Light GBM Machine Learning Algorithm to Online Click Fraud Detection

Article · April 2019

DOI: 10.5171/2019.263928

CITATIONS
67

READS
7,148

2 authors:

Elena Adriana Minastireanu
Universitatea Alexandru Ioan Cuza
6 PUBLICATIONS **96** CITATIONS
SEE PROFILE

Gabriela Mesnita
Universitatea Alexandru Ioan Cuza
27 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Light GBM Machine Learning Algorithm to Online Click Fraud Detection

Journal of Information Assurance & Cybersecurity

Elena-Adriana MINASTIREANU[1] and Gabriela MESNITA[2]

[1]"Alexandru Ioan Cuza" University of Iasi, Doctoral School of Economics and Business Administration, Iasi, Romania

[2]"Alexandru Ioan Cuza" University of Iasi, Faculty of Economics and Business Administration, Business Information Systems Department, Iasi, Romania

Volume 2019, Article ID 263928, Journal of Information Assurance & Cybersecurity, 12 pages, DOI: DOI: 10.5171/2019.263928

Received date : 15 October 2018; Accepted date : 11 January 2019; Published date : 4 April 2019

Academic editor: Maria-Iuliana Dascalu

Cite this Article as: Elena-Adriana MINASTIREANU and Gabriela MESNITA (2019), "Light GBM Machine Learning Algorithm to Online Click Fraud Detection ", Journal of Information Assurance & Cyber security, Vol. 2019 (2019), Article ID 263928, DOI: 10.5171/2019.263928

Abstract

In the current web advertising activities, the fraud increases the number of risks for online marketing, advertising industry and e-business. The click fraud is considered one of the most critical issues in online advertising. Even if the online advertisers make permanent efforts to improve the traffic filtering techniques, they are still looking for the best protection methods to detect click frauds. Hence, an effective fraud detection algorithm is essential for online advertising businesses. The purpose of our paper is to

identify the precision of one of the modern machine learning algorithms in order to detect the click fraud in online environment. In our research, we have studied click patterns over a dataset that handles 200 million clicks over four days. The main goal was to assess the journey of a user's click across their portfolio and flag IP addresses who produce lots of clicks, but never end up in installing apps. As a methodology, we use the experimental test for LightGBM - a Gradient Boosting Decision Tree-type method. This algorithm has enabled an accuracy of 98%. In our research, the literature review was the central source to verify our results.

Keywords: Keywords: online click fraud, machine learning, algorithm classifications, gradient methods

Introduction

Advances in web technologies and data science are the main triggers in transforming online advertising to be the the ideal choice for businesses to effectively target the appropriate marketing segments (Oentaryo, 2014). In the online advertising market, an advertiser plans a budget, then provides the ad network with advertisements and sets commissions for every action that a customer makes, like clicking an ad, making a bid etc. In turn, the ad network makes a contract with a publisher that displays the advertisement on the website and gets commissions based on the traffic it drives to the advertisers. This model may encourage fraudulent publishers to generate malicious clicks on website, either by employing people to click on the advertisement or by using computer scripts that simulate human click behavior (Berrar, 2012). This type of online fraudulent action is called click fraud and represents a major threat to the advertising market that transformed into a multi-billion dollar business during the last decade (Kutylowski and Vaidya, 2014). One in five paid clicks was fraudulent during the month of January 2017, according to paid advertising experts. Another example in this category is the ad fraud botnet "Chameleon" that cost advertisers over 6 million dollars a month (Chameleon Botnet, 2013). Thus, it is very important to develop algorithms that can monitor a publisher's behavior and can reliably predict whether a publisher is likely to be fraudulent (Berrar, 2012).

The structure of the paper is divided as follows: The first part will analyze the context of click fraud and the models developed to detect it in the advertising environment based on literature review. The second part will describe the methodology of the research and the results from the experimental test for LightGBM algorithm.

Background and Literature Review

At its essence, click fraud represents the act of clicking on a search engine sponsored listing or banner ad with the intention of falsely increasing clicks whereas consuming the advertiser's pay-per-click budgets Search advertisers are forced to trust that search engines detect click fraud even though the engines get paid for every undetected fraudulent click (Wilbur and Zhu, 2009). Although it looks a straightforward process, the detection of click fraud is not trivial, one example that can be considered here is the difficulty encountered in the detection of invalid clicks that come from IP addresses that are used by many people or the detection of invalid clicks designed to resemble clicks generated by normal human use.

The literature (Goodman, 2003; Kitts et al., 2014) states that click fraud phenomenon is an adversarial detection problem. Attackers exploit sophisticated methods to cloak their activities including mimicking

human behavior and sometimes hijacking legitimate human traffic. The challenges encountered in the click fraud detection problem can be summarized as:

- throughput requirements
- rapidity of model updates needed to combat attackers,
- low frequency nature of attacks
- user anonymity
- programmability of attacks
- accuracy requirements
- the need to detect and eliminate the effects of fraud within milliseconds.

Click fraud detection methods usually seek to identify anomalies in streams of click data. Different types of anomalies can be caught at different times in the processing of clicks. Some anomalies can be apparent in a single click, and such a click can be marked invalid by an online filter. Other types of anomalies may only become apparent in looking at an aggregate set of clicks over some time period (Daswani et al., 2008).

Click fraud schemes have been continuously evolving in recent years (Vacha et al., 2012; Miller et al., 2011; Alrwais et al., 2012; Li et al., 2012). Existing detection solutions attempt to identify click fraud activities from different perspectives, but each has its own limitations (Beránek et al., 2016). Some of the limitations that can be mentioned here are related to data limitations. If there is too little data available for clicks associated with a particular advertiser or publisher, it may be hard or impossible to make a determination as to whether or not some clicks should be marked invalid. But if there is too much click data associated with an advertiser or publisher, it may be hard to identify potentially fraudulent clicks as they could get “lost in the sea” of valid clicks. Nevertheless, identifying chunks of click data associated with advertisers and publishers that are “just the right size” is an important aspect of click fraud’s problem (Daswani et al., 2008). Google states that: “we use both automated systems and human reviews, analyzing all ad clicks and impressions for any invalid click activity that may artificially drive up an advertiser’s costs or a publisher’s earnings… Our system enables us to filter out most invalid clicks and impressions, and our advertisers are not charged for this activity”.

In literature, Haddadi (2010) proposed exploiting bluff ads to blacklist malicious publishers based on predefined threshold. These bluff ads can be either intended ads with improper display text message, or highly text message with improper intended information. Haddadi’s model works as a check text for legitimacy of individual clicking on ads. Motivated by Haddadi’s model, Dave, Saikat and Yin (2012) proposed an approach for advertisers to measure click ratios on their ads by creating fake ads. However, running bluff ads, despite decreasing click fraud rates, increases advertisers’ budget on advertisements.

Zhang and Yong (2008) presented a method to identify fraud in pay per click model. According to their description for the detection of click fraud, an essential problem is to detect the duplicate clicks in jumping windows and sliding windows. These windows model can be very effective in defining and determining click fraud. Despite having many algorithms which are available to detect duplicates, there is still a need of practical, realistic and effective solutions to detect click fraud in pay per click streams over decaying window architectures. Other results from several research studies (Wilbur and Zhu, 2009; Chen et al.,

2012) suggest that the pay per click industry would benefit from using a neutral third party to audit service provider's click fraud detection algorithms.

Mittal et al. (2006) proposed an advertising network model using online algorithms. This model works on cumulative data to accurately and practically find automated traffic, preserve victim's privacy without changing the business logic model. They proposed an absolute classification of the hit inflation technique and a stream analysis technique that detects a wide range of fraud attacks. They summarized the detection of fraud attacks of some classes as theoretical stream analysis problems that fetch to the data management research community as an open problem.

Goodman (2005) presented a model called pay-per-percentage of impression for mercantilism advertising. Pay-per-percentage of impression deals against each click fraud and impression fraud.

Chertov and Pavlov (2013) proposed a method in which the parameters of the advertising campaign are changed in a dynamic and adaptive way in order to maximize the advertiser's payoff function. Authors utilized game theory concepts for the advertiser's payoff function construction and optimization methods for its extremum searching.

Daniel Berrar (2012) used a committee of random forests with imbalanced bootstrap sampling in order to predict the status of a publisher based on its individual click profile. The average precision was 49.99% on the blinded validation set and 42.01% on the blinded test set.

In the literature, we find out that the use of the modern machine learning algorithm LightGBM was used in a study made by Xiaojun et al. (2018) to predict the default risk of loan projects in P2P (Peer-to-peer) platforms based on the real transaction data of Lending club, which is the largest globally operated P2P platform; and in another study made by Wei Min et al. (2018) that designed an accurate, efficient and scalable online fraud detecting mechanism by delivering a behavior language processing (BLP) framework. Both studies used huge datasets in their experiments and the results demonstrated that LightBGM exhibits powerful advantages like high performance, reliability and availability over traditional logistic regression models. Also, the literature states that gradient boosting decision tree (GBDT) (Friedman, 2001) is a widely-used machine learning algorithm, due to its efficiency, accuracy, and interpretability. GBDT achieves state-of-the-art performances in many machine learning tasks, such as multi-class classification (Li, 2012), click prediction (Richardson et al., 2007), and learning to rank (Burges, 2010).

The literature (Ma et al., 2018; Ke et al., 2017) states that LightGBM achieves algorithm control and optimization through the following main parameters; parameters that we also used in our experiment:

- num_leaves – the number of leaves per tree
- learning_rate – the learning rate of the algorithm
- max_depth – maximum learning depth of the algorithm, when max_depth < 0 there is no limit on the learning depth
- min_data – the minimum number of data in a leaf that can be used to control the fitting phenomenon
- feature_fraction – the proportion of the selected feature to the total of number of features, ranging from 0 to 1. When feature_fraction < 0, the algorithm randomly selects partial features at each

iteration, and feature_fraction is used to control the ratio of the total number of characteristics. This parameter can be used in order to accelerate the training speed and the control of overfitting
- bagging_fraction – the ratio of the selected data to the total data, ranging from 0 to 1. It is similar to the feature_fraction but is randomly and not repeatedly selected and must be greater than 0. This parameter can be used to accelerate the training speed as feature_fraction parameter and the control over the fitting phenomenon.

In the study of Yu Wang (2007) and Xiaojun et al. (2018), we can find some advantages for this algorithm like:

- Fast training speed – LightGBM buckets continuous feature values into discrete bins to accelerate the training procedure
- Low memory consumption – it replaces continuous values using discrete bins to reduce memory usage
- Higher accuracy – it can produce much more complex trees by following a leaf-wise split approach, which is the main reason for achieving higher accuracy
- Good model precision
- Parallel learning support – it supports both feature parallel and data parallel
- GPU support – makes training even faster
- Fast when dealing with big data

Ma et al. (2018) conclude saying that the “reliability and flexibility of LightGBM will greatly promote the development of a credit rating system”.

Research Methodology

In our experiment, we used a public dataset available on Kaggle (a platform for predictive modelling and analytics competitions) that handles 200 million clicks over four days. The dataset contains a total of 203,694,359 rows and 7 columns containing the following features for every click record: IP (ip address of click), app (application id for marketing), OS (os version id of user mobile phone), device (device type id of user mobile phone e.g., iphone 6 plus, iphone 7, huawei mate 7, etc.), channel (channel id of mobile ad publisher), click_time (timestamp of click UTC), click_id (reference for making predictions) and is_attributed as the binary target value; either 0 or 1- not fraudulent. Due to this binary target value, the dataset is highly imbalanced, with only 0.24% having _attributed = 1 – not fraudulent, the rest of 99.66% are 0. The solution to this problem should be in the same time specific and selective, in the sense of avoiding both type I error (false positive) and type II error (false negative). A type I error occurs when the null hypothesis (H0) is true, but is rejected. It is asserting something that is absent; a false hit. A type I error may be likened to a so-called false positive (a result that indicates that a given condition is present when it is actually not present). A type II error occurs when the null hypothesis is false, but erroneously fails to be rejected. It is failing to assert what is present; a miss. A type II error may be compared with a so-called false negative (where an actual ‘hit’ was disregarded by the test and seen as a ‘miss’) in a test checking for a single condition with a definitive result of true or false (2016).

Over this dataset, we used the LightGBM (Light Gradient Boosting Machine) algorithm. The algorithm is a type of GBDT (Gradient Boosting Decision Tree) and is usually used in classification, sorting, regression and supports efficient parallel training. The algorithm uses piece-wise constant trees and approximate loss functions with second-order Taylor approximation at each step. It then trains a decision tree to minimize the second-order approximation, which is analog to Newton's method (Shi et al., 2018).

LightGBM can be divided into three main categories (Ke et al., 2017):

- feature parallelism – which is used concurrently in scenes with many features
- data parallelism – which is applied in scenes with large amounts of data
- Voting parallel – which is applied in situations where there are many features and votes.

Over the dataset, we perform feature analysis to understand more about the data, to spot possible patterns and to decide on possible feature engineering. Whenever we add a new feature in the train dataset, we have to create the same feature in the test dataset. To avoid duplicate coding, we will keep the dataset as initial before we do anything with the features. Thus, from the original features (ip, OS, app, device, channel, click_time), we calculated time (extracted from click_time), count (grouped by multiple features), group by | count unique values for (ip – calculate unique channel; ip, day calculate unique hour etc.), group by (ip, day, channel) and calculate variance for hour etc. Also, we accumulated the values per category – combined (ip, app, device, os) – and calculated, based on click_time, the next_click sequence. The result of this conducted 26 features calculated from which we selected a total of 19 features.

Table 1: Feature engineering – original features

Original	Time	Count	Group by | Count unique	Group by | variance
ip	Hour	(ip, day, hour)	ip | channel (X0)	(ip, day, channel) | hour
os	Day	(ip, app)	(ip, day) | hour (X1)	(ip, app, os) | hour
app		(ip, app, os)	(ip, device, os) | app (X2)	(ip, app, channel) | day
device			ip | app (X3)	**Group by | mean**
channel			(ip, app) | os (X4)	(ip, app, channel) | hour
click_time			ip | device (X5)	**Next Click**
is_attributed			app | channel (X6)	(ip, app, device, os) -> category
			ip | os (X7)	next_click <- f(click_time, category)
			(ip, device, os) | channel (X8)	

Table 2: Feature engineering – selected features

Original	Time	Count	Group by | Count unique	Group by | variance
		(ip, day, hour)	ip | channel (X0)	
os	Day	(ip, app)	(ip, day) | hour (X1)	(ip, app, os) | hour
app		(ip, app, os)		(ip, app, channel) | day
device			ip | app (X3)	**Group by | mean**
channel			(ip, app) | os (X4)	(ip, app, channel) | hour
			ip | device (X5)	**Next Click**
is_attributed			app | channel (X6)	
				next_click <- f(click_time, category)
			(ip, device, os) | channel (X8)	

The strategy behind the feature engineering was to use k-fold cross-validation to select the best parameters values and to validate the features. This provides a method to evaluate the accuracy of a classifier by the division of the data into k numbered equal parts (Dursun et al., 2014). So, this procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. In our experiment k = 5.

We then split the dataset into a train set with 184,903,890 rows containing the main features (IP, app, OS, device, channel, click_time, is_attributed) and a test set with 18,790,469 rows containing the main features and click_id instead of is_attributed. This step is to avoid overfitting. We follow with modelling the datasets with LightGBM algorithm. The algorithm uses the leaf-wise tree growth algorithm (Shi, 2007), whereas other popular models use depth-wise tree growth. Compared with depth-wise growth, the leaf-wise algorithm can converge much faster. However, the leaf-wise growth may be overfitting if not used with the appropriate parameters (“Parameters Tuning”, 2018). In this way, we have to perform a lgb parameter tuning. In the tuning process, we’ve set up the following parameters:

- max_depth – this parameter controls the max depth of the trees. Higher value of max_depth increases model complexity but at the same time can lead to overfitting. Typical values range from 3-10. For this model, we set it to 3 considering that -1 means no limit
- learning_rate – this parameter controls the rate of learning considered in the algorithm. For this model, we set it to 0.20
- num_leaves – this parameter controls the number of leaves. For this model, we set it to 7.
- min_child_samples – this parameter controls the minimum number of data needed in a child. For this model, we set it to 100.
- max_bin – this parameter controls the number of bucketed bin for feature values. For this model, we set it to 100.
- subsample – this parameter controls the subsample ratio of the training instance. For this model, we set it to 0.7

- subsample_freq – this parameter controls the frequency of subsample, subsample <= 0 means no enable. For this model, we set it to 1.
- colsample_bytree – this parameter controls the subsample ratio of columns when constructing each tree. For this model, we set it to 0.9
- scale_pos_weight – this parameter controls the weights of positive class in binary classification task. For this model, we set it to 200 because the training data is extremely unbalanced.
- gamma – this parameter controls the penalty on model complexity. Higher gamma decreases model complexity and decreases the chance of over-fitting. Typical values range from 0-2. For this model, we set it to 0.9.
- min_child_weight – this parameter controls the minimum sum of instance weight of all observations need in a child (leaf). Higher value prevents over-fitting. Typical value ranges from 1-20. For this model, we set it to 0.

After this process, we can train the model until validation scores does not improve for 100 rounds.

The evaluation technique was based on feature importance and ROC curve (Receiver Operating Characteristic curve). The ROC curve is used to evaluate the performance of the system. The closer the curve approaches the top left corner in the plot, the better is the performance of the system, as we can see in a study made by Karasulu (2014).

In LightGBM, there are three ways to evaluate the importance of a feature:

- “gain” measure implies the relative contribution of the corresponding feature to the model calculated by taking each feature’s contribution for each tree in the model. A higher value of this metric when compared to another feature implies it as more important for generating a prediction (Xplorerthik, 2018).

Table 3: Gain results for original features

Feature	Gain
app	0.667
channel	0.206
os	0.022
device	0.004
day	0.003

Table 4: cover results for original features

Feature	Cover
channel	0.398
app	0.169
os	0.125
day	0.037
device	0.024

- “frequency” measure is the percentage representing the relative number of times a particular feature occurs in the trees of the model. In simple words, it tells us how often the feature is used in the model.

Table 5 : Frequency results for original features

Feature	Frequency
channel	0.321
os	0.194
app	0.187
day	0.055
device	0.015

The “gain” measure is the most relevant attribute to interpret the relative importance of each feature. Based on feature importance matrix, we can also choose to eliminate feature with the lowest (near zero) gain and try and test new features in order to improve model performance.

A ROC curve demonstrates several things for 2-class classification algorithms:

- it shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
- the closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
- the closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test (“Plotting and Interpreting an ROC Curve”).

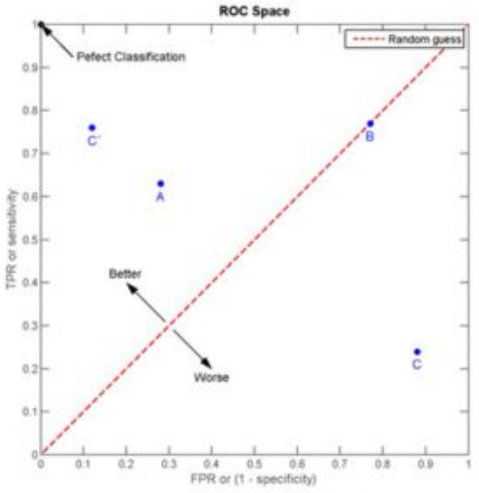

Figure 1: ROC graph

For this experiment, LightGBM presented the following advantages:

- has a built-in dummifier for categorical variables which saves us from engineering them
- uses subsampling of the data: the method used is gradient base one-side sampling (GBOSS)
- it uses histogram-based methods, allowing to reduce the dimensionality for training from n_features x n_rows to n_bins x n_rows which translates into low memory consumption as we've seen in Xiaojun et al. (2018) study and Yu Wang study (2007)
- it has a built-in functionality to ignore sparse inputs which translates to good model precision as we've seen in Xiaojun et al. (2018) study and Yu Wang study (2007)
- compensate for unbalanced data using large values for scale_pos_weight

Results and Discussion

For the experiment, we used a public dataset that handles 200 million clicks over four days. Over this large dataset, we used the LightGBM algorithm. The model used accounts for categorical values and data unbalance. Feature engineering and selection allowed us to improve the detection performance step by step.

The literature confirms that the chosen algorithm can significantly outperform XGBoost (eXtreme Gradient Boosting) and SGB (Stochastic Gradient Boosting) in terms of computational speed and memory consumption (Ke et al., 2017; Chen and Guestrin, 2016; Mitchell and Frank, 2017). Also, in the study made by Ke et al (2017), we find the idea that LightGBM algorithm is the fastest while maintaining almost the same accuracy as baselines.

The barrier in our experiment was the fact that the training used only a part of the train data and 19 features and performance was limited by insufficient resources for training with all the dataset.

As a feature work, we should continue improving performance to deal with large number of features, train more data and make more experiments.

Conclusions and Future Direction

Despite the results and practical usability of our solutions to click fraud detection tasks, there remains a considerable need for further work on this important topic. As far as the research direction is concerned, it is desirable to investigate how the current methods can be used to tackle more sophisticated types of click fraud. For instance, fraudsters can create groups allowing them not only to gain more with fewer resources, but also to reduce the risk of getting detected. Another interesting direction concerns the adaptability of the current method in face of rapidly evolving fraudulent behavior and strategies.

References

1. Alrwais, SA., Dun, CW, Gupta, M., Gerber, A., Spatscheck, O. and Osterweil, E. (2012) 'Dissecting Ghost Clicks: Ad Fraud Via Misdirected Human Clicks', *Proceedings of the Annual Computer Security Applications Conference.* [Online], [Retrieved August 06, 2018], http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.298.5810&rep=rep1&type=pdf

2. Beránek, L., Nýdl, V. and Remeš, R. (2016) 'Click Stream Data Analysis for Online Fraud Detection in E-Commerce', *The International Scientific Conference INPROFORUM.* [Online], [Retrieved August 06, 2018], http://ocs.ef.jcu.cz/index.php/inproforum/INP2016/paper/viewFile/814/577

3. Berrar, D. (2012) 'Random Forests for the Detection of Click Fraud in Online Mobile Advertising', *International Workshop on Fraud Detection in Mobile Advertising (FDMA)*. [Online], [Retrieved August 06, 2018], http://berrar.com/resources/Berrar_FDMA2012.pdf

4. Burges, CJC. (2010) 'From Ranknet to Lambdarank to Lambdamart: An Overview. Learning', *Microsoft Research Technical Report*, [Online], [Retrieved August 07, 2018], http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.634&rep=rep1&type=pdf

5. 'Chameleon Botnet' (2013), [Online], [Retrieved August 07, 2018], http://www.spider.io/blog/2013/03/chameleon-botnet/

6. M., Jacob. VS., Radhakrishnan. S. and Ryu. YU. (2012) 'Can Payment-per-click Induce Improvements in Click Fraud Identification Technologies?', *Information Systems Research*, 26 (4)

7. Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ISBN: 978-1-4503-4232-2, 13 – 17 August 2016, San Francisco, California, 785-794

8. Chertov, O. and Pavlov, D. (2013) 'Adaptive Change of the Web Advertising Campaign Parameters as a Click-Fraud Protection Method', *Procedia – Social and Behavioral Sciences,* 73, 81-84

9. Daswani N., Mysen C., Rao V., Weis S., Gharachorloo K., Ghosemajumder S. and the Google Ad Traffic Quality Team (2008) 'Online Advertising Fraud', *from the forthcoming book, Crimeware edited by Markus Jakobsson and Zulfikar Ramzan*. [Online], [Retrieved August 06, 2018], https://saweis.net/pdfs/crimeware.pdf

10. 'Desktop Click Fraud has risen from 20% to 25% in 2017' (2017), [Online], [Retrieved August 08, 2018], http://blog.pixalate.com/desktop-ad-click-fraud-rising-stats-data-2017

11. Dursun, B., Aydin, F., Zontul, M. and Sener, S. (2014) 'Modeling and Estimating of Load Demand of Electricity Generated from Hydroelectric Power Plants in Turkey using Machine Learning Methods', *Advances in Electrical and Computer Engineering*, 14 (1), 121 – 132

12. Friedman, JH. (2001) 'Greedy Function Approximation: A Gradient Boosting Machine', *Annals of statistics*, 29 (5), 1189–1232

13. Gao, F., Azo, P. and Mahato, R. (2015) 'Click Fraud Detection: Adversarial Pattern Recognition over 5 Years at Microsoft", Microsoft Corporation', One Microsoft Way, Redmond, WA. USA

14. Goodman, J. (2003) 'Spam Filtering: Text Classification with an Adversary', Invited Talk at KDD 2003 Workshop on Operational Text Classification Systems

15. Haddadi, H. (2010) 'Fighting Online Click-Fraud Using Bluff Ads', *ACM SIGCOMM Computer Communication Review*, 40 (2), 21 – 25

16. Joshua, G. (2005) 'Pay-per-percentage of impressions: an advertising method that is highly robust to fraud. Workshop on Sponsored Search Auctions', Workshop on Sponsored Search Auctions

17. Karasul, B. (2014) 'An Automatic Optic Disk Detection and Segmentation System using Multi-level Thresholding', *Advances in Electrical and Computer Engineering*, 14 (2), 161 – 172

18. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T-Y. (2017) 'LightGBM: A Highly Efficient Gradient Boosting Decision Tree', *Advances in Neural Information Processing Systems,* 30

19. Kitts, B., Zhang, JY., Wu, G., Brandi, W., Beasley, J., Morrill, K., Ettedgui, J., Siddhartha, S., Yuan, H.,

20. Kutylowski, M. and Vaidya, J. (2014) Computer Security – ESORICS 2014, Part II, LNCS 8713, Springer International Publishing Switzerland, 419–438

21. Li, P. (2012) 'Robust Logitboost and Adaptive Base Class (Abc) Logitboost', *arXiv preprint arXiv*
22. Li, Z., Zhang, K., Xie, Y., Yu, F. and Wang, XF. (2012) 'Knowing your enemy: Understanding and Detecting Malicious Web Advertising', *Proceedings of the ACM Conference on Computer and Communications Security*. [Online], [Retrieved August 10, 2018], https://pdfs.semanticscholar.org/0bec/9e9fee4bee287ed2ea1ef9059b573fb0b711.pdf

23. Linfeng, Z., and Guan, Y. (2008) 'Detecting Click Fraud in Pay-Per-Click Streams Of Online Advertising Networks', Proceedings of 28th International Conference Distributed Computing Systems, ISBN: 978-0-7695-3172-4, 17-20 June 2008, Beijing, China, 77-84

24. Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q. and Niu, X. (2018) 'Study on A Prediction of P2P Network Loan Default Based on the Machine Learning LightGBM and XGboost Algorithms according to Different High Dimensional Data Cleaning', *Electronic Commerce Research and Applications*, 31, 24 – 39

25. Miller, B., Pearce, P., Grier, C., Kreibich, C. and Paxson, V. (2011) "What's clicking what? Techniques and innovations of today's clickbots", *Computer Science Division and International Computer Science Institute*, 6739, pp. 164–183

26. Min, W., Tang, Z., Zhu, M., Dai, Y., Wei, Y. amd Zhang, R. (2018) 'Behavior Language Processing with Graph based Feature Generation for Fraud Detection in Online Lending', Proceedings of WSDM workshop on Misinformation and Misbehavior Mining on the Web, [Online], [Retrieved August 12, 2018], http://snap.stanford.edu/mis2/files/MIS2_paper_26.pdf

27. Mitchell, R. and Frank, E. (2017) 'Accelerating the Xgboost Algorithm Using Gpu Computing', *PeerJ Preprints*, [Online], [Retrieved August 10, 2018], https://peerj.com/articles/cs-127/

28. 'Parameters Tuning' (2018), [Online], [Retrieved August 10, 2018], https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

29. 'Plotting and Intrepretating an ROC Curve', [Online], [Retrieved August 10, 2018], http://gim.unmc.edu/dxtests/roc2.htm

30. Richardson, M., Dominowska, E., and Ragno, R. (2007) 'Predicting Clicks: Estimating the Click-Through Rate for New Ads', *Proceedings of the 16th international conference on World Wide Web*,

521–530. Oentaryo, R. (2014) ‘Detecting Click Fraud in Online Advertising: A Data Mining Approach’, *Journal of Machine Learning Research*, 15, 99-140

31. Sanjay, M., Gupta, R., Mohania, M., Gupta, SK., Iwaihara, M. and Dillon, T. (2006) ‘Detecting Frauds in Online Advertising Systems’, *Proceedings of 27th International Conference on Electronic Commerce and Web Technologies*, 4082, 222-231

32. Shi, H. (2007) ‘Best-first decision tree learning’, PhD thesis, The University of Waikato

33. Shi, Y., Li, J. and Li, Z. (2018) ‘Gradient Boosting with Piece-Wise Linear Regression Trees’, [Online], [Retrieved August 10, 2018], https://www.researchgate.net/publication/323217210_Gradient_Boosting_With_Piece-Wise_Linear_Regression_Trees

34. ‘Type I and type II errors’ (2016) [Online], [Retrieved August 10, 2018], https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Type_I_error_rate.htm

35. Vacha, D., Guha, S. and Zhang, Y. (2012) ‘Measuring and fingerprinting click-spam in ad networks’, *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication*. [Online], [Retrieved August 10, 2018], http://www.cs.utexas.edu/~yzhang/talks/clickspam-sigc12.pdf

36. Wilbur, KC. and Zhu, Y. (2009) ‘Click Fraud’, *Marketing Science*, 28 (2), 293-308

37. ‘Xplorerthik’ (2018), [Online], [Retrieved August 10, 2018], https://www.cnblogs.com/xinping-study/p/8780817.html