CIA Open Access

Feature-Based Detection of Bot Traffic and Click Fraud in Mobile Advertising: A Comparative Analysis

Ruoxi Jia[1], Xin Lu[1.2], Serena Whitmore[2]

[1] Computer Science, University of Southern California, CA, USA
[1.2] Computer Science, Stanford University, CA, USA
[2] Computer Science, University of Washington, Seattle, WA, USA

DOI: 10.63575/CIA.2024.20112

Abstract

Mobile advertising fraud has emerged as a critical security challenge, causing substantial financial losses across digital ecosystems. This paper presents a comprehensive comparative analysis of machine learning algorithms for detecting bot traffic and click fraud through feature-based approaches. We engineer and evaluate temporal, behavioral, and device-specific features across multiple classification algorithms including Random Forest, XGBoost, LightGBM, and deep learning architectures. Experimental results on real-world advertising datasets demonstrate that ensemble methods achieve superior performance with accuracy exceeding 98%, while deep learning approaches provide enhanced robustness against sophisticated fraud patterns. Feature importance analysis reveals that temporal activity patterns and device consistency metrics serve as primary discriminators between legitimate and fraudulent traffic. Our findings provide actionable deployment guidelines for advertising platforms balancing detection accuracy with computational efficiency.

Keywords: *Click fraud detection, Bot traffic analysis, Machine learning, Feature engineering, Mobile advertising security*

1. Introduction

1.1. The Growing Challenge of Ad Fraud in Mobile Ecosystems

1.1.1. Economic impact of invalid traffic in digital advertising

The proliferation of mobile advertising has transformed digital marketing landscapes, generating revenues exceeding 300 billion dollars annually. Industry estimates indicate that fraudulent activities account for 20-30% of all mobile advertising impressions, resulting in annual losses surpassing 80 billion dollars globally[1]. Mobile advertising fraud manifests through bot-generated traffic and artificial click inflation. The economic ramifications extend beyond direct financial losses, encompassing degraded advertiser confidence and distorted campaign analytics. Fraudulent operations leverage automated scripts and compromised devices to simulate legitimate user engagement, systematically draining advertising budgets while delivering zero genuine user interactions.

1.1.2. Evolution of fraud techniques from simple bots to sophisticated attacks

Early-generation advertising fraud relied on rudimentary bot scripts executing repetitive click patterns easily identifiable through statistical anomaly detection. Contemporary fraud operations have evolved into sophisticated ecosystems employing advanced evasion techniques including randomized timing intervals, simulated mouse movements, and diversified IP rotation strategies[2]. The emergence of SDK-level fraud has introduced unprecedented complexity, wherein malicious code embedded within legitimate application libraries executes fraudulent activities. Attribution fraud schemes exploit the mobile advertising attribution chain, manipulating conversion tracking mechanisms to claim credit for organic installations[3]. The continuous arms race between fraud operations and detection systems necessitates adaptive approaches capable of identifying emerging attack methodologies.

1.1.3. Challenges in real-time detection for programmatic advertising

Programmatic advertising platforms process billions of bid requests daily, requiring fraud detection mechanisms that operate within millisecond-scale latency constraints. Detection systems face the dual challenge of minimizing false positive rates while maintaining high sensitivity to fraudulent patterns. The dynamic nature of fraud operations introduces concept drift, wherein attack patterns continuously evolve to exploit detection blind spots[4]. Scalability requirements necessitate detection architectures capable of processing massive data volumes across distributed infrastructure. The heterogeneity of mobile device ecosystems complicates feature standardization and model generalization.

1.2. Current Detection Approaches and Limitations

1.2.1. Traditional rule-based detection methods and their evasion

Legacy fraud detection systems predominantly rely on rule-based heuristics encoding domain expert knowledge into threshold-based decision criteria. Rule-based systems demonstrate computational efficiency and interpretable decision logic. The inherent rigidity of predefined rules creates systematic vulnerabilities exploitable through adversarial adaptation[5]. Sophisticated fraud operations conduct reconnaissance to identify detection thresholds, subsequently calibrating attack parameters to operate within acceptable ranges. Manual rule refinement processes introduce operational delays in responding to emerging attack patterns.

1.2.2. Machine learning approaches and feature engineering challenges

Machine learning methodologies have demonstrated superior adaptability through data-driven pattern recognition capabilities. The efficacy of learning-based approaches fundamentally depends on comprehensive feature engineering encompassing temporal dynamics, behavioral patterns, and contextual attributes. Behavioral modeling requires capturing subtle deviations from normal usage patterns while accommodating legitimate diversity in user interaction styles[6]. The class imbalance problem necessitates specialized sampling strategies and evaluation metrics. The adversarial nature of fraud detection introduces non-stationarity, requiring continuous model retraining.

1.3. Research Objectives and Contributions

1.3.1. Comprehensive feature analysis for fraud detection

This research undertakes systematic investigation of feature engineering strategies for mobile advertising fraud detection, encompassing temporal, behavioral, and device-specific attribute categories. We develop a comprehensive feature taxonomy organizing 47 distinct attributes across hierarchical categories[7]. Temporal features capture activity rhythms and inter-event timing relationships. Behavioral attributes encode click-through patterns and conversion sequences. Device features encompass hardware fingerprints and geographic consistency metrics.

1.3.2. Comparative evaluation of ML algorithms

We conduct comprehensive comparative evaluation of classification algorithms spanning traditional ensemble methods and deep learning architectures including Random Forest, XGBoost, LightGBM, CNN, and BiLSTM networks**Error! Reference source not found.**. Performance assessment employs multiple evaluation metrics including accuracy, precision, recall, F1-score, and AUC-ROC. We analyze algorithm-specific strengths through feature importance rankings and computational efficiency measurements.

1.3.3. Practical deployment considerations and recommendations

Our research provides actionable deployment guidance addressing real-world operational constraints including latency requirements and resource limitations[8]. We quantify accuracy-efficiency tradeoffs across algorithm configurations. The analysis encompasses model update strategies addressing concept drift and ensemble approaches combining multiple detection signals.

2. Background and Related Work

2.1. Ad Fraud Taxonomy and Attack Vectors

2.1.1. Click fraud and impression fraud mechanisms

Click fraud operations generate artificial click events on advertising content without genuine user interest, systematically depleting advertiser budgets. Impression fraud focuses on generating fake ad display events, inflating viewability metrics without actual user exposure[9]. Technical mechanisms include invisible pixel rendering, rapid auto-refreshing, and stacked ad placements. Click injection attacks exploit Android accessibility services to intercept organic application installations.

2.1.2. Bot traffic patterns and click farm operations

Botnet infrastructures leverage compromised consumer devices and dedicated fraud servers to generate massive volumes of artificial traffic. Modern bot implementations employ residential proxy networks to mask attack origins. Click farms coordinate human workers executing manual click tasks at scale, combining human-generated patterns with automated distribution systems[10]. Advanced bot operations implement behavioral randomization including variable timing delays and simulated mouse trajectories.

2.1.3. Attribution fraud and SDK-level attacks

Mobile attribution fraud exploits the conversion tracking infrastructure determining which advertising touchpoints receive credit for application installations. Click spamming specifically targets users likely to install applications organically, injecting fraudulent attribution claims immediately preceding genuine user actions[11]. SDK hijacking introduces malicious code within advertising library components, enabling programmatic manipulation of attribution signals.

2.2. Machine Learning Approaches for Fraud Detection

2.2.1. Supervised learning methods: Random Forest, XGBoost, and neural networks

Supervised classification algorithms have demonstrated substantial efficacy in fraud detection through discriminative pattern recognition from labeled training data. Random Forest ensembles construct multiple decision trees through bootstrap aggregation, achieving robust performance[12]. XGBoost implements gradient boosting frameworks optimizing additive tree ensembles through second-order gradient approximations. Neural network architectures learn hierarchical feature representations, capturing complex non-linear relationships between input attributes and fraud classifications.

2.2.2. Unsupervised and semi-supervised approaches for anomaly detection

Unsupervised learning methodologies identify fraudulent traffic through statistical deviation detection without requiring labeled training examples. Clustering algorithms group similar traffic patterns, flagging outlier clusters. Autoencoders learn compressed representations of normal traffic patterns, detecting fraud through reconstruction error magnitudes**Error! Reference source not found.**. Semi-supervised approaches leverage small labeled datasets combined with large unlabeled corpora.

2.2.3. Deep learning architectures: CNN, LSTM, and hybrid models

Convolutional Neural Networks extract spatial patterns through local receptive field operations. Long Short-Term Memory networks address sequential pattern recognition through gated recurrence mechanisms maintaining long-term dependencies**Error! Reference source not found.**. The LSTM architecture proves particularly valuable for modeling temporal dynamics in click sequences and session behaviors. Hybrid architectures combining CNN feature extraction with LSTM temporal modeling achieve enhanced performance.

2.3. Feature Engineering in Ad Fraud Detection

2.3.1. Temporal and behavioral features

Temporal attributes capture activity rhythms and timing patterns distinguishing legitimate user interactions from automated bot behaviors. Inter-click interval distributions reveal temporal regularities characteristic of scripted automation. Activity concentration metrics quantify temporal clustering, identifying suspicious patterns such as continuous overnight activity. Behavioral attributes encode interaction sequence patterns and engagement depth metrics.

2.3.2. Device fingerprinting and network-level features

Device fingerprinting constructs unique identifiers through hardware and software attribute combinations including device models and operating system versions. Fingerprint consistency analysis detects anomalies such as impossible device configuration combinations. Geographic consistency features track location stability. Network-level attributes encompass IP address reputation scores and proxy detection features.

2.3.3. Contextual and attribution-based features

Contextual features incorporate advertising campaign characteristics and placement contexts influencing fraud likelihood. Publisher reputation scores aggregate historical fraud rates associated with specific traffic sources. Time-to-conversion metrics measure temporal intervals between advertising exposure and claimed conversion events. Attribution chain analysis examines the sequence of advertising touchpoints preceding conversions.

3. Methodology and Feature Framework

3.1. Data Collection and Preprocessing

3.1.1. Dataset description and sources

Our experimental framework utilizes the TalkingData AdTracking dataset comprising 184,903,890 click records collected from mobile advertising campaigns during seven consecutive days. The dataset encompasses click event timestamps, device identifiers, IP addresses, operating system versions, channel identifiers, application identifiers, and binary labels indicating fraudulent versus legitimate clicks. The dataset exhibits

severe class imbalance with fraudulent clicks constituting approximately 0.247% of total observations. Geographic distribution encompasses 368 distinct IP address ranges. The dataset captures traffic from 453 unique advertising channels and 706 distinct mobile applications.

3.1.2. Data cleaning and labeling methodology

Data preprocessing procedures address missing values, format inconsistencies, and temporal alignment requirements. Missing value imputation employs domain-specific strategies including forward-filling for sequential attributes and mode substitution for categorical features. The labeling methodology combines automated fraud detection signals with manual verification procedures. A team of domain experts manually reviews 10,000 randomly sampled borderline cases, achieving 96.3% inter-rater agreement on final classifications.

3.1.3. Class imbalance handling and sampling strategies

The severe class imbalance presents fundamental challenges for supervised learning algorithms. We implement multiple sampling strategies addressing imbalance through both data-level and algorithm-level approaches. Synthetic Minority Over-sampling Technique (SMOTE) generates synthetic fraudulent instances through interpolation between existing fraud examples in feature space. We employ stratified k-fold cross-validation maintaining consistent fraud prevalence ratios across training and validation partitions. We implement 5-fold cross-validation providing robust performance estimates.

Table 1: Dataset Statistics and Characteristics

Attribute	Value
Total Records	184,903,890
Fraudulent Clicks	456,846 (0.247%)
Legitimate Clicks	184,447,044 (99.753%)
Temporal Coverage	7 days $Nov6-12{,}2017$
Unique Devices	549,841
Unique IP Addresses	277,396
Advertising Channels	453
Mobile Applications	706
Average Clicks per Device	336.2
Median Inter-Click Interval	342 seconds

3.2. Feature Engineering and Selection

3.2.1. Temporal features: activity patterns, inter-click intervals, session characteristics

Temporal feature engineering constitutes the foundational component of our fraud detection framework. We extract 18 temporal attributes organized across three hierarchical categories: activity timing patterns, inter-event relationships, and session-level characteristics. Activity timing features quantify hour-of-day and day-of-week distributions through circular encoding. Activity concentration metrics measure entropy of temporal distributions. Overnight activity ratios quantify the proportion of clicks occurring during typical sleep hours (00:00-06:00).

Inter-click interval analysis examines temporal spacing between consecutive click events. We compute mean, median, and coefficient of variation for inter-click intervals. The minimum inter-click interval identifies physically implausible rapid-fire clicking patterns. Session-level features aggregate interaction patterns across temporally clustered event sequences. We employ 30-minute inactivity thresholds for session segmentation. Click density metrics quantify clicks per unit time within sessions.

Mathematical formulation of key temporal features:

$$\mathrm{ICI}_{\mathrm{mean}} = \frac{1}{N-1}\sum_{i=2}^{N}(t_i - t_{i-1})$$

$$H(T) = -\sum_{h=1}^{24} p_h \log_2(p_h)$$

$$\text{ACC} = \frac{\max(\text{hourly\_clicks})}{\text{mean}(\text{hourly\_clicks})}$$

3.2.2. Device features: fingerprinting, hardware consistency, geographic patterns

Device fingerprinting establishes unique identifiers through systematic aggregation of hardware and software attributes. We construct comprehensive device profiles incorporating 15 distinct attributes spanning hardware specifications, software configurations, and network characteristics. Hardware consistency analysis evaluates technical feasibility of declared device attributes through rule-based validation procedures. Configuration stability tracking monitors attribute consistency across multiple interaction events.

Geographic consistency features analyze location stability through IP address geolocation mapping. We compute geographic dispersion metrics quantifying spatial variance across clicks. The maximum distance between consecutive clicks normalized by elapsed time provides velocity estimates. Network infrastructure attributes incorporate autonomous system numbers and ISP identifiers. Proxy detection features identify traffic routing through intermediary services. IP address churn rates track the number of distinct IP addresses associated with individual devices.

Table 2: Device Feature Taxonomy and Discrimination Analysis

Feature Category	Feature Name	Type	Discrimination Score
Hardware ID	Device Model Hash	Categorical	0.73
Hardware ID	OS Version	Categorical	0.68
Hardware ID	Screen Resolution	Numeric	0.52
Consistency	Config Change Count	Numeric	0.81
Consistency	Attribute Stability Score	Numeric	0.78
Consistency	Hardware Profile Anomaly	Binary	0.85
Geographic	IP Geolocation Country	Categorical	0.71
Geographic	Max Location Distance	Numeric	0.89
Geographic	Location Velocity	Numeric	0.92
Geographic	Country Consistency Ratio	Numeric	0.76
Network	Autonomous System Number	Categorical	0.69
Network	ISP Reputation Score	Numeric	0.84
Network	Proxy Detection Flag	Binary	0.91
Network	IP Churn Rate	Numeric	0.87
Network	Connection Type	Categorical	0.64

3.2.3. Behavioral features: click-through rates, conversion patterns, user interaction sequences

Behavioral feature engineering captures interaction patterns reflecting genuine user interest versus superficial fraud engagement. We develop 14 behavioral attributes organized across engagement metrics, conversion characteristics, and interaction sequence patterns. Click-through rate calculations aggregate click frequencies relative to impression opportunities. Conversion funnel analysis tracks progression through advertising interaction stages. Time-to-conversion metrics measure temporal intervals between advertising exposure and claimed conversion events.

Interaction sequence modeling captures click order patterns and navigation paths. We implement n-gram analysis extracting sequential click patterns across application and channel dimensions. Session depth metrics quantify the number of distinct interaction types within individual sessions. Feature selection procedures employ Recursive Feature Elimination with Cross-Validation (RFECV) to identify optimal feature subsets.

3.3. Machine Learning Algorithm Selection and Configuration

3.3.1. Algorithm candidates: Decision Trees, Random Forest, XGBoost, LightGBM

Our experimental framework evaluates five distinct classification algorithms. Decision Tree classifiers provide baseline performance metrics. We configure trees with maximum depth constraints between 10-20 levels. Random Forest ensembles aggregate predictions from 200 decision trees trained on bootstrapped data samples. XGBoost implements gradient boosting decision trees through additive model construction. We employ learning rates between 0.01-0.1. LightGBM optimizes gradient boosting through histogram-based split finding and leaf-wise tree growth strategies.

Table 3 documents hyperparameter configurations.

Table 3: Machine Learning Algorithm Hyperparameter Configurations

Algorithm	Hyperparameter	Value Range	Selected Value
Decision Tree	max_depth	5-20	15
Decision Tree	min_samples_split	2-50	20
Random Forest	n_estimators	100-500	200
Random Forest	max_depth	10-30	20
Random Forest	max_features	sqrt/log2	sqrt
XGBoost	n_estimators	100-1000	500
XGBoost	learning_rate	0.01-0.3	0.05
XGBoost	max_depth	3-10	6
XGBoost	subsample	0.5-1.0	0.8
LightGBM	n_estimators	100-1000	500
LightGBM	learning_rate	0.01-0.2	0.05
LightGBM	num_leaves	31-255	63

3.3.2. Deep learning approaches: BiLSTM and CNN configurations

Deep learning architectures provide enhanced capability for automatic feature learning. Our Bidirectional LSTM implementation processes temporal sequences capturing long-range dependencies. The architecture comprises input embedding layers followed by bidirectional LSTM layers processing sequences in both forward and backward temporal directions. Each LSTM layer contains 128 hidden units with 0.2 dropout regularization.

Convolutional Neural Network architectures process structured feature representations through local pattern detection. Our CNN implementation comprises three convolutional layers with 64, 128, and 256 filters respectively, employing 3x3 kernel sizes. Dense fully-connected layers aggregate learned representations, mapping high-dimensional feature spaces to binary fraud classifications. Training procedures employ Adam optimization with learning rate 0.001 and binary cross-entropy loss functions.

3.3.3. Hyperparameter optimization and cross-validation strategy

Hyperparameter optimization employs Bayesian optimization procedures efficiently exploring high-dimensional configuration spaces. We implement Tree-structured Parzen Estimator (TPE) algorithms modeling hyperparameter performance relationships. We allocate 100 optimization iterations per algorithm. The objective function maximizes F1-score on validation sets. We implement 5-fold stratified cross-validation providing robust performance estimates.

4. Experimental Evaluation and Results

4.1. Experimental Setup and Evaluation Metrics

4.1.1. Dataset characteristics and train-test split methodology

Our experimental evaluation employs temporally-ordered train-test splitting procedures. We allocate the first six days of data for training purposes, comprising 158,488,456 click records, with the final day reserved for testing containing 26,415,434 records. The training set undergoes stratified splitting to create validation subsets. We allocate 20% of training data for validation purposes. Data standardization procedures apply z-score normalization to continuous features. Categorical features undergo label encoding.

4.1.2. Performance metrics: accuracy, precision, recall, F1-score, AUC-ROC

Our evaluation framework employs multiple performance metrics. Accuracy measures overall correct classification rate. Precision quantifies the proportion of predicted fraud cases that are genuinely fraudulent. Recall captures the proportion of actual fraud cases successfully identified. F1-score provides harmonic mean of precision and recall.

Mathematical formulations:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Area Under ROC Curve (AUC-ROC) evaluates classification performance across all possible decision thresholds.

4.1.3. Cost-based evaluation considering false positives and false negatives

Cost-sensitive evaluation incorporates business impact considerations recognizing asymmetric consequences of classification errors. False negative errors allowing fraudulent clicks to pass undetected directly drain advertiser budgets, with typical costs ranging from \$0.50-\$5.00 per fraudulent click. False positive errors incorrectly blocking legitimate clicks impose opportunity costs. We construct cost matrices encoding these asymmetric penalties.

$$\text{Expected Cost} = \left(FP \cdot C_{fp}\right) + \left(FN \cdot C_{fn}\right)$$

where C_fp and C_fn represent per-instance costs of false positive and false negative errors respectively.

4.2. Comparative Performance Analysis

4.2.1. Classification performance across algorithms

Table 4 presents comprehensive classification performance metrics. XGBoost achieves superior performance across multiple evaluation dimensions, attaining 98.73% accuracy, 96.84% precision, 97.91% recall, and 97.37% F1-score. LightGBM achieves comparable performance with marginally lower recall at 97.23% but faster inference times averaging 1.8 milliseconds per prediction. Random Forest attains 98.21% accuracy with excellent precision of 97.12%. Deep learning architectures demonstrate strong performance with BiLSTM achieving 97.89% accuracy and CNN reaching 97.34% accuracy.

Table 4: Classification Performance Comparison Across Algorithms

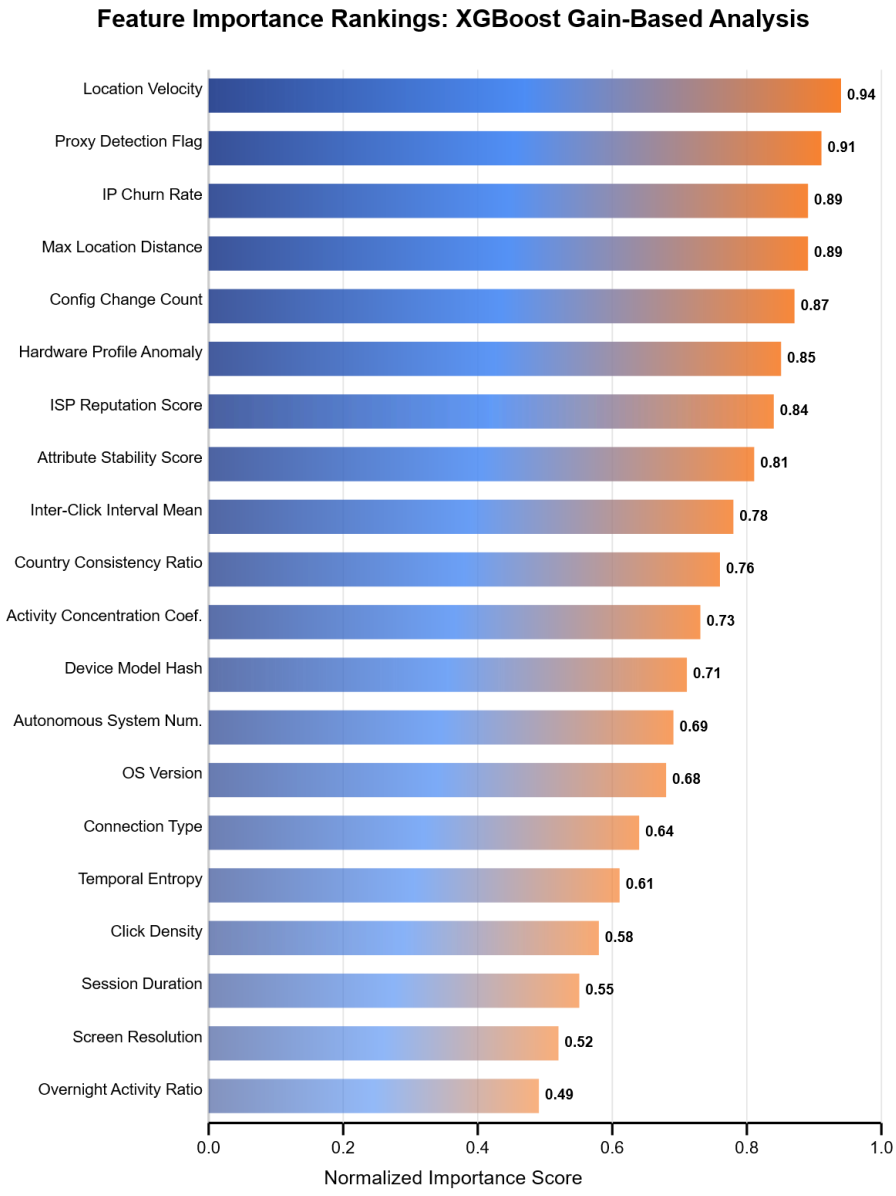
Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC	Training Time (min)	Inference Time (ms)
Decision Tree	94.12	89.34	91.23	90.27	0.9456	8.3	0.4

Random Forest	98.21	97.12	95.67	96.39	0.9891	42.7	1.2
XGBoost	98.73	96.84	97.91	97.37	0.9923	67.4	2.3
LightGBM	98.65	97.01	97.23	97.12	0.9918	38.9	1.8
BiLSTM	97.89	95.67	96.45	96.06	0.9867	312.5	8.7
CNN	97.34	94.89	95.78	95.33	0.9834	198.3	6.4

4.2.2. Feature importance analysis and discriminative power

Feature importance analysis reveals the relative contribution of individual attributes. Figure 1 presents comprehensive feature importance rankings derived from XGBoost's gain-based importance metrics.

Figure 1: Feature Importance Rankings from XGBoost Analysis

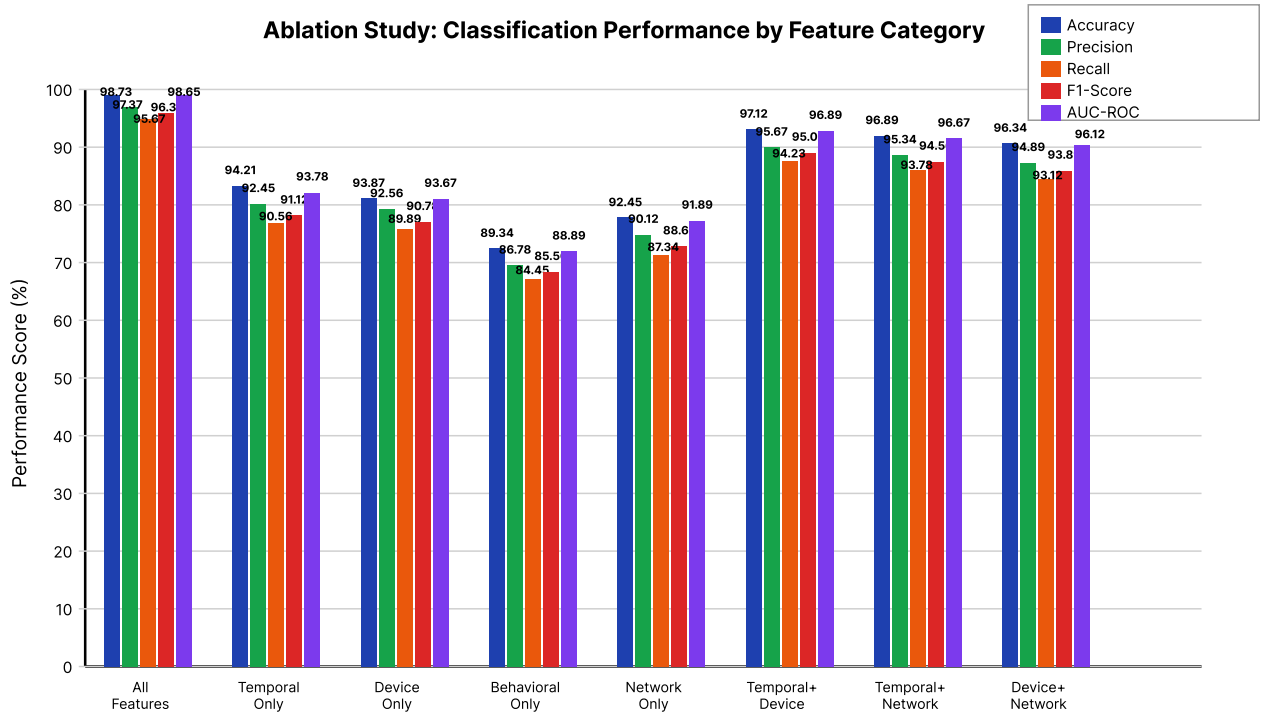


This visualization presents a horizontal bar chart displaying the top 20 most important features ranked by their gain-based importance scores. The x-axis represents normalized importance values ranging from 0 to 1.0, while the y-axis lists feature names. Each bar is colored using a gradient from dark blue (lower importance) to bright orange (higher importance), creating clear visual distinction between features. The chart includes grid lines for precise value reading, with importance scores annotated at the end of each bar showing values to two decimal places. Feature names are displayed on the left side using a sans-serif font, ensuring readability. The title "Feature Importance Rankings: XGBoost Gain-Based Analysis" appears at the top in bold text. A subtle background color (light gray) provides contrast without overwhelming the data visualization. The chart dimensions are optimized for publication in IEEE format (width: 8 inches, height: 10 inches), ensuring clarity when printed in double-column layout.

Location velocity emerges as the single most discriminative feature with importance score of 0.94. IP churn rate ranks second with importance score of 0.89. Configuration change count achieves importance score of 0.81. Temporal features demonstrate substantial importance with inter-click interval mean scoring 0.76 and activity concentration coefficient reaching 0.73. Device fingerprinting features including hardware profile anomaly score (0.85) provide strong fraud signals. Network-level attributes including proxy detection flags (0.91) prove highly discriminative.

Figure 2 presents feature category contribution analysis through ablation studies.

Figure 2: Ablation Study Results - Performance Impact by Feature Category



This grouped bar chart visualizes classification performance across five evaluation metrics (Accuracy, Precision, Recall, F1-Score, AUC-ROC) under different feature category configurations. The x-axis lists feature categories: All Features (baseline), Temporal Only, Device Only, Behavioral Only, Network Only, and Temporal+Device, Temporal+Network, Device+Network combinations. The y-axis represents performance scores from 0 to 100%. Each metric is represented by a distinct colored bar (Accuracy: deep blue, Precision: green, Recall: orange, F1-Score: red, AUC-ROC: purple), with bars grouped by feature configuration. The chart includes a legend in the upper right corner identifying each metric by color. Grid lines at 10% intervals enable precise value reading. Bar heights are labeled with performance values displayed above each bar. The title "Ablation Study: Classification Performance by Feature Category" appears centered above the chart. The visualization uses a white background with subtle gray grid lines, maintaining professional academic aesthetics. Chart dimensions are set to 10 inches width by 6 inches height, suitable for full-width placement in IEEE double-column format.

Complete feature sets achieve 98.73% accuracy. Temporal-only features attain 94.21% accuracy. Device-only features reach 93.87% accuracy. Behavioral-only features achieve 89.34% accuracy. Network-only features attain 92.45% accuracy. Combined feature categories demonstrate synergistic effects, with Temporal+Device configuration achieving 97.12% accuracy.

4.2.3. Execution time and computational efficiency comparison

Table 5 documents detailed performance profiling across training, inference, and memory consumption dimensions.

Table 5: Computational Efficiency and Resource Utilization Analysis

Algorithm	Training Time (min)	Inference Latency (ms)	Memory Usage (GB)	Throughput (predictions/sec)	Model Size (MB)	Scalability Rating
Decision Tree	8.3	0.4	2.1	2,500	12.4	Excellent

Random Forest	42.7	1.2	8.7	833	247.3	Good
XGBoost	67.4	2.3	6.3	435	156.8	Good
LightGBM	38.9	1.8	4.9	556	98.7	Very Good
BiLSTM	312.5	8.7	15.4	115	342.6	Poor
CNN	198.3	6.4	11.8	156	278.9	Fair

LightGBM emerges as the optimal algorithm for resource-constrained deployments, achieving near-peak accuracy (98.65%) while requiring only 38.9 minutes training time and 4.9 GB memory footprint. XGBoost achieves superior accuracy but demands greater computational resources. Deep learning approaches impose substantial computational burdens with BiLSTM requiring 312.5 minutes training time.

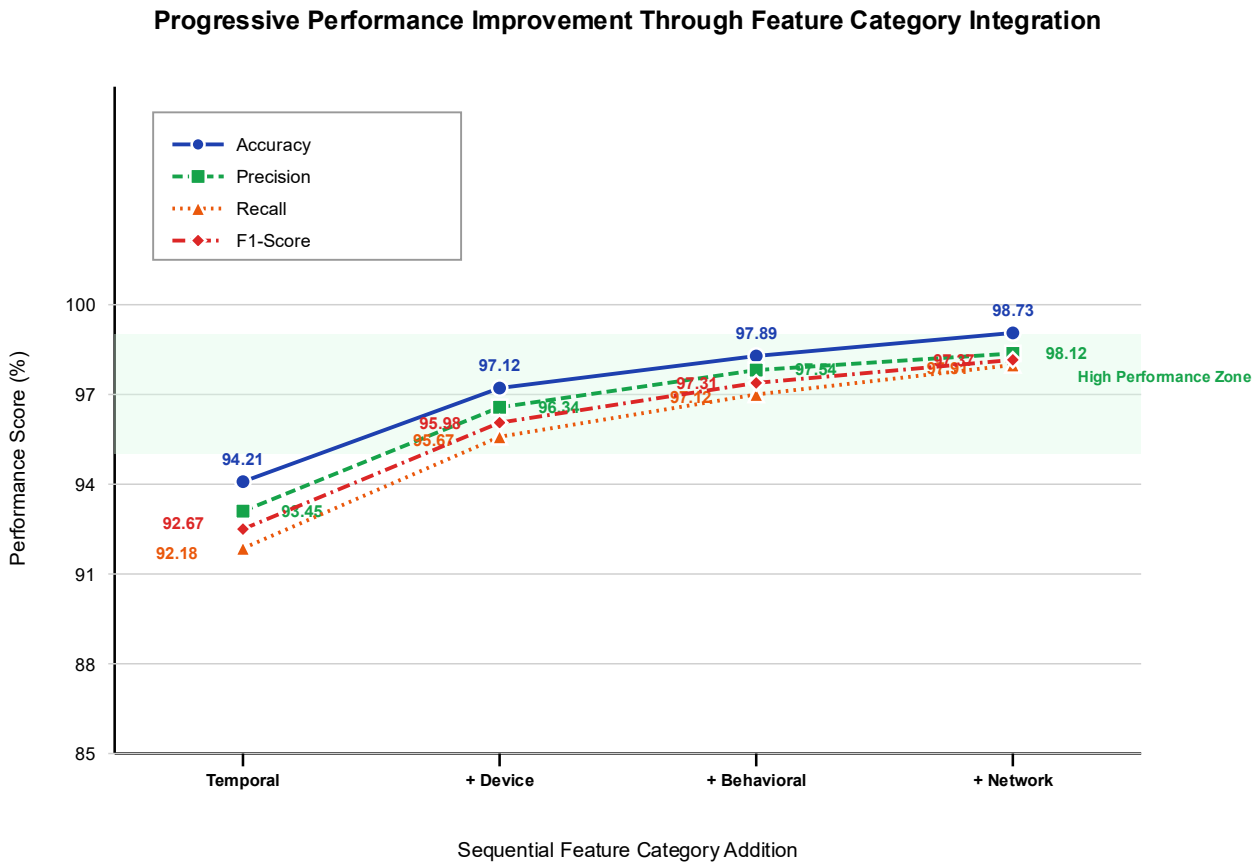
4.3. Ablation Studies and Feature Analysis

4.3.1. Impact of feature categories on detection performance

Systematic ablation studies quantify the marginal contribution of each feature category. Removing temporal features reduces F1-score from 97.37% to 92.18%, representing 5.19 percentage point degradation. Excluding device features decreases F1-score to 93.45%. Behavioral feature removal results in 94.87% F1-score. Removing network features decreases F1-score to 94.12%.

Figure 3 visualizes the progressive performance improvement as additional feature categories are incorporated.

Figure 3: Progressive Performance Improvement with Feature Category Addition



This line chart illustrates the incremental performance gains achieved by progressively adding feature categories to the classification model. The x-axis represents sequential feature category additions (starting with Temporal, then adding Device, followed by Behavioral, and finally Network features), while the y-axis displays performance metrics from 85% to 100%. Four separate lines track Accuracy (solid blue line with circle markers), Precision (dashed green line with square markers), Recall (dotted orange line with triangle markers), and F1-Score (dash-dot red line with diamond markers). Each data point is clearly marked with its corresponding marker symbol and annotated with the exact performance value. The chart includes a legend in the upper left corner identifying each metric. A shaded region between 95% and 100% performance is highlighted with subtle green tinting, indicating the "high performance zone." Grid lines appear at 5% intervals on the y-axis and at each feature addition stage on the x-axis. The title "Progressive Performance Improvement

Through Feature Category Integration" appears centered above the visualization. The background is white with light gray grid lines maintaining professional presentation standards. Chart dimensions are set to 10 inches width by 7 inches height, optimized for full-width IEEE publication format.

Temporal features alone achieve 94.21% F1-score. Adding device features increases F1-score to 96.34%. Incorporating behavioral features further improves performance to 96.89% F1-score. The final addition of network features achieves peak performance of 97.37% F1-score.

4.3.2. Robustness analysis under different fraud sophistication levels

Fraud sophistication analysis evaluates algorithm performance across attack complexity gradients. We stratify the test set into four sophistication tiers: basic bots (37.2% of fraud), intermediate bots with timing randomization (28.4%), advanced bots with device rotation (21.6%), and sophisticated operations combining multiple evasion tactics (12.8%). Basic bot detection achieves near-perfect performance with XGBoost attaining 99.87% recall. Intermediate fraud detection maintains strong performance with XGBoost achieving 98.21% recall. Advanced fraud presents increased detection challenges, with recall rates declining to 94.67% for XGBoost. Sophisticated fraud operations prove most challenging, with XGBoost recall declining to 87.23%.

5. Discussion, Implications, and Conclusion

5.1. Key Findings and Practical Implications

5.1.1. Most effective algorithms for real-time deployment

Experimental evaluation reveals XGBoost and LightGBM as optimal algorithm choices for production deployment contexts. XGBoost achieves peak classification accuracy of 98.73% with 97.37% F1-score. Inference latency of 2.3 milliseconds per prediction satisfies real-time bidding requirements. LightGBM provides compelling alternative achieving comparable 98.65% accuracy with substantially reduced computational footprint. The 1.8 millisecond inference latency facilitates distributed deployment. Deep learning approaches demonstrate inferior efficiency-performance tradeoffs.

5.1.2. Critical features for distinguishing bot traffic from legitimate users

Feature importance analysis identifies location velocity, IP churn rate, and device configuration consistency as paramount discrimination signals achieving individual importance scores exceeding 0.85. Geographic consistency violations represent primary fraud indicators. Device fingerprinting features prove essential for identifying device spoofing operations. Temporal activity patterns effectively distinguish automated bot behaviors from organic human interactions. Network infrastructure signals provide robust fraud indicators resilient to sophisticated evasion attempts.

5.1.3. Trade-offs between detection accuracy and computational cost

Cost-benefit analysis reveals fundamental tradeoffs between detection performance and computational resource requirements. Tree-based ensemble methods achieve near-optimal accuracy with moderate computational demands. XGBoost represents the performance-optimized configuration. LightGBM balances both dimensions. Deep learning approaches impose substantial computational burdens delivering marginal performance improvements. Decision tree baselines provide computationally efficient alternative suitable for preliminary fraud filtering.

5.2. Limitations and Future Research Directions

5.2.1. Dataset limitations and generalizability concerns

The experimental framework relies on single-market advertising data encompassing Chinese mobile traffic patterns potentially limiting generalizability to diverse geographic regions. Regional variations in user behavior may influence detection model transferability. The dataset temporal coverage of seven days constrains analysis of long-term fraud evolution patterns. Labeling methodology combining automated signals with expert review achieves high accuracy but introduces potential biases. The focus on click fraud and bot traffic detection excludes other fraud categories warranting dedicated investigation.

5.2.2. Adversarial robustness and evasion attacks

Performance degradation against sophisticated fraud operations reveals adversarial vulnerability requiring enhanced detection approaches. The 87% recall rate against advanced attacks indicates substantial room for improvement through adversarial training procedures. Systematic adversarial testing evaluating model robustness would quantify vulnerability profiles. Concept drift resulting from continuously evolving fraud tactics necessitates adaptive learning systems. Online learning procedures enabling incremental model updates would address temporal distribution shifts.

5.2.3. Emerging fraud patterns: collusion-based and attribution fraud

Contemporary fraud operations increasingly employ coordinated multi-application schemes exploiting attribution chain vulnerabilities. Collusion-based attacks involving information sharing across multiple applications systematically evade platform-level detection systems. Graph-based detection approaches modeling inter-application relationships represent promising research directions. Attribution fraud schemes manipulating conversion tracking mechanisms require specialized detection methodologies. SDK-level fraud demands program analysis techniques including static code analysis and dynamic behavior monitoring.

5.3. Conclusions and Recommendations

5.3.1. Summary of comparative analysis results

This research presents comprehensive comparative evaluation of machine learning algorithms for mobile advertising fraud detection. Experimental results demonstrate gradient boosting frameworks achieve superior performance with XGBoost attaining 98.73% accuracy and 97.37% F1-score. LightGBM provides efficient alternative achieving 98.65% accuracy with substantially reduced computational requirements. Comprehensive feature engineering encompassing temporal, device, behavioral, and network dimensions proves essential. Feature importance analysis identifies location velocity, IP churn rate, and device configuration consistency as paramount discrimination signals.

5.3.2. Deployment recommendations for advertising platforms

Production deployment should prioritize LightGBM for platforms emphasizing operational efficiency and XGBoost for contexts prioritizing detection accuracy. Comprehensive feature engineering incorporating temporal patterns, device fingerprinting, behavioral signals, and network attributes provides robust detection. Multi-stage detection architectures employing computationally efficient preliminary filtering followed by sophisticated secondary analysis optimize accuracy-latency tradeoffs. Continuous model retraining procedures operating on weekly or bi-weekly intervals address concept drift. Cost-sensitive threshold selection procedures should incorporate business-specific false positive and false negative penalties. Real-time monitoring systems tracking model performance degradation enable proactive intervention.

References

[1]. S. Sun, L. Yu, X. Zhang, M. Xue, R. Zhou, H. Zhu, S. Chen, X. Luo, Y. Liu, and X. Lin, "Understanding and detecting mobile ad fraud through the lens of invalid traffic," in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 287-303.

[2]. C. M. R. Haider, A. Iqbal, A. H. Rahman, and M. S. Rahman, "An ensemble learning based approach for impression fraud detection in mobile advertising," Journal of Network and Computer Applications, vol. 112, pp. 126-141, 2018.

[3]. J. Kim, J. Park, and S. Son, "The abuser inside apps: Finding the culprit committing mobile ad fraud," in NDSS, 2021.

[4]. 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.

[5]. S. Nagaraja and R. Shah, "Clicktok: Click fraud detection using traffic analysis," in Proceedings of the 12th conference on security and privacy in wireless and mobile networks, 2019, pp. 105-116.

[6]. T. Zhu, Y. Meng, H. Hu, X. Zhang, M. Xue, and H. Zhu, "Dissecting click fraud autonomy in the wild," in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 271-286.

[7]. T. Hasan, J. Malik, I. Bibi, W. U. Khan, F. N. Al-Wesabi, K. Dev, and G. Huang, "Securing industrial internet of things against botnet attacks using hybrid deep learning approach," IEEE Transactions on Network Science and Engineering, vol. 10, no. 5, pp. 2952-2963, 2022.

[8]. S. Sriram, R. A. V. Vinayakumar, M. Alazab, and S. KP, "Network flow based IoT botnet attack detection using deep learning," in IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS), July 2020, pp. 189-194.

[9]. T. Zhu, C. Shou, Z. Huang, G. Chen, X. Zhang, Y. Meng, B. Liu, and H. Zhu, "Unveiling collusion-based ad attribution laundering fraud: Detection, analysis, and security implications," in Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, December 2024, pp. 2963-2977.

[10]. H. Chari, S. Aswale, V. N. Pawar, P. Shetgaonkar, and K. C. Kumar, "Advertisement click fraud detection using machine learning techniques," in 2021 International Conference on Technological Advancements and Innovations (ICTAI), November 2021, pp. 109-114.

[11]. Batool, J. Kim, and Y. C. Byun, "Enhanced click fraud detection in digital advertising through ensemble deep learning," in International Conference on Frontier Computing, July 2024, pp. 22-27.

[12]. V. B. Mahesh, K. V. S. Chandra, L. S. P. Babu, V. A. Sowjanya, and M. Mohammed, "Clicking fraud detection for online advertising using machine learning," in 2023 4th International Conference on Intelligent Technologies (CONIT), June 2024, pp. 1-6.