Exploring Stargazer Influence and Growth Prediction on GitHub: A Network Analysis Approach

Andrea Zečević[1,2*]

[1*]Faculty of Science, University of Split, Ruera Bošković 33, Split, 21000, Croatia.

Corresponding author(s). E-mail(s): azecevic@pmfst.hr;

Abstract

This study explores the dynamics of stargazer activity within GitHub repositories through network analysis, community detection, and predictive modeling. Utilizing the GitHub API, data on repository stargazers are gathered and structured into a network where nodes represent GitHub users (stargazers) and edges mutual repository stargazing. Centrality measures, like Betweenness Centrality, are calculated to find important stargazers and understand how they affect repository popularity. Community detection techniques have enabled me to identify groups of stargazers within the network. Using daily data on the growth of stargazers allows to observe changes in repository interest over time. I trained a Long Short-Term Memory (LSTM) model on this dataset, incorporating both stargazer growth patterns and Betweenness Centrality measures.

Keywords: GitHub, Stargazers, Network Analysis, LSTM

1 Introduction

GitHub is one of the largest platforms for open source collaboration. Millions of developers can share, collaborate, and contribute to projects. Within this ecosystem, the concept of ”stargazers” has emerged as a key metric for measuring the popularity and user interest in a GitHub repository. When users find a repository particularly useful or interesting, they can ”star” it, similar to liking posts on social media platforms [1]. The number of stars a repository accumulates serves as an indicator of its relevance, quality, and community interest. Repositories with a higher number of stars are often perceived as more influential and successful.

The aim of this project is to predict the number of new stargazers for individual GitHub repositories. This study builds upon the work from Varuna and Mohan [2] which proposes using social network analysis for prediction. Therefore, within this project, I will construct a network of stargazers. Within the network, the most influential stargazers will be detected, and betweenness centrality measure for all stargazers will be extracted to be used in training prediction models. Additionally, communities within the network will be detected for further analysis. The focus will be on predicting the number of stargazers for newly created repositories. Based on data from the first 30 days, the aim is to forecast the count for the 31st day using LSTM, which has been proven to be a robust model for time series analysis [3].

Sections to follow will include a literature review process, detailing how I sourced relevant literature. Subsequently, the section will discuss related work to date. The methodology will then outline how I acquired and structured the dataset for model training. Following this, I will present the network analysis, including betweenness centrality calculations and community detection. This will be followed by the LSTM model. Results will be discussed, and suggestions for future work will be provided.

2 Literature Review

To review the existing literature on predicting the number of stargazers on GitHub repositories, I conducted a systematic search using key databases: Web of Science, IEEE Xplore, and Google Scholar. The primary keywords included terms such as GitHub, trending, stargazers, and repository, with GitHub as a core term. To exclude irrelevant results that reference GitHub for project hosting rather than as a study subject, the search string specifically excluded "github.com". I also included terms associated with popularity (e.g., popular*, star*, stargaz*, trend*) and prediction methodologies (e.g., predict*, model*) to refine my focus.

The search query was structured as follows: ((ALL=(GitHub NOT github.com)) AND ALL=(popular* OR star? OR stargaz* OR trend*)) AND ALL=(predict* OR model*).

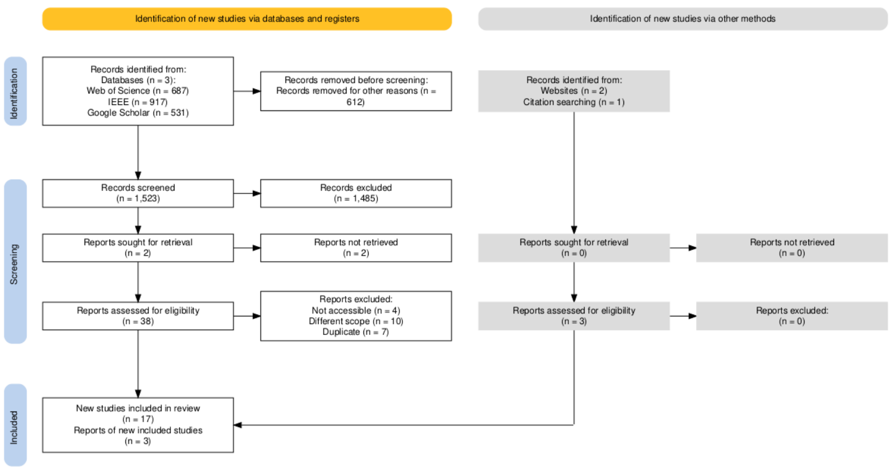
2.1 Search and Filtering Process

The search and filtering process followed the PRISMA methodology. The process is visualized in the PRISMA flow diagram shown in Figure 1.

The initial search in the Web of Science database yielded 687 papers. After excluding irrelevant categories like telecommunications, astrophysics, and geography, the number was reduced to 443. Further filtering by selecting relevant categories such as Computer Science Software Engineering and Computer Science Theory Methods narrowed the results down to 365 papers. Conference papers were included to avoid excluding pertinent studies.

The IEEE Xplore search initially produced 917 results. Applying thematic filters similar to those used in Web of Science reduced the results to 628 papers. Excluding certain document types was avoided to retain relevant papers.

Google Scholar lacks advanced filtering options by subject category. By specifying a date range from 2007 (the year GitHub was launched) to the present, I slightly narrowed the search results. This filter excluded only one paper, maintaining a broad scope for 530 potential inclusions.



Identification of new studies via databases and registers
Identification of new studies via other methods
Identification
Screening
Included
Records identified from: Databases (n = 3): Web of Science (n = 687) IEEE (n = 917) Google Scholar (n = 531)
Records removed before screening: Records removed for other reasons (n = 612)
Records screened (n = 1,523)
Records excluded (n = 1,485)
Reports sought for retrieval (n = 2)
Reports not retrieved (n = 2)
Reports assessed for eligibility (n = 38)
Reports excluded: Not accessible (n = 4) Different scope (n = 10) Duplicate (n = 7)
New studies included in review (n = 17) Reports of new included studies (n = 3)
Records identified from: Websites (n = 2) Citation searching (n = 1)
Reports sought for retrieval (n = 0)
Reports not retrieved (n = 0)
Reports assessed for eligibility (n = 3)
Reports excluded: (n = 0)

Fig. 1: PRISMA flow diagram illustrating the search and filtering process.

2.2 Screening and Selection

The initial pool of documents was screened based on their titles. Papers with titles closely aligned with the study's focus were further examined by reading their abstracts. This process led to the exclusion of studies unrelated to the core topic, such as those focusing on project recommendations, developer performance analysis, software development processes, code quality analysis, and the use of GitHub Copilot.

From the refined list, 38 papers appeared relevant. Upon deeper examination, 2 papers were inaccessible, and another 2 required payment for access. After evaluating the remaining papers, I concluded that 10 of them were not useful for the study's objectives. Additionally, 7 duplicates were identified and removed.

2.3 Final Selection

Ultimately, 17 references were highly relevant and selected for inclusion in this study. The following section will discuss these selected works, highlighting their contributions and methodologies. Additionally, two additional sources were identified through other methods: GitHub API documentation and NetworkX documentation. Furthermore, one paper was found through citation, which includes particularly significant information relevant to Long Short-Term Memory (LSTM).

3 Related Work

GitHub is one of the most popular platforms for software development. It attracts millions of users and projects. Successful and popular projects on the platform not only draw the attention of developers but also attract investors who see them as lucrative opportunities. But what exactly defines a project's popularity?

3.1 Metrics of GitHub Popularity

In 2014, researchers attempted to predict the number of forks a project might receive. A fork enables a user to duplicate the original project, creating their own version. This action signifies the user's interest in the repository. It also shows their intent to contribute, usually through code editing. When a project has more forks, it indicates higher interest among developers. This increased interest suggests a higher probability of the project's success in the future [4]. However, the number of stars, rather than forks, is predominantly used as the metric of popularity. While forks can indicate interest in a project, many forks remain inactive, which makes stars a more reliable indicator of a project's popularity and potential success [5]. A study by Hudson Borges, Andre Hora, and Marco Tulio Valente analyzed the top 2,500 starred GitHub repositories. They found that repositories owned by organizations attract more stars than those owned by individuals. Repositories tend to receive more stars immediately after their initial public release. However, for half of these repositories, the growth rate stabilizes over time. This indicates that bursts of popularity do not fully account for the sustained popularity growth of most repositories [6]. In their later study they applied multiple linear regressions to predict the popularity of GitHub repositories. They found that general models, trained with data from top GitHub repositories over six months, accurately predict the number of stars six months ahead [1]. They conducted a survey revealing that developers consider stars the most useful metric for assessing project popularity on GitHub. They star repositories for three primary reasons: to express appreciation for projects, to bookmark them, and because they actively use them. Three-quarters of developers surveyed indicated that they take into account the number of stars before deciding to use or contribute to GitHub projects [7]. While Borges et al.'s study mainly focused on the empirical analysis of the correlations between the features and the popularity of GitHub projects, Han et al. [8] proposes an automated approach to predict the popularity of a project. It implements a Random Forest classifier using 35 extracted features and a larger scale dataset composed of 409,784 GitHub projects.

3.2 Temporal and Predictive Aspects of GitHub Popularity

The concept of popularity is inherently tied to time. Over time, repositories generally experience an increase in the number of forks, stars, and watchers, often with sudden spikes in popularity, particularly for those created by large tech companies. The study by Abduljaleel Al-Rubaye and Gita Sukthankar used 36,000 randomly selected public repositories from GHTorrent. They introduced a weight-based popularity score (WTPS), which incorporates the historical data of other popularity indicators such as forks and stars. The results indicate that WTPS has a strong correlation with both forks and stars, although it is more closely related to stars. Thus, gaining stars has a more significant impact on increasing a repository's WTPS compared to gaining forks [9]. Not only that, GitHub features its own ranking system known as GitHub Trending. When a project accumulates a significant number of stars within a specific timeframe, such as a day or a week, it can appear in the GitHub Trending section, thereby gaining visibility to a broader audience of potential users [10]. The study titled "Trend Prediction of GitHub using Time Series Analysis" by Varuna T V and Anuraj Mohan utilizes time series analysis to forecast trends on GitHub across three categories including repository trends. This approach involves applying LSTM (Long Short-Term Memory) models to data sourced from GH Archive. This work is relevant to my research as it demonstrates the effectiveness of time series analysis in capturing temporal patterns and predicting trends on GitHub [2]. To my knowledge, no prior research has attempted to investigate the network structure of GitHub and apply social network analysis methods for trend prediction, as suggested by the authors of the above-mentioned study. Previous studies have primarily analyzed user influence [11], identified influential developers, repositories, languages, etc. [12], or focused on popular projects [13] without utilizing these metrics for predicting stargazer numbers.

3.3 Research Questions

My study aims to utilize network analysis techniques. These techniques are intended to identify influential stargazers and communities within the GitHub network. Subsequently, the goal is to predict the number of new stargazers.

Research Question 1: How can influential stargazers be identified within the GitHub user network?

Research Question 2: How can communities be effectively detected within the GitHub stargazer network?

Research Question 3: Can centrality measures of stargazers be used to predict the growth in the number of new stargazers for GitHub projects?

4 Methodology

To address the research questions outlined in Section 3, the methodology section will detail the systematic approach used in this study. This includes data collection methods, network analysis techniques, and the application of machine learning models.

4.1 Dataset

To conduct my study, I obtained data from GitHub repositories using the GitHub API due to its well-documented nature and accessibility for retrieving repository information and their stargazers [14]. Initially, I aimed to collect several thousand repositories; however, each repository typically has tens to tens of thousands of stargazers, which posed a significant challenge due to resource limitations for network analysis. Therefore, I limited my dataset to 50 repositories to manage the scale of stargazer networks effectively. For my dataset, I specifically chose 50 Rust repositories created after April 1, 2024. This selection criterion was made to balance the number of stars (refer Table 1) while ensuring community relevance and observing user influence within this community. This dataset allows us to analyze the network structure and predict the growth of stargazers from the inception of each repository.

Table 1: Summary statistics of the number of stars for selected repositories.

Metric	Min	Q1 (25th %)	Median (Q2)	Q3 (75th %)	Max
Number of stars	26.0	31.0	42.5	111.5	461.0

Subsequently, I accessed the @open-sauced/api.opensauced.pizza API to collect daily counts of new stargazers for each selected repository. Repositories created after April 28 were excluded from analysis due to insufficient consecutive daily data points (30 days). For days lacking new stargazer data, I inserted records with a date and a count of 0 to maintain a consistent time series across all repositories, crucial for training my LSTM model. Additionally, I identified the top 5 repositories based on total stargazers as representative samples for detailed analysis (see Table 2). Using the GitHub API's event endpoint, I chronologically sorted all stargazers for each repository and assigned them centrality measures derived from network analysis. Finally, for each repository and each day since its creation, I computed the average centrality measure based on the number of new stargazers. These calculations were incorporated into the final dataset used for training the LSTM model and analyzing centrality's impact on stargazer growth. For more details on the dataset, please refer to Table 3. In this study, I focused on repositories created after April 1, 2024, sourced using the GitHub API, and I collected daily stargazer counts using the @open-sauced/api.opensauced.pizza API.

Table 2: Top 5 Repositories Based on Stargazers Count

Full Name	Stargazers Count
AmrDeveloper/ClangQL	421
BenjaSOL/ore-cli-gpu	232
bluskript/nix-inspect	227
tonyke-bot/ore-miner	472
tsoding/good_training_language	270

4.2 Network Analysis

To analyze the structure and influence within the GitHub community, a stargazer network was constructed using the NetworkX library. Each stargazer is represented

Table 3: Dataset summary for selected repositories.

Full Name	Date	Days Since Creation	Stargazers Count	Average BC
AmrDeveloper/ClangQL	2024-04-05	0	2	0.000000
AmrDeveloper/ClangQL	2024-04-06	1	22	0.001326
AmrDeveloper/ClangQL	2024-04-07	2	79	0.001998
AmrDeveloper/ClangQL	2024-04-08	3	113	0.000545
AmrDeveloper/ClangQL	2024-04-09	4	75	0.002579

as a node in the network. Edges are formed between nodes if two stargazers have starred the same repository, indicating a mutual interest. The network was created as a MultiGraph to accommodate multiple edges between nodes when two stargazers have starred multiple repositories together, reflecting varying degrees of shared interest and interaction [15].

The dataset used for constructing the network consisted of the usernames of stargazers for 50 repositories. An example of the first 10 rows of this dataset is shown in Table 4, where each row represents a stargazer associated with a repository they have starred. The User ID provides a unique identifier for each GitHub user in the dataset. This dataset served as the basis for constructing the stargazer network, ensuring that each stargazer's interactions with repositories were captured to analyze their connectivity and influence within the GitHub ecosystem. While some studies have analyzed larger networks, such as the one by Kabakus Abdullah Talha et al. [13], which involved 84,737 developers and 209,100 repositories collected through the GitHub API, my study was constrained by limited resources and the environment in which it was conducted, specifically Google Colab. Therefore, I focused on a subset of data to perform analysis.

Table 4: Example of the dataset used for constructing the stargazer network.

Repository Name	Username	User ID
ore-miner	tommy-ca	140900186
ore-miner	Brucecarl	1773497
ore-miner	iczc	12002459
ore-miner	okeyzero	48344256
ore-miner	xiaoduzi1919	3840088
ore-miner	ConnorRepeat	41321054
ore-miner	lwsh123k	37106566
ore-miner	aa66609	13633514
ore-miner	meiwhu	15714433
ore-miner	xiomimin	108560806

Figure 2 visualizes the stargazer network. Nodes represent stargazers, and edges represent shared interest in the same repositories.

The resulting network contains 4,055 nodes and 464,139 edges. Based on the study by Bana and Arora [12] various types of networks were formed to analyze the influence

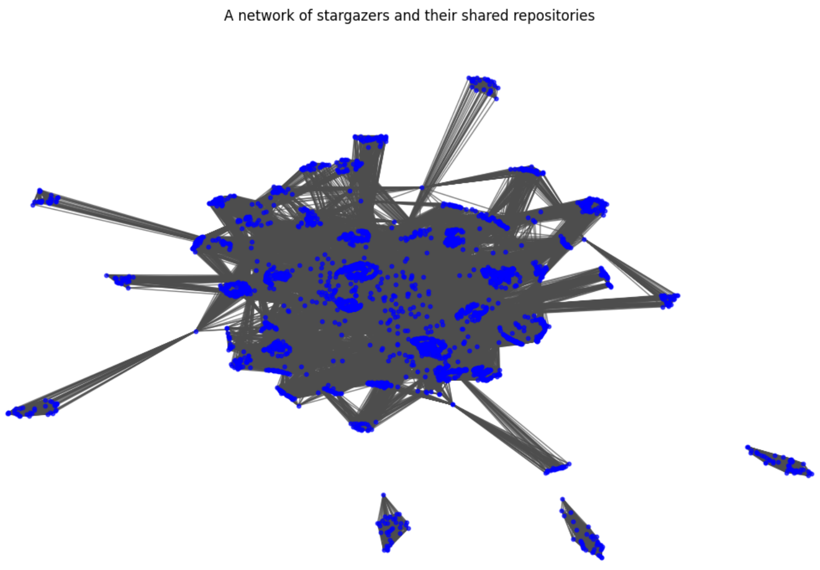


Fig. 2: Visualization of the stargazer network.

within the GitHub community. They constructed Developer-Developer Network (D-D) based on collaborations among developers and applied various social network measures to identify influential developers including Betweenness Centrality. Using betweenness centrality is essential for identifying important users, stargazers in GitHub networks. This measure helps pinpoint stargazers who act as vital connectors between different users and projects, facilitating effective communication and information sharing across the network. By assessing these bridging roles, betweenness centrality provides valuable insights into the influence and connectivity patterns among GitHub users [12] [15].

To calculate the betweenness centrality for each stargazer in the network, the `betweenness_centrality` function from the NetworkX library was used. The centrality values were then sorted in descending order, and the top 10 stargazers with the highest betweenness centrality scores were identified. The stargazers with the highest betweenness centrality in the network are shown in Table 5 and the distribution of the betweenness centrality measures is shown in Figure 3.

Username	Betweenness Centrality
IndieMinimalist	0.09795767482270298
antogerva	0.09574100429346141
Sandalots	0.057764893921729794
orociic	0.053645171380909164
appfromape	0.04999287353726945
vgecko	0.0376478644676185
kehoecj	0.022829343323723317
tkersey	0.01971285061860125
svercl	0.01735301475713779
ctsrc	0.015633507237782174

Table 5: Top 10 stargazers with the highest betweenness centrality.

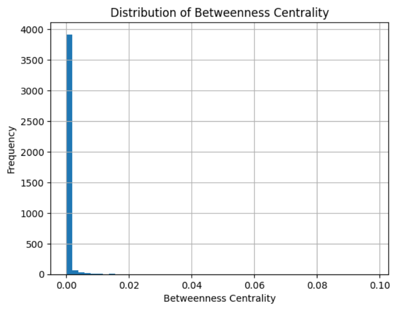


Fig. 3: Distribution of betweenness centrality measures.

4.3 Community Detection

Community detection is important for understanding social networks and how entities interact within them. It involves identifying groups or communities within a network where nodes (representing entities) are more connected to each other than to those outside their group. This process highlights the strong internal connections of these communities. Definitions of communities can vary based on different systems and applications [16].

NetworkX provides various functions and algorithms to help detect communities in different types of networks. These tools allow users to identify groups of nodes that are more connected to each other than to nodes outside their communities, making it easier to understand the network's structure.

One approach is the Girvan-Newman method, accessible via `girvan_newman(G)`, which iteratively removes edges with the highest betweenness centrality to reveal communities. This method is effective for identifying hierarchical community structures within complex networks.

For networks where modularity is a key metric, NetworkX offers `greedy_modularity_communities(G)` and `naive_greedy_modularity_communities(G)`. These algorithms maximize the modularity score, quantifying the strength of community structure based on the distribution of edges within and between communities.

Additionally, the Louvain Community Detection Algorithm, implemented in `louvain_communities(G)`, provides an efficient way to partition networks by optimizing modularity. It is particularly useful for large-scale networks due to its speed and scalability.

Other methods include label propagation algorithms (`asyn_lpa_communities(G)` and `fast_label_propagation_communities(G)`) and fluid communities algorithm (`asyn_fluidc(G, k)`), each offering distinct approaches to identifying communities based on node labels or fluid-like dynamics within the network [15].

The algorithm `community.greedy_modularity_communities(graph)` was used for community detection. A total of 17 communities were detected. Figure 4 visually represents the detected communities. Each node is colored according to its community membership identified by the algorithm. Nodes within the same community share similar colors, illustrating the cohesive groups identified by the community detection process.

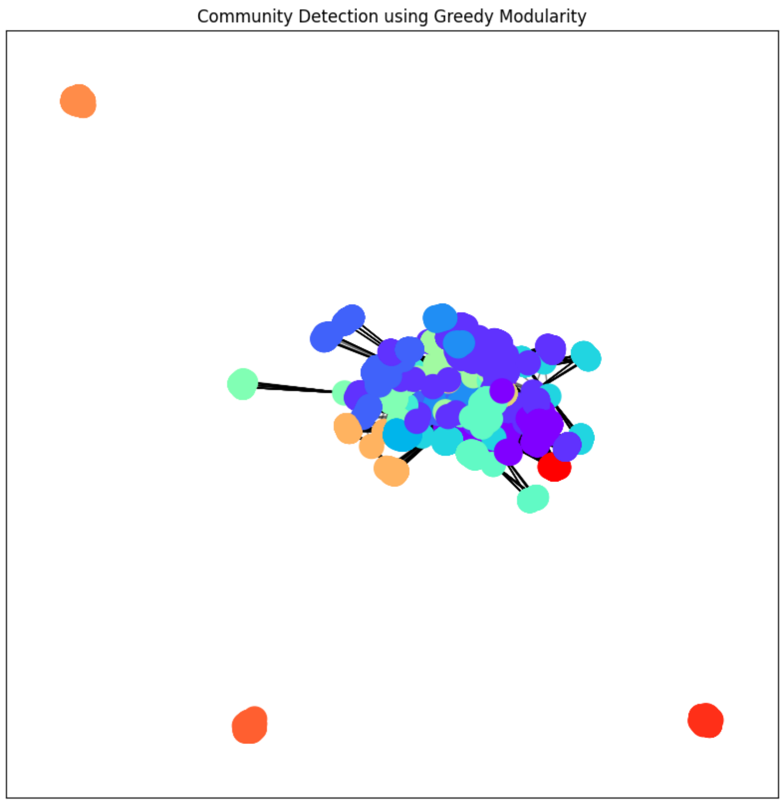


Fig. 4: Visualization of community detection using the Greedy Modularity algorithm

4.4 LSTM model

The prediction of the number of new stargazers was done through time series analysis. Time series is a sequence of well-defined data points taken at equally spaced intervals of time. Time series can be categorized into univariate time series and multivariate time series based on the number of features required for the analysis and prediction. Various models have been introduced for time series analysis, among which Recurrent Neural Networks (RNNs) gained wide acceptance. However, due to the vanishing gradient problem, a more efficient model, Long Short Term Memory (LSTM), was introduced to counteract the RNN problem. LSTM is a standard and well-known Recurrent Neural Network that can model long-term dependencies better than traditional RNNs, making them more preferable for time series analysis and prediction problems [2].

A general LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The three gates control the flow of information into and out of the cell, and the cell remembers values across arbitrary time intervals [17].

Previous studies have shown that LSTM models outperform traditional time series models like ARIMA [18]. It has proven successful in predicting cross-platform bursts of social media activity [19]. Additionally, LSTM models have been demonstrated to outperform models built with Support Vector Regression on both univariate and multivariate data for phone price prediction [3].

Given these advantages, LSTM was chosen for predicting the number of new stargazers on GitHub repositories.

4.4.1 Evaluation

Various classification methods have been employed to assess the prediction of online social network content popularity. Key metrics computed for evaluation include Mean Absolute Error (MAE), accuracy, precision, Mean Squared Error (MSE), recall, and F1-Score [17]. Varuna and Mohan [2] utilized Root Mean Square Error (RMSE) and the coefficient of determination (R2). RMSE provides an error value, while R2 indicates the correctness percentage, making the performance evaluation of their experiments efficient. RMSE is calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{1}$$

where Y_i are the observed values, $\hat{Y}_i$ are the predicted values, and n is the number of observations.

Sarhin et al. [20] employed MAPE, widely used for evaluating forecast model accuracy. MAPE is defined as follows:

$$\text{MAPE} = 100 \times \left(\frac{1}{n}\sum_{t=1}^{n}\frac{|Y_t - F_t|}{Y_t}\right) \tag{2}$$

where Y_t represents the observed values and F_t represents the forecasted values.

In this study, I will use Root Mean Square Error (RMSE), the coefficient of determination (R2), and Mean Absolute Percentage Error (MAPE) as evaluation metrics.

5 Discussion

The LSTM model was first trained on data from the top 5 repositories over a continuous 30-day period, using daily counts of new stargazers and average centrality measures for each day. The model's performance was evaluated using several metrics, including Root Mean Square Error (RMSE), R-squared (R2), and Mean Absolute Percentage Error (MAPE). Table 6 summarizes the model's performance metrics.

Metric	**Value**
RMSE	0.0025
R-squared (R2)	Undefined
MAPE	11465249636734.82%

Table 6: Performance Metrics of the LSTM Model

The obtained RMSE of 0.0025 suggests that, on average, the LSTM model predicts the number of new stargazers with high accuracy. However, the undefined R2 score indicates limitations in the model's ability to explain the variance in the data, prompting the need for further investigation. The unusually high MAPE of $1.146 \times 10^{12}\%$ highlights significant relative errors in the predictions.

The observed differences might be due to several factors, such as the relatively small dataset and the short 30-day period used to train the model. Previous studies employing LSTM for similar predictions utilized longer historical data, such as 180 days, to forecast trends over shorter periods [20].

A longer time period could not be used because my dataset consisted of newly created repositories. Additionally, targeting repositories with more stars and a longer time frame would increase the number of stargazers, expanding the network for analysis and betweenness centrality computation, which I lacked resources for.

6 Conclusion

In addressing Research Question 1: "How can influential stargazers be identified within the GitHub user network?", I successfully identified influential stargazers within the GitHub Rust stargazer network using centrality metrics such as Betweenness Centrality. This metric allows to pinpoint users who play crucial roles in the dissemination of information within the GitHub Rust community. By understanding their influence, community engagement and project promotion can be better strategized to enhance visibility and impact. This metric was calculated for every user in the network and used for prediction.

Research Question 2 focuses on effectively detecting communities within the GitHub stargazer network. This involves identifying cohesive groups of GitHub users (stargazers) who have stronger connections among themselves compared to connections with users outside their respective communities. Various network analysis techniques can be applied to achieve this, such as community detection algorithms like Girvan-Newman, Louvain Community Detection, and modularity-based methods. In this study, the community detection algorithm `community.greedy_modularity_communities(graph)` was employed, resulting in the identification of 17 communities within the GitHub stargazer network.

Research Question 3, ”Can centrality measures of stargazers be used to predict the growth in the number of new stargazers for GitHub projects?”. Whether centrality measures of stargazers could predict the growth in the number of new stargazers for GitHub projects remains uncertain due to the dataset’s insufficient size. The LSTM model requires an extended timeframe beyond 30 days for effective training and prediction.

For future work, it is strongly recommended to utilize a larger dataset with more repositories and stargazers. Additionally, utilizing a longer consecutive time period in LSTM models is crucial.

References

[1] Hudson Borges, M.T.V. Andre Hora: Predicting the Popularity of GitHub Repositories. Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering (2016)

[2] Varuna T V, A.M.: Trend Prediction of GitHub using Time Series Analysis. 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (2019)

[3] Chniti, B.H.Z.H. G.: E-commerce time series forecasting using lstm neural network and support vector regression. Int. Conf. on Big Data and Internet of Thing, 80–84 (2017)

[4] Chen, L.L.J.J..Z.L. F.: Predicting the Number of Forks for Open Source Software Project. Proceedings of the 2014 3rd International Workshop on Evidential Assessment of Software Technologies - EAST 2014. (2014)

[5] Fangwei Chen, J.J.L.Z. Lei Li: What Makes an Open Source Code Popular on Git Hub? 2014 IEEE International Conference on Data Mining Workshop (2014)

[6] Hudson Borges, M.T.V. Andre Hora: Understanding the Factors That Impact the Popularity of GitHub Repositories. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME) (2016)

[7] Hudson Borges, M.T.V.: What’s in a github star? understanding repository starring practices in a social coding platform. Journal of Systems and Software **146**, 112–129 (2018)

[8] Junxiao Han, X.X..W.J.Y. Shuiguang Deng: Characterization and Prediction of Popular Projects on GitHub. 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC) (2019)

[9] Babichev, S.A., Ries, J., Lvovsky, A.I.: Abduljaleel Al-Rubaye and Gita Sukthankar. Preprint at http://arxiv.org/abs/2011.04865 (2020)

[10] Mohammed Abdul Moid, M.F.A.A.O.A. Abdullah Siraj: Predicting Stars on

Open-Source GitHub Projects. 2021 Smart Technologies, Communication and Robotics (STCR) (2021)

[11] Yan Hua, S.W.Y.R.K.-K.R.C.b: User influence analysis for github developer social networks. Expert Systems with Applications **108**, 108–118 (2018)

[12] Bana Riyu, A.A.: Influence Indexing of Developers, Repositories, Technologies and Programming languages on Social Coding Community GitHub. 11th International Conference on Contemporary Computing (IC3) (2018)

[13] Kabakus, A.T.: Githubnet: Understanding the characteristics of github network **19**, 557–574 (2020)

[14] Github rest api documentation. https://docs.github.com/en/rest?apiVersion=2022-11-28

[15] Software for complex networks. https://networkx.org/documentation/stable/index.html#

[16] Cuijuan Wang, B.S.J.F.Y.W. Wenzhong Tang: Review on community detection algorithms in social networks (2015). 2015 IEEE International Conference on Progress in Informatics and Computing (PIC)

[17] G.Ananda, A.S.V.G. S. Srivastavaa: Recurrent neural networks in predicting the popularity of online social networks content: A review. ECS Trans. **107**, 19991 (2022)

[18] Neda Hajiakhoond Bidoki, H.K. Gita Sukthankar, Garibay, I.: A Cross-Repository Model for Predicting Popularity in GitHub (2018)

[19] Gita, S.: An lstm model for predicting cross-platform bursts of social media activity. Information **10**, 394 (2019)

[20] Sefa Eren Sahin, A.T. Kubilay Karpat: Predicting popularity of open source projects using recurrent neural networks (2019). 5th IFIP International Conference on Open Source Systems (OSS)