Detecting Click Fraud in Online Advertising:

A Data Mining Approach

Soyoung Kim

TABLE OF CONTENTS

Abstract

Due to the surge of smart phone users, click fraud is a growing nuisance for online advertisers who rely on paid search services. Fraud detection system is highly needed to help to identify dishonest publishers and correct Internet advertising market trustworthy. In this paper, based on real-world fraud data from BuzzCity Pte.Ltd., a global mobile advertising company in Singapore, a set of new 12 predictive features are derived from existing features with understanding of click behavior tendencies. Simple statistical analysis based on fine-grained time series is a valuable approach for accurate fraud detection. Ensemble methods are a promising solution to highly imbalanced nonlinear classification tasks with mixed variable types and noisy patterns with high variance.

1. Introduction

We are in an era where it is ever more important to understand data - specifically, patterns of numerical data. We are delving into new approaches for solutions in deeper viewpoints of data compared to traditional statistics. Click fraud is a type of fraud that occurs on the Internet in pay-per-click (PPC) online advertising. This 'bubble clicks' is generated as a charge per click without having actual interest in that link. Thus, it reduces the reliability of online advertising system. It is important for the commissioner of online advertising to proactively prevent click fraud so as to create an accurate census of clicks on links. Accordingly, a reliable click fraud detection system is needed to help to identify dishonest publishers and maintain the credibility of the commissioner. The use of data mining with machine learning methods will help improve the detection accuracy.

In this project, it is found that the most important data aspect from both domain knowledge and experimentation is that fraudulent clicks have particular temporal and spatial characteristics that make them distinguishable from normal clicks. Simple statistical approaches can retain powerful predictive features on time series analysis. MATLAB tool is used for feature engineering & classification (Bagging Decision Tree), and MySQL for storing, understanding, and evaluating statistics from a query.

1.1 Problems

For this project, datasets provided by a global mobile advertising network are used to analyze the behaviour of publishers and identify fraudulent publishers from legitimate publishers. Datasets have two databases in a CSV format. The first dataset contains the publisher information as *publisher ID, account number, address* and *status*, and the second dataset contains click details of all above publishers provided in Table 1, Table 2 respectively. These details include *clicker id,* (user) *publisher id , iplong* (IP address)*, agent* (mobile model)*, category, country, campaign id* and *referrer* (referred URL). In publisher database there are some missing values in *account number, address.*

partnerid	bankAccount	address	status
dv91f		tle0ao6u67qaiwgmek4817o3w	OK
dv8sy	hlshfjmd9ftb7uf7wquuv9r3y	j8hl8uuipl5ku56ere498tcwn	OK
dv8sd		rzqk95gpqy16bebgwo8znpbav	OK
dv3r1		igio2j93cz7di4insa0s1eoyp	Fraud

Table 1. Publisher sample in raw training data

id	iplong	agent	partnerid	cid	cntr	timeat	ct	referer
9794476	1071324855	SonyEricsson_K70	dv3va	dsfag	us	3/8 12:00:00.00	ad	
9794474	1000461055	Samsung_S5233	dv4gs	dswae	in	3/8 12:00:00.00	mg	riflql2a0yv8xoa9sq0recx4x
9794471	3386484265	Nokia_C3-00	duq7h	dr75h	py	3/8 12:00:00.00	co	
9794468	1907981997	Nokia_5233	dv6i3	ds3xq	vn	3/8 12:00:00.00	es	gp53lqr9njqd6z2ap5d364sip
9794467	1791989091	MAUI	duxto	dvb8g	in	3/8 12:00:00.00	ad	

Table 2. Click sample in raw training data

As explained below, since the provided raw data cannot be used directly with any models for classification with sufficient accuracy, data pre-processing is essential for efficient processing and quality end results. Database technology and SQL programming are suggested for this step. To be specific, the MySQL database management system is used to store the original datasets and then developed and used complex SQL queries as pre-processing. This results in extracting and storing the required summary and statistical information used in the actual mining process efficiently without requiring consulting the original datasets. Figures 1 presents the complex SQL queries we wrote and used in the data pre-processing phase. Writing these non-trivial SQL queries took a major time in my project, however, it resulted in increased performance in terms of both time and quality. The output of the queries is used in the Matlab environment for classification.

2. Data pre-processing and feature creation

2.1 Datasets

The experiment was performed using one set of data instead of three original datasets taken from http://palanteer.sis.smu.edu.sg/fdma2012/. There are two databases: *Publisher* file contains the records for each of the 3,081 partners labeled by either fraudulent as Fraud or legitimate as Ok, and *Click* file that contains datasets with three different dates and which include 9 different attributes with 1,173,834 (March 10th), 1,002,223 (March 11th) and 1,223,178 (March 12th) instances, respectively. The first two datasets are used as training and the third one is used as the test data.

2.2 Pre-processing and feature extraction

To create features simply, a relatively large number of clicks or rapid duplicate clicks are put in the scheme of approach. The model of the click pattern for each partner with respect to the attribute by creating parameters based on the particular attribute is considered. Simple statistical approaches have been used to select the affective features to classification such as maximum, average and standard deviation. Table 3 shows the first attempt feature candidates.

No	Attribute	Feature Name	Description
1	TimeAt	avg_click_per_min	Average number of clicks per minute for a given partner
2		avg_click_per_6hrs	Average number of clicks per 6 hours for a given partner
3		max_click_per_min	Maximum number of clicks per minute for a given partner
4		max_click_per_6hrs	Maximum number of clicks per 6 hours for a given partner
5		std_per_min_click	STD number of clicks per minute for a given partner
6		std_per_6hrs_click	STD number of clicks per 6 hours for a given partner
7	Iplong	max_same_IP_count	Maximum number of clicks for each IP address for a given partner
8		nb_of_IPs	The number of unique IP address for a given partner
9		ratio_IP_click	Ratio *nb_of_IPs* to the number of clicks from that partner
10	Agent	max_same_agent_count	Maximum number of clicks for each agent for a given partner
11	Referrer	ratio_refer_click	Ratio the nb of unique referrers for a given partner to the nb of clicks
12	Category	categroy_prior	Probability of being fraud if the click is for that category
13	Country	country_prior	Probability of being fraud if the click is for that country

Table 3. First Attempt of Feature Selection

Details of the feature selection method are as follows.

Attribute: *Timeat*

Fraudulent partners tend to generate sparse click sequences, changes in IP addresses, and clicks form diverse devices in different countries. The attribute is divided 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_avg_min_referrer* is the average number of the same *referrer* being duplicated within one minute at night for a given *partnerid*.

Attribute: *Iplong*

IP address is another attribute that can be used for characteristic behaviour of a partner, since it is a reaction of the number of mobile devices used or different times at which the user clicks on a particular advertisement. Since many IP addresses are dynamically allocated when users connect through ISP, it is not odd for the same user to have different IP addresses.

Many clicks originating from the same IP or an unusually large click to IP ratio can be a sign of fraudulent behaviour and might place the associated partner under suspicion.

Timeat	Publisher Count (Fraud)	Click Count (Fraud)	Percentage
Night	59	24,959	27.70
Morning	66	17.019	18.89
Afternoon	64	21,067	23.38
Evening	66	27,049	30.02

Table 4. Timeat attribute fraudulent clicks on four six-hour time zone

Attribute: *Agent*

The *Agent* attribute is the phone model that user used to browse the web and eventually make clicks on advertisements. As mentioned above, a particular fraudulent user might use one phone, but with many dynamically allocated IP addresses. Hence, as it is shown in Table 1 that *max_same_IP_count*, *nb_of_IPs* and *ratio_IP_click* are used to derive features from *Agent* attribute in the set of attributes. On top of that a new feature *brand_iPhone_percent* is added to a final feature selection. Table 5 shows top 5 high risks agent models in training dataset.

Fraud		Normal	
Model Name	Clicks Count	Model Name	Clicks Count
Apple_iPhone	4242	MAUI	148490
Generic	4080	Nokia_C1-01	33136
Blackberry_9700	3081	Nokia_2700c	29274
MAUI	2740	Nokia_C3-00	29198
Nokia_C3-00	2176	Nokia_5130	28522

Table 5. Top 5 high risks agent models in training dataset

In a similar way, new features are created on Country, Referrer, and Category as well. All statistic tables are provided in Microsoft Excel format (*FraudDetection-stats.xls*).

The final features are shown in Table 6 as below.

No	Feature Name	Description
1	Total_clicks	The number of total clicks for a given partner
2	distinct_iplong	The number of unique IP address for a given partner
3	distinct_referer	The number of unique referrer for a given partner
4	std_per_hrs	STD number of clicks per hour for a given partner
5	std_per_min	STD number of clicks per minute for a given partner
6	std_iplong	STD number of clicks of unique IP address for a given partner
7	avg_min_AgIpCntrRef	Average number of clicks of the same referrer, agent, country, and IP per minute for a given partner
8	night_avg_min_Referrer	Average number of clicks of the same referrer per minute for a given partner at night
9	avg_min_agent	Average number of unique agent for a given partner
10	avg_min_referer	Average number of unique referrer for a given partner
11	avg_min_RefAgCntr	Average number of clicks of same referrer, agent, and country per minute for a given partner
12	night_avg_min_RefAgCntrIp	Average number of clicks of same referrer, agent, country, IP per minute for a given partner at night

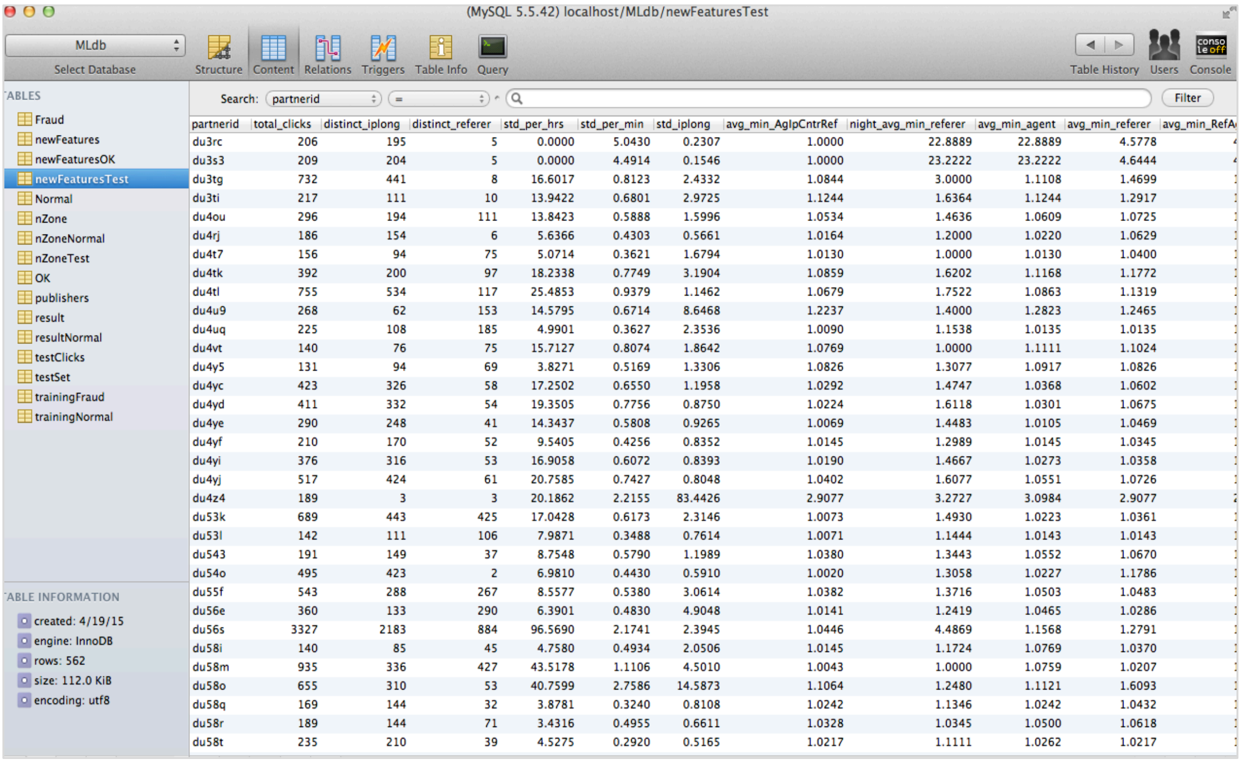
Table 6. List of Final Features

3. Method

Experiments are conducted using Matlab for classification model. Matlab however has a limitation to loading a large dataset. To handle a huge datasets with mixed numerical and categorical datasets, MySQL is used for creating features by querying SQL statement. Hence, only 2,034 rows of table with a new set of 12 features is generated in a CSV format instead of 2 million instances and it is mapped into inputs in Matlab classification model.

```
CREATE TABLE newFeatures AS(
SELECT t1.partnerid, t1.countid, t2.distinct_iplong, t3.distinct_referer, t4.std_per_hrs,
t5.std_per_min, t6.std_iplong, t7.avg_min_AgIpCntrRef,
     t8.night_avg_min_referer, t9.avg_min_agent, t10.avg_min_referer, t11.avg_min_RefAgCntr,
t12.night_avg_min_RefAgCntrIp
FROM(SELECT partnerid, countid FROM testSet) t1,
  (SELECT partnerid, count(x.countid) AS distinct_iplong
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, iplong) x
   GROUP BY partnerid) t2,
  (SELECT partnerid, count(x.countid) AS distinct_referer
     FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, referer) x
   GROUP BY partnerid) t3,
  (SELECT partnerid, std(x.countid) AS std_per_hrs
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, date(timeat),
hour(timeat)) x
   GROUP BY partnerid) t4,
  (SELECT partnerid, std(x.countid) AS std_per_min
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat)) x
   GROUP BY partnerid) t5,
  (SELECT partnerid, std(x.countid) AS std_iplong
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, iplong) x
   GROUP BY partnerid) t6,
  (SELECT partnerid, avg(x.countid) AS avg_min_AgIpCntrRef
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat), referer, agent, cntr, iplong) x
   GROUP BY partnerid) t7,
  (SELECT partnerid, avg(x.countid) AS night_avg_min_referer
   FROM(SELECT partnerid, count(id) AS countid FROM nZoneTest GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat)) x
   GROUP BY partnerid) t8,
  (SELECT partnerid, avg(x.countid) AS avg_min_agent
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat), agent) x
   GROUP BY partnerid) t9,
  (SELECT partnerid, avg(x.countid) AS avg_min_referer
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat), referer) x
   GROUP BY partnerid) t10,
  (SELECT partnerid, avg(x.countid) AS avg_min_RefAgCntr
   FROM(SELECT partnerid, count(id) AS countid FROM testSet GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat), referer, agent, cntr) x
   GROUP BY partnerid) t11,
  (SELECT partnerid, avg(x.countid) AS night_avg_min_RefAgCntrIp
   FROM(SELECT partnerid, count(id) AS countid FROM nZoneTest GROUP BY partnerid, date(timeat),
hour(timeat), minute(timeat), referer, agent, cntr, iplong) x
   GROUP BY partnerid) t12
WHERE t1.partnerid = t2.partnerid AND t1.partnerid = t3.partnerid AND t1.partnerid = t4.partnerid
AND t1.partnerid = t5.partnerid AND t1.partnerid = t6.partnerid AND
    t1.partnerid = t7.partnerid AND t1.partnerid = t8.partnerid AND t1.partnerid = t9.partnerid
AND t1.partnerid = t10.partnerid AND t1.partnerid = t11.partnerid AND
    t1.partnerid = t12.partnerid
GROUP BY partnerid);
```

Figure 1. MySQL query for generating new features



partnerid	total_clicks	distinct_iplong	distinct_referer	std_per_hrs	std_per_min	std_iplong	avg_min_AgIpCntrRef	night_avg_min_referer	avg_min_agent	avg_min_referer
du3rc	206	195	5	0.0000	5.0430	0.2307	1.0000	22.8889	22.8889	4.5778
du3s3	209	204	5	0.0000	4.4914	0.1546	1.0000	23.2222	23.2222	4.6444
du3tg	732	441	8	16.6017	0.8123	2.4332	1.0844	3.0000	1.1108	1.4699
du3ti	217	111	10	13.9422	0.6801	2.9725	1.1244	1.6364	1.1244	1.2917
du4ou	296	194	111	13.8423	0.5888	1.5996	1.0534	1.4636	1.0609	1.0725
du4rj	186	154	6	5.6366	0.4303	0.5661	1.0164	1.2000	1.0220	1.0629
du4t7	156	94	75	5.0714	0.3621	1.6794	1.0130	1.0000	1.0130	1.0400
du4tk	392	200	97	18.2338	0.7749	3.1904	1.0859	1.6202	1.1168	1.1772
du4tl	755	534	117	25.4853	0.9379	1.1462	1.0679	1.7522	1.0863	1.1319
du4u9	268	62	153	14.5795	0.6714	8.6468	1.2237	1.4000	1.2823	1.2465
du4uq	225	108	185	4.9901	0.3627	2.3536	1.0090	1.1538	1.0135	1.0135
du4vt	140	76	75	15.7127	0.8074	1.8642	1.0769	1.0000	1.1111	1.1024
du4y5	131	94	69	3.8271	0.5169	1.3306	1.0826	1.3077	1.0917	1.0826
du4yc	423	326	58	17.2502	0.6550	1.1958	1.0292	1.4747	1.0368	1.0602
du4yd	411	332	54	19.3505	0.7756	0.8750	1.0224	1.6118	1.0301	1.0675
du4ye	290	248	41	14.3437	0.5808	0.9265	1.0069	1.4483	1.0105	1.0469
du4yf	210	170	52	9.5405	0.4256	0.8352	1.0145	1.2989	1.0145	1.0345
du4yi	376	316	53	16.9058	0.6072	0.8393	1.0190	1.4667	1.0273	1.0358
du4yj	517	424	61	20.7585	0.7427	0.8048	1.0402	1.6077	1.0551	1.0726
du4z4	189	3	3	20.1862	2.2155	83.4426	2.9077	3.2727	3.0984	2.9077
du53k	689	443	425	17.0428	0.6173	2.3146	1.0073	1.4930	1.0223	1.0361
du53l	142	111	106	7.9871	0.3488	0.7614	1.0071	1.1444	1.0143	1.0143
du543	191	149	37	8.7548	0.5790	1.1989	1.0380	1.3443	1.0552	1.0670
du54o	495	423	2	6.9810	0.4430	0.5910	1.0020	1.3058	1.0227	1.1786
du55f	543	288	267	8.5577	0.5380	3.0614	1.0382	1.3716	1.0503	1.0483
du56e	360	133	290	6.3901	0.4830	4.9048	1.0141	1.2419	1.0465	1.0286
du56s	3327	2183	884	96.5690	2.1741	2.3945	1.0446	4.4869	1.1568	1.2791
du58i	140	85	45	4.7580	0.4934	2.0506	1.0145	1.1724	1.0769	1.0370
du58m	935	336	427	43.5178	1.1106	4.5010	1.0043	1.0000	1.0759	1.0207
du58o	655	310	53	40.7599	2.7586	14.5873	1.1064	1.2480	1.1121	1.6093
du58q	169	144	32	3.8781	0.3240	0.8108	1.0242	1.1346	1.0242	1.0432
du58r	189	144	71	3.4316	0.4955	0.6611	1.0328	1.0345	1.0500	1.0618
du58t	235	210	39	4.5275	0.2920	0.5165	1.0217	1.1111	1.0262	1.0217

Figure 2. Result of MySQL Query

Figure 2 shows the reduced output from the complex MySQL queries.

partnerid	total_clicks	distinct_iplong	distinct_referer	std_per_hrs	std_per_min	std_iplong	avg_min_AgIpCntrRef	night_avg_min_referer	avg_min_agent	avg_min_referer	avg_min_RfAgCntr	night_avg_min_RefAgCntrIp
du3rj	6	3	5	0.866	0.4	1.41	1	1.5	1.2	1	1	1
du4og	203	23	3	3.613	0	13.5	1	1	1	1	1	1
du4ou	683	471	236	17.18	0.730	2.14	1.054	1.5119	1.07	1.082	1.062	1.0793
du4p9	18	18	7	0.4	0	0	1	1	1	1	1	1
du4pp	3	3	2	0	0	0	1	1	1	1	1	1
du4qr	4	3	3	0.471	0.471	0.47	1.3333	1.5	1.33	1.333	1.333	1.5
du4qs	3	1	1	0.5	0.5	0	1.5	2	1.5	1.5	1.5	2
du4rj	383	317	11	6.316	0.382	0.76	1.0213	1.1694	1.02	1.060	1.026	1.0069
du4rz	6	4	4	0.5	0	0.5	1	1	1	1	1	1

Table 7. 12 predictive features in CSV format from MySQL Query

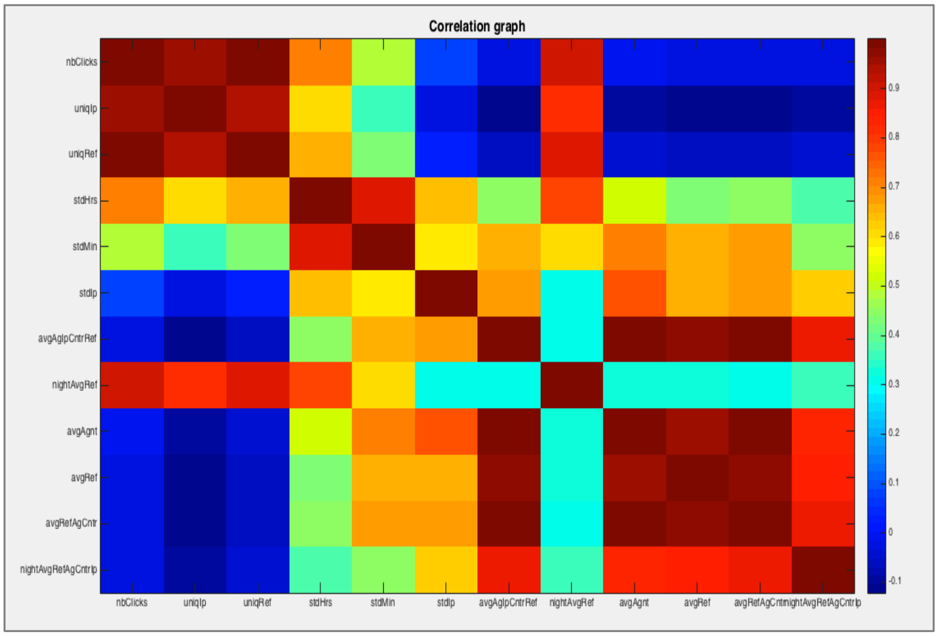


Figure 3. Correlation plot of final click behaviour features in the training set

Figure 3 plots the correlations among a set of new features derived using Matlab.

3.1 Ensemble learning: Bagging decision tree

A decision tree is a tree structure, where the classification process starts from a root node and is split on every subsequent step based on the features and their values. The exact structure of a given decision tree is determined by a tree induction algorithm; there are a number of different induction algorithms which are based on different splitting criteria such as information gain. Ensemble learning method constructs a collection of individual classifiers that are diverse yet accurate. One of the most popular techniques for constructing ensembles is boostrap aggregation called 'bagging'. In bagging, each training set is constructed by forming a bootstrap replicate of the original training set. So this bagging algorithm is promising ensemble learner that improves the results of any decision tree based learning algorithm.

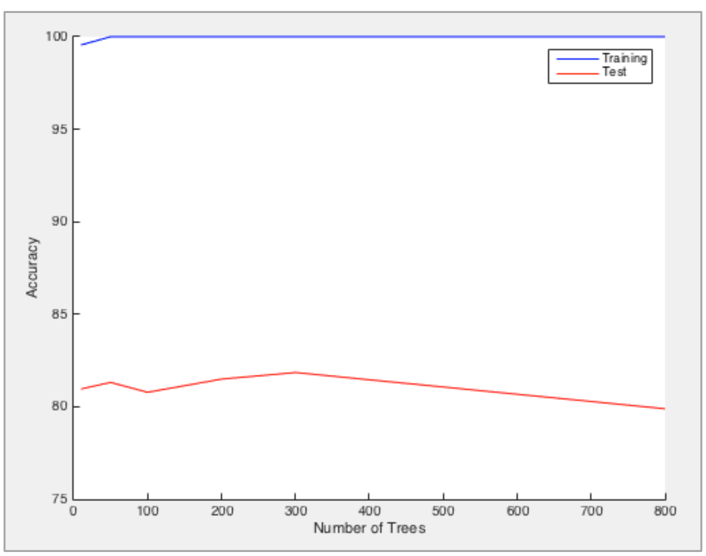
To model the classification algorithm, bagging decision tree is implemented. In Matlab, built-in function, *TreeBagger* generates in-bag samples by oversampling classes with large misclassification costs and undersampling classes with small misclassification costs. Function t*reeBagger(NTrees, X, Y)* creates an ensemble *B* of *NTrees* decision trees for predicting response *Y* as a function of predictors *X*. By default *TreeBagger* builds an ensemble of classification trees. For this project, *B = TreeBagger(300, features, classTR, 'Method', 'classification')* parameters are used; 300 number of trees, 12 columns of numeric features (2034-by-12), one column of labeled training set (2034-by-1), and method is classification.

4. Discussion

MY first attempt for classification model was decision tree since it is non-parametric algorithm meaning that it is easy to interpret and explain how the classifier gets the result. On top of that it does not concern outliers and whether the data is linearly separable. But the problem is that it is easily overfitting and influenced by high variance. Here, ensemble methods such as bagging algorithm (random forests) are recommended to reduce a variance. This justifies the reason for using the bagging decision tree algorithm.

5. Results

For the evaluation of accuracy of the *treeBagger* classifier, a confusion matrix is used. Mainly it provides the accuracy percentage of correctly and incorrectly classified instances.



		Predicted class	
		Normal	Fraud
Actual class	Normal	459	92
	Fraud	7	4

Figure 5. The accuracy of classification of *treeBagger* classifier based on leaf nodes in training and test datasets with confusion matrix on 300 trees (forests)

The graph and confusion matrix in Figure 4 show the accuracy of classification of *treeBagger* in test dataset. Training error and test error generally decrease with increasing leaf nodes. The best leaf node value is 300 with 82.38% of accuracy.

6. Conclusion

In order to pretend that they are valid users and to disguise their illicit activities, fraudulent partners often try to act rationally. Features are analyzed with simple statistics related to fraudulent partners to identify common patterns and to get insights into fraudulent behaviors. With the features derived from *timeat* attribute, 80% of the fraudulent partners have a very small numbers of clicks within the one min, 5-min and even 6-hour intervals. In addition, the variance and standard deviation of legitimate partners per min are very low and this is because most fraudulent partners seem to pretend to be valid publishers using a small numbers of clicks within one minute. Unfortunately, they fail to hide the fact that the sequences of most of clicks are quite systematic, hence, it can be a trigger that these partners as fraudulent. With the attribute *agent*, it is observed that Apple iPhone and Generic models are often used for invalid clicks. Multiple features are combined to get a meaningful analysis and the most affective pattern is the average number of clicks of the same *referrer*, *agent*, *country*, and *iplong* per minute for a given partner.

Compared to FDMA 2012 competition (original paper), due to the size of limited time and use of low memory computer, the experiment was performed using one set of dataset; it results in higher accuracy than FDMA 2012 competitors (Table 8)gm. In the future work, 6 million of original datasets are considered to build model and run my algorithm in the exact same way as FDMA 2012 competition environment.

Affiliation	Validation set	Test set
Institute of Infocomm Research	59.38%	51.55%
Masdar Institute of Science & Technology	59.39%	46.42%
National University of Singapore	62.21%	46.15%
Tokyo Institute of Technology	51.55%	42.01%
Singapore Management University	57.79%	55.64%_

Table 8. Results of the top teams on the validation and test sets in FDMA 2012 competition

References

R. Oentaryo, E.Lim, M.Finegold, D.Lo, F.Zhu, C.Phua, ... and D.Berrar, "Detecting click fraud in online adverting: A data mining approach," in *Journal of Machine Learning Research 15,* 2014, 99-140