IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 January 2025, accepted 14 January 2025, date of publication 20 January 2025, date of current version 23 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3532200

RESEARCH ARTICLE

Ad Click Fraud Detection Using Machine Learning and Deep Learning Algorithms

REEM A. ALZAHRANI[1], MALAK ALJABRI[2], AND RAMI MUSTAFA A. MOHAMMAD[3]

[1]SAUDI ARAMCO Cybersecurity Chair, Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
[2]Department of Computer and Network Engineering, College of Computing, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[3]SAUDI ARAMCO Cybersecurity Chair, Department of Computer Information Systems, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia

Corresponding author: Reem A. Alzahrani (2210500199@iau.edu.sa)

This work was supported by the SAUDI ARAMCO Cybersecurity Chair, Imam Abdulrahman Bin Faisal University.

ABSTRACT In online advertising, click fraud poses a significant challenge, draining budgets and threatening the industry's integrity by redirecting funds away from legitimate advertisers. Despite ongoing efforts to combat these fraudulent practices, recent data emphasizes their widespread and persistent nature. Toward detecting click fraud effectively, this study employed a comprehensive feature engineering and extraction approach to identify subtle differences in click behavior that could be used to distinguish fraudulent from legitimate clicks. Subsequently, a thorough evaluation was conducted involving nine diverse machine learning (ML) and Deep Learning (DL) models. After Recursive Feature Elimination (RFE), the ML models consistently demonstrated robust performance. DT and RF surpassed 98.99% accuracy, while GB, LightGBM, and XGBoost achieved 98.90% or higher. Precision scores, measuring accurate identification of fraudulent clicks, exceeded 98% for models like ANN. In parallel, deep learning (DL) models, including Convolutional Neural Network (CNN), Deep Neural Network (DNN), and Recurrent Neural Network (RNN), showcased strong performance. RNN, in particular, achieved 97.34% accuracy, emphasizing its efficacy. The study underscores the prowess of tree-based methods and advanced algorithms in detecting click fraud, as evidenced by high accuracy, precision, and recall scores. These findings contribute valuable insights to combat click fraud and establish the groundwork for the strategic development of anti-fraud measures in online advertising.

INDEX TERMS Click fraud, machine learning, deep learning, online-advertising, bot detection, pay-per-click, fraud.

I. INTRODUCTION

Today's world increasingly depends on web services, including marketing activities conducted on websites and smartphones. Among the most essential of these services are marketing or advertising campaigns, which are visible across a spectrum of websites and applications and take the form of advertisements to attract visitors and potential customers to draw attention to the promoted service or product. Through advertising campaigns, advertisements will be displayed on relevant web pages to increase profits. In these campaigns, advertisers pay fees for each click they receive on the ad, which is referred to as pay-per-click (PPC). Clicks on these advertisements may originate from legitimate, unsuspecting web users. Still, they may also occur due to malicious clicks conducted by individuals or software developed by rivals with illegal intentions. In these cases, the motivations may include maximizing the benefits of an organization or extracting excessive fees from advertisers.

Juniper Research's [1], the most recent ad fraud report estimates that by the end of 2023, the anticipated cost to advertisers will reach $84 billion, representing more than a fifth (22%) of all online advertising expenditures, based on the analysis of over 78,000 data sets of ad activity spanning 45 countries and eight major global regions.

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Wang.

Moreover, by the end of 2023, it's estimated that 17% of PCs and desktop clickthroughs will be invalid, not delivering a return on ad spend (ROAS). The forecast indicates that although the number of valid clickthroughs will rise from 160 billion in 2023 to nearly 235 billion by 2028, the number of fraudulent clickthroughs will increase from 37 billion in 2023 to more than 65 billion by 2028.

Since malicious bots are growing in number and diversity, extensive research has been conducted to understand what causes these misleading clicks and how they can be detected and predicted [2]. To detect instances of click fraud, a range of artificial intelligence (AI) models are used to analyze when an advertisement is clicked on by either a human or a computer program. By assessing the authenticity of a click, these models aim to distinguish between legitimate and fraudulent interactions.

Click fraud is frequently committed by automated means, such as bots or similar tools that resemble genuine human activity on websites. By repeatedly clicking on advertisements, these bots mislead platforms into believing that an actual human being is engaged in the advertised product or service [3]. Many clicks originating from a single device can be detected more quickly when an advertising network or advertiser becomes suspicious of click fraud activity. Cybercriminals, however, can bypass this mechanism by using virtual private networks (VPNs) to route bot communication through a wide array of frequently changing internet protocol (IP) addresses. Furthermore, they can commit click fraud by using multiple computers in different geographical locations, thus allowing them to diversify the source of clicks, which can be high, medium, or low in volume [4]. Despite marketers' efforts to fight click fraud, current statistics indicate that the issue is widespread and is expected to worsen in the future. According to the latest available data, marketers incurred about $71.37 billion in 2024 due to fraudulent clicks [5]. Due to the emergence of botnets based on fraudulent clicks, it has become increasingly essential to investigate this issue in detail to determine effective solutions.

Furthermore, due to recent advances in artificial intelligence (AI) technologies and their wide application in cybersecurity, various defensive systems designed for advertising networks have been developed to detect click fraud activity. Thus, attackers have become more proficient at committing click fraud, adopting tactics that mimic everyday user behavior to avoid detection. In light of this evolution, there is a growing need for more robust and cutting-edge solutions to detect click fraud.

Our objective in this study is to develop multiple machine learning (ML) and deep learning (DL) models capable of distinguishing between humans and bot visitors to a website. In this study, we worked on a real-time dataset containing the browsing behavior of internet users as they interact with websites. Following the preprocessing of the dataset, we derived a set of novel features and subsequently identified the most influential ones. Based on these selected features, ML and DL models were applied to the dataset, and then, a comprehensive evaluation of model performance was conducted using different measurements.

The remainder of this document is arranged as follows. A summary of relevant work on ad clicks and fraudulent clicks appears in Section II. The objectives and contributions of this paper are summarized in Section III. The proposed methodology and framework for this study are presented in Section IV. ML and DL models and performance metrics for evaluating their performance are given in Section V. Section VI offers and discusses the results of the training and testing of the models to detect click fraud. Finally, Section VII concludes the study.

II. RELATED WORKS

Previous approaches have focused on using AI technologies such as machine learning (ML) and deep learning (DL), in addition to several successful methods already used for detecting click fraud. The primary purpose here is to protect advertisers from suffering significant expenditure due to misleading clicks, which can significantly influence the success of their marketing campaigns. Earlier research has demonstrated that employing AI methods to differentiate between valid and false ad clicks has proven quite effective.

Most solutions for tracking the origin of a click have depended on ML techniques. Various tree-based models, such as Decision Trees (DT), Random Forests (RF), and Extremely Randomized Trees (ERT), have shown impressive performance. Decision Trees build individual tree-node models, while Random Forests and Extremely Randomized Trees create multiple decision trees simultaneously using different strategies. Furthermore, other variations of tree-based methods have been employed. For instance, Berrar [6] utilized Random Forests (RFs) with skewed bootstrap sampling to determine whether a publisher's clicks were legitimate or fraudulent. They included the click profile (time gaps between clicks) for analysis. Two tests were conducted using different subsets of included features. The first model exhibited the best performance, with an average accuracy of 49.99% in the validation and 42.01% in the test set. Yan and Jiang [7] trained several classifiers with numerical features like IP addresses and the count of clicks at different time intervals during the day, along with statistical features. Classifier models like RFs, Bayesian networks (BNs), decision tables, REPTree, and naive Bayes (NB) were employed. Their findings highlighted that tree-based methods outperformed Bayes's approaches due to the imbalance between fraudulent and valid clicks, with fraudulent clicks being the majority. Perera et al. [8] introduced a novel ensemble model based on user behavior patterns from click data. They derived new valuable features from raw data that couldn't be used in their original forms to detect click fraud. They experimented with various classifiers, ultimately creating an ensemble model that integrated the six most effective classifiers: bagging with J48, bagging with REPTree, bagging with RF, MetaCost with J48, LogiBoost with J48, and random subspace with J48. This ensemble approach demonstrated its effectiveness on

validation and test sets, showcasing its generalizability with an accuracy of 59.39%.

Gradient boosting models (GBM) and their variations have been extensively used in past research due to their effectiveness in extracting features and classifying clicks. These models work by sequentially creating many decision trees. A version of GBM known as extreme gradient boosting (XGBoost) offers a more controlled version of Gradient Boosting. For instance, Phua et al. [9] employed generalized boosted regression models and identified distinct spatiotemporal patterns in fraudulent clicks. They adopted simple statistical techniques to extract features related to click behavior, frequency, and high-risk actions, significantly improving model performance and preventing overfitting during training. In another study, Minastireanu and Mesnita [10] utilized a state-of-the-art ML algorithm called light gradient boosting (LightGBM) to analyze the actions of visitors who frequently click on ads but do not complete the desired action, like downloading an app. By engineering features and extracting time-related information from click times, they achieved an impressive 98% accuracy. Addressing the complexities of click fraud in the advertising sector, Singh and Sisodia [11] emphasized the need for careful algorithms. They chose the gradient tree boosting (GTB) model, which outperformed 11 other ML algorithms in detecting fraudulent clicks. Dash and Pal [12] constructed a click fraud detection model employing different ML methods, including SVM, KNN, DT, RF, and GBDT, to understand better the behavior of individuals who regularly click on advertisements without accomplishing the desired action. However, the study had limitations due to the lack of comprehensive user-related features and click behavior statistics, such as mouse movement patterns. Mouawi et al. [13] introduced the click fraud detection model incorporating custom-designed features and ML models like KNN, artificial neural network (ANN), and SVM. What sets this model apart is its ability to let a third party oversee the entire click fraud detection process through crowdsourcing, differentiating it from standard models where ad networks or advertisers control the task. Their findings showed that KNN with $K = 5$ (number of neighbors) achieved an impressive accuracy of 98.26%. In a study by Aljabri and Mohammad [14], various valuable features were extracted from Beacon's raw dataset, including User-Agent, Number of Webpages Viewed, Journey Duration, Visit Termination, Action Executed, and Number of Actions. In this regard, the study evaluated a variety of intelligence algorithms, including DT, SVM, NB, Ripper, PART, NN, and RF. The empirical findings were beneficial across all algorithms tested, confirming the usefulness of the mainly constructed features. Remarkably, RF outperformed other models in the experiment with 84% accuracy. Notably, the results highlighted that distinguishing bot gained higher accuracy than identifying human users. Using ML, Neeraja et al. [15] were able to predict fraudulent user clicks and, therefore, distinguish fraudulent users from legitimate ones. Using the Kaggle dataset; they tested KNN, SVC, and Random Forest, models. All models performed well, with an accuracy of up to 87%. As part of the new framework, Sisodia [16] proposes two new stack generalization structures: one for resampling and the other for classification. The proposed structure's performance was compared with that of the previous literature based on the FDMA 2012 dataset. Stack generalization with gradient tree boosting (GTB) achieved a 66% average precision (AP). In another work, according to Sisodia et al. [17], feature importance was tested using GTB, which proven to be an effective model design strategy that improves classification performance. Based on the features selected in the dataset, it achieved an AP of 64.86%. According to Dekou et al. [18], detecting fraud in e-commerce is challenging since datasets are often imbalanced and fraudulent humans or bots constantly adapt their behaviors to remain undetected. Models are built using the powerful open-source libraries H2O and Catboost (GB techniques for DT libraries) as well as AutoML, which uses multiple basic models to improve prediction quality and produce a more efficient aggregation model. Among the ensemble models, it was noteworthy that the stacked ensemble model achieved the highest performance with an area under the curve of 98.85%. In order to minimize capacity requirements and increase execution performance, Singh and Sisodia [19] developed a quad division prototype selection-based K-nearest neighbor classifier (QDPSKNN). An undersampling process is carried out by dividing the data into four quarters and then undersampling each quarter. It was compared with a traditional KNN model. Based on the results of the study, QDPSKNN improved classification and addressed data imbalance, obtaining the best accuracy in 13 out of 16 datasets and achieving a precision rate of 75.1% in the click-related dataset. Kirkwood et al. [20] introduced a solution using LR, RF, and Neural Network (NN) models to evaluate classification performance across different training/testing splits (80%-20%, 70%-30%, and 60%-40%). Their study highlighted that RF and NN models consistently achieved high recall and F1 scores, particularly with the 80%-20% split, which showcased the robustness of these models. This comparative approach demonstrated the models' effectiveness and stability, with minimal variation across different data splits, ensuring reliable performance in click fraud detection. Singh et al. [21] proposed a reliable click-fraud detection system. They tackled class imbalance using SMOTE and RUSBoost techniques, while introducing a Hybrid-Manifold Feature Subset Selection (H-MFSS) method to identify optimal features. Their approach leverages a GTB model to classify fraudster behavior effectively. Experimental results on the FDMA2012 dataset demonstrated that the GTB model achieved a significant performance improvement, with AP scores of 64.86% without sampling, 65.25% with RUSBoost, and 66.78% with SMOTE, showcasing its robustness and efficacy.

Batool and Byun [22] proposed an ensemble architecture combining machine learning and deep learning techniques

to detect click fraud in online advertising. Their model integrates a Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) for feature extraction, followed by a RF classifier to categorize clicks as fraudulent or non-fraudulent. The model achieved impressive results, with 99.19% accuracy, 99.89% precision, and 98.50% sensitivity, outperforming other ensemble and standalone models. Purwar et al. [23] aimed to address click fraud by developing a precise detection model using recursive feature elimination (RFE) and an ensemble classifier with Hellinger distance-based decision tree (HDDT). Their model was evaluated on the TalkingData dataset from Kaggle and achieved a remarkable accuracy of 99.72%, outperforming traditional DT methods. This approach demonstrates the effectiveness of combining RFE with HDDT in enhancing click fraud detection accuracy.

III. OBJECTIVES

The objectives of this study are outlined as follows:

1) Based on the examination of the features employed in previous studies, as discussed in our prior publication [24], to identify click fraud and ascertain which features are considered reliable indicators of a click's identity (whether benign or fraudulent). These aspects will be emphasized and incorporated during our feature engineering phase.

2) Additionally, this paper will concentrate on extracting all conceivable features associated with users' browsing behavior during ad clicks (such as duration of stay on the webpage, scrolling activity, correct termination of browsing sessions, etc.) from our novel dataset.

3) To extract novel features not previously addressed in the existing literature and utilize them to develop and train robust ML and DL models to detect fraudulent clicks.

IV. METHODOLOGY

The proposed framework is intended to handle a real-time novel dataset and engineer and extract all possible features from that dataset to spot click fraud. Specifically, the purpose is to distinguish between illegitimate (i.e., bot clicks) and genuine clicks (i.e., human clicks). This methodology has been developed to address the current issue of increasing advertising click fraud. Based on a thorough analysis and testing process, the design was developed. Furthermore, the dataset has been trained and tested extensively. The proposed methodology is illustrated in Figure 1 below.

In this section, we outline the workflow approach, beginning with a description of our dataset and how it was handled, features engineering, and extraction, followed by an assessment of the features.

A. DATASET DESCRIPTION

The dataset was obtained from July 2022 through February 2023. It was provided by The Veracity Trust Network Company (''Click Fraud Protection - Detect & Prevent Click Frau,'' n.d.) [25] as nested JSON files, which are just simple JSON files with a considerable fraction of their values consisting of other JSON objects. We utilized Python language in Jupyter Notebook to handle the nested JSON files. To begin, we combined all the nested JSON files into a single JSON file to handle and extract the features from it more easily.

Initially, there were 30 features before we flattened them to extract the nested features. In the end, we were able to obtain 290 features. Obviously, some features are either duplicated or do not relate to the focus of our study, which is clicking fraud detection. Thus, all of these features have been eliminated. As a result, we were able to obtain around 225 features. Notably, about 200 out of the 225 raw features are explicitly related to mouse movements action; these features are named c1, c2, c3,..., and c200. An advertiser's page may contain a variety of actions the user takes (for example, scroll down, scroll up, click a button, move a mouse, play a video, etc.). If the action is ''mouse movements,'' then the record corresponds to additional features that explain the behavior of one mouse movement at a given moment. Therefore, if the user moves the mouse five times, this will correspond to five additional features for that specific click (features from c1 to c5), each describing one mouse movement (such as time, coordinates (x, y) of mouse movement, etc.). Accordingly, if there are 77 mouse movements, then there will be 77 additional features (features from c1 to c77), which will describe detailed information regarding each mouse movement and so on.

B. DATASET PRE-PROCESSING

1) HANDLING MISSING VALUES

It is common for real-time datasets to contain missing values, one of the most common problems that threaten data quality [26].

In our dataset, approximately 30% of values are missing. Missing values on several features reached around 97% of the values; imputing such huge quantities will almost certainly result in bias, compromising the validity and accuracy of the model's predictions. Thus, if the missing data percentage reaches 80% or higher, the feature will be eliminated since the quantity of information captured in that feature is insufficient and will not contribute to the prediction model [27]. In this study, the MissForest approach [28] was used to impute missing data for the remaining features, given that most of the features in our dataset are categorical and imputing missing data using simple statistical methods such as mode (most frequent values) or mean and median (for numerical features) may affect the quality of the data and impose bias. MissForest outperformed and was more efficient than other imputation methods, such as traditional statistical approaches or K-NN-based imputation [29], [30].

2) ENCODING CATEGORICAL FEATURES

Given that some raw and extracted features, such as the '*action'* and '*user agent,'* contain categorical values, they are converted into numerical features.

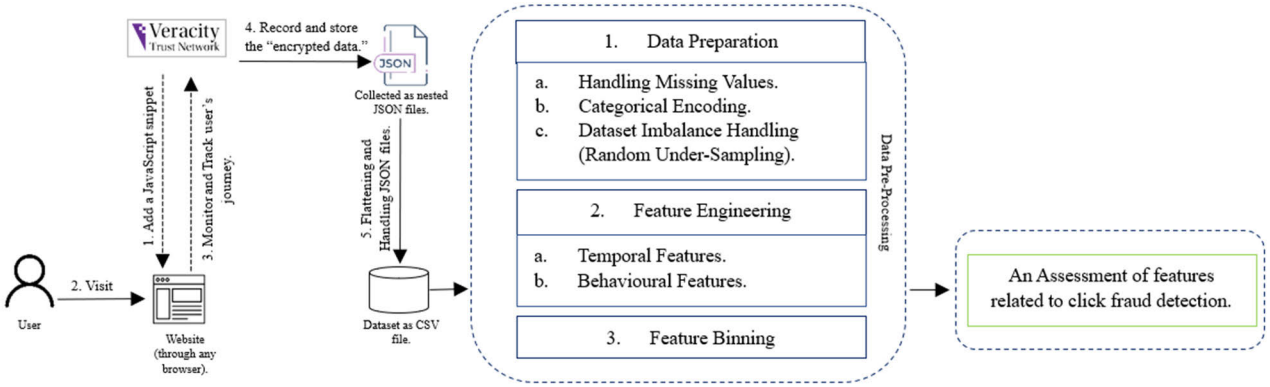

FIGURE 1. Research methodology.

Label Encoding techniques were employed for various features such as *'host', 'action,' 'os'* and so on. This approach allocates a unique integer to each distinct category or label within a categorical variable. This transformation simplifies the data, making it better suited for ML algorithms that require numerical input.

3) DATASET IMBALANCE HANDLING

Initially, our dataset consisted of 73,303 records, in which a total of 41,954 human clicks were recorded, while 32,362 bot clicks were recorded. As can be seen, there is an imbalance between the distribution of clicks on the two classes.

Dataset imbalance occurs when classes are distributed unevenly in a dataset, with some classes having significantly fewer instances than others [31].

Thus, in this paper, we applied a random under-sampling technique to address this issue [32]. We end up with 64,724 records and 32362 clicks in each class, benign or fraudulent.

All datasets have imperfections, especially those collected in real-world scenarios. As mentioned earlier, we encountered challenges such as missing data and imbalanced class distributions. In addition, some raw features were highly correlated with the target class, which might have produced overfitting. As a solution to this issue, we eliminated these strongly correlated features to enhance the model performance and generalization.

V. FEATURE ENGINEERING

In this section, we will address two significant topics: first, we list some of the raw features available in our dataset and categorize them based on the segmentation we structured in our previous paper [24]. This will be discussed in sub-section named Feature's Structure. Secondly, we engineer and extract additional features from existing raw features for both the bot and human classes. This was discussed in subsections II, and III named as Feature Extraction and

Feature Binning respectively. It should be noted that some of these features might not be utilized in future model training since the best feature set will be selected to train the model using the features selection technique. For efficient detection, informative features must be utilized to distinguish benign clicks from fraudulent ones. Therefore, it is imperative to pay close attention to the features.

TABLE 1. Temporal features.

Feature	Description
startOfVisit	A feature that reveals the start of a user's journey, including the date and time.
lastActionInVisit	A feature that indicates the date and time of the user's last action on the journey.
start	The timestamp of the beginning of the user's journey.
end	The timestamp of the end of the user's journey.

TABLE 2. Clicker Behavior features.

Feature	Description
actionsInVisit	The total number of actions taken by the user throughout the journey.
action	The user's interactions throughout the journey, like engaging with a video, scrolling upwards or downwards, and so on.

A. FEATURES' STRUCTURE

Listed below are the raw features that relate to our study focus, approximately 25 features, followed by the extracted features from our dataset.

1) TEMPORAL FEATURES

The temporal features obtainable in our dataset are listed in Table 1 below.

2) CLICKER BEHAVIOR FEATURES

Recognizing and tracking user behavior is one of the most crucial indicators for distinguishing bots from humans.

TABLE 3. Medium and IP features.

Feature	Description
IP	A distinct address that identifies a device on the internet or a local network.
OS	The operating system is running on the device.
name	The name of the web browser from which the journey was made.
Full_version	A number indicating the browser's full version.
Major_Version	A number indicating the browser's major version.
User-agent	HTTP (Hypertext Transfer Protocol) requests a header field describing the utilized browser.
Mobile	A binary feature that indicates whether a mobile phone was utilized.

TABLE 4. Ad-Related features.

Feature	Description
Website_id	A unique identifier for each website.
Host	The URL of each host/publisher.
landingPage	The landing page that the user went to after clicking on the ad.
Facebook	A feature that indicates whether the ad campaign took place on the Facebook platform.
Google	A feature that shows whether the ad campaign was carried out on Google or not platform.

Moreover, there are significant distinctions between them. We have some features in terms of user behavior, which are as follows in Table 2.

3) MEDIUM AND IP FEATURES

In Table 3, we go through the features related to the device or agent that the user utilized when surfing and clicking on ads.

4) AD-RELATED FEATURES

In addition to the observed features from the user's side, some other features relating to the advertisement itself, the publisher, or the advertising campaign may be valuable in fraud detection, as shown in Table 4.

5) OTHER FEATURES

Furthermore, key aspects define a user's journey as they navigate web pages, which can be a valuable predictor of the differences in bot and human journey patterns. Table 5 summarizes the most essential of these features.

subsectionFeature Extraction As we discussed before, various past efforts have been made to identify and extract

TABLE 5. Other features.

Feature	Description
Click id	A unique identifier for each click.
Cookie	A cookie refers to a data fragment originating from a website, which is stored within a web browser and can be subsequently retrieved by the same website.
visitHas-Ended	A binary feature that returns one if a session has ended and 0 otherwise.
Pages-Viewed	The number of web pages viewed per journey.
InVisit	

effective features for more reliably detecting click fraud. Phua et al. introduced a set of features among these attempts. [9] to extract more comprehensive aspects of the click pattern, whether it is a bot or a human click. As illustrated in Figure 2, some temporal features were extracted from the raw feature and inspired by the set of features in which we extracted new features from the *'startOfVisit'* raw feature. Thus, we used the same concept to extract precise and descriptive temporal features for bots and humans from our dataset. In addition to the extracted features inspired by the aforementioned work, we have extracted additional features associated with user behavior, such as features that monitor the mouse movements, for instance, whether the user spent time moving the mouse, average speed, direction of movement, and so on. We also investigated how long the person spent on their journey, among other features.

Moreover, Tables 11 and 12 in the appendix section show the extracted temporal and behavioral features from our dataset, along with their descriptions across the feature structure we organized above.

B. FEATURE BINNING

Feature Binning is a data preprocessing technique where continuous numerical features are grouped into discrete bins or intervals. When dealing with time-related features or variables that have many unique values, this approach can be applied in practice. In this study, features were binned based on strict criteria such as having over 50 distinct values or being temporal [103]. The width of each bin for all features was calculated using the Freedman-Diaconis rule. It offers a data-driven method for determining the right bin width and it will improve the fidelity and interpretability of discretization for continuous characteristics by making sure that their properties guide it [33].

VI. AN ASSESSMENT OF FEATURES RELATED TO CLICK FRAUD DETECTION

In the end, we obtained around 90 features. In this section, we analyze a few of these features. We begin by reviewing the raw features wherein the bot (Target = 0) and human

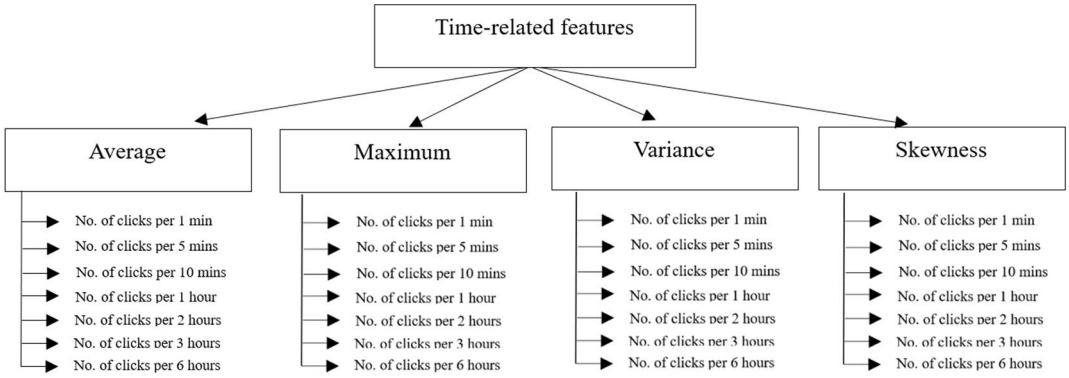

FIGURE 2. Time-related features extracted from 'startOfVisit' feature.

(Target = 1), Table 13 in the appendix details the original values and ranges for each feature shown on the x-axis.

'pagesViewedInVisit': When considering the number of pages visited, different trends can be identified concerning how humans and bots operate. Among the individuals who viewed 0-14 pages (pagesViewedInVisit' = 0), human beings represent 56.41% while for bots it is 43.59%. However, as more people view pages, there is a significant drop in the proportion of human viewers. Bots make up88.33% of all those who have accessed between 15 and 29 pages, with real users accounting for the remaining 11.67%, as seen in Figure 3. Nevertheless, bot interactions decrease in percentage as the number of viewed pages increases to reach zero percent at above thirty visits by a user, indicating that their behavior tends to involve fewer pageviews compared to humans on this platform.

A. 'ACTIONSINVISIT'

An examination of user behavior and the distinction between bots and humans through the number of activities done on a visit. An almost equal distribution of bots and humans can be observed when users engage in 0-72 actions in their visit as shown in Figure 4, where the human actions account for 48.18%, whereas bot interactions are responsible for 51.82%. As a visit's action count increases, the percentage of bot interactions drops progressively. Incredibly, it could be seen that bot actions take up the majority, with 76.86% for those who have taken between 73 and 145 actions, while human interactions represent only 23.14%. The ratio of bot interactions goes down as well, and human interaction increases with the rise in the number of actions, thereby continuing this trend.

B. 'NAME'

From the data, it has been established that most interactions between different browsers or applications are related to human activities. For instance, (0) in Figure 5 means Chrome, which is the dominant browser with a 46.16% representation of human activities. Moreover, Safari (1) and Firefox (3) alone contribute towards human interactions; this again confirms that these browsers relate to real user actions. Furthermore, other names of browsers or applications, such as Klarna (4), FBAN (7), Netscape (9), and OPT (102), show no bot interaction and are exclusively associated with human activities.

C. 'MOBILE'

'Mobile' feature analysis of user activity unveils patterns, which distinguish the human and bot classes. The majority (99.67%) of users who did not use a mobile device during their visit (shown by 'mobile' = 0) in Figure 6 are classified as humans. This demonstrates a visible split in user behaviour since it means that individuals commonly employ non-mobile devices to reach out to the system. However, visits by mobile users indicated by 'mobile' = 1 show a more even distribution; here, humans contributed to only 46.25% of interactions, while bots accounted for the rest 53.75%. Notably, there is a high number of bot interactions in the mobile category; this suggests that automated activities are more common on mobile devices than on any other platform.

D. 'OS'

Furthermore, the examination of user behavior based on the 'os' (operating system) feature reveals clear patterns separating human and bot classes of behavior. Actual humans with a 'Windows' operating system make up for 99.5% of the users as seen in Figure 7. This suggests a high level of correlation between human interactions on this platform and the 'Windows' operating system. Likewise, most users classified under 'Linux' (coded as os=1) and 'Mac OS X' (coded as os=2) also belong to a human class, constituting 100% and 99.91%, respectively, out of these groups. On the other hand, ''Android'' is linked to both bot and human activities such that while 54.13% are due to bots, 45.87 percent are due to humans. In 'Android,' there may be many bots, whereas none in 'Linux' or 'Mac OS X,' however, it looks like some devices tend to be more susceptible to automation than others.

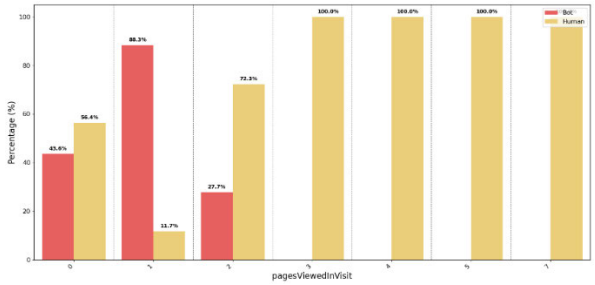

FIGURE 3. Assessment of 'pagesViewedInVisit' feature.

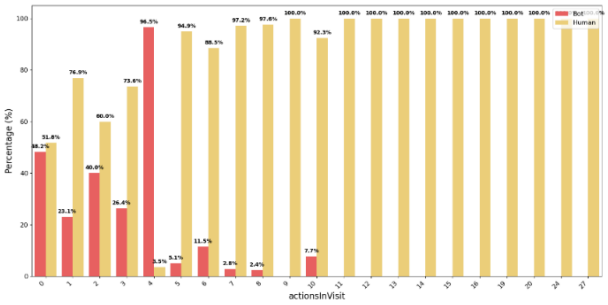

FIGURE 4. Assessment of 'actionInVisit' feature.

E. 'ACTION'

To understand the unique patterns of human and bot interaction, user behavior based on the 'action' feature should be analyzed. Human and bot classes are characterized by a distribution of interactions with 'click' (encoded as 'action' = 1) and 'first_pass_completed' (encoded as 'action' = 4). On the contrary, for 'click', 43.82% were related to bots while humans accounted for 56.18%. By contrast, there is a more significant discrepancy between those activated with ''first_pass_completed'': 30.94% that concern people and 69.06% that concern bots. In this regard, it can be assumed that automated activities might involve completion of the first pass only while clicks are more evenly divided among bots and humans. Additionally, two actions — 'mouse_updates' (encoded as 'action' = 10) and 'scroll_down' (encoded as 'action' = 11) have greater frequency in human interactions than any other; Figure 8 shows that 94.59% and 72.76%, respectively, belong to these classes. These behaviors could represent typical mouse movements or scrolling behavior by users.

Next, we will go over and examine some of the extracted features, in addition to their associations with the target class.

1) '# OF CLICKS_MORNING'

In the morning, the data displays a broad range of click counts and different bot-to-human interaction ratios. However, none at all or just one, two or three clicks in the morning are mainly related to human interactions and have very low bot engagement proportions. As the number of morning clicks rises, there is still an increase in the rate of bot interactions. Nonetheless, counts 5, 6 and 9 have a high percentage of

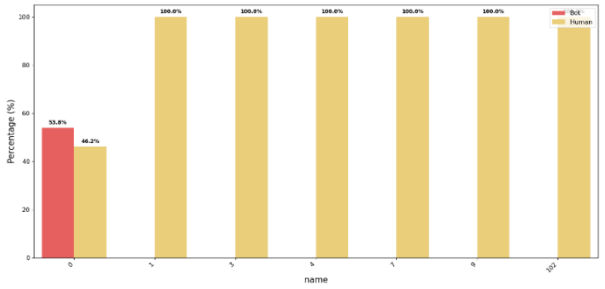

FIGURE 5. Assessment of 'name' feature.

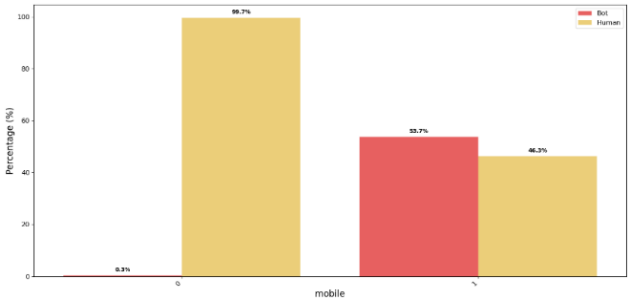

FIGURE 6. Assessment of 'mobile' feature.

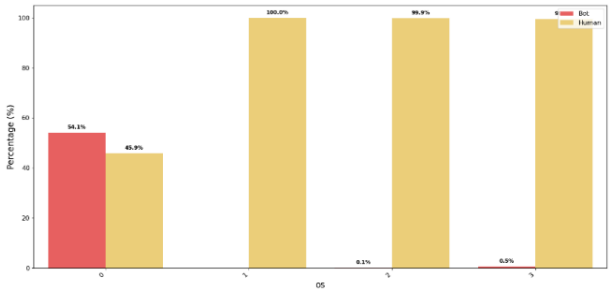

FIGURE 7. Assessment of 'os' feature.

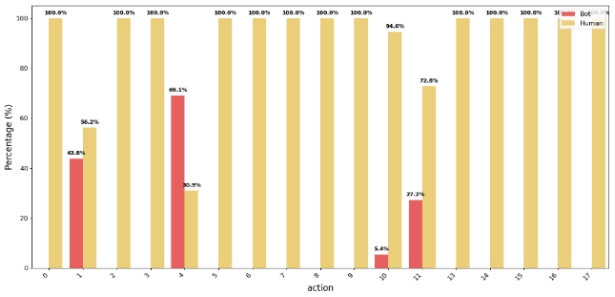

FIGURE 8. Assessment of 'action' feature.

bot interactions and these include 8%7%, 6%.3%75% respectively, as indicated in Figure 9. Examples of such counts with a high percentage of bots interacting are 2215,4525 as well as 6166. We could say that there's an association between automated behavior and high early morning click counts.

2) 'MAX #CLICKS PER 5MIN'

Max #Clicks per 5min is a measure of user behavior, which gives insights into the differences between interactions

involving humans and bots. For instance, when there are no clicks for five minutes, some examples show that all 100% of them are bot interactions, thus suggesting that this feature may be indicative of automated activities. On the other hand, counts 1,2 and 3 show a human-dominated pattern with a range of bot interactions from 21.01% to 25.13%. Higher click counts—5, 9, 10, and 20—show a more evenly distributed distribution of bot and human interactions with varying frequency levels in each class as indicated in Figure 10. Nevertheless, numbers such as 378,316 and 34 have more focused distributions towards human beings where at least 10.22% of them are said to be bots while others can go up to 15.15%.

3) 'GAP INTERVAL BTW CLICKS'

This feature helps to differentiate between human and bot interactions by analyzing user activity based on the gap interval between clicks. Some cases have the largest percentage of bot interaction at 97.28% when there is no delay between clicks, as can be seen in Figure 11, implying automated behavior with rapid click succession. On the other hand, the distribution is more diverse in cases where the gap intervals are 1, 2, 4, and 6 seconds; the bot percentages range from 25% to 71.58%. The distribution of situations with longer gap intervals—20,29 and 38 seconds—is relatively more balanced; thus, it can be a combination of human and bot interactions. Shorter periods may indicate automated fast movement of bots while longer ones may indicate human activity.

4) 'MOUSE_MOVES'

Different patterns are shown in mouse-move analysis of user behavior by the bot class (Target = 0) and human class (Target = 1). The distributions are nearly even when there is no movement in the mouse, with slightly more than half of all interactions being bot interactions (50.41 %) versus human ones (49.59 %). This means that not every automated task involves mouse movements, hence it becomes an important feature to distinguish between these two categories. Human interactions on the other hand tend to have a higher percentage with increasing number of mouse movements. For example, at least some mouse motion happens during human interactions, as indicated by only 18.36% of bots if there was only one move of the mouse. As shown in Figure 12 by the decreasing ratio of bots in relation to the rise in mouse moves.

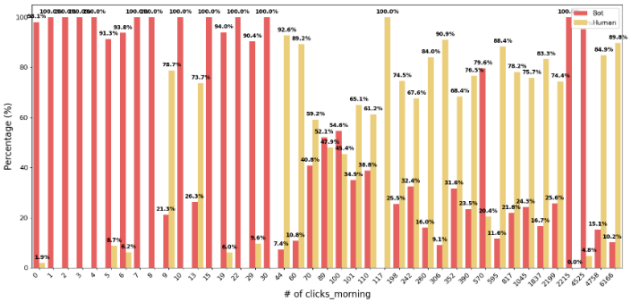

FIGURE 9. Assessment of '# of clicks_morning' feature.

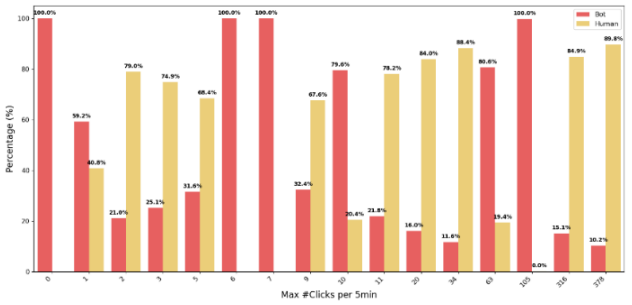

FIGURE 10. Assessment of 'Max #Clicks per 5min'feature.

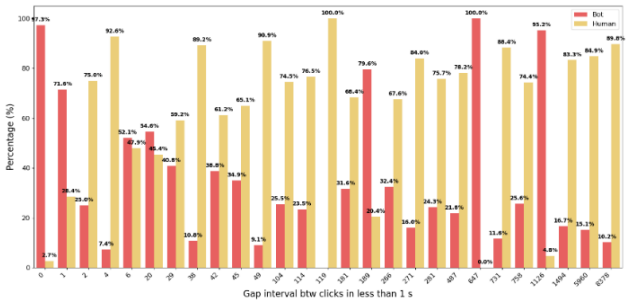

FIGURE 11. Assessment of 'Gap interval btw clicks' feature.

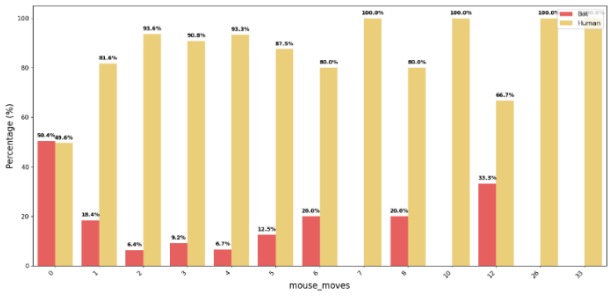

FIGURE 12. Assessment of 'mouse_moves' feature.

VII. ML AND DL TRAINING

In this section, we will outline the workflow approach, beginning with the feature selection phase. We will then provide an overview of the ML and DL models employed and the performance metrics used to assess their accuracy and efficiency.

A. FEATURE SELECTION

This study utilized Recursive Feature Elimination (RFE) to optimize the performance of the click fraud detection model. The RFE method is designed to identify and retain the most valuable features for modeling. It is primarily intended to reduce the risk of overfitting, improve interpretability, and enhance the model's efficiency by eliminating irrelevant and redundant features. In each iteration of the RFE process, the least significant features are eliminated based on model

TABLE 6. Selected features.

NO.	Selected Features
1.	actionsInVisit
2.	landingPage
3.	name
4.	mobile
5.	os
6.	action
7.	% of clicks_morning
8.	% of clicks_night
9.	# of clicks_morning
10.	Max #Clicks per 1min
11.	Max #Clicks per 5min
12.	Max #Clicks per 10 min
13.	Max #Clicks per 1h
14.	Var #Clicks per 10 min
15.	Var #Clicks per 2h
16.	Var #Clicks per 3h
17.	Var #Clicks per 6h
18.	#Clicks per 10 min
19.	#Clicks per 1h
20.	Gap interval btw clicks
21.	Gap interval btw clicks in less than 1 s
22.	Gap interval btw clicks in 1 h
23.	avg_clicks_afternoon
24.	avg_clicks_evening
25.	avg_clicks_morning
26.	avg_clicks_night
27.	STD #click in afternoon
28.	STD #click in evening
29.	STD #click in morning
30.	STD #click in niht
31.	Var #click in evening
32.	Var #click in morning
33.	Var #click in night
34.	#Period_clicks/#clicks in night
35.	#Period_clicks/#clicks in morning
36.	#Period_clicks/#clicks in afternoon
37.	#Period_clicks/#clicks in evening
38.	visit_duration

performance metrics. As a result of recursively removing features and assessing the model's performance at each step, RFE ensures that the final set of selected features maximizes predictive accuracy [34]. Following an RFE to determine the optimal subset of features, we obtained 38 features, as indicated in Table 5, from our initial set of roughly 90 features.

B. ML AND DL MODELS

Python carried out the experiment in Jupyter Notebook. This free, open-source, interactive web-based computing environment enables the user to develop and share documents, including live code, equations, graphics, explanatory text, and more. It's popular in data science, research, and other industries [35].

TABLE 7. Confusion matrix.

		True Class: Positive	True Class: Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
Predicted Class	Negative	False Negative (FN)	True Negative (TN)

1) ML MODELS

Machine Learning (ML) is an artificial intelligence subject that focuses on creating algorithms and statistical models that allow computers to learn from data and enhance their performance on specific tasks. At its foundation, machine learning is the creation of algorithms that can understand patterns and correlations from data without the need for human interaction. The key goal is to create models that generalize effectively to previously unknown data and generate accurate predictions or classifications [36].

2) DL LEARNING

A subset of machine learning techniques known as ''deep learning'' uses multiple-layered artificial neural networks—hence the term ''deep''—to identify and represent complex patterns in data. With the construction of several nonlinear transformations, these algorithms may automatically learn hierarchical data representations [37].

We implement three different DL models to distinguish between human and bot clicks. Moreover, in our study the grid search method selected the best combination of hyperparameters for DL models. As a popular method of hyperparameter tuning, it undertakes an exhaustive search through a predefined hyperparameter space to determine the optimal set of hyperparameters for a particular machine-learning model.

3) PREFORMANCE METRICS

The Confusion Matrix and performance metrics are essential concepts in machine learning and data analysis, notably in evaluating classification algorithms. These techniques assist in quantifying and assessing the efficiency of a model's predictions to determine how well the model performs across different classes. Performance metrics are quantitative measurements used to evaluate the accuracy of a machine learning model's predictions. They give information on the model's accuracy, precision, recall, and other performance indicators.

These measurements may be derived from the confusion matrix shown in Table 7.

Wherein, True Positive (TP): Instances correctly predicted; as positive (correctly classified Bot clicks as Bot clicks),

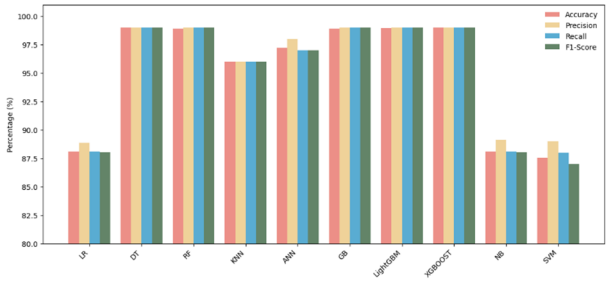

FIGURE 13. Performance of ML models.

False Positive (FP): Instances incorrectly predicted as positive (wrongly classified Human clicks as Bot clicks),

True Negative (TN): Instances correctly predicted as unfavorable (correctly classified Human clicks as Human clicks),

And False Negative (FN): Instances incorrectly predicted as unfavorable (wrongly classified Bot clicks as Human clicks).

VIII. RESULTS AND DISCUSSION

In the context of click fraud detection, it is essential to reduce false negatives to prevent real cases of fraudulent activity from going undetected. False negatives happen when fraudulent clicks are mistakenly categorized as genuine. This can cause advertisers to suffer financial losses as well as damage their reputations. Nevertheless, it's crucial to find a balance between this goal and the possible effects of false positives on business. False positives happen when legitimate clicks are mistakenly reported as fraud, which can cost advertisers real chances to engage with customers while generating revenue. In this regard, we should focus on performance metrics that prioritize both minimizing false negatives and managing the impact of false positives. Key metrics include Precision, Recall, and F1-score. We employed a comprehensive analysis of ML and DL models for click fraud detection, with a particular emphasis on minimizing false negatives due to the critical nature of this problem.

In this study, we employed ten-fold cross-validation to ensure robust model evaluation. The dataset was divided into 10 equal parts, where the model was trained on 9 folds and tested on the remaining fold, repeating this process across all folds. This method was repeated for all ten folds. An accurate assessment of the model's performance was obtained by averaging the final performance metric over the ten iterations. As illustrated in Table 8, Figure 13, and Figure 14, promising outcomes were found in the assessment of ML models, which included DT, RF, KNN, ANN, GB, LightGBM, NB, and SVM. All models showed good accuracy; DT, RF, and GB frequently achieved accuracy rates higher than 98%. However, given the nature of the task, it becomes imperative to concentrate on precision, recall, and F1-score.

The ML models performed better after feature selection using RFE. Notably, even with fewer features, DT, RF, KNN, ANN, GB, LightGBM, and XGBoost all maintained high accuracy levels, highlighting the models' robustness. The refinement also improved precision, recall, and F1-score, with DT and RF reaching 99% precision and recall. KNN showed remarkable stability, preserving at 96% a fair trade-off between recall and precision.

TABLE 8. Results OF ML and DL models.

Model	Accuracy	Precision	Recall	F1-score
ML models:				
LR	88.10%	88.86%	88.10%	88.04%
DT	**98.99%**	**99.00%**	**99.00%**	**99.00%**
RF	98.90%	99.00%	99.00%	99.00%
KNN	95.98%	96.00%	96.00%	96.00%
ANN	97.23%	98.00%	97.00%	97.00%
GB	98.90%	99.00%	99.00%	99.00%
LightGBM	98.94%	99.00%	99.00%	99.00%
XGBOOST	98.97%	98.98%	98.97%	98.97%
NB	88.10%	89.11%	88.10%	88.02%
SVM	87.55%	89.00s%	88.00%	87.00%
DL models:				
CNN	94.09%	91.08%	97.75%	94.30%
DNN	91.34%	89.06%	94.30%	91.58%
RNN	**97.34%**	**96.64%**	**98.16%**	**97.36%**

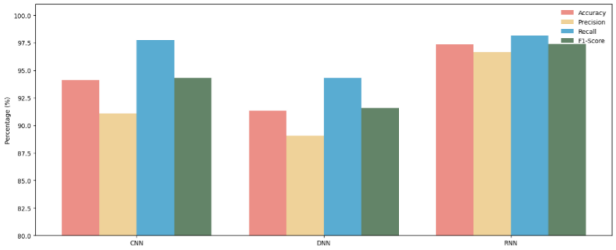

FIGURE 14. Performance of DL models.

Turning our attention to the DL models, namely CNN, DNN, and RNN, we observed competitive results. CNN demonstrated a balanced precision and recall, making it an effective model for click fraud detection, with an accuracy rate of 94.54%. Although its accuracy was marginally lower at 88.65%, DNN showed good precision and recall scores, offering a different choice within the DL framework. With an accuracy of 90.74%, RNN performed exceptionally well in precision, recall, and F1-score. Overall, the comparison between ML and DL models after RFE highlights the efficacy of feature selection in enhancing model performance. While ML models consistently maintained high accuracy rates, DL models demonstrated their proficiency in handling complex patterns and relationships within the data. Ultimately, the decision between ML and DL models is made based on the particular requirements of the click fraud detection

task, taking interpretability and computational resources into account. Whether or not RFE is present, DT and RF are excellent options when minimizing false negatives is crucial. On the other hand, DL models—particularly RNN—show considerable promise for tasks requiring a thorough comprehension of complex patterns. This comprehensive assessment provides valuable insights for practitioners aiming to deploy effective click fraud detection systems tailored to their needs.

However, it is very important to consider trade-offs between performance and computational complexity, especially with real-time applications which involve large amounts of data to analyze. Although DL models improve predictive performance, they require a lot of computational resources for training and inference due to several reasons, such as large parameter sizes and sequential operations. On the other hand, ML models such as DT remain computationally efficient compared to DL in limited resource and latency-contingent environments. Ideally, the selection of a real-time fraud detection system model depends on the application-specific requirements regarding achieving the desired balance between detection accuracy and constraints of computational resources and latency.

IX. COMPARISON WITH PREVIOUS STUDIES

A comparison of the results of our study with some previous studies using ML is presented in this section, as well as comparisons with the results of prior studies that utilized DL. More specifically, we will compare the best-performing model in our research with that in previous studies, as shown in Table 9 and Table 10. We selected previous studies related to the detection of click fraud, which had occurred within the last four years, and used the same performance measures as those used in this study.

Our study on click fraud detection using Decision Trees (DT) has yielded promising results. Compared to previous ML research, our survey on click fraud detection using DT has produced promising findings. A thorough comparison with previous research indicates that our approach effectively prevents fraudulent activity. Using the CFXGB model, Thejas et al.'s study [22] produced an accuracy of 94.53%, precision, recall, and F1-score of 94.00%. Leveraging Random Forest (RF), Aljabri and Mohammad [15] reported an accuracy of 84.00% with 84.00% precision and recall. Chari et al. [23] employed Logistic Regression (LR) and achieved a 98.69% accuracy rate and 99.00% precision, recall, and F1-score. Additionally, Viruthika et al. [24] utilized XGBoost and achieved an accuracy of 91.00%, with precision, recall, and F1-score all at 91.00%. In our study, using DT, we achieved an accuracy rate of 98.99% and precision, recall, and F1 scores of 99.0%. These findings highlight the effectiveness of our approach, and the features used, which demonstrate its capability to outperform existing ML models in accurately detecting fraudulent clicks. Through the use of DT and its inherent advantages, such as interpretability and scalability, our study contributes significantly to the ongoing effort in the domain of click fraud detection.

Regarding DL models, our investigation delves into the performance of RNN for detecting click fraud in online advertising. Comparing our findings with prior DL studies yields valuable insights. An RNN accuracy of 96.31%, for example, was reported by Chari et al. [23], who also reported high precision, recall, and F1-score metrics (93.00%, 96.00%, and 95.00%, respectively). A Cost-Sensitive CNN with 93.00% accuracy, 89.00% precision, 92.00% recall, and F1-score, respectively, was introduced by Liu et al. [25]. In addition, Gabryel et al. [26] presented a Weighted Multi-Layer Network Model that consistently achieved 98.60% accuracy and 98.60% precision, recall, and F1-score measures. Our RNN-based solution obtained similar results: accuracy of 97.34%, precision, recall, and F1-score of 96.64%, 98.16%, and 97.36%, respectively. These findings underscore the effectiveness of DL techniques, particularly RNN, in detecting fraudulent click activities. These results could be justified due to several reasons, for instance, RNN shows better adaptation in handling temporal characteristics within user behavioral data mainly, concerning mouse movements, time spent scrolling, and clicking patterns. While traditional ML models work independently with the features that we introduced, RNNs are capable of understanding the temporal dependencies that enable one to distinguish between humans and bots.2- LSTM layers help the model to have information about previous user actions while at the same time analyzing the current actions movements, scrolling durations, and click sequences. In contrast to conventional ML models that consider each feature in isolation, RNNs possess the ability to learn temporal dependencies that are indicative of distinguishing between human and bot behavior. The LSTM layers facilitate the model's ability to retain information regarding prior user actions while at the same time examining current behavior. This capability makes it possible to establish narrow behavioural distinctions separating the real clicks from the fake ones. Further, the RNN can self-learn complex interactions between temporal features without the need of sophisticated feature extraction. The architecture of the RNN used in this study was carefully optimized to achieve the best performance. The final model utilized a single-layer RNN with 32 units and employed the Adam optimizer for efficient gradient-based optimization. To prevent overfitting, a dropout rate of 0.2 was applied to the input connections. The model was trained using a batch size of 64 over 20 epochs, balancing computational efficiency and convergence.

Overall, our study highlights the potential of ML and DL techniques, particularly DT and RNN, in effectively addressing the challenges posed by fraudulent clicks in the advertisement domain, thus paving the way for more robust and reliable fraud detection mechanisms in the future.

We have previously covered and analyzed the datasets used in the studies that we compared ours with as the commonly used public datasets in the field of click fraud detection [24]. A further point to note is that the dataset used in [14] was provided by The Veracity Trust Network (previously known as Beacon), the same company that provided us with the dataset

TABLE 9. Comparison with ML models in previous studies.

Model	Dataset	Accuracy	Precision	Recall	F1-score
CFXGB (Thejas et al. [38])	TalkingData AdTracking Dataset	94.53%	94.00%	94.00%	94.00%
RF (Aljabri and Mohammad [14])	Private Dataset	84.00%	87.00%	84.00%	84.00%
LR (Chari et al. [39])	The Veracity Trust Network Dataset	98.69%	99.00%	99.00%	99.00%
XGBoost (Viruthika et al. [40])	Private Dataset	91.00%	91.00%	91.00%	91.00%
DT (Our study)	The Veracity Trust Network Dataset	**98.99%**	**99.00%**	**99.00%**	**99.00%**

TABLE 10. Comparison with DL models in previous studies.

Model	Dataset	Accuracy	Precision	Recall	F1-score
RNN (Chari et al. [39])	TalkingData AdTracking Dataset	96.31%	93.00%	96.00%	95.00%
Cost-Sensitive CNN (Liu et al. [41])	FDMA 2012 BuzzCity Dataset	93.00%	89.00%	93.00%	92.00%
RNN (Our study)	The Veracity Trust Network Dataset	**97.34%**	**96.64%**	**98.16%**	**97.36%**

included in this study. In addition to employing the RNN for creating the classification model, a major contribution of this study is the unique real-life dataset used for developing the intelligent models. While our dataset is much larger in volume and contains more in-depth features, we have also engineered additional reliable information to improve our detection capability. The novel dataset employed by this research enabled us to examine a wide range of critical aspects regarding users' journeys across various websites. These characteristics were carefully analyzed to determine how they relate to different phases of the web journey. For instance, temporal features were exhaustively investigated, leading to the identification and understanding of time-dependent variations in users' patterns/habits during different hours of the day. Additionally, an in-depth examination of temporal features was undertaken, which enabled the exploration and understanding of user patterns and click activities during different times of the day. Moreover, a precise analysis of click behavior was conducted, including the calculation of time intervals between clicks across various time intervals and the computation of standard deviations, variances, and other pertinent metrics of click activity over a specified period of time. We were able to obtain excellent and promising results, enhancing our understanding of user behavior and click activities in online environments.

X. CONCLUSION

In the landscape of online advertising, the persistent threat of click fraud looms large, undermining the industry's integrity and siphoning funds away from legitimate advertisers. Our study, aimed at addressing this challenge, embarked on a comprehensive exploration of feature engineering and extraction techniques to discern subtle nuances in click behavior, crucial for distinguishing between fraudulent and genuine clicks. We meticulously evaluated nine ML and three DL models to identify the most effective approaches in combating click fraud.

Our findings underscore the robust performance of ML models, particularly after RFE. Notably, DT and RF models surpassed 98.99% accuracy, while GB, LightGBM, and XGBoost achieved accuracy rates of 98.90% or higher. Impressively, models such as ANN exhibited precision scores exceeding 98%, indicating their adeptness at accurately identifying fraudulent clicks. Simultaneously, our exploration into DL models revealed promising results, with CNN, Deep DNN, and RNN demonstrating robust performance. RNN, in particular, achieved an accuracy of 97.34%, highlighting its efficacy in click fraud detection.

Yet, every research has some limitations that motivate further research, and our study is not an exception. However, looking ahead, our findings pave the way for future research directions in ad click fraud detection. These findings are considered the starting point for further improving our approach proposed in this study. A possible future work is by refine and optimize ML and DL models so that we will be able to enhance detection capabilities and reduce the impact of fraudulent clicks. Moreover, in order to develop multi-level security mechanism, advertisers and cybersecurity experts could be able to use the proposed fraud detection model on both the server and client sides. The model enables real-time tracking and analysis of click patterns across all ad campaigns and may be server-side linked into the ad platform's backend. This would then be handled on the server-side setup using incoming click data, which would then be passed through the model's algorithms to identify any anomalies and immediately flag any possible fraudulent activity. To supplement the server-side analysis and insights, additional information could be obtained from the client side by monitoring the browser or application. In order to provide a much higher level of detail on user activity, the scripts can then collect metadata such as user interaction, device

TABLE 11. Extracted temporal features.

No.	Feature	Description
1.	%Clicks in morning %Clicks in afternoon %Clicks in evening %Clicks in night	The percentage of clicks for each *'host'* across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM).
2.	%Clicks in morning _0/1 %Clicks in afternoon _0/1 %Clicks in evening _0/1 %Clicks in night'_0/1	The percentage of clicks for each *'host'* across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM). Calculated for each class 1 (Human) and 0 (Bot).
3.	#Clicks in morning #Clicks in afternoon #Clicks in evening #Clicks in night	The total number of clicks for each *'host'* across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM).
4.	#Clicks in morning _0/1 #Clicks in afternoon _0/1 #Clicks in evening _0/1 #Clicks in night _0/1	The total number of clicks for each *'host'* across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM). Calculated for each class 1 (Human) and 0 (Bot).
5.	Avg #Clicks per 1 min Avg #Clicks per 5 min Avg #Clicks per 10 min Avg #Clicks per 1 h Avg #Clicks per 2 h Avg #Clicks per 3 h Max #Clicks per 6 h	The average click count for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours.
6.	Avg #Clicks per 1 min _0/1 Avg #Clicks per 5 min _0/1 Avg #Clicks per 10 min _0/1 Avg #Clicks per 1 h _0/1 Avg #Clicks per 2 h _0/1 Avg #Clicks per 3 h _0/1 Max #Clicks per 6 h_0/1	The average click count for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours. Calculated for each class 1 (Human) and 0 (Bot).
7.	Max #Clicks per 1 min, Max #Clicks per 5 min, Max #Clicks per 10 min, Max #Clicks per 1 h, Avg #Clicks per 2 h, Max #Clicks per 3 h Max #Clicks per 6 h	The maximum number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours.
8.	Max #Clicks per 1 min _0/1, Max #Clicks per 5 min _0/1, Max #Clicks per 10 min _0/1, Max #Clicks per 1 h _0/1, Avg #Clicks per 2 h _0/1, Max #Clicks per 3 h _0/1 Max #Clicks per 6 h _0/1	The maximum number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours. Calculated for each class 1 (Human) and 0 (Bot).
9.	Var #Clicks per 1 min, Var #Clicks per 5 min, Var #Clicks per 10 min, Var #Clicks per 1 h, Var #Clicks per 2 h, Var #Clicks per 3 h Var #Clicks per 6 h	The variance number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours.
10.	Var #Clicks per 1 min _0/1, Var #Clicks per 5 min _0/1, Var #Clicks per 10 min 0/1, Var #Clicks per 1 h, Var #Clicks per 2 h _0/1, Var #Clicks per 3 h _0/1 Var #Clicks per 6 h _0/1	The variance number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours. Calculated for each class 1 (Human) and 0 (Bot).
11.	Skew #Clicks per 1 min, Skew #Clicks per 5 min, Skew #Clicks per 10 min, Skew #Clicks per 1 h, Skew #Clicks per 2 h, Skew #Clicks per 3 h Skew #Clicks per 6 h	The skewness number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours.
12.	Skew #Clicks per 1 min_0/1, Skew #Clicks per 5 min _0/1, Skew #Clicks per 10 min, Skew #Clicks per 1 h _0/1, Skew #Clicks per 2 h _0/1, Skew #Clicks per 3 h _0/1 Skew #Clicks per 6 h _0/1	The skewness number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours. Calculated for each class 1 (Human) and 0 (Bot).
13.	#Clicks per 1 min, #Clicks per 5 min, #Clicks per 10 min,	The total number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes,

TABLE 11. *(Continued.)* **Extracted temporal features.**

	#Clicks per 1 h, #Clicks per 2 h, #Clicks per 3 h #Clicks per 6 h	10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours.
14.	#Clicks per 1 min _0/1, #Clicks per 5 min _0/1, #Clicks per 10 min _0/1, #Clicks per 1 h _0/1, #Clicks per 2 h _0/1, #Clicks per 3 h _0/1 #Clicks per 6 h _0/1	The total number of clicks for each *'host'* is measured within time spans of 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, and 6 hours. Calculated for each class 1 (Human) and 0 (Bot).
15.	Gap interval btw clicks	The difference in time between two consecutive clicks for each 'host.'
16.	Gap interval btw clicks in 1 h Gap interval btw clicks in less than 1 s Gap interval btw clicks in from 1 s to 1 min Gap interval btw clicks in over 1 min	The difference in time between two consecutive clicks for each *'host'* in 1 hour, less than 1 second, from 1 second to 1 minute, and over 1 minute.
17.	Avg #click in morning Avg #click in afternoon Avg #click in evening Avg #click in night'	Average total number of clicks for each 'host' across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM).
18.	STD #click in morning, STD #click in afternoon, STD #click in evening, STD #click in night'	The standard deviation of the total number of clicks for each 'host' across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM).
19.	Var #click in morning Var #click in afternoon Var #click in evening Var #click in night	The variance of the total number of clicks for each 'host' across different times of the day: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to 12 PM), and night (12 PM to 6 AM).
20.	#Period_clicks/#clicks in morning #Period_clicks/#clicks in afternoon #Period_clicks/#clicks in evening #Period_clicks/#clicks in night	Total number of clicks generated for each *'host'* throughout four time periods: morning (6 AM to 12 PM), afternoon (12 PM to 6 PM), evening (6 PM to12 PM), and night (12 PM to 6 AM) divided by the overall number of clicks for that specific *'host.'*

TABLE 11. *(Continued.)* **Extracted temporal features.**

21.	Unique IP at 1 min Unique IP at 5 min Unique IP at 10 min Unique IP at 1 h Unique IP at 2 h Unique IP at 3 h Unique IP at 6 h Unique IP at 10 h	Number of distinct *'IPv4 'addresses'* clicks per *'host'* in 1 minute, 5 minutes, 10 minutes, 1 hour, 2 hours, 3 hours, 6 hours, and 10 hours.

TABLE 12. **Extracted behavioral features.**

No.	Feature	Description
1.	Time delta	The time spent by the user while moving the mouse.
2.	Mouse moves	The number of mouse moves.
3.	Up count	The count of mouse movements was conducted in an upward direction.
4.	Right_up count	The count of mouse movements was conducted in the upper-right direction.
5.	Right count	The count of mouse movements was conducted in the right direction.
6.	Right_down count	The count of mouse movements was conducted in the lower-right direction.
7.	Down count	The count of mouse movements was conducted in the downward direction.
8.	Left_down count	The count of mouse movements was conducted in the lower left direction.
9.	Left count	The count of mouse movements was conducted in the left direction.
10.	Left_up count	The count of mouse movements was conducted in the upper left direction.
11.	Not cardinal count	The count of mouse movements conducted made in directions other than those mentioned above.
12.	Visit Duration	The amount of time the user spent on the visit (the difference between the time the visit started and ended).

fingerprinting, and session behaviour. This client-side data may then be compared to server-side results to validate the legitimacy of clicks and identify sophisticated fraud patterns that could otherwise go undetected. By taking these aspects into account, an effective proactive fraud detection solution for advertisers is provided by server-side and client-side implementations together. Online learning or incremental learning which allows a periodic updating of the model and adaptiveness in learning from real-time data will further enhance the model to continue evolving toward emerging threats. This dual approach would guarantee that advertisers' investments are adequately safeguarded while creating a more transparent and resilient digital advertising environment.

TABLE 13. Original values/ranges of the assessed features.

Encoded Label/Bin	Original Value / Range
'pagesViewedInVisit'	
0	0 - 14
1	15 - 29
2	30 - 44
3	45 - 59
4	60 - 74
5	75 - 89
6	90 - 104
7	105 - 106
'actionInVisit'	
0	0 - 72
1	73 - 145
2	146 - 218
3	219 - 291
4	292 - 364
5	365 - 437
6	438 - 510
7	511 - 583
8	584 - 656
9	657 - 729
10	730 - 802
11	803 - 875
12	876 - 948
13	949 - 1021
14	1022 - 1094
15	1095 - 1167
16	1168 - 1240
17	1241 - 1313
18	1314 - 1386
19	1387 - 1459
20	1460 - 1532
21	1533 - 1605
22	1606 - 1678
23	1679 - 1751
24	1752 - 1824
25	1825 - 1897
26	1898 - 1970
27	1971 - 1989
'name'	
0	Chrome
1	Safari
3	Firefox
4	Klarna
9	Netscape
102	OPT
7	FBAN
'mobile'	
0	False (non-mobile)
1	True (mobile)
'os'	
0	Android
1	Linux
2	Mac OS X
3	Windows
'action'	
0	anchor
1	Click on element
2	download
3	external_link clicked
4	first_pass_completed - Initial step successfully done
5	form_submit
6	hit_bottom
7	hit_top
8	input_specified
9	internal_link
10	mouse_updates - Mouse position updated
11	scroll_down
12	scroll_up
13	Tel - Click on phone link
14	video_auto_play
15	video_buffering
16	video_pause
17	video_seeked
'mouse_moves'	
0	0 - 25
1	26 - 51
2	52 - 77
3	78 - 103
4	104 - 129
5	130 - 155
6	156 - 181
7	182 - 207
8	208 - 233
9	234 - 259
10	260 - 285
11	286 - 311
12	312 - 337
13	338 - 363
14	364 - 389
15	390 - 415
16	416 - 441
17	442 - 467
18	468 - 493
19	494 - 519
20	520 - 545
21	546 - 571
22	572 - 597
23	598 - 623
24	624 - 649
25	650 - 675
26	676 - 701
27	702 - 727

Additionally, more efforts are needed to investigate advanced fraud patterns, including adversarial attacks and evolving strategies, to enhance models' robustness. Incorporating unsupervised methods especially in case of limited

labeled data, including anomaly detection and clustering algorithms. Moreover, hybrid models that combine supervised and unsupervised techniques may achieve a balance between accuracy and adaptability in practice. Attention also must be focused on real-world challenges like imbalanced data, latency, and scalability for deployment in various environments.

APPENDIX

The tables below show the extracted features from our dataset with their descriptions across the feature structure we organized above.

A. TEMPORAL FEATURES

See Table 11.

B. CLICKER BEHAVIOUR

See Tables 12 and 13.

ACKNOWLEDGMENT

The authors express their sincere appreciation to Veracity Trust Network for graciously providing the data pivotal to this study, profoundly enhancing the depth and quality of the research. Their invaluable contribution has played a critical role in shaping this investigation's outcomes and conclusions.

The authors of this paper point out that they have used AI tools only to rephrase some parts of it.

They would like to thank SAUDI ARAMCO Cybersecurity Chair for funding this project.

AVAILABILITY OF DATA AND MATERIALS

The data that support the findings of this study are available on request from the corresponding author.

COMPETING INTEREST

The authors declare that they have no competing financial or non-financial interests.

AUTHORS' CONTRIBUTIONS

The authors declare equal contributions. The authors read and approved the final manuscript.

REFERENCES

[1] Juniper Research, Hampshire, U.K. *Quantifying the Cost of Ad Fraud: 2023–2028*. Accessed: Jul. 12, 2024. [Online]. Available: https://fraudblocker.com/wp-content/uploads/2023/09/Ad-Fraud-Whitepaper_Juniper-Research.pdf

[2] X. Zhu, H. Tao, Z. Wu, J. Cao, K. Kalish, and J. Kayne, *Fraud Prevention in Online Digital Advertising*. Cham, Switzerland: Springer, 2017.

[3] A. K. Wood and A. M. Ravel, "Fool me once: Regulating fake news and other online advertising," *S. Cal. L. Rev.*, vol. 91, p. 1223, Jan. 2017.

[4] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna, "Understanding fraudulent activities in online ad exchanges," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf.*, Nov. 2011, pp. 279–294.

[5] (2024). *Wasted Ad Spend Report 2024*. [Online]. Available: https://lp.lunio.ai/wp-content/uploads/2023/09/Lunio_Wasted_Ad_Spend_Report_2024_V2.pdf

[6] D. Berrar, "Random forests for the detection of click fraud in online mobile advertising," in *Proc. Int. Work. Fraud Detect. Mob. Advert. (FDMA)*, Singapore, 2012, pp. 1–10. [Online]. Available: http://berrar.com/resources/Berrar_FDMA2012.pdf

[7] J. H. Yan and W. R. Jiang, "Research on information technology with detecting the fraudulent clicks using classification method," *Adv. Mater. Res.*, vol. 859, pp. 586–590, Dec. 2013, doi: 10.4028/www.scientific.net/amr.859.586.

[8] K. S. Perera, B. Neupane, M. A. Faisal, Z. Aung, and W. L. Woon, "A novel ensemble learning-based approach for click fraud detection in mobile advertising," in *Mining Intelligence and Knowledge Exploration* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8284. Berlin, Germany: Springer, 2013, pp. 370–382, doi: 10.1007/978-3-319-03844-5_38.

[9] C. Phua, E.-Y. Cheu, G.-E. Yap, K. Sim, and M.-N. Nguyen, "Feature engineering for click fraud detection," in *Proc. Work. Fraud Detect. Mob. Advert.*, 2012, pp. 1–10. [Online]. Available: http://palanteer.sis.smu.edu.sg/fdma2012/doc/FirstWinner-Starrystarrynight-Paper.pdf%5Cnpapers2://publication/uuid/9290A6CF-A861-4058-99F4-D39706B0619A

[10] E.-A. Minastireanu and G. Mesnita, "Light GBM machine learning algorithm to online click fraud detection," *J. Inf. Assurance Cybersecur.*, vol. 2019, pp. 1–12, Apr. 2019, doi: 10.5171/2019.263928.

[11] D. Sisodia and D. S. Sisodia, "Gradient boosting learning for fraudulent publisher detection in online advertising," *Data Technol. Appl.*, vol. 55, no. 2, pp. 216–232, Apr. 2021, doi: 10.1108/dta-04-2020-0093.

[12] A. Dash and S. Pal, "Auto-detection of click-frauds using machine learning Auto-detection of click-frauds using machine learning," *Int. J. Eng. Sci. Comput.*, vol. 10, pp. 27227–27235, Sep. 2020.

[13] R. Mouawi, M. Awad, A. Chehab, I. H. E. Hajj, and A. Kayssi, "Towards a machine learning approach for detecting click fraud in mobile advertizing," in *Proc. Int. Conf. Innov. Inf. Technol. (IIT)*, Nov. 2018, pp. 88–92, doi: 10.1109/INNOVATIONS.2018.8605973.

[14] M. Aljabri and R. M. A. Mohammad, "Click fraud detection for online advertising using machine learning," *Egyptian Informat. J.*, vol. 24, no. 2, pp. 341–350, Jul. 2023, doi: 10.1016/j.eij.2023.05.006.

[15] S. Shaik and V. Kakulapati, "Fraud detection of AD clicks using machine learning techniques," *J. Sci. Res. Rep.*, vol. 29, no. 7, pp. 84–89, Jun. 2023, doi: 10.9734/jsrr/2023/v29i71762.

[16] D. Sisodia and D. S. Sisodia, "Stacked generalization architecture for predicting publisher behaviour from highly imbalanced user-click data set for click fraud detection," *New Gener. Comput.*, vol. 41, no. 3, pp. 581–606, Sep. 2023, doi: 10.1007/s00354-023-00218-1.

[17] D. Sisodia, D. S. Sisodia, and D. Singh, "Evaluating feature importance to investigate publishers conduct for detecting click fraud," in *Machine Intelligence Techniques for Data Analysis and Signal Processing* (Lecture Notes in Electrical Engineering), vol. 997. Berlin, Germany: Springer, 2023, pp. 515–524, doi: 10.1007/978-981-99-0085-5_42.

[18] R. Dekou, S. Savo, S. Kufeld, D. Francesca, and R. Kawase, "Machine learning methods for detecting fraud in online marketplaces," in *Proc. CEUR Workshop*, vol. 3052, Jan. 2021, pp. 3–7.

[19] D. Sisodia and D. S. Sisodia, "Quad division prototype selection-based K-nearest neighbor classifier for click fraud detection from highly skewed user click dataset," *Eng. Sci. Technol., Int. J.*, vol. 28, Apr. 2022, Art. no. 101011, doi: 10.1016/j.jestch.2021.05.015.

[20] B. Kirkwood, M. Vanamala, and N. Seliya, "Click fraud detection of online advertising using machine learning algorithms," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (eIT)*, May 2024, pp. 586–590. [Online]. Available: https://api.semanticscholar.org/CorpusID

[21] L. Singh, D. Sisodia, K. Shashvat, A. Kaur, and P. C. Sharma, "A reliable click-fraud detection system for the investigation of fraudulent publishers in online advertising," in *Applied Intelligence in Human-Computer Interaction*. Boca Raton, FL, USA: CRC Press, Jul. 2023.

[22] A. Batool and Y.-C. Byun, "An ensemble architecture based on deep learning model for click fraud detection in Pay-Per-Click advertisement campaign," *IEEE Access*, vol. 10, pp. 113410–113426, 2022, doi: 10.1109/ACCESS.2022.3211528.

[23] A. Purwar, A. K. Jain, I. Chawla, I. Gupta, M. Raj, and D. Jain, "Click fraud detection using ensemble classifier," in *Proc. Int. Conf. Artif.-Bus. Anal., Quantum Mach. Learn.*, Jan. 2024, pp. 15–23.

[24] R. A. Alzahrani and M. Aljabri, "AI-based techniques for ad click fraud detection and prevention: Review and research directions," *J. Sensor Actuator Netw.*, vol. 12, no. 1, p. 4, Dec. 2022, doi: 10.3390/jsan12010004.

[25] Veracity Trust Network. *Veracity Trust Network—Only Humans*. Accessed: Feb. 4, 2023. [Online]. Available: https://veracitytrustnetwork.com/
[26] K. Mehrabani-Zeinabad, M. Doostfatemeh, and S. M. T. Ayatollahi, "An efficient and effective model to handle missing data in classification," *BioMed Res. Int.*, vol. 2020, pp. 1–2, 2020, doi: 10.1155/2020/8810143.
[27] J. D. Kelleher, B. Mac Namee, and A. D'arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA, USA: MIT Press, 2020.
[28] D. J. Stekhoven and P. Bühlmann, "MissForest—Non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, Jan. 2012, doi: 10.1093/bioinformatics/btr597.
[29] S. Hong and H. S. Lynn, "Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction," *BMC Med. Res. Methodol.*, vol. 20, no. 1, pp. 1–12, Dec. 2020, doi: 10.1186/s12874-020-01080-1.
[30] A. K. Waljee, A. Mukherjee, A. G. Singal, Y. Zhang, J. Warren, U. Balis, J. Marrero, J. Zhu, and P. D. Higgins, "Comparison of imputation methods for missing laboratory data in medicine," *BMJ Open*, vol. 3, no. 8, Aug. 2013, Art. no. e002847, doi: 10.1136/bmjopen-2013-002847.
[31] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017, doi: 10.1016/j.eswa.2016.12.035.
[32] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, Nov. 2002, doi: 10.3233/ida-2002-6504.
[33] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L_2 theory," *Zeitschrift Für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, Dec. 1981, doi: 10.1007/bf01025868.
[34] H. A. Le Thi, V. V. Nguyen, and S. Ouchani, "Gene selection for cancer classification using DCA," in *Advanced Data Mining and Applications* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5139. Berlin, Germany: Springer, 2008, pp. 62–72, doi: 10.1007/978-3-540-88192-6_8.
[35] B. M. Randles, I. V. Pasquetto, M. S. Golshan, and C. L. Borgman, "Using the Jupyter notebook as a tool for open science: An empirical study," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries (JCDL)*, Jun. 2017, pp. 1–2, doi: 10.1109/JCDL.2017.7991618.
[36] A. Kumbhar, P. G. Dhawale, S. Kumbhar, U. Patil, and P. Magdum, "A comprehensive review: Machine learning and its application in integrated power system," *Energy Rep.*, vol. 7, pp. 5467–5474, Nov. 2021, doi: 10.1016/j.egyr.2021.08.133.
[37] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.
[38] G. S. Thejas, S. Dheeshjith, S. S. Iyengar, N. R. Sunitha, and P. Badrinath, "A hybrid and effective learning approach for click fraud detection," *Mach. Learn. With Appl.*, vol. 3, Mar. 2021, Art. no. 100016, doi: 10.1016/j.mlwa.2020.100016.
[39] H. Chari, S. Aswale, V. N. Pawar, P. Shetgaonkar, and K. M. C. Kumar, "Advertisement click fraud detection using machine learning techniques," in *Proc. Int. Conf. Technol. Advancements Innov. (ICTAI)*, Nov. 2021, pp. 109–114.
[40] B. Viruthika, S. S. Das, E. Manishkumar, and D. Prabhu, "Detection of advertisement click fraud using machine learning," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 5, pp. 3238–3245, 2020, doi: 10.13140/RG.2.2.23528.90881.
[41] *A Click Fraud Detection Scheme Based on Cost-Sensitive CNN and Feature Matrix*, vol. 1210. Google, Mountain View, CA, USA, 2020.

REEM A. ALZAHRANI received the bachelor's degree (Hons.) in computer science and the master's degree in big data and cloud computing from Imam Abdulrahman Bin Faisal University. She is currently a Teaching Assistant with the College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University. Her research interests include AI-based techniques, large language models (LLMs), applications in machine learning, data analysis, and intelligent systems.

MALAK ALJABRI received the B.Sc. degree (Hons.) in computer science from Umm Al-Qura University, Makkah, Saudi Arabia, in 2006, the M.Sc. degree (Hons.) in advanced internet applications from Heriot-Watt University, U.K., in 2010, and the Ph.D. degree in computer science from the University of Glasgow, U.K., in 2015. She is currently an Associate Professor with the Department of Computer and Network Engineering, College of Computing, Umm Al-Qura University. She has published several scientific papers in journals and ACM/IEEE/Elsevier/Springer conferences. Her research interests include parallel computing and systems, blockchain applications, and AI-based techniques in cybersecurity.

RAMI MUSTAFA A. MOHAMMAD is currently an Associate Professor with the College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University. His research interests include information security, digital forensics, data mining, machine learning, and web security. He has made notable contributions to the development of intelligent techniques for phishing website detection, intrusion detection systems, feature selection algorithms, and the application of machine learning to various security-related problems. Additionally, his research has addressed emerging challenges in domains, such as blockchain, cloud computing, and the Internet of Things. His research excellence has been widely recognized. For the past four years, he was among the Top Cited Researchers in his college. He has been enlisted in the prestigious "World Top 2% Scientists" list by Stanford University for two consecutive years, 2022 and 2023. These accolades underscore the profound impact and recognition of his scholarly work within the academic community.