UDC 925.17

On improving the reliability of recommender systems with users clustering

Eugene Yu. Shchetinin

Department of Data Analysis, Decision Making and Financial Technologies
Financial University under the Government of the Russian Federation
Leningradsky pr. 49, Moscow, 117198, Russia

Email: riviera-molto@mail.ru

The growing popularity of E-commerce services is increasingly attracting the attention of their users to the services of recommendation systems. Collaborative filtering is one of the well-known recommendation methods that help customers select possible products of interest. However, recommendation systems are open to malicious attacks to promote or discredit certain products. They create and implement in the system of recommendations fake user profiles, the so-called shilling attacks, causing a significant change in the ratings of products in the social network, thereby causing significant material and moral damage. This paper presents the characteristics of shilling attacks, describes the main models and their parameters, as well as the main, the most important methods of detecting fake profiles. We propose an algorithm of recommendations based on K-means clustering as a sustainable method of countering shilling attacks in social networks. Its resistance to shilling attacks, the most popular among attackers, are investigated, and the influence of various attack parameters on the results of its work is analyzed. Computer simulation of intrusions and analysis of their impact on the forecast of recommendations showed that the algorithm is resistant to shilling attacks without significant impact of introduced malicious profiles on the work of the recommendation system.

Key words and phrases: Recomendations, collaborative filtering, shilling attack, clustering, k-means.

In: K. E. Samouylov, L. A. Sevastianov, D. S. Kulyabov (eds.): Selected Papers of the IX Conference "Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems", Moscow, Russia, 19-Apr-2019, published at http://ceur-ws.org

1. Introduction

With the increasing in amount of information available in everyday life due to the widespread use of the Internet, social networks became one of the popular tools for the collection and analysis of the necessary information. Recomendation systems (RS) algorithms are effective in automating people's habits by collecting information about their preferences in products, goods and services such as movies, music CDs, books, etc. As a rule, RS are a user-product matrix containing product ratings compiled by users as a result of either a purchase or recommendations received from other users [1]. Whenever a user requests a rating for a product of interest to him, the system builds it in the form of a weighted average of ratings of other users of this product. Modern RS are not yet able to reliably distinguish the profiles of genuine users from malicious ones, which makes them vulnerable to manipulation by unscrupulous users. Malicious users or competing companies may invade databases to artificially increase or decrease the popularity of a particular product. This process was first described in paper [2] and is terminologically defined as a shilling attack. Identification of fake profiles and resistance to them are crucial properties for the success of recommendation systems. All this made it necessary to develop an effective algorithm of the RS to create personalized forecasts of recommendations with high accuracy in the conditions of inevitable attacks on the intrusion of fake profiles in the recommendation systems.

With the increasing popularity of RS there are a number of models attacking mechanisms for performing manipulations on the recommendations in favor of specific products. Shilling attacks on the system are generated by introducing fake profiles into user databases. The General attack strategy is shown in the Fig. 1. In the literature, shilling's attacks are classified in two ways: attacks aimed at improving the rating of the product chosen by attackers (push attack) as an object of attack, and attacks aimed at lowering its rating (nuke attack) [3, 4]. To increase the forecast, i.e. to increase the popularity of some product, the attackers form a new profile of the RS user , who assigns him a high rating by his vote in the network. To reduce the popularity of the target product, it is assigned a low rating. Attackers generate fake profiles, assign maximum or minimum rating values to their target products in accordance with their intentions and enter them into databases, thus manipulating recommendations in the system in their favour. A significant number of works are devoted to the use of clustering methods to solve the problem. The successful application of clustering methods in shilling attack detection algorithms inspired us to the hypothesis that the K-means clustering method can be proposed as a reliable prediction algorithm [4–6]. In addition to detecting malicious profiles, we suggest that the clustering method can be used to develop algorithms to build a reliable rating prediction for recommended products. In addition, repeated application of the clustering algorithm should eliminate all shilling profiles after some level of the constructed binary tree.

2. Models and properties of shilling attacks

The most famous strategy of the attacker on the introduction of fake profiles to a user group of the system is as follows [7, 8]. First of all, this user tends to more frequent mention of some product in the system. It is obvious that for this purpose it is necessary to change the predicted rating value of this product for as many users as possible. An effective shilling attack makes this value as high as possible. A measure of the effectiveness of the attack can be the value of the bias of the forecast rating of the target product. It is defined as the difference in the predicted value of the rating before and after the attack [9].

$$P = \sum_{u} \widetilde{r}_{u_i,y} - r_{u_i,y}. \tag{1}$$

$\mathbf{I_T}$			$\mathbf{I_S}$			$\mathbf{I_F}$			$\mathbf{I_N}$		
i_1^T	…	i_j^T	i_1^S	…	i_k^S	i_1^F	…	i_l^F	i_1^N	…	i_v^N
$\gamma(i_1^T)$	…	$\gamma(i_j^T)$	$\sigma(i_1^S)$	…	$\sigma(i_k^S)$	$\rho(i_1^F)$	…	$\rho(i_l^F)$	null	…	null

Figure 1. Shilling attacks general structure.

where $\widetilde{r}_{u_i,y}$ denotes the product rating forecast y for the user P_u after the attack, P_u– the forecast of the product rating shift y by users. Thus, the task of the attack is to maximize P. The prediction itself $\widetilde{r}_{u_i,y}$ can be calculated as follows

$$\widetilde{r}_{u_i,y} = \overline{r}_i + \frac{\sum_{u_j \in N}(r_{u_j,y} - \overline{r}_{u_i})}{\sum_{u_i \in N} C_{u_j,u_i}}, \tag{2}$$

where $r_{u_j,y}$ - product rating y for user u_j, $\overline{r_{u_i}}$, $\overline{r_{u_j}}$, – average ratings for users u_j, u_i; N-the set of the nearest neighbors of user u_i , C_{u_i,u_j} - similarity measure between users u_i, u_j, which is selected as Pearson correlation

$$C_{u_i,u_j} = \frac{\sum_y (r_{u_i,y} - \overline{r}_{u_i})(r_{u_j,y} - \overline{r}_{u_j})}{\sqrt{\sum_y (r_{u_i,y} - \overline{r}_{u_i})^2 (r_{u_j,y} - \overline{r}_{u_j})^2}}. \tag{3}$$

Note, that correlation (3) is calculated only for those products for which there are ratings from both users u_i, u_j. Other similarity measures should also be noted, for example, cosine [10]. On Fig. 1 the general structure of the shilling attack on the recommendation systemare is shown. The composition of the elements of a shilling attack is given below.

I_T: a set of target products consisting of one or more names, called a single or multiple attack. Rating $\gamma(i_j^T)$, usually defined as the maximum (minimum) value of the target product profiles in the network, depending on the type of attack (push/nuke);

I_S: a set of selected items with a rating defined by the function $\sigma(i_k^s)$;

I_F: a set of filler items obtained randomly from selected products with a random assigned ratings $\rho(i_l^F)$;

I_N: a Set of items with no rating.

Next, we consider the basic models of shilling attacks and describe their properties.

Random Attack (RN). The selected set of target products is empty, and the randomly selected set of placeholder elements contains random values derived from the normal distribution with the mean and standard deviation over the system. The target element is assigned the maximum rating available in the recommendation system. $I_S = \oslash$ and $\rho(i) \sim N(\overline{r}, \overline{\sigma}^2)$.

Average attack (AV). A more effective push attack model is aimed at the individual average rating of each item, rather than the average value of the entire recommendation system. The cost of this attack is related to the number of filler items in the attack profile, as knowledge of the average number of votes for such products is required. The selected set of I_S elements is empty, so that each randomly selected filler element is filled with a random value derived from the normal distribution with the average rating of the corresponding element and the standard deviation for recommendation system attacks. The target element is assigned the maximum rating available in the system: $I_S = \oslash$, $\rho(i) \sim N(\overline{r}_i, \overline{\sigma}_i^2)$.

Grouped Attack (BandWagon BW). As a shilling attack model of the push type, the bandwagon attack targets products that attract the special attention of many

consumers in order to manipulate people who are prone to purchase such products. The selected set of products consists of popular and well-priced items with high average values. For recommendation system attacks, the selected items are assigned the highest available rating, the content items are assigned random values, and the target item is assigned the highest rating. I_S contains popular names, $\sigma(i) = r_{max} \backslash r_{min}$ (push/nuke), $\rho(i) \sim N(\overline{r}_i, \overline{\sigma}_i^2)$.

Reverse Bandwagon Attack (RBW). I_S contains a set of unpopular items, $\sigma(i) = r_{min} \backslash r_{max} (push/nuke)$, $\rho(i) \sim N(\overline{r}, \overline{\sigma^2})(push/nuke)$.

Segmented attack (SG). A SG attack is designed as a model of push attacks aimed at a group of users who are more likely to purchase certain types of products than other users on the network. The composition of the shilling attack group I_S is chosen from elements with a high average rating with a certain property (for example, horror movies or jazz music). These selected items are assigned a maximum rating value, filler items are assigned a minimum rating value, and the target product is assigned the highest vote to increase its popularity. I_S contains a set of segmented products, $\sigma(i) = r_{max} \backslash r_{min}$ (push/nuke), $\rho(i) = r_{min} \backslash r_{max}$ (push/nuke).

Love / hate attack (L/H). Simple model of attack nuke, which does not require any knowledge about the system. TheI_S set of selected items is empty, and randomly selected filler items are assigned the highest available rating values, while the target item receives the minimum number of votes:$I_S = \oslash$, $\rho(i) = r_{max} \backslash r_{min}$ (push/nuke).

The main models of shilling attacks given above and their characteristics allow us to formulate some metrics that allow us to distinguish fake profiles from genuine users of the system.

3. Metrics of similarity and difference of the genuine and fake users

Let us consider the methods of detection of fake profiles in the structure of recommendation systems using different metrics. First these metrics were presented in [10, 11, 13].

1) Rating Deviation from Mean Agreement (RDMA). Measures the discrepancy between user u's rating and other genuine users in the system, inversely proportional to the number of products that user u rated

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - Avg_i|}{NR_i}}{N_u}, \tag{4}$$

where Avg_i is the average rating of product i, N_u is the number of products that user u rated.

2) Weighted Deviation from Main Agreement (WDMA) is based on RDMA, but it gives more weight to rating deviations for sparse elements [6]. It is defined as

$$WDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - Avg_i|}{NR_i^2}}{N_u}, \tag{5}$$

where $r_{u,i}$ is the rating of product i assigned by user u, NR_i the total range of ratings in the system assigned to item i.

3) Degree of similarity (DegSim). It is based on the hypothesis that the profiles of attackers are very similar to each other because of their characteristics and are generated in the same way.

$$DigSim_u = \frac{\sum_{v \in NB(u)} W_{u,v}}{k},$$

where $W_{u,v}$ is the similarity measure of the user u and k of his nearest neighbors u, for which we have chosen the correlation coefficient (3)

$$W_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r}_u)^2 (r_{v,i} - \overline{r}_v)^2}}.$$

4) Similarity of connections, relationships between users: the similarity between two users can also be measured by their common connections with other participants or similar recommendations, in general, all that is called common interests. A greater number of mutual ties would mean a greater similarity between them. For fake links, it is expected that the number of mutual links is small, and therefore the similarity between users is small. We determine the similarity of such connections as

$$C(u,v)\frac{|N(u) \bigcap N(v)|}{|N(u)|}.$$

Now we will look at fake users. Their properties are discussed in detail in [11], [15], [16].

5) Extreme rating behavior Indicator: Users assign only high ratings (for example, five) or low ratings (for example, 1) to products. We characterize such ratings of user u as

$$S(u) = 1 - \frac{\sigma(R_u)}{R_h - R_l}, \tag{6}$$

where $\sigma(R_u)$ is the standard deviation of all ratings for user u, R_h, R_l is the highest and lowest ratings in the system. In extreme user behavior, the $E(u)$ value will be close to 1.

6) also, the extreme behavior of the user can be characterized by an indicator of the variability of its ratings

$$B_j(u) = \frac{\sum_{v \in N(u)} |r_{u,j} - r_{v,j}|}{N(u)}.$$

4. Our approach for detecting shilling attacks

Our aproach to improving shilling detection consists in the next steps. First of all, the algorithm forms a user-product matrix $U_{n,m}$ where n and m are the number of users and products respectively. Typically, matrix U is a highly sparsed matrix consisting of ratings r_{u_i} (likes, purchase facts, etc.) that users u have assigned to products y. The task of detecting attackers is essentially reduced to the task of binary classification based on the differences in some properties of the key features of user profiles. The search for these differences is based on the designing of new features for each user, thus, the classification procedure becomes easier. The nature of these features can be different, both statistical and heuristic, for example, median, deviation or more complex features such as specific characteristics, based on attack models.

The main goal of the RS is to build a rating forecast $\widetilde{r}_{i,y}$ for the product y, which must be assigned to it by user u_i in case of purchase him the product y. It is estimated by other users to predict what rating a given user will assign to a product y, given with more weight those users that are more like this one. Weights are defined from (3), and the forecast that a given user will assign the target product is defined as (2). Usually, instead of taking into account all users from U, only N nearest neighbors would be selected N users who are most similar to this user u_i and have already rated this product. The criteria by which these N users will be formed we formulated above as expressions (4),(5),(6). This step is a weak chain of most RS algorithms, explored $K - nn$, SVM and other suprevised algorithms of machine learning [13, 15, 17]. These

studies showed unsatisfatory results because as usual filler or size attack parameters are small, so the considered problem caused by imbalanced sample. To improve the detection of shilling attacks we proposed some especially designed features to make classification fake and genuine users more easier task. Next step is concerned the implementation of the numerical method as we use the gradient boosting algorithm GBM [18, 19].

5. Computer simulations of the shilling attacks and recommender systems reliability analysis

To test the constructed algorithm was used a dataset with opinions on the films MovieLens, developed by GroupLens. MovieLens-100K was collected by the GroupLens Research Project at the University of Minnesota [12]. It contains ratings from 943 users on 1682 film and the range of ratings is from 0 to 5 points. We also conducted a computer analysis of the results of shilling attacks on two parameters: the Filler Size I_F (FS) and the Attack Size I_T (Attack Size). The FS parameter determines the percentage of cells that will be filled with fake ratings when creating attack profiles. The size of the attack (SA) can be described as the number of fake profiles proportional to the number of users in the database. We used the forecast shift metric (1) to estimate the changes in the forecast caused by shilling effects. Some programs that implement our algorithms in Python,are given in the appendix.

The shilling attacks were embedded in two separate sets of 100 movies for push and nuke attacks. These push and nuke attack groups were built randomly from different rating ranges to represent the characteristics of the original dataset. Since it is unwise to artificially increase the popularity of a film with high ratings or similarly reduce the rating of an unpopular film, we have mainly chosen films with low average rating for push attacks and high average rating for nuke attacks. Then binary decision trees were built by introducing fake profiles into the system in the way described above. Then the estimates of forecasts are constructed on the basis of the obtained results of binary classification, and the estimates of the forecast shift (1) for various shilling attacks are calculated. The stopping criterion for constructing binary decision trees was set at 30. We presented the results for push and nuke attacks as follows.

Investigation of the effect of the shilling attack parameter Filter Size (FS). We conducted computer experiments to show how the value of the FS parameter affects the reliability of the product rating prediction using clustering with respect to four push models and three nuke shilling attacks. Recall that the (FS) parameter specifies the number of fake recommendations added to fill the I_F attack field. The size attack parameter is set to 15 percents, and the FS is ranges from 3 to 30 percents . The average values of the shift forecast for push, nuke attacks are shown in Fig 3,4, respectively. Analysis of the results of rating manipulation with these attacks (see 3) showed that none of the four push attack models used led to significant changes in the rating forecast for the considered range of changes in the FS parameter. The maximum forecast shift about 5 percents is observed for the average attack at FS=20 percents. Compared to random and group attacks, average and segment attacks are more effective. We assume that their impact on the stability of the system is still insignificant, since the maximum shift of the forecast is only about 5 percents. If you increase the FS values from 3 to 20 percents, there is a noticeable increase in the forecast shift values for the average attack. Thus, we can conclude that the proposed algorithm for predicting the product rating based on the N-means clustering method is resistant to push attacks on recommendation systems.

The calculation results presented in Fig. 3 showed that the model of nuke attacks are ineffective for attacks of recommendation systems within the specified range of variation of the parameter FS $(0; 25)$ percents. Changes in the values of the forecast P after L/H attacks with an increasing the filler size are insignificant. So we aprove that L/ H attack is completely ineffective. Unlike attack L/H, RW - attack does some damage, and therefore more effective than the charge of L/H. However, the maximum shift value

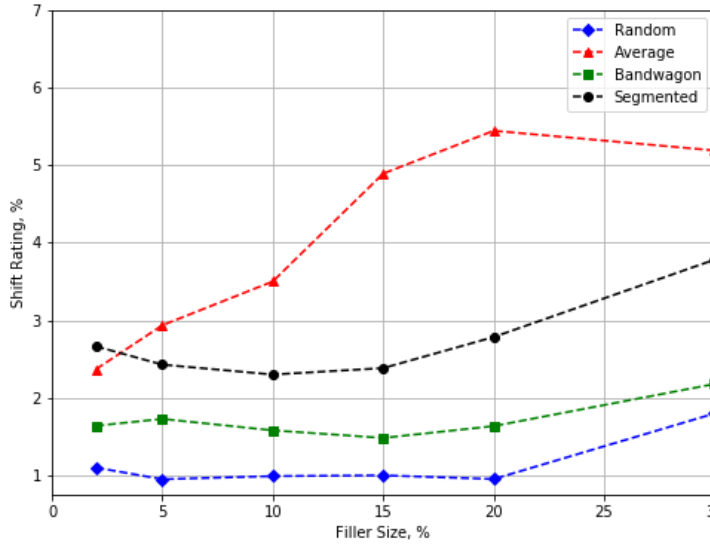

Figure 2. Graphs of forecast for shift rating depending of the attacked product on the Filler Size parameter for different push attack models. $SA = 15$ **percents.**

forecast gives an average attack of around 15 percents. However, such changes can be considered insignificant in relation to the ratings of the target products.

Next, we presented the computer experiments to estimate the influence of the Attack Size parameter on the stability of the algorithm for four push attack models and three nuke attack models. To assess the stability of our algorithm to different values of the Attack Size parameter, we made the Filter Size is equal to 15 percents, and the size of the attack varies from 1 to 15 percents. We estimated the forecast shift values and displayed their overall averages for the push and nuke attack models in Fig. 4, Fig. 5, respectively. As you can see from both figures, the SA parameter is more effective in attacks than the FS parameter. Results in Fig.4 shows that the most effective models are average and random attacks.

Compared to average and random attacks, segmented and group attacks proved to be ineffective. Segmented and grouped attacks cause stable changes in forecasts with an increase in the size of the attack. Almost all values of the forecast shift size for such attacks are close to 6 percents, which is negligible. Thus, we can conclude that our algorithm is robust to them without causing significant shifts in rating forecast (1). Although the mean and random attacks cause manipulations, the maximum shift does not exceed 15 % when the SA parameter is equal to 3 %. With increasing the size of the attack from 6 percents up to 15 percents the value of the shift forecast for average and random attacks are reduced that we can comment as increasing of accuracy of the algorithm. These results demonstrate that the proposed algorithm is robust to push attacks for the considered Attack Size values.

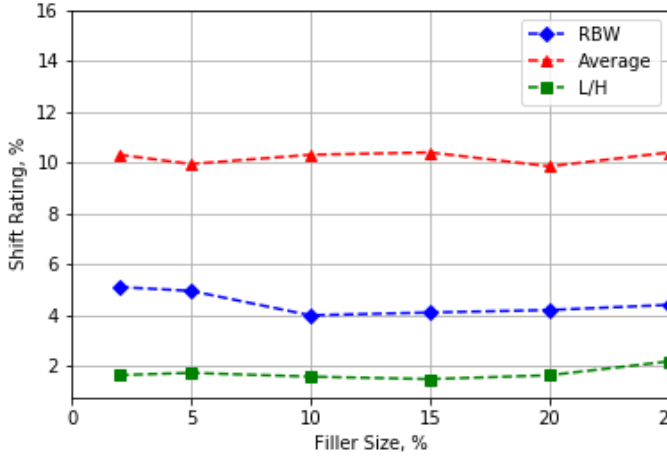

Figure 3. Graphs of forecast for shift rating of the attacked product depending on the parameter Filler Size for nuke attack models. $SA = 15$ **percents.**

From Fig. 5 it can be seen that the RBW attack is the most effective nuke shilling attack. When the value of the Attack Size parameter is 6 %, the forecast shift value for it reaches a maximum value about 7 %. Unlike RBW-attack L/H-attack almost ineffective. Thus, we can conclude that our algorithm is also robust to L/H attacks. In general, based on the extensive computer experiments we can say that the proposed algorithm is mainly robust to shilling attacks. We analyzed its reliability to six known shilling attacks (including models of both push and nuke attacks). Most of the investigated attacks proved to be ineffective, since the obtained estimates of the rating shift of the product to which these attacks were directed, as a rule, do not exceed 15 %. In most cases, a love/hate attack causes almost zero shifts. In some cases, the values of shift rating reached 16 %, but still it is acceptable in comparison with the range of rank values. The average push attack was the most effective attack against our algorithm.

6. Conclusions

The paper studies the most popular methods of manipulation of user behavior in recommendation systems, and also proposes an effective computer algorithm that can significantly reduce their impact on the forecast rating of the recommended product. This algorithm is based on the K-means clustering method with modification of cluster centers selection. To test the proposed method, computer simulation of shilling attacks and analysis of their impact on the estimates of the ratings of recommended movies from the MovieLens database were carried out. The results of computer experiments showed that the models of shilling attacks were not able to significantly change the forecasts of the rating of recommendations using the proposed clustering algorithm of fake users of the recommendation system.

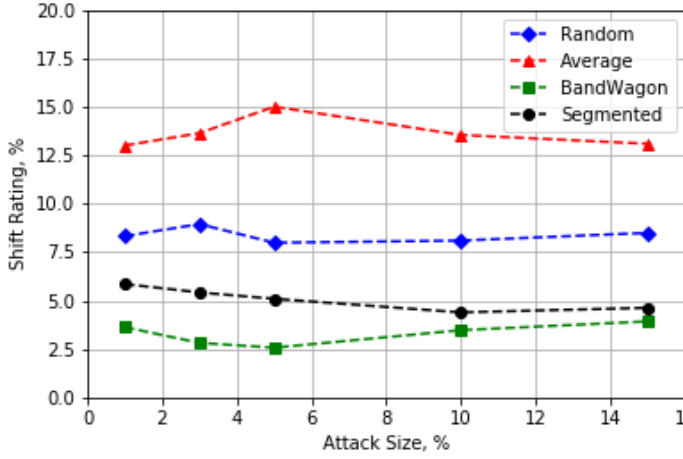

Figure 4. Graphs of forecast for shift rating of the attacked product depending on the parameter Attack Size for different push-attack models. Filler Size=15 percents

7. Program Code

PYTHON code for collaborative filtering modeling

```
#This function finds k similar users given the user_id and ratings matrix M
#Note that the similarities are same as obtained via
#using pairwise_distances
def findksimilarusers(user_id, ratings, metric = metric, k=k):
    similarities=[]
    indices=[]
    model_knn = NearestNeighbors(metric = metric, algorithm = 'brute')
    model_knn.fit(ratings)

    distances, indices = model_knn.kneighbors(
    ratings.iloc[user_id-1, :].values.reshape(1, -1), n_neighbors = k+1)
    similarities = 1-distances.flatten()
    print ('{0} most similar users for User {1}:\n'.format(k,user_id))
    for i in range(0, len(indices.flatten())):
        if indices.flatten()[i]+1 == user_id:
            continue;

        else:
 print ('{0}: User {1}, with similarity of {2}'.format(i,
 indices.flatten()[i]+1, similarities.flatten()[i]))

#This function predicts rating for specified user-item combination based
#    on user-based approach
```

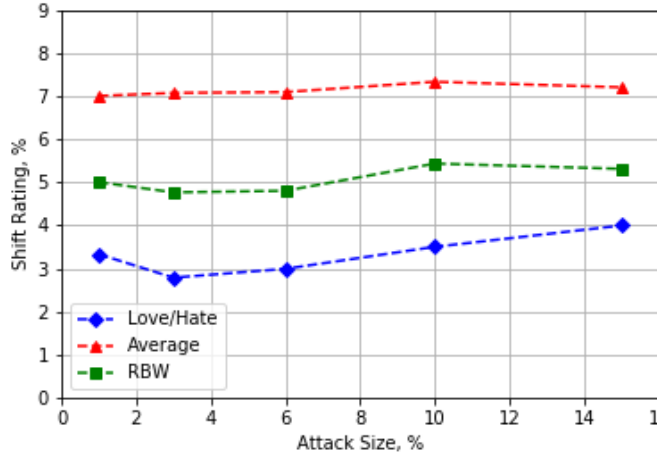

Figure 5. Graphs of the shift of the rating forecast of the attacked product depending on the parameter Attack Size for different nuke-attack models: 1 - L/H, 2-RBW, 3-AV.

```
def predict_userbased(user_id, item_id, ratings, metric = metric, k=k):
    prediction=0
    similarities, indices=findksimilarusers(user_id, ratings,metric, k)
   #similar users based on cosine similarity
    mean_rating = ratings.loc[user_id-1,:].mean()
   #to adjust for zero based indexing
    sum_wt = np.sum(similarities)-1
    product=1
    wtd_sum = 0

    for i in range(0, len(indices.flatten())):
        if indices.flatten()[i]+1 == user_id:
            continue;
        else:
            ratings_diff = ratings.iloc[indices.flatten()[i],
            item_id-1]-np.mean(ratings.iloc[indices.flatten()[i],:])
            product = ratings_diff * (similarities[i])
            wtd_sum = wtd_sum + product

    prediction = int(round(mean_rating + (wtd_sum/sum_wt)))
    print ('\nPredicted rating for user {0} ->
    item {1}: {2}'.format(user_id,item_id,prediction))
    return prediction

#This function finds k similar items given the item_id
#and ratings matrix M
```

```
def findksimilaritems(item_id, ratings, metric=metric, k=k):
    similarities=[]
    indices=[]
    ratings=ratings.T
    model_knn = NearestNeighbors(metric = metric, algorithm = 'brute')
    model_knn.fit(ratings)

 distances, indices =
model_knn.kneighbors(ratings.iloc[item_id-1, :].values.reshape(1, -1),
n_neighbors = k+1) similarities = 1-distances.flatten()
    print '{0} most similar items for item {1}:\n'.format(k,item_id)
    for i in range(0, len(indices.flatten())):
        if indices.flatten()[i]+1 == item_id:
            continue;

        else:
  print ('{0}: Item {1} :, with similarity of {2}'.format(i,
indices.flatten()[i]+1, similarities.flatten()[i]))
    return similarities,indices

#This function predicts the rating for specified user-item combination
#based on item-based approach
def predict_itembased(user_id, item_id, ratings, metric = metric, k=k):
    prediction= wtd_sum =0
    similarities, indices=findksimilaritems(item_id, ratings)
#similar users based on correlation coefficients
    sum_wt = np.sum(similarities)-1
    product=1

    for i in range(0, len(indices.flatten())):
        if indices.flatten()[i]+1 == item_id:
            continue;
        else:
            product =
            ratings.iloc[user_id-1,indices.flatten()[i]] *
            (similarities[i])
            wtd_sum = wtd_sum + product
    prediction = int(round(wtd_sum/sum_wt))
    print '\nPredicted rating for user {0} ->
    item {1}: {2}'.format(user_id,item_id,prediction)

    return prediction
```

References

1. K. Deepak Agarwal, Chen Bee-Chung, Statistical Methods for Recommender Systems, Cambridge University Press, 2016.
2. R. Bhaumik, B. Mobasher, R. D. Burke, A clustering approach to unsupervised attack detection in collaborative recommender systems, IEEE international conference on data mining, p. 181–187, 2011.
3. R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, Classification features for attack detection in collaborative recommender systems, ACM SIGKDD, p. 542–547, 2006.
4. Z. Cheng, N. J. Hurley, Robust collaborative recommendation by least trimmed squares matrix factorization, IEEE conference on tools with artificial intelligence, p. 105–112, 2010.

5. C. Y. Chung, P. Y. Hsu, S. H. Huang, A novel approach to filter out malicious rating profiles from recommender systems, Decision Support Systems, vol. 55, No. 1, p. 314–325, 2013.
6. Z. Zhang, S. Kulkarni, Detection of Shilling Attacks in Recommender Systems via Spectral Clustering, 2014 17th International Conference on Information Fusion (FUSION), p.1-8, 7-10 July 2014.
7. P. Harrington, Machine Learning in Action, 2012.
8. B. Mehta, Unsupervised shilling detection for collaborative filtering. In AAAI, p. 1402-1407, 2007.
9. D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, Recommender Systems: An Introduction. Cambridge University Press, 2010.
10. P. B. Thorat, R. M. Goudar, and S. Barve, Survey on Collaborative Filtering and Content-Based Recommending, Int. J. Comput. Appl., vol. 110,4, p. 31–36, 2015.
11. C. Chung, P. Hsu, S. Huang, A novel approach to filter out malicious rating profiles from recommender systems, Journal of Decision Support Systems, p. 314–325, 2013.
12. http://grouplens.org/datasets/movielens/
13. C. Li and Z. Luo. Detection of shilling attacks in collaborative filtering recommender systems. In: Proceedings of the international conference of soft computing and pattern recognition, Dalian, China, p.190–193, 2011.
14. E.Yu. Shchetinin, Cluster-based energy consumption forecasting in smart grids, Springer Communications in Computer and Information Science (CCIS), Springer,Berlin, 919, 46-656, 2018.
15. Mobasher, B., Burke, R., Bhaumik, F., Williams, C., Towards trustworthy recommender systems: An analysis of attack models and algorithm robustness, ACM Transactions on Internet Technology , 7,4, 2007.
16. Z. A. Wu, Y. Q. Wang, J. Cao, A survey on shilling attack models and detection techniques for recommender systems, Science China, 59(7), p.551-560, 2014.
17. I. Gunes, C. Kaleli, A. Bilge, H. Polat. Shilling attacks against recommender systems: A comprehensive survey. Arti
cial Intelligence Review, p. 1-33, 2012.
18. M. M. Azadjalal, P. Moradi, A. Abdollahpouri, M. Jalili, A trust-aware recommendation method based on Pareto dominance and confidence concepts, Knowledge-Based Systems, 116, p. 130–143, 2017.
19. L. Xu, S. B. Lin, Y. Wang, Z. B. Xu. Shrinkage degree in l2-rescale boosting for regression, IEEE Trans. Neural Netw. Learn. Syst., 2015.
20. J. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat., 29(5), p.1189-1232, 2001.