Journal of Machine Learning Research 14 (2013) 1-39 Submitted 3/13; Published x/13

Detecting Click Fraud in Online Advertising: A Data Mining Approach

Richard Oentaryo ROENTARYO@SMU.EDU.SG
Ee-Peng Lim EPLIM@SMU.EDU.SG
Michael Finegold MFINEGOL@ANDREW.CMU.EDU
David Lo DAVIDLO@SMU.EDU.SG
Feida Zhu FDZHU@SMU.EDU.SG
Living Analytics Research Centre
Singapore Management University / Carnegie Mellon University
80 Stamford Road, Singapore / 5000 Forbes Ave, Pittsburgh, PA
Clifton Phua CWCPHUA@I2R.A-STAR.EDU.SG
Eng-Yeow Cheu EYCHEU@I2R.A-STAR.EDU.SG
Ghim-Eng Yap GEYAP@I2R.A-STAR.EDU.SG
Kelvin Sim SHSIM@I2R.A-STAR.EDU.SG
Minh Nhut Nguyen MNNGUYEN@I2R.A-STAR.EDU.SG
Data Analytics Department, Institute for Infocomm Research
Connexis South Tower #21-01, 1 Fusionopolis Way, Singapore
Kasun Perera BPERERA@MASDAR.AC.AE
Bijay Neupane BNEUPANE@MASDAR.AC.AE
Mustafa Faisal MFAISAL@MASDAR.AC.AE
Zeyar Aung ZAUNG@MASDAR.AC.AE
Wei Lee Woon WWOON@MASDAR.AC.AE
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates
Wei Chen WEICHEN@COMP.NUS.EDU.SG
Dhaval Patel DHAVAL@COMP.NUS.EDU.SG
School of Computing, National University of Singapore
Computing 1, Computing Drive, Singapore
Daniel Berrar BERRAR.D.AA@M.TITECH.AC.JP
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama, Japan

Editor:

Abstract

Click fraud–the deliberate clicking on advertisements with no real interest on the product or service offered–is one of the most daunting problems in online advertising. Building an effective fraud detection method is thus pivotal for online advertising businesses. We organized a *Fraud Detection in Mobile Advertising* (FDMA) 2012 Competition, opening the opportunity for participants to work on real-world fraud data from BuzzCity Pte. Ltd., a global mobile advertising company based in Singapore. In particular, the task is to identify fraudulent publishers who generate illegitimate clicks, and distinguish them

from normal publishers. The competition was held from September 1 to September 30, 2012, attracting 127 teams from more than 15 countries. The mobile advertising data are unique and complex, involving heterogeneous information, noisy patterns with missing values, and highly imbalanced class distribution. The competition results provide a comprehensive study on the usability of data mining-based fraud detection approaches in practical setting. Our principal findings are that features derived from fine-grained time-series analysis are crucial for accurate fraud detection, and that ensemble methods offer promising solutions to highly-imbalanced nonlinear classification tasks with mixed variable types and noisy/missing patterns. The competition data remain available for further studies at `http://palanteer.sis.smu.edu.sg/fdma2012/`.

Keywords: Data mining, ensemble learning, feature engineering, fraud detection, imbalanced classification

1. Introduction

Advances in data management and web technologies have rendered online advertising as the ideal choice for small and large businesses to effectively target the appropriate marketing segments on the fly. The main coordinator in this setting is the advertising *commissioner* (also known as *ad network*), acting as a broker between advertisers and content publishers. An *advertiser* plans a budget, provides the commissioner with advertisements, and agrees on a commission for every customer action (e.g., clicking an ad, filling a form, bidding in an auction, etc). A *content publisher* contracts with the commissioner to display advertisements on their websites, and gets commissions based on the traffic it drives to the advertisers. This model, however, may incentivise dishonest publishers to generate illegitimate clicks on their sites–a major issue known as *click fraud*. Click fraud degrades the reliability of online advertising systems and, if not kept under control, can lead to a contraction of the advertising market in the long term. There have also been high-profile, costly litigations from unsatisfied advertisers, giving bad reputation for the commissioners. Thus, a reliable click fraud detection system is needed to help the commissioners proactively prevent click fraud and assure their advertisers that their dollars have been well spent.

To this end, we organized a *Fraud Detection in Mobile Advertising* (FDMA) 2012 Competition, centered around real-world mobile advertising data. The goal is to develop and crowdsource data mining and machine learning methods capable of building effective predictive models to detect fraudulent publishers. The competition offers a unique opportunity to work on click and publisher datasets provided by BuzzCity Pte. Ltd., a global mobile advertising network that has millions of consumers around the world (particularly in India, Indonesia and Africa) accessing internet contents and interacting on mobile phones and devices. Most publishers in the BuzzCity network adopt the *cost per click* (CPC) payment scheme, which is subject to abuses by malicious publishers through click fraud. In Q1 2012, over 45 billion ad banners were delivered across the BuzzCity network, having over 10,000 publisher sites and reaching an average of 300 million unique users per month. A fast and robust detection of the most predictive variables for fraudulent behavior is thus of great importance. Currently, BuzzCity uses an in-house developed detection mechanism to identify fraudulent publishers semi-automatically. The use of data mining and machine learning methods will provide more detailed insights for improving the detection accuracy, while reducing the efforts for manual interventions.

Accordingly, the FDMA 2012 Competition aims at providing an empirical platform to gauge the state-of-the-art data mining and machine learning methods in a setting typical of industrial applications. We summarize the key contributions of this paper below:

- We present an important application of machine learning and data mining methods to tackle real-world fraud detection problems, which serves as valuable resources for industrial and research practitioners. Thus far, there is a lack of comprehensive study on data mining/machine learning approaches for fraud detection in advertising.
- Our study involves proprietary, industrial data, which are rarely available and pose a challenging problem for many data mining and machine learning algorithms. The solutions presented in this paper address some important issues in data mining and machine learning research, including highly imbalanced distribution of the output variable, heterogeneous data (mixture of numerical and categorical variables), and noisy patterns with missing/unknown values.
- We show that exploratory data analysis and feature engineering are crucial milestones for effective fraud detection. In particular, we present systematic analysis of both *spatial* and *temporal factors* at different levels of granularity, which leads to creation of good, predictive features for accurate fraud detection.
- We investigate the applicability of a wide range of *single* and *ensemble learning* algorithms in fraud detection task. We found that the ensemble algorithms produce significant improvement over the single algorithms. Also, coupling ensemble learning with feature ranking analysis leads to discovery of the most important features for distinguishing between fraudulent and normal behaviors.
- To the best of our knowledge, FDMA 2012 is also the first open international competition and crowdsourcing initiative on fraud detection in online advertising. The results not only provide useful research insights, but also illustrate how companies (such as BuzzCity) can utilize data mining and machine learning methods to obtain useful, actionable knowledge for improving their business operations.

In this paper, we report the selected, winning entries of the FDMA 2012 Competition, which provide important insights on click fraud behavior. In Section 2, we give an overview of the competition data, challenges, and evaluation procedures. Table 1 summarizes the profiles of the winning teams and contributors of this paper, and Sections 3 to 6 elaborate in turn their "journeys" and key findings[1]. We next describe the work independently done by the competition organizer in Section 7. Section 8 provides concluding remarks.

2. Competition

2.1 Data

The raw data supplied by BuzzCity consist of two categories: *publisher database* and *click database*, both provided in comma-separated values (CSV) format. The publisher database

1. The full reports of all the teams are also available at our website: `http://palanteer.sis.smu.edu.sg/fdma2012/`

Rank	Team	Affiliation	Contribution
1	`starrystarrynight` • Clifton Phua • Eng-Yeow Cheu • Ghim-Eng Yap • Kelvin Sim • Minh-Nhut Nguyen	Data Analytics Department Institute for Infocomm Research, Singapore	Section 3
2	`TeamMasdar` • Kasun S. Perera • Bijay Neupane • Mustafa A. Faisal • Zeyar Aung • Wei Lee Woon	Masdar Institute of Science & Technology Abu Dhabi, United Arab Emirates	Section 4
3	`DB2` • Wei Chen • Dhaval Patel	School of Computing National University of Singapore, Singapore	Section 5
4	`Tea` • Daniel Berrar	Graduate School of Science & Engineering Tokyo Institute of Technology, Japan	Section 6

Table 1: Profile of the selected, winning teams in the FDMA 2012 Competition.

records the publisher/partners profile, and consists of several fields as listed in Table 2. On the other hand, the click database captures the click traffic associated with various publishers. Table 3 lists the fields in the click database. Table 4 provides a sample of the two largest publishers of each status in the training set, a `Fraud` and an `OK` publisher, and Table 5 lists three click samples from each publisher. There is another `Observation` status, comprising small number of new publishers, or publishers who have high click traffic and not yet deemed as fraudulent. Note that some fields in the publisher and click databases have been anonymized for privacy protection.

Field	Description
`publisherid`	Unique identifier of a publisher
`bankaccount`	Bank account associated with a publisher (anonymized; may be missing/unknown)
`address`	Mailing address of a publisher (anonymized; may be missing/unknown)
`status`	Label of a publisher, which falls into three categories: • `OK`: Publishers whom BuzzCity deems as having healthy traffic (or those who slipped their detection mechanisms) • `Observation`: Publishers who may have just started their traffic or their traffic statistics deviates from system wide average. BuzzCity does not have any conclusive stand with these publishers yet • `Fraud`: Publishers who are deemed as fraudulent with clear proof. BuzzCity suspends their accounts and their earnings will not be paid

Table 2: Fields in the publisher database.

2.2 Challenge

The FDMA 2012 competition aims at building a data-driven methodology for effective detection of fraudulent publishers. In particular, each participant is tasked to highlight potential `Fraud` publishers and distinguish them from `OK` and `Observation` (or collectively

Field	Description
`id`	Unique identifier of a particular click
`numericip`	Public IP address of a clicker/visitor
`deviceua`	Phone model/agent used by a clicker/visitor
`publisherid`	Unique identifier of a publisher
`campaignid`	Unique identifier of a given advertisement campaign
`usercountry`	Country from which the clicker/visitor is
`clicktime`	Timestamp of a given click (in yyyy-mm-dd format)
`referredurl`	URL where ad banners are clicked (anonymized; may be missing/unknown)
`channel`	Publisher's channel type, which consists of: • `ad`: Adult sites • `co`: Community • `es`: Entertainment and lifestyle • `gd`: Glamour and dating • `in`: Information • `mc`: Mobile content • `pp`: Premium portal • `se`: Search, portal, services

Table 3: Fields in the click database.

`publisherid`	`bankaccount`	`address`	`status`
8iaxj		14vxbyt6sao00s84	Fraud
8jljr			OK

Table 4: Publisher sample in raw training data. There are missing values in `bankaccount` and `address`. In pay per click online advertising, `Fraud` involves a large number of intentional click charges with no real interest in the advertisements, using automated scripts or click farms. The perpetrators can be the publishers themselves or their competitors, or the competitors of advertisers.

Normal) publishers, based on their click traffic and account profiles. This will help shed light on several key areas, such as identifying the common underlying fraud schemes or concealment strategies, understanding patterns of dishonest publishers, and developing new ways for effective prevention/detection plans.

`id`	`numericip`	`deviceua`	`publisherid`	`campaignid`	`usercountry`	`clicktime`	`channel`	`referredurl`
13417867	3648406743	GT-I9100	8iaxj	8fj2j	ru	2012-02-09 00:00:00	ad	26okyx5i82hws84o
13417870	3756963656	Samsung_S5233	8jljr	8geyk	in	2012-02-09 00:00:00	es	15vynjr7rm00gw0g
13417872	693232332	SonyEricsson_K70	8jljr	8gkkx	ke	2012-02-09 00:00:00	es	
13417893	2884200452	Nokia_6300	8jljr	8gp95	vn	2012-02-09 00:00:01	es	
13418096	3648406743	GT-I9100	8iaxj	8fj2m	ru	2012-02-09 00:00:08	ad	24w9x4d25ts00400
13418395	781347853	GT-I9003	8iaxj	8fj2j	ru	2012-02-09 00:00:20	ad	4im401arl30gc0gk

Table 5: Click samples in raw training data. There are missing values in `referredurl` and `deviceua`. The raw features include IP address of a clicker (`numericip`), mobile device model used by the visitor (`deviceua`), campaign ID of a particular advertisement campaign (`campaignid`), country of the visitor (`usercountry`), publisher channel type (`channel`), or an URL where the ad banner is clicked (`referredurl`).

More specifically, we seek to answer this question: Given historical patterns (of both fraudulent and normal publishers) in some time period (e.g., a 3 day period), how to detect fraudulent publishers in a future period (e.g., a 3 day period in the week after)? That is, we are interested in a detection/predictive model that can generalize well over time. To this end, BuzzCity provides three sets of publishers and clicks data taken from different time periods: a *training set* (for building predictive model), a *validation set* (for model selection/optimization), and a *test set* (for evaluating the models' generalization abilities and determining the competition winners). Each click dataset captures the click traffic over a 3 day period, while each publisher dataset records publishers receiving at least one click in that period. We summarize the count statistics of the publishers and clicks in Table 6.

It is worth noting that the publisher labels (i.e., `Fraud`, `Observation`, `OK`) were generated from BuzzCity's semi-automatic detection mechanism (cf. Section 1) that utilizes two types of auxiliary information together: *offline* and *online*; the former corresponds to information that BuzzCity deems impossible to automate or does not attempt to computerize (e.g., manually contact the publishers and verify their responses), while latter is obtained based on the statistical analysis of the click behavior done by BuzzCity's proprietary automated programs. Due to the proprietary nature of this practice and for simplicity, the details of the label generation process were not given as part of the competition. Here BuzzCity's primary interest is whether the competition participants can independently infer and discover fraudulent patterns based on the click and publisher databases alone, without using the auxiliary information. Also note that the `Fraud` and `Observation` publishers constitute very small portions of the population relative to the `OK` publishers (cf. Table 6), rendering this problem challenging for many contemporary classification methods.

			No. of publishers			
Dataset	Time period	No. of clicks	Fraud	Observation	OK	Total
Train	9-11 Feb 2012	3,173,834	72 (2.34%)	80 (2.60%)	2,929 (95.07%)	3,081
Validation	23-25 Feb 2012	2,689,005	85 (2.77%)	84 (2.74%)	2,895 (94.48%)	3,064
Test	8-10 Mar 2012	2,598,815	82 (2.73%)	71 (2.37%)	2,847 (94.90%)	3,000

Table 6: Statistics of the competition data.

2.3 Evaluation

For performance evaluation, we chose to use the *average precision* criterion, which favors algorithms capable of ranking the few useful items ahead of the rest. Such criterion is particularly suitable for detecting rare instances such as fraudulent publishers (Zhu, 2004). We briefly describe the criterion as follows: Let π be the number of relevant (i.e., actual `Fraud`) instances. We first ranked the instances according to the prediction/detection scores produced by each algorithm. Among the $t \times 100\%$ top-ranked instances, supposing $h(t) \leq t$ are truly relevant (also called *hits*), let $r(t) = \frac{h(t)}{\pi}$ and $p(t) = \frac{h(t)}{t}$ be the *recall* and *precision* respectively. Typically, $h(t)$ and $p(t)$ takes values only at a finite number of points $t_i = \frac{i}{n}, i = 1, 2, ..., n$. The average precision (AP) can then be computed using (1):

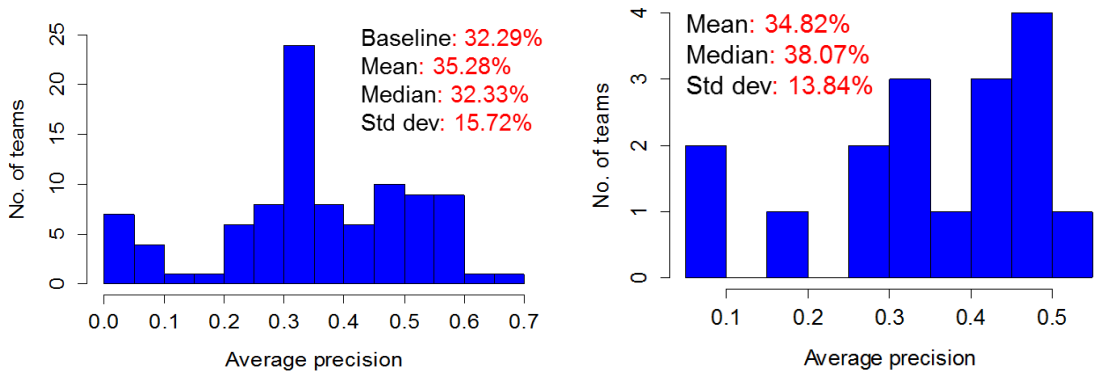
$$AP = \sum_{i=1}^{n} p(t_i) \Delta r(t_i) \tag{1}$$

where $\Delta r(t_i) = r(t_i) - r(t_{i-1})$.

Essentially, the AP criterion summarizes the precision-recall performances at different threshold levels, and corresponds to the area under the precision-recall curve (Zhu, 2004). In the case of fraud detection, simply evaluating precision and recall at a specific threshold level is inadequate, since these metrics vary with the strictness of a classification algorithm's threshold and the range of its prediction outputs. Further details on the AP criterion can be found in (Zhu, 2004).

2.4 Website and leaderboard

Our FDMA 2012 website supports a public leaderboard system displaying the best AP score and the submission time of each team on the validation set. We ran the competition for 1 month, from 1 to 30 September 2012, with two submissions per day allowed for each team. BuzzCity offered a total prize of 7,000 Singapore dollars (SGD) for the competition winners (i.e., SGD 4,000, 2,000, and 1,000 for the first, second, and third winners respectively). During the competition, the actual publishers' status labels (i.e., ground-truth) in the validation set were hidden, and our system computes the average precision for each submission. Rhe submitting teams received email notifications showing their current scores and submission time. This allows teams to track their progress and fine-tune their models. To ensure the models developed do not overfit the validation set, we used the test set for the final evaluations, the status labels of which were also hidden during the competition. The test set was only revealed 72 hours before the competition ended.



(a) (b)

Figure 1: Statistics of the average precision scores: (a) Public leaderboard (validation set), and (b) Private leaderboard (test set).

Figure 1(a) shows the overall statistics of the leaderboard scores as of September 30, 2012, while Figure 1(b) shows the statistics of the final test scores. In total, we had 127 teams registering for the competition, 88 of which explicitly specified their affiliation and country: 60 from academic institutions and 28 from industry, and 51 were local teams from Singapore and 37 were overseas teams. A total of 95 teams submitted to the leaderboard

Rank	Team	Average precision		Affiliation
		Validation set	Test set	
1	`starrystarrynight`	59.38%	51.55%	Institute of Infocomm Research
2	`TeamMasdar`	59.39%	46.42%	Masdar Institute of Science & Technology
3	`DB2`	62.21%	46.15%	National University of Singapore
4	`Tea`	51.55%	42.01%	Tokyo Institute of Technology
(*)	`LARC`	57.79%	55.64%	Singapore Management University

Table 7: Results of the top teams on the validation and test sets. Team ranks were determined using the test results.

during the competition period. As a baseline (for the leaderboard), we used the logistic regression method (Fan et al., 2008), which provides reasonable performance reference sufficiently close to the mean or median AP score.

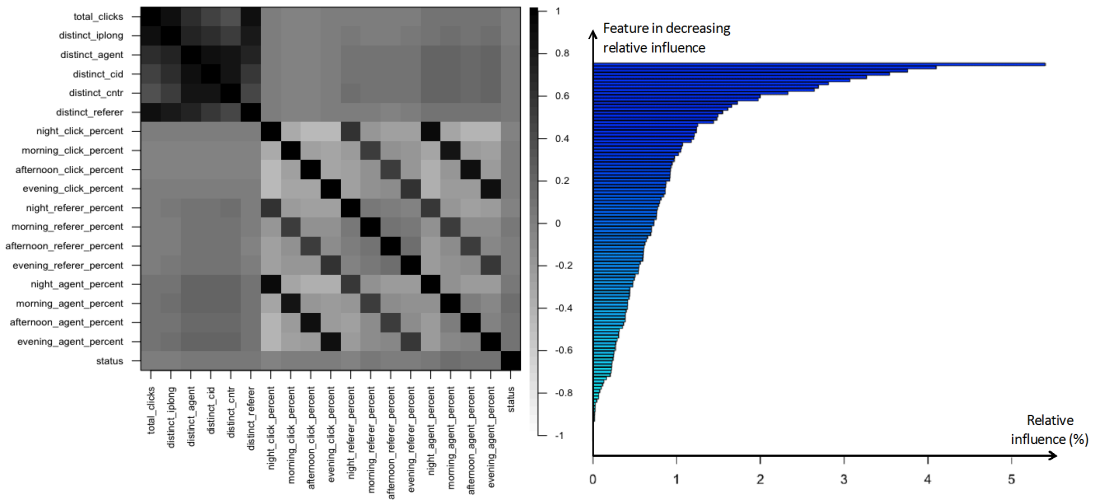
The final standings on the validation and test sets are summarized in Table 7. We also showed the best result obtained by the competition organizer (dubbed as team `LARC`) in the last row. Comparing the validation and test results, the position of the top-3 ranks were reversed, which may be attributed to overfitting.

3. First winner's entry

3.1 Preprocessing and feature extraction

Figure 2(a) plots the correlations among some of our features including `status`, which we used to ensure feature diversity - by excluding new features which are too similar to existing ones. In Figure 2(b), using specific model parameters described in the next section, we obtained the relative influence or importance of 118 predictive features in the final training set. A complete listing of the 118 features is available at `http://clifton.phua.googlepages.com/feature-list.txt`. In addition, there are two other features: `publisherid`, which is not used for model building, and `status`, which is the class or dependent feature. Adding all the features' relative influence will sum up to a score of one hundred. On one extreme, there are a few features with relative influence above three. On another extreme, there are a few features with negligible influence on the results such as `channel`-related features (see the Potential subsection on a discussion of leveraging the predictiveness of the raw `channel` feature). The average relative influence per feature is about 0.88%.

The 118 predictive features can be grouped into three types of features: 67 click behavior (57%), 40 repetitive click behavior (34%), and 11 high-risk click behavior (9%). The average rank of all the features (based on the model output, as described in Section 3.2) is 69, 35, and 69 respectively, meaning that duplicated clicks are likely to be invalid clicks. We use simple statistical features based on average, standard deviation, and percentages, and none of our features are created directly from `status` or specific values from raw anonymized features, such as `bankaccount`, `address`, `numericip`, `campaignid`, and `referredurl`.



(a) (b)

Figure 2: (a) Correlation plot of some click behavior features in the training set. (b) Relative influence of all features in the training set.

3.2 Method

3.2.1 Generalized boosted regression model

Gradient boosting is a machine learning technique used for classification problems with a suitable loss function, which produces a final prediction model in the form of an ensemble of weak prediction decision trees (Friedman, 2000). We used the implementation of generalized boosted regression model (GBM) in R's `gbm` package (Ridgeway, 2007). The final parameters used on the final training dataset for our best average precision on the test dataset are:

- `distribution` (loss function): "bernoulli" - also tested "Adaboost" distribution
- `n.trees` (number of iterations): 5000 - tested 100 to 5000 decision trees
- `shrinkage` (learning rate): 0.001 - tested 0.001 to 0.01
- `interaction.depth` (tree depth): 5 - tested 2 to 5
- `n.minobsinnode` (minimum observations in terminal node): 5 - tested 2 to 5

In the early to mid stages of the competition, we used two layers of GBM to select the most important features. During the final stages, we focused on only one layer of GBM as we had identified the three best types of features.

3.2.2 Other algorithms considered

When we first started, our team also tried out random forest (Breiman, 2001) (decision trees ensemble algorithm) in R's `randomForest` package, as well as RIPPER (Cohen, 1995)

(rule induction algorithm) in WEKA (Hall et al., 2009). As the random forest and RIPPER did not perform as well as GBM on the validation set, we did not conduct further explorations into them or other classification algorithms. If we did find alternative classification algorithms that perform as well as (or if not better than) GBM, we could train a set of base classifiers and combine them with stacking (Wolpert, 1992).

3.3 Result and discussion

3.3.1 Spatial and temporal patterns

In Table 8, we list the top-10 features of each type to show that our features capture some temporal and spatial aspects of clicks for each publisher. Within the one minute interval, fraudulent clicks have significantly more duplicates than normal ones. For repetitive click behavior features, the shorter intervals produce better results after we tested one, five, fifteen, thirty, and sixty minutes intervals using Chao-Shen entropy (Chao and Shen, 2003). Chao-Shen entropy is a non-parametric estimation of Shannon's index of diversity. It combines the Horvitz-Thompson estimator (Horvitz and Thompson, 1952) and the concept of sample coverage proposed by Good (1953) to adjust for unseen observations in a sample. For example, there are multi-feature duplicates such as `avg_spiky_ReAgCnIpCi` (average number of the same `referredurl`, `deviceua`, `usercountry`, `numericip`, and `campaignid` being duplicated in one minute), as well as single feature duplicates such as `std_spiky_numericip` (standard deviation of `numericip` being duplicated in one minute).

For our top click behavior and duplication features, we created conditional features based on finer-grained time intervals to better capture temporal dynamics of click fraud behavior. We divided a day into four six-hour periods: night (12am to 5:59am), morning (6am to 11:59am), afternoon (12pm to 5:59pm), and evening (6pm to 11:59pm). For example, `night_referredurl_percent` is the number of distinct referredurls at night divided by the total number of distinct referredurls, and `night_avg_spiky_referredurl` is the average number of the same referredurl being duplicated within one minute at night. Also, we divided an hour into four fifteen-minute periods: first (0-14), second (15-29), third (30-44), and last (45-59). For example, `second_15_minute_percent` is the number of clicks between 15th to 29th minute divided by total number of clicks.

Fraudulent clicks tend to come from some countries (or finer-grained spatial regions) more than others; for example, businesses in India and Indonesia are hardest hit by fraud (Kroll Advisory Solutions, 2012). Most clicks on mobile advertisements also come from these two countries. We tested the top five, ten, fifteen, twenty, and twenty-five high-risk countries (out of two hundred over countries), and found that the top ten high-risk countries works best. For example, `usercountry_in_percent` and `usercountry_id_percent` are the percentages of invalid clicks originating from India and Indonesia respectively. The main reason for large numbers of invalid clicks coming from `usercountry_sg_percent` or Singapore could be due to BuzzCity's penetration tests being conducted from there.

3.3.2 Performance

Using the GBM configuration in with the 118 features mentioned in section 3.1, our team was ranked fourth with an average precision of **59.38%** on the validation set, as displayed on the public leaderboard. After the competition ended and teams submitted their results

Rank	Feature	Relative influence
6	std_per_hour_density	3.12
12	total_clicks	1.76
14	brand_Generic_percent	1.71
15	avg_distinct_referredurl	1.62
19	std_total_clicks	1.43
23	night_referredurl_percent	1.19
24	second_ 15_minute_percent	1.18
27	distinct_referredurl	1.15
29	std_distinct_referredurl	1.12
30	morning_click_percent	1.1

(a)

Rank	Feature	Relative influence
2	std_spiky_numericip	3.81
3	avg_spiky_ReAgCnIpCi	3.69
4	night_avg_spiky_referredurl	3.66
5	avg_spiky_deviceua	3.29
8	avg_spiky_referredurl	2.67
9	avg_spiky_ReAgCn	2.59
10	night_avg_spiky_ReAgCnIpCi	2.49
11	afternoon_avg_spiky_ReAgCnIpCi	1.98
13	afternoon_avg_spiky_deviceua	1.75
16	std_spiky_referredurl	1.6

(b)

Rank	Feature	Relative influence
1	usercountry_id_percent	5.49
7	usercountry_sg_percent	2.69
49	usercountry_other_percent	0.72
72	usercountry_us_percent	0.44
77	usercountry_th_percent	0.39
84	usercountry_uk_percent	0.34
91	usercountry_in_percent	0.26
103	usercountry_ng_percent	0.05
104	usercountry_tr_percent	0.04
106	usercountry_ru_percent	0.04

(c)

Table 8: Top-10 features by type in final training dataset: (a) By click behavior, (b) By repetitive click behavior, and (c) By high-risk click behavior.

based on the test data, we were ranked the first with the average precision of **51.55%**. (The second winner finished the line with an average precision of 46.42%, and the third winner with 46.15%.) As such, comparing our result with that of the other top teams, we can conclude that our GBM model fits the data well.

3.3.3 Other potentials

In Table 9, we show that there is some potential for an alternative approach using the `channel` of each fraudulent publisher. Fraudulent *mobile* and *adult* content publishers tend

to produce much more invalid clicks than the other fraudulent publishers, especially at night and morning periods. In contrast, fraudulent *entertainment*, *lifestyle*, and *premium portal* publishers produce a lot less invalid clicks, and tend to have relatively more invalid clicks during afternoon and evening periods. We attempted to split the datasets and build models separately by channel, but did not have enough time to integrate/normalize the different sets of prediction scores in a meaningful way.

Channel	Publisher count	Fraud clicks (fraud %)	Night Fraud clicks (fraud %)	Morning Fraud clicks (fraud %)	Afternoon Fraud clicks (fraud %)	Evening Fraud clicks (fraud %)
Adult (ad)	10	**47226 (37%)**	**15435 (12%)**	6439 (5%)	**11299 (9%)**	**14053 (11%)**
Mobile content (mc)	**23**	41941 (33%)	13589 (11%)	**9284 (7%)**	9623 (8%)	9445 (7%)
Community (co)	12	16411 (13%)	7218 (6%)	3301 (3%)	2612 (2%)	3280 (3%)
Entertainment and lifestyle (es)	14	14433 (11%)	2649 (2%)	3265 (3%)	3573 (3%)	4946 (4%)
Search, portal, services (se)	4	3180 (3%)	682 (1%)	572 (0%)	689 (1%)	1568 (1%)
Premium portal (pp)	6	2926 (2%)	351 (0%)	608 (0%)	732 (1%)	904 (1%)
Information (in)	3	893 (1%)	49 (0%)	284 (0%)	428 (0%)	132 (0%)
Total	72	127010 (100%)	**39973 (31%)**	23753 (19%)	28956 (23%)	34328 (27%)

Table 9: High risk categories in final training dataset.

3.3.4 Recommendations

We conclude this section by addressing several key questions in relation to a broader context:

- **What is the underlying click fraud scheme?** Simply put, a relatively large number of clicks or rapid duplicate clicks, or a high percentage of clicks from high-risk countries have been shown to be important fraud indicators.

- **What sort of concealment strategies commonly used by fraudulent parties?** The Tuzhilin Report (Tuzhilin, 2006) on the Google AdWords/AdSense system lists ten possible strategies or sources of invalid clicks. The hard-to-detect click fraud tends to come from organized crime in hard-to-prosecute countries (Chambers, 2012). For example, hard-to-detect and hard-to-prosecute click fraud uses existing user traffic including 0-size iframes, forced searching, and zombie computers.

- **How to interpret data for patterns of dishonest publishers and websites?** From a machine learning point-of-view, some decision tree and rule induction algorithms can provide high interpretability to fraud patterns. However, the key step prior to this is still to engineer the best features using domain knowledge and experimentation, and to allow investigators to discern and validate these fraud patterns from top ranked features, even through black-box classification algorithms.

- **How to build effective fraud prevention/detection plans?** Effective fraud detection plans need to have elements of resilience, adaptivity, and quality data (Phua et al., 2012). Resilience is "defense-in-depth" with multiple, sequential, and independent layers of defense. For example, BuzzCity already has anomaly-based detectors to place some publishers under observation, and they can consider adding classifier-based detectors to their click fraud detection system. In the context of click fraud, the classifier-based detectors need to be adaptive to changing fraud and normal click behavior. The classifier-based detectors also need to use quality data with timely

updates when publishers are discovered to be fraudulent. Other than increasing advertisers' awareness of click/conversion ratios, having better customer service and fraud policies, and improving automated filters, one can pursue click fraud more aggressively, switch from cost-per-click to cost-per-action advertising model, or cultivate trust with advertisers by having independent audits (Jansen, 2007).

4. Second winner's entry

4.1 Preprocessing and feature extraction

We first analyzed each attribute in the raw click and publisher database, and evaluated its effects on the behavior of a publisher. We observed that not all features are useful; attributes from the publisher database such as `address`, `bankaccount` can be excluded from the feature construction process. To facilitate different experiment settings, we considered three datasets. In the first dataset, all publishers labeled as `Observation` were relabeled as `OK`. In the second dataset, all publishers labeled as `Observation` were relabeled as `Fraud`. Finally, the third dataset retains all the three (original) labels. Training and testing were performed accordingly on all the three datasets.

Feature extraction is another important facet for building the prediction model. Properly selected features should be able to capture properties or trends that are specific to fraudulent publishers and robust towards their evolving patterns of behavior. We took each raw attribute in the click database and model the publisher's click pattern by creating several statistical features based on that particular attribute. Details of the feature extraction procedures applied to different attributes are given hereafter.

4.1.1 Attribute: `clicktime`

Fraudulent publishers often disguise their activities using various tricks such as generating very sparse click sequences, changes in IP addresses, issuing clicks from different computers in different countries and so on. Others stick to the conservative approach of generating the maximum number of clicks in a given interval. It is important for any fraud detection system to recognize both kinds of concealment strategies. Accordingly, we derived several statistical features from the `clicktime` attribute in the click database, with the number of clicks for each publisher observed over different time intervals: 1 minute, 5 minutes, 1 hours, 3 hours and 6 hours. The goal is to capture both the short and long term behavior of the publishers, based on the observation that publishers often try to act rationally and have constant clicks in very sparse time intervals.

Specifically, for each time interval, we counted the number of clicks each publisher receives and aggregated these counts using several features: maximum clicks, average click, click skewness, and click variance. Click variance essentially measures the deviation of number of clicks from the average clicks (norm) of a publisher, while click skewness is a measure of the asymmetry of the click distribution. Figure 3 shows all the features we derived from the attribute `clicktime` from the raw click database.

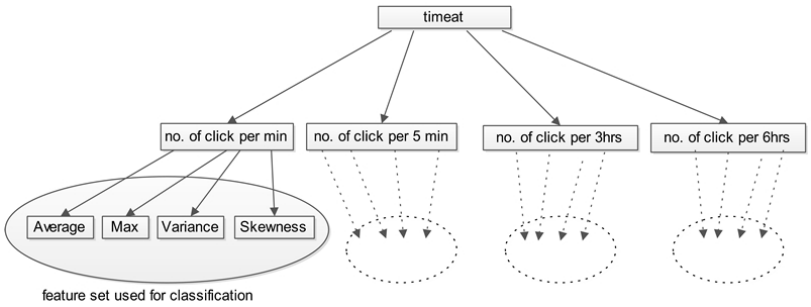


Figure 3: Feature creation from the `clicktime` attribute.

Feature	Description
`MaxSameIPClicks`	Maximum number of clicks from all unique IP addresses associated with a publisher
`NoOfIPs`	Number of clicks from all unique IP addresses associated with a publisher
`ClickOverIPRatio`	Ratio of the number of clicks over the number of unique IP addresses for a publisher
`EntropySameIPClicks`	Entropy of the number of clicks from all IP addresses associated with a publisher
`VarSameIPClicks`	Variance of the number of clicks from all IP addresses associated with a publisher

Table 10: Features derived from the `numericip` attribute.

4.1.2 Attribute: `numericip`

Internet protocol (IP) address is another attribute that can be used to characterize the behavior of a publisher, since it is a reflection of the number of computers/mobile devices used or different times at which the user clicks on a particular advertisement. Since many IP addresses are dynamically allocated when users connects via an internet service provider (ISP), it is not unusual for the same user to have different IP addresses. For a given 3-day period, we observed changes in the IP addresses and number of clicks from a given IP address for a given publisher id. We used parametric measures over IP address attribute (`numericip`) to define the behavior of a publisher. Table 10 lists the feature set created from the `numericip` attribute.

Some fraudulent publishers may try to increase their reward by clicking repeatedly on an advertisement but all of these clicks might come from the same IP. From the data, we observed that many clicks originating from the same IP or an unusually large click to IP ratio tend to be associated with fraudulent behavior, and may place the associated publisher under suspicion. We also observed that lower variance in the number of clicks from each IP is indicative of a legitimate publisher, whereas higher variances might indicate a fraudulent publisher. Similarly, the entropy for the distribution of the number of clicks originating from each IP can be another useful indicator of fraudulent activity.

4.1.3 Attribute: `deviceua`

The `deviceua` attribute is the phone model that the visitors use to browse the web and click on advertisements. As mentioned, a fraudulent visitor might use one phone, but with many dynamically allocated IP addresses. Thus, we use the following measures to derive features from `deviceua` attribute in the set of attributes: `MaxSameAgentClicks`,

`MaxSameAgentClicks`, `VarSameAgentClicks`, and `SkewnessSameAgentClicks`. These features also calculated in a similar way to those for the `numericip` attribute.

4.1.4 Other attributes

We also used the same method to generate features for `country` and `campaignid`. On the other hand, each publisher is assigned to only one channel, thus we avoid taking `channel` to derive more attributes. Instead, we defined the prior probability of being fraud for a given channel based on the training set. That is, we computed number of visitors for each channel, and then the number of fraudulent publishers in that set to obtain the prior probability. Finally, for the `referredurl` attribute we derived `ReferrerOverClickRatio` by computing the number of referred clicks over the total number of clicks for a given publisher.

At the end of feature extraction process, we had 41 different features created from different individual and set of attributes from the dataset. The full list of those 41 features is provided in `www.dnagroup.org/PDF/FDMA12_TeamMasdar_AppendixA.pdf`.

4.2 Method

4.2.1 Classification algorithms

Our approach to detecting fraud consists of employing contemporary classification models over data derived from click database. We tuned the parameters of these models such that they work robustly with the train and validation datasets, and can subsequently generalize to any unseen test set. We tried a range of different model parameters that yielded the highest precision and area under the receiver operating characteristics (AUC) curve, with low standard deviation to ensure performance consistency.

We tried a variety of classification methods including decision tree, neural network, and support vector machine. For each method, we also employed different learning algorithms. Our preliminary experiments on the train and validation sets revealed that the decision tree technique is particularly promising and gave good prediction results. As such, we shall focus on decision tree-based models in the subsequent sections.

4.2.2 Resampling

As mentioned, only a small fraction of publishers are fraudulent, and the skewed nature of the data would drive the prediction model to be more biased towards the majority class. In light of this issue, we used various resampling strategies such as up/downsampling and the synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002). Upsampling was done by replicating samples from the minority class until their number is equal to the that of the majority class. Conversely, downsampling randomly discards the majority class until the class distribution is balanced. In our experiments, we tried both up/downsampling and SMOTE, followed by shuffling of the data instances. Results obtained both with and without the use of sampling methods shall be discussed in Section 4.3.

4.2.3 Ensemble learning

Decision tree-based algorithms are weak learners known for its stability issue. To improve a weak classifier, one may construct many weak classifiers instead of a single one, and to

Base tree	Ensemble (meta) learner
C4.5 tree	Bagging, Metacost, Logitboost, random subspace
REP tree	Bagging, Metacost, Logitboost
Random forest	Bagging, Metacost, Logitboost

Table 11: Decision tree algorithms and the corresponding meta-learning algorithms.

combine them into a powerful decision rule. Recently, a number of combining techniques have been developed, the most popular being bagging (Breiman, 1996), boosting (Freund and Schapire, 1996) and the random subspace method (Ho, 1998). In bagging, one samples the training set, generating random independent bootstrap replicates, constructs the classifier on each of these, and aggregates them by a simple majority vote in the final decision rule. In boosting, classifiers are constructed on weighted versions of the training set, which depend on previous classification results. In the random subspace method, classifiers are constructed in random subspaces of the data feature space. These classifiers are typically combined by simple majority voting.

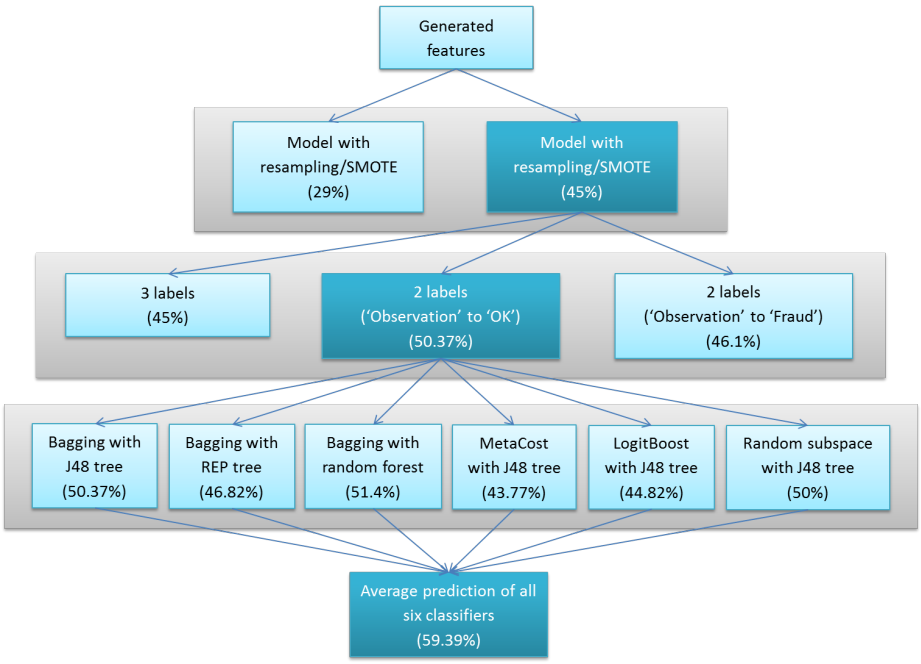


Figure 4: Final classification model comprising ensemble of six learners.

For bagging, we considered the standard algorithm (Breiman, 1996) as well as MetaCost (Domingos, 1999), a special type of bagging that produces a single cost-sensitive classifier of the base learner, giving the benefits of fast classification and interpretable output. For boosting, we considered the LogitBoost method (Friedman et al., 2000), which treats AdaBoost (Freund and Schapire, 1995) as a generalized additive model and applies the cost functional of logistic regression. Finally, we used the standard random subspace method (Ho, 1998), with decision trees as the base learners. We built classification models using

different combinations of base learners and meta-learning algorithm, as shown in Table 11, and evaluated them using the train and validations sets. The base learners are C4.5 decision tree (Quinlan, 1993), reduced error pruning (REP) tree (Su and Zhang, 2006), and random forest (Breiman, 2001). Our final classifier consists of an ensemble of six models, which gave the best overall performance in terms of precision, recall, and area under the ROC curve. The goal of evaluating all three measures is to build a model that can detect high percentage of `Fraud` cases, while maintaining high degree of precision. The final classification model and the corresponding scores of the constituent learners are summarized in Figure 4.

4.3 Results and discussion

4.3.1 Effect of resampling

To see the effect of resampling on the model performance, we conducted experiments with both up/downsampling and SMOTE (Chawla et al., 2002). The resampling and SMOTE performed very well on the training set but performed badly on the validation set. That is, results using the original data were found to be over 20% better than those obtained using resampling and SMOTE.

4.3.2 Two- versus three-class task

Different models were trained using datasets containing only 2 classes (i.e., `OK` and `Fraud`), and all 3 classes (i.e, `OK`, `Observation` and `Fraud`). For the 2-class setting, two approaches were taken: In the first approach, all `Observation` cases were converted to `OK`, and in the second approach all cases of `Observation` were regarded as `Fraud` cases. In the 3-class setting we simply used the labels provided with the data. Of all three approaches, the 2-class dataset gave the best performance. The precision obtained using the J48 tree as the learner and with 2 classes (`Observation` $\rightarrow$ `OK`) was 50.37%, with 2 classes (`Observation` $\rightarrow$ `Fraud`) was 46.1% and with all 3 classes was 45%. Thus, converting all of the `Observation` cases to `OK` was the best approach giving the highest AP score.

4.3.3 Performance

We evaluated the prediction performance of different algorithms for all data sets. Few algorithms which gave best result alone are mentioned above. There were many algorithms with very low true positive and false negative rates, thus giving very high precision scores. These algorithms were able to obtain high precision because of their low false positive value. We were only interested on algorithms which have high true positive rates and precision. The precision scores of the different algorithms when applied on the 2-class validation set were C4.5 tree: 50.37%, REP tree: 46.82%, and LogitBoost: 44.82%, as per Figure 4.

With our ensemble approach, we were able to pass the baseline score but none of the algorithms alone was able to obtain precision higher than 50.37%. We analyzed the results and found that every algorithm has a drawback, which was either a high false positive rate or a lower rate of true positives. This indicated that choosing any one of the algorithms represented a trade-off between high sensitivity on the one hand, and higher precision on the other. Subsequently, we combined the results from the different algorithms trained using the 2-class dataset. Six different algorithms were chosen which obtained higher values for

precision, recall and AUC when evaluated alone. This method proved to be the best as we obtained an average precision of **59.39%** on the validation set. It also performed well on the final test set, achieving a score of **46.42%**.

5. Third winner's entry

5.1 Preprocessing and feature extraction

Click fraud can be generated using a variety of approaches (Dave et al., 2012), such as (1) botnets (where malware on the user's computer clicks on ads in the background), (2) tricking or confusing users into clicking ads (e.g., on parked domains), and (3) directly paying users to click on ads. To deal with various fraud pattern, we first need to extract publisher's feature from various statistics such as mean, standard deviation, count from different views and different time granularity of the publisher. With these features, we can choose the most discriminative ones to build an effective classifier.

5.1.1 Click statistics by publisher

We calculate the basic click statistics of each publisher (i.e., `publisherid`), unique count of attribute such as `numericip`, `country`, `deviceua`, `referredurl`, `campaignid` and total visit. The features used in this work and their descriptions are shown in Table 12:

Feature	Description
unique_count(numericip)	unique count of the IP
unique_count(country)	unique count of the country
unique_count(deviceua)	unique count of the deviceua
unique_count(referredurl)	unique count of the referredurl
unique_count(campaignid)	unique count of the campaignid
total_visit	count of the click log's row

Table 12: Features derived from the click statistics of each publisher.

5.1.2 Click statistics by `numericip`

The fraudulent visitors may visit the advertisements from the same IP address. In order to capture this, we calculate the average access, standard deviation, counting by grouping each IP for each publisher in different time granularity (by second, by min, by day). For example, in the Table 13, suppose it is the full click log for the publisher "$8kxij$", we can get the average access by IP in minute granularity is 5, standard deviation is 0, and the counting for the IP $2,919,155,822$ visit is 5. We can also get the same statistics in different time granularity such as by day, hour, second. The feature created is shown in Table 14.

publisherid	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8kxij	2,919,155,822	Nokia X6	8gava	us	2012-02-09 07:23:13.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gava	us	2012-02-09 07:23:13.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gghw	us	2012-02-09 07:23:13.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gghw	us	2012-02-09 07:23:19.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gava	us	2012-02-09 07:23:19.0	mc	?

Table 13: Example of fraud pattern: Clicking from the same IP address.

Feature	Description
avg_IP_sec	average visit by IP per second
std_IP_sec	standard deviation of average visit by IP per second
count_IP_sec	sum of visit count (larger that 2) by IP per second
avg_IP_min	average visit by IP in minute level
std_IP_min	standard deviation of average visit by IP per minute
count_IP_min	sum of visit count (larger that 2) by IP per minute
avg_IP_hour	average visit by IP per hour
std_IP_hour	standard deviation of average visit by IP per hour
count_IP_hour	sum of visit count (larger that 2) by IP per hour
avg_IP_day	average visit by IP per day
std_IP_day	standard deviation of average visit by IP per day
count_IP_day	sum of visit count (larger that 2) by IP per day

Table 14: Click statistics by IP address.

We also conjecture that fraudulent visitors will visit the advertisements not from the same IP address but from the same subnetwork (see Table 15). For this, we tried to obtain the same statistics by each subnetwork instead of IP. The subnetwork of different granularity can be obtained by dividing the IP by 1,000 or 1,000,000 and rounding the result. Based on this, we got the same statistics as shown in Table 14 for different subnetwork.

publisherid	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8jk0d	1,917,853,114	MSIE_6.0	8gp6q	cn	2012-02-09 11:52:27.0	se	?
8jk0d	1,917,853,057	MSIE_6.0	8gp6q	cn	2012-02-09 11:56:13.0	se	?
8jk0d	1,917,853,952	MSIE_6.0	8gp6q	cn	2012-02-09 11:58:49.0	se	?
8jk0d	1,917,853,022	MSIE_6.0	8gp6q	cn	2012-02-09 12:06:55.0	se	?

Table 15: Example of fraud pattern from the same subnetwork.

5.1.3 Click statistics by `deviceua`

Sometimes the malicious publishers use the same `deviceua` but different IP to access a website at different time periods. For example, in Table 16, the fraudulent visitor using same `deviceua` MSIE 6.0 visits the same publisher using different IP at different times, while there is no other `deviceua` visiting this publisher during these time period. To capture this behavior, we sorted the click by `clicktime` and then `deviceua`. Afterwards, for each click log row, we can compare with the next click log row. If the `deviceua` is the same, we kept the current row, otherwise removed. Thereby, we can calculate the portion of the filtering rows over the total rows for each publisher. We called this feature sequence. This feature is named as *deviceua*1. For example, in the Table 16, after filtering, we have 8 rows left, so the final result will be 8/11. In addition, we tried to sort the click data by deviceua only and calculate the statistics again as discussed above. This feature is named as *deviceua*2.

5.1.4 Click statistics by `campaignid`

The click data also suggest that malicious publishers may access the same advertisement campaign repeatedly using the same IP address and phone agent at a brief time period. This is shown in Table 17, where `campaignid` "8*gkwy*" was accessed many times by the same `numericip` and `deviceua`. These clicks are thus likely to be illegitimate. Accordingly,

publisherid	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8jk0d	1,917,852,952	MSIE_6.0	8gp6q	cn	2012-02-11 02:55:50.0	se	?
8jk0d	1,917,853,022	MSIE_6.0	8gp6q	cn	2012-02-11 02:56:36.0	se	?
8jk0d	1,917,853,060	MSIE_6.0	8gp6q	cn	2012-02-11 03:53:12.0	se	?
8jk0d	1,917,852,993	MSIE_6.0	8gp6q	cn	2012-02-11 04:49:42.0	se	?
8jk0d	701,380,683	Nokia2600c	8k7xb	ng	2012-02-11 04:51:58.0	se	?
8jk0d	1,917,852,993	MSIE_6.0	8gp6q	cn	2012-02-11 05:33:51.0	se	?
8jk0d	1,917,853,114	MSIE_6.0	8gp6q	cn	2012-02-11 06:30:02.0	se	?
8jk0d	1,917,853,146	MSIE_6.0	8gp6q	cn	2012-02-11 07:09:23.0	se	?
8jk0d	1,917,853,146	MSIE_6.0	8gp6q	cn	2012-02-11 07:31:21.0	se	?
8jk0d	1,917,852,993	MSIE_6.0	8gp6q	cn	2012-02-11 07:53:16.0	se	?
8jk0d	1,917,852,952	MSIE_6.0	8gp6q	cn	2012-02-11 07:55:14.0	se	?

Table 16: Example of fraud pattern from one deviceua but different IP at different times.

we calculated the average access, standard deviation, counting by grouping `campaignid` for each publisher in different time granularity (by second, by minute, by day). We then obtained results similar to Table 14, grouped by `campaignid` instead of by `numericip`.

publisherid	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:36:05.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:36:16.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:36:40.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:38:01.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:38:09.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:38:26.0	co	?

Table 17: Example of fraud pattern on the same campaign ID.

5.1.5 Click statistics by `numericip+deviceua`

The `numericip` corresponds to a public IP address that may be assigned to different clickers at different time periods. Hence, an IP address may not uniquely identify a clicker. A better estimate is to use `numericip+deviceua` for identification. Using this identifier, we calculated the average access, standard deviation, counting by grouping `numericip+deviceua` for each publisher in different time resolutions. Again, we obtained the statistics similar to Table 14, grouped by `numericip+deviceua` instead of by `numericip`.

5.2 Method

5.2.1 Overall procedure

We employed a linear blending of many predictive models, and our approach consists of three main steps. In the first step, we created our own validation set from the training set (since the validation set provided by the competition organizer has no label). Specifically, the internal validation set was generated by randomly selecting a subset of publishers in the training set such that the original class (status) distribution is maintained (i.e., stratified sampling). We chose this approach instead of cross-validation procedure for simplicity and computational efficiency. In the second step, we optimized the meta-parameters of the

Type	Method	Description
Single	FT tree (Gama, 2004)	A type of decision trees with logistic regression functions at the inner nodes and/or leaves
	REP tree (Su and Zhang, 2006)	A decision tree is built using information gain or variance and then pruned using reduced-error pruning (REP) method
	Decision table (Kohavi, 1995)	Rule-based classifier that uses simple decision table majority voting, providing a precise yet compact representation
	Bayesian network (Neapolitan, 2003)	Directed graphical model for encoding statistical dependencies among a set of variables
	RPROP (Riedmiller and Braun, 1993)	Learning for feed-forward neural networks that locally adapts the weight updates based on the error function's behavior
Ensemble	LAD tree (Holmes et al., 2002)	A multi-class alternating decision tree that is built using the LogitBoost strategy
	NB tree (Kohavi, 1996)	Decision tree with Naive Bayes classifiers at the leaf nodes
	Random forest (Breiman, 2001)	Combination of tree predictors such that each tree depends on the values of a random vector sampled independently
	Rotation forest (Rodrguez et al., 2006)	Classifier ensemble based on feature extraction. Features are split into subsets and principal component analysis is applied to each
	Tree ensemble (Berthold et al., 2009)	Variant of random forest whereby each tree model is learned on different set of records and/or attributes

Table 18: Classification methods employed for the FDMA 2012 Competition.

models and picked the configuration that yields the best result on the internal validation set. Finally, using the best meta-parameters found, the third step consists of retraining the model on the combined train and internal validation sets. The model was then evaluated on the test set and the prediction scores were recorded for submission to the FDMA website.

5.2.2 Classification algorithms

The classification methods we considered for this competition are listed in Table 18. Most of these techniques are tree/rule-based classifiers (except for Bayesian network and resilient propagation (RPROP)), which we selected due to their relatively good performances. All methods were trained and evaluated using the procedure described in Section 5.2.1.

5.2.3 Blending

To combine the predictions of all the classification algorithms in Table 18, we used a linear blending method. We could simply compute the final prediction as the mean over all predictions in the ensemble, but better results can be achieved by computing weighted sum of the predictions. In this work, we performed blending via weighted sum approach, and the weight coefficients were learned using regularized linear regression. Furthermore, we normalized all inputs (i.e., the predictions) to [0, 1]. We optimized the weight coefficients according to the average precision evaluated on 10-fold cross-validation.

5.3 Results and Discussion

5.3.1 Performance

The performances of the individual models and their blending are shown in table 19. For the single models, RPROP performed the best, suggesting that neural network approach is suitable for the detection task. Among the ensemble approaches, only the tree ensemble outperformed RPROP, but the improvement was marginal. Lastly, the linear blending approach gave better result than all its constituent models. Hence, we chose it as our final model for the competition submissions. We obtained an average precision of **62.21%** and **46.15%** on the validation and test sets, respectively.

Type	Method	Average Precision
Single	FT tree	36.3%
	REP tree	35.8%
	Bayes network	33.7%
	RPROP	48.3%
Ensemble	LAD tree	37.0%
	NB tree	37.9%
	Random forest	47.7%
	Random subspace	38.9%
	Rotation forest	42.9%
	Tree ensemble	49.3%
Ensemble of ensemble	Blending	52.3%

Table 19: Performance for different algorithms on the internal validation set.

5.3.2 Important features

To prune inconsequential features and improve prediction performance, we performed iterative feature elimination (Guyon and Elisseeff, 2003) as follows. Initially, classification is performed using the complete N features. In the next $N-1$ iterations, each of the input features is disabled once. Then the algorithm discards the feature that influences the prediction result the least (in this case giving the smallest degradation in average precision). The subsequent $n-2$ iterations follow where each of the remaining features is ruled out once. The total number of iterations is therefore $N*(N+1)/2-1$. Finally, we had all computed levels of the feature elimination together with the average precision. We specified an error threshold and select the level with fewest features that has a prediction error below the threshold. Table 20 lists the final set of features after the elimination.

6. Runner-up's entry

6.1 Preprocessing and feature extraction

6.1.1 Basic attributes

For each publisher, we considered the following basic attributes: (1) total number of clicks, (2) number of clicks from the same computer (inferred from attribute `numericip`), (3) distinct IP addresses (inferred from `numericip`), (4) distinct parts of the IP addresses, (5) publisher's channel type (inferred from `channel`), (6) phone models used by clickers

Feature	Description
unique_count(referredurl)	Unique count of referredurl
unique_count(campaignid)	Unique count of campaignid
unique_count(country)	Unique count of country
total_visit	Count of the click log's row
count_ip_hour	Sum of visit count (> 2) by numericip per hour
count_ip_ag_sec	Sum of visit count (> 2) by numericip+deviceua per second
count_ip_ag_day	Sum of visit count (> 2) by numericip+deviceua per day
count_sip2_sec	Sum of visit count (> 2) by subnetwork (divided by 1,000,000) per second
count_sip2_min	Sum of visit count (> 2) by subnetwork (divided by 1,000,000) per minute
count_sip2_hour	Sum of visit count (> 2) by subnetwork (divided by 1,000,000) per hour
count_sip_day	Sum of visit count (> 2) by subnetwork (divided by 1,000) per day
avg_sip2_day	Average visit by subnetwork (divided by 1,000,000) per day
avg_ip_ag_min	Average visit by numericip+deviceua per minute
avg_ip_ag_day	Average visit by numericip+deviceua per day
avg_campaignid_min	Average visit by campaignid per minute
deviceua1	Statistics for click data sorted by time and deviceua, as discussed in Section 5.1.3
deviceua2	Statistics for click data sorted by deviceua, as discussed in Section 5.1.3

Table 20: Final feature set after backward elimination.

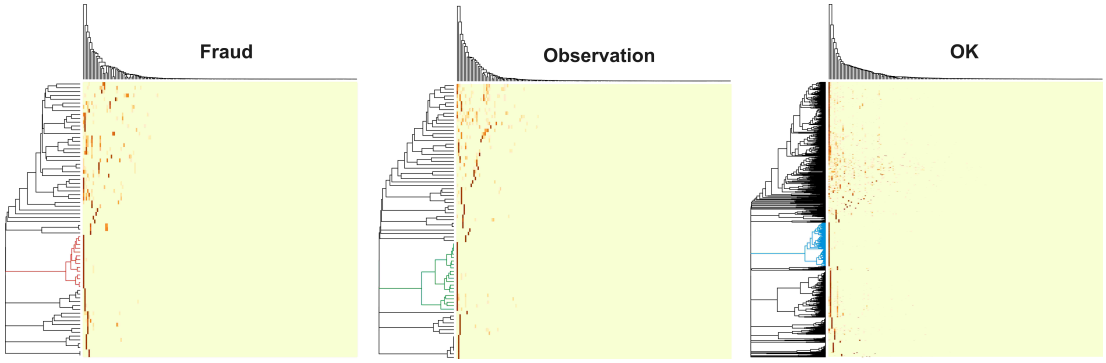


Figure 5: Geo tracking of clicks. Columns show the 197 countries of origin; rows show the publishers from the training set. For each publisher, the click percentage from each country is color-coded. Darker colors reflect higher percentages; yellow is 0% and dark red is 100%. Rows and columns are clustered based on complete linkage. For status `OK`, only those publishers with at least 50 clicks are shown.

(inferred from `deviceua`); (7) advertisement campaign (inferred from `campaignid`); and (8) number of clicks from different countries (*geo tracking*; inferred from `usercountry`).

Click fraud, notably manual click fraud, is known to correlate with the geographical location of the clicker. The heatmaps in Figure 5 visualize the geo tracking of clicks from the training set with respect to the status of the publisher. For all three groups (`Fraud`, `Observation`, and `OK`), the clicks originate from only a few countries (cf. columns). For about half of the publishers in each group (cf. rows), the associated clicks are distributed across these countries, while for the rest the majority of clicks come from only a very small number of countries. In fact, we found such clusters in all three groups.

Geo tracking alone, however, is not reliable for click fraud detection. There may be various reasons that could explain the observed clusters, such as, obviously, the type and the target of the advertisements. Furthermore, malicious scripts could generate fraudulent clicks, and these scripts could run on computers in different geographical locations. In that case, we might fail to detect any clusters.

6.1.2 Click profiles

For each publisher and each unique IP address, we investigated the *click profile*, i.e., the time delay between consecutive clicks. For the majority of fraudulent publishers in the training set, we observed that the number of unique IP addresses was below 3000. Only for two fraudulent publishers, we observed that clicks were coming from more than 3000 unique IP addresses. To derive the click profile, we discarded all publishers for which we observed clicks coming from more than 3000 unique IP addresses. This approach was of course far from being ideal, but it reduced the computational time considerably.

6.1.3 Long click profile

We assumed that many consecutive clicks from the same IP address in short time intervals were suspicious. So for each publisher, we counted how many clicks from the same IP address occurred each day in less than 5s, between 5s and 10s, between 10s and 20s, between 20s and 30s, and so on up to the interval > 300s. Furthermore, we required that at least 10 clicks must have come from each IP address. Table 21 shows an example of a set of consecutive clicks from the same IP address for the publisher 8ih09.

id	deviceua	campaignid	usercountry	date	clicktime	category	referredurl	same URL	gap
14090783	Opera_Mini	8flxe	fr	2012-02-09	08:53:21.0	in	24940f5c4q688oc8	0	0
14096272	Opera_Mini	8flxd	fr	2012-02-09	09:02:01.0	in	24940f5c4q688oc8	1	520
14096576	Opera_Mini	8flyo	fr	2012-02-09	09:02:30.0	in	24940f5c4q688oc8	1	29
14135449	Opera_Mini	8flyo	fr	2012-02-09	09:59:33.0	in	24940f5c4q688oc8	1	3423
14149730	Opera_Mini	8flyo	fr	2012-02-09	10:18:31.0	in	24940f5c4q688oc8	1	1138
14153291	Opera_Mini	8flyp	fr	2012-02-09	10:23:32.0	in	14qhcdsqvou88kos	0	301
14153584	Opera_Mini	8flyu	fr	2012-02-09	10:23:57.0	in	14qhcdsqvou88kos	1	25
14154864	Opera_Mini	8flyp	fr	2012-02-09	10:25:42.0	in	24940f5c4q688oc8	0	105
14197361	Apple_iPhone	8flyo	fr	2012-02-09	11:23:23.0	in	3gza50jfnzcw44wc	0	3461
14197602	Apple_iPhone	8jdc9	fr	2012-02-09	11:23:42.0	in	14qhcdsqvou88kos	0	19
14198413	Apple_iPhone	8flxc	fr	2012-02-09	11:24:50.0	in	14qhcdsqvou88kos	1	68
14198584	Apple_iPhone	8flyp	fr	2012-02-09	11:25:05.0	in	14qhcdsqvou88kos	1	15
14199113	Apple_iPhone	8flys	fr	2012-02-09	11:25:51.0	in	14qhcdsqvou88kos	1	46
14201181	Apple_iPhone	8flxe	fr	2012-02-09	11:28:31.0	in	14qhcdsqvou88kos	1	160
14206726	Apple_iPhone	8flxf	fr	2012-02-09	11:35:50.0	in	23ge85exom8084s0	0	439
14217945	Apple_iPhone	8flyu	fr	2012-02-09	11:50:32.0	in	23ge85exom8084s0	1	882
15754245	Opera_Mini	8gpd5	fr	2012-02-10	10:18:52.0	in	3cu2xmfag82sosk4	0	0
15764598	Opera_Mini	8gpd5	fr	2012-02-10	10:33:56.0	in	4jyefurnmxkwoo4w	0	904
15768527	HTC_Vision	8gdka	fr	2012-02-10	10:39:47.0	in	1rbl5y69ej34gg8w	0	351
15768829	HTC_Vision	8gpd5	fr	2012-02-10	10:40:17.0	in	3cu2xmfag82sosk4	0	30
15777019	Opera_Mini	8gpd5	fr	2012-02-10	10:52:51.0	in	3cu2xmfag82sosk4	1	754
783581	Opera_Mini	8gpd5	fr	2012-02-11	10:34:38.0	in	1rbl5y69ej34gg8w	0	0
901789	SPH-P100	8gpd5	fr	2012-02-11	13:01:12.0	in	1rbl5y69ej34gg8w	1	8794
902642	SPH-P100	8gdka	fr	2012-02-11	13:02:09.0	in	3cu2xmfag82sosk4	0	57
903031	SPH-P100	8gpd5	fr	2012-02-11	13:02:38.0	in	3cu2xmfag82sosk4	1	29

Table 21: Example of 25 consecutive clicks from the same IP for publisher 8ih09 (training set).

For the clicks in Table 21, we see the following interval frequencies: $(0, 0, 2, 4,\ 0, 1, 1, 1,\ 0, 0, 0, 1,\ 0, 0, 0, 0,\ 1, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 0, 0, 11)$, where the i^{th} element represents the count for the i^{th} interval. For example, we observe two clicks with a time gap between

10s and 20s (id 14197602 with 19s and id 14198584 with 15s)[2]. For each publisher, we proceeded analogously for all IP addresses and derived the cumulative interval frequency count, which is the *long click profile*. For the publisher 8ih09, for instance, this profile is $(0, 0, 17, 25,\ 26, 30, 20, 21,\ 19, 14, 7, 5,\ 9, 9, 7, 2,\ 7, 8, 4, 3,\ 7, 3, 3, 2,\ 1, 1, 2, 3,\ 1, 1, 1, 41)$, where the i^{th} element represents the cumulative count for the i^{th} interval. We saw that clicks from the same IP address tend to occur in relatively short sequences for 8ih09, which is in fact labeled as `Fraud`. By contrast, let us consider the profile of a publisher labeled as `OK`, for example, 8i7wi. Its long click profile is $(0, 0, 4, 3,\ 0, 0, 2, 1,\ 0, 1, 1, 0,\ 1, 1, 0, 1,\ 0, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 2, 0, 1,\ 1, 0, 0, 76)$; here, short click sequences are less frequent. Another publisher, 8jkh1, is labeled as `Observation`, and its long click profile is $(6, 13, 11, 19,\ 22, 8, 13, 13,\ 21, 17, 13, 14,\ 11, 7, 12, 9,\ 5, 8, 7, 7,\ 5, 8, 1, 5,\ 6, 5, 2, 1,\ 2, 2, 5, 404)$.

Figure 6(a) shows the click frequencies per interval, derived from all long click profiles per group. We see that, overall, consecutive clicks that follow one another rather quickly occur more often for fraudulent publishers than for those with status `Observation` or `OK`.

6.1.4 Short click profile

The short click profile was derived in the same way as the long click profile, except that at least 5 (and not 10) consecutive clicks must have come from the same IP address. Figure 6(b) shows the click frequencies per interval, derived from all short click profiles per group. Again, we observed that quick consecutive clicks occur more often in fraudulent publishers than in those with status `Observation` or `OK`.

6.1.5 Profile of clicks coming from the same URL

The long and short click profiles ignored the URL where an advertisement had been clicked on. It is possible, however, that a fraudulent (human) clicker does not navigate too often from one web site to another. To derive a pattern of clicks coming from the same URL, we asked for each publisher: how many clicks came from the same IP address *and* the same URL in less than 5s, between 5s and 10s, between 10s and 20s, and so on up to the interval > 300s. At least 5 clicks must have come from each IP address. A problem with this approach, however, was that the URL information was missing for many clickers.

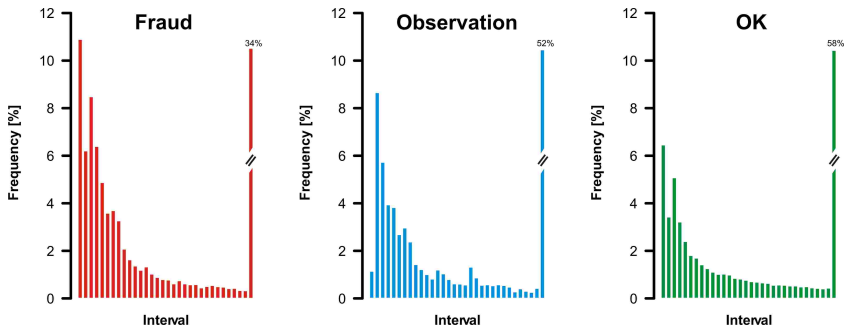
Consider again Table 21. The column **referredurl** contains encrypted information about the URL. The column **same URL** contains a flag, indicating whether the clicker has left ($= 0$) or stayed ($= 1$) on the same URL. We considered only those time gaps that refer to the same URL; thus, we ignored the gap of 19s for **id** 14197602, for example, because the clicker has navigated from 3gza50jfnzcw44wc to 14qhcdsqvou88kos.

Figure 6(c) shows the click frequencies per interval, derived from all click profiles from identical URLs. Similarly, we observed that quick consecutive clicks occur more often in fraudulent publishers than in those with status Observation or OK.

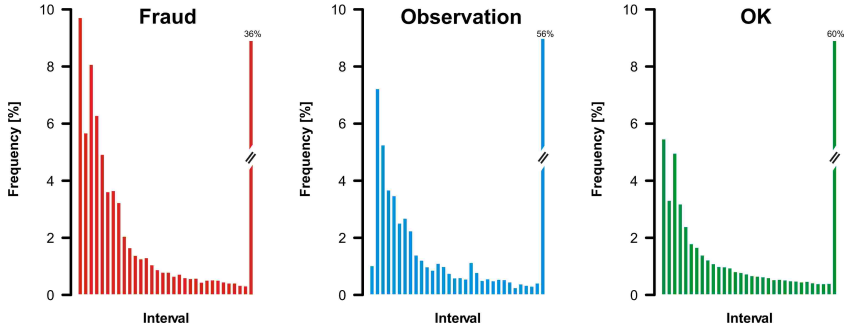
6.1.6 Redflag

For each publisher, we checked if there were at least 5 clicks from the same IP address and the same URL and with a time gap of less than 20s. If so, we incremented a flag (*redflag*)

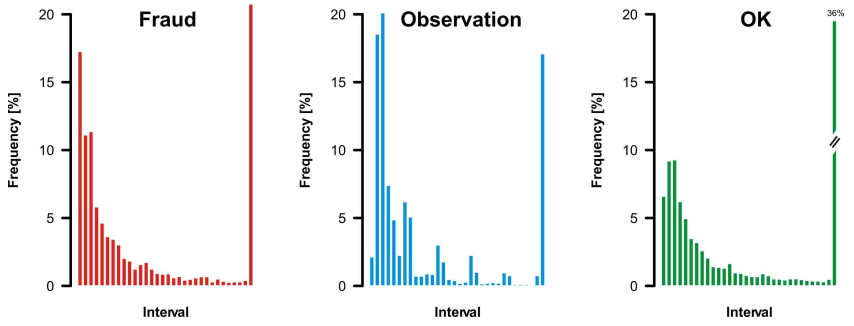
2. We also tracked the relative counts, i.e., the interval counts divided by the respective set size.



(a)



(b)



(c)

Figure 6: Click frequency per interval based on (a) long click profiles, (b) short click profiles, and (c) click profiles from the same URL. The intervals are $[0s, 5s],]5s, 10s],]10s, 20s], ...]300s, +\infty)$.

for that publisher. In Figure 7, publishers from the training set are ranked from left to right based on decreasing values of redflag. Fraudulent publishers (red) and those under observation (blue) are concentrated towards the left hand side. Thus, the larger redflag, the more suspicious is the publisher. Redflag is in fact a significant indicator of fraudulent behavior ($P < 0.001$, Kruskal-Wallis test).

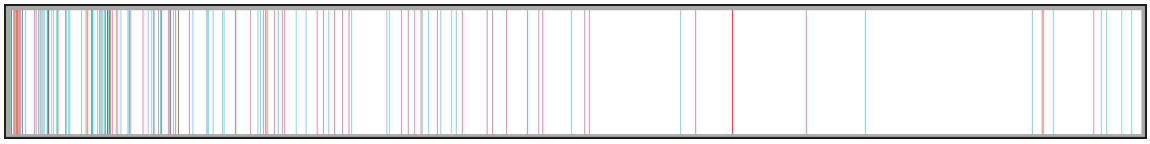


Figure 7: Publishers (from the training set) ranked from left to right based on decreasing values of redflag. Red bars denote publishers with status `Fraud`; blue bars denote publishers with status `Observation`; white bars denote publishers with status `OK`. Fraudulent publishers and those under observation are significantly concentrated towards the left hand side ($P < 0.001$, Kruskal-Wallis test).

6.2 Method

According to the rules of the competition, each team was allowed to submit the predictions of two models for the final evaluation. Below we describe our two models.

For the first model, we used only the basic attributes and the long click profile. The algorithm was random forests (Breiman, 2001), which first generates a number of unpruned decision trees (here, classification and regression trees) from bootstrap samples of the train set. Each tree uses only a random selection of the available attributes. Subsequently, the algorithm combines the trees into one committee ("forest") whose predictions result from aggregating the predictions of the individual trees.

Because of the drastic class imbalance in the train set, either cost-sensitive learning or up/down-sampling is necessary. For random forests, both approaches were shown to be on par in terms of performance (Chen et al., 2004). Up/down-sampling, however, is computationally less expensive because each tree uses only a small subset of the train set. Given the time constraints, we adopted only one approach: up/down-sampling. A multitude of sample sizes were tested, and the models were selected based on the out-of-bag (OOB) error rate. Our preliminary results suggested that the differences between the predictive performance of the models were not so large. From all models, we finally selected seven (Table 22) and combined them into one *ensemble of random forests*. This is model #1, and it was submitted for the final evaluation.

In addition to the reduced data set, we included the following data for our second model: short click profile, click profile from identical URLs, and redflag. The algorithm was random forest with up/down-sampling. We tested again various parameters (number of trees, terminal node size, sampling ratios) and selected the final model on the basis of the OOB error rate. The final model consisted of 50 trees with a terminal node size of 3. The percentages for the bootstrap samples were: 97% for class `Fraud`, 88% for `Observation`, and 61% for `OK`. This is model #2, and it was submitted for the final evaluation.

6.3 Results and Discussion

6.3.1 Evaluation of click profiles

How well can the three click profiles discriminate the publishers? To address this question, we trained three random forests, each using only one of these click profiles and no further

#	ntree	nodesize	Fraud	Observation	OK	OOB error
1	250	5	90%	63%	41%	4.64%
2	250	5	90%	63%	41%	4.67%
3	250	3	83%	75%	51%	4.64%
4	250	3	69%	38%	34%	4.48%
5	250	3	69%	38%	34%	4.74%
6	250	3	90%	44%	51%	4.54%
7	250	4	83%	63%	68%	4.45%

Table 22: Individual random forests with up- and down-sampling (ntree: number of trees in each forest; nodesize: number of terminal nodes in each tree).

data. Each model consisted of 250 trees, each with 3 terminal nodes. The sampling was 90% for `Fraud`, 75% for `Observation`, and 61% for `OK`. Table 23 shows the classification results of these three models (not submitted for the final evaluation).

6.3.2 Performance

Table 24 shows the classification results of the two models that were submitted for evaluation on the final test set. Model #2 (single random forest, 50 trees) achieved a better performance on both the training and the validation set. However, the performance on the final test remarkably deteriorated, compared with the performance of model #1 (ensemble of random forests, 1750 trees) that used only the reduced data set.

Click profile	OOB error	Average precision (validation)
Long	4.61%	47.67%
Short	4.64%	46.77%
Clicks from same URL	4.51%	47.89%

Table 23: Classification results of random forests using only click profiles.

Model #	OOB error	Average precision (validation)	Average precision (test)
1	4.61%	49.99%	42.01%
2	3.66%	51.55%	36.94%

Table 24: Classification results of the final two models.

6.3.3 Remarks

Two problems made this competition particularly challenging. First, there is the problem of concept drift. The train, validation, and test datasets came from different time windows. A publisher may appear across different sets with a similar pattern, but the provided, actual status label may be different. For example, a publisher may have been labeled as `OK` in early February and then as `Observation` in late February, perhaps because this publisher showed a suspicious pattern.

Second, there is no "ground truth" about which publishers are indeed really fraudulent, which are truly `OK`, and which should be under scrutiny. The real status labels were generated by some fraud detection algorithm (here called "ground model", for short), but

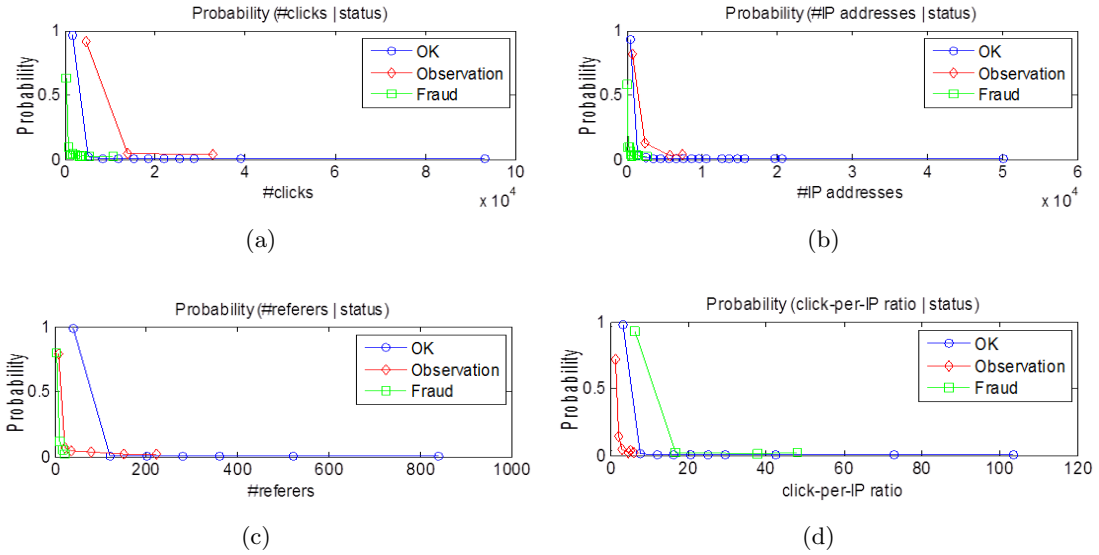


Figure 8: Distribution of the train dataset: (a) Number of clicks, (b) Number of unique visitors (`numericip`), (c) Number of unique referrers (`referredurl`), and (d) Click per visitor ratio (number of clicks divided by number of unique visitors).

how reliable are its predictions? Consider the following example. For one publisher in the train set, we observed 5706 total clicks. All clicks came from the same IP address, and 3307 (58%) occurred in less than 5s. Also, is it not suspicious that 1086 consecutive clicks (19%) have an interval of even 0s? It is tempting to speculate that a script generated these clicks. Surprisingly, the status of that publisher is `OK` in the train set. In the validation set, the click profiles of that publisher may also raise attention: more than 70% of all 1149 clicks – from one and only IP address – occurred in less than 5s, with 248 clicks having a time gap of 0s. Several similar examples of questionable status labels could be found.

7. Organizer's entry

7.1 Preprocessing and feature extraction

We first analyzed the basic statistics of the publishers, derived by grouping the entries in the click database by publisher. For each publisher, we computed the probability distribution (i.e., normalized frequency) of the number of clicks, number of visitors (identified by `numericip`), number of `referredurls`, and the ratio of the number of clicks over the number of visitors. Figure 8 shows the four distributions in the train set respectively, grouped by the publisher's status. Interestingly, we can observe from Figure 8(a) that the `Fraud` publishers have lower click probability than the `OK` publishers. This can be attributed to the fact that BuzzCity blocks the traffic of a publisher as soon as its system deems the publisher as fraudulent. Figure 8(b) shows a similar observation for the distribution of the visitors.

Feature type	#features
Click per visitor ratio	1
Click per ad ratio	1
Click per deviceua ratio	1
Click per country ratio	1
Click per referredurl ratio	1
Category (binary)	10
Nonempty account (binary)	1
Nonempty address (binary)	1

(a)

Feature type	#features
Click fraction from top 20 countries	20
Click fraction from missing country	1
Click fraction from other countries	1
Click fraction from missing referredurl	1
Click fraction from missing deviceua	1

(b)

Feature type	#features
Click series	$6 \times 3 = 18$
Visitor series	$6 \times 3 = 18$
Ad series	$6 \times 3 = 18$
Deviceua series	$6 \times 3 = 18$
Country series	$6 \times 3 = 18$
Referredurl series	$6 \times 3 = 18$
Click per visitor ratio series	$6 \times 3 = 18$
Click per ad ratio series	$6 \times 3 = 18$
Click per deviceua ratio series	$6 \times 3 = 18$
Click per country ratio series	$6 \times 3 = 18$
Click per referredurl ratio series	$6 \times 3 = 18$
Gap interval series	$6 \times 3 = 18$

(c)

Table 25: List of features extracted from the BuzzCity's databases: (a) Basic features, (b) Spatial features, and (c) Time series features.

For referrer (`referredurl`) distribution, we found `Fraud` publishers to be quite different from `OK` ones, and the former have low probability similar to the `Observation` publishers. Further investigation revealed that many `Fraud` publishers have missing/unknown `referredurl` fields. Hence, features derived from `referredurl` can be good indicators for fraudulent acts. Lastly, the distribution of the click per visitor ratio in Figure 8(c) shows that `Fraud` publishers have higher ratio than the other groups, suggesting that the former focus on more efficient use of resources (IP address, in this case) to inflate the click traffic. This motivates us to investigate other ratio-based features (e.g., click per referredurl ratio, click per deviceua ratio, click per country ratio, etc). Note that, if the denominator of the ratio (e.g., number of visitors) is zero, the ratio value will be set to zero.

We extracted several basic ratio features out of the click database, as listed in Table 25(a). Since each publisher is associated with only one of the 10 channel categories (cf. Table 3), we also derived 10 boolean features where only one feature can be set to 1 (true). From the publisher database, on the other hand, we computed two boolean features: whether a publisher has a bank account (nonempty bank account) and whether (s)he has an address (nonempty address). Altogether, we have $5 + 10 + 2 = 17$ basic features. Using these basic features alone, however, the detection results were found to be inadequate. To improve the detection results, we conducted fine-grained analysis on the spatiotemporal aspects of the publishers' click traffic, leading to two new types of features: *spatial* and *time series*. Table 25(b)-(c) respectively list the spatial and time series features we used in this work.

For the *spatial* features, we computed for each publisher the fraction of clicks coming from different `usercountry`, `referredurl` and `deviceua`. For the `usercountry` case, we considered the following top 20 countries in terms of their total number of clicks: {`bd` (Bangladesh), `br` (Brazil), `et` (Ethiopia), `gh` (Ghana), `id` (Indonesia), `in` (India), `ir` (Iran), `ke` (Kenya), `mx` (Mexico), `my` (Malaysia), `ng` (Nigeria), `pk` (Pakistan), `ru` (Russia), `sa` (Saudi Arabia), `th` (Thailand), `tr` (Turkey), `uk` (United Kingdom), `us` (United States), `vn` (Vietnam), and `za` (South Africa). We further computed the click fraction from missing/unknown country, and collate the click fraction from the remaining countries. Finally, we calculated the click fractions from missing `referredurl` and `deviceua`, which according to our preliminary studies correlate with the probability of a publisher being fraudulent.

For the *time series* features, we broke down the 3-day span of each (train, validation, or test) dataset into windows of 1 minute long, and tracked several values of interest (e.g., number of clicks in each minute, number of visitors per minute, duration between etc). This resulted in a time series vector of length 4,320 (i.e., 3 days = 4.320 minutes) for each value type. We also experimented with longer time interval (e.g., 1 hour and 1 day), but the results were worse than the 1 minute interval. Next, we computed several statistical features aggregating the time series values over the 3-day period. That is, we generated 6 statistical features for each time series: *nonzero count*, *mean*, *maximum*, *sum*, *sum of square*, and *standard deviation*. Note that, as most publishers have sparse time series, we filtered out all zero values prior to computing the statistical features. For example, given the series value $v(t)$ at time window $t = \{1, 2, \ldots, 4320\}$, the mean of the series is $\overline{v} = \frac{1}{N}\sum_{v(t)\neq 0} v(t)$, where $N = |\{v(t)|v(t) \neq 0\}|$ is the number of nonzero entries in the series. Our preliminary studies showed that such approach led to better results than including zero values.

To capture trending patterns in the time series, we also derived the same set of statistical features for the *positive* and *negative gradients* of the series. That is, given a count/ratio value $v(t)$ at time window t, the positive gradient $d^+(t)$ and negative gradient $d^-(t)$ are computed using (2) and (3), respectively:

$$d^+(t) = v(t+1) - v(t), \text{if } v(t) < v(t+1) \tag{2}$$

$$d^-(t) = v(t) - v(t+1), \text{if } v(t) > v(t+1) \tag{3}$$

where $t = \{1, 2, \ldots, 4319\}$. As before, we ruled out all zero values when constructing the gradient vectors $d^+(t)$ and $d^-(t)$, and in turn when computing the 6 statistical features summarizing the gradients. Thus, for each time series type, we have $6 \times 3 = 18$ features.

We also included a special time series called gap interval series. The gap interval refers the gap between the timestamps of two consecutive clicks. So if a publisher receives C clicks, then we have a gap interval series of length $C-1$ (in contrast to the other time series that consists of 1-minute windows). We summarizes this series using the same $6 \times 3 = 18$ feature set. In sum, each publisher has 12 types of time series (as per Table 25) and hence a total of $18 \times 12 = 216$ time series features.

Adding all the basic, spatial, and time series features, we have a total of $17+216+24 = 257$ features. Not all these features were useful though, and feature elimination steps shall be carried out to improve the classification results. This is discussed in Section 7.2.2. Finally, after generating all the features, we normalized the feature values to be within $[0, 1]$.

7.2 Method

7.2.1 Classification algorithms

We considered various single and ensemble-typed classification algorithms for the fraud detection task. As our single classifiers, we employed the following algorithms established in machine learning and data mining literature:

- *Logistic regression*: A popular classification method which extends linear regression analysis to model the relationship between a set of predictive variables and a binary outcome variable. It produces an outcome probability between 0 and 1. In this work, we employed the L2-regularized logistic regression implemented in the LIBLINEAR framework (Fan et al., 2008). To cope with the imbalanced class distribution, we adjusted the class weights to be inversely proportional to class frequencies.

- *Support vector machine* (SVM): A state-of-the-art classification method that aims at maximizing the margin of separation between data points from different classes. Intuitively, larger margin implies lower generalization error. We employed the SVM implementation in the LIBSVM framework (Chang and Lin, 2011), with different kernel functions including *linear*, *polynomial*, and *radial basis* kernels. As with logistic regression, we defined class weights to be inversely proportional to class frequencies.

- *k-nearest neighbors* (k-NN): A type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data (Cover, 1967). Classification is computed from a simple majority vote of the nearest neighbors of each point; a query point is assigned the class that has the most representatives within the nearest neighbors of the point.

On the other hand, we employed several decision tree-based ensemble classifiers, which have been widely used in data mining competitions:

- *Random forest*: Each tree in the forest is built from a sample drawn with replacement (i.e., bootstrap sample) from the train set (Breiman, 2001). Moreover, when splitting a node during the tree construction, the split chosen is not the best split among all features, but rather the best among a random subset of the features. In contrast to the original work (Breiman, 2001), which lets each classifier vote for a single class, we combined the tree classifiers by averaging their probabilistic predictions.

- *Gradient tree boosting* (GTB): Generalization of boosting to arbitrary differentiable loss functions (Friedman, 2000). GTB is an accurate off-the-shelf classification method that builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage, regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function.

- *Extremely randomized trees* (Extra tree): In this tree-based ensemble approach, randomness goes one step further in the way splits are computed. Similar to random forest, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate

feature and the best of these randomly-generated thresholds is picked as the splitting rule (Geurts et al., 2006). To arrive at the final decision, we again combined the tree classifiers by averaging their probabilistic predictions.

7.2.2 Feature elimination

As mentioned, the complete 257 features are not all useful, and further improvement can be achieved by removing inconsequential or noisy features. We adopted a feature selection approach (Guyon and Elisseeff, 2003) that is wrapped around our classifiers providing them subsets of features and improving their performance. In particular, we devised a simple wrapper approach which we refer to as *backward feature elimination*, as outlined in Algorithm 1. The algorithm requires (as input) the set of features and their predefined ranking or relative importance. The feature importance can be computed in several ways. In tree-based ensemble methods (e.g., random forest and GTB), features at the top of the tree are used to contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features. In logistic regression or linear SVM, the feature importance can be estimated from the absolute values of its weight coefficients. For other methods that do not naturally give feature ranking (e.g., nonlinear SVM or k-NN), we estimated the feature importance using the F-value of the analysis of variance (ANOVA) test (Box, 1953). The higher the F-value, the more important the feature is.

Algorithm 1 Backward feature elimination.

Require: Features set $\mathbf{F} = \{f_i\}$ and their ranks r_i, minimum number of features m, train data $\mathbf{T}$ and validation data $\mathbf{V}$

$\mathbf{R} \leftarrow \{\}$ and $\mathbf{F}_{best} \leftarrow \mathbf{F}$ // Initialization
$M \leftarrow \text{train}(\mathbf{F}, \mathbf{T})$ // Train M using features $\mathbf{F}$ from $\mathbf{T}$
$AP \leftarrow \text{evaluate}(M, \mathbf{F}, \mathbf{V})$ // Evaluate M using features $\mathbf{F}$ from $\mathbf{V}$
$AP_{best} \leftarrow AP$
$\mathbf{F}_s \leftarrow \text{sort}(\mathbf{F})$ // Sort the features set in ascending order of r_i
for $i = 1$ **to** $|\mathbf{F}_s| - m$ **do**
  $\mathbf{R} \leftarrow \mathbf{R} \cup \{f_i\}$
  $M \leftarrow \text{train}(\mathbf{F}_s \setminus \mathbf{R}, \mathbf{T})$ // Train M using features $\mathbf{F}_s \setminus \mathbf{R}$ from $\mathbf{T}$
  $AP \leftarrow \text{evaluate}(M, \mathbf{F}_s \setminus \mathbf{R}, \mathbf{V})$ // Evaluate M using features $\mathbf{F}_s \setminus \mathbf{R}$ from $\mathbf{V}$
  if $AP_{best} \leq AP$ **then**
    $AP_{best} \leftarrow AP$
    $\mathbf{F}_{best} \leftarrow \mathbf{F}_s \setminus \mathbf{R}$
  end if
end for
return $\mathbf{F}_{best}$

Our feature elimination method starts from the full feature set and eliminates features one by one from the lowest rank. An updated classifier is then generated from the remaining features. Our approach bears some similarities to the recursive feature elimination (Guyon et al., 2002), but instead of re-ranking the features after each elimination step, we still refer to the original (full) feature ranking. We also kept track of all performances obtained by

Type	Method	Best parameter	Average precision	
			Validation	Test
Single	Logistic regression	$C = 512$	41.20%	29.65%
	SVM (Linear)	$C = 32$	29.74%	23.73%
	SVM (Polynomial)	$C = 8192$, degree $= 2$	29.33%	20.02%
	SVM (Radial basis)	$C = 128$	30.37%	23.63%
	k-NN	$k = 5$, weight = "distance"	28.78%	33.46%
Ensemble	Random forest	$ntrees = 100, maxFeat = 6$	57.53%	51.44%
	GTB	$ntrees = 900, maxFeat = 28, depth = 5$	48.78%	49.25%
	Extra trees	$ntrees = 200, maxFeat = 28$	55.36%	54.04%

Table 26: Performances of various classifiers for the validation and test sets. For SVM and logistic regression, C is the penalty parameter controlling the tradeoff between training errors and margin maximization. For the ensemble methods, *ntrees* is the number of tree models in the ensemble, $maxFeat$ is the maximum number of features allowed for each tree, and *depth* is the maximum depth of a tree.

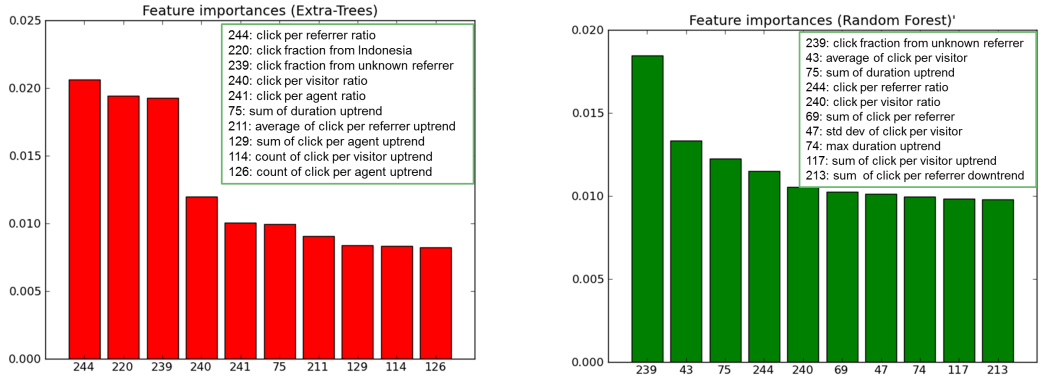
each elimination, rather than stopping the elimination when a performance degradation is detected. The process is repeated until the minimum number of features m is reached. For SVM, logistic regression, and k-NN, we set $m = 1$. For tree-based ensemble methods, we set m as the maximum number of features $maxFeat$ (i.e., $m = maxFeat$), since the largest tree in the ensemble most likely has $maxFeat$ features.

7.3 Results and Discussion

7.3.1 Performance

Table 26 shows the prediction results of all classifiers using the complete features listed in Figure 25. The best configuration for each classifier was determined using *grid search* on the corresponding (meta) parameter space. For logistic regression and SVM, we varied the penalty parameter C within the range $\{2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, \ldots, 2^{15}\}$ and then selected the parameter that gave the best AP score on the validation set. We also experimented with different kernels for the SVM, including linear, polynomial (with degree 2), and radial basis kernels. As for k-NN, we took the parameter k from the range $\{1, 3, 5, ..., 15\}$. We also considered "uniform" and "distance" weight functions for the k-NN predictions, giving equal weight and weight proportional to the inverse of distance to the neighborhood points, respectively. Out of these single classifiers, k-NN with 'distance' weight yielded the best score on the test set, suggesting that the decision boundary for the `Fraud` class is complex. For SVM, employing nonlinear kernel in place of linear kernel did not improve the result.

We further experimented with the tree ensemble methods (i.e., random forest, GTB, and Extra trees). For each method, we selected the best combination of *ntree*, $maxFeat$, and *depth* from the range $\{100, 200, 300, \ldots, 1000\}$, $\{2, 4, 6, \ldots, 30\}$, and $\{1, 2, 3, 4, 5\}$, respectively. Among the three methods, we found that the extremely randomized trees yielded the best AP score int the test set, followed by the random forest which performed best on the validation set. In comparison to the single classifiers, we can see that the ensemble methods produced substantially higher AP scores. A plausible explanation is that ensemble methods try to exploit the local different behavior of the base learners to enhance the



(a) (b)

Figure 9: Feature ranking as obtained by: (a) Extra trees, and (b) Random forest.

accuracy of the overall system. Also, using mixture of base models (instead of choosing just one) can help reduce the risk of (accidentally) selecting a poorly performing classifier, thereby reducing the overall system variance. Yet another key ingredient of effective ensemble system is the diversity of its constituent models. In the case of extremely randomized trees, the randomization of the split thresholds (in addition to the bootstrapping step in random forest) helps promote the diversity, leading not only to faster ensemble construction but also reduced variance of the overall system.

7.3.2 Feature ranking

In Figure 9, we show the top 10 features found by our two best classifiers, Extra trees and random forest, whereby feature importance was estimated from the expected fraction of the samples the tree components contribute to (cf. Section 7.2.2). These features revealed several interesting observations. For instance, the top three features of the extremely randomized trees suggest that `Fraud` publishers tend to have missing/unknown `referredurl`, which would be captured by the high click per referredurl ratio and high click fraction from unknown `referredurl`. Moreover, high click traffic from high-risk countries such as Indonesia can indicate fraudulent behavior (similar to the findings from the first competition winner). We can also observe that the ratio features and their related time series features, particularly the sum, count and uptrend (positive gradient) features, may be indicative of fraudulent behaviors. Similar observations were found by random forest, although the click fraction from high-risk countries were not deemed as important. This, in turn, may explain its inferior performance with respect to Extra trees.

7.3.3 Model simplification

Using the feature elimination procedure outlined in Algorithm 1, we can remove inconsequential features and further improve the detection performance. Table 27 shows the consolidated results of different classifiers after feature elimination. In general, feature elimination improved both performances on the validation and test sets, while simplifying the classification models. We observed a large portion of features being removed by the

		Full features		Reduced features		
Type	Method	Validation AP	Test AP	#features	Validation AP	Test AP
Single	Logistic regression	41.20%	29.65%	46 / 257	46.02%	31.18%
	SVM (Linear)	30.45%	21.89%	38 / 257	36.75%	26.91%
	SVM (Polynomial)	22.69%	16.72%	256 / 257	28.42%	20.23%
	SVM (Radial basis)	34.32%	23.38%	255 / 257	39.10%	23.66%
	k-NN	28.78%	33.46%	257 / 257	28.78%	33.46%
Ensemble	Random forest	57.53%	51.44%	59 / 257	58.84%	52.17%
	GTB	48.78%	49.25%	235 / 257	58.33%	49.90%
	Extra trees	55.36%	54.04%	118 / 257	57.79%	55.64%

Table 27: Performances of single and ensemble classifiers after feature elimination.

logistic regression, linear SVM, random forest, and Extra trees, leading to a fair amount of improvements. By contrast, only small improvements were observed in GTB and nonlinear SVMs, which can be attributed to the marginal feature removals.

8. Conclusion

The results of the FDMA 2012 competition exceeded our expectations in several ways. First, we have high level of participation, although this is our first time organizing such competition. Second, the participants turned in good results quickly, and the performances continually improved toward the end of the competition, showing the interesting potentials of various feature engineering and data mining methods. We conclude this paper by summarizing the solutions proposed by the winning teams, and then providing several important lessons we can learn from the competition results.

8.1 Methods employed

We briefly comment on the methods commonly used by the winning teams as well as the remaining participants, as follows:

- **Preprocessing**: From the competition, we can see that most participants focused on time-series features generated through analyzing the click traffic of a publisher at multiple time resolutions (windows) and taking the statistics across time. Only a few participants utilized spatial features by grouping the click traffic based on `country`, `referredurl`, `channel`, etc. Simple normalization was also often used to improve the performance of the classifiers, such as normalization to $[0, 1]$. Feature transformation methods (e.g., principal component analysis) were rarely used and reported not to bring performance improvements.

- **Feature selection**: Feature selection approaches broadly fall into two types: *filter* and *wrapper* methods (Guyon and Elisseeff, 2003). Filter methods include feature selection algorithms that are independent of any predictors, filtering out features that have little chance to be useful in data analysis. Filters are usually less computationally intensive than wrappers, but they produce a feature set which is not tuned to a specific type of predictive model. On the other hand, algorithms of the wrapper type are wrapped around predictors, providing them subsets of features and receiving

their feedback (usually accuracy). These wrapper approaches are aimed at improving results of the specific predictors they work with. Some participants used feature selection methods and reported that, in general, the wrapper methods performed better than the filter methods.

- **Classification algorithm**: In this competition, ensembles of decision trees were the most widely used approach, providing fairly fast learning and well suited to highly-skewed class distribution, noisy nonlinear patterns, and mixed variable types. More specifically, this success can be attributed to several factors. First, there is a lack of data to properly represent the true distribution, in which case the learning algorithm can find many different hypotheses that all give the same accuracy on the train data. By constructing an ensemble out of these accurate classifiers, the algorithm can "average" their votes and reduce the risk of choosing the wrong classifier. Second, many learning algorithms work by some form of local search that may get stuck in local optima. An ensemble built by running the local search from many different initial conditions may provide better approximation to the true, unknown function than any of the individual classifiers. Lastly, for many datasets, the true function cannot be captured by a single hypothesis. By forming weighted sums of hypotheses in an ensemble, we can expand the space of representable functions.

All in all, ensemble methods combined with wrapper methods proved to be an effective approach for fraud detection. In practice, however, single models would still be preferred, owing to interpretability and tractability reasons. Future endeavors in this enterprise include devising a novel class of single classification algorithms, capable of matching the performances obtained by the top ranking participants.

8.2 What we have learned

The results from the FDMA 2012 Competition offer important insights on fraudulent behavior in online advertising. Below we summarize the key findings of the winning teams:

- **First winner's entry**: From a temporal standpoint, detecting large and/or duplicate clicks and distinguishing between morning and night traffics are important indicators for fraudulent acts. In spatial context, it was shown that high click fractions from top 10 high-risk countries provide strong signals for click fraud. For model selection, it is also important have a model that balances between accuracy and overfitting.

- **Second winner's entry**: Fraudulent partners often try to act rationally resembling legitimate ones. Large variance on `deviceua` suggests that fraudulent partners use many agents to act rationally. An ensemble model that averages the predictions of different algorithms helps to gracefully deal with highly skewed class distribution, and can lead to better performance than that of the individual methods.

- **Third winner's entry**: Analyzing time series at different granularity levels (e.g., sec, min, hour, day) is important, and simple statistics (e.g., average, count) worked well in detecting click fraud. Backward feature elimination can be used to derive the most important fraud indicators. Linear blending of different models were found to give the best performance and improve the performance of the individual models.

- **Runner-up's entry**: Many consecutive clicks from the same IP address in a short-time interval are considered suspicious. The ensemble of random forests was found to be an effective fraud detection method. The problem of concept drift was observed in the competition data, and the "ground-truth" labels in the data may be inaccurate and biased toward BuzzCity's internal detection procedure.

- **Organizer's entry**: Fine-grained analysis of time series at short interval (1 minute) is crucial for deriving informative features for fraudulent publishers. The best model showed that `referredurl`-based features, e.g., click per referredurl ratio and fraction of missing/unknown `referredurl`, provide informative indicators for fraud. Also, the high fraction of clicks from high-risk countries may be used as a signal for click fraud. Lastly, the combination of tree-based ensemble classifiers and backward feature elimination leads to a promising approach to tackle highly-imbalanced data.

8.3 Research outlook

Despite the encouraging results and practical usability of our solutions to fraud detection task, there remains a considerable need for further work on this important topic. An obvious room for improvement is how the current methods can be used to tackle more sophisticated types of click fraud. For instance, fraudsters can work together as a group, allowing them to not only gain more with less (shared) resources, but also reduce the risk of getting detected. The solutions presented in this paper are currently unable to catch such coalition attacks. Several works have been dedicated to identify coalition fraud in online advertising (Metwally et al., 2007; Kim et al., 2011). However, these methods have so far been focused on investigating network topology structure (e.g., a bipartite graph of publishers and site visitors), without probing into the detailed spatial and temporal characteristics of either the individuals or group of individuals involved. Future research is needed to augment extrinsic network/group features with fine-grained analysis of spatio-temporal features.

A related research question concerns the adaptability of the current methods in the face of rapidly-evolving fraudulent behavior and strategies. For instance, several solutions presented in this paper utilize `deviceua` (user agent) as a feature, which may be easily exploited by malicious parties once it is known. Although ensemble approaches and the use of multiple complementary features can address this issue to some extent, there is a need for more general and robust learning methodologies. A plausible approach is to employ *online learning* (Shalev-Shwartz, 2012), which helps deal with concept drift pertaining to the behavioral changes of fraudulent parties, without having to retrain the prediction model from scratch. Another interesting direction to address the issue is to develop a *transfer learning* capability (Pan and Yang, 2010), utilizing information from (multiple) auxiliary domains. Such approach is useful to deal with future data having different feature space and/or different distribution, while minimizing the model re-calibration efforts.

Acknowledgments

We are grateful to BuzzCity Pte. Ltd. for the competition data, prizes, and fruitful discussions. We also thank the organizers of the Asian Conference on Machine Learning (ACML) 2012 for hosting the FDMA 2012 Workshop successfully. This work is supported by the Na-

tional Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media (IDM) Programme Office.

References

M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. KNIME–The Konstanz information miner: Version 2.0 and beyond. *SIGKDD Explorations Newsletter*, 11(1):26–31, 2009.

G. E. P. Box. Non-normality and tests on variances. *Biometrika*, 30(3/4):318–335, 1953.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

C. Chambers. Is click fraud a ticking time bomb under Google? *Forbes Magazine*, 2012. URL `http://www.forbes.com/sites/investor/2012/06/18/is-click-fraud-a-ticking-time-bomb-under-google/`.

C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems Technology*, 2(3):27:1–27:27, 2011.

A. Chao and T. Shen. Nonparametric estimation of shannon's index of diversity when there are unseen species in sample. *Environmental and Ecological Statistics*, 10:429–443, 2003.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

C. Chen, A. Liaw, and L. Breiman. Using random forests to learn imbalanced data. Technical report, Technical Report No. 666, Department of Statistics, University of California, Berkeley, 2004.

W. Cohen. Fast effective rule induction. In *Proceedings of the International Conference on Machine Learning*, pages 115–123, Tahoe City, California, 1995.

T. Cover. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. In *ACM SIGCOMM Computer Communication Review*, volume 42, pages 175–186, Helsinki, Finland, 2012.

P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.

J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

J. Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63 (1):3–42, 2006.

I. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Maching Learning*, 46(1–3):389–422, 2002.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall. Multiclass alternating decision trees. In *Proceedings of the European Conference on Machine Learning*, pages 161–172, Helsinki, Finland, 2002.

D. Horvitz and D. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47:663–685, 1952.

B. Jansen. Click fraud. *IEEE Computer*, 40(7):85–86, 2007.

Chulyun Kim, Hui Miao, and Kyuseok Shim. CATCH: A detecting algorithm for coalition attacks of hit inflation in internet advertising. *Information Systems*, 36(8):1105 – 1123, 2011. ISSN 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2011.04.001. URL `http://www.sciencedirect.com/science/article/pii/S0306437911000524`.

R. Kohavi. The power of decision tables. In *Proceedings of the European Conference on Machine Learning*, pages 174–189, Heraclion, Crete, Greece, 1995.

R. Kohavi. Scaling up the accuracy of naïve Bayes classifiers: A decision-tree hybrid. In *Proceedings of the AAAI International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, OR, 1996.

Kroll Advisory Solutions. Global fraud report, 2012. URL `http://www.krolladvisory.com/insights-reports/global-fraud-reports/`.

Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. DETECTIVES: Detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 241–250, 2007.

R. E. Neapolitan. *Learning Bayesian networks*. Prentice-Hall, Inc., 2003.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

C. Phua, K. Smith-Miles, V. Lee, and R. Gayler. Resilient identity crime detection. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):533–546, 2012.

J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993.

G. Ridgeway. Generalized boosted models: A guide to the gbm package. Technical report, 2007.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

J. J. Rodrguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012. ISSN 1935-8237.

J. Su and H. Zhang. A fast decision tree learning algorithm. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 500–505, Boston, MA, 2006.

A. Tuzhilin. The lane's gifts v. Google report. Technical report, 2006.

D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

M. Zhu. Recall, precision and average precision. Technical Report (Working Paper 2004-09), University of Waterloo, 2004.