BotFinder: A Novel Framework for Social Bots Detection in Online Social Networks Based on Graph Embedding and Community Detection

Shudong Li (lishudong@gzhu.edu.cn)
Guangzhou University
Chuanyu Zhao
Guangzhou University
Qing Li
Shandong Jianzhu University
Jiuming Huang
Hunan Singhand Intelligent Data Technology Co., Ltd
Dawei Zhao
Qilu University of Technology (Shandong Academy of Sciences)
Peican Zhu
Northwestern Polytechnical University

Research Article

Keywords: Online Social Networks, Social Bots, Feature Engineering, Community Detection, Graph Embedding

Posted Date: July 28th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1871702/v1

BotFinder: A Novel Framework for Social Bots Detection in Online Social Networks Based on Graph Embedding and Community Detection

Shudong Li[1,2*#], Chuanyu Zhao[1#], Qing Li[3*], Jiuming Huang[4], Dawei Zhao[5], Peican Zhu[6]

1. Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, 510006, China.
2. Peng cheng Laboratory, Shenzhen, 518000, China.
3. Shandong Jianzhu University, Jinan, 250101, China.
4. Hunan Singhand Intelligent Data Technology Co., Ltd, Changsha, China
5. Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China.
6. School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an 710072, China.

# Shudong Li and Chuanyu Zhao contributed equally to this work.

*Corresponding authors: Shudong Li (lishudong@gzhu.edu.cn), Qing Li (147797877@qq.com).

Abstract

With the widespread popularity of online social networks (OSNs), the number of users has also increased exponentially in recent years. At the same time, Social bots, i.e. accounts that controlled by program, are also on the rise. Service providers of OSNs often use them to keep social networks active. Meanwhile, some social bots are also registered for malicious purposes. It is necessary to detect these malicious social bots to present a real public opinion environment. We propose BotFinder, a framework to detect malicious social bots in OSNs. Specifically, it combines machine learning and graph methods so that the potential features of social bots can be effectively extracted. Regarding the feature engineering, we generate second order features and use coding methods to encode variables that have high cardinality. These features make full use of both labelled and unlabeled samples. With respect to the graphs, we firstly generate node vectors through embedding method, following which the similarity between vectors of humans and bots can be further calculated; Then, we use an unsupervised method to diffuse labels and thus the performance can be improved again. To valid the performance of the proposed method, we conduct extensive experiments on the dataset provided by an artificial intelligence contest which is composed of over eight million records of users. Results show that our approach reaches a F1-score of 0.8850, which is much better compared to the state of the art.

Keywords—Online Social Networks, Social Bots, Feature Engineering, Community Detection, Graph Embedding.

1 Introduction

Over the past few decades, Online Social Networks (OSNs) have played an increasingly important role in our daily life, by the aid of which human beings can communicate with each other in real time and maintain their social relationships more conveniently and efficiently. There exist many world-wide popular social platforms that connect users all over the globe, such as Facebook, Twitter and etc. Furthermore, OSNs have also become the most popular channel for individuals to obtain social news compared with traditional medias, such as newspaper.

Social bots, i.e., accounts controlled by program may be used for keeping social networks active. Although there are beneficial social bots in OSNs, the emergence of some malicious social bots has harmful effect. For example, some people can register a large number of accounts for various purposes, such as increasing the number of fans or likes maliciously. These malicious behaviors have become an important information security problem which threatens the healthy development of social network platforms [1-2]. Therefore, it is necessary to detect those malicious social bots, which are also referred to as social bots detection. In particular, majority of current studies deal with Twitter and other foreign platforms, whereas few studies are conducted to investigate OSNs in China.

Hence, various scholars devote to study the problem of social bots detection. The current works related to social bots detection are mainly divided into two categories, i.e. machine learning approaches and graph-based approaches. However, there still exists some challenges for this topic:

1) In general, most methods rely on a single algorithm to identify social bots, which might not be desirable options due to the diversity of the dataset.

2) In practice, most of the data is unlabeled, which indicates that the numbers of labels are usually very small. Hence, it is a great challenge to effectively exploit the unlabeled data.

Aiming to tackle the above challenges, we here consider the users' profiles, behaviors and relationships among them jointly. Furthermore, we proposed an integrated mechanism BotFinder through combining feature engineering and graph methods to detect social bots. Firstly, feature engineering is conducted on the dataset to extract global information. Then, we generate node vectors through embedding methods. After that, we calculate the similarity between the vectors for humans and bots. Finally, we adopt unsupervised method (here, community detection algorithm is considered) in order to further improve the performance. With the proposed algorithms, we can easily detect those machine accounts.

The contributions of this paper are summarized as follows.

1) Firstly, graph algorithm may not perform well when there are many isolated nodes. Whereas machine learning method is suffering from the incapability of learning topological structure. Hence, we combine machine learning method and graph approach to overcome these problems.

2) Secondly, in feature engineering, we try to obtain second order features and adopt coding methods to encode variables that have high cardinality, or in other words, that contain a large number of distinct values. In terms of graphs, we generate node vectors through embeddings methods. Then, we exploit unsupervised method to diffuse labels to improve the performance. These approaches make full use of both labelled and unlabeled samples.

The rest of this paper is organized as follows. In Section 2, we review some related works. In Section 3, we present the proposed framework BotFinder. Then, in Section 4, we describe the studied dataset in detail and experiments are conducted with sufficient analysis. Eventually, we conclude our research in Section 5.

2 Related works

In this section, we review the recent research on social bots detection in OSN: machine learning approaches and graph-based approaches.

2.1 machine learning approaches

Among the machine learning approaches, supervised ones are widely investigated. Early anti-cheating algorithms only utilize user profiles or user behaviors to build models. Breno et al. [3] proposed a methodology using Artificial neural networks with data preprocessing and mining. Chang et al. [4] proposed a feature selection method followed by decision trees to detect bots. Ganji et al. [5] applied K-

nearest Neighbors (KNN) in credit card fraud detection. Ferrara et al. [6-7] utilized machine learning and cognitive behavioral modeling techniques to analyze social bots in 2017 French presidential election and 2017 Catalan referendum for independence. Denis et al. [8] proposed an ensemble learning method for detecting bots on Twitter.

With the development of deep learning method (LSTM, CNN, etc.), researchers also try to develop new methods in order to detect social bots aiming to further improve the detecting accuracy. Through viewing user content as temporal text data, Cai et al. [9] proposed BeDM method for bot detection. Kudugunta et al. [10] extracted user metadata and tweet text and these data are regarded as the inputs to the LSTM deep nets. In practice, most of the real-world data is unlabeled, while unsupervised learning methods are widely investigated, which usually relies on the common feature of social bots. Cresci et al. [11-12] proposed a revised approach based on DNA-inspired techniques in order to model online user behavior. Chen et al. [13] proposed an unsupervised approach to detect Twitter spam campaigns in real-time. Jiang et al. [14] proposed CATCHSYNC to detect suspicious nodes using only the topology without label. Su et al. [15] proposed IoT-RU. Mazza et al. [16] converted the retweet time series into feature vectors and then cluster.

2.2 Graph-based approaches

Machine learning approaches only consider the features of nodes. Whereas, the relationships among nodes also contain valuable and useful information. With the development of deep learning and graph algorithm, topology information of graphs is necessary to be considered for further improvement.

Social bots have the characteristics of aggregation in graph. While community detection is used to discover community structures in network, which can also be viewed as a generalized clustering algorithm. Thus, community detection algorithms might be applicable to detect social bots. Many researchers have devoted endless efforts to the study of this topic. Guillaume et al. [17] proposed a heuristic method based on modularity optimization. Li et al. [18] proposed WCD algorithm based on a deep sparse autoencoder. For samples with rich features, it is hard to fully mine the information existing in the features. Then, new methods are proposed which first convert the topology information of nodes into feature vector, and then use machine learning algorithms to train and infer. For instance, Pytorch-BigGraph proposed by Lerer et al. [19], NetWalk proposed by Yu et al. [20], Node2Vec proposed by Grover et al. [21] and Bot2Vec proposed by Pham et al. [22]. Moreover, Kipf et al. [23] proposed Graph Convolutional Networks (GCN) which models the features of nodes and network topology, and Aljohani et al. [24] apply GCN to detect bots on Twitter. Li et al. [25] proposed BPD-DMP algorithm for network immunization. Nie et al. [26] considered the social network and posted content; then, they proposed DCIM algorithm. Gao et al. [27] characterized dynamic behaviors and proposed a network-based model. Zhu et al. [28] investigated the epidemic spreading process on multi-layer networks. Su et al. [29] proposed IDES to detect malicious nodes in the vehicular network.

Most methods rely on a single algorithm to identify social bots. In terms of both accuracy and other relevant evaluation metrics, the previous identification methods still have significant limitations.

3 Our Proposed Method: BotFinder

In this section, we mainly illustrate BotFinder which mainly consists of three steps: 1) we represent feature engineering techniques on tabular data; 2) we derive node embeddings, and then measure the similarity between humans and bots; 3) we applied community detection algorithm to further improve performance.

3.1 Overview

Figure 1 illustrates the steps in detail. Step1, we exploit feature engineering techniques to generate feature matrix. Step2, we use graph embedding method to generate similarity matrix, and then merge these two matrixes. After that, we adopt LightGBM [30] to train the merged matrix and infer temporary results. Step3, we apply community detection method to generate partial results, and use these results to correct the results of LightGBM.

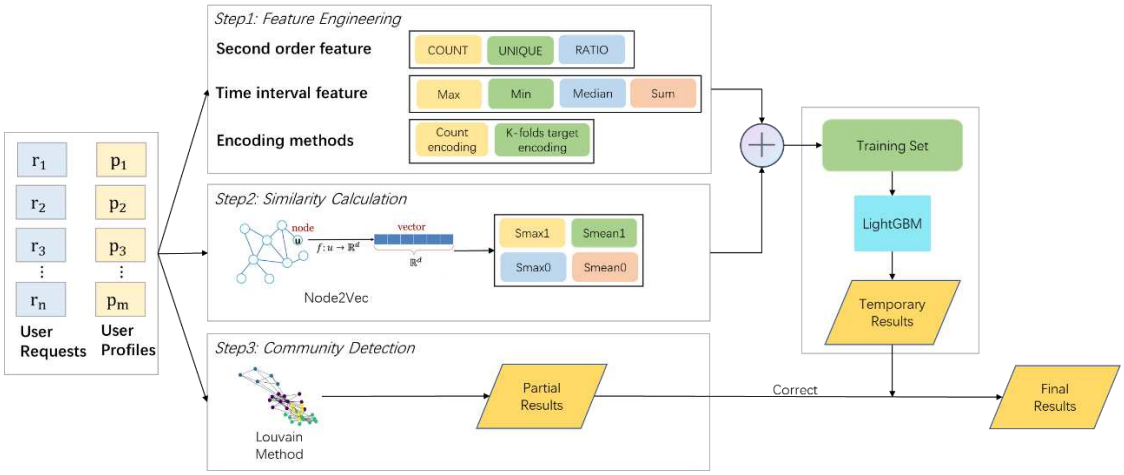

FIGURE 1. The Framework of BotFinder

3.2 Step1: Feature Engineering

Here, we try to obtain the second order features, time interval feature, count encoding and k-folds target encoding. Then, we apply the LightGBM to train the obtained features and infer temporary results.

Second order feature: To represent combinations of categorical variable in table, we assume the second order feature is represented as $(COUNT, NUNIQUE, RATIO)$.

Here, $COUNT$ reflects the degree of activity. Specifically, we select a pair of variables (i.e., V_1 and V_2) and we are anticipated to record the number of times this pair occurs in dataset. We abbreviate it to $COUNT(V_1, V_2)$. For example, a user gives a thumb-up to someone using the combination of device type (V_1) iPhone12,1 and app version (V_2) 126.7.0, and this combination appears $\boldsymbol{k}$ times in the dataset. Then, the users who use iPhone12,1 and 126.7.0 will get a $COUNT$ value of $\boldsymbol{k}$.

While $NUNIQUE$ indicates the diversity in a given extent. We use a variable (V_1) as the primary key, and record number of unique categories in the other variable (V_2). We abbreviate it to $NUNIQUE(V_1)[V_2]$. For example, for the users who use device type (V_1) iPhone12,1, there are $\boldsymbol{k}$ different app versions in the dataset. Then, the users who use iPhone12,1 will get a $NUNIQUE$ value of $\boldsymbol{k}$.

$RATIO$ describes the proportion of count. It is calculated as $COUNT(V_1, V_2)/COUNT(V_1)$. For example, the combination of device type (V_1) iPhone12,1 and app version (V_2) 126.7.0 appears $\boldsymbol{k}$ times, and device type (V_1) iPhone12 appears for $\boldsymbol{v}$ times in the dataset. Then, all the users who use iPhone12,1 and 126.7.0 will get a $RATIO$ value of $\boldsymbol{k/v}$.

Time interval feature: The request time interval varies for different user. Here, we mainly consider max, min, median and sum of the time interval.

Count encoding: Count encoding is conducted through replacing categories with their counts computed on the dataset. However, count may be the same for some variables, which may result in the collision that two categories might be encoded as the same value. This will lead to a degradation in the performance of model. Hence, we here introduce a target encoding technique.

K-folds target encoding (or likelihood encoding, impact encoding, mean encoding): Target encoding is numeration of categorical variables via target (label). Here, we replace each category of the categorical variable with corresponding probability of the target. To reduce target leak, we apply k-folds target encoding. This is implemented as follows: (a) Split the training data into 10-folds. (b) Regard the mean of the folds #2-10 target as the coding value of the fold #1, and calculate the coding value of #2~#10 similarly. (c) Use the target of training data to determine the coding value of testing data.

3.3 Step2: Similarity Calculation

Here, we adopt the Node2vec [21] to obtain the node embeddings (vectors) of users, and then calculate the cosine similarity of embeddings between a user and a labeled one. The similarity value indicates the probability of having the same label for the two users; for example, if the cosine similarity between user1 and user2 is relatively large, then they are likely to have the same label with a high probability.

For example, $\boldsymbol{A}$ and $\boldsymbol{B}$ denotes two node vectors for users/accounts. The cosine similarity between two vectors is calculated as

$$S(\boldsymbol{A},\boldsymbol{B}) = \frac{\boldsymbol{A}\cdot\boldsymbol{B}}{|\boldsymbol{A}||\boldsymbol{B}|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

where A_i and B_i denote the element of vector $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively.

Then, for each node vector $\boldsymbol{C}$ in training set and testing set, we calculate it's max and mean cosine similarity between bots and humans, which is represented as a vector $[Smax1, Smean1, Smax0, Smean0]$ as follows:

$$Smax1 = max(S(\boldsymbol{C},\boldsymbol{D_i})), \boldsymbol{D_i} \in bots \ \ and \ \ \boldsymbol{C} \neq \boldsymbol{D_i} \tag{2}$$

$$Smean1 = mean(S(\boldsymbol{C},\boldsymbol{D_i})), \boldsymbol{D_i} \in bots \ \ and \ \ \boldsymbol{C} \neq \boldsymbol{D_i} \tag{3}$$

$$Smax0 = max(S(\boldsymbol{C},\boldsymbol{E_i})), \boldsymbol{E_i} \in humans \ \ and \ \ \boldsymbol{C} \neq \boldsymbol{E_i} \tag{4}$$

$$Smean0 = mean(S(\boldsymbol{C},\boldsymbol{E_i})), \boldsymbol{E_i} \in humans \ \ and \ \ \boldsymbol{C} \neq \boldsymbol{E_i} \tag{5}$$

where $\boldsymbol{D_i}$ and $\boldsymbol{E_i}$ represents a node vector.

The process is illustrated as follows:

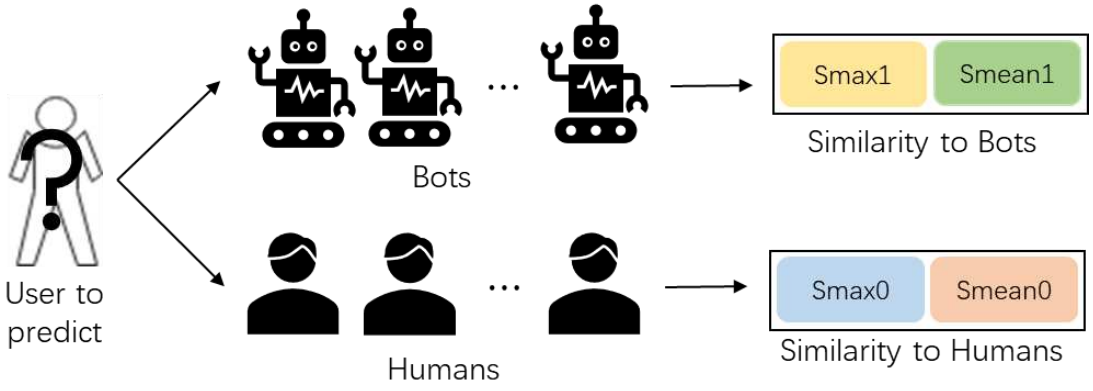

FIGURE 2. Presentation of The Similarity Calculation Process

3.4 Step3: community detection

For community detection, we adopt the typical Louvain Method [17] which divides the constructed graph into communities. After that, we will label communities with rules as follows:

1) All users in the community are supposed to be of the same label if the users with labels belongs to the same community.

2) If the users in a community do not have any label, or if the users are of different labels, we will not make prediction.

However, prediction may not cover all users. So, performance in this rule is limited. But the result of this rule is more accurate than LightGBM. Through combining the above two steps, performance can be further improved.

4 Experiment

In order to evaluate the performance of the proposed mechanism, we collect a dataset from an artificial intelligence contest (https://security.bytedance.com/fe/ai-challenge#/sec-project?id=2&active=1). It contains over eight million records consisting of user profiles and user requests (follow or like someone). Basic information of the dataset is shown in Table 1 and Table 2: Table 1 shows the users' personal information (profile), while Table 2 illustrates the users' behavior (request), including the device and the app version used to initiate the request at that time.

The task is described as follows:

Given user profiles and their requests. Only a small percentage of users are labeled. Hence, we have to build a reasonable, explanatory and effective model to detect malicious bots from users.

Variable name	Sample	Description
user	653fb3fab452eeb99711c27b0c98811d	unique user id
user_name	Q2Fsb882bbacf6ccf335a4=	mask
user_profile	8f00b204e9800998	mask
user_register_time	40003200	mask
user_register_type	0	mask
user_register_app	16	mask
user_least_login_time	132969600	mask
user_least_login_app	16	mask
user_fans_num	30	
user_follow_num	3550	
user_post_num	20	
user_post_like_num	50	
user_freq_ip	447.509.456.453	mask

TABLE 1. User Profiles

Variable name	Sample	Description
request_id	e4c5045ef7382cbd3bdb0381130fc95c	unique request id
request_time	4153460	mask
user	1b004c3c48ee0f2cc3e89a6e2d5ebe9a	unique user id
Request_device_id	f38f442e41bc96c7293bb211021f3177	unique device id
request_ip	533.541.443.535	mask
request_target	c0772f5bb13fc57824aa69376e8a31fb	follow/like
request_device_type	iPhone12,1	
request_app_version	126.7.0	
request_app_channel	App Store	

TABLE 2. User Requests

Then, we perform a simple statistical analysis in Table 3. Request with label 1 indicates that this request is blocked and corresponding user is a bot. We find that the number of bots is significantly less than the number of humans.

Types	Number
Total users	4417182

User with label 0	99565
User with label 1	9905
Total request	8227327
Request with label 0	167782
Request with label 1	243890
Users in testing data	151117

TABLE 3. Label Distribution

4.1. Evaluation Metric

To evaluate the performance, we need to take Recall and Precision into consideration comprehensively, while we are anticipated to excavate bots as many as possible (to improve Recall). Meanwhile, we are supposed to make accurate prediction without harming normal users (to improve Precision). Hence, the traditional F1 score is adopted as the evaluation metric.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recal}} \tag{6}$$

4.2 Data processing

In this section, we describe the data processing in detail. Overall, we merge user profiles matrix to user requests matrix on variable 'user', and make prediction for user requests. If any request is predicted as 1, then this user is labeled as 1.

For Feature Engineering, the process on different variables is presented in Table 4. After the operation, we obtain the feature engineering matrix.

Dataset	Applied method	Variables
User requests	Second order feature	[user, request_device_id, request_ip, request_target, request_device_type, request_app_version, request_app_channel]
	Count encoding	[user,request_device_id, request_ip, request_target, request_device_type, request_app_version, request_app_channel]
	10-folds target encoding	[request_device_type, request_app_version, request_app_channel]
User profiles	Second order feature	[user_name, user_profile, user_register_type, user_register_app, user_least_login_app, user_freq_ip]
	Count encoding	[user_name, user_profile, user_freq_ip]

TABLE 4. The Process in Step 1

For Similarity Calculation, we exploit 3 kinds of relationships to derive the graphs; and then we calculate similarity. We obtain graphs by using 'request_ip, 'request_device_id' and 'request_target' relationships. For example, there will be an edge between two users if they use the same IP, or follow the same target, or share the same device.

1) Constructing the graph by using 'request_ip': Here, we analyze the number of users associated with each IP while corresponding results are provided in Table 5. We find that for most scenarios, each IP is associated with only one user. However, the constructed graph may not cover all users.

Percentiles	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	Max
Number	1	1	1	1	1	1	1	1	2	2	320000

TABLE 5. Original Percentiles

Hence, we are anticipated to exclude the IP with only 1 user. Furthermore, there also exists some IPs with more than 1000 users which may be public IPs. The existence of such public IPs will lead to a large number of edges in the constructed graph, which weaken the association between users. Hence, we also exclude the IPs with more than 1000 users for simplicity. Table 6 shows the quantile of the number of users associated with each IP after exclusion. There are 26,836,730 edges and 1,953,559 nodes, accounting for 44.23% of all users.

Percentiles	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	Max
Number	2	2	2	2	2	2	2	3	3	5	820

TABLE 6. Percentiles After Exclusion

2) Constructing graph by using 'request_device_id': This correlation is much stronger than IP; however, corresponding graph is much smaller. For the constructed graph, there exists only 17,193 nodes and 5,432,432 edges.

3) Constructing graph by using 'request_target': If we use the full data, there will be 60,544,191 edges and 2408,814 nodes; however, this will incur the requirement of high computing resources to address such data. Similar to IP association, if a target is liked/followed by lots of users, this indicates the target might be a celebrity. Therefore, we exclude targets that are associated with more than 20 users' likes. After the operation, we obtained a graph with 5121,010 edges and 1317,012 nodes.

Figure 3 shows the number of edges and nodes in different graphs. The number of edges in IP Graph is much higher than that of the other two graphs.

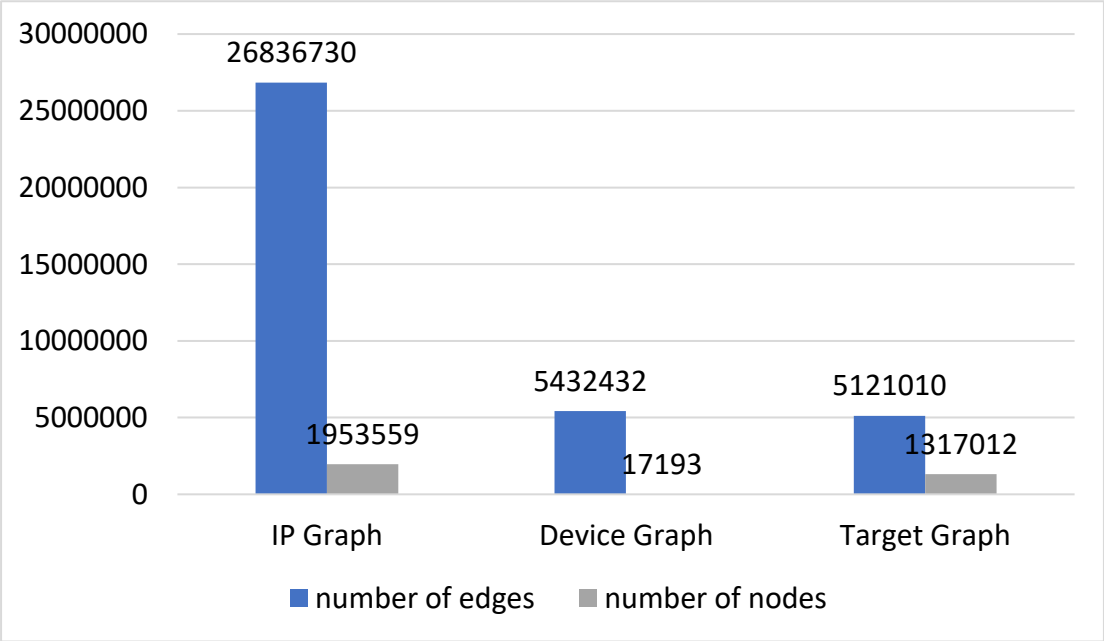

FIGURE 3. Number of Edges and Nodes

Figure 4 represents our graphing solution. We generate a vector $[Smax1, Smean1, Smax0, Smean0]$ by using 'request_ip, 'request_device_id' and 'request_target' relationships, namely, merging these 3 kinds of edges. Similarly, we utilize 'request_ip' and 'request_device_id' relationships to do the same process and eventually obtain $[\_Smax1, \_Smean1, \_Smax0, \_Smean0]$.

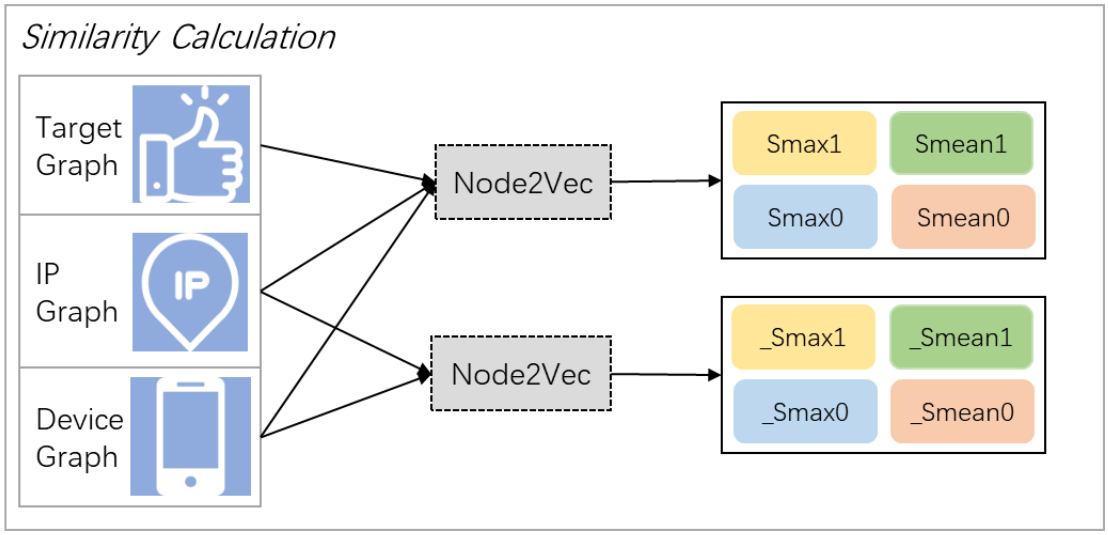

FIGURE 4. Similarity Calculation

For Community Detection, we only apply the community detection algorithm to address the device graph obtained in the above step.

4.3 Result analysis

In order to valid the superiority of the proposed model, extensive experiments are conducted on the considered data. For comparison, we also implement several baseline models using raw data, including Decision Tree [31], AdaBoost [32], XGBoost [33], Random Forest [34], CatBoost [35] and LightGBM [30]. Corresponding results are provided in Table 7; as indicated, BotFinder is of the best performance indicated by the largest F1-score.

Implemented Models	F1 score
Decision Tree	0.7031
AdaBoost	0.7616
XGBoost	0.8002
Random Forest	0.8187
CatBoost	0.8292
LightGBM	0.8398
BotFinder	**0.8850**

TABLE 7. Results of the Classification Model

As presented in Figure 5, results are provided to verify the validity of different steps indicated by the obtained F1-score. As indicated, we can find that the F1-score can be improved by a large extent with the consideration of step 2; while step 3 can only slightly improve the F1-score.

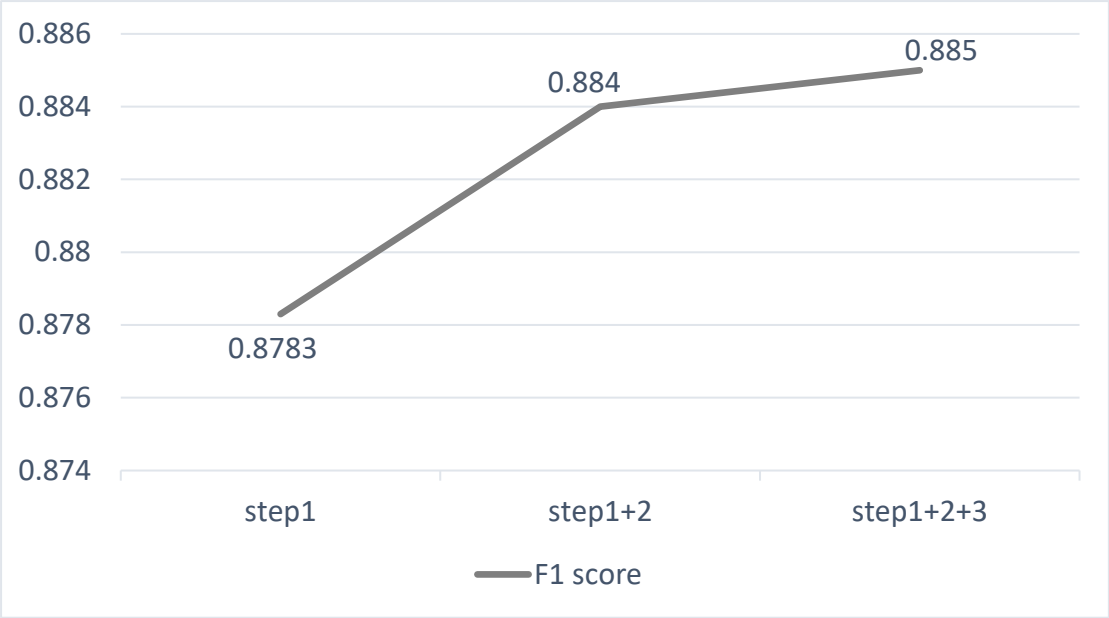

FIGURE 5. F1 Score Obtained for the Testing Set

Figure 6 shows the feature importance generated in step1 and step2. The score of target encoding of device types seems to be high, indicating that social bots tend to use fixed types of device. Furthermore, the personal information of users such as login time and register time also has a high score.

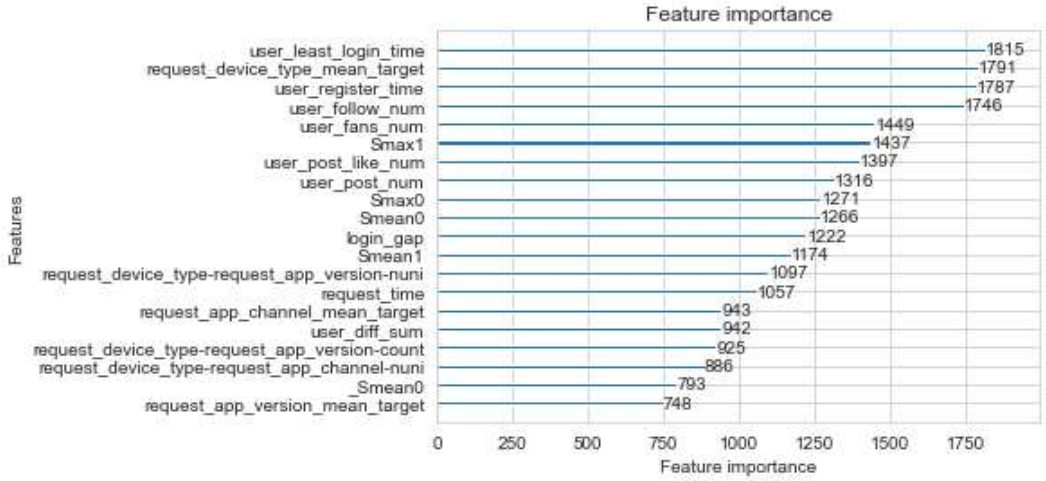

FIGURE 6. Feature Importance

To visually show the differentiation between positive and negative samples in different variables, we apply Kernel Density Estimation (KDE) In Figure 7. We find that social bots show the characteristics of aggregation, that is, they are tend to register or login at a fixed time.

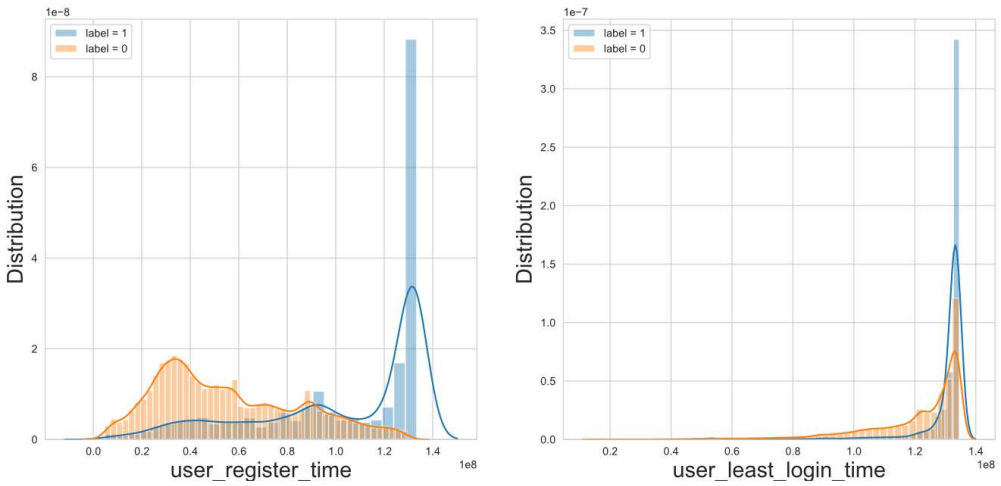

FIGURE 7. Feature Histogram in User Side

For variable 'request_time' in Figure 8, In addition to the characteristics of aggregation, we also find that the request time of bots shows obvious periodicity, that is, bots are programmed regularly.

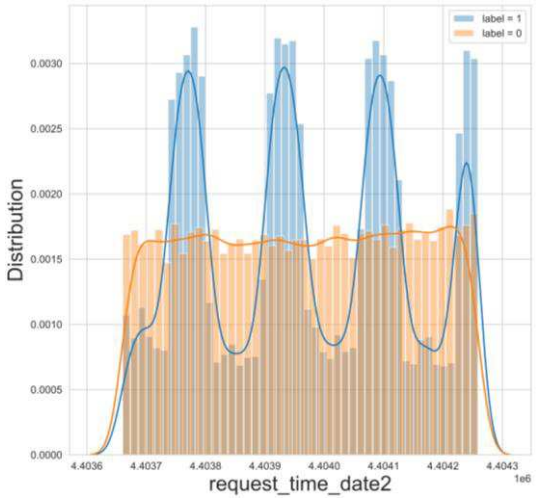

FIGURE 8. Feature Histogram in Request Side

For second order features in Figure 9 and similarity features in Figure 10, there also exists obvious differentiation between the positive and negative samples. We find that some bots may use special devices, resulting in a low $NUNIQUE$ and high $RATIO$ in app_version.

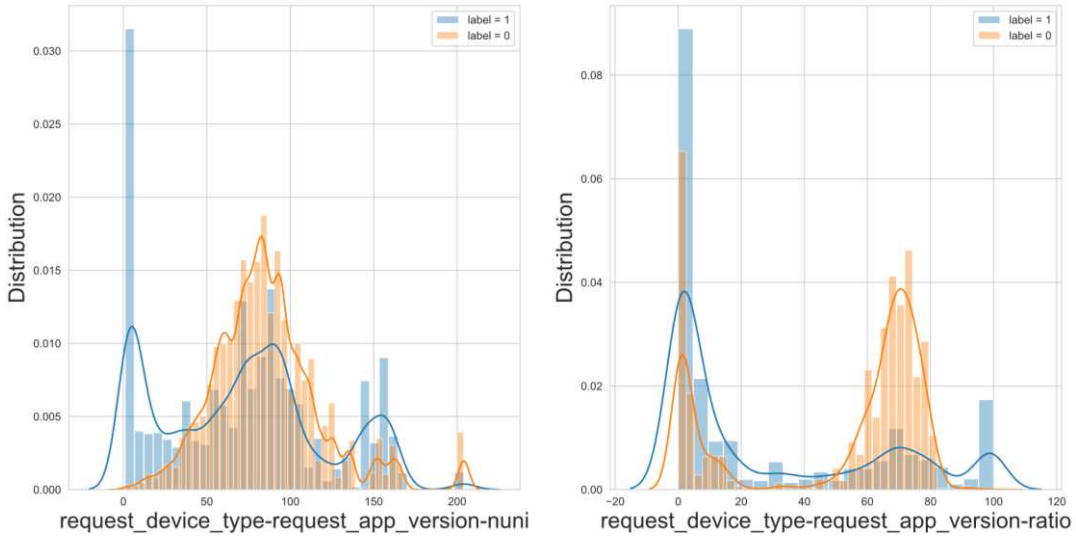

FIGURE 9. Second Order Feature Histogram

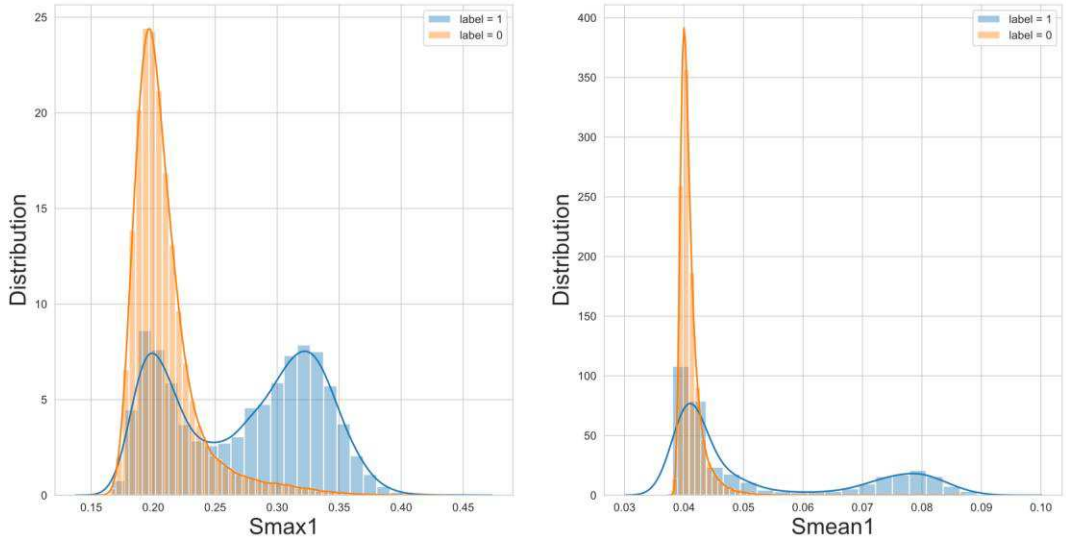

FIGURE 10. Similarity Feature Histogram

5. Conclusions

We propose a social bots detection method, BotFinder, in this paper. In order to valid the performance of the developed approach, we collected a dataset with more than eight million records of users. Meanwhile, machine learning and graph methods are applied to extract potential features of social bots from such dataset. In particular, for feature engineering, we generate second order features and use coding methods to encode high-cardinal variables. In terms of graphs, we generated node vectors for accounts and then exploit unsupervised method (here we utilize community detection) to diffuse labels in order to further improve the performance. Through experiments conducted on the collected dataset, the effectiveness of the proposed integrated mechanism is guaranteed by a relatively large F1-score of 0.8850. The performance is super compared with existing methods.

Declarations

- Ethical Approval and Consent to participate

Not applicable

- Human and Animal Ethics

Not applicable

- Consent for publication

Not applicable.

- Availability of supporting data

Not applicable

- Competing interests

The authors declare that they have no competing interests.

- Funding

This research was funded by NSFC (Grant Nos. U1803263, 62072131, 62073263), Science and Technology Projects in Guangzhou (No.202206030001, 202102010442), Guangdong Basic and Applied Basic Research Foundation (No.2022A1515011401), the Major Key Project of PCL (Grant No. PCL2021A09, PCL2021A02, PCL2022A03), Guangdong Higher Education Innovation Group (Grant No.2020KCXTD007) and Guangzhou Higher Education Innovation Group (Grant No.202032854).

- Authors' contributions

Conception and design of study: Shudong Li and Chuanyu Zhao

data processing: Qing Li and Jiuming Huang

analysis of experimental result: Dawei Zhao

manuscript revision: Peican Zhu

- Acknowledgment

This research was funded by NSFC (Grant Nos. U1803263, 62072131, 62073263), Science and Technology Projects in Guangzhou (No.202206030001, 202102010442), Guangdong Basic and Applied Basic Research Foundation (No.2022A1515011401), the Major Key Project of PCL (Grant No. PCL2021A09, PCL2021A02, PCL2022A03), Guangdong Higher Education Innovation Group (Grant No.2020KCXTD007) and Guangzhou Higher Education Innovation Group (Grant No.202032854).

- Authors' information

Shudong Li[1,2*#], Chuanyu Zhao[1#], Qing Li[3*], Jiuming Huang[4], Dawei Zhao[5], Peican Zhu[6]

1. Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, 510006, China.
2. Peng cheng Laboratory, Shenzhen, 518000, China.
3. Shandong Jianzhu University, Jinan, 250101, China.
4. Hunan Singhand Intelligent Data Technology Co., Ltd, Changsha, China
5. Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China.
6. School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an 710072, China.

# Shudong Li and Chuanyu Zhao contributed equally to this work.

*Corresponding authors: Shudong Li (lishudong@gzhu.edu.cn), Qing Li (147797877@qq.com).

Shudong Li (lishudong@gzhu.edu.cn)

Chuanyu Zhao (chuanyu0317@163.com)

Qing Li (147797877@qq.com)

Jiuming Huang (hjm@singhand.com)

Dawei Zhao (zhaodw@sdas.org)

Peican Zhu (ericcan@nwpu.edu.cn)

References

[1] Yang F, Liu Y, Yu X, et al. Automatic detection of rumor on sina weibo[C]//Proceedings of the ACM SIGKDD workshop on mining data semantics. 2012: 1-7.

[2] Bessi A, Ferrara E. Social bots distort the 2016 US Presidential election online discussion[J]. First monday, 2016, 21(11-7).

[3] Costa B C, Alberto B L A, Portela A M, et al. Fraud detection in electric power distribution networks using an ann-based knowledge-discovery process[J]. International Journal of Artificial Intelligence & Applications, 2013, 4(6): 17.

[4] Chang W H, Chang J S. An effective early fraud detection method for online auctions[J]. Electronic Commerce Research and Applications, 2012, 11(4): 346-360.

[5] Ganji V R, Mannem S N P. Credit card fraud detection using anti-k nearest neighbor algorithm[J]. International Journal on Computer Science and Engineering, 2012, 4(6): 1035-1039.

[6] Ferrara E. Disinformation and social bot operations in the run up to the 2017 French presidential election[J]. arXiv preprint arXiv:1707.00086, 2017.

[7] Stella M, Ferrara E, De Domenico M. Bots increase exposure to negative and inflammatory content in online social systems[J]. Proceedings of the National Academy of Sciences, 2018, 115(49): 12435-12440.

[8] Stukal D, Sanovich S, Bonneau R, et al. Detecting bots on Russian political Twitter[J]. Big data, 2017, 5(4): 310-324.

[9] Cai C, Li L, Zengi D. Behavior enhanced deep bot detection in social media[C]//2017 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE, 2017: 128-130.

[10] Kudugunta S, Ferrara E. Deep neural networks for bot detection[J]. Information Sciences, 2018, 467: 312-322.

[11] Cresci S, Di Pietro R, Petrocchi M, et al. DNA-inspired online behavioral modeling and its application to spambot detection[J]. IEEE Intelligent Systems, 2016, 31(5): 58-64.

[12] Cresci S, Di Pietro R, Petrocchi M, et al. Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling[J]. IEEE Transactions on Dependable and Secure Computing, 2017, 15(4): 561-576.

[13] Chen Z, Subramanian D. An unsupervised approach to detect spam campaigns that use botnets on twitter[J]. arXiv preprint arXiv:1804.05232, 2018.

[14] Jiang M, Cui P, Beutel A, et al. Catching synchronized behaviors in large networks: A graph mining approach[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2016, 10(4): 1-27.

[15] Su S, Tian Z, Liang S, et al. A reputation management scheme for efficient malicious vehicle identification over 5G networks[J]. IEEE Wireless Communications, 2020, 27(3): 46-52.

[16] Mazza M, Cresci S, Avvenuti M, et al. Rtbust: Exploiting temporal patterns for botnet detection on twitter[C]//Proceedings of the 10th ACM Conference on Web Science. 2019: 183-192.

[17] Guillaume L. Fast unfolding of communities in large networks[J]. Journal Statistical Mechanics: Theory and Experiment, 2008, 10: P1008.

[18] Li S, Jiang L, Wu X, et al. A weighted network community detection algorithm based on deep learning[J]. Applied Mathematics and Computation, 2021, 401: 126012.

[19] Lerer A, Wu L, Shen J, et al. Pytorch-biggraph: A large-scale graph embedding system[J]. arXiv preprint arXiv:1903.12287, 2019.

[20] Yu W, Cheng W, Aggarwal C C, et al. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 2672-2681.

[21] Grover A, Leskovec J. node2vec: Scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

[22] Pham P, Nguyen L T T, Vo B, et al. Bot2Vec: a general approach of intra-community oriented representation learning for bot detection in different types of social networks[J]. Information Systems, 2022, 103: 101771.

[23] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[24] Aljohani N R, Fayoumi A, Hassan S U. Bot prediction on social networks of Twitter in altmetrics using deep graph convolutional networks[J]. Soft Computing, 2020: 1-12.

[25] Li S, Zhao D, Wu X, et al. Functional immunization of networks based on message passing[J]. Applied Mathematics and Computation, 2020, 366: 124728..

[26] Nie Y, Jia Y, Li S, et al. Identifying users across social networks based on dynamic core interests[J]. Neurocomputing, 2016, 210: 107-115.

[27] Gao C, Liu J. Network-based modeling for characterizing human collective behaviors during extreme events[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, 47(1): 171-183.

[28] Zhu P, Zhi Q, Guo Y, et al. Analysis of epidemic spreading process in adaptive networks[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2018, 66(7): 1252-1256.

[29] Su S, Tian Z, Li S, et al. IoT root union: a decentralized name resolving system for IoT based on blockchain[J]. Information Processing & Management, 2021, 58(3): 102553.

[30] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[J]. Advances in neural information processing systems, 2017, 30: 3146-3154.

[31] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[32] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html?highlight=adaboost#sklearn.ensemble.AdaBoostClassifier

[33] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016: 785-794.

[34] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random#sklearn.ensemble.RandomForestClassifier

[35] Dorogush A V, Ershov V, Gulin A. CatBoost: gradient boosting with categorical features support[J]. arXiv preprint arXiv:1810.11363, 2018.