Journal of Network and Computer Applications 112 (2018) 126–141

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

An ensemble learning based approach for impression fraud detection in mobile advertising

Ch. Md. Rakin Haider, Anindya Iqbal, Atif Hasan Rahman, M. Sohel Rahman *

Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, ECE Building, West Palasi, Dhaka, 1205, Bangladesh

ARTICLE INFO

Keywords:
Mobile advertising
Ad fraud
Machine learning
Security

ABSTRACT

Mobile advertising enjoys 51% share of the whole digital market nowadays. The advertising ecosystem faces a major threat from ad frauds caused by false display requests or clicks, generated by malicious codes, bot-nets, click-firms etc. Around 30% revenue is being wasted due to frauds. Ad frauds in web advertising have been studied extensively, however frauds in mobile advertising have received little attention. Studies have been conducted to detect fraudulent *clicks* in web and mobile advertisement. However, detection of individual fraudulent display in mobile advertising is yet to be explored to the best of our knowledge. We have proposed an ensemble based method to classify each individual ad display, also called an *impression*, as fraudulent or non-fraudulent. Our solution achieves as high as 99.32% accuracy, 96.29% precision and 84.75% recall using real datasets from an European commercial ad server. We have proposed some new features and analyzed their contribution using standard techniques. We have also designed a new mechanism to offer flexibility of tolerance to different ad servers in deciding whether an ad display is fraudulent or not.

1. Introduction

Digital advertising is one of the fastest growing industries all over the world. According to McKinsey Global report (Sonja Murdoch Moinak Bagchi, 2016) of 2015, total global spending in digital advertising, is predicted to be 231 billion by 2019, almost equal to that of TV advertising. Among the whole spread of digital advertising, the growth of mobile advertising in particular is even higher. Currently, mobile advertising enjoys 51% share of the whole digital market which is expected to be 70% by 2019 (Mobile, 2016). A major concern for this otherwise enviable industry is that almost 30% of the revenue is consumed by different types advertisement frauds (Rajab et al., 2006). Naturally, detecting ad fraud has become a significant research problem.

To display an advetisement, the advertiser purchases a position/placement from a publisher for a particular period or, more popularly, for a fixed number of displays. Usually the advertiser places the display material in an ad server. An impression is generated when the advertisement material is loaded in user's device. If an user clicks on an advertisement, it is recorded as a *click*. The details of the digital advertising ecosystem will be presented shortly in a later section (Section 3).

Ad servers are responsible to maintain logs of all the actions generated from users' interactions such as clicks, impressions etc. with an advertisement. Publishers are paid according to the impression or click count or both that they drive for an advertisement. Hence, there are incentives for fraudulent publishers to inflate the number of impressions and/or clicks that their site generates. There is yet another angle of fraudulent activities not involving the publishers. Dishonest advertisers tend to simulate requests on the advertisements of their competitor to deplete the competitors' advertising budgets.

The dominant type of advertisement frauds is known as bot-driven fraud that employs bot networks (botnet for short) or paid human beings to generate fake ad impressions or clicks. The attacker typically gains access to several valid cookies and IP addresses using several machines with several accounts on ISPs. Alternatively, it exploits the cookies and IP addresses of legitimate users through spywares and Trojans. A botnet tries to simulate impressions and clicks in such a way that the generated traffic resembles real traffic as much as possible. For this type of fraud detection, the most common way is to analyze server logs or network traffic (Liu et al., 2014). To detect those irregular/fraud requests, ad servers try to identify each user using multiple IDs. Universally unique identifier (UUID), Session ID, Cookie ID,

* Corresponding author.
E-mail addresses: rakinhaider@cse.buet.ac.bd (Ch.Md.R. Haider), anindya@cse.buet.ac.bd (A. Iqbal), atif@cse.buet.ac.bd (A.H. Rahman), msrahman@cse.buet.ac.bd (M.S. Rahman).

https://doi.org/10.1016/j.jnca.2018.02.021
Received 5 November 2017; Received in revised form 21 February 2018; Accepted 27 February 2018
Available online 27 March 2018

browser properties, IP, location etc. are usually used to identify a single user's request.

While existing techniques mostly focus on internet based web advertising area, the faster growing alternative, i.e., the mobile advertising domain has not received enough attention from researchers. Although mobile ad fraud detection is philosophically quite similar to that of web based ad fraud detection, there indeed exist significant new research challenges in practice. In particular, it is essential to consider new features related to mobile devices that have not been hitherto investigated in the literature. Since the performance of fraud detection techniques (especially the machine learning based ones) presumably depend on the selected features, research on mobile fraud detection requires significantly new technical analysis and innovation. To the best of our knowledge, there is only one attempt (Oentaryo et al., 2014) to detect mobile ad frauds using a real-world fraud data from *BuzzCity Ltd.*, a mobile ad server company from Singapore. However, their fraud identification is based on detecting a source of huge number of fraud activities; rather than detecting individual fraud events (impressions) which is more challenging. Both (Stone-Gross et al., 2011) and (Oentaryo et al., 2014) detected fraudulent/suspicious publishers without addressing the problem of impression fraud. It should be mentioned here that while generating features for fraudulent publisher detection, multiple clicks or impressions generated by the same publisher are usually available and this facilitates the generation of aggregate features. On the other hand, each individual impression is independent in nature, which makes generation of derived features difficult. In this paper, we address this more challenging problem of fraud impression detection of mobile advertising for the first time to the best of our knowledge.

Inspired by the working model of (Oentaryo et al., 2014), our research is also based on real life data. Recall that in (Oentaryo et al., 2014) the authors used the real life data produced by Buzzcity Ltd. Here, we have used a dataset from a renowned European mobile ad server company, receiving about 5 Million impressions every day. We were provided 260, 678 data of delivery to numerous sites over a duration of 7 days. A labeled dataset was generated using fraud traffics purchased from two sources and a bot traffic generator (Traffic predator. 2016) and also with the help of an audit company named *Forensiq*, which provides reports on ad fraud. The characteristics of this dataset is analyzed and presented later.

The major contributions of this paper are summarized below:

- We have made an attempt to detect individual fraudulent impressions in mobile advertisement that is to the best of our knowledge has not been addressed hitherto in the literature. It should be mentioned here that relevant works in literature (e.g. fraud publisher or fraudulent impression detection in web advertisement) were significantly different in terms of technical approach from our context.
- We have identified and generated 20 derived features specially tailored to facilitate the work of impression-based fraud detection. We have captured some properties of user interactions (e.g., swipe, on-screen appearance and time, etc.) with mobile devices that led to the derivation of features hitherto unexplored in the literature.
- We have trained impression based fraud detection classifiers using state-of-the-art ensemble learning methods. Performance of these classifiers have been validated using 10-fold cross validation technique to ensure the claimed accuracy thereof. As has been reported in a later section, the decision tree classifier J48 along with boosting ensemble learning technique performed the best in terms of accuracy, precision, and recall (99%, 96%, and 84%, respectively).
- We have applied an oversampling technique, SMOTE (Chawla et al., 2002), to handle the hugely imbalanced nature of the training dataset and have reported the consequent classifier performance.
- We have applied several feature selection techniques such as information-gain based feature selection, wrapper selection technique etc. to find out the features that have higher impact on fraud detection. We have also applied dimension reduction technique such as principal component analysis (PCA) and achieved similar accuracy with fewer dimensions.
- We have proposed a scheme to offer flexibility of tolerance to different ad servers by training a classifier using logistic regression to assign each impression a probability of being non-fraudulent. This will allow different ad servers to set different thresholds while detecting fraudulent impressions. This facility is yet to be offered by any existing ad fraud solutions to the best of our knowledge. From our study and interaction with a renowned ad server we have experienced that ad servers tend to establish different tolerance level towards fraudulent activities. To this end, previous classifiers lack the flexibility to adapt to this particular need of multiple ad servers.

2. Related works

Major ad service providers such as Google, Microsoft, Yahoo have documented about their invalid traffic monitoring activities without disclosing significant technical details to raise trust for advertisers (Inc Google, 1673; Bing, 2016; Yahoo, 2016). However, there is little information available on fraud offered by intermediaries. It has been repeatedly stated that detecting and preventing fraud in advertising networks presents significant challenges (Schwartz, 2016; Alexander Tuzhilin, 2016). Advertisers have found hardly any way to independently estimate or defend against click-spam. Sometimes they rely on a measure provided by third-party analytics companies (e.g. Adometry, Visual IQ, and ClearSaleing). However, validity of these is non-transparent since their methodology and models are not open to public review (Kirk, 2016).

Classical fraud detection techniques evaluate the possibility of a publisher being fraud considering how the ratio of impressions to clicks for advertisements differs from the norms. One of the notable early-stage works were done by (Metwally et al., 2005a) where the authors proposed a simple Bloom filter-based algorithm that detects a naive click fraud attack. They could detect the publisher running a script that continuously loads its page and simulates clicks on the advertisements in the page. In (Metwally et al., 2005b), the same authors proposed another solution for detecting ad fraud attack via cooperation between commissioners and Internet Service Providers (ISPs) through identifying associations in a stream of HTTP requests. The commissioners need to preserve surfers' privacy and consequently they can only perform traffic-mining techniques on aggregate data using temporary surfers' identification, cookie IDs and IP addresses. The same authors continued work on this problem and proposed a generalized coalition attacks detection mechanism by discovering sites that have similar traffic (Metwally et al., 2007). They modeled the problem of detecting fraud coalitions in terms of the set similarity problem. A Similarity-Seeker algorithm is developed that uncovers coalitions of site pairs. They extended the detection algorithm to detect coalitions of any size by finding all maximal cliques in a sites' similarity graph.

Reference (Metwally et al., 2008) models discovering single-publisher attacks as a new problem of finding correlations in multi-dimensional data and devise the SLEUTH algorithms for detecting such attacks. They try to detect fraud based on mining the ad server's traffic logs. Their approach is to draw correlation between the fraudster's site and the machines used in the attack using data mining techniques. For attacks using a small number of IPs, this work provide good prevention without incurring high cost for the ad server. This technique is not applicable when the botnet uses tens of hundreds IP addresses. Moreover, they cannot detect attacks other than those generated by bots or

malwares.

Despite these attempts, advertisement fraud activities were increasing powered by the development of smart bots (Kirk, 2016; Zeroaccess, 2016). Hence, fraud detection and prevention for both click and impression also continued. Haddadi (2010) measured click fraud activity and created blacklists for IP addresses to prevent them. This work is named exploited Bluff Ads with unrelated targeting information (e.g., dog food ads for cat lovers). Clicks on Bluff ads are assumed to be click-spam, which the ad network should discount. This technique only applies to click-spam driven by malware as identified by (Dave et al., 2012). Dave et al. (2012) proposed a methodology for advertisers to independently measure click-spam rates on their ads. Formulating a hypothesis that a bot would be less likely to make any extra effort to reach the target website than a user legitimately interested in the ad, they developed a detection mechanism which mitigated the possibility of false positive using a Bayesian framework. They have measured click attack on mobile ad and identified seven new types of attack signatures. This click-spam detection approach can be independently applied by an advertiser at the granularity of an individual ad. This works for non-bot attacks as well.

Another empirical study by Zhang et al. (2011) also developed measurement methods that purchased traffic aimed at a honeypot website, and reported on a range of characteristics related to click fraud. Online impression fraud has been empirically studied by (Springborn and Barford, 2013). They analyzed purchased traffic for a set of honeypot websites. Data collected from these sites were used to reveal the mechanism of impression fraud. They also provided a case study of scope of Pay per View networks and showed that these networks deliver around 500 millions of fraudulent impressions per day, resulting in hundreds of millions of lost advertising dollars annually.

SbotMiner (Yu et al., 2010) tries to identify bot activity by using KL-divergence to detect change in query distributions and removed false positives due to flash crowds. Approaches like Premium Clicks (Juels et al., 2007), access control gadgets (ACG) (Roesner et al., 2012) and CDN fraud prevention (Majumdar et al., 2007) try to prevent attacks beyond botnets. Premium clicks employs economic disincentives that devalue clicks from non-gold-standard users. ACGs ensure authentic UI interactions by users clicking a link. CDN fraud prevention proposes a heavy-weight challenge-response protocol for publisher-payee CDN models. The latter suffers from the limitation of requiring re-architecting the browser or the ad network infrastructure. Crussell et al. (2014) first focused on the mobile ad fraud and developed MAdFraud with an aim to automatically detect fraud behavior through black-box testing of apps alone, without the need to manually reverse engineer apps. They built request trees to link causally related HTTP requests and used machine learning to discover new ad request pages based on known ones. Finally, based on the properties of ad clicks, they created rules to identify clicks from the HTTP request trees. From analyzing the datasets, they found that about 30% of apps with ads make ad requests while in running in the background, which may be malicious or indicate misconfiguration. MAdFraud exposes potentially fraudulent behaviors in apps but it is up to the ad providers to decide if the behavior is indeed fraudulent. However, the features used by them miss some important ones as identified in our work.

3. Mobile advertisement ecosystem

In this section, we briefly describe the mobile advertisement ecosystem along with a few common techniques that may be employed with malicious intents. Readers familiar with the mobile advertisement scenario may skip to the next section.

3.1. Terminologies

Publishers and **advertisers** may be seen as the two main actors of this ecosystem. Publishers are willing to display advertisements on their websites or applications in exchange of payment and advertisers want to display promotion of their products to the customers. Now, **Advertising network** (also known as ad network or ad server) acts as a mediator between the publishers and advertisers as follows. Ad servers store advertisements provided by the advertisers and deliver them to the publishers upon request. They implement a mechanism to display different advertisements on the publishers' contents every time it is "accessed". Every time an advertisement is displayed or loaded in the advertisement frame, it is called an *impression* and when a user clicks on the advertisement frame, it is counted as a *click* event.

3.2. How ad network works

If an advertiser wishes to promote her product(s), she contacts an ad server authority and supplies the contents aimed for the potential customers. The ad server stores these contents. Most ad networks provide custom APIs to the publishers. When a publisher wishes to generate revenue through her website or application using online advertising, she signs up with an ad network and accesses its APIs to enable mobile advertising. Generally, these APIs contain a UI component that is used to display the advertisement and the publisher places this UI component in her interface.

When a user first opens the website or the application of the publisher, ad API loads the ad UI component with advertiser's contents and this event is logged as an *impression*. Now if the user clicks on the ad component-a *click* event-another request is sent to the ad server to redirect the user to appropriate URLs. Ad servers keep track of each such event. The advertiser has to pay the ad server based on their revenue model and the publisher receives a certain percentage of the payment from the ad server. Among the revenue models, Pay-per-click and Pay-per-impression models are most popular. Fig. 1 illustrates the flow of the entire process.

Any (malicious) activity that hampers the advertiser's plan to draw a desired number of customers from the money he is willing to pay in advertising, can be termed as an ad fraud. Such frauds can be classified into click or impression frauds based on the revenue model used by the ad server. If the ad server follows pay-per-click method then fraudulent entities will try to generate artificial clicks on the ads resulting in click frauds. Similarly, a fraudulent entity would choose to generate artificial impressions when the pay-per-impression revenue model is used and this type of frauds are labeled as impression frauds. Impression frauds can be performed by a publisher who is willing to earn more through his contents, other ad server who are wiling to undermine the reputation of an ad server and even by competitors of the advertiser with a goal to depleting advertiser's resources.

Since advertisement shown on a publisher's contents may not be well-received by a user, a fraudulent publisher always aims to receive

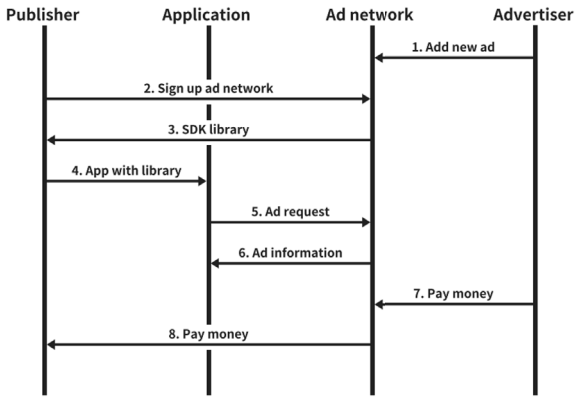


Fig. 1. How ad networks work to manage publishers, applications, and advertisers. Figure borrowed from (Cho et al., 2016).

the maximum possible revenue displaying minimal advertisement. A primitive method is to set the size of the ad frame to zero, so that the advertisement is not visible to the user even if it is being retrieved from the ad server and logged as an impression. In this way the publisher manages to earn revenue from showing advertisement on his contents and still manages to provide better user experience. At the same time, the advertiser is charged for these impression but doesn't gain any publicity in return.

Another approach to increase publishers' revenue is buying invalid impressions from traffic generating companies who employ a large number of people to generate increased traffic to the publisher's contents at a cheaper rate. In this way, the publisher creates impression towards his contents by using users who may have no interest in the advertisements at all thereby making the advertisement useless for the advertiser. Moreover, the publisher receives more impressions than he would have usually received. Thus the advertiser ends up paying more without gaining any kind of publicity for his products.

Although many defense mechanisms have been set up to protect against common fraud techniques, more innovative ones are continuously being introduced by malicious publishers. A more sophisticated and recent way to increase invalid traffic to a publisher's content is to set up a bot network. In this method a network of devices are employed to generate traffic in a controlled manner. Moreover, in case of mobile applications, fraudulent impressions can be generated through emulators and even on real devices while the application is active in the background. Fraudulent ad servers and fraudulent competitors of the advertiser can employ both automated and human driven traffic generation techniques. For the sake of brevity, we refer to (Cho et al., 2016; Liu et al., 2014) for further details on these techniques.

4. Our Approach

In this work, our goal is to detect individual fraudulent impressions in mobile advertising. We start with a brief overview of our entire approach. We have accumulated a dataset containing various attributes corresponding to a list of impressions. Using this dataset we have performed a characteristic analysis of each individual impression. Following this analysis we were able to generate 21 derived attributes with which we have trained our classifiers. We have also applied dimension reduction techniques to support classification with fewer number of features.

4.1. Dataset generation

We have used two tables of data provided by a privately owned mobile advertising company which was ranked as one of the top 5 fastest growing companies in Sweden. The tables are *Delivery* table and *Event* table. The *Delivery* table contains 44 attributes such as *deliveryId, timestamp, clientIP, UID, accountId, siteId, ipCountryCode, cookie, isImpression, impressionTime* and many more. These attributes hold information about a delivery such as when the delivery was made, to whom the delivery was made, on which site the advertisement was shown, the location of the client to whom the delivery was made, etc. The second table contains information related to user interaction, i.e., swipe, scroll, tap, etc. with their mobile during the advertisement life-cycle. These interactions are named as *events* and each entry in the *Event* table corresponds to an event with respect to an impression. The attributes of *Event* table are *deliveriId, eventId, eventTimestamp, eventType, eventValue*, etc.

Since we need labeled data to conduct classifier training, we have used the labeled data provided by a company named *Forensiq* (Winning, 2017), that provides reports on advertisement frauds. While we are unable to present the methodology followed by this company, we have relied on their discretion and judgment especially because of their strong track record[1] of fraud detection. In addition to using purchased labeled traffic from *Forensiq*, we have generated fraud impressions using a widely-used bot named *Traffic Predator* (Traffic predator, 2016). Traffic obtained from multiple sources is expected to cover wider range of fraud patterns/characteristics.

Since the raw data did not reflect the global behavior of the clicks we had to derive some aggregate attributes from the dataset as part of the *Feature Extraction* step. These features were later used to train classifiers and test classifier behavior.

4.2. Dataset Characteristics

Using the labeled data, we have analyzed the behavior of fraudulent impressions in our dataset to find out some existing patterns in them.

4.2.1. Temporal distribution

With a view to finding out the temporal distribution of *OK* and *Fraud* impressions, we have calculated the number of impressions during each hour of the day. As we see from Fig. 2a there are very few valid impressions during the owl hours and quite high number of valid impressions during working hours of the day. However, it can be easily noticed that the fraudulent impressions are distributed uniformly throughout the day. This gives us the idea about the targeted attempt of the fraud traffic generators (i.e., bot-nets and human generated ones) to avoid raising suspicions.

4.2.2. Site-based distribution

The ratio of number of fraud impression to that of total impressions on each site gives us an insight about the behavior of fraud publishers. It can be seen from Fig. 2b that sites with lower number of impressions have higher tendency to generate impressions through traffic generators. However, they choose not to generate excessive amount of traffic to avoid suspicion. From Fig. 2b we can see only one site generates about 10^{10} impressions whereas other fraudulent sites choose to generate impressions less than 10^5 in number in the same time period.

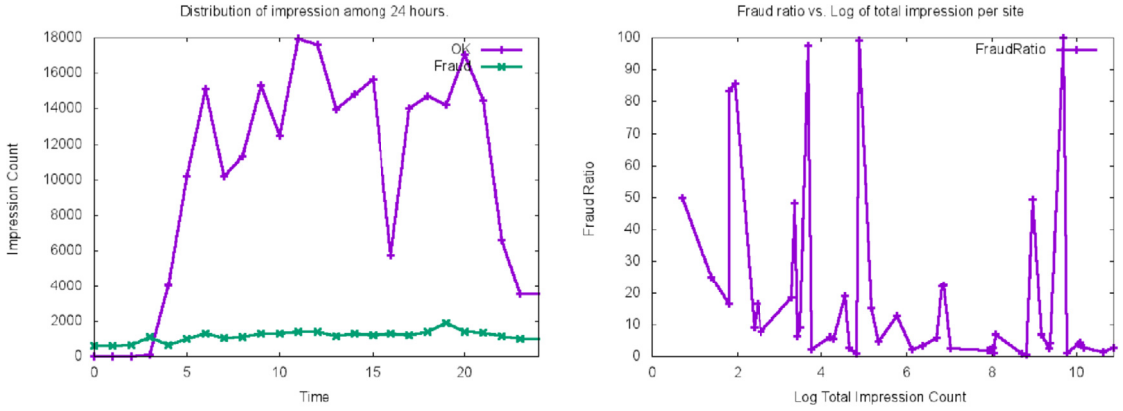
4.2.3. Event distribution

Most modern ad servers track users' interactions on their devices. It is expected that fraudulent impressions will generate less interactions since they don't have any interest in the contents appearing on the screen. Hence, we tried to analyze the distribution of events with respect to each impression. Table 1 shows that non-fraudulent impressions spent more time (about 10s) on screen as compared to that of the fraudulent ones. We can also conclude that valid impressions generate more events than invalid ones. From Fig. 2c we see that benign impression generates higher percentage number of events than the fraudulent impressions.

4.3. Preprocessing

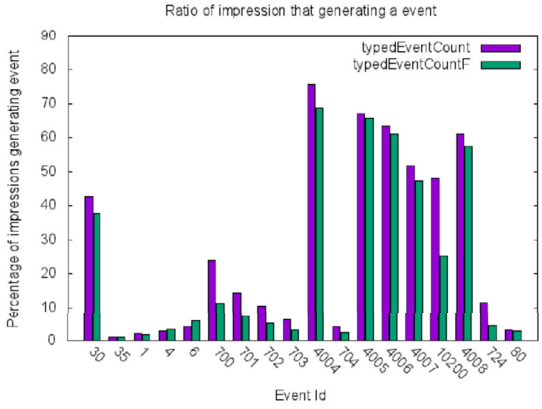
Proper design of features plays an important role in machine learning based approaches. Therefore, we first analyzed the dataset. We investigated the effect of each attribute towards the classification of an impression. For example, we found that for each delivery that was caused by a fraudulent actor, the events generated from the device are somewhat similar whereas it varies to a significant degree when the delivery is non-fraudulent. This gives us an insight about the impact of event-related attributes towards the classification. We could also identify that some attributes didn't have much impact on the fraudulent pattern detection of a delivery. One such example is *adSpaceReqURL*

[1] Notably this company was awarded Leads Council LEADER Award (Leader awards, 2016) for Best-In-Class Fraud Detection in 2014.



(a) Distribution of impressions among 24 hours.

(b) Fraud Ratio Per Log Total Impression



(c) Ratio of impression generating a certain event

Fig. 2. Dataset characteristics.

Table 1
Analysis results of events with respect to impressions.

Impression Status	Maximum # of Event Per Impression	Maximum Onscreen Time (ms)	Maximum # of Distinct Event Per Impression	Average Number of Events Per Impression	Average Onscreen Time (ms)
OK	2337	14,320,876	21	5.1492	35327
Fraud	54	9,645,572	21	4.3343	24832

attribute, which is a hash code of the URL from which the advertisement contents will be received. We have omitted such attributes in subsequent processing.

After selecting the attributes from raw data we performed our feature extraction step. Using these features we conducted feature selection procedure so that the selected features can be used for training classifiers. Since the ratio of fraudulent to non-fraudulent delivery is low in the dataset under consideration, introduction of an unwanted bias in the classifier is apprehended. Hence we applied an oversampling technique namely SMOTE and trained classifiers on oversampled dataset in parallel to the training using original dataset. The impact of oversampling is reported in Section 5.4.2.

4.4. Feature extraction

Accuracy of a classifier largely depends on the quality of feature extraction step. A feature is a numerical index that quantifies a behavior or pattern of an impression. Therefore, properly selected features should be able to reflect the trend or pattern of relation among attributes and impression classification. In order to create new features we have used the concepts described in (Alexander Tuzhilin, 2016; Perera et al., 2013; Oentaryo et al., 2014). According to (Alexander Tuzhilin, 2016), Google defines invalid clicks as those clicks that are generated through prohibited means. As candidates of prohibited means authors of the work in (Alexander Tuzhilin, 2016) refers to automated software, low-cost workers to click on links and some other methods. Additionally, they have mentioned some criteria, i.e., frequency of clicks, delay between clicks, users of the clicks, users' intention behind the click, etc., to consider while trying to detect invalid clicks. Although Google is mostly using anomaly-based and rule-based invalid click detection, they are neither using classifier based techniques extensively, nor they have mentioned about fraudulent impression detection. Oentaryo et al. (2014) and Perera et al. (2013) generated features that were later used to train classifiers to detect fraudulent publishers rather than detecting fraudulent impressions or clicks. Please recall our brief discussion on the work of Oentaryo et al. (2014) in an earlier section. Like in

Table 2
The list of all the features.

MaxSameDelCount	NoOfEvents	Event/DelRatio
EventVariance	OnscreenTime	avgEventTimeDelay
totCount/IP	uniqDel/IP	delAvgDelay/IP
imAvgDelay/IP	uniqDelCount/UID	delPerUID
impressionToDelRatio/UID	viewToDelRatio/UID	delPerSite
uniqDelCount/Site	impressionToDelRatio/Site	viewToDelRatio/Site
averageTimeDifference/Site	delPerDeviceId	status

(Oentaryo et al., 2014), (Perera et al., 2013) authors derived features from attributes available in the Buzzcity dataset, i.e., time of click, agent, ip address, country, campaign etc.

In out work, each attribute was considered separately to find out the scope and suitability of creating new features. We created a number of parameters or statistical measures based on a particular impression to model the relationship of impression pattern to impression classification. Since we applied feature selection on the created features, we tried to create as much features as possible in the feature extraction phase without considering the dimensionality of derived features at least for the time being. In a later section we have described the effect of dimension reduction and report the analysis thereof. The details of each derived feature are described below. A complete list of all the features is given in Table 2.

4.4.1. Event-related attributes

Every delivery of an advertisement corresponds to one or more events. We initially assumed that if deliveries were caused by *bot-nets*, then there would be no event at all. Recall that we use the term event to refer to user interactions like swipe, view etc. It seemed unlikely for a bot to simulate interactions such as swipe, scroll etc. In such a scenario, detection of fraudulent deliveries would have been much easier. However, analyzing deliveries of an advanced traffic generator bot (Traffic predator, 2016) we found that advanced bots are designed to simulate some of there events. Besides, human generated impression-fraud can also be responsible for fraudulent deliveries, thereby causing these events in a difficult to distinguish manner in many cases.

In case of a non-fraudulent delivery a variety of user events and variation in the sequence thereof can be observed. But in case of a fraudulent delivery, i.e., deliveries to a bot or users who have no interest to the contents of the advertisement, a pattern seems apparent in the number and type of events as well as the sequence thereof. In an effort to capture the relation and the pattern we computed *MaxSameDelCount, NoOfEvents, Event/DelRatio, EventVariance, OnscreenTime, avgEventTimeDelay* as briefly defined below. In previous literature these events were mostly ignored in classification.

- **MaxSameDelCount** attribute represents the maximum number of occurrences of an event for a particular delivery Id.
- **NoOfEvents** attribute refers to the count of distinct events related to a delivery.
- **Event/DelRatio** is the average number of events per distinct event for a delivery.
- **EventVariance** is the variance among the counts of a distinct event for a delivery.
- **OnscreenTime** is the amount of time a delivered advertisement was on the screen of a user device. It is computed as the difference between the *timeStamp* of the last event to the *timeStamp* of the initial delivery.
- **avgEventTimeDelay** is the average of the delays between consecutive events.

4.4.2. IP-related attributes

IP address is an important attribute that may characterize the behavior of a delivery. We tried to create features that capture the relationship of a delivery being fraudulent to the IP address that caused the delivery. If an IP address is responsible for generating high number of deliveries through the same site the respective deliveries are likely to be labeled as fraudulent and may be labeled as such. Keeping these in mind we created 4 attributes, namely, *totCount/IP, uniqDel/IP, delAvgDelay/IP, imAvgDelay/IP*as discussed below.

- **totCount/IP** is the number of impressions caused by the IP address of the corresponding impression.
- **uniqDel/IP** is the unique number of delivery entries caused by the address corresponding to the delivery. There is a difference between *uniqDel/IP* and *totCount/Ip*, because there can be more than one entries with the same *deliveryId.* Different entry with the same *deliveryId* corresponds to different phases of delivery life-cycle, i.e., a delivery request is made, the delivery is loaded, the delivery is viewed, etc.
- **delAvgDelay/IP** is the inter-arrival delay of consecutive deliveries in response to corresponding requests from a particular IP address. This gives us an idea of the frequency in which the IP address generates delivery requests. Frequency of impression count in a certain interval from a specific IP address is one of the most basic fraudulent impression detection criteria.
- **imAvgDelay/IP** is the average delay between consecutive impressions caused by an IP address. Note that every request may not yield an impression and hence delAvgDelay/IP and imgAvgDelay/IP vary significantly.

4.4.3. UID-based attributes

In web-base ad-fraud detection, IP addresses can be used as a fixed identifier of a device. However, in case of mobile advertising, IP address of a mobile device frequently changes. Hence, the mobile advertising companies develop their own policies to identify different devices. In our context, an attribute named *UID* is used as a unique identifier for each device. The attribute *UID* considers many information like device model, operating system version, platform of the device, device id, device sdk-version, session id, cookie etc. to generate a unique identifier for each user. Although it can be argued that a fraudulent user can in fact change its *UID* by performing a ”Restore Factory Setting”, we assumed that for a device there is one unique *UID*.

Assuming *UID* as an identifier of devices, we can create attributes that may aid in the detection of fraudulent impressions. We have created the following attributes based on *UID*.

Table 3
This table reports accuracy, precision and recall of decision tree classifiers, trained on the original dataset containing 243,650 OK labeled impressions and 9243 Fraud labeled impressions, where all of the 21 derived features are considered. This result were obtained without attempting to improve their performance with ensemble method.

Classifier	Accuracy	Precision	Recall
J48	98.19	98.10	98.20
REPTree	98.06	97.90	98.10
Random Forrest	98.21	99.30	98.90

Table 4
This table reports accuracy, precision and recall of classifiers, trained (with random seed 1) on the original dataset containing 243,650 OK labeled impressions and 9243 Fraud labeled impressions, where all of the 21 derived features are considered. The standard deviation was computed by training the corresponding classifier with random seeds from 1 to 10.

Classifier	Accuracy	Precision	Recall	Specificity	95% confidence interval	Std. Deviation
Bag with J48	98.26	95.86	54.67	99.91	[1.69, 1.79]	0.003
Bag with REPTree	98.10	94.78	50.69	99.89	[1.85, 1.96]	0.010
Bag with RF	98.86	96.75	71.25	99.91	[1.10, 1.18]	0.005
Boost with J48	99.32	96.29	84.75	99.88	[0.65, 0.71]	0.000
Boost with REPTree	99.14	96.50	79.38	99.89	[0.82, 0.89]	0.009
Boost with RF	99.31	96.61	83.98	99.89	[0.66, 0.73]	0.003

Table 5
Confusion matrices of best performing classifiers.

(a) Confusion matrix resulting from the classifier "Boosting with J48"

Original	Predicted	
	Fraud	OK
Fraud	7833	1410
OK	302	243348

(b) Confusion matrix resulting from the classifier "Boosting with RF"

Original	Predicted	
	Fraud	OK
Fraud	7762	1481
OK	272	243378

- **uniqDelCount/UID** represents the amount of delivery to that *UID*. A very high number of delivery from a single *UID* can be an indication of the device being responsible for fraudulent impressions.
- **delPerUID** is the amount of delivery entry from a given *UID*. *uniqDelCount/UID* differs from *delPerUID* due to having multiple entries with same *deliveryId* each of which represents a different phase of advertising life-cycle.
- **impressionToDelRatio/UID** is the ratio of the impression count to that of delivery count, given a *UID*.
- **viewToDelRatio/UID** is the ratio of viewed advertisement count to that of delivery count, given a *UID*. A device with no interest to the contents of the advertisements is likely to have higher number of impressions than the number of views of the advertisement.

4.4.4. Site-based attributes

A specific site may be more prone to fraudulent activities than others. To model the impression pattern for a particular site, we created attributes named *delPerSite, averageTimeDifference/Site, uniqDelCount/Site, impressionToDelRatio/Site, viewToDelRatio/Site* that are discussed below.

- **delPerSite** refers to the amount of delivery entry for a given site.
- **uniqDelCount/Site** is the amount of unique delivery entry corresponding to a site. *delPerSite* differs with *uniqDelCount/Site* in the same way *totCount/IP* as differs with *uniqDel/IP*.
- **averageTimeDifference/Site** is the average of the delays between consecutive deliveries to a given site. This gives us an insight to the frequency of delivery to the corresponding site.
- **impressionToDelRatio/Site** is the ratio of impression count and delivery count, corresponding to a given site.
- **viewToDelRatio/Site** refers to the ratio between viewed delivery count and delivery count, for a given site.

After computing all the attributes they are merged together. It is done using a column wise join operation. At the end of the feature extraction step, we have 21 features that are candidate of the selection process as described in the following subsection.

4.5. Feature selection

Feature selection step helps us focus on the most relevant attributes rather than focusing everywhere. It gives us an idea on which features are more relevant to our classification than the others. The selection process also helps reducing over-fitting of and memory requirement for building classifier. It also gives us an idea about correlation of different features with the classification. To perform feature selection, we have used Principle Component Analysis (PCA) (Lovric, 2011), Correlation Feature Selection (CFS) (Hall, 1998) subset evaluation, Wrapper subset evaluation (Kohavi and John, 1997) and Information Gain attribute evaluation techniques. Using each of these independently, we generated a ranking of the features. Subsequently we performed training of the classifier with the features having higher ranks.

Table 6
This table reports accuracy, precision and recall of classifier, trained (with random seed 1) on 12 principal components of the original dataset that contains 243,650 OK labeled impressions and 9243 Fraud labeled impressions. The standard deviation was computed by training the corresponding classifier with random seeds from 1 to 10.

Classifier	Accuracy	Precision	Recall	Specificity	95% confidence interval	Std. Deviation
Bag with J48	98.10	93.37	51.78	99.86	[1.84, 1.95]	0.004
Bag with REPTree	97.99	92.29	49.09	99.84	[1.96, 2.07]	0.007
Bag with RF	98.63	91.48	69.04	99.76	[1.32, 1.41]	0.010
Boost with J48	99.07	91.80	81.86	99.72	[0.89, 0.97]	0.000
Boost with REPTree	98.90	92.08	76.35	99.75	[1.06, 1.15]	0.008
Boost with RF	99.15	92.43	83.59	99.74	[0.81, 0.89]	0.003

Table 7
Confusion matrices of best performing classifiers.

(a) Confusion matrix resulting from the classifier "Boosting with RF"

Original	Predicted	
	Fraud	OK
Fraud	7726	1517
OK	633	243017

(b) Confusion matrix resulting from the classifier "Boosting with J48"

Original	Predicted	
	Fraud	OK
Fraud	7566	1677
OK	676	242974

Table 8
This table reports the number of instances of each class in the dataset after oversampling, accuracy, precision and recall of classifiers trained (with random seed 1) on oversampled dataset where the number of Fraud instance are 2,4,8,16 times than that of the original one. To facilitate comparison, we have included the results of the original dataset as the first row. The standard deviation was computed by training the corresponding classifier with random seeds from 1 to 10.

C_i	OK count	Fraud Count	Accuracy	Precision	Recall	Specificity	95% confidence interval	Std. Deviation
C_1	243650	9243	99.32	96.29	84.75	99.88	[0.65, 0.71]	0.000
C_2	243650	18486	98.87	97.54	86.10	99.84	[1.09, 1.17]	0.000
C_4	243650	36972	98.64	98.35	91.17	99.77	[1.32, 1.41]	0.000
C_8	243650	73944	98.32	98.53	94.20	99.57	[1.63, 1.72]	0.000
C_{16}	243650	150058	97.92	99.25	95.26	99.55	[2.04, 2.13]	0.000

Informatively, among the features *delPerSite, delAvgDelay/IP and imAvgDelay/IP* are attributes that received higher ranking irrespective of search and evaluation techniques.

4.6. Classifiers

We have trained several classifiers and evaluated there accuracy, true positive rate and true negative rate in the training set, the validation set and the test set. Our Approach is based on Ensemble Learning. We have used Bagging and Boosting technique on the decision tree classifier and support vector machine. Bagging is an ensemble learning method that learns classifier on various different distributions of the training set and uses all the classifiers for classification. On the other hand, in each iteration of boosting technique, it tries to learn classifier on the samples that was wrongly classified using previous classifiers and assign a corresponding weight to each classifier. Both bagging and boosting help to reduce error rate of an individual classifier. In our classifier, we have used J48, REPTree and Random Forrest as the decision tree algorithm.

5. Experimental evaluation

5.1. Data

In our experiment, the dataset contains two tables as described earlier in Section 4. There are 260,678 delivery entries in the *delivery* table and 918,222 entries in the *event* table.[2] We used these tables to derive the attributes described in Section 4.4. After the *feature extraction* step, we had 252,893 deliveries with all the derived attributes among which 243,650 deliveries were labeled as *OK* and 9243 deliveries as *Fraud*. Since the ratio of *Fraud* to *OK* is very low, it may give rise to an unwanted bias in the classifier. To overcome this issue, we have applied an oversampling technique, namely, SMOTE to produce a more balanced training set. After oversampling our dataset contained 243650 OK deliveries and 150058 Fraud deliveries. We used both datasets, namely, the original one as well as the oversampled one for training classifiers and used 10-fold cross validation.

[2] The raw data is available in this link. goo.gl/2z79nP.

5.2. Platform and tools used

We conducted our experiments in a computer with 4 GB RAM and Intel Core i7 processor. As there are large number of deliveries and events, computing their aggregate attributes was a major challenge. We used *awk* which is a standard command of most unix-like operating systems. Few of the computations required sorting as a pre-computation step. We used unix command *sort* to perform this step. Among all the steps, sorting consumed the principle share of time. After generating all the attributes we used unix command *join* to merge all the attributes and generated a tab separated file (*.tsv*). We used the popular tool WEKA (University of Waikato) to convert this *.tsv* file into *.arff* file to use them to train classifiers as well as to test their accuracy.

5.3. Overall results

Table 3 reports the results of decision tree classifiers. Accuracy, precision, recall and specificity of our classifiers trained using decision tree algorithm as well as emsemble technique and on training set are reported in Table 4. To validate the fraud detection results, we have also applied 10-fold cross validation technique. Furthermore, we have varied the randomization seed 10 times (from 1 to 10) and computed the standard deviation of the accuracy. Moreover, we have computed 95% confidence interval of error for each of the classifiers and reported them in Table 4. Among the classifiers, Boosting with J48 performs best in terms of accuracy (accuracy 99.32%, precision 96.29%, and recall 84.75% respectively) (For the confusion matrix of the best performing classifier see Table 5).

5.4. Experimental results from feature selection strategies

As has been previously mentioned, to perform the feature selection, we have used Principal Component Analysis (PCA), Correlation-based Feature Subset Selection (CFS), Wrapper Subset Evaluation technique with different search techniques. In the following sections, we present the results of the classifiers following the application of the different feature selection techniques.

5.4.1. Principal component analysis (PCA)

The mobile advertising company from which we have received our dataset reportedly receives 5 Million impressions every day. In such a

Table 9
Confusion matrices of best performing classifiers.

(a) Confusion matrix resulting from the classifier C_1(best with respect to accuracy)

Original	Predicted	
	Fraud	OK
Fraud	7833	1410
OK	302	243348

(b) Confusion matrix resulting from the classifier C_5(best with respect to precision, recall, and specificity)

Original	Predicted	
	Fraud	OK
Fraud	142946	7112
OK	1085	242565

situation large number of features may not be feasible. In recent studies, a number of dimension reduction techniques such as *low variance filter, high correlation filter, principle component analysis, backward feature elimination* etc. have been developed. We have performed principal component analysis on the data to find out 12 principal components and trained classifiers using the principal components as attributes. The accuracies, precisions, recalls and specificities of these classifiers are presented in Table 6. 95% confidence interval of error for each of the classifier reported as well. The results in Table 6 shows that classifiers can achieve similar level of accuracy as shown in Table 4 even after performing PCA and reducing the dimension of the features (For the confusion matrix of the best performing classifier see Table 7).

5.4.2. Results after applying SMOTE

In order to reduce the imbalance in the dataset, we have applied SMOTE (Chawla et al., 2002) technique. Using SMOTE technique we oversampled the minority class, in our case the Fraud class, in 4 incremental steps where in each step we doubled the number of instances of the minority class. This resulted into 4 oversampled datasets where the number of Fraud instances is, respectively, 2, 4, 8, and 16 times the number of the original dataset. Since the best overall performance was achieved when boosting technique was applied together with J48 decision tree classifier (Table 4 in Section 5.3), we have further trained this classifier on each of the oversampled datasets. Table 8 shows the number of instances of both classes along with accuracy, precision, and recall for the classifiers trained on the original dataset as well as for the 4 oversampled datasets. For the sake of ease in presentation in Table 8, we have referred to these classifiers as $C_1, C_2, C_4, C_8, C_{16}$ respectively, where the subscripts actually indicates the degree of oversampling. From the results presented in Table 8, it would be fair to claim that the imbalance of the original dataset did not affect our classifiers much (For the confusion matrix of the best performing classifier see Table 9).

Another interesting discussion is in order here. We found that C_1 has a higher false negative rate (15.25%) than others (13.9%, 8.83%, 5.80% and 4.74%, respectively) but achieves slightly lower false positive rate (0.12%) than others (0.16%, 0.23%, 0.43%, 0.45%, respectively). In such a scenario, the selection of the best classifier among these ($C_1 - C_{16}$) should be subject to the priority of false positive rate to that of false negative rate. On a more practical note, a publisher who is eager to label as much fraudulent impression as possible as Fraud, should prefer the technique that gives the highest precision and the one who prefers not to label any benign impression as Fraud, should select the one with the highest recall.

Finally, we have also performed PCA on the oversampled datasets. Here we report (in Table 10) only the results of classifiers that were trained on the principal components of the oversampled dataset with 16 times fraud instances compared to the original one.

5.4.3. Information gain based attribute selection

In a different experiment, we have derived a ranking for the attributes using information gain of each attribute and used this ranking for feature selection. We have selected n attributes with highest ranking to train classifiers and determined the accuracy of the classifiers. We have varied the value of n to determine whether increasing attributes results in an over-fitting. The values of n that are used in our experiments are 5, 10, 12, 15 and 17.

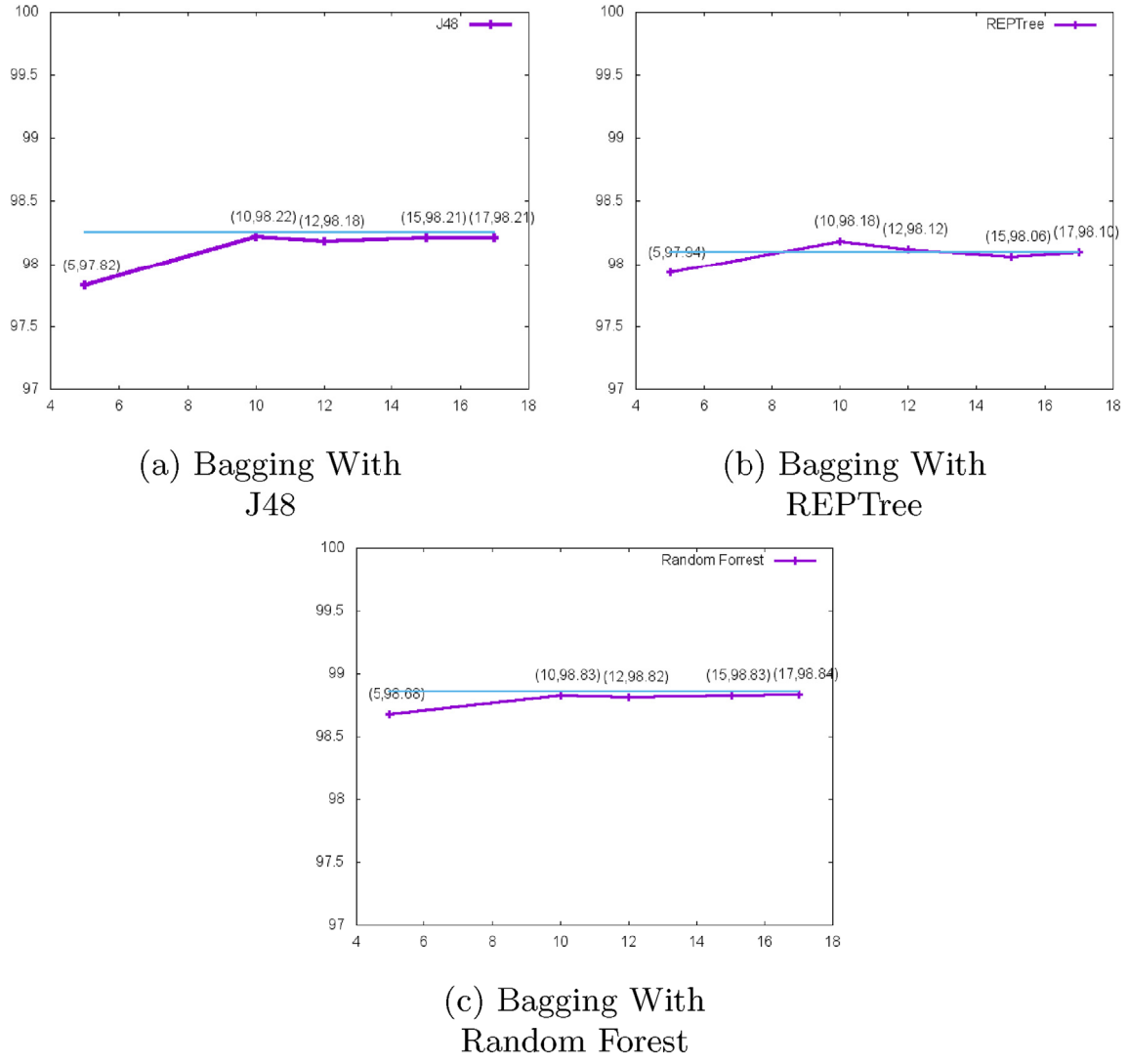
Fig. 3 shows the change of accuracy of the classifiers with the increase in the number of attributes. In each of these figures, the blue line indicates the respective accuracy when all the features are considered and the other line indicates the same with n attributes. We can observe in Fig. 3a and c that the accuracy of the classifier increases with the number of attributes and almost achieves similar accuracy of the classifier with all the attributes. In Fig. 3b only for $n = 10$ the classifier reports higher accuracy than the accuracy achieved using all features. Fig. 4 shows the amount of time required to train each classifier with n features where the values of n are 5, 10, 12, 15, 17. We can conclude that accuracy of the classifiers increases as more and more features are used.

5.4.4. Aggregate scoring of the attributes

We generated a scoring of the attributes with three attribute selection techniques, namely, correlation based attribute selection technique, information gain based attribute selection technique, and wrapper selection technique. These scores indicate how much impact an attribute has towards building a classifier. Each of these scores are normalized by dividing each score with the highest score provided to any attribute for that particular feature selection technique, and then averaged for each attribute to generate an aggregate scoring of the

Table 10
This table reports accuracy, precision and recall of classifier, trained (with random seed 1) on 12 principal components of the oversampled dataset that contains 243,650 OK labeled impressions and 150,058 Fraud labeled impressions. The standard deviation was computed by training the corresponding classifier with random seeds from 1 to 10.

Classifier	Accu-racy	Preci-sion	Recall	Speci-ficity	95% confidence interval	Std. Devia-tion
Bag with J48	96.04	97.64	91.82	98.64	[3.90, 4.03]	0.015
Bag with REPTree	95.63	97.28	91.07	98.43	[4.31, 4.44]	0.009
Bag with RF	96.58	98.29	92.64	99.01	[3.36, 3.47]	0.006
Boost with J48	96.44	98.11	92.44	98.90	[3.50, 3.62]	0.000
Boost with REPTree	96.62	98.27	92.77	98.99	[3.32, 3.44]	0.009
Boost with RF	96.89	98.74	93.03	99.27	[3.05, 3.16]	0.009



(a) Bagging With J48

(b) Bagging With REPTree

(c) Bagging With Random Forest

Fig. 3. Accuracy of different classifiers with different number of attributes.

attributes. The top 10 attributes based on their scores are listed in Table 11. The Pearson Correlation Coefficients for each of the attributes are also reported in Table 11.

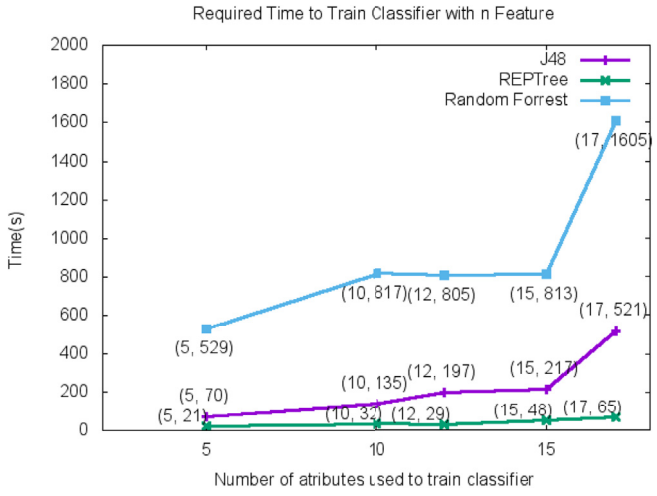


Fig. 4. Time required to train classifier with *n* features.

In prior works of ad fraud detection, several characteristics such as the frequency of impression to a specific site, the time difference between each impression, and number of clicks from the same IP or device were considered to classify an impression. Table 11 reflects that the scores are in agreement with our previous knowledge as *delPerSite, averageTimeDelay/Site, delAvgDelay/IP, totCount/IP* etc. are assigned higher scores. The high score of the attribute *viewToDelRatio/Site, impressionToDelRatio/Site* and *impressionToDelRatio/UID* represent the fact that in case of fraudulent impressions the user have no interest in the content of the advertisement and are not viewed or

Table 11
Top 10 attributes according average score.

Rank	Attribute	Score (0–1)	PCC
1	delAvgDelay/IP	0.957	−0.31
2	imAvgDelay/IP	0.950	-.01
3	delPerSite	0.859	0.22
4	uniquqDelCount/Site	0.850	0.19
5	averageTimeDifference/Site	0.8448	-.09
6	impressionToDelRatio/Site	0.835	0.28
7	delPerDeviceID	0.836	−0.06
8	viewToDelRatio/Site	0.817	0.25
9	NoOfEvents	0.804	0.16
10	impressionToDelRatio/UID	0.795	0.07

Table 12
Top 10 attributes according average score.

Attribute	U-value	P-value
MaxSameDelCount	213250.0	5.98e-101
NoOfEvents	240000.0	7.57e-148
Event/DelRatio	161750.0	3.70e-19
EventVariance	175000.0	8.92e-30
OnscreenTime	230000.0	4.64e-121
avgEventTimeDelay	230000.0	4.64e-121
totCount/IP	244875.0	7.53e-159
uniqDel/IP	229875.0	1.35e-128
delAvgDelay/IP	209750.0	6.11e-80
imAvgDelay/IP	238750.0	1.28e-141
delPerDeviceID	243500.0	4.97e-156
delPerSite	250000.0	1.99e-171
averageTimeDifference/Site	250000.0	1.99e-171
uniqDelCount/Site	250000.0	1.99–171
impressionToDelRation/Site	125750.0	0.43
viewToDelRatio/Site	125000.0	0.50
delCOunt/UID	223250.0	1.02e-117
impressionToDelRation/UID	163250.0	7.02e-21
viewToDelRatio/Uid	140625.0	1.67e-04

even properly displayed after being delivered and thus have a higher probability of being fraud. Also the score of *NoOfEvents* gives us the idea of the role of events in fraudulent impression classification. With respect to the above discussion, we conclude that the results of our experiments on the impact of each attribute towards training a classifier are aligned with our prior knowledge of ad fraud and have provided some more knowledge with respect to specific domain of mobile advertising.

The Pearson Correlation Coefficients were computed after mapping the categorical class values (OK and Fraud) to numeric values (1 and −1). In spite of having high impact on classification, most of these attributes show lower values of correlation coefficient. The reason behind such a result may be due to the non-linear relationship of the attributes with the class values. Since from the values of correlation coefficient we are unable to draw any conclusion about the effect of the features in classification we have also performed hypothetical testing. For each attribute we have conducted Mann–Whitney U test to establish the fact that the samples of the two classes are significantly different from each other. The null hypothesis (H_0) is that the distributions of both populations are equal. Whereas the alternative hypothesis (H_1) is that the distributions are different. For each attribute we have randomly selected 500 observations from each sample and performed the Mann-Whitney test. Table 12 shows the U-value and P-value for each attribute. We reject the null hypothesis where $P < 0.05$. The P-values of Table 12 shows that we can reject the *null* hypothesis for most of the features and thus conclude that there is significant difference between the distributions of almost all of the features.

For better visualization of the values of these attributes, we have generated histograms with ratio of OK impressions and Fraud impressions to total impressions, for each attribute. The range of values of the attribute are divided into 25 equally spaced buckets. These histograms can be found in Appendix A.

5.5. Classification with different tolerance level

As has been discussed in Section 1, due to cost and other relevant issues, tolerance level for fraudulent impressions may vary from publisher to publisher. Specifically, smaller publisher may opt for a higher tolerance level in their classifier. In an effort to propose a framework for such situations, we tried to score each delivery on a scale from 0 to 1. The score denoted the probability of the delivery being non-fraudulent. We can then set a threshold of probability and separate the fraudulent deliveries from the non-fraudulent ones based on the predefined threshold. To fulfill our goal, we used logistic regression as our classifier. As we know, in the learning phase logistic regression fits the data into a logistic function which is later used to estimate probabilities of a categorical class to have one of two or more values. In our main problem we have a binary class variable, i.e., the class attribute can have one of two values, Fraud and OK. In binary logistic regression, usually the threshold of probability is assumed to be 0.5. Here we have used the estimated probabilities with different threshold values of probability and performed the classification. The number and percentage of deliveries that are classified as non-fraudulent among 393708 deliveries, using logistic regression model with different threshold or tolerance level is listed in Table 13. The appropriate threshold level for a company should be selected in such a way so that it neither looses large number of impression nor it allows many fraudulent impressions. From Table 13, we note that the appropriate threshold for the above mentioned company should be 0.7 because with this threshold the number of allowed impressions is closest to the actual amount of valid impressions.

Table 13
Classification with different level of tolerance or threshold.

Threshold	Classified As OK	In Percent (%)
0.5	393490	99.94%
0.55	341117	86.64%
0.6	308188	78.28%
0.65	273771	69.54%
0.7	233819	59.39%
0.75	168022	42.68%
0.8	138599	35.20%
0.85	70866	18.00%
0.9	20066	5.10%

6. Conclusion

In this paper, we have studied a new type of ad-fraud detection based on real life data which aims to classify each individual impression as fraudulent or non-fraudulent. To fulfill this purpose, we have identified and generated 21 features and proposed ensemble based classifier based on these features. To perform classification of each individual, we have trained some classifiers on 21 generated features and applied 10-fold cross validation. In our experiment, we have achieved as much as 99.3% accuracy on the training set. To properly handle the issue of imbalanced dataset, we have also applied SMOTE and trained the classifier on over-sampled dataset which resulted in 97.95% classifier accuracy. To support classification with reduced dimension of features, we have also applied PCA. Finally, in order to satisfy the need of different mobile advertising companies, we have proposed a framework that offers flexibility of tolerance in fraudulent impression detection using logistic regression. We believe this research work will help the relevant industry to successfully detect impression based ad-fraud and trigger more research work in the related area.

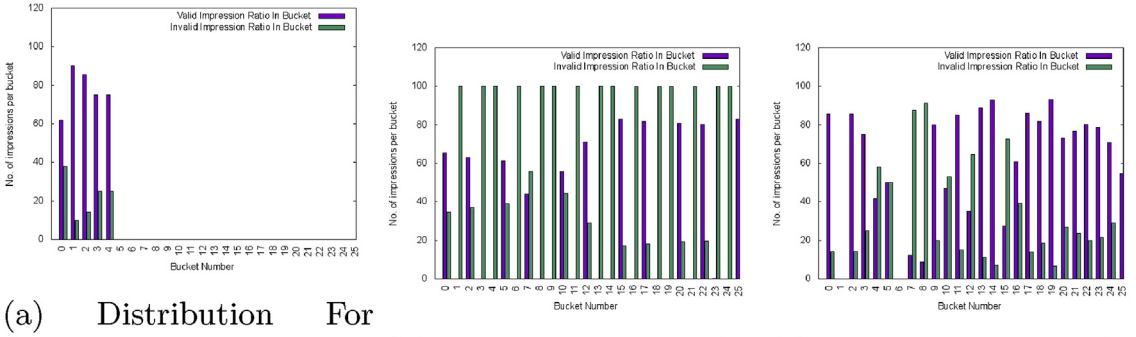
Acknowledgement

The authors gratefully acknowledge the constructive comments of the reviewers and editor that helped to improve the manuscript.

Appendices

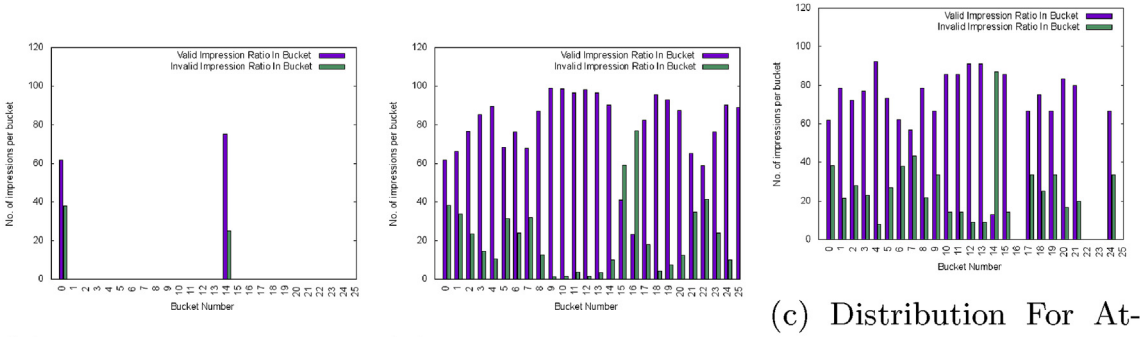
A. Relative frequency historgrams of attributes

The range of values for each of the attributes was measured. For each attribute, the range was divided into 25 equally spaced sub-ranges or buckets. Then the frequency within each of these buckets is calculated. The ratio of *Ok* and *Fraud* impressions was computed and for each attribute the ratios were plotted and as illustrated in the following figures. From these figures we can gather knowledge about the probability of an impression being *Ok* or *Fraud* given that its value lies in a specific bucket. As an example if we consider Fig. 7c we can see that if the value of the feature *delAvgDelay/IP* of an impression lies in bucket 5 or higher then it is more likely to be fraudulent.



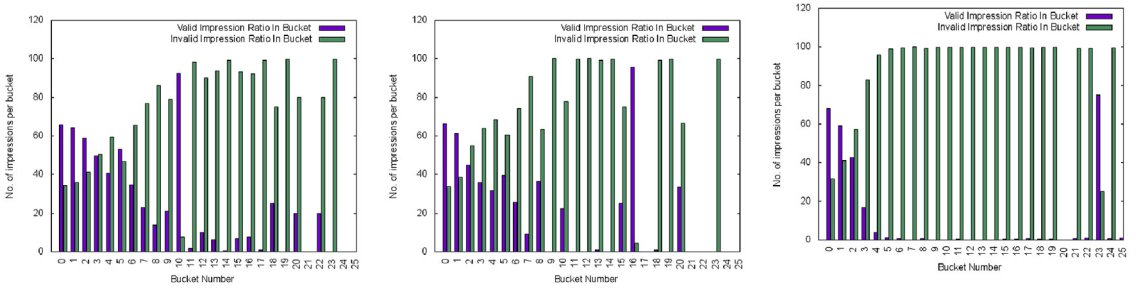
(a) Distribution For Attribute MaxSameDelCount (b) Distribution For Attribute NoOfEvents (c) Distribution For Attribute Event/DelRatio

Fig. 5. Frequency histogram of ok and fraud impressions for each feature, where the range of the feature is divided into 25 buckets.



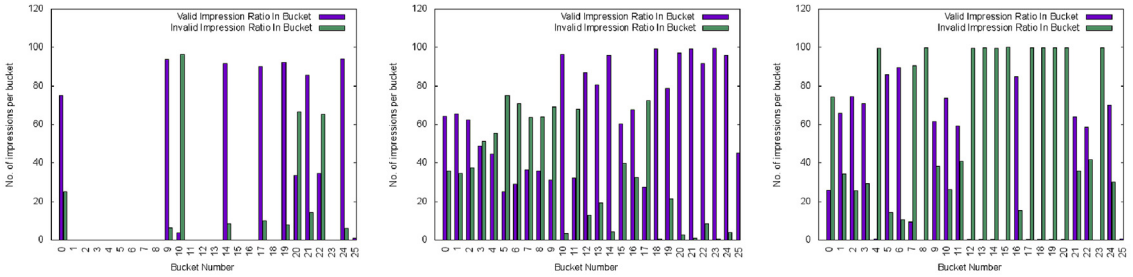
(a) Distribution For Attribute EventVariance (b) Distribution For Attribute OnscreenTime (c) Distribution For Attribute avgEventTimeDelay

Fig. 6. Frequency histogram of ok and fraud impressions for each feature, where the range of the feature is divided into 25 buckets.



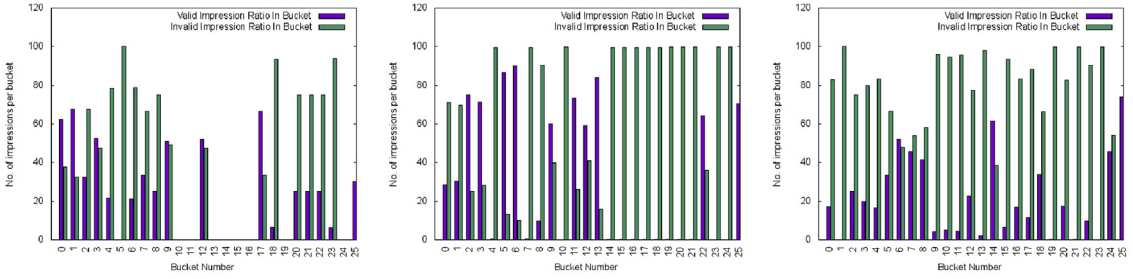
(a) Distribution For Attribute totCount/IP (b) Distribution For Attribute uniqDel/IP (c) Distribution For Attribute delAvgDelay/IP

Fig. 7. Distribution For Attribute delAvgDelay/IP.



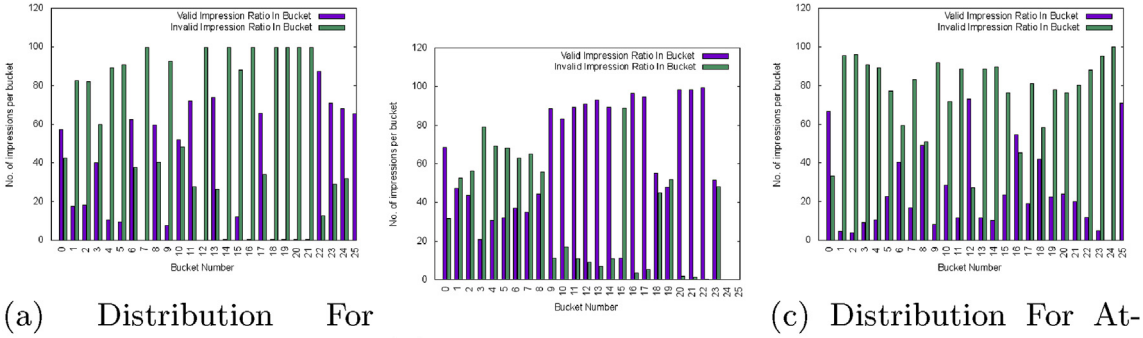
(a) Distribution For Attribute imAvgDelay/IP (b) Distribution For Attribute delPerDeviceId (c) Distribution For Attribute delPerSite

Fig. 8. Frequency histogram of ok and fraud impressions for each feature, where the range of the feature is divided into 25 buckets.



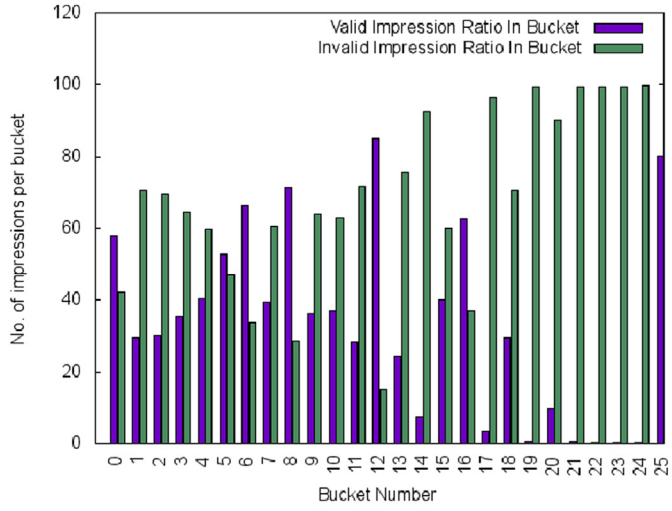
(a) Distribution For Attribute averageTimeDifference/Site (b) Distribution For Attribute uniqDelCount/Site (c) Distribution For Attribute impressionToDelRatio/Site

Fig. 9. Frequency histogram of ok and fraud impressions for each feature, where the range of the feature is divided into 25 buckets.



(a) Distribution For Attribute viewToDelRatio/Site (b) Distribution For Attribute delCount/UID (c) Distribution For Attribute impressionToDelRatio/UID

Fig. 10. Frequency histogram of ok and fraud impressions for each feature, where the range of the feature is divided into 25 buckets.



(a) Distribution For Attribute viewToDelRatio/UID

Fig. 11. Frequency histogram of ok and fraud impressions for each feature, where the range of the feature is divided into 25 buckets.

B. *Structure of Raw Data*

The following tables contain examples of raw data (from Tables 14–17) and the derived data (from Table 18 to Table 19). Due to the high number of features each entry in raw data and derived data are divided in to multiple tables.

Table 14
Raw Data Features Table 1.

deliveryId	timestamp	valid	clientIp	ad Rel Type	no Ad Reason Code	forced Ad	test Ad	uid	dnt	market Id	operator Id	surfSessionId
108622529703	1461038590850	true	82.132.222.131	1	null	false	false	1437449715949209025	0	38	27	7233217910121725041
108622535801	1461038599414	true	2.217.92.189	1	null	false	false	3095678361550945129	0	38	null	70035778878362091 19
108623578679	1461039979573	true	78.32.150.174	1	null	false	false	1257400425310567492	0	38	null	6954611415863115989
108652369173	1461058303705	true	90.209.191.67	1	null	false	false	4003745188221743484	0	38	null	2560402886813982492
108652369173	1461058303705	true	90.209.191.67	1	null	false	false	4003745188221743484	0	38	null	2560402886813982492
108697970322	1461081126021	true	90.218.177.189	1	null	false	false	645800178837198762	0	38	null	7034923235144336819

Table 15
Raw Data Features Table 2.

deliveryId	uid Creation Timestamp	ad Space Id	account id	siteid	clientId	client Version Id	web View Sdk Version	unknown Device Id	osId	platform Id	device Model Id	platform Type Id	browser Id
108622529703	1456818163978	11269	1553	2028	3	112	null	13904157	25000	200	100304	1	46
108622535801	1441205235000	8995	1553	2848	5	110	null	14510468	19300	100	104772	1	33
108623578679	1459597280284	11741	1553	2010	5	110	null	13978136	26000	200	106955	1	2
108652369173	1452812897267	11979	1553	3243	5	110	null	14457833	26000	200	106526	1	40
108652369173	1452812897267	11979	1553	3243	5	110	null	14457833	26000	200	106526	1	40
108697970322	1459629862321	13315	1553	2013	5	110	null	14510881	19300	100	101767	1	33

Table 16
Raw Data Features Table 3.

deliveryId	browser Version	ip Market Id	ip Country Code	ip Region	ip City	ip Isp	ids	connection Type	mcc	device Model
108622529703	4.0	38	GB	null	null	Telefonica O2 UK	null	null	null	null
108622535801	9.3.1	38	GB	A7	Birmingham	Sky Broadband	null	null	null	“iPhone7,2”
108623578679	49.0.2623.105	38	GB	H8	Liverpool	Entanet	null	null	null	null
108652369173	49.0.2623.105	38	GB	R3	Belfast	Sky Broadband	null	null	null	null
108652369173	49.0.2623.105	38	GB	R3	Belfast	Sky Broadband	null	null	null	null
108697970322	9.3.1	38	GB	C6	Newquay	Sky Broadband	null	null	null	“iPhone5,2”

Table 17
Raw Data Features Table 4.

deliveryId	ad Space Req Url	cookie	private Mode	is Impression	impression Type	impression Time	status
108622529703	feda889bd2325811783d61ff4e320c3b	64c83b632389ca1145858e9636f9ea07	null	true	0	1461038591182	OK
108622535801	1970e5e5d8c2db3bfdd044c9c2c32e79	db2dd401739cdac83088b4e86a826656	null	false	0	1461038599683	OK
108623578679	6450e38eef745e520d45e7b4847356e4	4686146a629c74e33fe4ba3847c02cb4	null	false	1	1461040714192	OK
108652369173	f9599c7dcd0ae6ae077331888fbe04cc	335ba4fa27b4e049621092dd488de3dd	null	true	0	1461058306526	Fraud
108652369173	f9599c7dcd0ae6ae077331888fbe04cc	335ba4fa27b4e049621092dd488de3dd	null	false	1	1461058307982	Fraud
108697970322	95449231ad1a1ea8c03ecc833c1a29c5	ab1b3353e46688995c5d2fe60a073f71	null	false	1	1461081198623	Fraud

Table 18
Derived Features Example Table 1.

deliveryId	Max Same Del Count	No. of Events	Event/ DelRatio	Event Variance	On screen Time	avg Event Time Delay	tot Count/IP	uniq Del/IP	delAvg-Delay/IP	imAvg-Delay/IP	del- Per Device ID	del- Per Site	average Time Difference/ Site	Uniq delCOunt/ Site
108622246907	5	12	0.75	14.6667	56486	3530.38	12	5	2860323	2861136	4	924	563794	586
108622272508	2	7	0.875	0.857143	13904	1738	5	3	7434949	7434943	5	9613	55143	5673
108622274694	1	8	1	0	11180	1397.5	3	2	4574357	4574435	2	17503	30261	7746
108622396899	1	5	1	0	1030	206	12	6	41399784	41399845	4	132	4009304	66
108622791780	1	5	1	0	59861	11972.2	2	1	0	29652	2	11401 8	45764	6694
108623043489	19	12	0.4	297	32502	1083.4	98	62	5360335	5360370	3	105	13542	58

Table 19
Derived Features Example Table 2.

deliveryId	click ToImpressionRat/Site	click ToViewRat/Site	del Count/UID	click ToImpressionRat/Uid	click ToViewRat/Uid	status
108622246907	0.406143	0.163823	1	1	1	OK
108622272508	0.610435	0.0802045	3	0.666667	0	OK
108622274694	0.784534	0.471469	1	1	0	OK
108622396899	1	0	2	1	0	Fraud
108622791780	0.583209	0.109949	1	1	0	Fraud
108623043489	0.706897	0.0862069	1	1	1	Fraud

References

Alexander Tuzhilin. Alexander tuzhilin report. https://googleblog.blogspot.com/pdf/Tuzhilin_Report.pdf. (Online; Last Accessed 13 November 2016).

Bing. How microsoft advertising helps protect advertisers from invalid traffic. https://advertising.yahoo.com/. (Online; Last Accessed 13 November 2016).

Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O., Kegelmeyer, W Philip, 2002. Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.

Cho, Geumhwan, Cho, Junsung, Song, Youngbae, Choi, Donghyun, Kim, Hyoungshick, 2016. Combating online fraud attacks in mobile-based advertising. EURASIP J. Inf. Secur. 2, 2016.

Crussell, Jonathan, Stevens, Ryan, Chen, Hao, 2014. Madfraud: investigating ad fraud in android applications. In: The 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'14, Bretton Woods, NH, USA, June 16-19, 2014, pp. 123–134.

Dave, Vacha, Guha, Saikat, Zhang, Yin, 2012. Measuring and fingerprinting click-spam in ad networks. In: ACM SIGCOMM 2012 Conference, SIGCOMM '12, Helsinki, Finland - August 13-17, 2012, pp. 175–186.

Haddadi, Hamed, 2010. Fighting online click-fraud using bluff ads. Computer Communication Review 40 (2), 21–25.

Hall, M.A., 1998. Correlation-based Feature Subset Selection for Machine Learning PhD thesis. University of Waikato, Hamilton, New Zealand.

Inc Google. Ad Traffic Quality Resource Center. Definition of invalid click activity. https://support.google.com/adsense/answer/16737?hl=en&ref_topic=1348720%&rd=1. (Online; Last Accessed 13 November 2016).
Juels, Ari, Stamm, Sid, Jakobsson, Markus, 2007. Combating click fraud via premium clicks. In: Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007.
Kirk, J. Take down bamital click-fraud botnet. http://www.infoworld.com. (Online; Last Accessed 13 November 2016).
Kohavi, Ron, John, George H., 1997. Wrappers for feature subset selection. Artif. Intell. 97 (1–2), 273–324 Special issue on relevance.
Leader awards. http://www.leadscouncil.com/leader-awards/. (Online; Last Accessed 13 November 2016).
Liu, Bin, Nath, Suman, Govindan, Ramesh, Liu, Jie, 2014. DECAF: detecting and characterizing ad fraud in mobile apps. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014, pp. 57–70.
Lovric, Miodrag (Ed.), 2011. International Encyclopedia of Statistical Science. Springer.
Majumdar, Saugat, Kulkarni, Dhananjay, Ravishankar, Chinya V., 2007. Addressing click fraud in content delivery systems. In: INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA, pp. 240–248.
Metwally, Ahmed, Agrawal, Divyakant, Abbadi, Amr El, 2005. Duplicate detection in click streams. In: Proceedings of the 14th International Conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, pp. 12–21.
Metwally, Ahmed, Agrawal, Divyakant, Abbadi, Amr El, 2005. Using association rules for fraud detection in web advertising networks. In: Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30-September 2, 2005, pp. 169–180.
Metwally, Ahmed, Agrawal, Divyakant, Abbadi, Amr El, 2007. Detectives: detecting coalition hit inflation attacks in advertising networks streams. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007, pp. 241–250.
Metwally, Ahmed, Emekçi, Fatih, Agrawal, Divyakant, Abbadi, Amr El, 2008. SLEUTH: single-publisher attack detection using correlation hunting. PVLDB 1 (2), 1217–1228.
Mobile ad spend to top $100 billion worldwide in 2016, 51% of digital market. http://www.emarketer.com/Article/Mobile-Ad-Spend-Top-100-Billion-Worldwide-2016-51-of-Digital-Market/1012299. (Online; Last Accessed 13 November 2016).
Oentaryo, Richard Jayadi, Lim, Ee-Peng, Finegold, Michael, Lo, David, Zhu, Feida, Phua, Clifton, Cheu, Eng-Yeow, Yap, Ghim-Eng, Sim, Kelvin, Nguyen, Minh Nhut, Perera, Kasun S., Neupane, Bijay, Faisal, Mustafa Amir, Aung, Zeyar, Woon, Wei Lee, Chen, Wei, Patel, Dhaval, Berrar, Daniel, 2014. Detecting click fraud in online advertising: a data mining approach. J. Mach. Learn. Res. 15 (1), 99–140.
Perera, Kasun S., Neupane, Bijay, Faisal, Mustafa Amir, Aung, Zeyar, Woon, Wei Lee, 2013. A novel ensemble learning-based approach for click fraud detection in mobile advertising. In: Mining Intelligence and Knowledge Exploration, First International Conference, MIKE 2013, Tamil Nadu, India, December 18-20, 2013, pp. 370–382.
Rajab, Moheeb Abu, Zarfoss, Jay, Monrose, Fabian, Terzis, Andreas, 2006. A multifaceted approach to understanding the botnet phenomenon. In: Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006, pp. 41–52.
Roesner, Franziska, Kohno, Tadayoshi, Moshchuk, Alexander, Parno, Bryan, Wang, Helen J., Cowan, Crispin, 2012. User-driven access control: rethinking permission granting in modern operating systems. In: IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA, pp. 224–238.
Schwartz, B. Google: Investigating invalid adsense traffic is extremely difficult. http://www.seroundtable.com. (Online; Last Accessed 13 November 2016).
Sonja Murdoch Moinak Bagchi and Jay Scanlan. The state of global media spending. http://www.mckinsey.com/industries/media-and-entertainment/our-insights/the-state-of-global-media-spending. (Online; Last Accessed 13 November 2016).
Springborn, Kevin, Barford, Paul, 2013. Impression fraud in on-line advertising via pay-per-view networks. In: Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013, pp. 211–226.
Stone-Gross, Brett, Stevens, Ryan, Zarras, Apostolis, Kemmerer, Richard A., Kruegel, Christopher, Vigna, Giovanni, 2011. Understanding fraudulent activities in online ad exchanges. In: Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2–, 2011, pp. 279–294.
Traffic predator. https:/, 2016 Traffic predator. https://dbtechlabs.com/blog/video-tutorials/traffic-predator. (Online; Last Accessed 13 November 2016).
University of Waikato. Weka. www.cs.waikato.ac.nz/ml/weka.
Winning the fight against ad fraud. https://forensiq.com/. (Online; Last Accessed 10 June 2017).
Yahoo. Traffic Quality. We work to protect you in a variety of ways. http://community.bingads.microsoft.com. (Online; Last Accessed 13 November 2016).
Yu, Fang, Xie, Yinglian, Ke, Qifa, 2010. Sbotminer: large scale search bot detection. In: Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010, pp. 421–430.
Zeroaccess is top bot in home networks. http://www.infosecurity-magazine.com. (Online; Last Accessed 13 November 2016).
Zhang, Qing, Ristenpart, Thomas, Savage, Stefan, Voelker, Geoffrey M., 2011. Got traffic?: an evaluation of click traffic providers. In: Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality. ACM, pp. 19–26.

Ch. Md. Rakin Haider is working as a Lecturer of Department of Computer Science and Engineering (CSE) in Bangladesh University of Engineering and Technology (BUET). Currently, he is pursuing M.Sc in Computer Science and Engineering from BUET. Before this, he obtained his B.Sc. degree from the same department. His research area includes Spatial Database, Machine Learning, Big Data Analytics.

Anindya Iqbal received B.Sc.Eng. (hons.) and M.Sc.Eng degrees in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 2005 and 2009, respectively. He has received PhD degree from Monash University, Australia in 2013. He is an Assistant Professor in the department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). His major research interests are in the fields of wireless sensor networks, participatory sensing system, security & privacy, and empirical software engineering. He has published more than 15 refereed publications in reputed journals and conferences including Ad Hoc Networks, Journal of Network and Computer Application, WoWMoM, LCN, NCA, WCNC, etc. He has rich consultancy experience in the area of System Analysis and Design and Security.

Atif Hasan Rahman is an Assistant Professor in the Department of Computer Science and Engineering (CSE) at the Bangladesh University of Engineering & Technology (BUET). He completed his PhD from the University of California, Berkeley in 2015 under the supervision of Lior Pachter. His research area is bioinformatics and computational biology and is currently focusing on statistical methods for genome assembly and analysis. He was a recipient of Fulbright Science and Technology Fellowship in 2009.

Dr. M. Sohel Rahman is a Professor of the CSE department of BUET. He had previously worked as a Visiting Senior Research Fellow of King's College London. He is a Senior Member of both IEEE and ACM and a member of American Mathematical Society (AMS) and London Mathematical Society (LMS). He is also a Peer-review Associate College Member of EPSRC, UK. Dr. Rahman's research interest includes, but is not limited to, theory and algorithms, stringology, metaheuristics, applied informatics, computational biology and bioinformatics. Among his highly cited results are the work on high dimensional Knapscak problems, algorithms on different variants of sequence alignment problems, data structures for different variants of string and sequence matching and sufficient conditions for Hamiltoninicity and metaheuristics solutions for hard real-life problems in different branches of science and engineering. Dr. Rahman regularly writes reviews at Mathematical Review and ACM Computing Review.