Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 19(7): 7248–7271.
DOI: 10.3934/mbe.2022342
Received: 11 March 2022
Revised: 02 May 2022
Accepted: 06 May 2022
Published: 17 May 2022

Research article

Shilling attack detection for collaborative recommender systems: a gradient boosting method

Chen Shao and Yue zhong yi Sun*

School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

* **Correspondence:** Email: syzy@hrbust.edu.cn.

Abstract: Organized malicious shilling attackers influence the output of the collaborative filtering recommendation systems by inserting fake users into the rating matrix within the database. The existence of shilling attack poses a serious risk to the stability of the system. To counter this specific security threat, many attack detection methods are proposed. Some of the past methods suffer from two disadvantages, the first being that they only analyze the rating matrix from a single perspective of user rating values and ignore other perspectives. Another is that some methods only use a single classifier to handle the classification of malicious attackers. Considering the above disadvantages, this paper proposes a gradient boosting method (named XGB-SAD) to achieve attack detection by combining double-view and gradient boosting. We first analyze the rating matrix with a double-view of time and item, which in turn defines the TPUS collection. Then our method uses eXtreme Gradient Boosting to perform heuristic iterative optimization of the model's objective function and uses the idea of ensemble learning to integrate multiple sets of base classifiers into strong classifier. The integrated strong classifiers are used to complete the detection of malicious attackers. Finally, we perform several experiments and the results demonstrate that XGB-SAD outperforms the comparison methods in terms of small-scale attack detection and overall detection, which proves the performance of our method.

Keywords: recommendation system; collaborative filtering; shilling attack detection; ensemble learning; gradient boosting

1. Introduction

In an age of information overload, the amount of data that far exceeds our processing power has permeated our society. Recommendation systems are one of the most important tools for solving information overload due to their unique technical advantages. In recent years, the application of new machine learning and deep learning techniques [1–3] to the recommendation systems has continued to receive attention from many academic researchers. Collaborative filtering [4] recommendation systems (CFRS) help people to make personalized choices while alleviating the problem of information overload. The subject of this paper is a recommendation system based on collaborative filtering techniques. One of the foundations for the implementation of collaborative filtering technology is the user profile file in the rating matrix. There is a positive correlation between the higher order of magnitude of the user profile and the output of the recommendation system [5]. Therefore, this requires the recommendation system to have strong open nature to facilitate different user registration ratings.

Malicious attackers inject a percentage of fake users into the recommendation system through certain methods, with the ultimate goal of changing the rating of targeted items and gaining illicit financial gain [6]. The above operations can lead to system misclassification, causing the prediction score of the items to change and producing the recommended results expected by the malicious attacker. Past literature has also referred to profile injection attack as shilling attack, this paper takes the latter terminology. Shilling attacks can interfere with the normal output of the recommendation system, or even cause it to crash as a whole. It is important for recommendation systems to perform the task of shilling attack detection in a reasonable and efficient way.

The attributes of all user profiles in a collaborative filtering recommender system can only be normal or fake users. Derive reasonable features and training high-performance classifiers are of great significance in improving the precision of shilling attack detection. Some past attack detection methods were used to analyze from the perspective of user ratings. But when a malicious attacker employs certain obfuscation techniques, the rating patterns of fake users and normal users will become similar. Some shilling attack detection algorithms that use features extracted from user rating values can cause misclassification for normal users. At the same time, shilling attack detection methods that rely on another single perspective also suffered from low precision problems.

To counter the disadvantages of past methods, we propose the XGB-SAD model based on a double-view and the mathematical principle of gradient boosting. First, we analyze it from the point of view of user rating time. The time window left for the shilling attacker to complete the attack profile injection is very short due to the time cost. However, the average interval between ratings of two items by normal users is generally not less than the length of an item. The above foundation provides the basis for solving the problem of shilling attack detection from a rating time perspective. Then, we analyze it from the point of view of the way users rating item. In order to eliminate the anomalous effects of user rating values, this paper analyses the problem in terms of the boolean values of the user rating items. Based on the above double-view, we build the TPUS collection. Thirdly, related literature shows that single classifiers are less effective in detecting shilling attacks. To address this limitation, our method uses eXtreme Gradient Boosting to perform heuristic iterative optimization of the model's objective function. Then, we use the idea of ensemble learning to integrate multiple sets of base classifiers into strong classifier. Finally, we perform multi-group experiments. The results demonstrate that XGB-SAD outperforms the comparison methods in terms of small-scale attack detection and overall detection.

In Section 2, we introduce the background of the paper; In Section 3, we introduce theoretical preparation; In Section 4, we present our approach. In Section 5, we performed an experimental evaluation; In the final section, we present a comprehensive summary of the article and describe future work.

2. Background and related work

2.1. Attack profile and shilling attack model

Malicious attackers inject fake user ratings into the rating matrix to change the target item's recommended frequency. We refer to the above actions of malicious attackers as shilling attack. The attackers will choose different types of shilling attacks depending on the specific operating environment and knowledge base. In the push attack, attacker's ultimate goal is to improve the attack item's recommended frequency. In the nuke attack, attacker's goal is the opposite. Table 1 lists the general forms used in rating matrix and shilling attacks. The user's rating values for items range from 1 to 5, with "0" representing no rating, "1" representing the lowest preference for items and "5" representing the highest preference for items. Taking push attack as an example, suppose the objective of the CFRS is to predict the rating value of $user_2$ to $item_6$. The principle underlying the technical implementation of the CFRS is to predict the rating of unknown items based on the similar nearest neighbours of the user. Before the push attack, the similar nearest neighbor of $user_2$ is $user_1$ and the system's prediction score is 1. The probability that the final recommendation list contains $item_6$ is very low. After the push attack is executed, the malicious attacker injects the attack profiles of $attacker_{1-4}$ into the system database. Then, the similar nearest neighbors of $user_2$ will become $attacker_{1-4}$ and the prediction score of the recommendation system will become 5. There is a high probability that the system will recommend the $item_6$ to the $user_2$. The attacker has successfully executed the push attack to reverse the system's interest in $user_2$'s items.

Table 1. The general rating matrix and shilling attack.

	$item_1$	$item_2$	$item_3$	$item_4$	$item_5$	$item_6$
$user_1$	0	0	4	1	0	1
$user_2$	2	0	4	1	0	**?**
$user_3$	0	1	2	0	2	0
$user_4$	0	3	0	0	3	1
$attacker_1$	2	0	4	1	0	5
$attacker_2$	2	0	4	1	0	5
$attacker_3$	2	0	4	1	0	5
$attacker_4$	2	0	4	1	0	5

Malicious attackers choose different rating items depending on their purpose. The collection of these ratings and their parameters is called an attack profile. Table 2 shows the specific components used in attack profile. I_S is a collection of items selected to be scored according to the specific needs of shilling attackers. In some attack models the I_S is not a necessary component. I_F refers to the filler items. The purpose of its existence is to more closely match the way real users rate their behaviour. $I_\emptyset$ refers to the collection of blank rating value items. I_T refers to the target item. There can only be one target item in this paper.

Table 2. The general attack profile.

I_S			I_F			$I_\emptyset$			I_T		
i_1^S	…	i_k^S	i_1^F	…	i_l^F	$i_1^\emptyset$	…	$i_v^\emptyset$	i_1^T	…	i_l^T
$\delta(i_1^S)$	…	$\delta(i_K^S)$	$\sigma(i_1^F)$	…	$\sigma(i_l^F)$	*null*	*null*	*null*	$\gamma(i_1^T)$	…	$\gamma(i_l^T)$

Table 3 lists the parameters used in shilling attack models.

• Random attack: There is no I_S in this attack model. The selection of I_F and the rating values are random. I_T is set to r_{max} in the push attack.

• Average attack: There is no I_S in this attack model. The value of I_F is set to r_{avg} in the system. The value of I_T is set to r_{max} in the push attack.

• Bandwagon attack: The value of I_s is set to r_{max} in this attack model. The I_F items in bandwagon attack are selected from the Top-N items in the recommendation system. I_T is set to r_{max} in the push attack.

The above models are the most widely used and impactful shilling attack models. Correct analysis and detection of the above three attack models are important for our understanding of other attack models.

Table 3. Several common shilling attack models.

Shilling attack model	Push attack/Nuke attack
Random attack	$I_S=\emptyset;\ I_F=r_{ran};\ I_T=r_{max}/I_S=\emptyset;\ I_F=r_{ran};\ I_T=r_{min}$
Average attack	$I_S=\emptyset;\ I_F=r_{avg};\ I_T=r_{max}/I_S=\emptyset;\ I_F=r_{avg};\ I_T=r_{min}$
Bandwagon attack	$I_S=r_{max};\ I_F=r_{ran};\ I_T=r_{max}/I_S=r_{min};\ I_F=r_{ran};\ I_T=r_{min}$
Segment attack	$I_S=r_{max};\ I_F=r_{min};\ I_T=r_{max}/I_S=r_{min};\ I_F=r_{max};\ I_T=r_{min}$
Love/Hate attack	$I_S=\emptyset;\ I_F=r_{min};\ I_T=r_{max}/I_S=\emptyset;\ I_F=r_{max};\ I_T=r_{min}$

2.2. User rating value

The rating matrix is the core data source for collaborative filtering recommendation systems. The user ratings behind the rating matrix represent the rating habits and rating preferences of normal users. The malicious attacker generates the user's rating by constructing a relevant attack model in bulk to simulate it. Items that are of real interest to malicious attackers are known as target item. In various shilling attack models, the target item is rated as an anomalous extreme score (highest or lowest score). However, other items are scored as average or random. Attack detection from the above perspective was the initial thinking of the researchers. The related researchers analyzed the differences between normal users and fake users in terms of user rating values and extracted classification features. The following equations show the mathematical definition of the rating matrix Eq (1) and some key classification features extracted from the user rating values in the rating matrix Eqs (2)–(7).

Rating matrix
Definition:

$$R_{i\times j} = [u_1, u_2, u_3 \dots u_i]^T \tag{1}$$

Rating deviation from mean agreement
Definition:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{\left|r_{u,i} - \bar{r}_{u.i}\right|}{NR_i}}{N_u} \tag{2}$$

Weighted degree of agreement
Definition:

$$WDA_u = \sum_{i=0}^{n_u} \frac{r_{u,i} - \bar{r}_i}{I_i^2} \tag{3}$$

Weighted deviation from mean agreement
Definition:

$$WDMA_u = \frac{\sum_{i=0}^{n_u} \frac{\left|r_{u,i} - Avg_i\right|}{NR_i^2}}{N_u} \tag{4}$$

Mean variance
Definition:

$$MeanVar_u = \frac{\sum_{j\in p_{u,F}} (r_{u,j} - \bar{r}_u)^2}{\left|P_{u,F}\right|} \tag{5}$$

Filler mean variance
Definition:

$$FMV_u = \frac{1}{|U_u^{F_m}|} \sum_{i\in U_u^{F_m}} (r_{u,i} - \bar{r}_i)^2 \tag{6}$$

Filler mean target difference
Definition:

$$FMTD_u = \left|\left(\frac{\sum_{i\in P_{u,T}} r_{u,i}}{\left|P_{u,T}\right|}\right) - \left(\frac{\sum_{k\in P_{u,F}} r_{u,k}}{P_{u,F}}\right)\right| \tag{7}$$

2.3. Shilling attack detection

Among the large group of technologies for recommender systems, CFRS is the most mature, widely implemented and theoretically well-developed technique. The technique of collaborative filtering is based on the following idea: In the rating matrix, the specific set with the most similarities to the target user is generated in a certain way. This user group is used as the basis for predicting the target user's rating value for the unrated items. Then the above results are combined to complete the recommendation to the target user. CFRS have important applications in the business world. It solves the problem of information overload for consumers and significantly reduces the economic costs for both sides of the transaction. However, the existence of shilling attacks seriously hinders the proper functioning of recommendation systems.

Shilling attackers influence the output of the CFRS by inserting fake users into the rating matrix within the database. The existence of shilling attack poses a serious risk to the security and stability of the CFRS. If not dealt with in a timely manner, the financial interests of both the recommendation service platform and the consumer can be greatly compromised. As a result, research on shilling attacks has received a lot of attention and has become a hot topic in the recommendation systems. A well-designed shilling attack detection method maintains the stability of the overall system and protects the property of both the recommendation system service provider and the consumer. Based on the different perspectives and concepts involved, a variety of detection algorithms have been designed by those involved. There are three main research directions for attack detection algorithms.

In the supervised learning, Chirita et al. [7] were among the first researchers to focus on shilling attacks. They first proposed the RDMA attribute and used it as the basis for a shilling attack detection algorithm. These pioneering studies provided a good basis for subsequent researchers. Burke et al. [8] extracted these detection features from the user profile and then combined them with the KNN algorithm model to perform the task of attack detection. Williams et al. [9] proposed several detection features and combine them with several machine learning algorithms. Specifically, the SVM has the best overall results. Tang et al. [10] proposed three detection factors based on rating interval as a basis for the task of shilling attack detection. Xia et al. [11] used the slope and first order derivative of the time series rating values to dynamically divide the time interval as a way to detect anomalous rating items. The above methods for identifying anomalous users or anomalous items based on rating time are mainly used to detect a large number of attacks with concentrated rating time, but the above methods are not effective for attacks with decentralized injection. Yang et al. [12] proposed three new detection features based on rating fill rate. Combining 15 existing detection features such as WDA and RDMA, the Re-scale AdaBoost algorithm is used to detect malicious users. Wu et al. [13] selected the features that work best for a specific attack type from many shilling attack features as detection features by means of feature selection for different shilling attack types. They also proposed two detection methods based on classical machine learning algorithms. Li et al. [14] proposed a method. The method uses statistical analysis of item popularity, which in turn reveals the difference in popularity distribution between fake and normal users. In the semi-supervised learning methods, Wu et al. [15] used an expectation max method to filter the rating features proposed by Williams. The method is effective in improving the detection accuracy due to the use of a more efficient subset of features. In the unsupervised learning approach, Mehta et al. [16] proposed unsupervised learning methods PCA-SelectUsers to detect malicious fake users. More importantly, this detection method requires certain information to be obtained in advance, such as the specific size of the attack, in order to intercept the attack situation based on a ranked list of principal components. Yang et al. [17] constructed an unprivileged user relationship graph, used graph mining methods to calculate user similarity, and used a clustering algorithm to detect some suspicious users. Finally, the attacking users were further screened based on target item analysis. This threshold value is difficult to determine in practical applications.

3. Preliminaries

3.1. User rating time

Malicious shilling attackers change the system recommendation frequency of target item with the aim of increasing their illegal financial interests. Because of the specific implementation principle of collaborative filtering technology, this malicious change can only be accomplished if the shilling attacks reach a certain size. The time cost of the shilling attacker makes the difference between the injected fake user and the normal user. Therefore, the number of ratings of the attack profile increases significantly in a relative unit of time. The shilling attack on recommendation systems is a short-term utilitarian act. This short-term behaviour is extremely evident in the rating intervals. The relative rating time interval for a normal user of an item is not less than the length of a movie. However, the relative rating interval [18] for a malicious shilling attacker will be significantly less than that of a normal user. Unlike normal users, the purpose of the shilling attacker is to gain illegal financial interests. Therefore, the economic cost is an important consideration for the attacker. The cost of time can have a very significant impact on the expected economic benefit of a malicious attacker, which is why there is a difference in the user rating time [19] of the fake user and the normal user. On the basis of the above, we have defined A Eq (8), B Eq (9) and C Eq (10). TPUS-1 in the TPUS set is defined in Eq (11).

A. Collection of user rating time (UTC)

Definition: The elements in the set are the timestamps of the user's rating items, sorted in descending order. That is, the higher the serial number of the element, the closer the element's rating time is to the current time. (*u* refers to a particular user and n refers to the *n* items rated by this user)

$$UTC_u = \{t_1, t_2, \dots, t_n\} \tag{8}$$

B. Maximum interval of user rating time (UTM)

Definition: UTM is the subtraction of the first and last element values in the user's UTC set. (*n* refers to elements in the user's UTC)

$$UTM_u = UTC_n - UTC_1 \tag{9}$$

C. Aggregation index of user rating time (IUT)

Definition: IUT refers to the ratio of the user's UTM value to the total number of rated items N_u.(*u* refers to a particular user)

$$IUT_u = \frac{UTM_u}{N_u} \tag{10}$$

$TPUS_1$: Relative aggregation index of user rating time (RIUT)

Definition: RIUT is the first component of the TPUS collection and is used to describe the relative degree of aggregation of rating time. $\overline{UTM}$ is the average of the UTM values of the users within the database and $\overline{N}$ is the average of all user's rated items.

$$TPUS_1 = RIUT_u = \frac{|UTM_u - \overline{UTM}|}{|N_u - \overline{N}|} \tag{11}$$

3.2. User rating item

Past methods have mostly analysed the rating matrix from the single perspective of user rating values. However, there are a number of problems with the above methods. When recommendation system information is obtained through public or semi-public channels, a malicious attacker will use the above basis and various obfuscation techniques to design an attack model. The attack profile generated on the basis of this attack model will be very similar to the normal user's rating value habits. As a result, such shilling attack detection methods based on rating values can misjudge normal and fake users, affecting the accuracy of the detection algorithm. In order to eliminate the anomalous effects of user rating values, this paper analyses the problem from the user rating items.

In the rating matrix of a recommendation system, users select items based on preference and assign them different rating values. In this paper, we are not concerned with the specific rating values of the users, so we booleanize the users' rating values. The URB is defined in Eq (12). URB only cares if the user has a rating for the item, reducing the potential negative impact of obfuscation techniques on the rating value. The user rating matrix is transformed using Eq (12) to convert the original user rating values to Boolean values, and the transformed rating matrix is called the URB matrix. In the URB matrix, "1" indicates that the user has a rating action for the item and "0" indicates that the user does not have any rating action. Based on the user's URB value, we define the user's TCIB value in Eq (13). For an item in the URB matrix, TCIB calculates the sum of all non-zero URB values for the column in which the item is located. The TCIB value for each item represents how many times the item has been rated and how well it has been accepted. On the basis of the above, we have defined F Eq (14), TPUS-2 Eq (15) and TPUS-3 Eq (16).

D. Boolean values of user rating (URB)

Definition: To eliminate the effect of user rating values, definition URB booleans the user's rating values. R_{ij} represents the rating value of $user_i$ to $item_j$ in the rating matrix $R_{m\times n}$. The specific definitions are as follows:

$$URB_{ij} = \begin{cases} 0 & ,R_{ij} = 0 \\ 1 & ,R_{ij} \neq 0 \end{cases} \tag{12}$$

E. The coefficient of item boolean (TCIB)

Definition: TCIB is the sum of the URB values of all users in the column in which the item is located in a rating matrix (R_{N-F}) that does not contain fake user. The TCIB value for j is defined as follows.

$$TCIB_j = \sum_{i=1}^{n} URB_{ij} \tag{13}$$

F. The set of user rating boolean item (UBS)

Definition: Among user's rating items, if an item has a TCIB value not equal to "0", it is placed in the UBS collection. The elements in the set are arranged in ascending order. The user's UBS is defined as follows.

$$UBS_u = \{TCIB_1, TCIB_2, \dots, TCIB_k\} = \{UBS_1, UBS_2, \dots, UBS_k\} \tag{14}$$

$TPUS_2$: Mean Index of User boolean (MIUB)

Definition: In the user's UBS collection, the elements in the collection are first summed in order from the beginning to the end. The ratio of the summed value to the total number of users in the set is defined as $TPUS_2$.

$$TPUS_2 = MIUB_u = \frac{1}{n}\sum_{i=1}^{n} UBS_i \tag{15}$$

$TPUS_3$: Sub-Range Index of User boolean (RIUB)

Definition: In the UBS collection of user u, subtract the value of the penultimate element from the value of the second element in the collection. n refers to the maximum number of elements in the UBS set.

$$TPUS_3 = RIUB_u = \frac{1}{n}(UBS_2 - UBS_{-2}) \tag{16}$$

3.3. Ensemble learning and gradient boosting

Ensemble learning refers to a theoretical idea in which multiple base classifier models are combined with the ultimate goal of obtaining a more comprehensive model. Ensemble learning is based on the idea that even if a few weak base classifiers output incorrect predictions, most of the other base classifiers can correct the bias so that the model outputs the correct results. Compared to methods using only a single classifier, ensemble learning methods can continually minimize the error of the base classifier and output a more comprehensive and robust model. There are four common integration learning methods, namely boosting, bagging, stacking and blending. This paper focuses on the first type of method. Boosting is an important branch of ensemble learning. Specifically, each iteration of the boosting model is accompanied by the generation of a new classifier that focuses on the parts that were in error during the previous iteration. The new classifier generated by the boosting model is combined with the classifier from the previous iteration, and the process is repeated to achieve the desired learning goal and result in a more robust classifier model.

Gradient boosting is one of a large group of boosting algorithms that can be used in machine learning for regression, classification and sorting tasks. The basic principle is as follows: the negative gradient of the loss function in the current mathematical model is first calculated and used as the basis for training the next newly added weak classifier. The trained weak classifiers are then continuously accumulated into the existing model. Gradient boosting often chooses a decision tree (usually a CART tree) as the base learner, a method known as GBDT. There are many advantages of GBDT, such as robustness of the model, high accuracy, ability to handle non-linear data, etc.

eXtreme Gradient Boosting [20] is a specific implementation of GBDT and improves and enhances it in many ways. Since its invention, the XGBoost algorithm framework has boasted excellent performance (e.g. high accuracy rate, wide applicability, etc.) in various machine learning competitions and has gradually gained the attention of related researchers. Essentially, the idea of the XGBoost algorithm framework is ensemble learning, which is an advanced and improved version of the GBDT algorithm. The XGBoost makes important improvements to the GBDT algorithm in several ways, the most important of which is the improvement of the loss function of the GBDT algorithm. XGBoost has a number of advantages over GBDT, such as a second-order Taylor expansion of the loss function, the addition of a regular term, reduced model complexity, etc.

eXtreme Gradient Boosting uses a heuristic algorithm to optimize the objective function in a step-by-step sequence. Figure 1 illustrates the framework of general flow framework for optimizing the XGBoost mathematical model. During this process, a new function is added to the new mathematical model each time based on the original mathematical model, and the optimal result is gradually achieved by iterations.

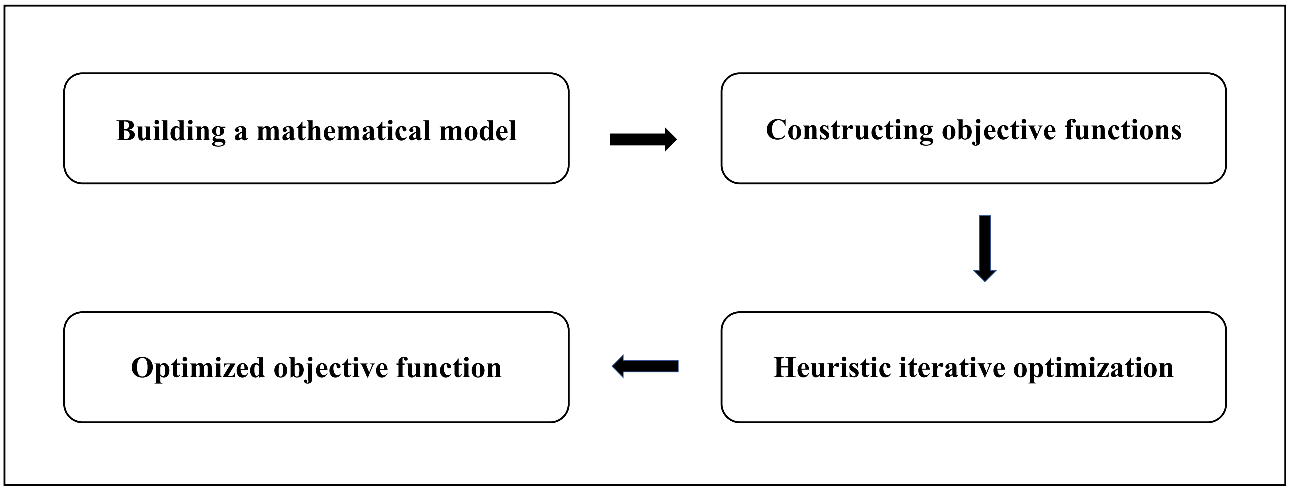
Building a mathematical model
Constructing objective functions
Heuristic iterative optimization
Optimized objective function

Figure 1. The general framework for the optimization of XGBoost mathematical model.

In the following, we describe in detail the variables and parameters involved in Figure 1.

The mathematical model of XGBoost is shown in Eq (17) below. (x_i, y_i) is the constituent element in the specified dataset $E = \{(x_i, y_i)\}$. $f(x)$ represents a specific cart tree model. Γ is the hypothesis space consisting of all base classifiers Cart Tree (Classification), defined as shown in Eq (18) below.

$$\phi(x_i) = \hat{y} = \sum_{k=1}^{K} f_k(x_i), f_k \in \Gamma \tag{17}$$

$$\Gamma = \{f(x) = w_{q(x)}\}(q: R^m \to T, w \in R^T) \tag{18}$$

The following Eq (19) represents the objective function of the model waiting to be optimized, which has two components. The first part represents the training error of the model. The second part is the regularization term, representing the sum of the complexity of all the trees. The definition of the regularization term of the objective function is shown in Eq (20) below.

$$L(\phi) = \sum_i \iota(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{19}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2 \tag{20}$$

The following second-order Taylor expansion transformation is applied to the error function. In Eq (21), the prediction of the model after the tth iteration has two components, the previous (t-1) model prediction and the prediction of the t-tree. The objective function can be rewritten as the following Eq (22) below.

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_i(x_i) \tag{21}$$

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\,(y_i, \hat{y}_i^{(t-1)} + f_t(\mathrm{x}_i)) + \Omega(f_t) \tag{22}$$

The model needs to calculate the learning is the tree f_t. Changing Eq (21) into (23), g_i and h_i are defined in Eq (24) below.

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathrm{x}_i) + \frac{1}{2} h_i f_t^2(\mathrm{x}_i)] + \Omega(f_t) \tag{23}$$

$$g_i = \partial \hat{y}^{(t-1)} l(y_i, \hat{y}^{(t-1)}) \qquad h_i = \partial^2 \ \hat{y}^{(t-1)} l(y_i, \hat{y}^{(t-1)}) \tag{24}$$

Removing the constant term from Eq (23) gives Eq (25) below.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathrm{x}_i) + \frac{1}{2} h_i f_t^2(\mathrm{x}_i)] + \Omega(f_t) \tag{25}$$

Take the content of the following Eqs (26) and (27) into the (25) objective function

$$f_k(x) = w_{q(x)}, w \in R^K, q\colon R^K \to \{1,2,\ldots,K\} \tag{26}$$

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \parallel w \parallel^2 \tag{27}$$

The following equation is obtained.

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \tag{28}$$

Defining the set of samples on each leaf node j as $I_j = \{i \mid q(x_i) = j\}$. The objective function (28) can be rewritten in the cumulative form of the leaf nodes as Eqs (29) and (30) below.

$$\tilde{L}^{(t)} = \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \tag{29}$$

$$\tilde{L}^{(t)} = \sum_{j=1}^{T} [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T \tag{30}$$

If q(x) has been determined (the structure of the tree has been determined), then the most predictive score for each leaf node can be found. The objective function is later minimised by making its derivative zero, as defined in Eq (31) below.

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{31}$$

After substitution into the objective function, the final optimization result can be obtained, as defined in Eq (32) below.

$$\tilde{L}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{32}$$

4. Our approach

4.1. Overview of the framework

The XGB-SAD attack detection model can be seen as a four-stage process. Figure 2 illustrates the basic framework of the ensemble detection method XGB-SAD. Module 1 is the stage where the fake user is generated using the relevant parameters and the shilling attack model. The normal users are mixed with fake users into a rating matrix which is used as input for subsequent modules. Module 2 is the stage where the analysis rating matrix is processed using a double-view. We analyzed user rating time and the prevalence of user rating items, based on which we obtained the TPUS and UFM (user feature matrix). Module 3 is the stage where the set of base classifiers is generated and the model is constructed. The mathematical principle of gradient boosting is used to complete the iterative optimization of the objective function and loss function. Module 4 is the stage where the classification of malicious fake users is completed using the integrated strong classifier.

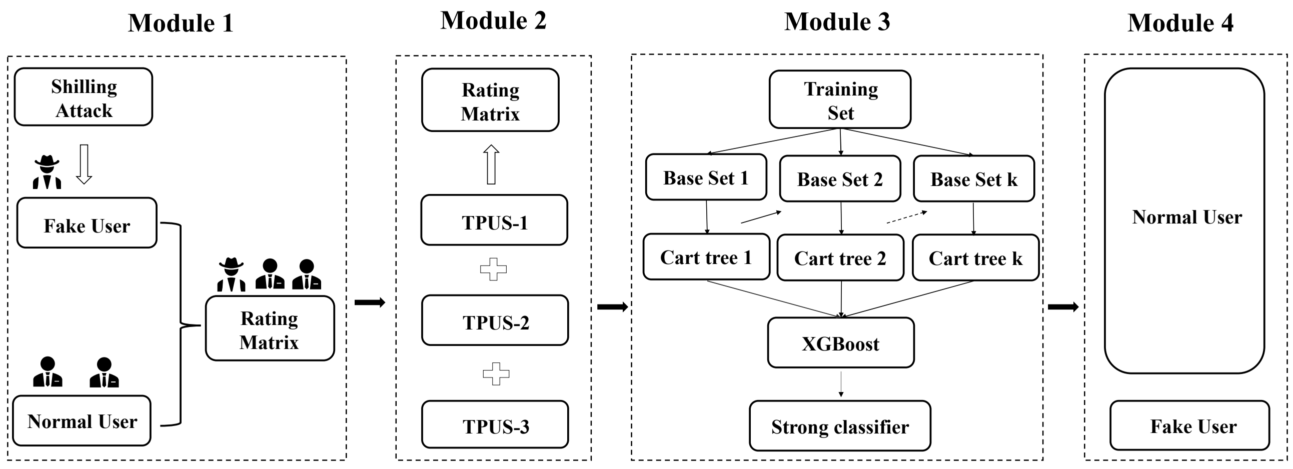

Figure 2. The framework of ensemble detection model XGB-SAD.

4.2. Double-view data pre-processing module

In Module 1, the parameters related to the shilling attack are as follows: attack size, fill size, and attack model. In this stage, we do some basic processing of the dataset in preparation for the subsequent modules. We first took some of the users from the original dataset as the data for the subsequent training set. Table 3 lists the parameters used in shilling attack models. The dataset was attacked using the attack model in Table 3 to generate the dataset required for the experiment. The set of fake users and normal users are mixed into one rating matrix. Finally, this rating matrix is used as input to subsequent modules.

In Module 2, we will analyse and process the rating matrix. Table 4 lists the TPUS-PRE algorithm used in data pre-processing. In order to solve the problem that past methods mostly used a single angle to analyse the rating matrix, we combined the double-view perspective in the TPUS-PRE algorithm. The underlying idea of the TPUS-PRE algorithm is to use double-view to analyse the rating matrix.

The TPUS-PRE algorithm has four inputs, TRS, I, U and N. In Sections 3.1 and 3.2, we have analysed and defined TPUS collection based on the user's time and the user rating item. The TPUS-PRE algorithm calculates benchmark definitions A, B and C in the training set based on Eqs (8)–(10), followed by benchmark definitions D, E and F based on Eqs (12)–(14). Construct a matrix V_t consisting of TPUS1 and a matrix V_b consisting of TPUS2 and TPUS3. Then, combine V_b and V_t and assign the value to V_u. The final output of the algorithm is the UFM (user feature matrix). The role of the TPUS-PRE algorithm in the detection framework is to perform further processing of the rating matrix, which is provided as output to subsequent modules.

Table 4. The algorithm of double-view data pre-processing (TPUS-PRE).

```
Input: TRS (Train-Set), I (collection of items), U (collection of users), N (collection of normal users)
Output: UFM (user feature matrix)
Begin
1 For each $i \in \mathrm{TRS.I}$ do
2     For each $u \in \mathrm{TRS.N}$ do
3         Calculate definitions A, B and C according to Eqs (8)–(10) respectively
4         Calculate definitions D, E and F according to Eqs (12)–(14) respectively
5     End For
6 End For
7 For each $u \in \mathrm{TRS.U}$ do
8     $V_t \leftarrow \{ TPUS_1 \}$
9     $V_b \leftarrow \{ TPUS_2, TPUS_3 \}$
10    $V_u \leftarrow V_b \cup V_t$
11 End For
12 $UFM \leftarrow V_u$
13 Return UFM
End
```

4.3. Constructing detection mathematical model module

In Module 3, we will use the mathematical principles of eXtreme gradient boosting to construct the set of base classifiers. Table 5 lists the algorithms that use the training-set to generate the base classifier (named GCTS). The underlying idea of the GCTS algorithm uses the mathematical principle of extreme gradient boosting to construct each subtree and base classifier set. To address the problems that existed in past detection methods that mostly used a single classifier to handle attack detection, this paper considers the use of a collection of base classifiers in CGTS. We divide the training set into

k subsets and use this as the basis for generating k base classifiers (Cart tree). The GCTS algorithm has four inputs, including number of iterations, train-set, base classifier cart tree and type label set. The final output of the algorithm is the base classifier set (Cart Tree Set).

After the start of the GCTS algorithm, we first use the TPUS-PRE algorithm in Table 4, with TRS as input. Then, we initialize the CTS set to the empty set. The mathematical model for XGBoost is given in Eqs (17)–(32) in Section 3.3, along with how to optimize the mathematical model and the loss function. The objective function is optimized using Eq (32) in Section 3.3. In each iteration, the objective function is first calculated according to Eq (19). Calculate g_i and h_i as parameter variables in the subsequent optimization process. Calculate $f_i(x)$ using Eq (26). After minimizing the objective function and loss function, $f_i(x)$ is incorporated into the CTS set. Execute the for loop repeatedly until N is reached. The result of the GCTS algorithm is a CTS. The role of the GCTS algorithm in the overall detection framework is to construct the set of base classifiers and prepare the predictions of subsequent base classifiers for combination.

Table 5. The algorithm for generating base classifiers set (GCTS).

```
Input: NT (Number of iterations), TRS (Training-Set), CT (base classifier-Cart Tree), TLS (Type Label Set)
Output: CTS (Cart Tree Set)
Begin
1 Using the TPUS-PRE algorithm in Table3, with TRS as input
2 $TRV_u \leftarrow UFM$
3 Initialize CTS and optimization model
4 Create mathematical model based on TRS and Eq (17)
5 $\{f_i(x) \in CTS\} \leftarrow NULL$
6 For i = 1 to NT do
7     $L(\phi) \Leftarrow TRV_u$ (Obtain objective function $L(\phi)$ by Eq (19) and TRV)
8     Calculate the parameters after the second order Taylor expansion by using Eq (24)
9     Obtain i base classifier CT subtree $f_i(x)$ according Eq (26)
10    Optimization of loss functions for mathematical model according to Eq (32)
11    $CTS \leftarrow CTS \cup \{f_i(x)\}$ (Merge the trained $f_i(x)$ into CTS)
12 End For
13 Return CTS
End
```

4.4. Malicious shilling attack detection module

In Module 4, we use the TPUS-PRE and the GCTS to construct the attack detection algorithm. Table 6 lists the algorithms used for shilling attack detection (called XGB-SAD). Two disadvantages of some past methods are that they only analyze the rating matrix from a single perspective of the user's rating value, ignoring other perspectives about the user. The other is that some methods use only a single classifier to handle the classification of malicious attackers. In the meantime, we have introduced the TPUS-PRE and GCTS algorithms. So based on the above foundation and in order to deal with the above problem in a reasonable way, the underlying idea of the XGB-SAD algorithm is to use UFM and CTS to combine the predictions of all the base classifiers to obtain a strong learner.

There are four inputs to the XGB-SAD, namely TES, CTS, TLS and THR. First, the algorithm takes the TES as input, calls the TPUS-PRE algorithm in Table 4, gets the output and assigns it to $\mathrm{V_u}$. In a double loop, initializing threshold THR and using $\mathrm{TEV_u}$ for the prediction of classifier f_k. After the predicted results $pre_{CTS^i}(u)$ are obtained, the predicted for each user $p(u)$ are combined. At this point, the inner for loop is ended. Combining the prediction results and threshold THR of the base classifier (cart tree) for user u, the category $TLS(u)$ of user u is determined. At this point, the outer for loop is terminated. Merge $TLS(u)$ with TLS_r and assign the merged result to ULS. The final output of the XGB-SAD algorithm is user label set (ULS). At the end of the XGB-SAD algorithm, the set of categories of normal user and fake user in Figure 2 is obtained.

Table 6. The algorithm of shilling attack detection (XGB-SAD).

Input: TES (Test-Set), **CTS** (base classifiers set - Cart Tree Set), **TLS** (Type Label Set), **THR** (Threshold)
Output: ULS (user label set)

```
Begin
1 Using the TPUS-PRE algorithm in Table3, with TES as input
2  $\mathrm{TEV_u} \leftarrow \mathrm{UFM}$
3 For each u $\in$ $\mathrm{TEV_u}$ do
4     $\mathrm{THR} \leftarrow \mathrm{INI}$ (Initialization Threshold)
5     For each $\mathrm{f_k} \in \mathrm{CTS}$ do
6           $\mathrm{f_k} \Leftarrow \mathrm{TEV_u}$ (Prediction detection of the test-set with the trained base classifier)
7         $\mathrm{pre_i} \leftarrow \mathrm{pre_{CTS^k}}$ (u) (Predicted results for the combined user u)
8         $\mathrm{p(u)} \leftarrow \mathrm{pre_k}$
9     End For
10  $\mathrm{RES(u)} \leftarrow \mathrm{COM(p(u))}$ (Prediction results of the combined base classifier cart tree for each sample)
11  $\mathrm{TLS(u)} \Leftarrow \mathrm{RES(u)} \cup \mathrm{THR}$ (Final category is determined by a combination of thresholds and res.)
12 End For
13  $\mathrm{TLS_r} \leftarrow \mathrm{TLS_r} \cup \{\mathrm{TLS(u)}\}$
14  $\mathrm{ULS} \leftarrow \mathrm{TLS_r}$
15 Return ULS
End
```

5. Experimental evaluation

5.1. Experimental setup

This subsection first describes the pre-experimental preparations, including the experimental dataset, attack size, filler size, attack model and comparison algorithms. The proposed method can be used for datasets (Netflix, Amazon, MovieLens, etc.) with various numbers of users and data sparsity. For the purpose of our study, the following dataset were extracted from Movielens-100k [21]. The Movielens-100k contains detailed rating information for 1682 items from 943 users. The user's rating values for items range from "1" to "5", with "0" representing no rating, "1" representing the lowest preference for items and "5" representing the highest preference for items. Table 7 illustrates the user-item rating matrix after simple processing of the Movielens-100k dataset.

Table 7. The rating matrix of Movielens-100k dataset.

	$item_1$	$item_2$	$item_3$	$item_4$	…	$item_{1682}$
$user_1$	5	3	4	3	…	0
$user_2$	4	0	0	0	…	0
$user_3$	0	0	0	0	…	0
$user_4$	0	0	0	0	…	0
…	…	…	…	…	…	…
$user_{943}$	0	5	0	0	…	0

The key parameters used in the experiments are presented in the following Eqs (33) and (34). In Eq (33), N refers to the quantity and the ratio of $N_{attck\ user}$ to N_u in the system database refers to the attack size. In Eq (34), N refers to the quantity. In an attack profile, the ratio of N_{IF} to N_{item} in the system refers to the filler size.

$$Atack\ \ size = \frac{N_{attack\ \ user}}{N_u} \tag{33}$$

$$Filler\ \ size = \frac{N_{IF}}{N_{item}} \tag{34}$$

The three comparison methods used in the experiments were PCA [16], Semi [22], and BAY [23]. PCA-SAD is a method that uses the unsupervised learning method PCA-SelectUsers to detect malicious fake users. Semi-SAD is a semi-supervised learning method. BAY-SAD combines multiple sets of base classifiers and uses the combined output to detect shilling attack.

5.2. Evaluation metrics

This section lists several evaluation metrics used in this experiment to evaluate the performance of methods. Precision Eq (35) indicates the percentage of the sample with positive predicted outcomes that also had positive true cases. Recall Eq (36) indicates the percentage of all samples with positive true cases that also have a positive predicted outcome. F1-Measure Eq (37) is a weighted average of precision and recall, reflecting the comprehensive effect of the model.

$$Precision = \frac{TP}{TP + FP} \tag{35}$$

$$Recall = \frac{TP}{TP + FN} \tag{36}$$

$$F1 - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{37}$$

5.3. Experimental results and discussion

5.3.1. Performance of three attack models

This subsection tests the evaluation metrics of the XGB-SAD under the attack models with the experimental parameters of filler size (3, 7, 10 and 15%) and attack size (3, 5, 7, 10, 12 and 15%). Figures 3–5 show the performance of our method for three attack models with the filler size fixed at 10%.

The Figure 3 below illustrates an overview of the variation in precision values. What can be clearly seen in this figure is that the precision value of the algorithm keeps improving and remains above approximately 0.9. When the attack size is small, the precision values under bandwagon attack are relatively low compared to random and average attack. As the experiments continued to increase the attack size, the precision of the algorithm gradually improves and remains at a high level.

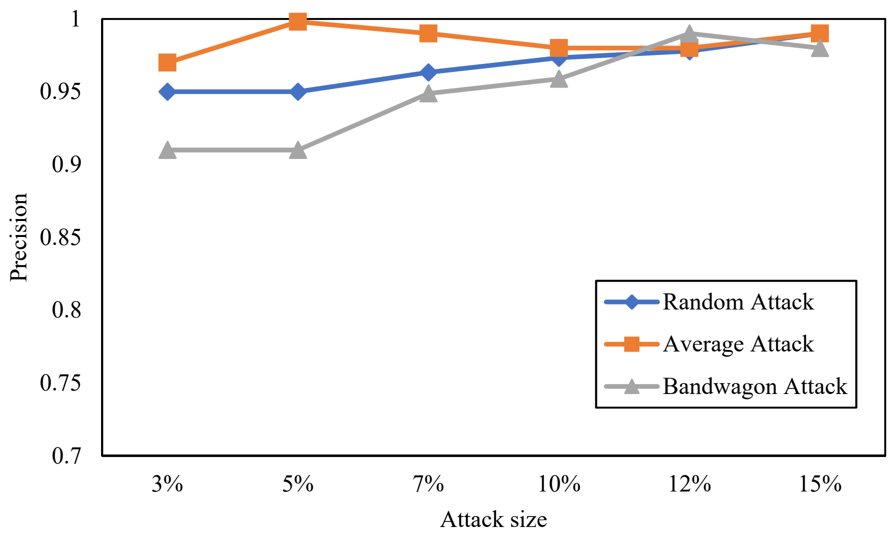

Figure 3. Precision of the three attack models when filler size is 10%.

The Figure 4 below illustrates an overview of the variation in recall values. When the size of the attack is small, the recall value of the algorithm under the random attack model is relatively low. The detection of the method under the random attack model is relatively poor compared to the other two models.

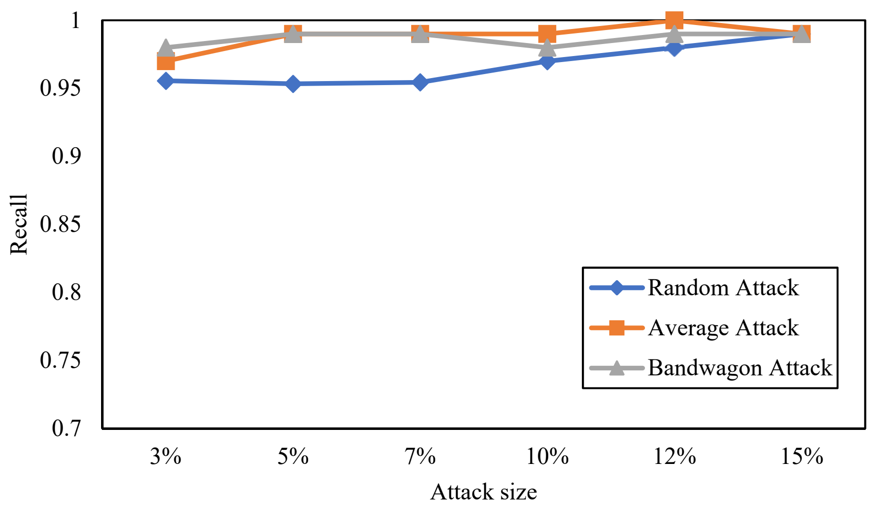

Figure 4. Recall of the three attack models when filler size is 10%.

The Figure 5 below shows the trend of F1 values under three attack model. What can be clearly seen in this figure is that the F1 value of the method keeps improving and remains above approximately 0.94. When the attack size is small, the F1-measure under the bandwagon attack model is relatively low. As the value of the attack size increases, the F1-Measure under the three attack models gradually increase and reach the same value. The F1 value represents the comprehensive detection effect of a model. So based on the experiments in Figure 5, we can conclude the following. The comprehensive detection effectiveness of the algorithm is better under average attack.

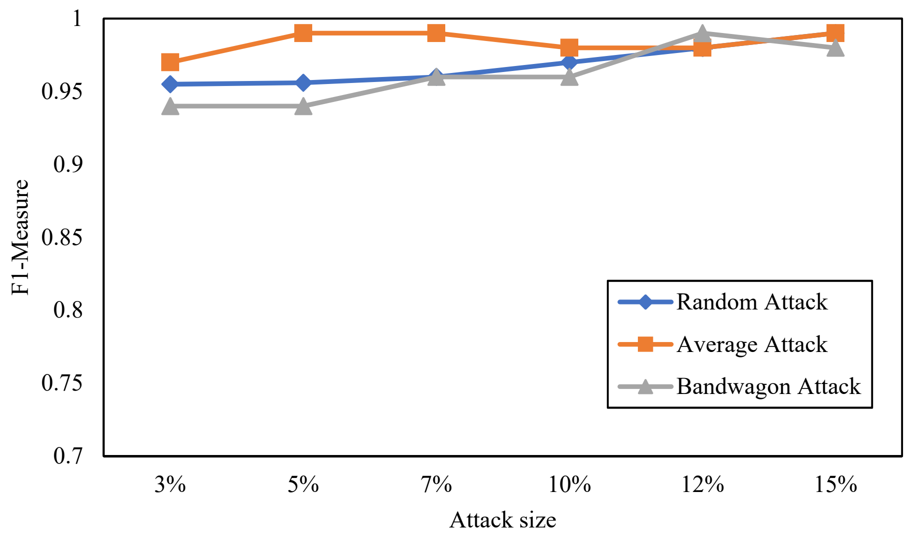

Figure 5. F1-Measure of the three attack models when filler size is 10%.

In the above section, several sets of experiments were carried out and the results were summarized and analyzed. These experiments confirm the relatively good detection performance of our method even when the size of the attack is small. At the same time, the overall detection effectiveness of our approach can be maintained at a relatively high level in the face of three different attack models.

5.3.2. Detection performance of comparative methods

This section tests the performance of several comparison algorithms under the attack models with the experimental parameters of filler size (3, 7, 10 and 15%) and attack size (3, 5, 7, 10, 12 and 15%). For a clear analysis of the experimental results, we have divided the following figures of the experimental data into two parts with filler sizes fixed at 3 and 10%. Figures 6–8 show one set of comparative experiments and Figures 9–11 show another.

1) When the fill size is fixed at 3%, the experimental results are as follows.

Figure 6 illustrates the F1 values for several methods under the random attack model. When the size of the attack is small, the F1 value of the XGB-SAD method is higher than the other methods, which indicates that the detection effectiveness of our method is better at this time. With the increase in attack size, the curve of BAY-SAD shows a small fluctuation, while the F1 values of all other methods are steadily increasing. It can be seen that the XGB-SAD method is more effective in detecting random attack.

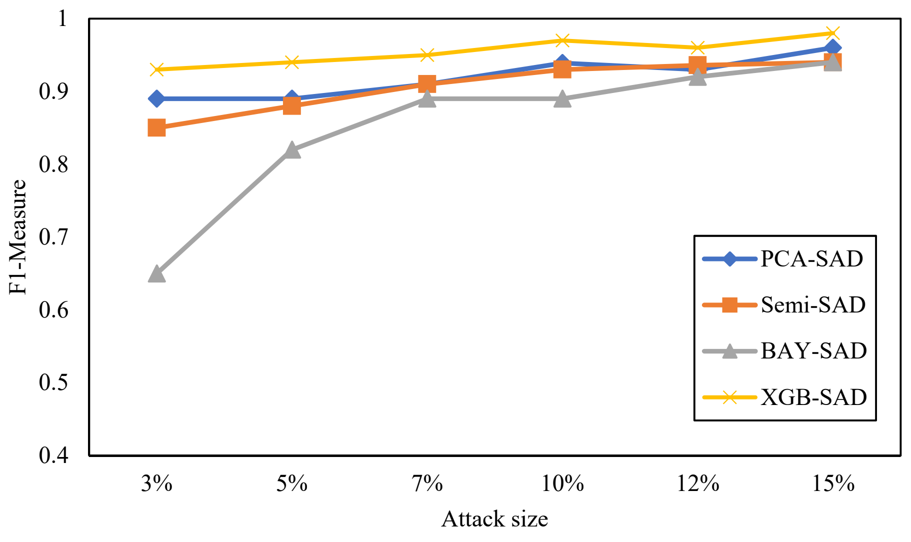

Figure 6. Performance of different methods for random attack when fill size is 3%.

The Figure 7 below illustrates the F1 values for several methods under average attack model. When the size of the attack is small, the F1 value of the XGB-SAD method is higher than the other methods. However, the F1 value of the BAY-SAD method is lower, which indicates that the BAY-SAD method is less effective in attack detection at this time under the average attack. With the increase in attack size, the curve of BAY-ASD and Semi-SAD show small fluctuations, while the F1 values of the other methods gradually increase converging to the same. As can be seen from the Figure 7, our approach is more effective in terms of the size of small-scale attacks and overall detection effectiveness.

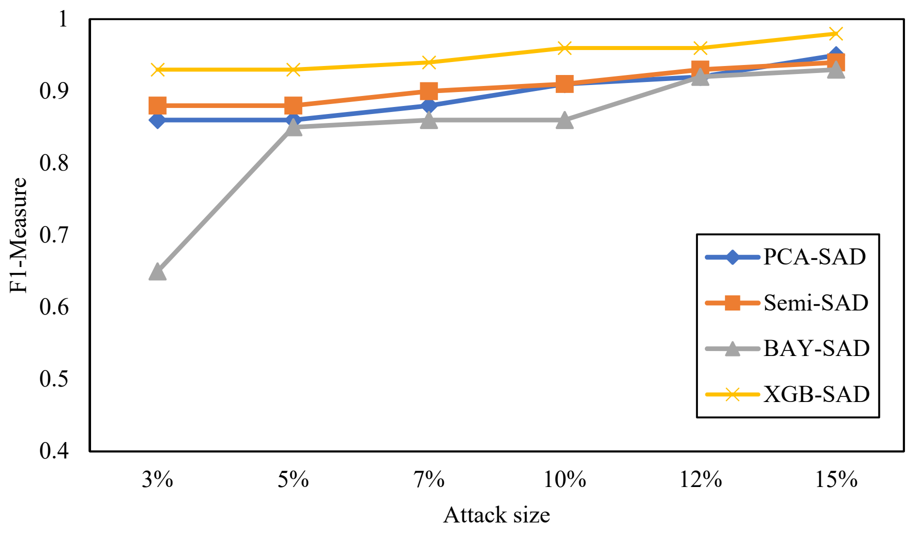

Figure 7. Performance of different methods for average attack when the fill size is 3%.

Figure 8 illustrates the F1 values for several methods under bandwagon attack model. When the size of the attack is small, the F1 are lower for the BAY-SAD method. From the graph above we can see that the bandwagon attack model has added selected items compared to the above two basic attack models, which affects the detection performance of the method.

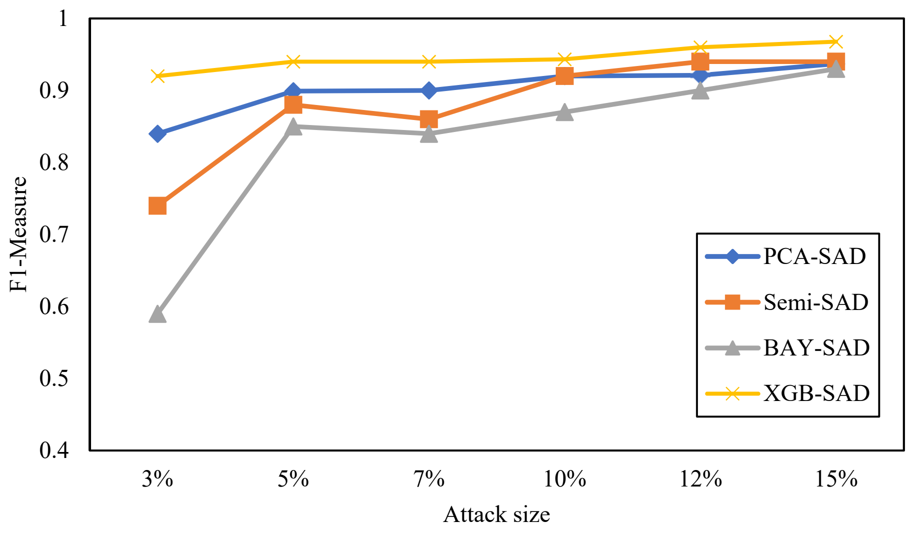

Figure 8. Performance of different methods for bandwagon attack when fill size is 3%.

2) When the filler size is fixed at 10%, the experimental results are as follows.

The Figure 9 below illustrates the F1 values for several methods under the random attack model. When the size of the attack is small, the F1 value of the XGB-SAD method is higher than the other methods. With the increase in attack size, the curve of BAY-SAD shows a small fluctuation, while the F1 values of all other methods are steadily increasing. It can be seen that the XGB-SAD method is more effective in terms of the size of small attack size and overall detection effectiveness.

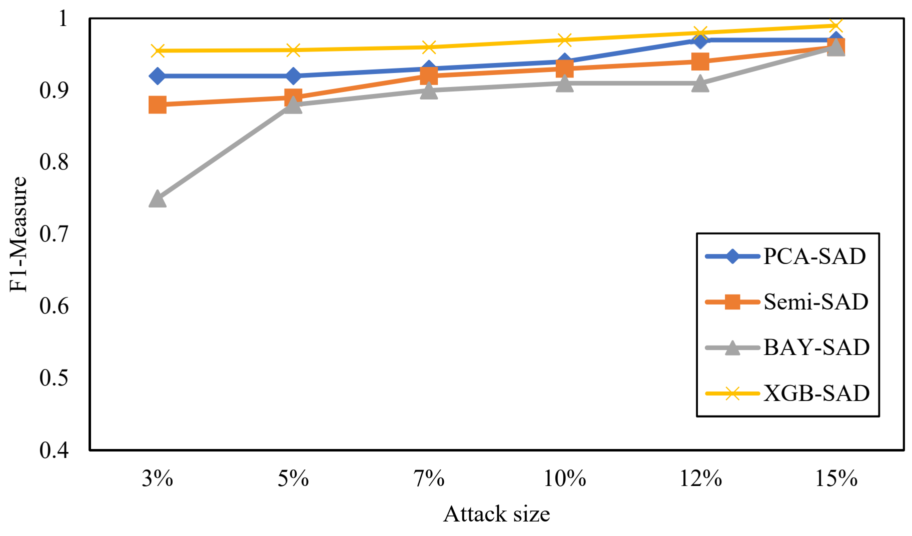

Figure 9. Performance of different methods for random attack when filler size is 10%.

Figure 10 illustrates the F1 values for several methods under the average attack. When the size of the attack is small, the F1 values are higher for the XGB-SAD method and lower for the BAY-SAD method. With the increase in attack size, the curve of Semi-SAD shows small fluctuations, while the F1 values of the other methods gradually increase converging to the same. From the figure below we can see that the XGB-SAD method is more effective in detecting average attack.

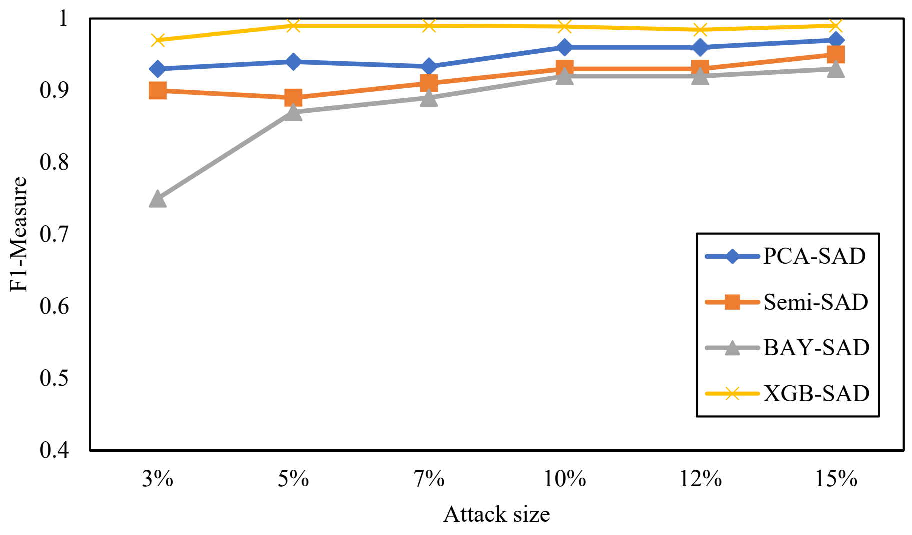

Figure 10. Performance of different methods for average attack when filler size is 10%.

The Figure 11 shows the F1 values for Comparison algorithms under bandwagon attack model. When the size of the attack is small, the F1 value of the XGB-SAD method is higher than the other methods. With the increase in attack size, the curve of BAY-SAD shows small fluctuations. As shown in figure below, the XGB-SAD method is more effective in detecting bandwagon attack.

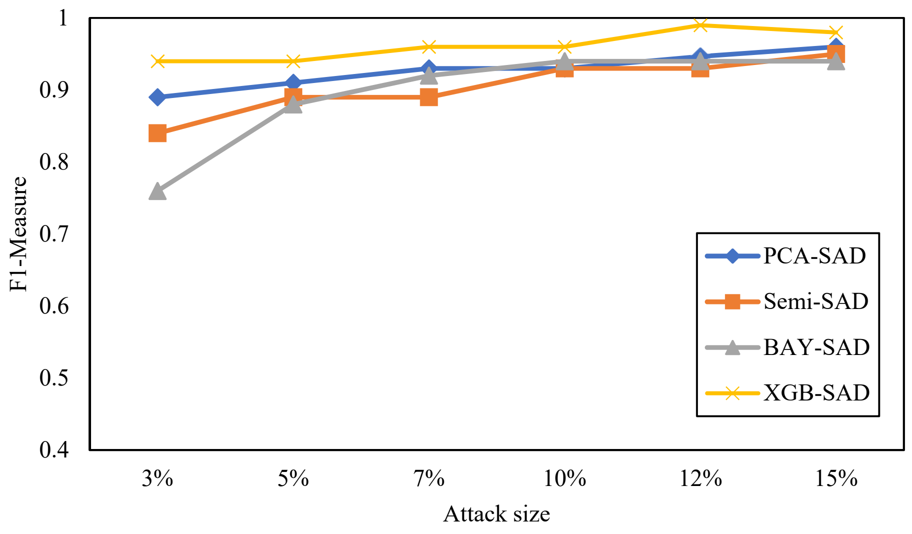

Figure 11. Performance of different methods for Bandwagon Attack when filler size is 10%.

6. Conclusions

To counter the security threat posed by the shilling attack, many detection methods are proposed. The XGB-SAD improves on the limitations of past methods that mainly used a single view and a single classifier to analyze the user rating values. Our method analyzes the rating matrix using a doubleview of user rating time and user rating item, which in turn defines the TPUS collection. Then we perform heuristic iterative optimization of the model's objective function using eXtreme gradient boosting and integrate multiple sets of base classifiers into a strong classifier using the idea of ensemble learning. Finally, the strong classifier generated in the previous stage are used to complete the identification and detection of malicious shilling attackers. The experimental results show that XGB-SAD outperforms the comparison methods in terms of overall detection and detection precision and at small attack size.

In the future, our research has two directions, the first of which is the combination of external variables and high-performance classifiers with different settings base classifiers in our approach. Another is that although our method can theoretically be extended to other datasets, balancing the sparsity of different datasets and improving the generalizability of the method in the face of sophisticated attack models remains a key concern.

Acknowledgments

This work is supported by the University Nursing Program for Young Scholars with Creative Talents in Heilongjiang Province (No. UNPYSCT-2018208).

Conflict of interest

The authors declared that they have no conflicts of interest to this work. We declared that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. K. Wang, Z. Zou, Q. Deng, R. Wu, J. Tao, C. Fan, Reinforcement learning with a disentangled universal value function for item recommendation, in *Proceedings of the AAAI conference on artificial intelligence*, **35** (2021), 4427–4435. https://doi.org/10.48550/arXiv.2104.02981
2. R. Yu, Y. Gong, X. He, B. An, Y. Zhu, Q. Liu, et al., Personalized adaptive meta learning for cold-start user preference prediction, preprint, arXiv:2012.11842. https://doi.org/10.48550/arXiv.2012.11842
3. A. Javari, Z. He, Z. Huang, R. Jeetu, C. C. Chang, Weakly supervised attention for hashtag recommendation using graph data, in *Proceedings of The Web Conference*, (2020), 1038–1048. https://doi.org/10.1145/3366423.3380182
4. C. Tong, X. Yin, J. Li, T. Zhu, R. Lv, L. Sun, et al., A shilling attack detector based on convolutional neural network for collaborative recommender system in social aware network, *Comput. J.*, **7** (2018), 949–958. https://doi.org/10.1093/comjnl/bxy008
5. C. Rami, O. S. Shalom, D. Jannach, A. Amir, A black-box attack model for visually-aware recommender systems, in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, (2021), 94–102. https://doi.org/10.1145/3437963.3441757

6. Y. J. Hao, P. Zhang, F. Z. Zhang, Multiview ensemble method for detecting shilling attacks in collaborative recommender systems, *Secur. Commun. Netw.*, **2018** (2018). https://doi.org/10.1155/2018/8174603
7. P. A. Chirita, W. Nejdl, C. Zamfir, Preventing shilling attacks in online recommender systems, in *Proceedings of the 7th annual ACM international workshop on Web information and data management*, (2005), 67–74. https://doi.org/10.1145/1097047.1097061
8. R. Burke, B. Mobasher, C. Williams, R. Bhaumik, Classification features for attack detection in collaborative recommender systems, in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2006), 542–547. https://doi.org/10.1145/1150402.1150465
9. C. A. Williams, B. Mobasher, R. Burke, Defending recommender systems: detection of profile injection attacks, *Serv. Oriented Comput. Appl.*, **1** (2007), 157–170. https://doi.org/10.1007/s11761-007-0013-0
10. T. Tong, Y. Tang, An effective recommender attack detection method based on time SFM factors, in *2011 IEEE 3rd International Conference on Communication Software and Networks*, (2011), 78–81. https://doi.org/10.1109/ICCSN.2011.6013780
11. H. Xia, B. Fang, M. Gao, H. Ma, Y. Tang, J. Wen, A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique, *Inf. Sci.*, (2015), 150–165. https://doi.org/10.1016/j.ins.2015.02.019
12. Z. Yang, L. Xu, Z. Cai, Z. Xu, Re-scale AdaBoost for attack detection in collaborative filtering recommender systems, *Knowl.-Based Syst.*, **100** (2016), 74–88. https://doi.org/10.1016/j.knosys.2016.02.008
13. Z. A. Wu, Y. Zhuang, Y. Q. Wang, J. Cao, Shilling attack detection based on feature selection for recommendation systems, *Chin. J. Electron.*, **8** (2012), 1687. https://doi.org/10.3969/j.issn.0372-2112.2012.08.031
14. W. T. Li, M. Gao, H. Li, J. Zeng, Q. Xiong, S. Hirokawa, Shilling attack detection in recommender systems via selecting patterns analysis, *IEICE Trans. Inf. Syst.*, **99** (2016), 2600–2611. https://doi.org/10.1587/transinf.2015EDP7500
15. Z. Wu, J. Wu, J. Cao, D. Tao, Hysad: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation, in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2012), 985–993. https://doi.org/10.1145/2339530.2339684
16. B. Mehta, Unsupervised shilling detection for collaborative filtering, in *2007 National Conference on Artificial Intelligence*, **2** (2007), 1402–1407. https://dl.acm.org/doi/10.5555/1619797.1619870
17. Z. Yang, Z. Cai, X. Guan, Estimating user behavior toward detecting anomalous ratings in rating systems, *Knowl.-Based Syst.*, **111** (2016), 144–158. https://doi.org/10.1016/j.knosys.2016.08.011
18. S. Zhang, A. Chakrabarti, J. Ford, F. Makedon, Attack detection in time series for recommender systems, in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2006), 809–814. https://doi.org/10.1145/1150402.1150508
19. G. S. Oestreicher, A. Sundararajan, Recommendation networks and the long tail of electronic commerce, *MIS Q.*, **36** (2012), 65–83. https://dx.doi.org/10.2139/ssrn.1324064
20. T. Q. Chen, G. Carlos, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, (2016), 785–794. https://doi.org/10.1145/2939672.2939785

21. F. M. Harper, J. A. Konstan, The movielens datasets: History and context, *ACM Trans. Interact. Intell. Syst.*, **5** (2015), 1–19. https://doi.org/10.1145/2827872
22. J. Cao, W. Zhang, B. Mao, Y. Zhang, Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system, *World Wide Web,* **16** (2013), 729–748. https://doi.org/10.1007/s11280-012-0164-6
23. W. Bhebe, O. P. Kogeda, Shilling attack detection in collaborative recommender systems using a meta learning strategy, in *2015 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, (2015), 56–61. https://doi.org/10.1109/ETNCC.2015.7184808