EarthPorn | 23374821 | toastibot | 0 | 0.000032 | 0.017686 | Outdoors and Nature | 0.012085 | 0.115727 | 0.123469 | 0.401873 | 1.000000 | 0.003593 | -1.3515 |

*Figure 18: Final Dataframe*

## 7.9. Splitting The Dataset:

In order to evaluate the performance of the ALS model accurately, implementing ALS requires the dataset to be split into train and test sets. This separation allows the model to be trained on the train set while enabling the evaluation of its performance on the unseen data within the test set. However, there are various ways a train/test split can be conducted, such as random split or splitting by date, etc. The train/test split that was conducted for this study is taking the most recent interaction made by the user and storing it in the test set, while the other interactions made by the user will be stored in the training set.

Before splitting the data, an initial step is required, which is string indexing certain string columns to convert categorical variables into numerical indices. String indexing is a process that involves converting categorical string variables or labels into numerical representations. This transformation is critical in machine learning tasks where algorithms usually expect numerical input. String indexing assigns a unique numerical identifier to each distinct category or label within a particular column, creating an index to represent each categorical value. This numeric representation enables machine learning algorithms to process and analyze the categorical data effectively.

In the context of ALS or CF models, string indexing is important for two reasons. The first one is that ALS models require numeric input for users and items or any categorical variable used in the recommendation process. The second reason is that it facilitates the matrix factorization process due to the indexing of the strings to numeric values. Hence, it becomes feasible to perform ALS on categorical data, particularly when building recommendation systems, allowing the model to generate predictions based on user-item interactions. The following figure exhibits the transformation of the user and subreddit string values into numerical ones.

| User_ID_index | Subreddit_index | Community_Label | Normalized_Implicit_Score |
|---|---|---|---|
| 2139.0 | 14.0 | 0 | -0.6717461375715454 |
| 2072.0 | 5.0 | 0 | -0.5914450154068424 |
| 2164.0 | 1.0 | 0 | -0.09702742315152872 |
| 434.0 | 10.0 | 0 | 0.6873559697262259 |
| 707.0 | 13.0 | 0 | -0.7873870170632525 |
| 141.0 | 25.0 | 0 | 0.6988568213669437 |
| 1319.0 | 1.0 | 0 | -0.12136829687276052 |
| 85.0 | 0.0 | 0 | 0.25847855337914905 |
| 1368.0 | 47.0 | 0 | -1.269921330224879 |
| 669.0 | 0.0 | 0 | -0.04856035015716567 |
| 472.0 | 20.0 | 0 | -0.8644516633137516 |
| 1733.0 | 15.0 | 0 | -0.5862754285930266 |
| 307.0 | 42.0 | 0 | -0.4723044401438635 |
| 214.0 | 15.0 | 0 | 0.36515875233758005 |
| 2137.0 | 16.0 | 0 | -0.6716358010012643 |
| 224.0 | 28.0 | 0 | -0.5878985004250146 |
| 425.0 | 4.0 | 0 | 0.4748006542588755 |
| 1489.0 | 10.0 | 0 | -0.7740436421187318 |
| 545.0 | 2.0 | 0 | -0.3342948243858021 |
| 159.0 | 0.0 | 0 | 2.4648477950549688 |

*Figure 19: String Values Transformed into Numerical Ones*

When implementing the ALS technique, it's important to note a potential issue that might arise if not properly addressed. For example, if a user has only made a single interaction, this lone interaction might be isolated within the test set, preventing the model from learning about this specific user's preferences or behavior adequately. To address this issue, various thresholds have been experimented with to ascertain which threshold value could lead to higher model accuracy. These thresholds represent the minimum count of unique interactions a user should have, and they are set at 3, 4, and 5. These thresholds are applied to ensure that users included in the test set have made a minimum number of interactions, enabling the model to learn more effectively from their behavior. This leads to having six train sets and six test sets of different sizes.

## 7.10. Fitting The ALS Models:

The subsequent phase in the process of generating personalized recommendations involves fitting ALS models. This step utilizes specific hyperparameters to create and train multiple ALS models. This is accomplished through a two-step procedure:

- Generating a series of ALS models, incorporating various hyperparameters. This step involves iterating through different combinations of parameters such as rank, iterations, and regularization to produce a list of ALS models.
- The generated ALS models are trained on the provided training dataset and each model is fitted using the corresponding ALS algorithm parameters.

The ultimate objective of this phase is to explore various configurations of ALS models and determine the most effective model for the recommendation system. Each model is trained with different settings, and the best-performing models are saved for further evaluation and recommendation generation.

The selected hyperparameters, shown in Figure 20, for this phase are the following:

- ranks = [3, 4, 5]
- maxIters = [10, 15]
- regParams = [0.001, 0.01, 0.1]

```
model_0, rank: 3, maxIterations: 10, regParam: 0.001
model_1, rank: 3, maxIterations: 10, regParam: 0.01
model_2, rank: 3, maxIterations: 10, regParam: 0.1
model_3, rank: 3, maxIterations: 15, regParam: 0.001
model_4, rank: 3, maxIterations: 15, regParam: 0.01
model_5, rank: 3, maxIterations: 15, regParam: 0.1
model_6, rank: 4, maxIterations: 10, regParam: 0.001
model_7, rank: 4, maxIterations: 10, regParam: 0.01
model_8, rank: 4, maxIterations: 10, regParam: 0.1
model_9, rank: 4, maxIterations: 15, regParam: 0.001
model_10, rank: 4, maxIterations: 15, regParam: 0.01
model_11, rank: 4, maxIterations: 15, regParam: 0.1
model_12, rank: 5, maxIterations: 10, regParam: 0.001
model_13, rank: 5, maxIterations: 10, regParam: 0.01
model_14, rank: 5, maxIterations: 10, regParam: 0.1
model_15, rank: 5, maxIterations: 15, regParam: 0.001
model_16, rank: 5, maxIterations: 15, regParam: 0.01
model_17, rank: 5, maxIterations: 15, regParam: 0.1
```

*Figure 20: Fitted Models with Different Combinations of Hyperparameters*

# 8. Generating and Evaluating Recommendations:

## 8.1. Generating Recommendations:

This section delves into the process of generating recommendations using the ALS models that were previously trained. The aim is to utilize the trained models in order to predict and generate personalized recommendations for the individual user based on the user's interactions within the overall dataset. A total of 15 recommendations were chosen as the standard quantity for the recommendation system. These recommendations are intended to offer a diverse yet limited set of suggestions, ensuring a more manageable and focused outcome. Figure 21 exhibits the generation of ten subreddit recommendations for five users. While Figures 22 and 23 display how the recommendations are sorted in descending order based on similarity scores, which ensures the user of getting the top ten subreddit recommendations.

```
User_ID | | recommendations
adviceKiwi ['television', 'movies', 'todayilearned', 'videos', 'technology', 'funny', 'mildlyinteresting', 'pics', 'OldSchoolCool', 'AdviceAnimals', 'WTF',
dandroid126 ['todayilearned', 'funny', 'mildlyinteresting', 'pics', 'AdviceAnimals', 'technology', 'WTF', 'aww', 'videos', 'explainlikeimfive', 'OldSchool(
Schiffy94 ['politics', 'news', 'worldnews', 'technology', 'todayilearned', 'funny', 'AdviceAnimals', 'mildlyinteresting', 'pics', 'nottheonion', 'interesti
Rosebunse ['television', 'todayilearned', 'movies', 'videos', 'technology', 'funny', 'mildlyinteresting', 'pics', 'AdviceAnimals', 'WTF', 'OldSchoolCool',
McFeely_Smackup ['television', 'movies', 'todayilearned', 'videos', 'technology', 'funny', 'mildlyinteresting', 'OldSchoolCool', 'pics', 'AdviceAnimals', '
----------------------------------------------------------------
```

*Figure 21: Ten Subreddit Recommendations for Six Users*

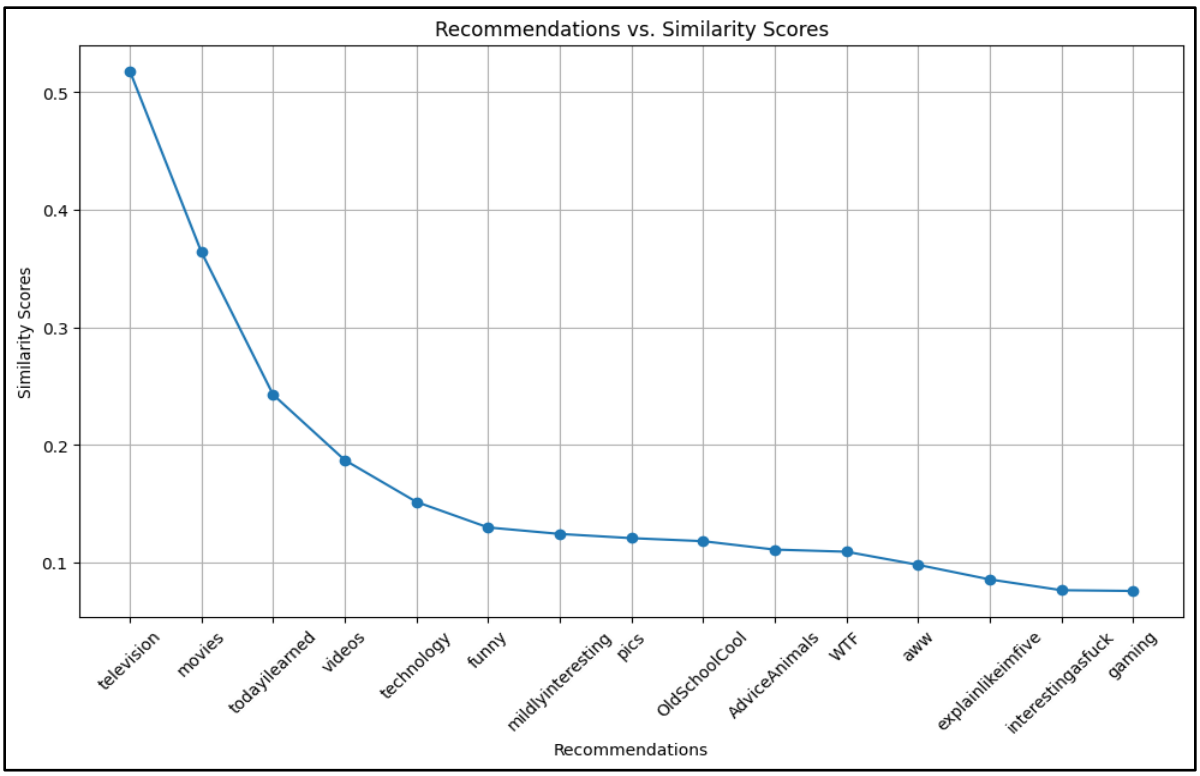


*Figure 22: Top Ten Recommendations Sorted in Descending Order (Within Community)*

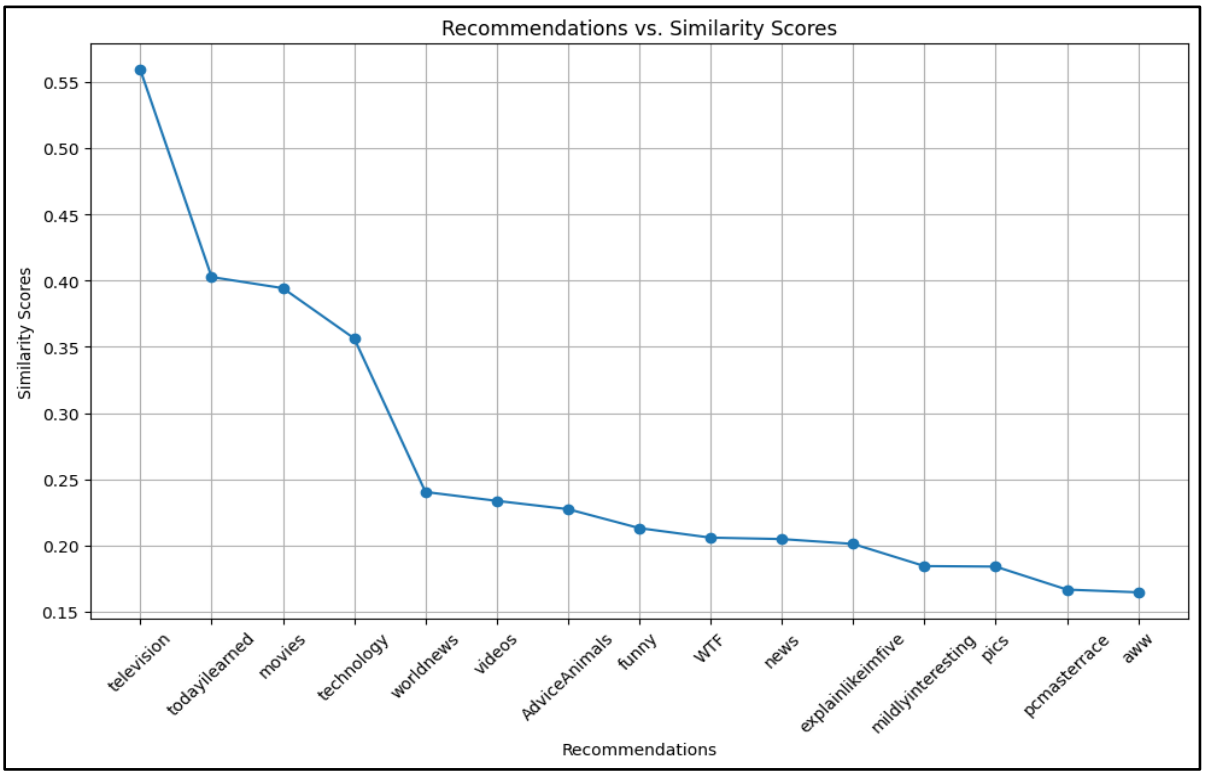


*Figure 23: Top Ten Recommendations Sorted in Descending Order (Overall Dataset)*

## 8.2.Evaluating Recommendations:

This section aims to evaluate the recommendations generated by the various ALS models that have been trained on the training dataset. Since the test set comprises of a single user interaction, which is the most recent one, a specific evaluation metric is suitable for such case. The evaluation metric that have been used for this project is Hit@k. This evaluation metric determines whether the system's recommendation list includes the relevant items within the top-k recommendations for a user. It computes the percentage of users for whom the recommended subreddit(s) is within the top-k recommendations. Figures 24 and 25 demonstrate an example of the models' Hit@k precision scores. The mathematical formula of Hit@k is given below:

$$Hit@k = \frac{Number\ of\ Hits\ at\ k}{Total\ number\ of\ users}$$

Where:

- Number of Hits at k refers to the number of cases where the relevant subreddit (or subreddits) appears in the top-k recommendations for a user.
- Total number of users is the entire user set considered for evaluation.

```
model_0 Hit@k: 0.675, Top Recommendation Precision: 0.075
generate_recommendation complete
model_1 Hit@k: 0.675, Top Recommendation Precision: 0.075
generate_recommendation complete
model_2 Hit@k: 0.7, Top Recommendation Precision: 0.075
generate_recommendation complete
model_3 Hit@k: 0.675, Top Recommendation Precision: 0.075
generate_recommendation complete
model_4 Hit@k: 0.675, Top Recommendation Precision: 0.075
generate_recommendation complete
model_5 Hit@k: 0.7, Top Recommendation Precision: 0.075
generate_recommendation complete
model_6 Hit@k: 0.625, Top Recommendation Precision: 0.05
generate_recommendation complete
model_7 Hit@k: 0.65, Top Recommendation Precision: 0.05
generate_recommendation complete
model_8 Hit@k: 0.65, Top Recommendation Precision: 0.025
generate_recommendation complete
model_9 Hit@k: 0.7, Top Recommendation Precision: 0.025
generate_recommendation complete
model_10 Hit@k: 0.65, Top Recommendation Precision: 0.05
generate_recommendation complete
model_11 Hit@k: 0.675, Top Recommendation Precision: 0.025
generate_recommendation complete
model_12 Hit@k: 0.575, Top Recommendation Precision: 0.025
generate_recommendation complete
model_13 Hit@k: 0.55, Top Recommendation Precision: 0.0
generate_recommendation complete
model_14 Hit@k: 0.6, Top Recommendation Precision: 0.05
generate_recommendation complete
model_15 Hit@k: 0.575, Top Recommendation Precision: 0.025
generate_recommendation complete
model_16 Hit@k: 0.625, Top Recommendation Precision: 0.0
generate_recommendation complete
model_17 Hit@k: 0.65, Top Recommendation Precision: 0.05
min_subreddit: 5 complete
```

*Figure 24: Models' Hit@k Precision (Within Community)*

```
model_0 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_1 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_2 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_3 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_4 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_5 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_6 Hit@k: 0.49019607843137253, Top Recommendation Precision: 0.0196078431372549
generate_recommendation complete
model_7 Hit@k: 0.49019607843137253, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_8 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0196078431372549
generate_recommendation complete
model_9 Hit@k: 0.49019607843137253, Top Recommendation Precision: 0.0196078431372549
generate_recommendation complete
model_10 Hit@k: 0.49019607843137253, Top Recommendation Precision: 0.0196078431372549
generate_recommendation complete
model_11 Hit@k: 0.5098039215686274, Top Recommendation Precision: 0.0196078431372549
generate_recommendation complete
model_12 Hit@k: 0.5490196078431373, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_13 Hit@k: 0.5686274509803921, Top Recommendation Precision: 0.058823529411764705
generate_recommendation complete
model_14 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0784313725490196
generate_recommendation complete
model_15 Hit@k: 0.5490196078431373, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_16 Hit@k: 0.5294117647058824, Top Recommendation Precision: 0.0392156862745098
generate_recommendation complete
model_17 Hit@k: 0.5490196078431373, Top Recommendation Precision: 0.0784313725490196
```

*Figure 25: Models' Hit@k Precision (Overall Dataset)*

After analyzing the Hit@k precision scores for each model, the next step involves selecting the highest precision scores. To achieve this, the three top scores will be retrieved, each reflecting the highest precision score with regards to the minimum subreddit threshold set during the dataset splitting phase. After implementing the ALS recommendation system on the overall dataset, the following highest precision scores are achieved (Figure 26 and 27).

| ***Minimum Subreddit*** | **Alpha** | **Highest Precision Score** |
|---|---|---|
| *3* | 0.4 | 0.489796 |
| *4* | 0.2 | 0.515464 |
| *5* | 0.2 | 0.568627 |

*Figure 26: Overall Dataset Highest Precision Scores*

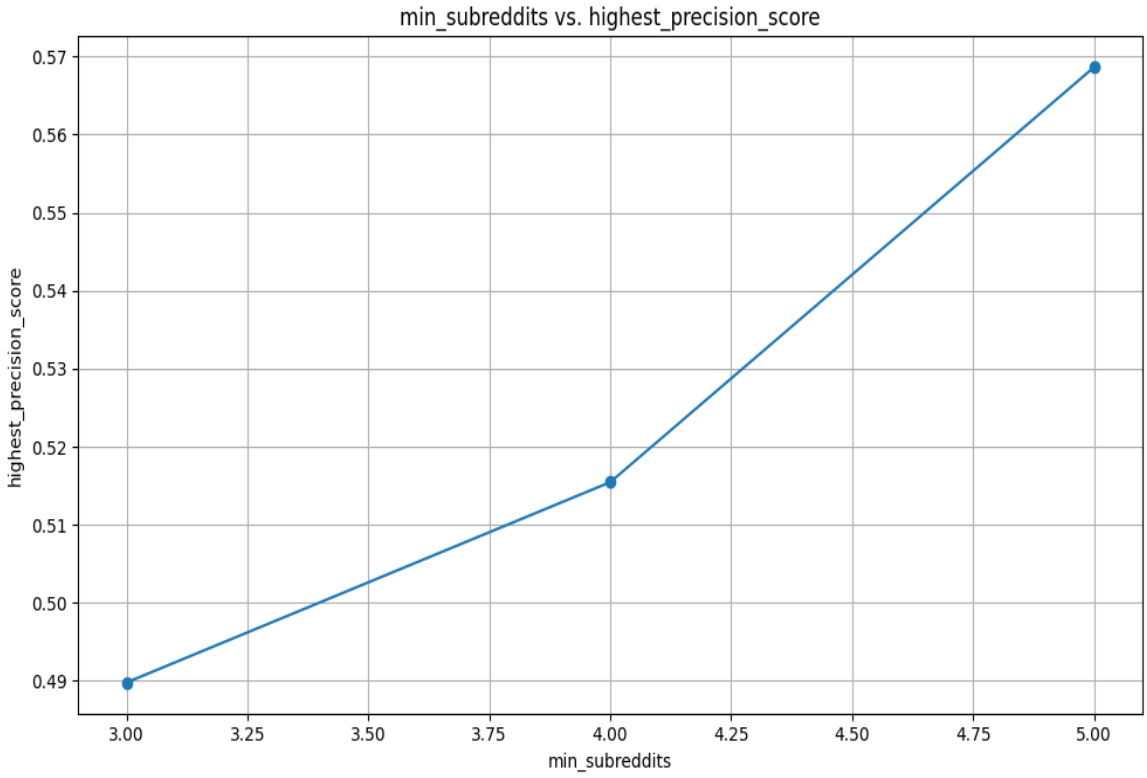


*Figure 27: Highest Precision Score per Minimum Subreddit (Overall Dataset)*

Alternatively, the following figures demonstrate highest precision scores achieved after implementing the ALS recommendation system on the community 0 dataset.

| ***Minimum Subreddit*** | **Alpha** | **Highest Precision Score** |
|---|---|---|
| *3* | 0.8 | 0.523256 |
| *4* | 0.8 | 0.615385 |
| *5* | 0.4 | 0.725000 |

*Figure 28: Community 0 Dataset Highest Precision Scores*

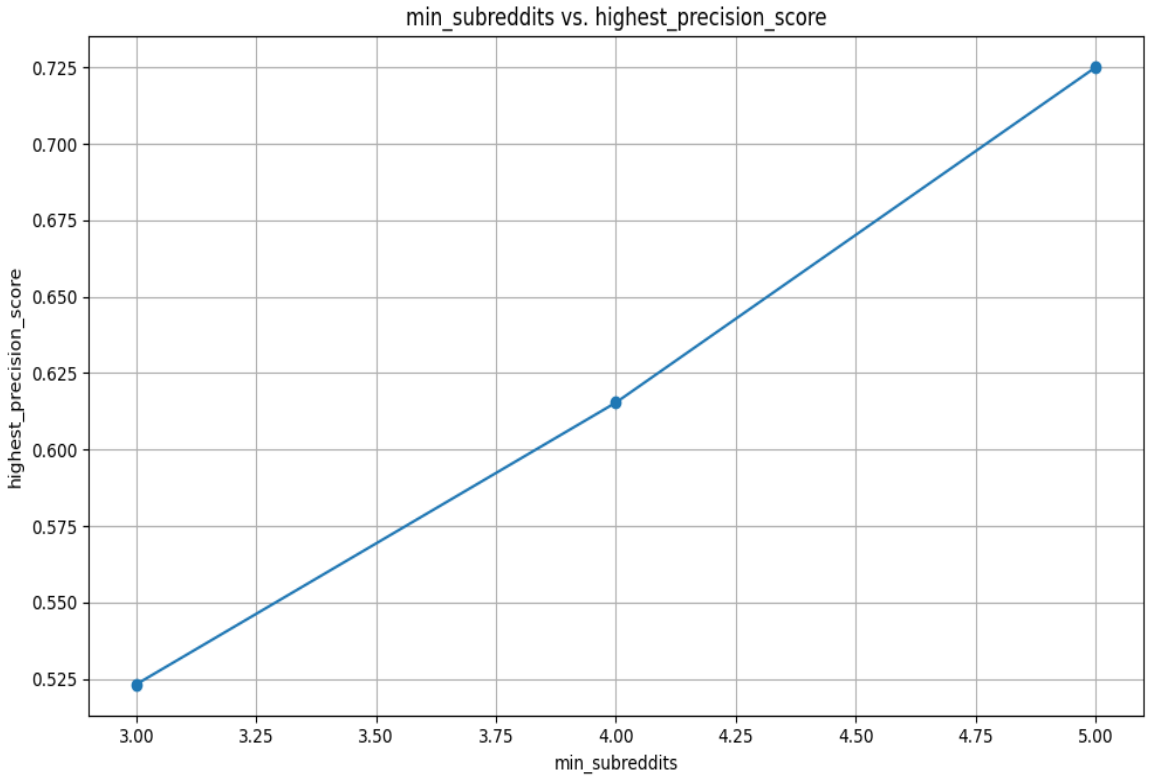


*Figure 29: Highest Precision Score per Minimum Subreddit (Within Community)*

## 9. Results and Discussion:

This section aims to interpret the findings obtained and their relevance to the project's objectives. The evaluation focuses on two aspects. The first aspect is the influence of users within the community and the wider dataset. Figure 26 illustrates that user influence in the dataset does not affect as significantly the precision accuracy, as the α values are considered low (α = 0.2 and 0.4) across the minimum subreddit trials. In contrast, Figure 28 indicates that within the community, higher α values are observed in each minimum subreddit trial. Particularly, when the minimum subreddit thresholds are 3 and 4 (α = 0.8). It's worth noting that the highest precision score 0.725 was achieved when minimum subreddit threshold was 5. These results suggest that user influence within the community has a more significant impact on generating personalized recommendations compared to user influence in the broader dataset.

The second aspect is the Hit@k precision scores, which reveals a noteworthy trend: models exhibit superior performance when applied to a community dataset compared to a wider dataset. Precision scores significantly improve when utilizing the recommendation system on the community dataset in contrast to the broader dataset. Additionally, increasing the minimum subreddit threshold correlates with higher precision scores. This is an expected outcome, as a greater minimum subreddit threshold allows the model to learn more about user behavior, resulting in more accurate recommendations aligned with user preferences.

# 10. Conclusion:

In conclusion, this thesis project explored the development and evaluation of a graph-based recommendation system leveraging community and influence features. This research confirmed the initial hypothesis or in other words answered this thesis project main question, which is whether the implementation of the recommendation system on a community dataset will result in a more personalized recommendations for the individual user, and indeed this has been confirmed through this experimentation. Additionally, the study uncovered the significant impact of user influence within the community compared to the broader dataset, demonstrating the effectiveness of community-based recommendations for personalized suggestions. Moreover, the observed increase in precision scores with higher minimum subreddit thresholds highlighted the role of user behavior in refining recommendation accuracy.

In essence, this project could be considered as an initial step towards evolving the field of recommendation systems, as it demonstrates the importance of community-specific approaches in refining and enhancing the precision and relevance of recommendations. This concept can be implemented on various social media platforms such Facebook, YouTube, Instagram, etc., which in return would lead to more relevant user interactions.

Future considerations might involve the implementation of the integrated approach combining community detection algorithms with collaborative-filtering on a cloud-based framework. This infrastructure could handle significantly larger network graphs, allowing for efficient matrix factorization and the generation of highly accurate, personalized recommendations.

# 11. References:


1. Deshmukh, Dipika, and Dr. D. R. Ingle. "A Community Detection and Recommendation System." International Research Journal of Engineering and Technology (IRJET), vol. 04, no. 07, July 2017, pp. 752-757.
2. Shokrzadeh, Zeinab, Mohammad-Reza Feizi-Derakhshi, Mohammad-Ali Balafar, Jamshid Bagherzadeh Mohasefi. "Graph-Based Recommendation System Enhanced with Community Detection."
3. Gasparetti, Fabio, Alessandro Micarelli, Giuseppe Sansonetti. "Community Detection and Recommender Systems." January 2017.