Ingénierie des Systèmes d'Information
Vol. 31, No. 10, October, 2026, pp. 3581-3595
Journal homepage: http://iieta.org/journals/isi

QDPSKNN-Based Undersampling with XGBoost for Highly Imbalanced Click-Attribution Classification

Kevin[1], Jansen Wiratama[2*], Aditiya Hermawan[1], Gredion Prajena[3]

[1] Department of Informatics Engineering, Buddhi Dharma University, Tangerang 15115, Indonesia
[2] Department of Information Systems, Universitas Multimedia Nusantara, Tangerang 15811, Indonesia
[3] Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia

Corresponding Author Email: jansen.wiratama@umn.ac.id

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/isi.311028

Received: 26 May 2026
Revised: 30 August 2026
Accepted: 15 September 2026
Available online: 31 October 2026

Keywords:
click-attribution classification, class imbalance, prototype-based undersampling, QDPSKNN, XGBoost, pay-per-click advertising

ABSTRACT

The detection of suspicious activity in pay-per-click (PPC) advertising is complicated by highly imbalanced click records and by the limited availability of reliable fraud labels. This study evaluates an existing prototype-based undersampling method, Quad Division Prototype Selection with K-Nearest Neighbor (QDPSKNN), in combination with an optimized Extreme Gradient Boosting (XGBoost) classifier for classifying click-attribution outcomes. A 100,000-record subset of the TalkingData AdTracking Fraud Detection dataset was used, comprising 99,773 non-attributed clicks and 227 attributed clicks (an imbalance ratio of approximately 439.5:1). The evaluation compared QDPSKNN with Random Undersampling (RUS), NearMiss-3, Synthetic Minority Oversampling Technique (SMOTE), and Adaptive Synthetic Sampling (ADASYN), using XGBoost, Random Forest (RF), Support Vector Machine (SVM), and Light Gradient Boosting Machine (LightGBM) classifiers. Hyperparameters were selected through RandomizedSearchCV with 20 sampled configurations and stratified five-fold cross-validation. Under the study's 70:30 record-level split and resampling protocol, QDPSKNN–XGBoost achieved 96.77% accuracy, 96.91% precision, 99.30% recall, an F1-score of 98.09%, and a receiver operating characteristic (ROC) curve with an area under the curve (AUC) of 0.98. The positive class for the reported precision, recall, and F1-score was label 0 (non-attributed clicks). The results indicate that QDPSKNN–XGBoost performed better overall than the evaluated sampling–classifier combinations in this experimental setting. However, the dataset's attribution label does not establish fraudulent activity, and the record-level evaluation does not demonstrate generalization to unseen entities or future click periods. The findings should therefore be interpreted as click-attribution classification results rather than verified click-fraud detection performance.

1. INTRODUCTION

Online pay-per-click (PPC) advertising is vulnerable to click fraud, in which illegitimate clicks may increase publishers' revenue or deplete advertisers' advertising budgets [1]. Artificial-intelligence-based techniques have therefore been increasingly investigated for the detection and prevention of fraudulent advertising clicks [2]. Comparative studies evaluate multiple machine-learning classifiers to distinguish legitimate users from fraudulent or bot-generated click activities in online advertising [3]. More recent research extends this direction by evaluating both machine-learning and deep-learning models with feature engineering to identify differences between legitimate and fraudulent click behavior [4]. Previous studies also propose stacking-based classification for click-fraud prediction by combining multiple learners, together with sampling techniques to address class imbalance [5].

Despite these advances, click fraud remains a persistent challenge in PPC advertising because fraudulent interactions can undermine campaign reliability and impose financial losses on advertisers [1]. Recent studies therefore regard click fraud as an important threat to the integrity and security of online advertising environments [6]. Such fraudulent activities can reduce the trustworthiness and efficiency of digital advertising operations [4]. Consequently, robust, efficient, and scalable detection mechanisms are still required to identify suspicious click behavior and support more reliable PPC advertising environments [7].

Machine learning is widely known as a promising solution in detecting click fraud. This approach works by learning the differentiating patterns between legitimate and fraudulent click behaviors [2]. Recent studies evaluate several algorithms for detecting click fraud, such as Decision Tree, Random Forest (RF), Support Vector Machine (SVM), Naive Bayes, Neural Network, Adaptive Boosting, and Extreme Gradient Boosting (XGBoost) [3]. Besides, other studies apply more complex approaches, such as stacking, multi-algorithm fusion, convolutional neural network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and hybrid learning

frameworks. The aim is to capture nonlinear and sequential clickstream patterns [5]. Batool and Byun [8] develop an ensemble architecture combining CNN and BiLSTM for feature extraction, then using RF for click fraud classification. This result shows that hybrid deep learning models are widely used for PPC advertising. Sisodia and Sisodia [9] then propose a transfer learning framework to detect behavior changes and concept drift among fraudulent publishers according to a user-click dataset. Even though several advantages have been obtained, the problem of imbalance in the extreme class still becomes a challenge. In this condition, minor fraud data count can be polished by class majority. As a result, detection sensitivity toward fraud cases that seldom happen can be decreased, especially in real-world conditions [10].

Imbalanced data is the main challenge in detecting fraud, including click fraud, since conventional classification tends to bias toward the majority class. Hence, the model is less sensitive in detecting a small number of fraud cases, though these cases are important [11]. In the PPC context, errors in classifying click fraud as legitimate clicks may directly impact financial loss and inaccurate results of campaign analysis [12]. To address these problems, several researches apply strategy in data level before model training. For the example, Random Undersampling (RUS), NearMiss, and oversampling methods such as Synthetic Minority Oversampling Technique (SMOTE). The objective is to rebalance the dataset before the training process [13]. Each method has its own advantages and disadvantages. RUS is simple and efficient in computing, yet this method may discard important data randomly from the majority class. SMOTE keeps the data from the majority class since it adds new data in the minority class or synthetic data, yet SMOTE may create noise, enhance the overlap between classes, and increase computational cost. Further, NearMiss chooses class samples according to distance, which maintains the samples that are close to the minority class. Yet this approach may be too focused on local boundaries and delete the actual samples that represent the distribution of the majority class widely [14].

The most recent development in handling imbalanced data is QDPSKNN. This method is a undersampling strategy with prototype selection-based. It separates majority class data into four quarters, then chooses the most representative sample with the K-Nearest Neighbor (KNN) principle [15]. Quad Division Prototype Selection with K-Nearest Neighbor (QDPSKNN) has main differences compared to other methods: compared to RUS, QDPSKNN does not erase the majority sample randomly; compared to SMOTE, QDPSKNN does not make synthetic minority data; and compared to NearMiss, QDPSKNN does not only focus on sample data that is close to the minority class, but spreads the prototype selection to several data partitions. With this mechanism, QDPSKNN can reduce redundant data while preserving the majority class structure and representative decision boundary. Prior empiric results indicate that QDPSKNN outperforms several conventional undersampling methods in various imbalance dataset. These advantages are especially visible in precision, recall, and geometric mean metrics. Therefore, QDPSKNN is considered potentially suitable for high-impact classification tasks, like click fraud detection [15]. Although there are recent studies using deep learning ensemble and transfer learning for detecting click fraud [8, 9], researches that discuss if controlled undersampling with QDPSKNN based can enhance the performance of optimized XGBoost in extreme imbalance classes are still limited [7].

XGBoost is a powerful decision tree-based ensemble algorithm. This algorithm is widely used for classification tasks, especially when the data is large-sized, high in variety, and potentially imbalanced [16, 17]. XGBoost offers several important advantages. First, it applies regularization to reduce the risk of overfitting. Second, it can efficiently handle sparse data. Third, it employs an advanced tree-boosting mechanism to model complex relationships among features. However, in highly imbalanced click-fraud data, classifier performance is also strongly influenced by the strategy used to handle the skewed class distribution. Previous research on click-fraud detection shows that oversampling, undersampling, and hybrid sampling strategies can produce different classification outcomes because each strategy modifies the class distribution in a different way [18]. Despite its strong performance, XGBoost highly relies on appropriate hyperparameter configurations. If the chosen parameters are not suitable, the model's ability to capture patterns in the minority class may drop. This matter is highly visible in cases with imbalanced data, like click fraud detection [19]. Therefore, systematic hyperparameter optimization is highly needed to maximize the ability and advantages of XGBoost.

Random search is used in this study as a practical strategy for hyperparameter optimization. In this approach, hyperparameter combinations are randomly selected from a predefined search space and evaluated according to model performance. This procedure allows different parameter configurations to be explored without exhaustively evaluating every possible combination. Previous fraud-detection research also highlights the importance of hyperparameter optimization when evaluating different sampling strategies and machine-learning classifiers [14]. When combined with cross-validation and appropriate performance metrics, random search provides a systematic procedure for selecting suitable hyperparameter configurations for XGBoost. This approach is particularly relevant for imbalanced fraud-classification problems, where model performance can be sensitive to both parameter settings and data-sampling strategies [16].

According to the identified gaps in prior studies, this study evaluates an integrated click-fraud detection framework that combines the existing QDPSKNN undersampling method with optimized XGBoost classification. For hyperparameter tuning, this study utilizes RandomizedSearchCV [7]. The TalkingData AdTracking Fraud Detection dataset is used as a historical benchmark representing a highly imbalanced click-data setting commonly encountered in PPC advertising [5, 20]. The experimental results are therefore interpreted within this historical dataset distribution rather than as evidence of performance in live PPC environments. Besides QDPSKNN, this study also compares several common undersampling techniques, such as RUS and NearMiss. This comparison is done to provide an overview of the influence of each method on the prediction performance of XGBoost [10]. Model evaluation is carried out using several performance indicators, such as accuracy, precision, recall, F1-score, and Area Under the receiver operating characteristic curve (ROC-AUC). With these metrics, comprehensive assessments are ensured, especially to measure the detection effectiveness in the minority attributed-click class [3].

There are two objectives of this study. First, this study applies optimized XGBoost to classify click-attribution outcomes in a highly imbalanced PPC dataset. Second, it evaluates and compares QDPSKNN-based undersampling with conventional undersampling methods, namely RUS and

NearMiss-3, to assess their effects on classification performance [19]. Accordingly, the contribution of this study does not lie in proposing a new undersampling or classification algorithm. Instead, it lies in the integration and controlled empirical evaluation of the established QDPSKNN undersampling method with an optimized XGBoost classifier for highly imbalanced click-fraud data. The study further provides a consistent comparison of QDPSKNN, RUS, and NearMiss-3 across the evaluated classifier architectures under the same experimental setting and hyperparameter-optimization procedure [4].

2. METHOD

The workflow of this study explains the systematic process, starting with preparation of the raw data and continuing until model evaluation. First, the research starts with the data understanding stage. In this stage, the TalkingData AdTracking Fraud Detection Challenge dataset is analyzed to determine the characteristics, such as the number of data points, attributes, and class distribution. Then, data preprocessing is done through several steps. These steps include removing irrelevant attributes, temporal feature transformation, and normalization into numeric variables; hence, all features have a consistent scale. After the data is cleaned and transformed, class imbalance handling is done. In this step, five resampling strategies are used: RUS, NearMiss-3 (NMU-3), SMOTE, Adaptive Synthetic Sampling (ADASYN), and QDPSKNN, the prototype-based undersampling method evaluated in this study. Next, the balanced data is divided into training and testing data. This separation is used as the foundation for model construction. Model training is conducted using XGBoost as the main classifier, with RF and SVM representing different learning mechanisms. Light Gradient Boosting Machine (LightGBM) is additionally included as a gradient-boosting baseline to provide a direct within-family comparison with XGBoost. All classifiers are evaluated using the same preprocessing, sampling configurations, train–test ratio, and validation framework to ensure a consistent experimental comparison.

To enhance the performance of each algorithm, hyperparameters are tuned using RandomizedSearchCV. This method searches for the best parameter through random sampling from the determined parameter space. Lastly, the model is evaluated using several metrics, such as accuracy, precision, F1-score, and ROC-AUC. This evaluation is aimed to determinde the most effective approach in detecting click fraud at imbalance data, as shown in Figure 1.

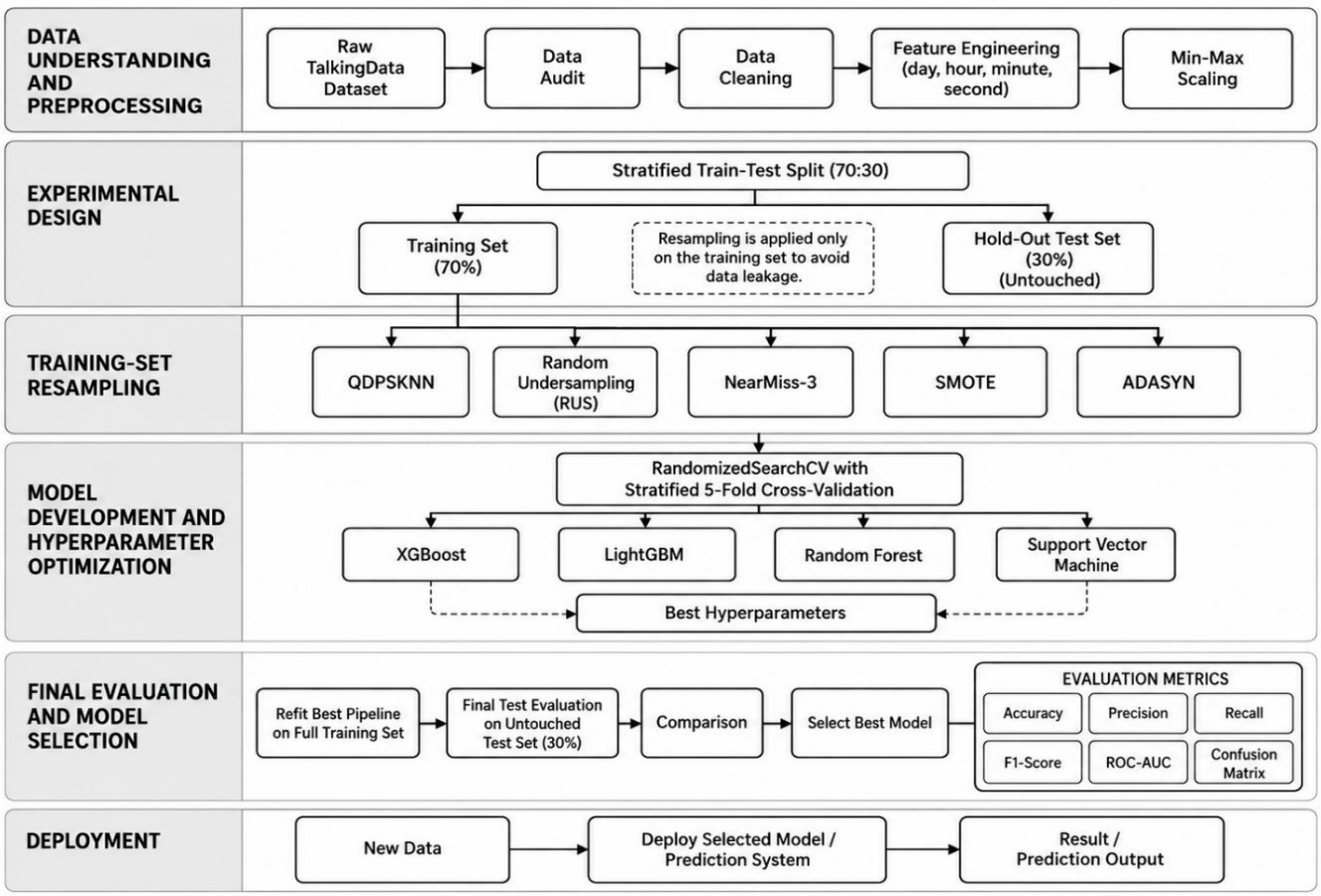

Figure 1. Research workflow for click fraud detection

2.1 Dataset

This study uses the TalkingData AdTracking Fraud Detection Challenge dataset. This dataset is provided by TalkingData, a major big-data platform provider in China, and is publicly available through the Kaggle competition platform. This dataset consists of 100,000 click records and has eight attributes. These attributes include IP address, application ID, device type, operating system, advertising channel, click time, attribution time, and a binary attribution label indicating whether an application download occurred after the advertisement click.

The dataset is highly imbalanced, consisting of 99,773 non-attributed clicks (label 0) and only 227 attributed clicks (label 1). In the original TalkingData dataset, an attributed click indicates whether an application download occurred after an advertisement click; therefore, these labels should not be interpreted directly as verified fraud and legitimate ground-truth classes. In the context of this study, non-attributed clicks are considered potentially suspicious click activity, whereas attributed clicks represent clicks followed by a recorded application download. The severe class imbalance poses a major challenge for classification because the model may become biased toward the majority class and perform poorly in identifying the minority attributed-click class. The summary of the dataset attributes used is presented in Table 1.

For the purposes of this research, attributes that are not useful in model construction are excluded. Specifically, the attributed_time feature is removed since it does not provide meaningful predictive value. On the contrary, the click_time attribute, which was originally stored as a timestamp, is now decomposed into four numeric features, such as day, hour, minute, and second. This decomposition helps machine learning algorithms capture temporal click patterns more clearly and effectively.

2.2 Data preprocessing

Before the models are trained, several preprocessing stages are done to enhance the quality and consistency of the dataset. The first stage is data cleaning. In this stage, non-relevant attributes for the classification task are removed. Especially, the attributed_time attribute is not used since it does not provide predictive value in fraud detecting. The second stage is data transformation. The click_time attribute, which is originally stored as a timestamp, is now decomposed into four numeric features, such as day, hour, minute, and second. By this transformation, temporal behavior patterns can be represented in a pre-structured and suitable format to be processed by a machine learning algorithm.

Table 2 presents selected attributes from the original dataset before preprocessing. The table includes the click_time attribute in its original timestamp format, while attributed_time is omitted because it is not used in model training. The complete dataset structure, consisting of eight attributes, is summarized in Table 1.

Table 1. Summary of dataset attributes

Sample Attribute	Description	Non-Null Count	Data Type
ip	IP address of the click	100,000	Integer
app	Application ID	100,000	Integer
device	Device type ID	100,000	Integer
os	Operating system version ID	100,000	Integer
channel	Advertising channel ID	100,000	Integer
click_time	Timestamp of the click (transformed)	100,000	DateTime
attributed_time	Timestamp of the app download	227	DateTime
is_Attributed	Label: 0 = non-attributed click, 1 = attributed click	100,000	Integer

Note: IP = Internet Protocol; App = Application; OS = Operating System.

Table 2. Sample of selected dataset attributes before preprocessing

IP	App	Device	OS	Channel	Click_time	Is_attributed
87540	12	1	13	497	2017-11-07 09:30:38	0
105560	25	1	17	259	2017-11-07 13:40:27	0
101424	12	1	19	212	2017-11-07 18:05:24	0
94584	13	1	13	477	2017-11-07 04:58:08	0
68413	12	1	1	178	2017-11-09 09:00:09	0
…	…	…	…	…	…	…

Note: IP = Internet Protocol; App = Application; OS = Operating System.

Table 3. Sample of dataset after preprocessing

IP	App	Device	OS	Channel	Day	Hour	Minute	Second	Is_attributed
0.239977	0.020000	0.000259	0.015012	0.997980	0.333333	0.391304	0.508475	0.644068	0
0.289381	0.043636	0.000259	0.019630	0.517172	0.333333	0.565217	0.677966	0.457627	0
0.278041	0.020000	0.000259	0.021940	0.422222	0.333333	0.782609	0.084746	0.406780	0
0.259289	0.021818	0.000259	0.015012	0.957576	0.333333	0.173913	0.983051	0.135593	0
0.187538	0.020000	0.000259	0.001155	0.353535	1.000000	0.391304	0.000000	0.152542	0
…	…	…	…	…	…	…	…	…	…

Note: IP = Internet Protocol; App = Application; OS = Operating System.

To enhance the performance of model prediction, this research applies a data normalization process. The normalization is done using the Min-Max normalization technique, which changes the values of each feature into the standardized range of 0 to 1. This stage prevents larger numeric values from dominating the model learning process. By normalization, all features can contribute equally to classification tasks. Min-Max normalization technique is commonly used in machine learning research since this method is simple and effective for handling heterogeneous features [21-23]. The dataset result after cleaning, transformation, and normalization is illustrated in Table 3.

2.3 Feature importance analysis

To provide empirical explanations about preprocessing

strategies, this study performs feature importance analysis using the final model of QDPSKNN-XGBoost. This analysis uses a gain-based importance method. This method measures the average of improvement in classification loss that happens while decision tree does the split by using a feature. The higher the improvement, the bigger the feature's contribution in distinguishing between the two click classes. The evaluated features include IP, App, Device, OS, Channel, Day, Hour, Minute, and Second.

Feature importance is calculated from the trained model after the hyperparameter optimization process. The result is only used for model interpretation, not to change the model. Therefore, this analysis does not change these elements: the chosen features, data balancing procedure, hyperparameter configuration, and previously reported performance results. The main goal of this analysis is to ensure that the retained variables, such as platform variables, device, channel, and temporal variables, are contributing to click classification. Further, this analysis also provides more reason why click_time is split into separate temporal components (e.g., day, hour, minute, and second).

2.4 Data balancing

The dataset used in this study is highly imbalanced. Attributed clicks account for less than 0.3% of the total sample. This condition may cause the model to become biased toward the majority class. To address this problem, this study applies five resampling strategies: RUS, NMU-3, QDPSKNN, SMOTE, and ADASYN. RUS randomly reduces majority-class samples, while NMU-3 retains samples near the decision boundary using three nearest neighbors. QDPSKNN selects representative prototypes from quartile divisions of the majority class to reduce information loss. In contrast, SMOTE and ADASYN generate synthetic minority-class samples to improve class balance. The current development in imbalanced data learning shows that resampling effectiveness is not only influenced by the class ratio after being balanced. Another important factor is the local neighborhood structure between majority and minority class samples [24]. Besides, comparative studies in detecting fraud also indicate that oversampling, undersampling techniques, or hybrid methods can produce different precision-recall trade-offs. The result depends on classifier types and dataset characteristics [25]. This study evaluates five resampling strategies comprising three undersampling approaches and two oversampling approaches. The undersampling methods include RUS, NearMiss-3, and QDPSKNN, representing random, distance-based, and prototype-based majority-class reduction, respectively. SMOTE and ADASYN are included as representative oversampling methods because they generate synthetic minority-class samples rather than remove majority-class samples. This combination enables a broader comparison across different resampling paradigms. The detailed parameter configurations for each method are presented in Table 4.

Table 4. Summary of resampling methods and parameters

Method	Key Parameters	Parameter Values
RUS	random_state	42
	replacement	False
	sampling_strategy	0.25
NMU-3	n_neighbors	3
	n_neighbors_ver3	3
	sampling_strategy	0.25
	version	3
	n_jobs	None
QDPSKNN	algorithm	Auto
	leaf_size	30
	metric	Minkowski
	p	2
	n_neighbors	5
	weights	Uniform
	metric_params	None
	n_jobs	None

Note: RUS = Random Undersampling; NMU-3 = NearMiss-3; QDPSKNN = Quad-Division Prototype Selection-Based K-Nearest Neighbor.

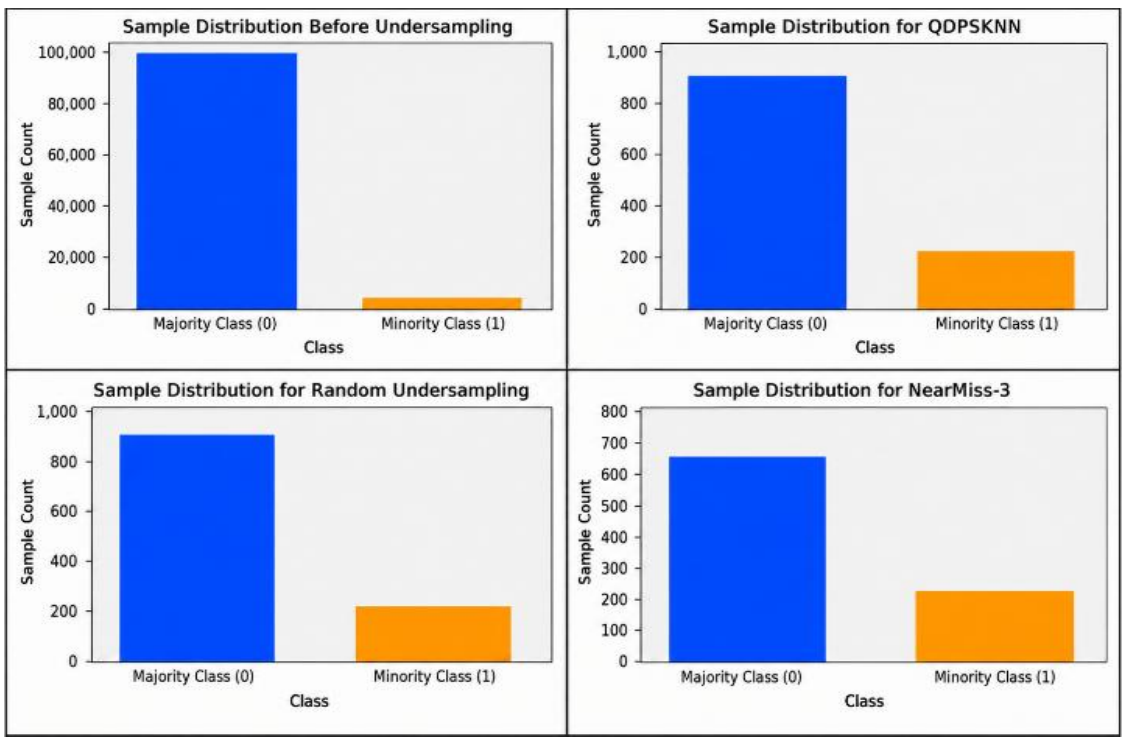

Figure 2. Class distribution before and after undersampling

To illustrate the impact of balancing strategies, Figure 2 shows the class distribution before and after the undersampling process. Before balancing is done, the dataset is dominated by the majority class. After the undersampling method is applied, the ratio between majority and minority classes becomes more balanced. This condition helps in preventing the model from learning to lean toward the majority class. However, the purpose of data balancing is not only to balance the data in each class. This balancing has to retain important samples. These samples need to represent the actual class structure and decision boundary from the original data.

The application of undersampling successfully reduces the dominance of the majority class and produces a more balanced sample distribution between classes. By decreasing bias toward the majority class, the balanced dataset provides a stronger foundation to train a machine learning model. The current study in fraud detection indicates that resampling can be paired with feature selection that considers class imbalance. The purposes are to retain the discriminative attributes and enhance the classification performance in extreme imbalance conditions [26]. Hence, undersampling has an important role in enhancing model performance. More balanced data representation helps the classifier recognize patterns in the minority class better. After this adjustment, the dataset is more balanced and supports the development of a more accurate prediction model [27].

2.5 Model training and hyperparameter tuning

During model training, the dataset is divided into training and testing subsets using a 70:30 ratio. This ratio is commonly used in machine learning studies because it provides sufficient data for model training while retaining an independent subset for performance evaluation [28]. The training subset is used to learn patterns distinguishing non-attributed and attributed clicks, while the testing subset contains held-out records not used during model fitting. Because the split is performed at the record level rather than at the entity or temporal-group level, repeated IP, App, Device, OS, Channel, and temporal patterns may occur in both subsets. Such recurrence does not necessarily constitute direct data leakage, as these attributes can naturally recur across independent click records; however, it may increase similarity between the training and testing distributions and lead to optimistic performance estimates. Therefore, the reported results should be interpreted as performance on held-out records from the same historical data distribution rather than evidence of generalization to completely unseen entities or future click periods. Three primary classifiers with different learning mechanisms are evaluated: XGBoost as a gradient-boosting classifier, RF as a bagging-based tree ensemble, and SVM as a maximum-margin kernel-based classifier. In addition, LightGBM is included as a second gradient-boosting classifier to provide a controlled within-family baseline for XGBoost. The same preprocessing steps, sampling configurations, 70:30 train–test ratio, and evaluation procedure are applied consistently across the classifiers. This design enables the effects of the classifier and sampling strategy to be compared under a common experimental setting. Table 5 presents the training and testing sample counts following the 70:30 split of each resampled dataset, rather than class-specific counts. NearMiss-3 (NMU-3) produces a smaller dataset of 888 samples (621 training and 267 testing) compared with 1,135 samples for the other methods, as its nearest-neighbor-based selection mechanism retains fewer majority-class samples under the applied configuration. The class composition of each test set is also reported in Table 5.

Table 5. Data split distribution

Resampling Method	Data Training (70%)	Data Testing (30%)	Total
QDPSKNN	794	341	1,135
RUS	794	341	1,135
NMU-3	621	267	888
SMOTE	794	341	1,135
ADASYN	794	341	1,135

Note: QDPSKNN = Quad-Division Prototype Selection-Based K-Nearest Neighbor; RUS = Random Undersampling; NMU-3 = NearMiss-3.

To optimize model performance, hyperparameter tuning is conducted using RandomizedSearchCV. For each classifier and sampling configuration, 20 hyperparameter combinations are randomly sampled from the predefined classifier-specific search space (n_iter = 20). Each configuration is evaluated using stratified five-fold cross-validation, resulting in 100 model fits per RandomizedSearchCV procedure. F1-score is used as the optimization criterion because it balances precision and recall and is suitable for imbalanced classification problems [28]. A random state of 42 is used to support reproducibility. The same optimization protocol is applied to XGBoost, RF, SVM, and LightGBM, while the hyperparameter search spaces are adapted to the characteristics of each classifier. Further, this study uses stratified five-fold cross-validation. This technique keeps class proportions consistent in each training and validation fold. So, the estimation of model performance is more stable and trustworthy [29]. After all evaluations are done, the configuration with the highest mean cross-validation F1-score is selected as the final model.

For the XGBoost model, hyperparameter search includes several parameters, as follows: n_estimators from 100 to 300; max_depth of 3 to 10; learning_rate of 0.01 to 0.30; subsample of 0.50 to 1.00; and colsample_bytree of 0.50 to 1.00. These parameters control how the model is built, such as the total number and complexity of decision trees, the learning-step size, the proportion of the training data used in every boosting iteration, and the feature proportion used while building each tree. The searching process of hyperparameter is optimized according to the average of F1-score that is obtained from five validation folds. The range of parameter values is determined according to the experiment design of this study. Further, prior studies also highlight that these parameters are important in controlling bias-variance trade-offs in XGBoost [30].

For the RF model, the hyperparameter search space includes n_estimators of 100 to 500; max_depth of 10 to 50; and max_features with category choices of “sqrt” and “log2”. The meaning of these parameters is as follows: the number of trees (n_estimators) determines the ensemble size. The more trees, the more stable the model is, but computation time also increases. Tree depth (max_depth) influences the model complexity. A tree that is too deep may be overfitting, while a too shallow tree is usually less effective in capturing data patterns. Further, feature selection (max_features) influences the variety across trees. A good variety can help in enhancing model generalization ability. As with XGBoost, the RF hyperparameter search evaluates 20 randomly sampled parameter configurations using stratified five-fold cross-validation, resulting in 100 model fits. The configuration with

the highest mean cross-validation F1-score is selected as the optimal parameter setting. The value range is defined according to the experiment design of this study. Further, prior studies highlight that these parameters are important in controlling variety and generalization ability of RF [31].

For the SVM model, hyperparameter searching includes three main parameters: C: 0.1, 1, and 10; kernel: "linear" and "RBF"; gamma: 0.01, 0.1, and 1. The function of each parameter is as follows: the C parameter controls the balance between making the classification margin as wide as possible and penalizing classification errors. A larger C value usually makes the model stricter in reducing errors, but it can still increase the risk of overfitting. For the gamma parameter, it is influential once the used kernel is RBF. This parameter determines how big the influence of each training data point is. A bigger gamma score makes the data influence become more localized, while a smaller score makes its influence broader. The combinations of the tested parameters are evaluated in the same way as the other models. This study uses F1-score as the main criterion and applies stratified five-fold cross-validation. The purpose is to keep the consistency of evaluation methods among classifiers [32].

For LightGBM, hyperparameter optimization follows the same RandomizedSearchCV protocol used for the other classifiers, consisting of 20 randomly sampled configurations, F1-score as the optimization criterion, stratified five-fold cross-validation, and a random state of 42. The LightGBM search space includes classifier-specific parameters such as n_estimators, learning_rate, num_leaves, max_depth, min_child_samples, subsample, colsample_bytree, reg_alpha, and reg_lambda. Although the parameter search space differs from that of XGBoost because of algorithm-specific characteristics, the preprocessing, sampling procedure, optimization criterion, number of randomized configurations, cross-validation strategy, and final evaluation procedure are kept consistent. This controlled setting enables a fair within-family comparison between XGBoost and LightGBM.

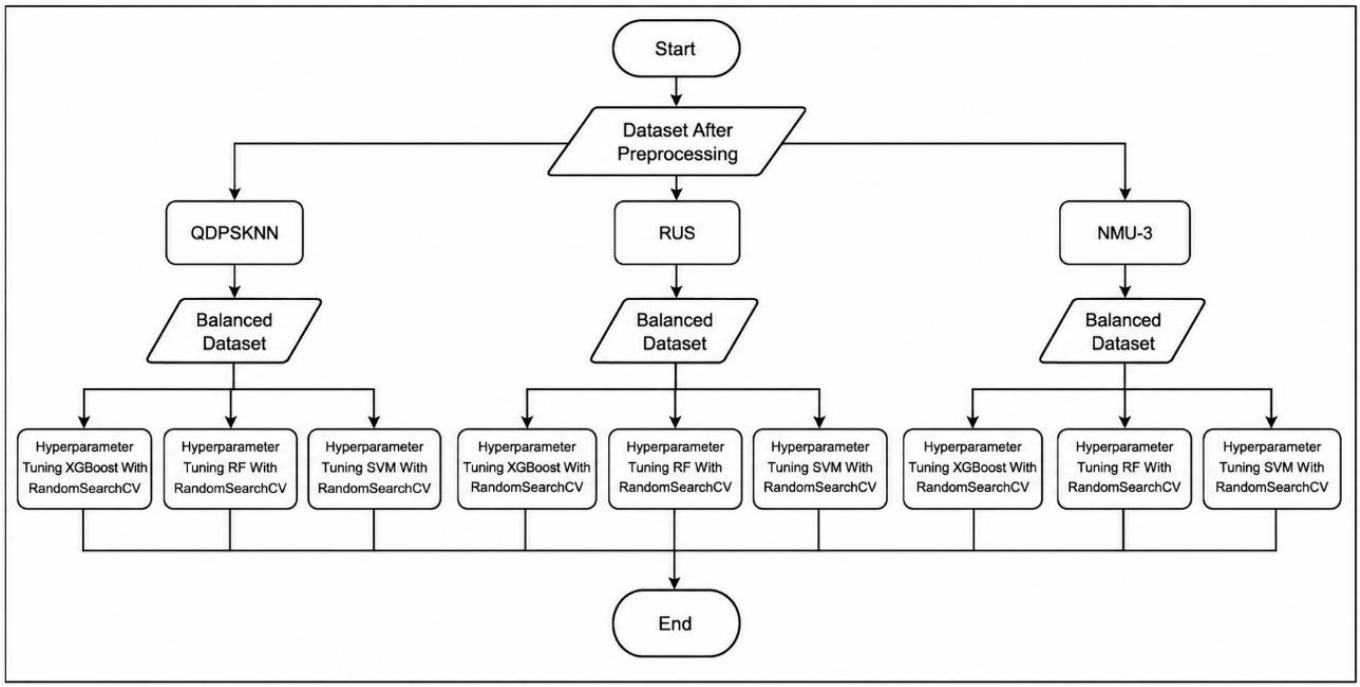

Figure 3. Flowchart of the hyperparameter tuning and model training process

The systematic exploration of parameters through RandomizedSearchCV ensures classifiers are trained using the best parameter configuration in the search space that has been determined. Configuration with the best performance that is found from this process then used in the final stage. Hence, the produced model becomes more robust and accurate [33, 34]. LightGBM is included as an additional gradient-boosting baseline to enable a direct within-family comparison with XGBoost. To maintain experimental consistency, LightGBM uses the same preprocessing pipeline, sampling configurations, train–test ratio, optimization criterion, and stratified five-fold validation strategy as the other evaluated classifiers. Only the hyperparameter search space is classifier-specific, reflecting differences between the LightGBM and XGBoost architectures. This comparison is expected to assess several aspects, such as prediction performance, training time, memory efficiency, and scalability. The complete workflow of the hyperparameter tuning process and model training is shown in Figure 3. This figure shows the sequential stages, starting from parameter searching and cross-validation, choosing the best configuration, until the training of the final.

2.6 Evaluation metrics

The validation process in a machine learning algorithm provides a quantitative way to assess model performance. Validation is done by evaluating how many correct and incorrect predictions. For this context, model performance is measured using several metrics, such as accuracy and precision. Accuracy is one of the commonly used metrics. Accuracy counts the correct prediction proportion, compared to the total of all observed data. This metric is also commonly used to assess how well a model differentiates data from defined categories [1].

$$\text{Accuracy}=\frac{(\text{TP}+\text{TN})}{(\text{TP}+\text{FP}+\text{TN}+\text{FN})} \quad (1)$$

For this context, True Positive (TP) denotes the data that is correctly defined as positive, while False Positive (FP) denotes the negative data that are incorrectly defined as positive. True Negative (TN) denotes negative instances that are correctly classified as negative, whereas False Negative (FN) denotes positive instances that are incorrectly classified as negative. For datasets with extreme class imbalance, accuracy is not enough. Hence, metrics like precision, recall, and F1-score are more informative since they provide an overview of specific performance for each class. Precision shows the proportion of correct positive predictions. This means, from all "positive" prediction results, how many of them are actually positive. Recall shows the proportion of successfully detected positive data. This means, from all actually positive data, how many of them are successfully found by the model [35].

$$\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP}+\text{FN})} \tag{3}$$

F1-score is commonly used in machine learning. This metric is not only used for binary classification, but also for multiclass classification problems. For multiclass classification, there are several ways to calculate the overall F1-score. Two common ways are micro-averaging and macro-averaging. Both of them are used to combine the performance results from all classes. In this averaging method, the F1-score can provide a more balanced model performance overview. F1-score is also often used as an optimization target, especially for classification problems with imbalanced data [15].

$$\text{F1-score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{4}$$

A confusion matrix is a commonly used validation tool for analyzing classification error in machine learning tasks. This tool shows how the model classifies data into defined categories. A confusion matrix shows more specific prediction results per class. The result is organized into four categories: TP, TN, FP, and FN. With this display, the confusion matrix helps to assess the error classification that is made by model. This is more informative and detailed compared to using only one combined metric (e.g., only accuracy) [36].

Besides the confusion matrix-based matrix, model performance is also evaluated using the ROC curve and area under the curve (AUC) value. ROC curve shows the trade-off between the TP rate and FP rate at several classification threshold values. Hence, this curve helps to assess the changes in model performance when the threshold is increased or decreased. Meanwhile, AUC summarizes the information from the ROC curve into one number. This value overviews the model ability in differentiating positive class and negative class. The higher the AUC score, the better the model's ability to differentiate those two classes. For this reason, AUC is very useful for imbalanced data classification evaluation.

3. RESULT

This section reports the experimental results and provides the interpretation. The provided results show model performance in several undersampling strategies and several classifiers. Next, the discussion section will analyze the significance, compare with prior studies, and outline the implications and limitations of the study.

3.1 Feature importance analysis

Figure 4 shows the gain-based feature importance values that have been normalized from the optimized QDPSKNN-XGBoost model. The result shows that Day, Channel, and App are the three most influential features. The score of each importance is 51.44%, 13.61%, and 12.53%. These three features are the most helpful in reducing classification loss while the model constructs decision-tree splits and differentiates between two classes of clicks. The minute feature also provides a meaningful contribution, with a score of 8.44%. After that, the contribution of other features is smaller, such as 4.12% for IP, 3.27% for Device, and 3.00% for Hour. On the contrary, Second and OS have the lowest importance scores, each with 2.23% and 1.35%. This indicates that both contributions to model classification are relatively limited.

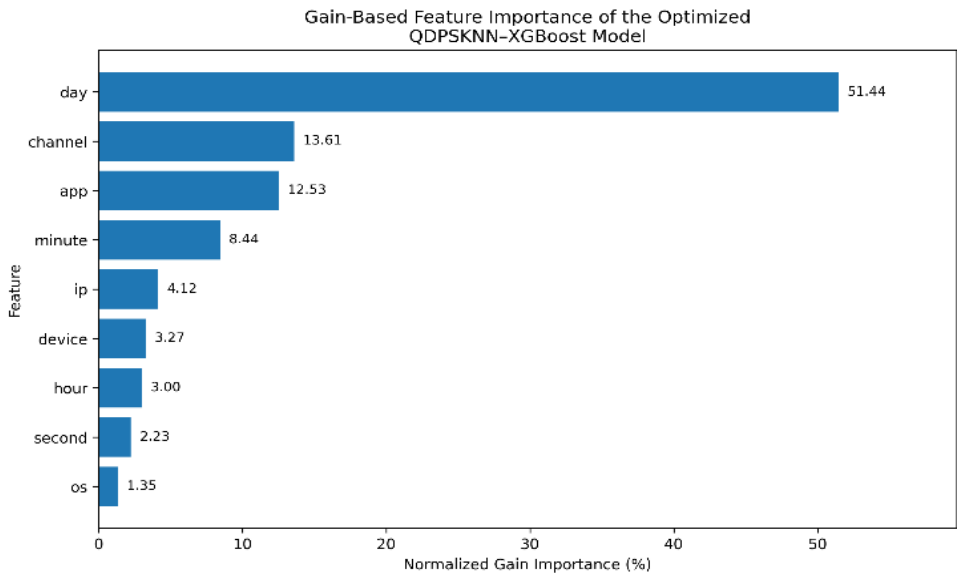

Figure 4. Gain-based feature importance of the optimized QDPSKNN–XGBoost model

The feature importance results provide empirical support for decomposing click_time into Day, Hour, Minute, and Second variables during preprocessing, as the dominant contribution of Day indicates that the click period represented an important source of information for the model, while Minute, Hour, and Second provided additional temporal detail. However, the dominant importance of Day should be interpreted cautiously because it may reflect dataset-specific collection patterns or temporal distributions rather than general fraud-related behavior. Its gain-based importance therefore indicates predictive relevance within this historical dataset rather than a causal relationship or universally applicable fraud pattern.

The contributions of Channel, App, IP, Device, and OS further show that the model relied on both temporal characteristics and the technological context associated with each click. Channel and App were substantially more influential than Device and OS, suggesting that advertising-channel and application-related patterns played a stronger role in the resulting decision-tree structure. Overall, the analysis supports retaining the selected predictors, while gain-based importance should be interpreted only as their relative predictive contribution within this dataset and not as evidence that any feature causally determines click behavior.

3.2 Performance metrics

Table 6 presents a comparative evaluation of all model configurations using five performance metrics: accuracy, precision, recall, F1-score, and ROC-AUC. These metrics provide complementary perspectives on classification performance under imbalanced data conditions. Accuracy measures overall prediction correctness, precision reflects the reliability of positive-class predictions, recall measures the proportion of positive-class observations correctly identified, F1-score balances precision and recall, and ROC-AUC evaluates the model's ability to distinguish between the two classes across different classification thresholds. LightGBM was additionally included as a gradient-boosting baseline to enable a within-family comparison with XGBoost. All LightGBM configurations were evaluated using the same preprocessing, corresponding sampling procedures, 70:30 train–test ratio, RandomizedSearchCV optimization criterion, and stratified five-fold cross-validation protocol used for the other classifiers.

QDPSKNN-XGBoost demonstrates the strongest overall performance among the evaluated configurations, achieving the highest accuracy (96.77%), F1-score (98.09%), and ROC-AUC (0.98), with a recall of 99.30% that is tied with RUS-XGBoost. Although NMU-3-XGBoost records the highest precision (98.31%), its lower recall results in a less balanced precision-recall trade-off. In the reported evaluation, label 0 is treated as the positive class. Following the original TalkingData definition, label 0 represents non-attributed clicks, whereas label 1 represents attributed clicks. Therefore, the recall of 99.30% indicates that most non-attributed click records were correctly identified by the model. These attribution labels should not be interpreted directly as verified fraudulent and legitimate click classes. The additional LightGBM baseline enables a direct comparison within the gradient-boosting family. QDPSKNN–XGBoost outperforms QDPSKNN–LightGBM across all five evaluation metrics, achieving higher accuracy (96.77% vs. 94.72%), precision (96.91% vs. 80.95%), recall (99.30% vs. 89.47%), F1-score (98.09% vs. 85.00%), and ROC-AUC (0.98 vs. 0.96). These results indicate that QDPSKNN–XGBoost demonstrates stronger overall classification performance under the evaluated experimental setting. QDPSKNN combined with XGBoost provides stronger overall performance than the corresponding LightGBM configuration, while confirming that the effectiveness of an undersampling strategy depends on the classifier with which it is combined.

Table 6. Performance of all model configurations

Resampling Method	Accuracy	Precision	Recall	F1-Score	ROC-AUC
QDPSKNN-XGBoost	96.77%	96.91%	99.30%	98.09%	0.98
RUS-XGBoost	96.48%	96.58%	99.30%	97.92%	0.95
NMU-XGBoost	89.14%	98.31%	87.06%	92.35%	0.94
SMOTE-XGBoost	95.65%	88.28%	80.88%	84.42%	0.94
ADASYN-XGBoost	93.94%	90.36%	76.76%	83.01%	0.94
QDPSKNN-RF	95.31%	96.85%	97.54%	97.19%	0.92
RUS-RF	95.01%	96.84%	97.18%	97.01%	0.91
NMU-RF	94.76%	96.06%	97.01%	96.53%	0.90
SMOTE-RF	95.25%	97.62%	88.82%	93.01%	0.93
ADASYN-RF	94.58%	97.28%	89.95%	93.47%	0.94
QDPSKNN-SVM	89.15%	93.33%	93.66%	93.50%	0.89
RUS-SVM	82.99%	93.46%	85.56%	89.34%	0.88
NMU-SVM	77.90%	90.80%	78.61%	84.27%	0.87
SMOTE -SVM	84.87%	88.92%	77.75%	82.96%	0.81
ADASYN-SVM	86.25%	86.35%	79.17%	82.60%	0.83
QDPSKNN-LightGBM	94.72%	80.95%	89.47%	85.00%	0.96
RUS-LightGBM	95.31%	85.96%	85.96%	85.96%	0.93
NMU-3-LightGBM	92.13%	80.00%	90.91%	85.11%	0.95
SMOTE -LightGBM	93.80%	79.45%	90.29%	84.52%	0.95
ADASYN -LightGBM	92.21%	76.82%	90.23%	82.99%	0.95

Note: QDPSKNN = Quad-Division Prototype Selection-Based K-Nearest Neighbor; RUS = Random Undersampling; NMU-3 = NearMiss-3; ROC-AUC = area under the receiver operating characteristic curve.

3.3 ROC curves and confusion matrices

Figure 5 shows ROC-AUC curves from three XGBoost models that are trained using QDPSKNN, RUS, and NMU undersampling strategies. QDPSKNN-XGBoost curve always stays above RUS and NUM curve in every score thresholds. This model achieves AUC > 0.98, which is categorized as excellent [37]. This means that this model is highly effective

for both non-attributed and attributed clicks. On the contrary, RUS-XGBoost and NMU-XGBoost have lower AUC scores, with 0.95 and 0.94. These scores indicate weaker class differentiation than QDPSKNN-XGBoost. A higher ROC-AUC score than QDPSKNN-XGBoost means this model is better at differentiating two classes at various thresholds. However, ROC-AUC only is not enough to explain the effect of classification errors in practice. The reason is that ROC-AUC summarizes performance across many thresholds, not at the final decision threshold that is actually utilized. To further examine classification errors at the selected decision threshold, Figure 6 presents the confusion matrices of three classifiers—XGBoost, Random Forest (RF), and Support Vector Machine (SVM)—combined with QDPSKNN undersampling, enabling a detailed comparison of false-positive and false-negative predictions.

QDPSKNN-XGBoost achieves a recall of 99.30% and precision of 96.91%, resulting in an F1-score of 98.09%. In the reported evaluation, label 0 is treated as the positive class. Following the original TalkingData definition, label 0 represents non-attributed clicks, whereas label 1 represents attributed clicks. Therefore, the recall of 99.30% indicates that the model correctly identifies most non-attributed click records. Compared with RUS-XGBoost, QDPSKNN-XGBoost achieves the same recall of 99.30% but higher precision, F1-score, and ROC-AUC, indicating better overall classification performance. In contrast, NMU-3-XGBoost achieves the highest precision of 98.31% but a substantially lower recall of 87.06%, indicating a more conservative classification pattern that produces fewer false-positive predictions but misses more fraudulent clicks. For the best-performing QDPSKNN-XGBoost model, the confusion matrix records 9 FPs and 2 FNs. Because label 0 is treated as the positive class, the two FNs represent non-attributed clicks incorrectly classified as attributed clicks, whereas the nine FPs represent attributed clicks incorrectly classified as non-attributed clicks.

Overall, the confusion-matrix analysis confirms that QDPSKNN-XGBoost provides the most favorable balance between the classification of non-attributed and attributed clicks among the evaluated XGBoost configurations. The relatively low number of classification errors indicates that the model distinguishes the two attribution classes effectively within the evaluated historical dataset. However, because the TalkingData attributed variable represents attribution outcome rather than verified fraud ground truth, these errors should not be interpreted directly as undetected fraudulent clicks or legitimate clicks incorrectly flagged as fraud. In the click-fraud context, non-attributed clicks may be considered potentially suspicious activity, but non-attribution alone does not establish that a click is fraudulent. Therefore, the confusion-matrix results should primarily be interpreted in terms of attribution-class classification performance rather than direct fraud-versus-legitimate classification.

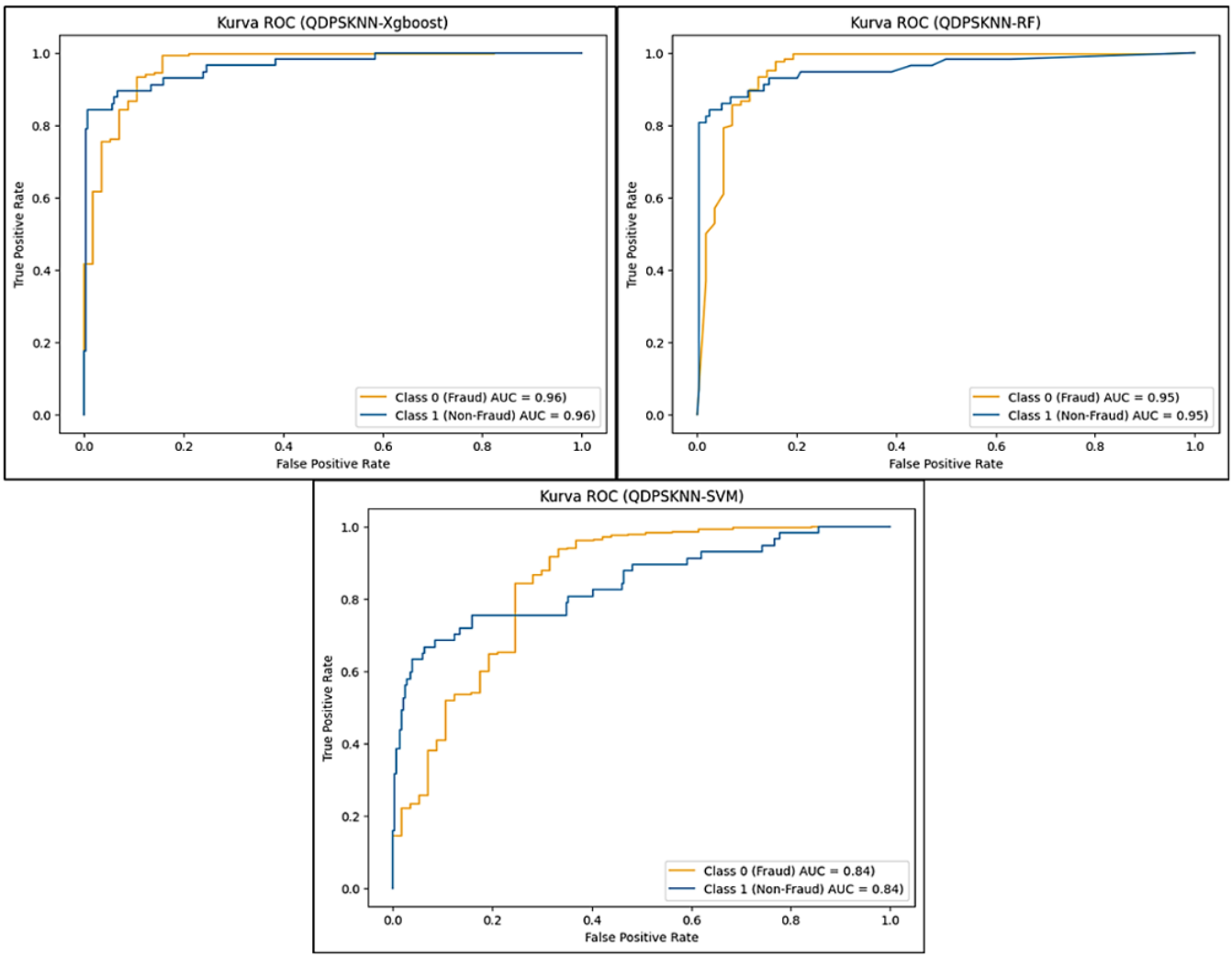

Figure 5. Receiver operating characteristic (ROC) curves for three XGBoost models

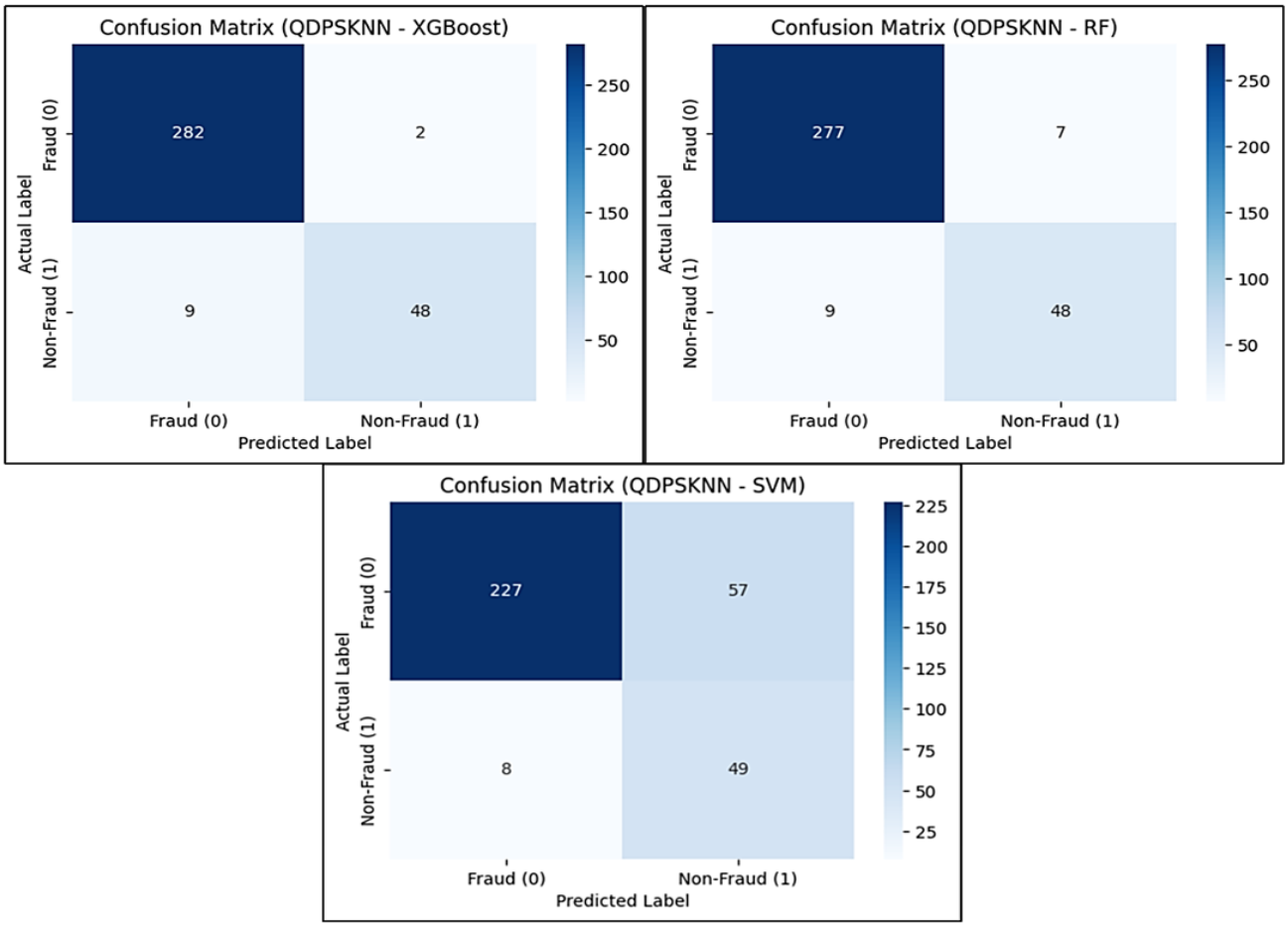

Figure 6. Confusion matrices of 3 models

4. DISCUSSION

The comparative results in Table 6 indicate that QDPSKNN provides relatively consistent performance across the tested classifiers. This suggests that its contribution is not limited to a single classifier. The mechanism of prototype choosing in QDPSKNN seems able to retain more informative majority-class representation than RUS and NMU-3. Thus, QDPSKNN reduces majority-class data while retaining informative samples. The additional comparison with SMOTE and ADASYN further shows that QDPSKNN-XGBoost remains competitive when evaluated against representative oversampling approaches. The best result is obtained when QDPSKNN is combined with XGBoost. This suggests that QDPSKNN prototype selection is compatible with the gradient-boosting mechanism of XGBoost. This combination also helps XGBoost in modelling nonlinear relationships among click features. However, the benefit of QDPSKNN is not always the same for all classifiers. This is reflected in the lower SVM performance, indicating that prototype selection alone cannot overcome limitations associated with a classifier's decision mechanism. From an experimental perspective, this balance is relevant to PPC analysis, but its operational implications should be interpreted cautiously because live PPC traffic was not evaluated.

RUS removes data from the majority class randomly. Therefore, RUS may remove informative majority-class samples while retaining redundant data. NMU-3 prioritizes majority data near minority data. However, this distance-based strategy may concentrate excessively around the decision boundary, resulting in uneven data representation. This limitation is reflected in the NMU-3-XGBoost configuration. This model produces high precision, but far lower recall. It assigns fewer positive-class labels but also misses more positive-class cases. In contrast, SMOTE and ADASYN increase the minority class by generating synthetic samples rather than reducing majority-class observations. The experimental comparison shows that QDPSKNN-XGBoost achieves a more favorable overall performance than the SMOTE-XGBoost and ADASYN-XGBoost baselines under the evaluated setting. QDPSKNN selects nearest-neighbor prototypes from separate quartile partitions, maintaining broader coverage while reducing redundant data.

QDPSKNN requires an additional prototype-selection stage compared with RUS, but it produces a more compact dataset by reducing the number of majority-class samples used in subsequent model training. QDPSKNN is also different from SMOTE or ADASYN because it does not generate synthetic minority samples and therefore does not increase the dataset size through oversampling. However, this study did not directly measure training time, inference time, computational complexity, or memory consumption. Therefore, the reduced sample size should not be interpreted as evidence that QDPSKNN is computationally more efficient or scalable than the compared methods [27].

Previous studies have applied various sampling strategies and classifiers in the TalkingData dataset, as summarized in Table 7. Generally, XGBoost shows competitive performance compared to other classifiers [1]. This study strengthens that finding by combining XGBoost with QDPSKNN. XGBoost

works by sequentially correcting errors from each iteration. Besides, XGBoost is supported by regularization, hyperparameter setting flexibility, and class weight adjustment through the scale_pos_weight parameter. These features help the learning process in imbalanced data conditions [30]. Stronger results of QDPSKNN-XGBoost suggest that the boosting process may benefit from prototypes preserved by QDPSKNN. With these prototypes, the model retains useful information without using all majority data. RF is still competitive since it can reduce the variance and capture nonlinear patterns. In contrast, SVM shows weaker performance in handling cross-feature interactions involving temporal features, application, device, operating system, and channel. The weaker NMU-3-SVM result may reflect the combination of locally concentrated undersampled data and its maximum-margin decision mechanism. SVM performance may be influenced by kernel configuration, class overlap, and the shape of data distribution after undersampling. The evaluated feature types are commonly available in advertiser click logs. However, the present historical evaluation only supports further investigation of QDPSKNN–XGBoost as an analytical approach and does not establish performance in live PPC environments.

RandomizedSearchCV strengthens the study procedure by evaluating the sampled parameter combination in a defined range. Compared to manual choosing, this method reduces the dependence on the researcher's judgement and makes a more systematized optimization process, also easier to replicate. However, the chosen configuration is the best combination limited to the defined search space. The result does not necessarily become the best solution globally. Further, configuration with weaker performance shows important thing, that hyperparameter optimization does not always able to repair any problems if data distribution of resampling result is not suitable with the applied classifier architecture.

Table 7. Comparative performance of prior studies and this work

No.	Authors	Classifiers	Data Sampling Method	Accuracy
1	Duan et al. [6]	Cascaded Forest and XGBoost	RUS	94.53%
2	Gupta et al. [11]	RF	-	88.00%
3	Sisodia and Sisodia [15]	KNN	QDPSKNN	Not Reported
4	Wijaya et al. [25]	XGBoost, RF, DT, LR	SMOTE, ADASYN	Not Reported
5	Makkineni et al. [31]	XGBoost	-	96.20%
6	Babic et al. [32]	Autoencoder ANN-GAN	Semi-supervised GAN	94.70%
7	This Study	XGBoost	QDPSKNN	96.77%

Note: ANN = Artificial Neural Network; GAN = Generative Adversarial Network; RF = Random Forest; DT = Decision Tree; LR = Logistic Regression; KNN = K-Nearest Neighbors; QDPSKNN = Quad-Division Prototype Selection-Based K-Nearest Neighbor; RUS = Random Undersampling; SMOTE = Synthetic Minority Oversampling Technique; ADASYN = Adaptive Synthetic Sampling; XGBoost = Extreme Gradient Boosting.

The comparison in Table 7 is contextual, not the actual evidence. Even though QDPSKNN-XGBoost shows higher accuracy than the XGBoost configuration from prior studies [1], the comparison between studies may be influenced by many differences. For example, differences in pre-processing, sampling ratio, class definition, hyperparameters, and testing data distribution. Hence, controlled comparison in a study provides stronger evidence. Table 7 also shows that previous studies have adopted different strategies for handling class imbalance. Previous QDPSKNN research [15] applied prototype-based undersampling with KNN, whereas SMOTE- and ADASYN-based approaches [25] address class imbalance through synthetic minority-sample generation. Other fraud-detection studies have also employed RUS, ensemble learning, and hybrid classifier frameworks. In contrast, this study integrates QDPSKNN with optimized XGBoost and compares it with RUS, NearMiss-3, SMOTE, and ADASYN under the same experimental setting. Potential operational use remains conceptual because the present study does not evaluate live traffic, unseen entities, future click periods, threshold calibration, or platform integration. Accordingly, the reported results should not be interpreted as evidence of deployment readiness or general applicability to real-world PPC environments.

Several limitations remain. First, this study only uses one dataset. This dataset may only represent a certain click fraud pattern; hence, the result is not necessarily applicable to other conditions. Second, QDPSKNN is quite sensitive to the n_neighbors parameter. If the value does not match, the chosen prototype may not represent the actual data. Third, evaluation is done using static historical data. This study has not discussed concept drift, which is the shift of fraud patterns from time to time. In addition, the record-level train–test split did not enforce entity-disjoint or chronological separation; therefore, repeated IP, App, Device, and temporal patterns could occur in both subsets and may limit generalization to completely unseen entities or future click periods. Besides, computation time and memory usage are not specifically measured. Fourth, this study has not conducted instance-level error analysis. Hence, the analysis of what combination of features are often causing repetitive errors, both FPs and FNs. For future studies, several matters need to be explored: using more diverse datasets, more automatic parameter optimization, temporal validation, adaptive model renewal mechanism, and direct testing of scalability in larger-sized data.

Overall, these findings show that controlled prototype choice can preserve more informative majority-class representation compared to random and distance-based undersampling, while remaining competitive with the evaluated synthetic oversampling baselines. This approach produces a more compact training set, but this should not be interpreted as evidence of lower computational cost. However, weaker results in some NMU-3, SMOTE, ADASYN, and SVM configurations indicate that performance is not determined only by the resampling method. Performance also depends on how the method transforms the data distribution, and how well the classifier creates a decision boundary from the new distribution. Accordingly, the evaluated QDPSKNN–XGBoost integration should presently be regarded as an experimental approach for historical click-data classification rather than a validated operational mechanism for real-world PPC fraud detection. This framework is not a replacement for invalid traffic control that is owned by advertising platforms, which is usually proprietary. The effectiveness and scalability of this approach are influenced by several factors. These

factors include classifier architecture, parameter configuration, operational cost caused by incorrect detection, and compatibility with the data pipeline at the platform level. Before deployment in real-time digital advertising environments, broader validation is required, particularly using entity-disjoint and chronological evaluation designs, diverse datasets, instance-level failure analysis, computational-cost measurements, and threshold analysis based on operational error costs.

5. CONCLUSIONS

This study evaluates QDPSKNN-based undersampling for addressing severe class imbalance in click fraud detection and examines its integration with XGBoost. Within the evaluated experimental setting, QDPSKNN preserves informative majority-class samples more effectively than RUS and NMU and remains competitive against the SMOTE and ADASYN oversampling baselines. When combined with XGBoost, QDPSKNN achieves the highest classification accuracy of 96.77%, indicating the strongest overall performance among the compared configurations. These findings support the effectiveness of QDPSKNN–XGBoost for classifying held-out click records drawn from the same historical data distribution. However, they should not be interpreted as evidence of generalization to completely unseen entities or future click periods.

RandomizedSearchCV provides a systematic procedure for exploring hyperparameter configurations and selecting the best-performing configuration within the predefined search space. Combined with stratified cross-validation, this approach improves the consistency and reproducibility of model optimization. The controlled comparison further shows that classification performance depends not only on the classifier but also on how the selected resampling strategy modifies the training-data distribution. Therefore, the contribution of this study lies primarily in the controlled empirical evaluation of QDPSKNN combined with optimized XGBoost under highly imbalanced click-fraud data rather than in demonstrating universal superiority across different datasets or operational environments.

Several limitations should be considered when interpreting the findings. First, the study uses a single historical dataset that may represent only specific click-fraud patterns. Second, QDPSKNN performance remains sensitive to the n_neighbors parameter, which requires further optimization. Third, the 70:30 partition was performed at the record level rather than using entity-disjoint or chronological separation. Consequently, repeated IP, App, Device, and temporal patterns may occur in both training and testing subsets, potentially increasing similarity between the two distributions and leading to optimistic estimates of predictive performance. The reported results should therefore be interpreted as performance on held-out records from the same historical distribution. Future studies should evaluate the framework using multiple click-fraud datasets, entity-level and chronological validation, automatic optimization of QDPSKNN parameters, and explicit measurements of training time, inference time, and memory usage.

In conclusion, the QDPSKNN–XGBoost integration demonstrates promising performance for highly imbalanced click-fraud classification within the evaluated historical record-level setting. Further validation using entity-disjoint and temporally separated test sets is necessary to determine whether this performance can be maintained for unseen entities, evolving fraud patterns, and future click periods. Such evaluation, together with broader datasets and computational-efficiency analysis, is required before stronger conclusions can be drawn regarding the framework's generalizability, scalability, and suitability for real-time digital advertising environments.

ACKNOWLEDGMENT

The authors gratefully acknowledge Universitas Multimedia Nusantara, Buddhi Dharma University, and Bina Nusantara University for their institutional support and research funding, which enabled the completion of this study.

AUTHOR CONTRIBUTIONS

Regarding author contributions, Kevin was primarily responsible for conceptualization, data curation, data preprocessing, implementation of the QDPSKNN undersampling strategy, model development, hyperparameter optimization, experimental evaluation, and drafting of the original manuscript. Jansen Wiratama and Aditiya Hermawan contributed to the methodological design, feature engineering strategy, validation of experimental results, and interpretation of findings, while also providing technical input on machine learning modeling and performance evaluation. Gredion Prajena contributed to supervision, research coordination, critical review and editing of the manuscript, and ensuring methodological rigor and alignment with the existing literature. All authors participated in reviewing and revising the manuscript, approved the final version, and agree to be accountable for the accuracy, originality, and integrity of the work.

STATEMENT ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE

The final manuscript underwent thorough refinement, including minor paraphrasing, clarity checks, and grammatical enhancements to ensure stylistic consistency and academic readability, with the assistance of an AI-powered writing tool used solely for language polishing. This tool did not influence the scientific content, data analysis, interpretation of results, or conclusions.

DATA AVAILABILITY

The dataset used in this study is publicly available on Kaggle under the TalkingData AdTracking Fraud Detection Challenge and can be accessed at https://www.kaggle.com/competitions/talkingdata-adtracking-fraud-detection/.

REFERENCES

[1] Aljabri, M., Mohammad, R.M.A. (2023). Click fraud detection for online advertising using machine learning.

Egyptian Informatics Journal, 24(2): 341-350. https://doi.org/10.1016/j.eij.2023.05.006
[2] Alzahrani, R.A., Aljabri, M. (2022). AI-based techniques for Ad click fraud detection and prevention: Review and research directions. Journal of Sensor and Actuator Networks, 12(1): 4. https://doi.org/10.3390/jsan12010004
[3] Abbas, Z.A., Hilal, Z.M., Jabbar, H.G. (2025). Click fraud detection in online advertising: A comparative study of machine learning models. International Journal of Safety & Security Engineering, 15(3): 427-441. https://doi.org/10.18280/ijsse.150303
[4] Alzahrani, R.A., Aljabri, M., Mohammad, R.A.M. (2025). Ad click fraud detection using machine learning and deep learning algorithms. IEEE Access, 13: 12746-12763. https://doi.org/10.1109/ACCESS.2025.3532200
[5] Sahllal, N., Souidi, E.M. (2023). Click fraud prediction by stacking algorithm. Intelligenza Artificiale, 17(1): 131-141. https://doi.org/10.3233/IA-221069
[6] Duan, G., Liu, J., Kong, W., Cui, B., Li, J. (2023). Research on click fraud prediction based on multi-algorithm fusion. In Eighth International Symposium on Advances in Electrical, Electronics, and Computer Engineering (ISAEECE 2023), 12704: 127043P. https://doi.org/10.1117/12.2680157
[7] Thejas, G.S., Dheeshjith, S., Iyengar, S.S., Sunitha, N.R., Badrinath, P. (2021). A hybrid and effective learning approach for click fraud detection. Machine Learning with Applications, 3: 100016. https://doi.org/10.1016/j.mlwa.2020.100016
[8] Batool, A., Byun, Y.C. (2022). An ensemble architecture based on deep learning model for click fraud detection in pay-per-click advertisement campaign. IEEE Access, 10: 113410-113426. https://doi.org/10.1109/ACCESS.2022.3211528
[9] Sisodia, D., Sisodia, D.S. (2023). A transfer learning framework towards identifying behavioral changes of fraudulent publishers in pay-per-click model of online advertising for click fraud detection. Expert Systems with Applications, 232: 120922. https://doi.org/10.1016/j.eswa.2023.120922
[10] Bounab, R., Guelib, B., Zarour, K. (2024). A novel machine learning approach for handling imbalanced data: Leveraging SMOTE-ENN and XGBoost. In 2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS), EL Oued, Algeria, pp. 1-7. https://doi.org/10.1109/PAIS62114.2024.10541220
[11] Gupta, P., Varshney, A., Khan, M.R., Ahmed, R., Shuaib, M., Alam, S. (2023). Unbalanced credit card fraud detection data: A machine learning-oriented comparative study of balancing techniques. Procedia Computer Science, 218: 2575-2584. https://doi.org/10.1016/j.procs.2023.01.231
[12] Rubaidi, Z.S., Ammar, B.B., Aouicha, M.B. (2022). Fraud detection using large-scale imbalance dataset. International Journal on Artificial Intelligence Tools, 31(8): 2250037. https://doi.org/10.1142/S0218213022500373
[13] Zhang, F. (2022). Improved credit card fraud detection method based on XGBoost algorithm. BCP Business Management, 38: 2888-2895. https://doi.org/10.54691/bcpbm.v38i.4206
[14] Ahmad, I., Waleed, M., Ullah, A., Jamil, M.U. (2025). Enhancing credit card fraud detection with hybrid sampling and machine learning models. In 2025 6th International Conference on Advancements in Computational Sciences (ICACS), Lahore, Pakistan, pp. 1-7. https://doi.org/10.1109/ICACS64902.2025.10937888
[15] Sisodia, D., Sisodia, D.S. (2022). Quad division prototype selection-based k-nearest neighbor classifier for click fraud detection from highly skewed user click dataset. Engineering Science and Technology, an International Journal, 28: 101011. https://doi.org/10.1016/j.jestch.2021.05.015
[16] Priscilla, C.V., Prabha, D.P. (2020). Influence of optimizing XGBoost to handle class imbalance in credit card fraud detection. In 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, pp. 1309-1315. https://doi.org/10.1109/ICSSIT48917.2020.9214206
[17] Juliandri, R., Johan, M.E., Wiratama, J., Sanjaya, S.A. (2024). Adverse media classification: A new era of risk management with XGBoost and gradient boosting algorithms. In 2024 5th International Conference on Big Data Analytics and Practices (IBDAP), Bangkok, Thailand, pp. 18-21. https://doi.org/10.1109/IBDAP62940.2024.10689708
[18] Sisodia, D., Sisodia, D.S. (2022). Data sampling strategies for click fraud detection using imbalanced user click data of online advertising: An empirical review. IETE Technical Review, 39(4): 789-798. https://doi.org/10.1080/02564602.2021.1915892
[19] Jayanto, E.D., Girsang, A.S. (2025). Optimizing classification of suspicious financial transactions using machine learning: A comparative study of random forest, XGBoost, and SVM. In 2025 4th International Conference on Electronics Representation and Algorithm (ICERA), Yogyakarta, Indonesia, pp. 1-8. https://doi.org/10.1109/ICERA66156.2025.11087364
[20] Wibowo, J.J., Kristiyanti, D.A., Wiratama, J. (2025). Enhancing heart disease classification: A comparative analysis of SMOTE and naïve bayes on imbalanced data. International Journal on Informatics Visualization, 9(5): 2072-2079. https://doi.org/10.62527/joiv.9.5.3248
[21] Chen, W., Yang, K., Yu, Z., Shi, Y., Chen, C.P. (2024). A survey on imbalanced learning: Latest research, applications and future directions. Artificial Intelligence Review, 57(6): 137. https://doi.org/10.1007/s10462-024-10759-6
[22] Mqadi, N.M., Naicker, N., Adeliyi, T. (2021). Solving misclassification of the credit card imbalance problem using near miss. Mathematical Problems in Engineering, 2021(1): 7194728. https://doi.org/10.1155/2021/7194728
[23] Huang, H., Liu, B., Xue, X., Cao, J., Chen, X. (2024). Imbalanced credit card fraud detection data: A solution based on hybrid neural network and clustering-based undersampling technique. Applied Soft Computing, 154: 111368. https://doi.org/10.1016/j.asoc.2024.111368
[24] Koziarski, M., Woźniak, M. (2024). Local neighborhood encodings for imbalanced data classification. Machine Learning, 113(10): 7421-7449. https://doi.org/10.1007/s10994-024-06563-6
[25] Wijaya, M.G., Pinaringgi, M.F., Zakiyyah, A.Y. (2024). Comparative analysis of machine learning algorithms and data balancing techniques for credit card fraud detection. Procedia Computer Science, 245: 677-688.

https://doi.org/10.1016/j.procs.2024.10.294
[26] Breskuvienė, D., Dzemyda, G. (2024). Enhancing credit card fraud detection: Highly imbalanced data case. Journal of Big Data, 11(1): 182. https://doi.org/10.1186/s40537-024-01059-5
[27] Sisodia, D., Sisodia, D.S. (2023). Stacked generalization architecture for predicting publisher behaviour from highly imbalanced user-click data set for click fraud detection. New Generation Computing, 41(3): 581-606. https://doi.org/10.1007/s00354-023-00218-1
[28] Joseph, V.R. (2022). Optimal ratio for data splitting. Statistical Analysis and Data Mining: The ASA Data Science Journal, 15(4): 531-538.https://doi.org/10.1002/sam.11583
[29] Wen, J., Tang, X., Lu, J. (2024). An imbalanced learning method based on graph tran-smote for fraud detection. Scientific Reports, 14(1): 16560. https://doi.org/10.1038/s41598-024-67550-4
[30] Mehdary, A., Chehri, A., Jakimi, A., Saadane, R. (2024). Hyperparameter optimization with genetic algorithms and XGBoost: A step forward in smart grid fraud detection. Sensors, 24(4): 1230. https://doi.org/10.3390/s24041230
[31] Makkineni, N., Ciripuram, A., Subhani, S., Kakulapati, V. (2023). Fraud detection of AD clicks using machine learning techniques. Journal of Scientific Research and Reports, 29(7): 84-89. https://doi.org/10.9734/jsrr/2023/v29i71762
[32] Babic, L., Zeljkovic, V., Jovanovic, L., et al. (2024). Click fraud detection with recurrent neural networks optimized by adapted crayfish optimization algorithm. Journal of Industrial Intelligence, 2(4): 230-239. https://doi.org/10.56578/jii020404
[33] Petrovic, A., Antonijevic, M., Strumberger, I., Jovanovic, L., Savanovic, N., Janicijevic, S. (2023). The XGBoost approach tuned by TLB metaheuristics for fraud detection. In 1st International Conference on Innovation in Information Technology and Business (ICIITB 2022), Atlantis Press, pp. 219-234. https://doi.org/10.2991/978-94-6463-110-4_16
[34] Dalal, S., Seth, B., Radulescu, M., Secara, C., Tolea, C. (2022). Predicting fraud in financial payment services through optimized hyper-parameter-tuned XGBoost model. Mathematics, 10(24): 4679. https://doi.org/10.3390/math10244679
[35] Vanacore, A., Pellegrino, M.S., Ciardiello, A. (2024). Fair evaluation of classifier predictive performance based on binary confusion matrix. Computational Statistics, 39(1): 363-383. https://doi.org/10.1007/s00180-022-01301-9
[36] Theissler, A., Thomas, M., Burch, M., Gerschner, F. (2022). ConfusionVis: Comparative evaluation and selection of multi-class classifiers based on confusion matrices. Knowledge-Based Systems, 247: 108651. https://doi.org/10.1016/j.knosys.2022.108651
[37] Nahm, F.S. (2022). Receiver operating characteristic curve: Overview and practical use for clinicians. Korean Journal of Anesthesiology, 75(1): 25-36. https://doi.org/10.4097/kja.21209