A Solution for Class Imbalance Problem in Ad-Click Fraud Detection using Ensemble Learning Model

Le Ngoc Anh,[1,2] Phan Anh Khoi,[1,3] Duong Nhat Thanh,[1,4] Tran Phuong Anh,[1,5] Nguyen Thuy Linh[1,6]

[1] *DSEB 63, Faculty of Mathematics Economics, National Economics University, 207 Giai Phong str., Hai Ba Trung dist., Ha Noi, Vietnam*
[2] *Student ID: 11219261*
[3] *Student ID: 11212902*
[4] *Student ID: 11219287*
[5] *Student ID: 11219258*
[6] *Student ID: 11219276*

Ha Noi, May 2024

Abstract
Click fraud is a type of fraud that occurs on the Internet in pay per click (PPC) online advertising. In this type of advertising, the owners of websites that post the ads (publishers) are paid based on how many site visitors click on the ads, creating intuitive for publishers to make money illegally by cheating massive number of clicks. Ad click fraud detection helps save billions of dollars each year on marketing campaigns and makes the digital marketing market more transparent. A common problem with ad click fraud detection models is class imbalance in data, when the majority of users, including regular users, click on the ad without downloading the product. Previous studies have shown ensemble of classifiers to be more effective than traditional individual Machine Learning models to solve this imbalanced data problem, but no studies have compared the effectiveness of different ensemble learning models in this context. In this study we compared the performance of some state-of-the-art ensemble models such as XGBoost, Light GBM, CatBoost, as well as combined them with data level and algorithm level approaches, to overcome the problem of data imbalance. We achieved best performance of 0.929 of F1-score and 0.976 of AU ROC by using SMOTE, Linear Disriminant Analysis (LDA), Optuna tuning and CatBoost, and propose the combination of SMOTE, LDA, hyperparameter tuning with CatBoost or LightGBM for this problem. In this study, we demonstrate that the integration of feature selection, oversampling, hyperparameter tuning and ensemble of classifiers can overcome the challenge of class imbalance in a click dataset, helps detect lists of potentially fraudulent IP addresses and take necessary prevention measures at an early stage.

Key words: Click Fraud, Class imbalance, Ensemble learning, SMOTE, Linear Discriminant Analysis

Introduction

Brand advertising and performance advertising are the two biggest types of digital ads (Neeraja et al. 2023). In brand advertising, the focus is on building the image and brand awareness by repeatedly conveying messages through various media channels such as TV, print, and more recently, platforms like TikTok and Facebook. It involves reaching out to customers by making them passively accept the message. On the other hand, performance advertising centers around evaluating and measuring the effectiveness of advertisements through metrics like video views and clicks from regular users. This method assesses the success of advertising campaigns based on specific data and measures results accurately.

With the explosion of the Internet and mobile apps, pay-per-click (PPC) models are becoming increasingly popular, emerging as the predominant model for performance advertising. PPC operates as follows: advertisers (those being advertised) pay ad-networks (platforms acting as intermediaries between advertisers and publishers, which often internet giants like Google, Microsoft, Yahoo,...) to find partners willing to display their ads. Publishers (owners of websites, mobile apps, etc.) receive requests from ad-networks and display ads on these platforms to reach users. Ad-networks track the number of ad clicks and then pay publishers, who, in turn, are paid by advertisers based on user engagement. Due to the nature of the model in which is the publisher is paid by the number of clicks performed, there are many ways to intentionally perform a mass number of clicks for nefarious purposes. This has led to various forms of fraud by publishers to incentivize, deceive, or attack devices to make users click on ads and increase the earnings of publishers. Some websites appeal to users' sympathy through pleas such as "click on the ads to help the website stay active," some mobile game apps reward users after watching a 30-second ad, while some links automatically redirect to ads on other tabs

when clicked, entirely against the user's intent. Types of click fraud as follow: Brute Force Attack: A straightforward method of performing click fraud involves using a single computing device (such as a computer, laptop, or mobile). In this attack, a person repeatedly taps on an advertisement. Crowdsourcing Attacks: Publishers deploy ads in their apps, often appealing to users' sense of compassion or support for a cause. This encourages the audience to click on the ads. Reward-Based Tactics: Taking click fraud to the next level, some fraudsters reward individuals for clicking on ads. These rewards may come in the form of discount coupon codes or bonus points for playing games. Click Farm Attack: Publishers influence a group of people to generate click traffic on their ads in exchange for small profits. Hit Inflation Attacks: In these attacks, legitimate user click traffic is redirected to an ad without their knowledge. Botnet Attacks: Professional attackers use malware to infect a large number of computing devices. Once infected, the attacker can instruct the botnet to perform repetitive clicks by installing apps in the background, all without the device owner's awareness (Thejas et al. 2019).

Alongside the growth of digital advertising, the prevalence of click fraud poses significant challenges for advertisers, publishers, and the transparency of online marketing platforms. Click fraud refers to generating fraudulent clicks on online ads to increase click-through rates or deplete an advertiser's budget without genuine user interest. This unethical behavior not only diminishes the effectiveness of online advertising campaigns but also causes substantial financial damage, estimated in the billions of dollars annually. Some instances of ad-click fraud are directly linked to cybercrime, such as botnets, a sophisticated cybercrime tactic that covertly runs programs on the devices of multiple users to generate fake clicks. A study showed that one of the largest click fraud botnets, called ZeroAccess, induces advertising losses on the order of $100,000 per day (Liu et al. 2014).

Proving click fraud can be very difficult since it is hard to know who is behind a computer and what their intentions are. The Tuzhilin Report (Tuzhilin 2006) produced by Alexander Tuzhilin as part of a click fraud lawsuit settlement defines "the Fundamental Problem of invalid (fraudulent) clicks": "There is no conceptual definition of invalid clicks that can be operationalized [except for certain obviously clear cases]... An operational definition cannot be fully disclosed to the general public because of the concerns that unethical users will take advantage of it, which may lead to a massive click fraud. However, if it is not disclosed, advertisers cannot verify or even dispute why they have been charged for certain clicks."

To minimize losses for clients and increase market share in the online advertising industry, recently ad networks have begun implementing ad-click fraud detection models. Detecting click fraud involves combining high-dimensional feature spaces with complex machine-learning algorithms. However, the datasets involved are often massive. For major advertising firms, daily ad clicks can reach hundreds of millions. Consequently, the cost of training and using such models for lookups can become prohibitive. Most research prioritizes addressing botnets as they represent the most damaging form of fraud for advertisers. Several detection and prevention measures for ad-click fraud have been explored. (Costa et al. 2012) propose using CAPTCHAs to prevent clicks not performed by actual users. (Kerkyra 2011) suggests using Splay trees to store IP addresses with unusually high ad-click counts. (Crussel, Steven & Hao Crussel et al.) uses attributes of Android apps to flag suspicious apps. A statistical approach to prevent click fraud for app developers is to measure the journey of a user's click across their portfolio, and flag IP addresses who produce lots of clicks, but never end up installing apps. With this information, they've built an IP blacklist and device blacklist. From the complex problem of ad-click fraud detection, with an abstract and ethical definition of "fraud", we propose to simplify the problem to "download or not" without reducing the problem's practicality. "Fraud" is considered as "not downloading the advertiser's product" in the remainder of this study.

Machine learning-based classification models are widely used for ad-click fraud detection. Machine learning is a field of artificial intelligence (AI) that has brought about many innovations. This learning relies on mathematics and statistics, with the goal of aiding future predictions by analyzing historical data (Amlathe 2018). Currently, many machine learning algorithms are utilized by researchers, including prediction analysis. Machine learning plays a crucial role in computing the expected utility of advertising for customers and the performance of marketing costs. The effectiveness of this advertising depends on the accuracy of predicting clicks on ads, where prediction analysis must be robust and flexible in the prediction model due to the large volume of data. Furthermore, the chosen research methods will bring about changes in the predictive analysis results of ad clicks (Kononova et al. 2020).

With the dataset we use, the number of observations is very large while the number of features provided is quite small. This creates a challenge for optimizing the amount of information available within the hardware constraints. The scope of this work is strategies for weeding out normal users who are likely to download advertised app, and identifying those most likely to be fraudsters. Advertisers and ad networks alike would benefit from the increased transparency that would result from the use of such prediction algorithms. Whether or not a user downloads the app is a yes/no decision in this binary classification issue. Since the dataset is skewed, even if we label all the test dataset as "will not download the app", we still achieve 90% or better accuracy, so Area Under Receiver Operating Characteristic curve (AUROC) and F1-score are prioritized above accuracy.

There are three categories of strategies to handle imbalanced data: the data-level approaches, the algorithm-level approaches and ensemble learning approaches. The algorithm-level method is mainly assigning higher weights to the minority class samples to modify their preference for the majority class. The data-level approach including oversampling, undersampling and feature engineering. Oversampling generate new observations for the minor class. Its advantages are high time complexity and it might causes overfitting. Undersampling, on the other hand, remove data

from the majority class, which causes loss of information.

(Liu et al. 2009) and (Feng et al. 2018) suggest the combination of sampling techniques and ensemble methods as an effective solution for an imbalanced data problem. Ensemble learning techniques have been proven to perform better than individual classifiers, especially in scenarios where the dataset is imbalanced (Hosni et al. 2019), (Khoshgoftaar et al. 2011). These techniques combine multiple weak classifier models to create a more robust and comprehensive strong model. Two common approaches for integrating base classifiers into a strong classifier are bagging and boosting. Bagging employs parallel ensemble techniques, generating base classifiers independently. In contrast, boosting is a sequential method where base classifiers are generated sequentially, with later classifiers influenced by earlier ones. However, boosting tends to run slowly and is sensitive to abnormal data and noise.

In real applications, a single strategy often cannot effectively solve the class imbalance problem. Researchers typically combine several strategies to tackle this issue. For instance, (Feng et al. 2020) improved the performance of the general vector machine (GVM) by combining feature selection and cost-sensitive learning methods. (Tao et al. 2019) adopted cost-sensitive SVM and the boosting ensemble method for imbalanced dataset classification. (Mustafa et al. 2015) successfully addressed the class imbalance problem by combining undersampling techniques with the MultiBoost ensemble method. Additionally, (Seifert et al. 2010) demonstrated that both sampling and ensemble techniques can significantly enhance the accuracy of skewed data streams. (Sainin et al. 2021) applied feature selection and sampling methods to further enhance ensemble models for class imbalance problems.

As mentioned in the previous paragraph, there have been studies where the performance of single-classifier methods and ensemble methods are compared in class imbalance scenarios. However, to the best of our knowledge, no study has compared ensemble methods to each other in ad click fraud detection context. The significant contributions of this study are as follows. Firstly, compare the performance of state-of-the-art ensemble classification methods such as Gradient Tree Boosting (GTB), CatBoost, XGBoost, LightGBM in the ad click fraud detection problem. Secondly, propose an ensemble model that integrates with data level and algorithm level methods to effectively overcome the imbalance problem in such a context. We choose some infamous data level methods such as SMOTE, an oversampling technique, Linear Discriminant Analysis, threshold adjusting and hyperparameter tuning to examine in this study.

The rest of the paper is organized as follows: We review the literature on PPC's click fraud problem, source of click fraud, single-classifier methods and some ensemble methods performance in the context of click fraud detection. Then, the theoretical background of techniques used in the study and characteristics of the dataset are presented in the Methodology and Data section. Results are presented next along, and findings are analyzed and concluded in the Discussion section.

Literature review

Click fraud and the PPC model

The term hit inflation attack, commonly known as click fraud, refers to a deceptive practice where either automated scripts or human intervention generate fraudulent clicks on online advertisements. The primary motivation behind this activity is to increase revenue for web publishers, with each click contributing to their financial gain. Importantly, the clicker does not have any genuine interest in the advertised content itself.

Several researchers have explored this phenomenon. (Knight 2005) and (Metwally et al. 2007) highlight the prevalence of click fraud, emphasizing the deliberate access to pay-per-click (PPC) links. In such cases, the intent is either to boost revenues for web publishers (as noted by Kshetri (2010)) or to deplete a competitor's PPC advertising budget. (Midha 2009) introduces the term "affiliate fraud" to describe the former type of click fraud, and "competitor fraud" specifically addresses the latter type.

Within the academic literature, click fraud emerges as a widely acknowledged drawback associated with employing pay-per-click (PPC) advertising (Cudmore et al. 2009). In a 2005 New York Times article by (Ives 2005), the use of clickbots and hitbots—programs designed to generate invalid clicks on advertisements—is highlighted. Additionally, the article points out the involvement of disgruntled employees who intentionally engage in click fraud to deplete their company's advertising budget.

Source of click fraud

In their study, (Xu et al. 2014) explore the phenomenon of click fraud, which occurs when a malicious actor initiates HTTP requests for the destination URLs displayed in advertisements. This fraudulent activity can be carried out by either human clickers or automated click bots, each exhibiting distinct characteristics.

Human clickers, as (Xu et al. 2014) assert, are motivated by financial incentives to rapidly click on multiple ads. Distinguishing them from genuine users—who typically engage in thoughtful browsing before making a purchase—becomes feasible due to this behavior. In contrast, (Pooranian et al. 2021) focuses on the differences between automated click bots and human click fraudsters. The phenomenon of "crowd fraud" involves exploiting real individuals to inflate fraudulent traffic in online advertising. While human fraudsters may coordinate crowd-click fraud through various individual accounts or computers, automated fraudulent traffic can originate from a smaller number of machines.

Detecting fraudulent traffic generated by real people proves more challenging than identifying noisy traffic produced by machines. Consequently, methods designed to detect automatic click fraud often struggle to pinpoint man-made fraud (Sadeghpour & Vlajic 2021). Hackers find it

more cost-effective to employ automated scripts simulating human ad clicks rather than hiring laborers for the same purpose (Sadeghpour & Vlajic 2021).

When a large number of clicks originate from a single computer, it raises suspicion for click fraud victims, as well as advertising networks and advertisers. However, fraudsters employ tactics to evade detection. For instance, they utilize virtual private networks (VPNs) to route bot communication through a constantly changing array of Internet protocol (IP) addresses. Additionally, click fraudsters may distribute their activity across multiple computers located in various geographic regions, thereby becoming the primary sources of high-, medium-, or low-volume clicks (Wood & Ravel 2017).

Single Machine Learning classifiers

(Mouawi et al. 2019) developed various machine learning (ML) and deep learning (DL) models to detect publishers with a high click fraud rate in the mobile advertising domain. They specifically employed Support Vector Machines (SVM), K-Nearest Neighbor (KNN), and Artificial Neural Network (ANN) algorithms. The proposed model integrated click details and user information obtained from advertising networks and advertisers to identify click behavior. The researchers utilized multiple simulated advertisement traffic datasets, each containing 500,000 advertising requests and 1000 distinct publishers.

Five key features were extracted based on information collected from advertisers. These features reflected user behavior after redirection to advertisements and included:

1. Duration of time spent examining advertisement content.
2. Number of clicks from the advertising network.
3. Percentage of suspicious clicks.
4. Duration of clicks.
5. Percentage of unique IP addresses among total clicks.
6. Percentage of app downloads.
7. Changes in the number of advertisement clicks over time intervals.

Among the models tested, KNN achieved the highest accuracy, with results reaching 98%. In (Li & Jia 2020), researchers applied SVM, Random Forest (RF), Naive Bayes (NB), and Decision Tree (DT) algorithms to an open-source fraud detection dataset in mobile advertising (Oentaryo 2014). Due to severe class imbalance, they employed balancing techniques by oversampling positive samples and undersampling negative samples. The RF algorithms outperformed other methods, achieving a prediction accuracy of 91%.

Regarding models based on TalkingData's dataset, the one we used in this study, there are some noteworthy results. (Neeraja et al. 2023) and (Thejas et al. 2019) both propose using the Random Forest Classifier for ad-click fraud detection. Makkineni et al. achieve an accuracy of 88% with Random Forest, slightly outperforming Support Vector Classifier (SVC) and KNN Classifier, which have accuracies around 87%. Thejas et al. compare Random Forest with Logistic Regression, Support Vector Machine (SVM), and Multinomial Naive Bayes using three metrics: Precision, Recall, and Accuracy.

Ensemble techniques

In the field of click fraud detection, researchers have explored ensemble techniques to mitigate various performance concerns, including noise, bias, and variance. (Dash & Pal 2020) pursued this avenue by evaluating the accuracy of both individual and ensemble machine learning (ML) algorithms for detecting click fraud in an online context. Their investigation focused on click patterns using a dataset containing 32,119 clicks collected over several days. The primary objective was to analyze users' click journeys across their portfolios and differentiate between legitimate click streams and click-spam. Specifically, they identified IP addresses that generated numerous clicks but never resulted in app installations. Among the algorithms tested, the Gradient Tree Boosting (GTB) algorithm achieved the highest accuracy, reaching 97.2%.

(Minastireanu & Mesnita 2019) followed a similar approach but used a different model. They utilized LightGBM, a recently developed decision tree-based gradient boosting algorithm known for its speed, distribution, and high performance. The dataset they used was collected by TalkingData and publicly available on (Aaron et al. 2018), which is also the dataset we used in this study. Feature engineering and selection played a crucial role in enhancing fraud detection performance. Notably, their proposed approach outperformed existing methods, achieving an accuracy of 98Similarly, Sisodia et al. (Sisodia 2021) proposed an ensemble classifier based on GTB to address challenges in identifying publisher behaviors from raw user click data. They applied their model to the FDMA 2012 dataset (Oentaryo 2014), demonstrating that GTB surpassed other ML models in terms of average precision, recall, and F-score.

Methods and Data

Dask

Dask is a newer system for dealing with big data, and has been actively gaining popularity recently. It is a new flexible parallel computing library for high-performance data analytics, designed primarily to provide scalability and extensibility to existingPython packages (like NumPy, SciPy, Pandas and others), and it can run locally or scaled to run on a cluster. Dask allows running Python programs in parallel mode with minimal code changes, taking advantage of the full computing power (Rocklin 2015). Dask also provides a real-time dashboard that highlights key metrics of user processing tasks such as project progress, memory consumption, and more (Daniel 2019). In paper (Henriques et al. 2020) it had been demonstrated the applicability of Dask to producing simple and interpretable rules for highlighting anomalies in application log data to scale and in a distributed environment.Authors implemented the k-means algorithm to separate anomalies from normal events,and a gradient tree boosting classification model to produce the interpretable meaningful rationale rule set for generalizing its application

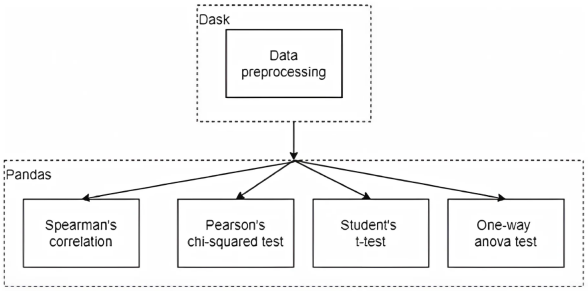

Figure 1. The system architecture

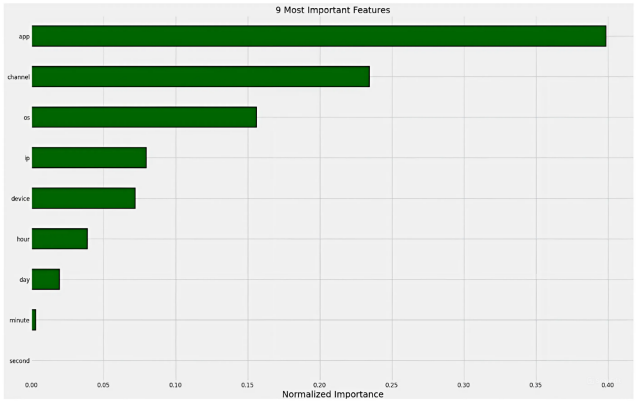

Figure 2. Feature importance in the original dataset, using XGBoost to visualize after extracting click_time

to a massive number of unseen events.

In order to obtain probabilistic and statistical models to describe the educational process, we used the methods of mathematical statistics (Spearman's correlation coefficient, Parson's test, Student's t-test, and one-way ANOVA test). High-level view of the analytical system is presented in (Fig. 1).

To test the hypothesis about the equality of the general means of two independent samples the classic Student's t-test had been implemented. Then to test the hypothesis about the correspondence of the empirical distribution to the assumed theoretical distribution Pearson's goodness-of-fit test is included. The system then used the chi-squared statistic's p-value by comparing the value of the statistic to a chi-squared distribution. The one-way analysis of variance (ANOVA) was implemented to investigate the significance of differences between the means in different groups (Scheffe 1963).

The analytical system has been deployed on high-performance computing and was carried out on a cluster consisting of 4 virtual machines with 32 GB of RAM and 16 computing cores. As already mentioned above, in computational cluster-based processing of big data, (Baig et al. 2020) based tools are also quite often used. However, Dask has advantages over Spark,which are important for our project. Dask is smaller, lighter and simpler than Spark,and flexible as Pandas with more power to compute on a cluster based computational systems, and fully based on Python. It works well on a single machine to make use of all of the cores on a laptop and processes larger- than-memory data. Therefore, program scripts are easy to debug on personal computers before deploying on a cluster.

Feature engineering

Since attributed_time has many null values, we decided to drop this feature. As we originally only have 7 features, we first splitting click_time into day, hour, minute, and second then drop click_time to estimate baseline performance and use XGBoost for feature importance visualization, as shown in Fig 1. Then we decided to make new features related to the number of clicks associated with different combinations of raw features, and create click count per day based on all combinations of ip, app, device, os and hour. For example: "ip_device_os_nunique_app" counts the number of different apps each user has clicked on, and indicates the diversity of click activities for each user, namely the number of apps they have interacted with within a certain timeframe. "ip_device_os_day_nunique_app" measures how many different apps a user clicks on within a single day. "ip_nunique_hour" counts the number of different times in the day that each IP address has clicked. This shows the time range of the day that each IP usually performs the click activity, thus providing information about the hourly behavior patterns of each IP. And since attributed_time has many null values, we decided to drop this feature. The total number of features is 31. Next, we add another 2 features of 'future click' and 'past click' for our defined user instance. A same ip_device_os combination is defined as a user and for each click data, we count the number of previous click and future click of the same user at the same day.

Interaction variable

An interaction variable or interaction feature is a variable constructed from an original set of variables to try to represent either all of the interaction present or some part of it. In exploratory statistical analyses it is common to use products of original variables as the basis of testing whether interaction is present with the possibility of substituting other more realistic interaction variables at a later stage. When there are more than two explanatory variables, several interaction variables are constructed, with pairwise-products representing pairwise-interactions and higher order products representing higher order interactions.

In this research we create interaction variables to handle categorical values. For example, we have two categorical features, "app" and "channel", with "app" being the id of the mobile app to be advertised, like Duolingo, Temple Run. . . and "channel" being the channel id of ad-publisher: Facebook, Instagram... We can create a new feature, "app_channel" to capture the interaction between the two features, as shown in the Table below. These features will be encoded by Label Encoder before modelling.

app	channel	app_channel
5	3	5_3
6	5	6_5
7	9	7_9

Linear Discriminant Analysis to predict topic of each apps

Mathematical Formulation of LDA

LDA can be derived from simple probabilistic models which model the class conditional distribution of the data $P(X|y = k)$ for each class k. Predictions can then be obtained by using Bayes' rule:

$$P(y = k|X) = \frac{P(X|y = k)P(y = k)}{P(X)} = \frac{P(X|y = k)P(y = k)}{\sum_l P(X|y = l)P(y = l)} \quad (1)$$

To use this model as a classifier, we just need to estimate from the training data the class priors $P(y = k)$ (by the proportion of instances of class k), the class means μk (by the empirical sample class means) and the covariance matrices (either by the empirical sample class covariance matrices, or by a regularized estimator). In the case (1). Multi-Level Classification 55 of LDA, the Gaussians for each class are assumed to share the same covariance matrix: $\sum k = \sum$ for all k.

This leads to linear decision surfaces between, as can be seen by comparing the log-probability ratios

$$\log \frac{P(y = k|X)}{P(y = l|X)} = 0$$
$$\iff (\mu_k - \mu_l) \sum^{-1} X = \frac{1}{2}(\mu_k^t \sum^{-1} \mu_k - \mu_l^t \sum^{-1} \mu_l)$$

Using LDA in topic predicting

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. LDA is closely related to analysis of variance (ANOVA) and regression analysis, which also attempt to express one dependent variable as a linear combination of other features or measurements(Aggarwa & C. 2014), (Agrawal et al. 1993).

However, ANOVA uses categorical independent variables and a continuous dependent variable, whereas discriminant analysis has continuous independent variables and a categorical dependent variable (i.e. the class label)(Analytics 2016). Logistic regression and probit regression are more similar to LDA than ANOVA is, as they also explain a categorical variable by the values of continuous independent variables. These other methods are preferable in applications where it is not reasonable to assume that the independent variables are normally distributed, which is a fundamental assumption of the LDA method.

In this research, the algorithm focuses on understanding network usage behavior by analyzing user usage based on their applications and devices. It aggregates user activities into sentences, which are converted into numbers for computer comprehension. The algorithm then uses Latent Dirichlet Allocation (LDA) to identify major themes in the data, such as "Entertainment" or "Work". Users are mapped to these topics based on their usage behavior, creating an expanded data table with detailed information about each user's technology usage.

LDA - A Simplified Workflow

· ***Vocabulary Selection:*** Identify the most frequently used applications to create a focus list for analysis.

· ***Aggregate Activities:*** For each IP address, create a list of activity strings by combining application names and device types.

· ***Vectorization:*** Convert these activity strings into numerical form for algorithmic processing.

· ***Apply LDA***

· ***Integrate Results:*** Add new columns to the dataset representing each topic, showing the proportion of each topic for every IP address.

Ensemble learning and diversity

Ensemble learning (Zhou 2012) is a machine-learning paradigm where multiple learners (e.g. classifiers) are trained and combined for a task. It is well known that an ensemble can usually achieve better generalization performance than single learners. To construct a good ensemble, the individual learners should be accurate and diverse. Combining only accurate learners is often inferior to combining some accurate learners with some relatively weaker ones, because the complementarity is more important than pure accuracy. Actually, a beautiful equation has been theoretically derived from errorambiguity decomposition [16]: $E = \bar{E} - \bar{A}$ where E denotes the error of an ensemble, $\bar{E}$ denotes the average error of individual classifiers in the ensemble, and $\bar{A}$ denotes the average ambiguity, later called diversity, among the individual classifiers.

In practice, diversity enhancement is based on randomness injection during training. Roughly speaking, there are four major categories of strategies (Zhou 2012). The first is data sample manipulation, which works by generating different data samples for different individuals, e.g. bootstrap sampling for bagging (Breiman 1996) and sequential importance sampling for AdaBoost (Freund & Schapire 1997). The second is input feature manipulation, which works by generating different feature subspaces for different individuals, e.g. random subspace (Ho 1998) randomly picks different subsets of features for different individuals. The third is learning parameter manipulation, which works by using different parameter settings of the base learning algorithm to generate different individuals, e.g. different initial weights can be used for different individual neural networks. The fourth is output representation manipulation, which works by using

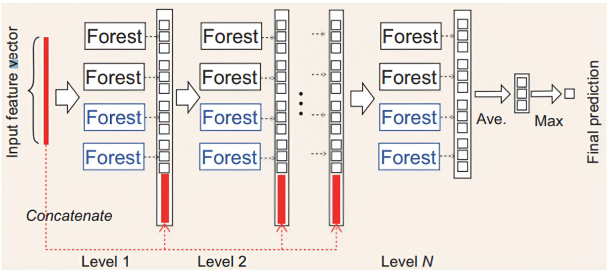

Figure 3. Illustration of the cascade forest structure. Suppose that each level of the cascade consists of two random forests (black) and two completely random forests (blue). Suppose that there are three classes to predict; thus, each forest will output a 3D class vector, which is then concatenated for re-representation of the input.

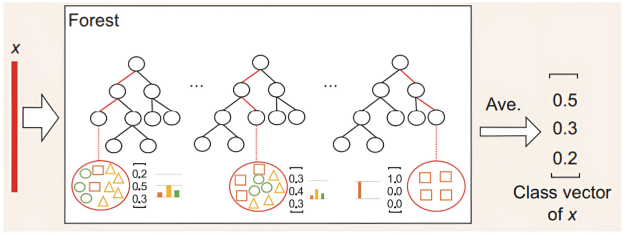

Figure 4. Illustration of class vector generation. Different marks in leaf nodes imply different classes; red highlights paths along which the concerned instance traverses to leaf nodes.

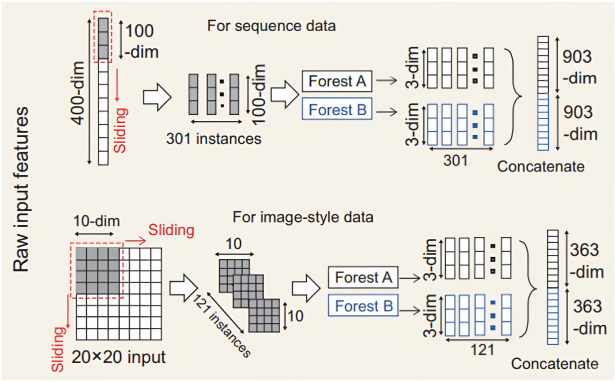

Figure 5. Illustration of feature re-representation using sliding window scanning. Suppose that there are three classes, the raw features are 400-dim, and the sliding window is 100-dim.

different output representations for different individuals, e.g. ECOC (Ho 1998) employs error-correcting output codes, whereas flipping output (Breiman 2000) randomly switches labels of training instances. Different strategies can be used together. No strategy is always effective, e.g. data sample manipulation does not work well with stable learners whose performance does not significantly change according to slight modification of training data. More information about ensemble learning can be found in (Zhou 2012). Our gcForest can be viewed as a decision-tree ensemble approach that utilizes almost all categories of strategies for diversity enhancement.

The GC Forest approach

Cascade forest structure

To realize layer-by-layer processing, gcForest employs a cascade structure, as illustrated in (Fig. 3), where each level of the cascade receives feature information processed by the preceding level.

Each level is an ensemble of decision-tree forests, i.e. an ensemble of ensembles. Here, we include different types of forests to encourage diversity. For simplicity, suppose that we use two completely random forests and two random forests (Breiman 2001). Each completely random forest contains 500 completely random trees (Liu et al. 2008), generated by randomly assigning a feature for splitting at each node, and growing a tree till pure leaf, i.e. each leaf node contains only the same class of instances as illustrated in (Fig. 4)

To improve the accuracy, gcForest also uses a technique called "multi-grained scanning" to extract different parts of the data for each layer of the forest. This allows gcForest to capture both local patterns and global relationships in the data. Finally, gcForest automatically determines the number of layers it needs by stopping training when there is no further improvement. This makes it suitable for working with datasets of different sizes.

For instances extracted from the windows, we simply assign them with the label of the original training example. Some label assignments are inherently incorrect. For example, suppose that the original training example is a positive image about 'car'; it is clear that many extracted instances do not contain a car, and therefore they are incorrectly labeled as positive. This is actually related to flipping output (Breiman 2000), an approach for ensemble diversity enhancement. Note that when transformed feature vectors are too long to be accommodated, feature sampling can be performed, e.g. by subsampling the instances generated by sliding window scanning, as completely random trees do not rely on feature split selection whereas random forests are quite insensitive to feature split selection. The feature sampling process is also related to random subspace (Ho 1998), an approach for ensemble diversity enhancement. (Fig. 5) shows only one size of sliding window. By using multiple sizes of sliding windows, multigrained feature vectors will be generated.

Overall procedure and hyper-parameters

The overall procedure of gcForest. Suppose that the original input is of 400 raw features, and three window sizes are used for multigrained scanning. Formtraining examples, awindow with a size of 100 features will generate a data set of 301 × m 100D training examples. These data will be used to train a completely random forest and a random forest, each containing 500 trees. If there are three classes to be predicted, a 1806D feature vector.

Synthetic Minority Over-sampling Technique

Fraud analysis often divides data into two groups: fraud and non-fraud, set as binary variables (value 0 - if the observation is not fraudulent and value 1 if the observation is fraudulent). However, fraud data differs from conventional data sets, with low fraud rates making the data imbalanced, making it difficult to implement binary data analysis methods. Therefore, in this study, the data preprocessing technique - Synthetic Minority Oversampling Technique (SMOTE) was used to solve the problem of imbalanced data.

The SMOTE method is implemented by increasing the fraud rate in imbalanced data from the KNN (K Nearest Neighbor) algorithm introduced by Batista et al. (2004). This method generates synthetic samples for the minority class by creating new data points based on existing ones in the minority class. More specifically, the additionally generated observations with value 1 have metric properties close to the original fraud observations. The KNN algorithm uses observations that are close to each other to create a group and find the central observation of the group. Based on the distance between observations, the method finds K neighboring observations that meet the requirements such that the distance from that observation to the center observation is not greater than the set maximum distance. The result creates additional new observations in between the original fraud observations.

Data

The dataset provided by TalkingData for the Kaggle competition contains approximately 200 million clicks over a span of 4 days. It includes information on user clicks on mobile app ads, such as click time, device information, app ID, and whether the user ultimately downloaded the app. The dataset aims to help participants build algorithms that predict whether a user will download an app after clicking on a mobile app ad, thereby aiding in the prevention of click fraud. The feature descriptions are provided in (Table. 1) below.
Note that ip, app, device, os, and channel are encoded.

Results

Most datasets related to click fraud detection suffer from significant imbalance, with the majority of the data being from the “0” class (i.e., fraudulent clicks), while the “1” class (i.e., legitimate clicks) constitutes a small fraction. This imbalance skews the prediction models towards the majority class, decrease their ability to accurately detect fraudulent clicks (Alzahrani & Aljabri 2022). Similarly, the TalkingData dataset exhibits a severe imbalance, where legitimate clicks account for only 0.25% (Is_attributed feature = 1) of all clicks, while fraudulent clicks (Is_attributed feature = 0) make up 99.75% (Fig. 6).

Consequently, models trained on such imbalanced datasets tend to be biased towards predicting the majority class, posing a significant challenge in accurately identifying the minority class. Even if we classify all new instances in the test data as 0, we still could achieve more than 99% of accuracy. Addressing this imbalance is crucial for improving

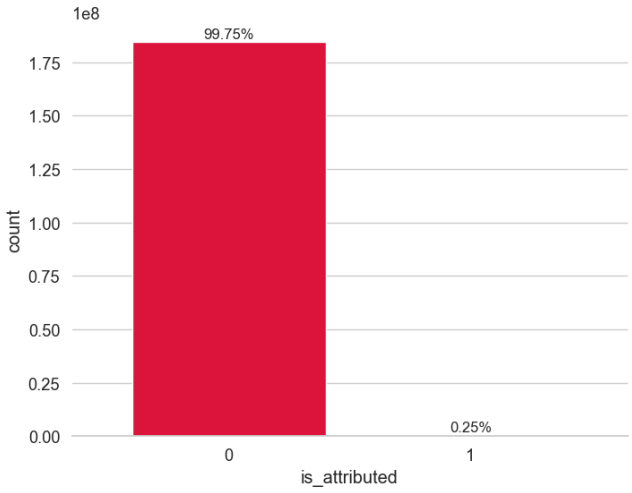

Figure 6. The distribution of Target variable

the reliability and effectiveness of click fraud detection systems.

To tackle this challenge, for the data level approach, we applied both Latent Dirichlet Allocation (LDA) and Synthetic Minority Over-sampling Technique (SMOTE). LDA is a topic modeling technique used to uncover hidden topics within a collection of documents, while SMOTE is a data augmentation technique used to address class imbalance by generating synthetic samples of the minority class. The performance of various models after applying SMOTE and LDA is presented in (Table 3).
We use two main metrics which are F1-Score and AUC Score as these two are robust in performance evaluating for imbalanced dataset.

Notable improvements were observed across most models in terms of both F1-Score and AUC. LightGBM and Majority Vote showed the highest F1-Score, indicating their effectiveness in detecting ad-click fraud. Additionally, the AUC scores demonstrated the models’ ability to discriminate between fraudulent and legitimate ad-clicks, with LightGBM and the Stacked Model achieving the highest AUC scores.

To enrich our study and provide a comprehensive analysis of class imbalance handling methods, we explored an alternative approach by replacing SMOTE with parameter and threshold adjustments. This method focuses on enhancing the model’s sensitivity to the minority class by directly modifying the model’s internal mechanics and decision criteria. The performance of the models after balancing the data by adjusting parameters and thresholds is presented in (Table 4).

The models generally show lower F1-Score and AUC scores compared to when SMOTE was used, as seen in (Table 4). This indicates that the application of SMOTE significantly improved the performance of the models in terms of data classification and separation between classes. Adjusting parameters and thresholds alone did not yield the same level of improvement, highlighting the effectiveness of

Table 1. Features Description

Features	Description
ip	IP address of click
app	App ID for marketing
device	Device type ID of user mobile phone (e.g., ip 6 plus, ip 7, huawei, . . .)
os	OS version ID of user mobile phone
channel	Channel ID of mobile ad publisher
click_time	Timestamp of click (UTC)
attributed_time	If user download the app for after clicking an ad, this is the time of the app download
is_attributed	The target that is to be predicted, indicating the app was downloaded

Table 2. Baseline Performances

Models	F1-Score	AUC
Logistic Regression	0.75322	0.82966
XGBoost	0.67022	0.67022
Light GBM	0.88791	0.88791

Table 3. Performance of models after SMOTE and LDA

Models	F1-Score	AUC
Logistic Regression	0.725726	0.823371
XGBoost	0.917244	0.970725
Light GBM	0.924723	0.972906
Gradient Boosting Tree	0.921116	0.968064
CatBoost	0.922436	0.972311
Random Forest	0.922436	0.967995
Cascaded Forest (gcForest)	0.883909	0.946987
AdaBoost	0.912749	0.960887
Majority Vote	0.926169	0.929092
XGBoost + Random Forest	0.919847	0.971870
Stacked Model	0.921172	0.972369

Table 4. Performance of models not using SMOTE

Models	F1-Score	AUC
Logistic Regression	0.798935	0.868834
XGBoost	0.926912	0.972849
Light GBM	0.924694	0.971516
Gradient Boosting Tree	0.919235	0.967547
CatBoost	0.921139	0.971044
Random Forest	0.922707	0.969053
Cascaded Forest (gcForest)	0.8942	0.94623
AdaBoost	0.897763	0.955918
Majority Vote	0.925181	0.927981
XGBoost + Random Forest	0.925864	0.971799
Stacked Model	0.927547	0.973271

SMOTE in handling class imbalance for click fraud detection.

Table 5 summarizes the enhancements in model performance achieved through Optuna tuning, with CatBoost standing out for its superior predictive accuracy in both F1-Score and AUC metrics. Our study rigorously optimized various machine learning models using Optuna to improve their performance in classification tasks. Among these models, CatBoost exhibited exceptional performance, achieving an impressive F1-Score of 0.92927 and an AUC of 0.97556. Additionally, our exploration of ensemble techniques, such as Majority Vote, XGBoost with Random Forest, and the Stacked Model, also yielded competitive results, highlighting

Table 5. Performance of models after Optuna Tuning

Models	F1-Score	AUC
Logistic Regression	0.79479	0.88279
XGBoost	0.92530	0.97577
Light GBM	0.92399	0.97330
Gradient Boosting Tree	0.92635	0.97391
CatBoost	0.92927	0.97556
AdaBoost	0.92064	0.96951
Majority Vote	0.92519	0.92799
XGBoost + Random Forest	0.92585	0.97182
Stacked Model	0.92752	0.97335

the effectiveness of leveraging diverse algorithms. These findings emphasize the importance of hyperparameter tuning in maximizing the predictive capabilities of machine learning models for classification tasks.

Discussion

Pay-per-click is a model in digital advertising in which the advertiser pays a fee to the publisher every time one of their ads is clicked on. This has given rise to various forms of fraud by publishers, aimed at incentivizing, deceiving, or even manipulating devices to encourage users to click on ads and boost publishers' earnings. This type of fraud costs advertisers billions of dollars each year, increasing costs and significantly reducing the effectiveness of marketing campaigns. Machine learning techniques have been widely used to detect click fraud. However, click datasets are often massive, extending to hundreds of millions of observations. The data is also often very imbalanced, with only a small number of clicks downloading the app. The lack of a clear definition of what constitutes click fraud also makes it difficult to find a solution. To be able to focus on the technical aspects without losing the applicability of the problem, we decided to treat the problem as a binary classification of whether a person who makes a click will download the app or not. The results can be used to create a blacklist of IP addresses that represent potential fraudsters.

Class imbalance is a common problem with this approach, because it intuitively can be seen that most people that click on a ad do not download the app. The solution of class imbalance problem can be devided into three category: data level approach, algorithm level approach, and ensemble learning approach. Many researchers have explored solutions to imbalance problem. SMOTE, an oversampling technique where the synthetic samples are generated for the

minority class, is often used to solve this problem. However, overfitting is an expected problem when using SMOTE or any other oversampling method. Ensemble learning methods can be used to reduce overfitting problem as a result of using SMOTE. There are two main kinds of ensemble learning methods: bagging and boosting. Bagging reduces bias by using multiple models on just randoms subsets sampled with replacement from data samples (Boostrapping technique). Since the results of each of these models tend to be low bias and high variance, final output is considered based on Majority Voting to increase bias and reduce variance. On the other hand, boosting train models iteratively, while making the current model focus on the mistakes of the previous ones by increasing the weight of misclassified samples.

Most previous works showed that Random Forest Classifier, the most well-known ensemble learning model, outperform traditional single-classifier models. (Neeraja et al. 2023) achieve an accuracy of 88% using Random Forest, outperform Support Vector Classifier (SVC) and K Nearest Neighbor (KNN). (Thejas et al. 2019) compared Random Forest with Logistic Regression, SVC and Multinomial Naive Bayes. The results show that Random Forest performs the best in term of precision, recall and accuracy. There also papers that compare some ensemble models with traditional Machine Learning classifier. (Dash & Pal 2020) compared several individual Machine Learning model to Gradient Tree Boosting (GTB). Among the algorithms tested, GTB achieved the highest accuracy, reaching 97.2%. (Minastireanu & Mesnita 2019) followed a similar approach but using LightGBM. In this study, LightGBM outperforms existing methods with accuracy of 98%. However, there has never been any research comparing the performance of GTB with LightGBM, Random Forest with GTB, or between ensemble models with each other.

The scope of this study is focusing on using ensemble methods, integrated with data level and algorithm level approaches to handle class imbalance in the context of ad click fraud detection. We tried to use SMOTE, an oversampling approach, to generate new minority instances and overcome imbalanced data problem. We also used LDA for feature engineering to create new variables from original variables. We also adjust parameters and thresholds, an algorithm based method, to check the effectiveness of our data level methods. We achieved best performance of 0.929 of F1-score and 0.976 of AU ROC by using SMOTE, LDA and Optuna tuning with CatBoost. From the results, we propose the combination of these techniques as a solution to class imbalance problem in ad click fraud detection problem. Another alternative is LightGBM, whose performance just slightly worse than CatBoost but better in term of runtime and tuning time (Florek & Zagdanski 2023), (Daoud 2019).

A limitation of this study is it only deals with detecting fraud in a supervised learning context. Although supervised learning methods seem attractive and yield good results, they do not perform well in dynamic environments. Forms of fraud often change over time and are difficult to detect. New datasets will need to be collected and machine learning models need to be optimized.

For future work, with further investment of time in this topic, we propose exploring the following avenues. First, although the amalgamation of CatBoost and XG Boost has shown promising outcomes, there exists potential for improvement through the integration of additional deep learning methodologies. Exploring convolutional neural networks (CNNs) and transformer-based architectures such as BERT could offer enhanced performance and more robust fraud detection capabilities. Secondly, leveraging comprehensive insights into the activities associated with each IP address and implementing an IP blacklisting strategy based on anomalous behavior holds significant potential. This proactive approach can effectively mitigate fraudulent activities by preventing suspicious IPs from engaging in ad-click interactions, thereby fostering a safer and more reliable advertising environment.

Additionally, we could explore the use of advanced deep learning models, such as DeepFM, to potentially enhance performance. DeepFM combines the strengths of factorization machines for recommendations and deep learning for feature learning within a unified neural network architecture. This model utilizes a shared input for its "wide" and "deep" components, eliminating the need for extensive feature engineering beyond raw features. This approach could leverage the complex interactions in the data more effectively, leading to improved predictive accuracy.

References

Aaron Y., Kleinman J., Elliott J., Yan T., 2018, Kaggle
Aggarwa C. C., 2014, Frequent Pattern Mining. Springer, 2014, pp 1–17
Agrawal R., Imielinski T., Arun S., 1993, IEEE transactions on knowledge and data engineering, pp 914–925
Alzahrani R., Aljabri M., 2022, Journal of Sensor and Actuator Networks
Amlathe P., 2018, ProQuest Diss. Theses
Analytics C., 2016, Anaconda 4.3.1 Documentation
Baig M., Shuib L., Yadegaridehkordi E., 2020, Int. J. Educ. Technol. High. Educ. 17, pp 1–23
Breiman L., 1996, Mach Learn
Breiman L., 2000, Mach Learn
Breiman L., 2001, Random forests. Mach Learn
Costa R. A., de Queiroz R. J. G. B., Cavalcanti E. R., 2012, IEEE Sixth International Conference on Software Security and Reliability Companion, pp 62–67
Crussel J., Steven R., Hao C., , Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14). ACM, NY, USA, pp 123–134
Cudmore B. A., McCoy J., Shuhy J., Taylor J., 2009, Journal of Internet Commerce, pp 288–308
Daniel J., 2019, Manning Publications
Daoud E. A., 2019, Engineering and Technology International Journal of Computer and Information Engineering, 13
Dash A., Pal S., 2020, Indones J Educ Sci, 10
Feng W., Huang W., Ren J., 2018, Appl Sc
Feng F., Li K., Shen J., Zhou Q., Yang X., 2020, IEEE Access
Florek P., Zagdanski A., 2023, Benchmarking state-of-the-art gradient boosting algorithms for classification
Freund Y., Schapire R., 1997, J Comput SystSci
Henriques J., Caldeira F., Cruz T., Simões P., 2020, Electronics 9(7), 1164
Ho T., 1998, IEEE Trans Pattern Anal Mach Intell

Hosni M., Abnane I., Idri A., de Gea JM C., JL F. A., 2019, Comput Meth Programs Biomed
Ives N., 2005, The New York Times
Kerkyra G., 2011, IEEE Symposium on Computers and Communications (ISCC), pp 1111–1116
Khoshgoftaar T., J. V. H., A. N., 2011, IEEE Trans Syst Man Cybern A Syst Hum
Knight W., 2005, Software Bots Could Menace Google Ads
Kononova A., Kim W., Joo E., Lynch K., 2020, Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, 39
Kshetri N., 2010, IEEE Security Privacy, pp 45–53
Li Z., Jia W., 2020, J Comput, 31, pp 256–265
Liu F., Ting K., Y. Y., 2008, J Artif Intell Res
Liu X., J. W., Zhou Z., 2009, IEEE Trans Syst Man Cybern
Liu B., Nath S., Govindan R., Liu J., 2014, Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, pp 57–70
Metwally A., Agrawal D., El Abbadi A., Zheng Q., 2007, 27th International Conference, pp 52–52
Midha V., 2009, International Journal of Electronic Commerce, pp 91–111
Minastireanu E., Mesnita G., 2019, J Inf Assur Cybersecurity
Mouawi R., Awad M., Chehab A., El Hajj I. H., Kayssi A., 2019, Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing
Mustafa G., Niu Z., Yousif A., J. T., 2015, 2015 10th Iberian conference on information systems and technologies (CISTI)
Neeraja N., Ciripuram A., Sriram N., Subhani S., Kakubalati V., 2023, Journal of Scientific Research and Reports, 29, 84
Oentaryo R., 2014, Detecting click fraud in online advertising: A data mining approach. Vol. 15
Pooranian Z., Conti M., Haddadi H., Tafazolli R., 2021, IEEE Commun. Surv. Tutor, pp 2494–2524
Rocklin M., 2015, Proceedings of the 14th Python in Science Conference, Austin, pp 130–136
Sadeghpour S., Vlajic N., 2021, Computers 2021
Sainin M., Alfred R., Ahmad F., 2021, J Inf Commun Technol
Scheffe H., 1963, Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation
Seifert C., Khoshgoftaar T., J. V. H., Napolitano A., 2010, IEEE Trans Syst Man Cybern A Syst Humans
Sisodia D., 2021, Data Technol Appl, pp 216–232
Tao X., Li Q., Guo W., Ren C., Li C., Liu R., 2019, Inf Sci
Thejas G., G.Boroojeni K.Chandna I.Bhatia S.S.Iyengar N.R.Sunitha 2019, ACM Southeast Conference – ACMSE 2019 – Session 2: Short Papers, pp 84–89
Tuzhilin A., 2006, The Lane's Gifts v. Google Report
Wood A., Ravel A., 2017, Fool me once: Regulating fake news and other online advertising
Xu H., Liu D., Koehl A., Wang H., Stavrou A., 2014, European Symposium on Research in Computer Security, pp 419–438
Zhou Z., 2012, BocaRaton, FL: CRC Press