Hindawi
Complexity
Volume 2022, Article ID 5864545, 11 pages
https://doi.org/10.1155/2022/5864545

Research Article

Friend Recommender System for Social Networks Based on Stacking Technique and Evolutionary Algorithm

Aida Ghorbani,[1] Amir Daneshvar ,[2] Ladan Riazi,[2] and Reza Radfar[3]

[1]*Department of Information Technology Management, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[2]*Department of Information Technology Management, Electronic Branch, Islamic Azad University, Tehran, Iran*
[3]*Department of Technology Management, Science and Research Branch, Islamic Azad University, Tehran, Iran*

Correspondence should be addressed to Amir Daneshvar; a_daneshvar@iauec.ac.ir

Received 10 April 2022; Revised 12 June 2022; Accepted 29 July 2022; Published 31 August 2022

Academic Editor: Shahzad Sarfraz

In recent years, social networks have made significant progress and the number of people who use them to communicate is increasing day by day. The vast amount of information available on social networks has led to the importance of using friend recommender systems to discover knowledge about future communications. It is challenging to choose the best machine learning approach to address the recommender system issue since there are several strategies with various benefits and drawbacks. In light of this, a solution based on the stacking approach was put out in this study to provide a buddy recommendation system in social networks. Additionally, a decrease in system performance was caused by the large amount of information that was accessible and the inefficiency of some functions. To solve this problem, a particle swarm optimization (PSO) algorithm to select the most efficient features was used in our proposed method. To learn the model in the objective function of the particle swarm algorithm, a hybrid system based on stacking is proposed. In this method, two random forests and Extreme Gradient Boosting (XGBoost) had been used as the base classifiers. The results obtained from these base classifiers were used in the logistic regression algorithm, which has been applied sequentially. The suggested approach was able to effectively address this issue by combining the advantages of the applied strategies. The results of implementation and evaluation of the proposed system show the appropriate efficiency of this method compared with other studied techniques.

1. Introduction

In recent years, by the development of social networks, individuals and organizations can easily interact with each other. People can get their favorite connections in different fields and share their knowledge as well. Most of the connections that individuals are making on social networks exist only in the virtual world and are not often accessible there. A social network is a graph in which each node stands for a person, group, or organization, and each link between nodes depicts the relationships among them. Understanding and describing the processes that create social interactions is one of the fundamental problems in social network analysis. In this regard, the problem of link prediction in social networks states that two nodes in a network will be connected in the near future or not[1].

One of the most important issues in social networks, which leads to its superiority over other networks, is the friend recommender system in it. In recent years, many methods were proposed to suggest a friend recommender system that had been used machine learning techniques and artificial intelligence [2, 3]. The performance of its methods is not, however, clearly categorized in the area of machine learning. This is mostly because there are so many methods and suggested adjustments in the literature. As a result, choosing a proper machine learning algorithm is difficult and confusing that fits the needs of the issue when developing a recommender system. Thus, considering that each machine learning method has its advantages and limitations, an approach based on stacking technique is presented that can combine the advantages of machine learning methods and improve the results.

Before moving on to the next phase of this study, a review of earlier techniques is provided. The different parts of the research's suggested system are then discussed. The outcomes of the system's implementation are then examined using a number of tests, and conclusions are provided at the end.

2. Related Work

The problem of link prediction, which first raised by Getoor and Diehl, is presented as a problem of predicting the presence of a link between two entities. The prediction is based on the properties of other objects and other observed links [4]. This is widely used in a wide range of real and important areas, especially those involving the detection of complex events from highly structured data [5]. A thorough summary of link prediction in various networks was given in a research by Daud et al. The report gave thorough descriptions of link prediction algorithms, cutting-edge technology, programs, problems, and future research objectives. Besides, several directions for future research in the field of link prediction in social networks were expressed [6].

In a research by Chen et al. [7], the encoder-LSTM-decoder (E-LSTM-D) system is proposed as a new deep learning model based on stacked long short-term memory (LSTM) in the encoder-decoder architecture. The various experiments performed in this paper show that the E-LSTM-D model in different datasets performs significantly better than the existing dynamic link prediction methods. Behera et al. used potential future linkages using a variety of machine learning algorithms, including K-NN, MLP, bagging, SVM, and decision trees, based on attributes retrieved from the topological structure. The performance of the proposed system of this research was evaluated in terms of various criteria [8].

In a research by Chonghuan [9], to solve the problem of sparsity of recommender systems, a new recommendation method for social network using matrix factorization technique proposed. In this method, users clustered and various complex factors considered as well. The simulation results showed that the proposed socialized recommendation method based on matrix factorization (SRM-MF) system performed better than the methods available on the tested dataset. For instance, the precision of the real dataset and the Book-Crossing dataset are 0.088 and 0.095, respectively, assuming the Hamming distance is 20. While 0.073 and 0.086, respectively, represent the greatest accuracy for other procedures used in similar circumstances, Pecli et al.'s experiments were performed on three datasets (Microsoft Academic Network, Amazon, and Flickr) that included more than twenty different features, including topological features and domain-specific features. The program combines three feature selection strategies, six different classification algorithms (support vector machines (SVM), k-nearest neighbors (K-NN), simple Bayesian, Classification and Regression Tree (CART), random forest, and multilayer perceptron) and three evaluation criteria (precision, F-measure, and area under the curve). Their research's findings revealed an intriguing relationship between the majority of the chosen characteristics and the dataset. The findings demonstrated that using feature selection techniques to condense the feature set produces better classification models than classifiers built using the whole set of features [10]. In the paper by Manshad et al., a new time series link prediction (TSLP) method based on irregular cellular learning automaton (ICLA) and evolutionary computation (EC) proposed. ICLA-EC had been used to analyze network evolution through neighborhood dynamicity. Based on experiments performed on different datasets, ICLA-EC-TSLP achieved significant results (0.7212–0.8650) in AUC criterion compared with other methods [11].

In Cai et al.'s research, a new link prediction model based on line graph neural networks is proposed that achieves good performance for the link prediction problem. Studies on 14 datasets revealed that the suggested approach of this study consistently outperformed all fundamental techniques in terms of area under the curve (AUC) by identifying more relevant features [12]. In the research of Parveen et al., the friends' recommendation system performed using different types of machine learning algorithms, such as Random Forest Classifier, XGBoost, Light GBM, and Cat Boost. The performance of the mentioned methods compared in F1-score criteria, accuracy, recall, and confusion matrix. The results of this study showed that Random Forest and Light GBM are less accurate than the XGBoost and CatBoost algorithms. The accuracy of the XGBoost and CatBoost algorithms was the same and equal to 95% [13].

Kumar et al. proposed a friend recommendation system that uses a random forest to advise a buddy. The data collection used in this study has 94,000 nodes. The achieved accuracy for this suggested model is 89%. It is stated that the accuracy obtained in relation to the available hardware and data volume is quite reasonable [14]. In the research of Murali et al., a recommendation system presented in which each user is offered the best research articles in this field. This recommendation method is based on the individual queries and similarities found from other users based on their queries. This recommendation system uses a collaborative filtering approach and helps to avoid user time-consuming [15].

In the research done by ZhengWei et al. [16], a solution based on XGBoost is proposed for classifying and recommending journals to researchers. The doc2vec is used to get better results. The accuracy of this method was measured at 84.24 percent after testing on Common SCI English publications in the computing industry to verify the findings. A unique Graph Neural Network for Reciprocal Recommendation (GraphRR) was suggested for exploiting multiplex user interactions in the research written by Chang et al. [17]. To display each user's preference, attraction, and likeness, three ego graphs are created for each user depending on the directions of interaction. Then, multiplexity-aware GNN modules are applied to measure participation. Extensive tests were conducted on large-scale real-world online gaming datasets from NetEase Games, which demonstrated the system's good performance.

As it is clear from the reviewed researches, recommender systems used in different fields and have of special importance. Studies shown that using machine learning techniques in this field is high and could be developed due to the nature of artificial intelligence methods. In Table 1, a summary of the reviewed related works is given.

Table 1: Summary of related works.

Research	Field	Method	Dataset
[7]	Link prediction	Encoder-LSTM-decoder (E-LSTM-D) system	CONTACT, ENRON, RADOSLAW, FB-FORUM, LKML
[8]	Link prediction	K-NN, MLP, bagging, SVM, and decision tree	Collected dataset
[9]	Recommendation system in social network	SRM-MF	Actual dataset, Book-Crossing
[10]	Link prediction applications	Selected feature method with SVM, K-NN, simple Bayesian, CART, random forest, and multilayer perceptron	Microsoft Academic Network, Amazon, and Flickr
[11]	Link prediction	ICLA-EC-TSLP	Hep-th, Hep-ph, Astro-ph, Email-Enron, CollegeMsg
[12]	Link prediction	Line graph neural networks	14 various datasets
[13]	Friend recommendation system	Random Forest Classifier, XGBoost, Light GBM, and Cat Boost	Facebook dataset
[14]	Friend recommendation system	Random forest	Facebook recruiting
[15]	Research recommendation system	Collaborative filtering	Collected dataset
ZhengWei et al. [16]	Classification and recommendation of academic journals	XGBoost with doc2vec	Common SCI English journals
Chang et al. [17]	Reciprocal recommendation	GraphRR	Real-world large-scale online games from NetEase Games

3. Proposed System

This paper aims to propose a new way as an effective approach based on the use of particle swarm optimization and ensemble methods for friend recommendation in social network. Details of the proposed system are shown in Figure 1.

The proposed system of this research uses particle swarm optimization algorithm to select effective features. Each particle in this method represents a set of characteristics that iteratively progress toward the best answer. In this algorithm, a particle that chooses the optimal attributes for the issue is ultimately chosen. The method used to learn the objective function of the particle swarm optimization algorithm involves a stacked model of several machine learning algorithms.

There are several ways to learn the model, and usually each of them has power in a particular area, and to use one of them, the necessary studies must be done to understand how they work. There is no algorithm that is always the best, and each approach performs differently depending on the data and the situation. This is a key concept in these algorithms and models. In these situations, combining the output of many classifiers is preferable than selecting a specific approach or technique. Since each of them may have different strengths and weaknesses, it is expected that their participation would have a good compensatory effect.

One of the best and most effective combining methods is the use of stacking technique in which multiple models are combined. This method is used to increase the accuracy of models and improve results and reliability in a wide range of business and research programs. Stacking is a learning-based method, which combines multiple classification models or regression models. There are two stages to the categorization process based on this model. The output of the first level of classifiers serves as the input for the classifiers at the second level in the stacking approach. In other words, it can be said that classifier prediction at one level is considered as a feature for the next level classifier.

In this regard, in the proposed method of this research, XGBoost methods and two forms of random forest are implemented for the first level. The results of the first-level algorithms are used in logistic regression, which is implemented in the second level of the proposed system. Finally, the obtained results are used as the fitness result in the objective function of the particle swarm optimization algorithm.

3.1. Data Preparation. The link prediction collection has been used. This collection could be find at Noesis, nd. This collection of information has collected 22 networks from different sources and fields. This dataset includes a wide range of different features and information. A summary of the information in these datasets is given in Table 2. Name of the network, number of nodes, number of edges, average degree, average clustering coefficient, and average length of shortest route, dimension, heterogeneity, and assortativity are all listed in the table from left to right [21]. BUP dataset and a few additional datasets were used to test the proposed system. The BUP dataset represents the network information of political blogs. This dataset includes 105 nodes, 441 links, and 8.4 degrees (Table 2).

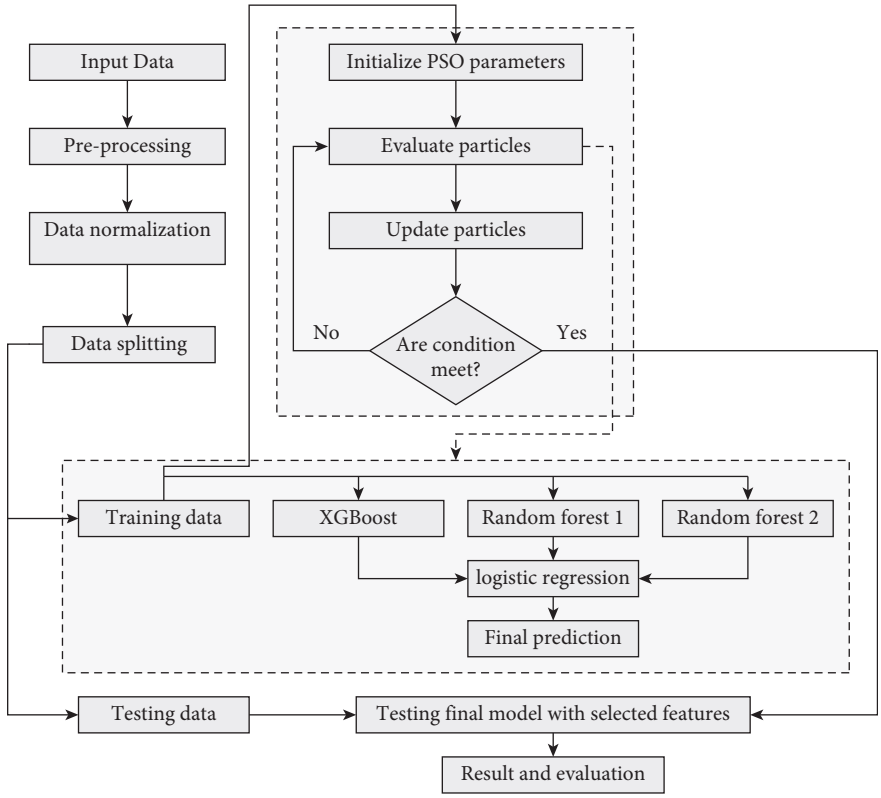

Figure 1: Proposed system.

Table 2: Topological properties of networks [21].

Name	$\|V\|$	$\|E\|$	k	C	ASPL	D	H	r
UPG	4941	6594	2.67	0.08	18.99	46	1.4504	0.0035
HPD	8756	32331	7.38	0.11	4.19	14	4.5133	−0.051
ERD	6927	11850	3.42	0.12	3.78	4	12.6708	−0.1156
YST	2284	6646	5.82	0.13	4.29	11	2.8479	−0.0991
EML	1133	5451	9.62	0.22	3.61	8	1.9421	0.0782
ADV	5155	39285	15.24	0.25	3.22	9	5.4060	−0.0951
KHN	3772	12718	6.74	0.25	3.63	12	9.422	−0.1205
PGP	10680	24316	4.55	0.27	7.49	24	4.1465	0.2382
CEG	297	2148	14.46	0.29	2.46	5	1.8008	−0.1632
LDG	8324	41532	9.98	0.31	4.37	16	6.188	−0.0997
SMG	1024	4916	9.6	0.31	2.98	6	3.9475	−0.1925
ZWL	6651	54182	16.29	0.32	3.85	10	2.5851	0.0006
INF	410	2765	13.49	0.46	3.63	9	1.3876	0.2258
BUP	105	441	8.4	0.49	3.08	7	1.4207	−0.1279
HTC	7610	15751	4.14	0.49	5.68	19	2.0986	0.2939
CGS	6158	11898	3.86	0.49	3.62	14	3.9467	0.2426
GRQ	5241	14484	5.53	0.53	5.05	17	3.0523	0.6593
EMT	2426	16630	13.71	0.54	3.15	10	3.1011	0.0474
FBK	4024	87887	43.68	0.59	3.98	13	2.432	0.0707
UAL	332	2126	12.81	0.63	2.74	6	3.4639	−0.2079
CDM	16264	47594	5.85	0.64	5.82	18	2.2087	0.1846
NSC	1461	2742	3.75	0.69	2.59	17	1.8486	0.4616

3.1.1. Normalization. Normalization is an operation on raw data that rescale the data or transform the data so that each attribute has a uniform contribution. Doing this will solve the problem of dominant features and outliers. Based on statistical criteria, there are many techniques for normalizing data within a certain range. The Z-score normalizing approach is used in this case for normalization. In this method, the criteria of mean and standard deviation are used to rescale the data so that the resulting features have a zero mean and a unit variance.

Each instance, $x_{i,n}$, of the data is transformed into $x'_{i,n}$ as follows:

$$X'_{i,n} = \frac{X_{i,n} - \mu_i}{\sigma_i}, \quad (1)$$

where μ and σ denote the mean and standard deviation of ith feature, respectively [22].

3.1.2. Feature Engineering. Nod2vec and NetworkX packages are used for feature engineering in this research. One of the introduced solutions for selecting features from a graph is known as Node2Vec. Node2vec is a flexible neighborhood sampling strategy that allows us to gently interpolate between BFS (Breadth First Search) and DFS (Depth First Search). This method is implemented by developing a biased flexible random walking method that can explore neighbors in both BFS and DFS methods [20].

A random walk is defined by two parameters p and q. We assume that the current random walking position is node v. The position of the previous step is node t. In order to determine the next position, the probabilities of π_{vx} transfer at the edges (v, x) leading to v must be evaluated. We set the probability of anomalous transfer to $\pi_{vx} = \alpha_{pq}$ (t, x)·w_{vx}. In particular, α_{pq} is defined as follows:

$$\alpha_{pq} = \begin{cases} \frac{1}{p}, & d_{tx} = 0, \\ 1, & d_{tx} = 1, \\ \frac{1}{q}, & d_{tx} = 2, \end{cases} \quad (2)$$

where d_{tx} defines the shortest distance between node t and node x, and the value of d_{tx} must be 0, 1, or 2. The p parameter controls the possibility of revisiting a node during a random walk. When the p value is high, the visited nodes are rarely sampled. This strategy promotes moderate exploration and eliminates redundant sampling across two hops. Alternatively, if p is small, the walk is directed backward by one step (Figure 2), keeping it "local" and near to the initial node u.

The q parameter allows the search to distinguish between "local" and "global" nodes. As shown in Figure 2, if $q > 1$, a random walk is more likely to be sampled from nodes around the node. BFS samples the nodes in a small location. Conversely, if $q < 1$, the random walk is farther away from v, which can receive more general information about the features. Therefore, the distance between the sampling node and the given source node does not increase strictly [21].

Recently, huge amounts of network data in various fields such as web pages, social networks, and power grids are being generated and collected. NetworkX package was created in April 2005 to analyze these massive and complex networks in Python [22]. This Python package is intended for building, modifying, and researching the composition and operation of complicated networks. A variety of networks or diagrams are shown using its data structures. In contrast to many other technologies, NetworkX is extremely versatile and built to handle data at a scale appropriate for contemporary issues. In this package, nodes can represent any object in Python, and edges can contain arbitrary data. In Figure 3, a graph plot of BUP data that created by NetworkX is shown. This figure shows the nodes and how they (edges) are connected.

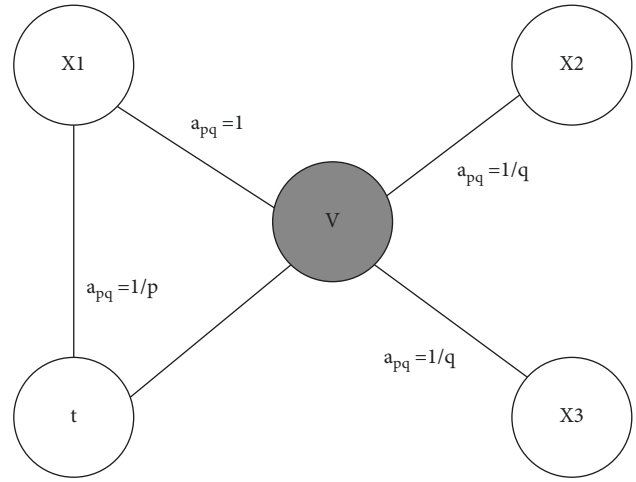

Figure 2: Node selection in Node2Vec algorithm. The current position in a random walk is at node v, and the previous step is at node (t). In this example, x1, x2, and x3 are neighbors. The values of a_{pq} are calculated based on the distance between v and t. [21].

3.1.3. Data Splitting. To implement and evaluate the efficiency of the proposed system, the data studied in this research are divided into two segments: training data and test data. In this classification, 70% of the total data examined is used for system training. To evaluate the system, the remaining 30% is considered as test data.

3.2. Classification Model. In these methods, classifiers had been combined to produce better predictions compared with single-level models. To do this, the stacking technique is used to implement several consecutive classifiers. As mentioned in the proposed system, several XGBoost is execution technique to increase the accuracy and performance of the Gradient Boosting Machine (GBM) and especially to increase the classification accuracy of regression trees proposed in [23], and two random forest algorithms have been ran in the first level, and finally, logistic regression model is one of the statistical tools used for data analysis, in which the relationship between a dependent variable and independent variables defined based on a series of observational values [24] and in the second level combined with them sequentially. The results of the first-level classifiers combined as the input of the second level and the final prediction in the second level are based on the results obtained from the first level.

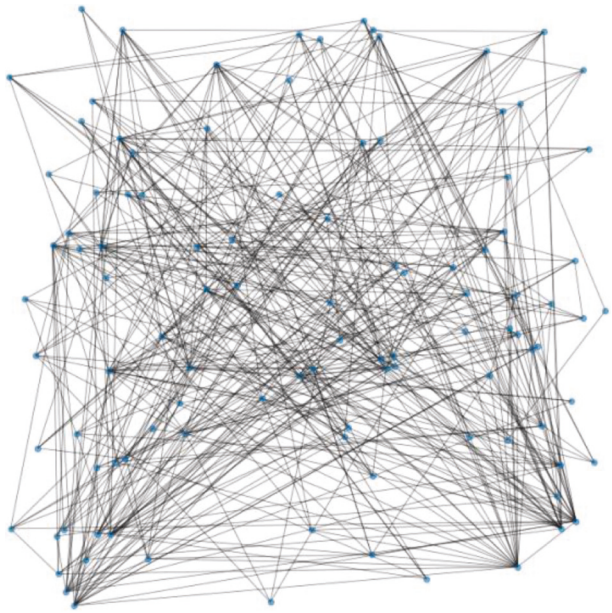

Figure 3: A graph plot of BUP data that created by NetworkX.

Because of the regular and parallel processing, XGBoost outperforms GBM. This approach integrates all predictors simultaneously for enhanced training [25]. The XGBoost algorithm is a system that successively generates decision trees. This algorithm can perform calculations relatively faster than all computing environments. XGBoost is widely used in modeling and classification for its performance.

Many decision trees grow in the classification of random forest algorithms, which is a batch algorithm. A decision tree algorithm can easily perform classification operations on events. The random forest algorithm uses several decision trees. In fact, a set of decision trees together produce a forest, and this forest can make better decisions (than a tree). In general, the decision tree is prone to overtraining and has little generalizability. The volatility of the decision tree's findings in the presence of noise in the input data is another drawback. A slight shift in learning patterns during the construction of a decision tree may result in significant changes to the tree's structure. Random forest, which operates by averaging the outcomes of all decision trees, is used to tackle these issues. The most important feature of stochastic forests is their high performance to measure the importance of variables, thus determining the role of each variable in predicting the response [26].

3.3. Feature Selection. In solving many problems, machine learning methods have difficulty in dealing with a large number of input features. One of the most crucial strategies in data preparation and feature selection is crucial for the efficient and accurate use of machine learning technologies. One of the important steps in the machine learning process is feature selection. This process identifies relevant features and removes irrelevant and additional data [27]. This process speeds up data mining algorithms, improves prediction accuracy, and increases comprehensibility. Irrelevant features are those that do not provide any useful information, and additional features do not provide more information than currently selected features.

In the proposed method of this research, particle swarm optimization algorithm is used to select the features. In this section, more relevant features are selected so that the performance of the friend recommender system is improved. The particle swarm optimization algorithm is a social search algorithm based on the social behavior and regular collective movements of birds and fish [28]. Despite the limited ability of each particle to find the best pattern, their collective behavior has a great ability to find the best path (in other words the best answer to optimization problems) as the position of each particle changes based on the particle's experience in previous movements and neighboring particle experiences. In fact, each particle is aware of its superiority or lack of superiority over neighboring particles as well as the whole group.

Two perspectives were considered to model the order in the collective movement of these particles. One dimension is the social interactions between group members, and the other dimension is the individual superiority that each group member may have. In the first dimension, all members of the group are obliged to always change their position by following the best person in the group. In the second dimension, it is necessary for each member to keep in their memory the best situation they have personally experienced and to have a tendency toward the best perceived situation of their past. Each of these members may become the leader of the group so that the other members have the duty to follow them.

After generating the initial population (particles) and considering an initial velocity for each particle, the fitness of each particle is calculated based on its position. Each particle in the search space represents one solution for the problem and changes its speed based on the best answer obtained in the particle group (best person in the group) and the best place that it has ever been. This velocity is added to the position of the particle, and a new position of the particle is obtained. In subsequent iterations, the best particle in terms of fitness helps the other particles and corrects their motion, and after successive iterations, the problem will converge towards the optimal answer.

The position vector for the ith particle with dimension d is $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]$. The velocity vector is defined as $V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,d}]$.

During motion, the best position that each particle can reach during the execution of the algorithm is called pbest = $[p_{b,1}, p_{b,2}, \ldots, p_{b,d}]$, and the best position that all particles have gained during the execution of the algorithm is called gbest = $[g_{b,1}, g_{b,2}, \ldots, g_{b,d}]$. The position and velocity vectors of each particle are defined as follows:

Table 3: Some of important parameters considered for different methods.

Method	Parameters
XGBoost	Number of boosting stages = 100 loss function = "log_loss" Learning rate = 0.1 Maximum depth of the individual regression estimators = 3
RandomForest_1	The number of trees in the forest = 20 measures the quality of a split = "gini" The number of features to consider when looking for the best split = "sqrt (n_features)"
RandomForest_2	The number of trees in the forest = 30 measures the quality of a split = "gini" The number of features to consider when looking for the best split = "sqrt (num of features)"
Logistic regression	Solver = "lbfgs" penalty term = "L2" Tolerance for stopping criteria = 1e-4

Table 4: Result for class 1 in BUP dataset.

	Precision	Recall	F-measure	Accuracy
XGBoost	0.59	0.57	0.58	0.59
RandomForest_1 (n_estimators = 20)	0.67	0.63	0.65	0.66
RandomForest_2 (n_estimators = 30)	0.66	0.64	0.65	0.65
Logistic regression	0.58	0.59	0.59	0.58
Proposed method	0.68	**0.70**	**0.69**	0.69

$$v_{i,d}(t+1) = w_t \cdot v_{i,d}(t) + c_1 \cdot \text{rand}_1\left(p\text{best}_{i,d}(t) - x_{i,d}(t)\right) + c_2, \quad (3)$$
$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1).$$

c_1 is learning coefficient related to personal experiences of each particle and c_2 is learning coefficient related to group experiences. The rand_1 is random number between [0 1]. The w_t is a control parameter that controls the effect of the current particle velocity on the next velocity and creates a balance between the algorithm's ability to search locally and globally.

4. Results

In this section, the results of the implementation of proposed system reviewed. To do this, the performance of the friend recommender system has been examined in 5 different modes. In these 5 modes, the XGBoost, first Random Forest, second Random Forest, Logistic Regression, and the proposed system were used as learning models. First, some of important parameters considered for different methods are stated in Table 3.

The results of this implementation are given in Table 4. The values obtained for precision, recall, F-measure, and accuracy criteria for the mentioned methods for class one are given in this table. Based on the results of this table, the values of the criteria if the proposed system is used are 0.68, 0.70, 0.69, and 0.69, respectively. These findings demonstrate that if the suggested stacking strategy is used, the best results are achieved for every analyzed criterion. Better outcomes than individual base models are attained in the proposed system because basic learning models are stacked and their unique capabilities are used. The confusion matrix obtained from the stacking for the train and test data is given in Figure 4. Each column of the confusion matrix represents a sample of the predicted value, and each row contains actual sample. To classify two classes, each member sample will be either positive or negative. Therefore, for each data sample, four states may occur that are represented by the confusion matrix.

The sample is a member of a positive class and is recognized as a member of the same class (true positive). The sample is a member of the positive class and is recognized as a member of the negative class (false negative). The sample is a member of a negative class and is recognized as a member of the same class (true negative). Finally, the sample is a member of the negative class and is recognized as a positive class member (false positive).

Since the elements on the main diameter show the correct samples (true positive and true negative), as shown in Figure 4 for the training dataset, their sum is equal to the total number of samples. The values of the elements on the subdiameter show the incorrect samples (false negative and false positive), which are zero for the training dataset. Therefore, the confusion matrix for training data showed the highest possible performance.

The confusion matrix findings for the test data show that there are not many false positive instances or false negative cases, which is acceptable. This demonstrates how well the suggested stacking approach works in both classes.

4.1. Results for Other Datasets. In this section, the proposed method on other datasets was also examined. These datasets are INF, CEG, and UAL, respectively. The results

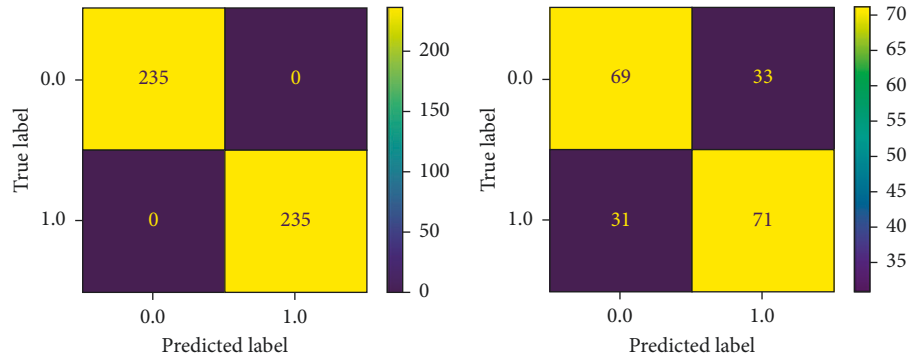

Figure 4: Confusion matrix for proposed method for BULP dataset. Left figure is for training, and right figure is for testing dataset.

Table 5: Result for class 1 in INF dataset.

	Precision	Recall	F-measure	Accuracy
XGBoost	0.66	0.63	0.69	0.67
RandomForest_1 (n_estimators = 20)	0.71	0.67	0.69	0.70
RandomForest_2 (n_estimators = 30)	0.71	0.67	0.69	0.70
Logistic regression	0.53	0.53	0.53	0.53
Proposed method	0.73	0.68	0.70	0.71

Table 6: Result for class 1 in CEG dataset.

	Precision	Recall	F-measure	Accuracy
XGBoost	0.57	0.54	0.55	0.57
RandomForest_1 (n_estimators = 20)	0.65	0.61	0.63	0.64
RandomForest_2 (n_estimators = 30)	0.66	0.62	0.64	0.65
Logistic regression	0.53	0.54	0.54	0.53
Proposed method	0.63	0.66	0.64	0.64

of applying different methods to INF dataset are listed in Table 5.

The results of applying different methods to CEG dataset are listed in Table 6.

The results of applying different methods to UAL dataset are listed in Table 7.

As can be observed, the suggested method performs rather well on the investigated datasets. The findings from the INF and UAL datasets demonstrated that the suggested system of this study outperformed the alternative algorithms in every analyzed criterion. In these two experiments, Random Forest_2 had the best performance after the proposed system. In testing the methods on CEG dataset, Random Forest_2 performed better in the precision and accuracy criteria, and the proposed system performed better in the recall and F-measure criteria. However, due to the fact that in the case of this study, overlooked cases (false negatives) are more costly than false alarms (false positive), and recall is more important than other criteria. Therefore, like the other two datasets, the proposed system of this research performs better on this dataset.

4.2. Research Limitations. Because a portion of the methodology utilized in this work is based on stacking several machine learning techniques, training the system takes a disproportionately long amount of time. The training procedure for huge graphs might take a long time if hardware resources were limited. For future work, according to the stated point, one can focus on reducing system training time. Applying intelligent sampling methods and using a subset of data for the training process can be considered.

Table 7: Result for class 1 in UAL dataset.

	Precision	Recall	F-measure	Accuracy
XGBoost	0.65	0.64	0.64	0.64
RandomForest_1 (n_estimators = 20)	0.68	0.68	0.68	0.68
RandomForest_2 (n_estimators = 30)	0.68	0.68	0.68	0.68
Logistic regression	0.63	0.59	0.61	0.62
Proposed method	0.75	0.75	0.75	0.75

5. Conclusion

In this research, a friend recommender system based on a combination of XGBoost, random forest, and logistics regression techniques is proposed. The results of this approach's implementation of XGBoost methods and two types of random forests were integrated using the stacking method and the logistics regression algorithm. The particle swarm optimization algorithm in this method chooses the most efficient characteristics to achieve the highest efficiency. For better investigation, in addition to the proposed stacking system, XGBoost-based system, linear regression, and random forest were implemented The results of this comparison showed that the proposed stacking system can achieve higher precision, recall, F-measure, and accuracy than other implemented approaches. This system has been able to make good diagnoses in both existing classes and achieve good results.

6. Introducing the Tool

Python programming language is used to implement the proposed solution. Python is a powerful programming language that is easy for people to learn. High-level data structures in this programming language are very efficient, and object-oriented programming is made possible. Python Interpreter and the extensive standard library are freely available on all major platforms on the Python website [29]. For the Python programming language, there are several libraries on machine learning and data mining issues. These features have led to the widespread use of this language in the field of artificial intelligence.

Appendix

Some of the codes used in this article are shown in this appendix.

```
Load Data
with open ("Dataset\\files\\INF_full.net") as f:
    fb_links = f.read ().splitlines ()
counter = 0
for tmp in fb_links:
    counter + = 1
    if (tmp = = " * edges"):
        break;
fb_links = fb_links[counter:]
Process Graph Data
node_list_1 = []
node_list_2 = []
for i in tqdm (fb_links):
    node_list_1.append (i.split (" ")[0])
    node_list_2.append (i.split (" ")[1])
fb_df = pd.DataFrame ({"node_1": node_list_1, "node_2": node_list_2})
G = nx.from_pandas_edgelist (fb_df, "node_1", "node_2", create_using = nx.Graph ())
node_list = node_list_1 + node_list_2
node_list = list (dict.fromkeys (node_list))
adj_G = nx.to_numpy_matrix (G, nodelist = node_list)
all_unconnected_pairs = []
offset = 0
for i in tqdm (range (adj_G.shape[0])):
    for j in range (offset, adj_G.shape[1]):
        if i ! = j:
            #print (str (i), str (j))
            if nx.shortest_path_length (G, str (i+1), str (j+1))
                if adj_G[i,j] = = 0:
                    all_unconnected_pairs.append ([node_list[i],node_list[j]])
    offset = offset + 1
node_1_unlinked = [i[0] for i in all_unconnected_pairs]
node_2_unlinked = [i[1] for i in all_unconnected_pairs]
data = pd.DataFrame ({"node_1":node_1_unlinked, "node_2": node_2_unlinked}) data["link"] = 0
initial_node_count = len (G.nodes)
fb_df_temp = fb_df.copy ()
omissible_links_index = []
for i in tqdm (fb_df.index.values):
    G_temp = nx.from_pandas_edgelist (fb_df_temp.drop (index = i), "node_1", "node_2", create_using = nx.Graph ())
    if (nx.number_connected_components (G_temp) = = 1) and (len (G_temp.nodes) = = initial_node_count):
omissible_links_index.append (i)
fb_df_temp = fb_df_temp.drop (index = i)
Balancing Data
ytrain = np.reshape (ytrain.values, (-1, 1))
total_train = np.append (xtrain, ytrain, axis = 1)
total_train_class0 = total_train[total_train[:,-1] = = 0]
total_train_class1 = total_train[total_train[:,-1] = = 1]
total_train_class0_sampled = total_train_class0[0:total_train_class1.shape[0],:]
totalTrainBalanced = np.append (total_train_class0_sampled, total_train_class1, axis = 0)
xtrain = totalTrainBalanced[:, 0: totalTrainBalanced.shape[1]-1]
ytrain = totalTrainBalanced[:, -1]
ytest = np.reshape (ytest.values, (-1, 1))
total_test = np.append (xtest, ytest, axis = 1)
total_test_class0 = total_test[total_test[:,-1] = = 0]
total_test_class1 = total_test[total_test[:,-1] = = 1]
total_test_class0_sampled = total_test_class0[0:total_test_class1.shape[0],:]
totalTestBalanced = np.append (total_test_class0_sampled, total_test_class1, axis = 0)
xtest = totalTestBalanced[:, 0: totalTestBalanced.shape[1]-1]
ytest = totalTestBalanced[:, -1]
Modeling and Evaluation
def confusion_matrix_scorer (clf, X, y):
    y_pred = clf.predict (X)
    cm = confusion_matrix (y, y_pred)
    return {"accuracy": accuracy_score (y, y_pred),
            "precision": precision_score (y, y_pred),
            "recall": recall_score (y, y_pred),
            "f1": f1_score (y, y_pred),}
def numpy2dataframe (nparray):
```

```
    panda_df = pd.DataFrame (data = nparray,
            index = ["Row_" + str (i + 1)
            for i in range (nparray.shape[0])],
            columns = ["Column_" + str (i + 1)
            for i in range (nparray.shape[1])])
    return panda_df
    def showReuslt (clf, xtrain, xtest, ytrain, ytest):
    print ("-------------Train Result----------------")
    y_pred = predictions = clf.predict (xtrain)
    plot_confusion_matrix (clf, xtrain, ytrain)
    print (classification_report (ytrain, y_pred))
    print ("-------------Test Result----------------")
    y_pred = predictions = clf.predict (xtest)
    plot_confusion_matrix (clf, xtest, ytest)
    print (classification_report (ytest, y_pred))
maxScore = 0
final_xTrain = []
final_xTest = []
def objective_function_topass (model, X_train, y_train,
X_valid, y_valid):
    global maxScore
    global final_xTrain
    global final_xTest
    model.fit (X_train, y_train)
    P = accuracy_score (y_valid, model.predict (X_valid))
# accuracy_score, precision_score, f1_score, recall_score
    if (P > maxScore):
        maxScore = P
        final_xTrain = X_train
        final_xTest = X_valid
    return P
estimators = [ ("xg", GradientBoostingClassifier (n_esti
mators = 100, learning_rate = 1.0, max_depth = 1,
random_state = 0)),
    ("rf_1", RandomForestClassifier (n_estimators = 20,
random_state = 10)),
    ("rf_2", RandomForestClassifier (n_estimators = 30,
random_state = 100))]
clf = StackingClassifier (
    estimators = estimators, final_estimator = Logistic
RegressionCV ()
)
algo_object = ParticleSwarmOptimization (objective_-
function_topass, n_iteration = 10, population_
size = 10, minimize = False)
best_feature_list = algo_object.fit (clf, xtrain_df,
pd.DataFrame (ytrain), xtest_df, pd.DataFrame (ytest),
verbose = True)
#plot your results algo_object.plot_history ()
```

Data Availability

The data are tabulated in the article and included in the appendix.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. P. Muniz, R. Goldschmidt, and R. Choren, "Combining contextual, temporal and topological information for unsupervised link prediction in social networks," *Knowledge-Based Systems*, vol. 156, pp. 129–137, 2018.

[2] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: a systematic review," *Expert Systems with Applications*, vol. 97, pp. 205–227, 2018.

[3] A. P. Chobar, M. A. Adibi, and A. Kazemi, "Multi-objective Hub-Spoke Network Design of Perishable Tourism Products Using Combination Machine Learning and Meta-Heuristic Algorithms," *Environment, Development and Sustainability*, pp. 1–28, 2022.

[4] L. Getoor and C. P. Diehl, "Link mining," *Acm Sigkdd Explorations Newsletter*, vol. 7, no. 2, pp. 3–12, 2005.

[5] T. E. Senator, "Link mining applications," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 76–83, 2005.

[6] N. N. Daud, S. H. Ab Hamid, M. Saadoon, F. Sahran, and N. B. Anuar, "Applications of link prediction in social networks: a review," *Journal of Network and Computer Applications*, vol. 166, Article ID 102716, 2020.

[7] J. Chen, J. Zhang, X. Xu et al., "E-lstm-d: a deep learning framework for dynamic network link prediction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 6, pp. 3699–3712, 2021.

[8] R. K. BeheraBehera, K. S. Sahoo, D. Naik, S. K. Rath, and B. Sahoo, "Structural Mining for Link Prediction Using Various Machine Learning Algorithms," *International Journal of Social Ecology and Sustainable Development*, vol. 12, no. 3, pp. 66–78, 2021.

[9] C. Xu, "A novel recommendation method based on social network using matrix factorization technique," *Information Processing & Management*, vol. 54, no. 3, pp. 463–474, 2018.

[10] A. Pecli, M. C. Cavalcanti, and R. Goldschmidt, "Automatic feature selection for supervised learning in link prediction applications: a comparative study," *Knowledge and Information Systems*, vol. 56, no. 1, pp. 85–121, 2018.

[11] M. Khaksar, M. R. Meybodi, A. Salajegheh, and A. Salajegheh, "A new irregular cellular learning automata-based evolutionary computation for time series link prediction in social networks," *Applied Intelligence*, vol. 51, no. 1, pp. 71–84, 2021.

[12] L. Cai, J. Li, J. Wang, and S. Ji, "Line Graph Neural Networks for Link Prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, 2021.

[13] R. Parveen and N. S. Varma, "Friend's recommendation on social media using different algorithms of machine learning," *Global Transitions Proceedings*, vol. 2, no. 2, pp. 273–281, 2021.

[14] K. N. Kumar and K. Vineela, "Friend Recommendation Using Graph Mining on Social Media," vol. 4, no. 5, 2020.

[15] M. V. Murali, T. G. Vishnu, and N. Victor, "A collaborative filtering based recommender system for suggesting new trends in any domain of research," in *Proceedings of the 2019*

5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 550–553, IEEE, Coimbatore, India, March 2019.

[16] H. ZhengWei, M. JinTao, Y. YanNi, H. Jin, and T. Ye, "Recommendation method for academic journal submission based on doc2vec and XGBoost," *Scientometrics*, vol. 127, pp. 2381–2394, 2020.

[17] Y. Chang, L. Shu, E. Du et al., "GraphRR: a multiplex graph based reciprocal friend recommender system with applications on online gaming service," *Knowledge-Based Systems*, vol. 251, Article ID 109187, 2020.

[18] M. Gómez-Víctor, "Supervised Data Mining in Networks: Link Prediction and Aplications," Doctoral Thesis, University of granda, Spain, 2018.

[19] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, Article ID 105524, 2020.

[20] A. Grover and A. J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, San Francisco, CA, USA, August 2016.

[21] J. Peng, J. Guan, and X. Shang, "Predicting Parkinson's disease genes based on node2vec and autoencoder," *Frontiers in Genetics*, vol. 10, p. 226, 2019.

[22] A. Hagberg and D. Conway, "NetworkX: Network Analysis with Python," 2020.

[23] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system,"vol. 13, pp. 785–794, in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, vol. 13, pp. 785–794, ACM, San Francisco CA USA, August 2016.

[24] M. Saadi and R. Abolfazl, "Analysis and estimation of deforestation using satellite imagery and GIS," *GIS Application in Environment*, 2000.

[25] J. Fan, X. Wang, L. Wu et al., "Comparison of Support Vector Machine and Extreme Gradient Boosting for predicting daily global solar radiation using temperature and precipitation in humid subtropical climates: a case study in China," *Energy Conversion and Management*, vol. 164, pp. 102–111, 2018.

[26] L. Braiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[27] V. Kumar and S. Minz, "Feature selection: a literature review," *The Smart Computing Review*, vol. 4, no. 3, pp. 211–229, 2014.

[28] R. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory, Sixth International Symposium on Micro Machine and Human Science," in *Proceedings of the 1995 MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, October 1995.

[29] G. V. Rossum, *Python Tutorial*, Vol. 3, 12th Media Services, Suwanee GA USA, 2018.