

Detection of Fraudulent Behaviours using Graph Cluster Analysis on Unsupervised Datasets

Rekha G¹, Juno Bella Gracia S V² and Suresh Kumar M³

¹⁻³Sri Sairam Engineering College/Department of Information Technology, Chennai, India
Email: rekha.guna1997@gmail.com, {bellagodwin.it, sureshkumar.it}@sairam.edu.in

Abstract—The widespread use of mobile Application rulers has resulted in fraudulent or misleading practises aimed at boosting the visibility of applications in the top suggestion list. Indeed, fake application developers are constantly using fraudulent ratings and fake feedback to raise their app sales and commit fraud. Although the value of preventing these forged activities is widely acknowledged, there is a lack of knowledge and research in this field. In this paper, we suggest using the Aggregation Algorithm to discover rating fraud for mobile apps, a better way to test and analyse the playstore datasets. In particular, we propose that through analysing datasets using an unsupervised learning method and clustering, we can accurately locate rating fraud. Furthermore, it uses a supervised learning strategy. By using statistical hypotheses experiments to model the agglomerative behaviours of apps, We look at five different kinds of evidence: ranking-based proof, number of installations evidences, rating-based evidences, review-based evidences, and App description mining evidences. Finally, after combining all of the evidences, a graphical pattern is created on the imported data, which is used for fraud detection discovery.

Index Terms— Review of applications, Rating, K-means algorithm, Fraud application.

I. INTRODUCTION

Because of the advent of mobile computing, the use of smart mobile devices is on the rise in today's world. Because of the popularity of mobile apps for smart devices, fraudsters target app marketers who purposefully raise their apps to the top of the suggested apps list. The number of mobile apps has risen in recent years. The Scope of the approach relays on popular apps which gets targeted easily for malicious practices. Whenever an application is launched , it appears in the leaderboard based on popularity and ranking which drives a competition between the applications. The malicious developers target the high ranked apps to developed a cloned version of the similar apps to increase their revenue by giving fake reviews and fake installation counts. App markets are commonly used as launchpad for malware. The leaderboard displays the app's chart rankings. A higher ranking in the app suggestion list typically translates to thousands of downloads and millions of dollars in sales. The first and the most important one is to detect the duplicate app in the playstore. To this end, in this paper, we used Evidence Aggregation Algorithm for better way to evaluate the Ranking system. In traditional marketing solutions, there seems to be limited understanding and research in this area. Specifically, we first propose to accurately locate the ranking fraud by analyzing the datasets using unsupervised learning approach by means of clustering. We evaluate our approach from Google playstore datasets. The approach would involve five types of evidences and they are Ranking based datasets, review based datasets, number of installation evidences,

rating based evidence and app description mining evidence through statistical hypotheses tests. Overall our major contribution involves the following points:

1. Monitoring the complaint which is given by the application owner.
2. Admin collects the first filtration of evidences i.e. ranking, category, rating, app description and reviews.
3. The second level of filtration can also be considered i.e. Date and version, Installation counts of the app.
4. Accumulating those evidences and finding the efficiency for the particular app.
5. Finally, based on the evidences the Fraud application can be detected.

II. PROPOSED SYSTEM

In Proposed System, We propose to develop a discovery of fraud detection system for mobile App. Clustering is one of the most common exploratory data analysis technique used to know about the structure of the data. It is the process of identifying the group of objects which are similar in functionalities within the groups and not between the groups. Several subgroups are identified in the given data. Datapoints between the clusters always varies. We use R studio to analyse datasets through unsupervised approach. Centralized installation and configuration of R packages provide ease sharing of code and data. Data are grouped in such a way that one data belong to one cluster only. It splits the data based on their similarity to perform an agglomerative approach. We use statistical hypotheses test to quantify the evidences based on sample data to find the unknown patterns.

A. Advantages

- Detect fraud using evidence aggregation in daily App leader boards.
- One user can rate the application only once.
- Application data owner can raise a complaint against cloned apps to initiate for competitor check.

III. SYSTEM IMPLEMENTATION

A. User Interface and User Authentication

In this module, We Design new Client side web page to actively interact with server and with the application developer. we generate unique Authentication scheme for each and every user[Application Data Owner]for the check of Competitor App Data Ranking History's And also we design another Client side pages for user search, reviews, rating, the App. In this module the user authentication is required to rate the application. Once the user purchased the application he will be provided with a link for rating. This is the one time rating link the link will be sent to the mail id of user who purchased the app. Through the link he will rate the app which is linked with his mobile. This prevent the same user to rate the product again. user can rate the app only once.

B. Dataset Analysis

In this approach, dataset analysis is done based on the type of data. We use Google playstore data which comes under unstructured data. Unsupervised learning is used to find unknown patterns in the imported datasets. Clustering is the foremost concept used in unsupervised learning. There are many types of clustering as part of machine learning. In this approach we use K-means for exclusive partitioning. The dataset involves listed parameters- App, Category, Reviews, Installs, Ranking, Genres, Size, Rating, Type, Price, Content rating, Android version. The data is composed of several leading events. Initially, basic characteristics of leading events need to be analysed for extracting theft evidences. In Review behaviours, translated review, sentiment polarity and sentiment subjectivity is calculated. Through API's historical ranking records, We observe that API's ranking behaviors in a leading event always satisfy a specific ranking pattern. It consists of three phases they are rising(growing) phase, maintaining(stable) phase and recession(dropping) phase. The graphical pattern is raising to its peak position then it is considered as rising phase. When the session remains in the same stage for a period of time then it is referred as maintaining phase. Recession phase is the phase where it reaches to the end of the event.

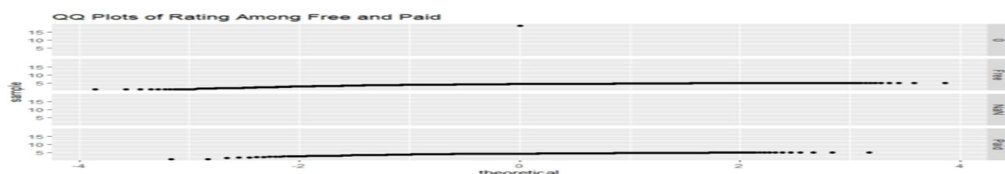


Figure 1. Plot Between Free and Paid Application

Other than ratings, most of the Applaunch market also allow users to write some textual comments as Application reviews, which reflects their thoughts on the application usage. In this approach we apply sentimental analysis on the textual comments to detect the review behaviours and to categorize the phrases based on keywords.

Figure 2. Review, Ranking and Rating value

C. Clustering By K-means Algorithm

A playstore dataset consists of n data points, each of which is in R^d . The problem statement is to extract the minimum variance by clustering the playstore data into K clusters in order to find K points (m_1) ($j=1,2,3,...,k$). The default distance between the K nearest neighbours is termed as Euclidian distance. M_1 points ($j=1,2,...,k$) in R^d such that $n \sum_{i=1}^n [\min d_2(x_i, m_j)]$ is minimized, where $d(x_i, m_j)$ denotes the Euclidean distance between x_i and m_j . The points $\{m_j\}$ ($j=1,2,...,k$) are known as cluster centroids. k cluster centroids are calculated using above expression. so that the average squared error which uses loss function for least squared regression. Due to this, mean squared error among the imported datasets and nearest cluster centroid is minimized. The k-means algorithm is termed as method of vector quantization where the datasets does not have a fixed target variable to predict. We only have the independent variables to predict. The main cause of the popularity of k-means is ease and scalability, simplicity of execution, timely speed of convergence and adaptability to sparse data. It is a gradient descent procedure, which initiate at K cluster centroids and iteratively modify these centroids to decrease the objective function in the equation list above.

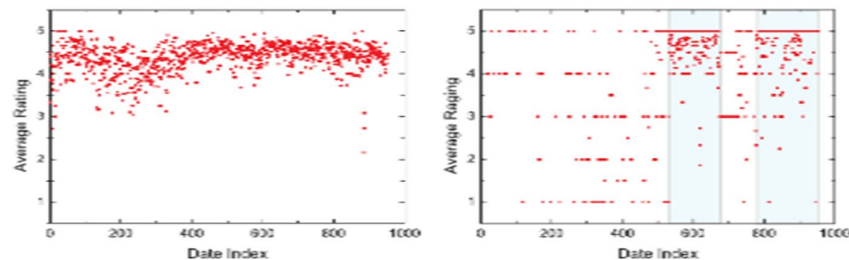


Figure 3. Blot graph data index

Algorithm:

1. Specify the count of clusters K .
2. The centroids are c_1 (rating of the application), c_2 (installation count), c_3 (reviews of the application), c_4 (ranking) and c_5 (description). They are first shuffled and randomly selected without the replacement.
3. Add the squared distances between all data points and centroids to get the total distance.
4. Assign each data point to the cluster that is nearest to it (centroid).
5. Calculate the cluster centroids through mean of all the data points that belongs to each cluster in the datasets.

D. Evidence Aggregation

Evidence aggregation is the module in which all of the evidences obtained from unknown labelled patterns in imported datasets are combined. We present a method for obtaining relationships between permissions and functionalities from data resources in App stores using sensitive APIs as a bridge, and we use the results to assist developers. It searches for common association rules, itemsets, and subsequences that render API calls. We aggregate all of the data and use an agglomerative approach to detect fraud in this approach. After extracting five

different types of theft evidence, the next step is to combine them in order to rank fraud detection. Based on sentimental analysis, positive, negative, and neutral parameters are set.

App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
10 Best Foods	I like eat delicious food. That's I'm co	Positive	1	0.533333333
10 Best Foods	This help eating healthy exercise regul	Positive	0.25	0.288461538
10 Best Foods	nan	nan	nan	nan
10 Best Foods	Works great especially going grocery s	Positive	0.4	0.875
10 Best Foods	Best idea us	Positive	1	0.3
10 Best Foods	Best way	Positive	1	0.3
10 Best Foods	Amazing	Positive	0.6	0.9
10 Best Foods	nan	nan	nan	nan
10 Best Foods	Looking forward app,	Neutral	0	0
10 Best Foods	It helpful site I It help foods get I	Neutral	0	0
10 Best Foods	good you.	Positive	0.7	0.6
10 Best Foods	Useful information The amount spelli	Positive	0.2	0.1
10 Best Foods	Thank you! Great app!! Add arthritis, c	Positive	0.75	0.875
10 Best Foods	Greatest ever Completely awesome n	Positive	0.9921875	0.866666667
10 Best Foods	Good health..... Good health first pri	Positive	0.55	0.511111111
10 Best Foods	nan	nan	nan	nan
10 Best Foods	Health It's important world either life	Positive	0.45	1
10 Best Foods	Mrs sunita bhati I thankful developers	Positive	0.6	0.666666667

Figure 4. Evidence aggregated datasets

There are a number of rating and proof aggregation approaches in the literature, including permutation-based models, score-based models, and so on. Some of these approaches, on the other hand, concentrate on studying a global ranking for all candidates. Other approaches rely on supervised learning strategies, which are difficult to exploit because they rely on labelled training data. Instead, we suggest combining these evidences using an unsupervised approach focused on fraud similarities.



Figure 5. Graph-cluster Analysis

The concluded evidence score $\Psi(s)$ is described as a linear combination of all the existing evidences. Note that, here we are using the linear combination, $\Psi(s) = \sum_{i=1}^N w_i \times \Psi_i(s)$, s.t. $\sum_{i=1}^N w_i = 1$, Where $N = 5$ is the number of evidences, and weight $w_i \in [0, 1]$ is the proof aggregation evidence parameter $\Psi_i(s)$. Thus, the problem of evidence Aggregation becomes how to learn the exact parameters $\{w_i\}$ from the extended training sessions.

E. Recommending security and app requirements

In the literature there are many models for detecting fraud. But here we propose an unsupervised approach in Google playstore datasets. Futher the similar datasets are trained and anlaysed through supervised learning. When a user feels worst about the Apps which they currently using in our Application also they can able to raise the complaint. Once complaint raised by the user even though product is trustworthy and efficient admin will compare the original with duplicate as per evidences aggregation output and comparison from the version date.

Declarations

1st input: c1's historical ranking records R_a ;

2nd input: c1's previous ranking records

Input 2: The rating threshold is K .

The merging maximum point is the third input.

The collection of cluster-leading sessions as an output. As part of the security perspective, the Sa Complaint module is set up for each application owner.

IV. CONCLUSION AND FUTURE ENHANCEMENT

In order to detect rating fraud in a mobile application, we looked at five separate evidences. Furthermore, a method of aggregation based on optimization is used to incorporate all of the facts in order to assess the stealing of extended sessions from mobile Apps. This method is unique in that all of the evidences can be modelled using

statistical hypothesis tests, making it simple to incorporate other evidences from domain information to detect rating fraud. Finally, we test the proposed method by looking at real-world data from the Google Playstore. In the future, we plan to look at more effective fraud evidences as well as examine the latent relationship by integrating multiple evidences. Furthermore, in order to improve user experience, we would combine our rating theft detection approach with other Application related services, such as recommendation of mobile Apps. We will continue in this research direction in the future.

REFERENCES

- [1] "Enriching iTunes API store categories through topic modelling," S. Vakulenko, O. Müller, and J. v. Brocke, The 35th International Conference on Information Systems, 2014.
- [2] "A study of app store analysis for software engineering," by W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. *IEEE Trans. Softw. Eng.*, vol. 43, no. 9, September 2017, pp. 817–847.
- [3] "PScout: Analyzing the Android permission specification," in *Proc. AcM Conf. Comput. Commun. Secur.*, 2012, pp. 217–228. K.W.Y. Au, Y.F. Zhou, Z. Huang, and D. Lie,
- [4] "Automated static code analysis for classifying Android Applications using machine learning," by A. Shabtai, Y. Fledel, and Y. Elovici, in *Computational Intelligence Society (CIS)*. 329–333, IEEE Computer Society, 2010.
- [5] "In Conference on Information and Knowledge Management (CIKM), 2013, pp. 619–628. "Ranking fraud detection for mobile APIs: A holistic view," in *Conference on Information and Knowledge Management (CIKM)*, 2013, pp. 619–628. H. Zhu, H. Xiong, Y. Ge, and E. Chen.
- [6] "Identifying spam in the iOSAPI store," in *Joint WICOW or AIRWeb Workshop on Web Quality*, 2012, pp. 56–59. R. Chandy and H. Gu, "Identifying spam in the iOSAPI store," in *Joint WICOW or AIRWeb Workshop on Web Quality*, 2012, pp. 56–59.
- [7] "Why people hate your API: making sense of user feedback in a mobile API store," in *International Conference on Information Discovery and Data Mining (SIGKDD)*, 2013, pp. 1276–1284. B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, *Data Mining (SIGKDD)*, 2013, pp.
- [8] "API relationship calculation: An iterative process," *IEEE Transactions on Information and Data Engineering (IEEE TKDE)*, vol. 27, no. 8, pp. 2049–2063, 2015. M. Liu, C. Wu, X.-N. Zhao, C.-Y. Lin, and X.-L. Wang, "API relationship calculation: An iterative process,"