Computing and Informatics, Vol. 40, 2021, 606–627, doi: 10.31577/cai_2021_3_606

REAL TIME MOBILE AD INVESTIGATOR: AN EFFECTIVE AND NOVEL APPROACH FOR MOBILE CLICK FRAUD DETECTION

Iroshan Aberathne

Department of Information and Communication Technology
Faculty of Technology, University of Sri Jayewardenepura
Sri Lanka
e-mail: iroshan@sjp.ac.lk

Chamila Walgampaya

Department of Engineering Mathematics
Faculty of Engineering, University of Peradeniya
Sri Lanka
e-mail: ckw@pdn.ac.lk

Abstract. Today, mobile advertising is considered as the most effective medium to convey promotional messages to customers because of the excessive usage of mobile phones and tablets all around the world. However, this ecosystem has severely been affected by fraudulent activities due to a large sum of money circulated in the advertising industry. The term ad fraud is referred to as any kind of fraudulent activities that are executed by fraudulent users either a human or an automated script. The combat between researchers and fraudulent users never ends because more smarter strategies are being used by the fraudsters to bypass the significant number of detection and prevention solutions. The Real Time Mobile Ad Investigator-RTMAI is proposed as a software solution to address this problem where a novel supervised learning algorithm based on the hidden Markov model along with a rule engine have been proposed to classify fraudulent impressions in real time. Furthermore, RTMAI proposed a solution to address the class imbalance problem which is generic to most of the classification datasets. The experimental results show the significance of the proposed approach to classify the fraud or non-fraud clicks/events, impressions and even user sessions more confidently in real time.

Keywords: Mobile advertising, click fraud, classification, supervised learning, class imbalance, hidden Markov model

1 INTRODUCTION

The inception of online advertising goes to the early nineties where the very first online advertisement, which was a banner advertisement of a web magazine named HotWired, had been published in 1994 on a web page owned by AT & T [1]. The Mobile Marketing Association has defined the term Mobile Advertising as *a form of advertising that transmits advertisement messages to users via mobile phones or other wireless communication devices* [2]. The recent statistics show that 33 % of the world population has been using smartphones, which is nearly more than 2.5 billion users [3]. Significant increase of mobile Internet browsing in recent years has led to an increase in the popularity of advertising in mobile devices. The analysis states that currently 51 % of digital advertising market share will be dominated by mobile advertising and predicted to be 70 % by 2019 with the worth of US$ 200 billions [4].

The key contributors of this market are *User, Advertisers, Publishers,* and *Advertising Networks known* as *Ad networks.* Users are individuals who surf websites or use mobile apps where they see the ads shown on web pages or within the apps that they may click on the ads. Advertiser is the one who designs the ads and makes a contract with an ad network to publish advertisements on behalf of himself or a company [5]. Publisher can be a web site or mobile application which displays the advertisements to the site/app visitors [5]. Generally, Large publishers often sell around 60 % of their ad space known as ad inventories though ad networks and smaller ones sell their entire inventories [6]. Ad network are online companies which play a broker role between advertiser and publisher. Advertisers are charged by the ad networks for publishing their advertisements. Ad networks find suitable publishers to display ads [6].

The most popular revenue models in this industry are *Pay-Per-Impression (PPI)* and *Pay-Per-Click (PPC)* models where internet content providers are paid by the advertiser per each impression or click [7, 8, 9] An impression is defined as the displaying or loading event of advertisement into an advertisement frame while click would be any kind of user interaction/event on the displayed advertisement.

In this study, we propose a software solution called real Time Mobile Ad Investigator (RTMAI). The RTMAI has been implemented by encapsulating the Rule Engine, a Behavioural Pattern Recognition module and a novel supervised machine learning model. These modules interact with each other simultaneously to achieve the final goal of classifying clicks/events, impressions and subsequently user sessions in real time. The remainder of this paper is organised as follows. In Section 2 discuss existing detection and prevention systems. Proposed approach of this study is discussed in detail under Section 3. The experimental results are available in Section 4. Conclusion is given in Section 5.

2 LITERATURE SURVEY

The researchers have proposed and implemented a number of click or impression fraud detection tools using distinct methodologies. Xu et al. [10] have developed a detection system where they used a stepwise evaluation process including proactive functionality test at front end (i.e. user interface) backed by JavaScript and passive examination of browsing behaviour to differentiate a clickbots from a human clicker. DECAF [11] proposed an offline click fraud detection approach using rule-based methods to detect placement fraud by analysing the advertisement user interface status in mobile apps/pages. NAB (Not-A-Bot) [12] is a system that enables a range of verifier policies for applications that would like to separate human-generated requests from bot traffic. NAB approximately identifies and certifies human-generated activities. Client machines of the NAB system attest the legitimacy of individual requests to remote parties with a trusted component that monitors keyboard and mouse input. FCFraud is an operating system anti-malware service proposed by Iqbal et al. [13] inspects HTTP packets from all user processes and analyzes the ad-related traffic from captured HTTP packets. The FCFraud detects malware that is a part of a botnet and launches attacks using technologies similar to the desktop malware.

MAdFraud [14] studies mobile ad fraud perpetrated by Android apps and identifies two kinds of fraudulent behavior. First one is requesting ads in the background and the second one is clicking on ads without user interaction. They further developed an analysis tool to automatically trigger and expose ad fraud in Android emulators. However, the intrinsic limitation of offline testing on coverage and the lack of a reliable way to distinguish benign from fraud ads make it hard for such approaches to detecting sophisticated means of doing frauds, especially bot-driven frauds. AdAttester [15] tries to detect and prevent well-known ad fraud by identifying incoming click or impression is actually delivered by a real user with two primitives called verifiable display and unforgeable clicks. AdAttester cannot detect mobile ads that violate the ad policy of the ad provider. Walgampaya et al. [16] has used a multi-level data fusion process to detect and prevent click fraud in real time and decision. Agarwal [17] also suggested a real time click fraud detection technique which mainly focuses on the advisor side. Either approach has used the same mathematical theory called the Dempster-Shafer evidence theory to tackle fraudulent clicks in desktop environments.

Several machine learning (ML) approaches have also been experimented by researchers to improve the accuracy, performance and reliability of fraudulent clicks and impressions detection mechanisms in mobile advertising. Perera et al. [18] evaluated a number of ML algorithms such as decision trees, regression trees, artificial neural networks and support vector machines on real data produced by Buzzcity Ltd. The researchers were able to identify a number of different fraudulent patterns in the data set but did not focus on detecting each individual event called impression. Haider et al. [4] has discussed another ML based approach which is similar to previous authors' approach but these authors were able to achieve better improvement with identifying each individual fraud impression than the common pattern of fraudulent events. Botnets detection approach was proposed by Gobel [19] using the hidden Markov model in their study. The proposed approach has used network traffic generated by computers to model the HMMs so that bots can be identified through measuring the distances between these HMMs.

3 REAL TIME MOBILE AD INVESTIGATOR-RTMAI

Real Time Mobile Ad Investigator – RTMAI is an extended version of Real Time Mobile Bot Miner – RTMBM [5] to address click and impression fraud problems in the pay-per-click internet advertising model. The proposed solution is capable enough to integrate not only in mobile but also in desktop environments. The RT-MAI is a composition of rule engine, behavioural pattern recognition techniques [5], advanced algorithms, novel machine learning and resampling algorithms to identify fake user events/clicks, impressions and subsequently user sessions in real time.

3.1 System Architecture

The abstract view of RTMAI architecture is graphically represented in Figure 1. There are four modules named Data Collection, Data Processing, Decision Making and Admin modules that incorporate each other to build the overall system. The RTMAI has neither functionality nor implementation change made to Data Collection or Data Processing modules compared to RTMBM [5]. However, a new Admin Module has been implemented in RTMAI while enhancing the capability of Decision Making module with novel supervised learning algorithm.

RTMAI uses two approaches to make a decision on whether the use event or session is valid or fraud. Rule engine is implemented with two subsystems called Horizontal and Vertical Analysis Sub System as the first approach. Secondly, a novel supervised learning algorithm has been implemented based on the hidden Markov model to classify each individual click or impression into either fraud or non fraud category in real time. The proposed model is enhanced with another new sequence data resampling technique to solve the class imbalanced problem in the dataset.

The admin module is the place where all the admin tasks are performed. Session handler is responsible to manage session validation through heart-beat and scheduler algorithms [5]. Report handler is the admin dashboard to visualize all the analytics results along with user session and event details.

3.2 Rule Engine

The RTMAI is capable of identifying user events via a data filtering process which is implemented with novel advanced algorithms. The front end java script consists of all the business logics to identify advanced user events such as Touch Zoom Event (TZE) over Swap Touch Zoom Event (STZE) where TZE event is triggered when

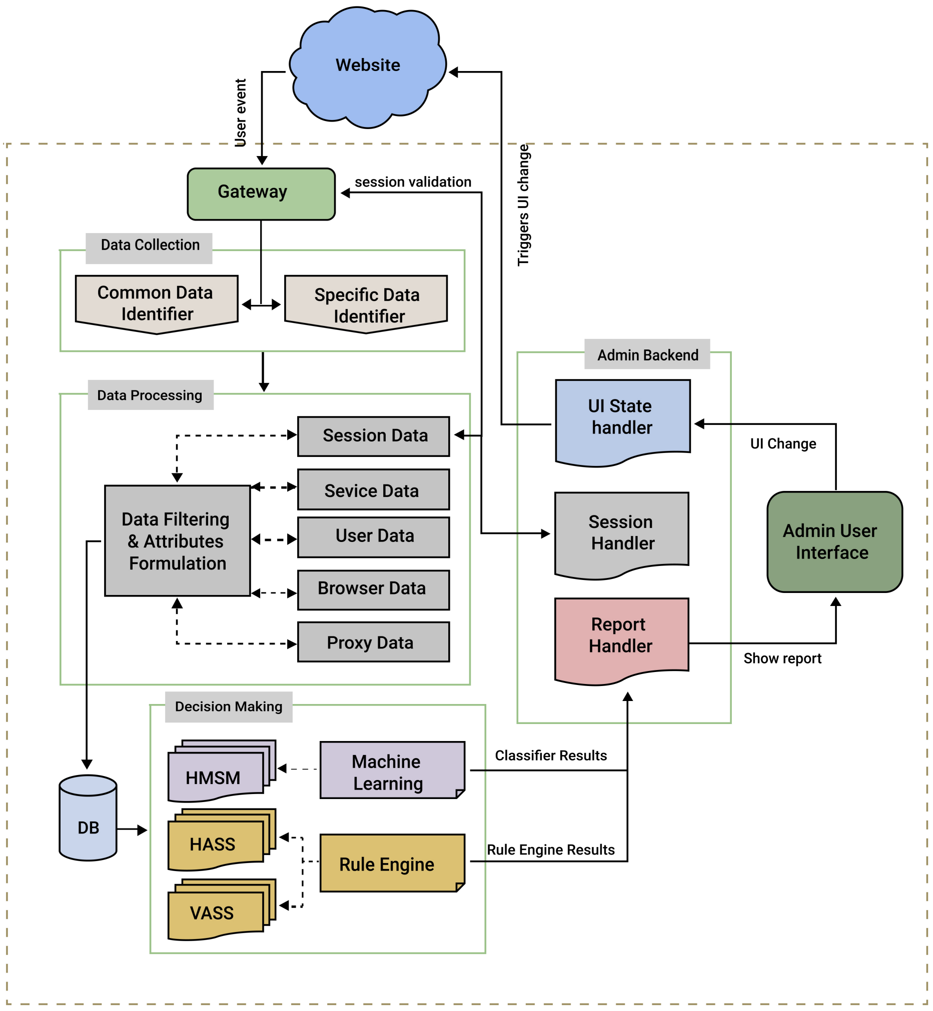

Figure 1. System architecture

user clicks on mobile screen twice in order to zoom HTML component. The STZE is identified when a particular user does the HTML component zoom using his/her finger tips rather perform two clicks during a tiny time period. Such tricky user events are captured by the RTMAI very accurately. The decision making module has a rule engine where pre-processed data are being analysed to label as fraud or not. A set of static rules have been defined under Horizontal and Vertical analysis sub systems [5].

3.3 Hidden Markov Scoring Model-HMSM

The Hidden Markov Scoring Model – HMSM is a novel supervised learning classification algorithm which is implemented as a part of the decision making module. The HMSM is based on the scoring approach of HMM rather than conventional probabilistic models of HMM. The proposed methodology discusses an N to N process from feature selection to target state classification via fully automated process. The initial version of the HMSM algorithm was evaluated based on a labeled dataset borrowed from Haider et al. [4] before production implementation.

3.3.1 Dataset

The dataset contains a number of attributes such as deliveryId, timestamp, clientIp, marketId, adSpaceId, accoundId, siteId, unknownDeviceId, clientVersionId, ipMarketId, ipCountyCode, ipIsp, adRelType, forcedAd, eventType, eventId, eventTimeStamp and status, etc. Details of the attributes can be found in [4].

3.3.2 Derived Attributes

Derived attributes were introduced to the dataset out of existing variables with respect to individual impression so that dimension of the feature vector will be reduced. eventCount, distEventTypes, surfTimeSec are some of the derived attributes:

- *eventCount* is the number of triggered events per impression,
- *distEventTypes* is the number of distinct event types of a given impression,
- *surfTimeSec* is the number of seconds users engage with an impression,
- *maxEventCount* is the maximum event count out of *distEventTypes*,
- *distEventFreqGroups* is the number of distinct event frequency groups,
- *dayOfWeek* is an impression triggered date.

Finally, the dataset arranged in ascending order of the timestamp variable to guarantee the state transition from previous state to the next state which is a fundamental nature of HMM.

3.3.3 Feature Vector

HMM basically interacts with state transition. All the numerical variables were transformed into categorical variables through entropy-based binning technique based on the target variable. Once numerical variables transformations are completed, all the categorical variables were evaluated with chi-square test of independence against the target variable with a significance level of 0.05. The variables which have higher chi-square test statistics than critical value were identified as the observe or emission variables.

Altogether nine variables were identified by the chi-square test which have higher test statistics than critical value. Out of these nine variables, five were derived attributes and the rest of them were raw attributes. The selected feature vector is fed into the proposed hidden Markov scoring model as emission variables.

3.3.4 HMSM Algorithm Implementation

The HMSM algorithm calculates scores for each target class based on observe variables to classify the data point in supervised learning approach. In a supervised learning approach, a model should be trained first and then makes the predictions with trained parameters. Calculation of A, B and π is referred to as training the model in HMSM. Equations (1), (2) and (3) were used to calculate the A, B and π with the training data set. The mathematical representation of the hidden Markov model is defined by A, B, π and can be denoted by λ [20] where $\lambda = (A, B, \pi)$.

$$A_{(i,j)} = p\left(\frac{S_t = j}{S_{t-1} = i}\right), \quad \forall_i = 1, \ldots, M, \sum_{i=1}^{M} A_{(i,j)} = 1, \tag{1}$$

$$B_{(j,k)} = p\left(\frac{O_t = k}{S_t = j}\right), \tag{2}$$

$$\pi_i = p(S_1 = i), \quad \forall_i = 1, \ldots, M, \sum_{i=1}^{M} \pi_i = 1, \tag{3}$$

where,

- M = total number of hidden states,
- $i, j = 1, \ldots, M$ index the state,
- k = number of possible discrete observations,
- s = hidden state,
- s_t = hidden state at time t,
- π = initial state distribution, a vector of size M,
- A = state transition probability matrix, a matrix of size $M * M$,
- B = emission probability matrix, a matrix of size $M * K$,
- x_t = observation at time t.

HMSM algorithm classifies the test data with a scoring model based on HMM. Target variable of the experimental data set has two states called OK and Fraud, where OK being the click is genuine and the Fraud being that the click is not genuine. The HMSM classifies each individual record in test data into either state.

Feature Score ($\mathbf{fscore_{f,s}}$): Calculates scores towards each hidden state (i.e. Fraud or OK) of the target variable called $fscore_{Fraud}$ and $fscore_{OK}$ for each individual record in test dataset with respect to each emission feature. Equation (4) is defined as the mathematical representation of fscore:

$$fscore_{f,s=i} = -\left\{\log \pi_i + \log A_{i,s} + \log B_{s,k}\right\}. \tag{4}$$

Mean Deviation Feature Score ($\mathbf{mdfscore_{f,s}}$): Equation (5) calculates the mean scores for each subset (i.e. $fscore_{Fraud}$ and $fscore_{OK}$) in order to calculate the deviation of the feature score from its subset mean represented in Equation (6):

$$\mu_{s=i} = mean\left(fscore_{s=i}\right), \tag{5}$$

$$mdfscore_{f,s=i} = fscore_{f,s=i} - \mu_{s=i}, \tag{6}$$

$$\forall_i = \{Fraud, OK\}, \forall_f = 1, \ldots, F.$$

Minimum Mean Deviation Score ($\mathbf{mdscore_{min,s=i}}$): Select minimum mean deviation score out of *mdfscore* for each hidden status as in Equation (7):

$$mdscore_{min,s=i} = \min\left(mdfscore_{s=i}\right). \tag{7}$$

HMSM identifies the most probable hidden state of a given record as the state of the maximum $mdscore_{min,s=i}$, as shown in Equation (8):

$$hiddenState = \max\left(mdscore_{min,OK}, mdscore_{min,Fraud}\right) \tag{8}$$

where s = hidden states and F = number of observe features.

3.4 Step-Factor Resampling and Smoothing Technique

Any kind of dataset that is related to fraud detection including the click fraud suffers with Class Imbalance problem [21, 22]. The class imbalance occurs when the dataset carries only a small number of data points representing the minor class compared to the major class in a particular dataset [23]. A novel methodology is introduced to solve this class imbalance problem in a sequence data via a resampling technique and smoothing approach.

3.4.1 Smoothing Technique

One of the major smoothing techniques called Additive smoothing is used in this study to upturn the zero probabilities for unseen data using ε as a tuning parameter since it provides better estimates compared to the other methods. Here, ε is referred to as the smoothing factor in Additive smoothing. Estimated probabilities p_ε based on the actual counts can be calculated as in Equation (9) for each value x

of a variable X in a sample of N observations [24]:

$$p_\varepsilon = \frac{(x+\varepsilon)}{(N+\varepsilon * N_x)} \tag{9}$$

where N_x is the number of possible values contained in the sample space.

3.4.2 Resampling Technique

The proposed resampling technique identifies the major and minor classes from a dataset and then calculates the optimum number of records to be converted so called *Conversion Factor* (Z) from major class into minor class in order to balance the dataset. The derivation of the proposed resampling technique is illustrated as follows:

- C^i_{major} – number of major class instances before resample,
- C^i_{minor} – number of minor class instances before resample,
- C^f_{major} – number of major class instances after resample,
- C^f_{minor} – number of minor class instances after resample,
- Z – number of instances to be converted from major to minor class,

$$R_i = \frac{C^i_{major}}{C^i_{minor}}, \tag{10}$$

$$R_f = \frac{C^f_{major}}{C^f_{minor}}. \tag{11}$$

Since $C^f_{major} < C^i_{major}$ and $C^f_{minor} > C^i_{minor}$, the relationship among the major and minor classes before and after the resampling can be expressed with Z, as shown in Equations (12) and (13):

$$C^f_{major} = C^i_{major} - Z, \tag{12}$$

$$C^f_{minor} = C^i_{minor} + Z. \tag{13}$$

The Equation (11) is rearranged according to the Equations (12) and (13) so that Equation (14) is derived:

$$R_f = \frac{C^i_{major} - Z}{C^i_{minor} + Z}. \tag{14}$$

The formula to calculate conversion factor – Z is derived with Equations (10) and (14):

$$Z = \left(\frac{R_i - R_f}{R_f + 1}\right) C^i_{minor}. \tag{15}$$

The R_f is considered as a tuning parameter to the algorithm where different values from 1 to $R_i - R_f = 1$ are assigned to R_f so that a set of Z values can be calculated according to the Equation (15).

Once the conversion factors are calculated, the respective number of records should be converted from major to minor class in order to overcome the class imbalance problem. There should be a consistent procedure to perform this conversion. The proposed method calculates the index distance of adjacent minor class instances and arranges them in three different ways called ASC, DESC and MID to perform the conversion so that the consistency is guaranteed. The term Step-Factor is used to refer to these minor class instances arrangement methods. The target class of the dataset contains two states called OK and Fraud where OK is genuine and Fraud means not genuine instances. The target class sequence is arranged as a list so that the index distance of adjacent minor class instances can be calculated. The graphical representation of this process is shown in Figure 2.

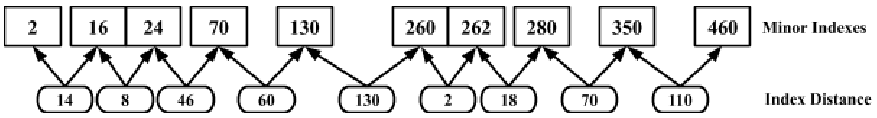

Figure 2. Index distance of minor class instances

The implementation of the first two step-factors are straightforward. The index distances are arranged in ascending order in ASC and descending order in DESC method. Merge sort algorithm is used to perform these said sorting because the worst case time complexity of the merge sort is $O(n \log n)$ which enhances the efficiency of step-factor algorithm. The index distances are arranged by the MID step-factor following a shuffling mechanism. The MID step-factor algorithm first calculates the number of index distances available in the list. If the number of indexes is odd, the last index value of the list is omitted and then the index distance list so called initial index distance list is divided into two sublists named left and right index distance sub lists. Then the shuffling starts at the last index value of the left index distance sub list and then first index value of the right index distance sub list and so on. The algorithm performs conversion of major to minor class instance with respect to selected step-factor. The shuffling process of the indexes based on the MID step-factor is illustrated in Figure 3.

4 EXPERIMENTAL RESULTS

The Rule engine was evaluated with real world data which have been gathered from a web based application. The HMSM and its enhancement with resampling and smoothing technique was tested with labeled dataset [4] before integrating with the current version of RTMAI in production.

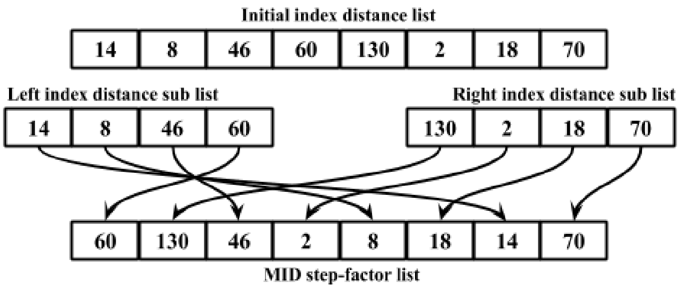
Initial index distance list
14
8
46
60
130
2
18
70
Left index distance sub list
14
8
46
60
Right index distance sub list
130
2
18
70
60
130
46
2
8
18
14
70
MID step-factor list

Figure 3. MID step-factor index shuffling process

4.1 Rule Engine

We have tested RTMAI with automatically generated scripts which are implemented with selenium web driver to imitate real user behaviuor and mobile emulators such as android and web browser emulators.

4.1.1 Horizontal Analysis Sub System – HASS

Table 1 shows the experimental results of a user session with respect to the horizontal analysis subsystem in the decision making module where each individual user event is analysed. The user has triggered 6 events from a desktop device and out of those 6 events three were identified as mobile events. The rule engine applies static rules on top of this information and then identifies that this user session has been created by an automated script and flagged respective events as fraudulent events. The particular user tried to simulate mobile user behaviour in a desktop but RTMAI smartly identified its fraudulent activities.

4.1.2 Vertical Analysis Sub System – VASS

Experimental results of vertical analysis or user session analysis categorised into three segments called device, user and event data analysis. Figure 4 is a real screenshot of VASS analysis dashboard view. *Device Analysis* graph shows that both mobile and desktop events have been triggered in this particular user session. Then the device behaviour can be fraud to some extent.

The attribute values in user analysis must be constant and cannot be changed for a given session. If there is a variance of an attribute, the system identifies it as a negative behaviour. The experimental results in the *User Analysis* graph shows that there are deviated behaviours in time zone, event sequence and user location. This user is trying to hide his or her real geolocation by altering time zone offset and location details. Meanwhile, there is a mismatch in the event sequence as well.

Final analysing aspect of the vertical analysis sub system is event behaviour analysis where the experimental results of a particular user events are shown in the

Event	Tag	EventTime	FingerTips	Event (X, Y)	TimeZone	ZoneTime	Device	Device (W, H)	View (W, H)	View	Browser	Proxies	Status
TS	DIV	20:17:50.208	1	1 152, 289	+05:30	20:17:52.926	Desktop	1 366, 768	1 366, 447	−1	Chrome	1	Fraud
TZE	H3	20:17:49.83	1	982, 216	+05:30	20:17:51.797	Desktop	1 366, 768	1 366, 447	−1	Chrome	1	Fraud
TS	H3	20:17:48.708	1	540, 205	+05:30	20:17:51.441	Desktop	1 366, 768	1 366, 447	−1	Chrome	1	Fraud
SE	−1	20:16:59.744	−1	0, 0	+05:30	20:17:02.493	Desktop	1 366, 768	1 366, 447	−1	Chrome	1	OK
LC	DIV	20:16:59.447	1	438, 345	+05:30	20:17:02.16	Desktop	1 366, 768	1 366, 447	−1	Chrome	1	OK
SE	−1	20:16:33.931	−1	0, 0	+05:30	20:16:37.336	Desktop	1 366, 768	1 366, 447	−1	Chrome	1	OK

Table 1. Horizontal analysis

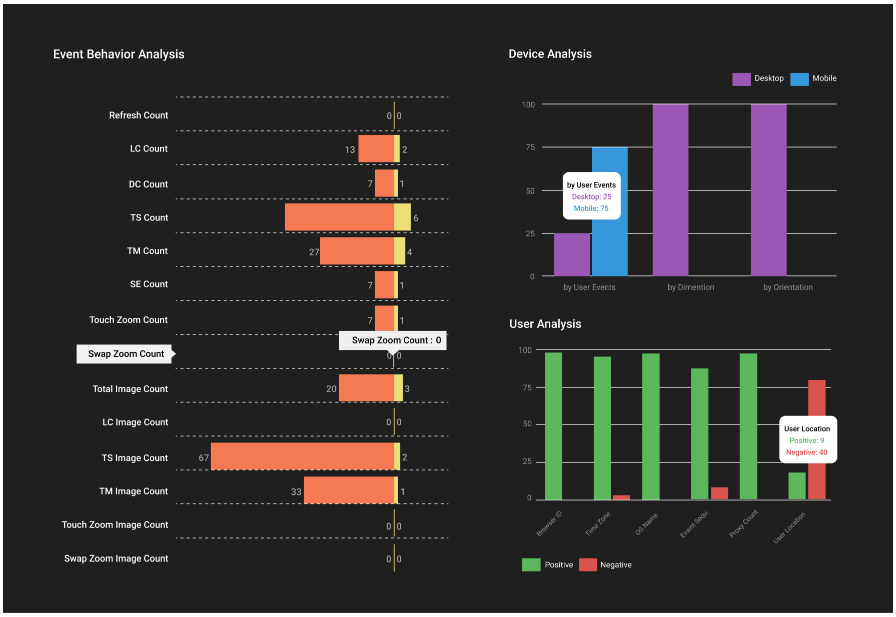

Figure 4. Dashboard screenshot in VASS

Event Behaviour Analysis graph. The user tries to pretend as a mobile user but a small number of desktop events have been captured by the RTMAI. There are 6 touch start events (40 % out of total events) but there is less number of complex mobile events such as touch zoom event (only 1), swap zoom event (zero event). The real reason is, it is difficult to automate more advanced mobile events by scripts. Comparing the probability of user events RTMAI can make appropriate decisions.

4.2 Hidden Markov Scoring Model – HMSM

The HMSM was evaluated with 20 different test samples in order to verify the performance of the model across the dataset. An individual train/test sample is represented by S-xx notation as shown in Figure 5 where the samples are equally distributed throughout the dataset so that consistency of the classifier can be evaluated by calculating accuracy, precision, recall, specificity and F-score for each individual sample as the performance measures.

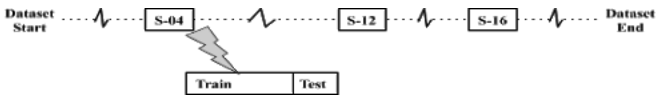

Figure 5. Train/test samples distribution

The fundamental problem of any machine learning algorithm is finding optimum training set size to perform regression or classification tasks accurately. To solve this issue HMSM trains with 10 distinct training samples where sample size starts from 3 000 records to 25 000 records to find the optimum training sample size. Each training set evaluated with 20 different test samples and calculated the mean value and mean of the standard deviation of performance measures to identify the optimum training set record size.

The HMSM model performs well in classifying test data when training sample size is 5 000. Figure 6 illustrates the model performance with all training data sets where mean values for accuracy, recall, precision and F-score have been reached to above 80 % and specificity reached to 79 % at the sample size containing 5 000 records. The stability of the model can be evaluated with the mean values of standard deviations of each training sample. The results show that the model is more stable when the training data set has 5 000 records where standard deviations of all the performance measures are less than 0.25 including specificity.

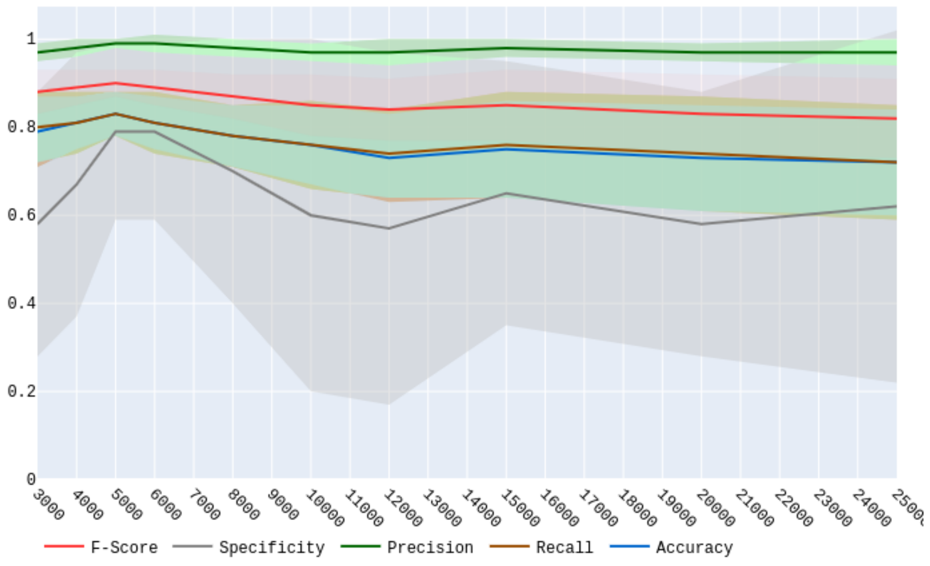

Figure 6. Training set evaluation by mean and standard deviation

The observe variables which have initially been identified with the chi-square test of independence were categorised into three groups.

Derived features: all derived features,

Raw features: all initial features without any prepossessing,

Combined features: all derived and raw features.

The proposed HMSM classifier tested with a random test sample containing 50 data points based on each feature group so that the best performing feature vector can be identified. The experimental results are visualised in Figure 7 where the x axis represents the data point and the y axis represents the *Minimum Mean Deviation Score Difference: (mmdsd)* of each data point, as shown in Equation (16).

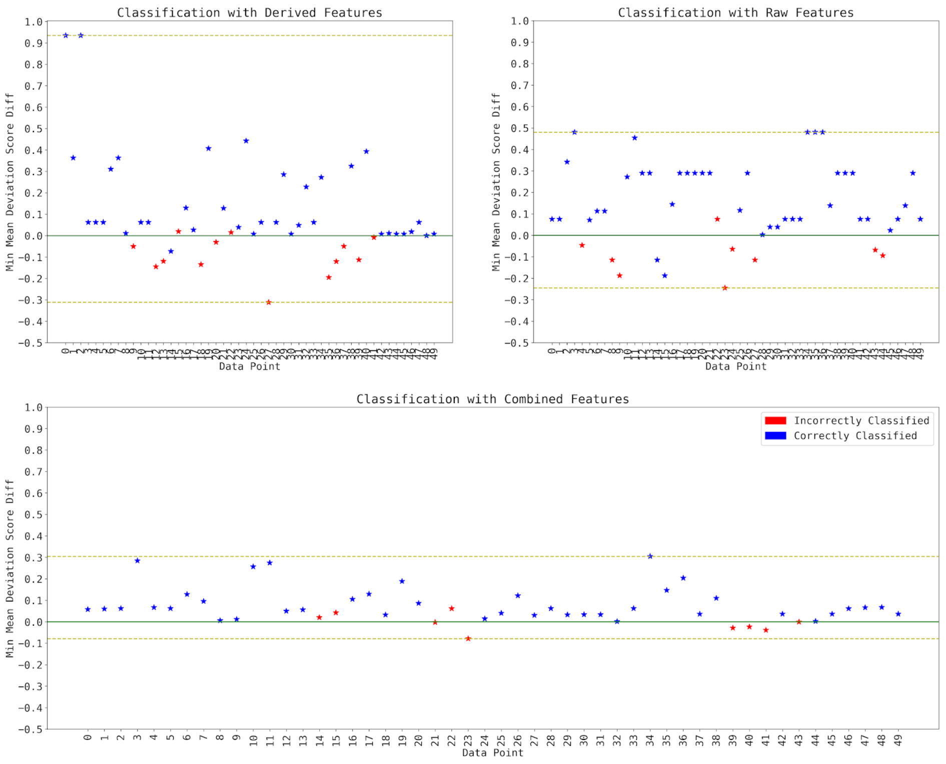

Figure 7. Classifier behaviour with types of features

$$mmdsd = mdscore_{min,OK} - mdscore_{min,Fraud}. \tag{16}$$

4.3 Step-Factor Resampling and Smoothing Technique

The HMSM was enhanced with smoothing factor – ε as a tuning parameter to the model in order to scale the performance of the model. The major advantage of smoothing factor is that it ensures the first priority objective of HMSM by increasing specificity while keeping almost constant values for all other performance measures. The experimental results in Figure 8 show that there is a significant effect on the model performance and stability for all training sets with the smoothing factor.

Moreover, it is clear that specificity is much more sensitive to the smoothing factor than all the other performance measures where a slight change in the smoothing factor produces significant performance change in specificity.

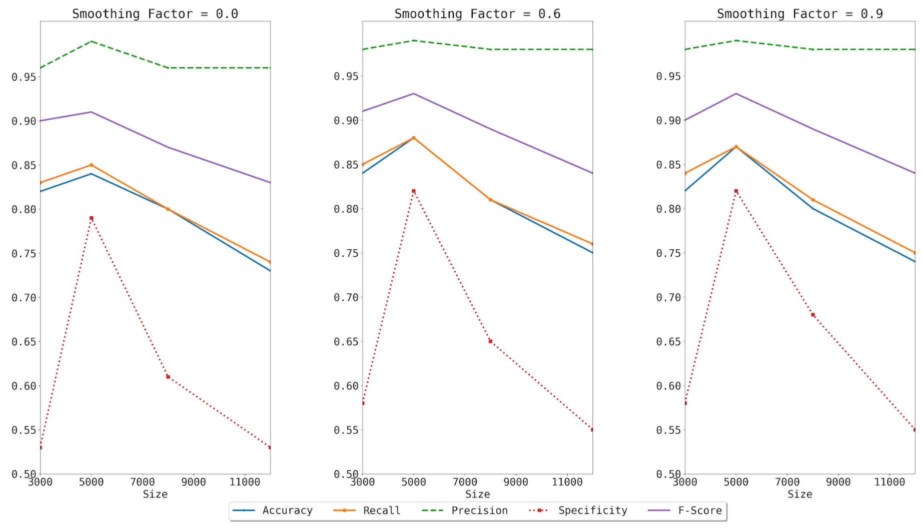

Figure 8. Model performance with smoothing factor

The proposed resampling technique is applied to sample datasets in order to experiment the performance enhancement of HMSM. Different datasets are validated against each step-factor to identify the effectiveness of each individual step-factor in this resampling approach. The model trains with the same training data set sizes that are used in smoothing factor to compare the performance variation between the two approaches. The experimental results in Table 2 shows the association of step-factor resampling approach towards a sequence data set. The model performs better when training data with 5 000 records as in smoothing factor. The experimental results illustrate that all step-factors are almost equally contributed to the model performance along with each training data set size.

The step-factor resampling technique has achieved comparatively greater statistics for each performance measure than smoothing with each training record size. Moreover, drastic changes in performance measures can not be experienced with respect to training record size in this approach like in smoothing approach. For instance, the statistics in specificity varies from 55 % to 82 % with the range of 17 in smoothing but that of resampling is from 75 % to 84 % where the range has been reduced to 9 % by step-factor resampling.

The resampling technique was applied to all 20 sample training datasets and then evaluated the HMSM model with the smoothing factor as 0.06. The experimental

Records	Step Factor	Acurracy		Recall		Precision		Specificity		F-Score	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
	ASE	0.86	0.06	0.87	0.06	0.99	0.02	0.77	0.29	0.92	0.04
3 000	MID	0.86	0.06	0.87	0.06	0.98	0.02	0.65	0.36	0.92	0.03
	DESC	0.85	0.08	0.86	0.08	0.99	0.02	0.68	0.36	0.92	0.05
	ASE	0.88	0.04	0.89	0.05	0.99	0.02	0.82	0.24	0.94	0.03
5 000	MID	0.89	0.03	0.89	0.04	0.99	0.02	0.8	0.27	0.94	0.02
	DESC	0.89	0.03	0.90	0.03	0.99	0.02	0.84	0.26	0.94	0.02
	ASE	0.84	0.10	0.84	0.10	0.99	0.02	0.79	0.30	0.90	0.07
8 000	MID	0.84	0.07	0.84	0.07	0.99	0.02	0.74	0.29	0.91	0.05
	DESC	0.84	0.05	0.84	0.06	0.99	0.02	0.73	0.31	0.91	0.03
	ASE	0.79	0.14	0.79	0.14	0.99	0.02	0.84	0.26	0.87	0.10
12 000	MID	0.80	0.11	0.80	0.11	0.99	0.02	0.86	0.23	0.88	0.07
	DESC	0.81	0.11	0.81	0.11	0.98	0.02	0.75	0.29	0.89	0.08

Table 2. Model performance with step-factor resampling

results show that step-factor plays a vital role in classifying test data. Each step-factor equally contributes to enhance the performance of the model, as shown in Figure 9, where 7 samples with DESC, 8 samples with ASC and 5 samples with MID step-factor has enhanced the performance of the model. Moreover, 17 out of 20, as a percentage of 85 %, samples show highest performance when the sample size equals or is less than 6 000 records. This approach including smoothing and resampling has been ensured the most valuable capability of HMSM to classify sequence data with small training dataset. The maximum percentage of conversion factor -Z out of the training size is 25 % and the rest of the samples are below 20 %. Furthermore, 16 samples have been reached to the optimum model performance when the conversion factor is below 10 %. There are only three samples which have been reached to 1 000 records to train the model in order to absorb better classification accuracy but out of these three, only one sample has a higher conversion rate.

There are four models for HMSM. The first model is identified as the initial model where neither smoothing factor nor resampling technique is applied. The second and third models are incorporated with smoothing factor and step-factor resampling technique respectively. The final model enhanced with both smoothing and step-factor resampling approaches to evaluate the HMSM model. The individual performance of both smoothing and resampling models are almost equal but have gained better classification optimisation compared with the initial model. A significant improvement of all performance measures can be identified after applying smoothing factor and resampling technique together as a hybrid approach on HMSM. The key objective of any fraud detection approach is to increase the specificity while keeping the statistics of all other performance measures in satisfactory level. The HMSM has achieved not only said objective with a dramatic increase of the specificity from 79 % to 91 %, but also other performance measures with almost

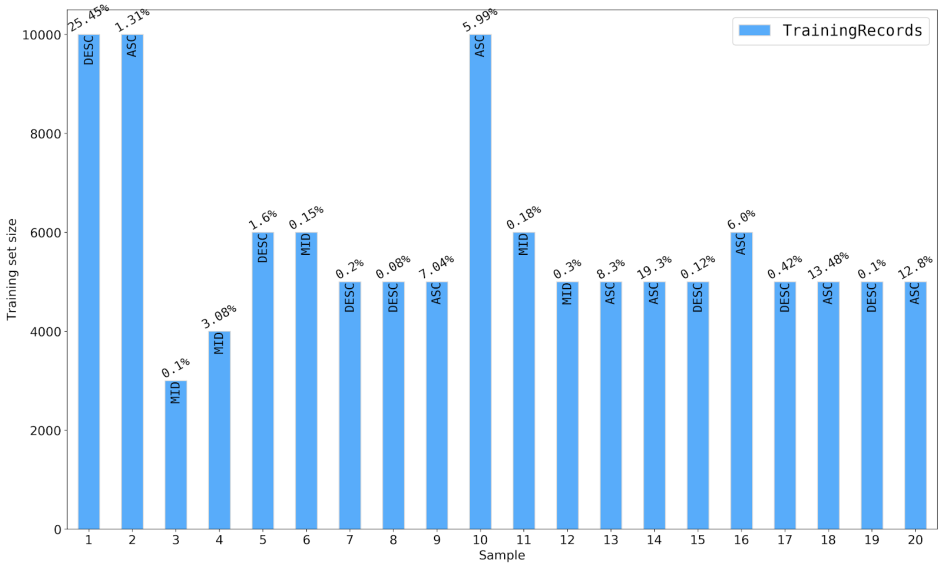

Figure 9. Best performing training set size along with step-factor

the same quantity. The accuracy and recall have been improved by 10 %, F-score by 6 % and precision has remained the same throughout the initial to final model with a tiny standard deviation change since it has already reached a higher statistics of 99 %. The consistency and stability of the model has also been guaranteed by the diminishing rate of standard deviation of each performance measure from initial model to final model. The experimental results for each individual model (Inital: Model-1, Smoothing: Model-2, Resampling: Model-3, Resampling and Smoothing: Model-4) have been summarised in Table 3.

	Model-1		Model-2		Model-3		Model-4	
	μ	σ	μ	σ	μ	σ	μ	σ
Accuracy	0.84	0.05	0.88	0.04	0.89	0.03	0.94	0.03
Recall	0.85	0.05	0.88	0.04	0.90	0.03	0.95	0.03
Precision	0.99	0.01	0.99	0.01	0.99	0.02	0.99	0.02
Specificity	0.79	0.26	0.82	0.23	0.84	0.26	0.91	0.17
F-Score	0.91	0.03	0.93	0.02	0.94	0.02	0.97	0.01

Table 3. Performance evaluation of the models

Experimental results show that HMSM model performs better after enhancing the model with smoothing factor and resampling technique. Figure 10 illustrates the HMSM performance across 20 different samples after applying the resampling and smoothing factor. There are only four instances with specificity less than 70 % and the lowest is 50 %. All other performance measures are above 90 % for all samples. Moreover, Figure 11 illustrates the performance of the all four models with the Receiver operating characteristic (ROC) curve to get a much better understanding.

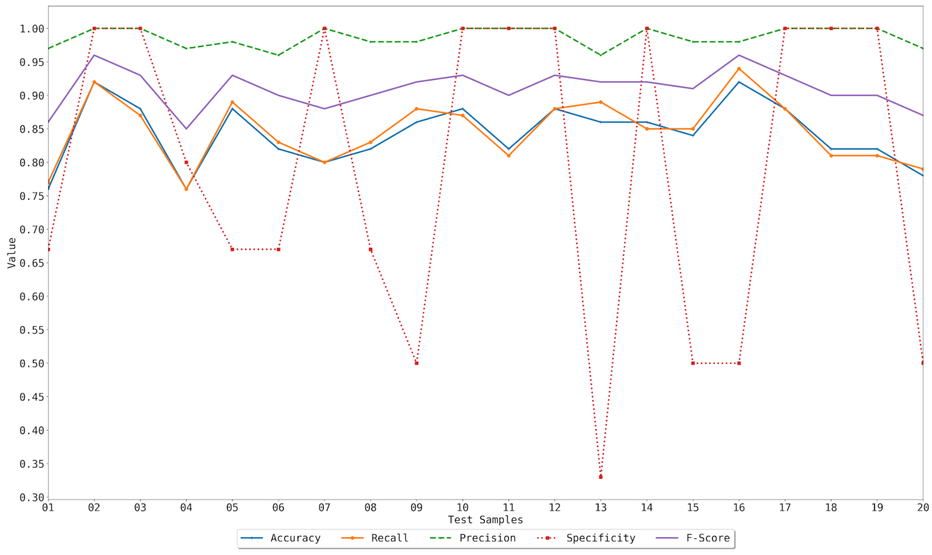

Figure 10. Best performing model

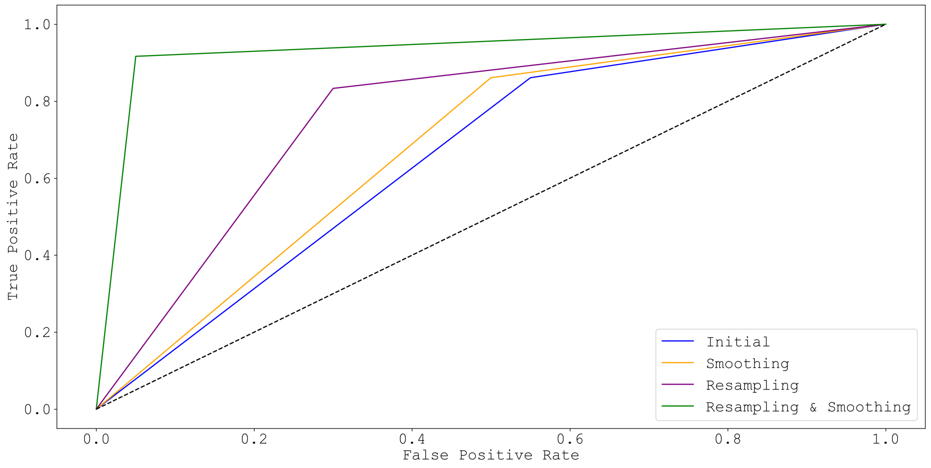

Figure 11. Models performance

5 CONCLUSION

The RTMAI is smart enough to identify fraud user events/clicks and sessions with a rule engine where HASS and VASS analyse each individual user event and session accordingly. Since RTMAI is a extended version of RTMBM, it encapsulated behavioural pattern recognition capability that facilitates more accurately second level verification on top of the VASS for user session identification. The Hidden Markov Scoring Model (HMSM), a novel supervised learning algorithm was introduced to the RTMAI to classify fraudulent impressions in addition to the classification of fraudulent clicks and subsequently user sessions with minimal computational power. The HMSM guarantees high performance with minimum training data in a smaller amount of time. A new resampling technique is proposed to address the class imbalance problem in sequence data. The findings of this study prove that proposed resampling and smoothing techniques perform well with the model in sequence data. There are a number of interesting features in this resampling technique. No synthetic data are introduced to the dataset. Natural patterns and the characteristics of the initial dataset are narrowly affected because the number of conversions from major to minor instances is very low. Altogether, the HMSM is a good alternative algorithm in supervised learning which reduces the computation time in training along with higher performance and higher learning efficiency on unseen data. The proposed model can be identified as a stable classification algorithm because it performs really well by identifying both positive and negative classes in higher performance measures. The optimum model shows significant capability as a fraudulent impression classifier with an average accuracy of 94 %, average precision of 99 %, average recall of 95 %, average specificity of 91 % and average F-score of 97 % across 20 different test samples with the standard deviation of 0.03, 0.02, 0.03, 0.17 and 0.01, respectively. Thus, this proposed RTMAI is a composition of several advanced techniques to solve almost all the dimensions of this click fraud detection domain.

REFERENCES

[1] Evans, D. S.: The Online Advertising Industry: Economics, Evolution, and Privacy. Journal of Economic Perspectives, Vol. 23, 2009, No. 3, pp. 37–60, doi: 10.1257/jep.23.3.37.

[2] Martins, J.—Costa, C.—Oliveira, T.—Gonçalves, R.—Branco, F.: How Smartphone Advertising Influences Consumers' Purchase Intention. Journal of Business Research, Vol. 94, 2019, pp. 378–387, doi: 10.1016/j.jbusres.2017.12.047.

[3] Tao, K.—Edmunds, P.: Mobile APPs and Global Markets. Theoretical Economics Letters, Vol. 8, 2018, No. 8, pp. 1510–1524, doi: 10.4236/tel.2018.88097.

[4] Haider, C. M. R.—Iqbal, A.—Rahman, A. H.—Rahman, M. S.: An Ensemble Learning Based Approach for Impression Fraud Detection in Mobile Advertising. Journal of Network and Computer Applications, Vol. 112, 2018, No. 15, pp. 126–141, doi: 10.1016/j.jnca.2018.02.021.

[5] Aberathne, I.—Walgampaya, C.: Smart Mobile Bot Detection Through Behavioral Analysis. In: Kolhe, M., Trivedi, M., Tiwari, S., Singh, V. (Eds.): Advances in Data and Information Sciences. Springer, Singapore, Lecture Notes in Networks and Systems, Vol. 38, 2018, pp. 241–252, doi: 10.1007/978-981-10-8360-0_23.

[6] Fridgeirsdottir, K.—Najafi-Asadolahi, S.: Cost-Per-Impression Pricing for Display Advertising. Operations Research, Vol. 66, 2018, No. 3, pp. 653–672, doi: 10.1287/opre.2017.1697.

[7] Alrwais, S. A.—Gerber, A.—Dunn, C. W.—Spatscheck, O.—Gupta, M.—Osterweil, E.: Dissecting Ghost Clicks: Ad Fraud via Misdirected Human Clicks. Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12), 2012, pp. 21–30, doi: 10.1145/2420950.2420954.

[8] Hu, Y.—Shin, J.—Tang, Z.: Performance-Based Pricing Models in Online Advertising: Cost Per Click Versus Cost Per Action. 2012, Georgia Institute.

[9] Mahdian, M.—Tomak, K.: Pay-Per-Action Model for Online Advertising. In: Deng, X., Graham, F. C. (Eds.): Internet and Network Economics (WINE 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4858, 2007, pp. 549–557, doi: 10.1007/978-3-540-77105-0_59.

[10] Xu, H.—Liu, D.—Koehl, A.—Wang, H.—Stavrou, A.: Click Fraud Detection on the Advertiser Side. In: Kutyłowski, M., Vaidya, J. (Eds.): Computer Security – ESORICS 2014. Springer, Cham, Lecture Notes in Computer Science, Vol. 8713, 2014, pp. 419–438, doi: 10.1007/978-3-319-11212-1_24.

[11] Liu, B.—Nath, S.—Govindan, R.—Liu, J.: DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps. 11th USENIX Symposium on Networked Systems Design and Implementation, 2014, pp. 57–70.

[12] Gummadi, R.—Balakrishnan, H.—Maniatis, P.—Ratnasamy, S.: Not-a-Bot: Improving Service Availability in the Face of Botnet Attacks. Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI '09), 2009, pp. 307–320.

[13] Iqbal, M. S.—Zulkernine, M.—Jaafar, F.—Gu, Y.: Protecting Internet Users From Becoming Victimized Attackers of Click-Fraud. Journal of Software: Evolution and Process, Vol. 30, 2017, No. 3, Art. No. e1871, doi: 10.1002/smr.1871.

[14] Crussell, J.—Stevens, R.—Chen, H.: MAdFraud: Investigating Ad Fraud in Android Applications. Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14), 2014, pp. 123–134, doi: 10.1145/2594368.2594391.

[15] Li, W.—Li, H.—Chen, H.—Xia, Y.: AdAttester: Secure Online Mobile Advertisement Attestation Using TrustZone. The 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15), Vol. 1, 2015, pp. 75–88, doi: 10.1145/2742647.2742676.

[16] Walgampaya, C.—Kantardzic, M.—Yampolskiy, R.: Real Time Click Fraud Prevention Using Multi-Level Data Fusion. Proceedings of the World Congress on Engineering and Computer Science (WCECS 2010), Vol. 1, 2010, 6 pp.

[17] Agarwal, A.: Automatic Detection of Click Fraud in Online Advertisements. M.Sc. Thesis, Texas Tech University, 2012.

[18] Perera, K. S.—Neupane, B.—Faisal, M. A.—Aung, Z.—Woon, W. L.: A Novel Ensemble Learning-Based Approach for Click Fraud Detection in Mobile Advertising. In: Prasath, R., Kathirvalavakumar, T. (Eds.): Mining Intelligence and Knowledge Exploration. Springer, Cham, Lecture Notes in Computer Science, Vol. 8284, 2013, pp. 370–382, doi: 10.1007/978-3-319-03844-5_38.

[19] Gobel, W.: Detecting Botnets Using Hidden Markov Models on Network Traces. 2008.

[20] Stamp, M.: A Revealing Introduction to Hidden Markov Models. 2004, pp. 26–56.

[21] Berrar, D.: Random Forests for the Detection of Click Fraud in Online Mobile Advertising. Proceedings of the 1st International Workshop on Fraud Detection in Mobile Advertising, 2012, pp. 1–10.

[22] Blundo, C.—Cimato, S.: SAWM: A Tool for Secure and Authenticated Web Metering. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE '02), 2002, pp. 641–648, doi: 10.1145/568867.568871.

[23] Somasundaram, A.—Reddy, U. S.: Data Imbalance: Effects and Solutions for Classification of Large and Highly Imbalanced Data. 1st International Conference on Research in Engineering, Computers and Technology (ICRECT 2016), 2016, pp. 1–16.

[24] Nivre, J.: Sparse Data and Smoothing in Statistical Part-of-Speech Tagging. Journal of Quantitative Linguistics, Vol. 7, 2000, No. 1, pp. 1–17, doi: 10.1076/0929-6174(200004)07:01;1-3;ft001.

Iroshan Aberathne is Senior Lecturer at the Department of Information and Communication Technology at Faculty of Technology, University of Sri Jayewardenepura, Sri Lanka. He completed his M.Phil. degree in engineering mathematics from Faculty of Engineering, University of Peradeniya, Sri Lanka in 2020 and received his B.Sc. degree with honours in computation and management from the Faculty of Science at the same university in 2015. He has industry and research experience in software engineering and data science in addition to his teaching experience. His research focuses on data science, machine learning, software engineering and computational modelling.

Chamila Walgampaya is Senior Lecturer at the Department of Engineering Mathematics at Faculty of Engineering, University of Peradeniya, Sri Lanka. He earned his B.Sc. in computer engineering with honours in 2001 from the Faculty of Engineering, University of Peradeniya, Sri Lanka and completed his M.Sc. and Ph.D. degrees from the School of Engineering at the University of Louisville, Kentucky, U.S.A. in 2006 and 2011, respectively. He has almost 20 years of extensive and diverse experience as an administrator, computer programmer, researcher and teacher. His research focus lies on click fraud mining, automatic web robots and agents, data and evidence fusion, ensemble methods and machine learning.