Ingénierie des Systèmes d'Information
Vol. 31, No. 5, May, 2026, pp. 1575-1587
Journal homepage: http://iieta.org/journals/isi

CTGAN–ENN Resampling for Imbalanced Electricity Theft Detection: Generalization and Computational Trade-offs

Ali Tareq Radeef[1*], Salman Yussof[1,2], Saja Jabbar Mohammed[3], Moamin A. Mahmoud[1,2]

[1] College of Computing and Informatics, Universiti Tenaga Nasional, Kajang 43000, Selangor, Malaysia
[2] Institute of Informatics and Computing in Energy (IICE), Universiti Tenaga Nasional, Kajang 43000, Selangor, Malaysia
[3] Department of Computer Engineering Techniques, College of Technical Engineering, University of Al-Maarif, Al Anbar 31001, Iraq

Corresponding Author Email: ali.tareq@uoa.edu.iq

https://doi.org/10.18280/isi.310515

Received: 22 February 2026
Revised: 28 April 2026
Accepted: 5 May 2026
Available online: 31 May 2026

Keywords:
electricity theft detection, non-technical losses, smart meters, class imbalance, Conditional Tabular Generative Adversarial Network, Edited Nearest Neighbors, XGBoost, computational complexity

ABSTRACT

Electricity theft detection (ETD) from smart-meter data is hindered by severe class imbalance, noisy class boundaries, and the computational burden of high-capacity classifiers. This study proposes a Conditional Tabular Generative Adversarial Network–Edited Nearest Neighbours (CTGAN–ENN) resampling framework for imbalanced ETD in tabular consumption data. Conditional Tabular Generative Adversarial Networks are used to generate minority-class theft samples by learning the conditional joint distribution of engineered consumption features, while Edited Nearest Neighbors removes ambiguous majority-class instances near the decision boundary. The framework is coupled with stability-aware feature selection, robust scaling, regularized classifiers, threshold tuning, and explicit complexity profiling. Experiments were conducted on the State Grid Corporation of China dataset, containing 42,372 consumers with a 10.7:1 normal-to-theft imbalance ratio. Five classifiers—XGBoost, LightGBM, Random Forest, an Artificial Neural Network (ANN), and a Convolutional Neural Network (CNN)-Attention model—were evaluated using hold-out testing and stratified five-fold cross-validation. XGBoost achieved the best composite selection score, with test Area Under the Curve (AUC) = 0.9078, PR-AUC = 0.6529, F1 = 0.5433, recall = 0.6902, and balanced accuracy = 0.8054. LightGBM obtained the same test AUC and higher recall of 0.7178 with markedly lower computational cost. In contrast, the CNN-Attention model failed to generalize, confirming that attention-based architectures are not necessarily suitable for fixed tabular feature vectors. The results show that CTGAN–ENN, combined with feature selection and regularized tree-based classifiers, offers a practical accuracy–efficiency trade-off for large-scale electricity theft screening.

1. INTRODUCTION

Electricity theft is a persistent global problem that causes substantial annual losses to power utilities, undermines grid stability, and imposes costs that are ultimately borne by legitimate consumers [1, 2]. Worldwide, non-technical losses (NTL) attributed to electricity theft and meter manipulation are estimated to drain tens of billions of dollars from electric utilities every year, with the largest share borne by developing economies and by utilities operating without modern metering infrastructure [2, 3]. The proliferation of smart metering infrastructure and Advanced Metering Infrastructure (AMI) has generated large-scale consumption datasets that enable data-driven theft detection [1, 4, 5]. Machine learning (ML) and deep learning (DL) classifiers trained on these data can in principle identify anomalous consumption profiles indicative of meter tampering, meter bypass, or illegal connections [3, 5, 6].

Although previous electricity theft detection (ETD) studies have reported promising results, three interrelated problems continue to limit real-world deployment.

First, and most fundamentally, electricity consumption datasets are heavily imbalanced: NTL users account for fewer than 10% of records in benchmark datasets such as the State Grid Corporation of China (SGCC) dataset. Standard classifiers trained on such data are biased toward the majority (normal) class, yielding high overall accuracy but poor recall for the minority (theft) class [7-9].

Second, raw smart meter data contain hundreds of consecutive daily readings together with a substantial proportion of missing values, producing a high-dimensional feature space in which deep and ensemble models are prone to overfitting; the trained classifier memorises training-set patterns rather than learning generalisable behavioural signatures of theft [10-12].

Third, many deep-learning architectures require substantial computational resources during training and inference. This requirement may limit their practical deployment in large-scale smart-grid environments involving millions of smart meters [13, 14].

Previous studies have addressed class imbalance using oversampling (SMOTE, ADASYN, Borderline-SMOTE), undersampling (Tomek Links, Edited Nearest Neighbours (ENN)), or hybrid strategies [8, 9, 15, 16]. Generative methods such as Conditional Variational Autoencoders (CVAE) and Wasserstein GANs (WGAN, BiWGAN) have shown promising performance over linear interpolation methods for tabular data because they can model complex, multi-modal distributions and conditional dependencies between consumption features [17-19]. However, no prior ETD study has combined Conditional Tabular Generative Adversarial Network (CTGAN)-based oversampling with ENN-based boundary cleaning while simultaneously reporting an explicit analysis of overfitting and computational complexity for several competing classifiers.

This paper makes three contributions:

•We propose a Conditional Tabular Generative Adversarial Network–Edited Nearest Neighbours (CTGAN–ENN) hybrid resampling strategy that targets the imbalance problem with realistic synthetic minority samples and mitigates overfitting through targeted boundary cleaning. We explain what each component adds that existing methods (SMOTE-ENN, ADASYN, SMOTE-Tomek) cannot provide.

•We pair the resampling strategy with strong anti-overfitting measures (L2 regularisation, low learning rates, subsampling, dropout, batch normalisation, early stopping, and a model-selection criterion that explicitly penalises the train–validation gap) and evaluate overfitting through both train–validation and train–test generalisation gaps for every model.

•We report both theoretical Big-O complexity and measured wall-clock training and inference times for all five classifiers, enabling a principled trade-off between accuracy and computational cost for large-scale deployment.

The remainder of the paper is organized as follows. Section 2 reviews related work, organized by resampling strategy. Section 3 states the research problem and gap. Section 4 describes the methodology. Section 5 presents and discusses experimental results. Section 6 concludes.

2. RELATED WORK

We organize prior ETD studies into four categories: (i) oversampling-based, (ii) undersampling-based, (iii) hybrid approaches, and (iv) studies that do not address class imbalance. Table 1 summarizes the key works.

2.1 Oversampling-based approaches

Le et al. [20] combined a convolutional autoencoder with a Transformer encoder for feature extraction from 28-day consumption windows and used K-Means SMOTE to balance the SGCC dataset. Yu et al. [21] proposed a dual-attention Convolutional Neural Network (CNN) with channel attention and frequency-spatial attention modules, also using SMOTE on the Irish and SGCC datasets. Ullah et al. [22] combined CNN and Gated Recurrent Unit (GRU) with Particle Swarm Optimization for hyperparameter tuning and used oversampling to address class imbalance. Qu et al. [23] introduced an improved SMOTE variant that incorporates a density-based weighting scheme and pairs it with a Random Forest classifier, reporting that the targeted oversampling reduces the noise injected by standard SMOTE in borderline regions. Gong et al. [18] used a CVAE to generate theft samples that respect the conditional consumption distribution, reporting improved Area Under the Curve (AUC) and F1 over SMOTE on a Chinese utility dataset. Li et al. [19] proposed a WGAN tailored to distributed photovoltaic (PV) electricity theft to mitigate the scarcity of theft labels in renewable-rich grids and showed that adversarial synthesis outperforms SMOTE under that setting.

Table 1. Summary of related electricity theft detection (ETD) studies

Ref.	Year	Dataset	Method	Imbalance Handling	Category	Limitation
[22]	2020	SGCC	CNN-GRU-PSO	Oversampling	Oversampling	Overfitting not analysed; no complexity analysis
[8]	2020	SGCC	LSTM-UNet-AdaBoost	IQMOT	Hybrid	Very high complexity; no boundary cleaning
[24]	2021	SGCC + UMass	DE-RUSBoost / Jaya-RUSBoost	Random under-sampling (RUSBoost)	Undersampling	Discards informative majority samples; metaheuristic training overhead
[10]	2021	SGCC	CNN + BM	None	No handling	Accuracy misleading under imbalance
[15]	2023	SGCC	FractalNet + LightGBM	HOUBC	Hybrid	High computational complexity; single-run results only
[17]	2023	SGCC	VAE-GAN	VAE-GAN synthesis	Hybrid	No reproducibility details; no complexity analysis
[25]	2025	SGCC	Adaptive DL	None	No handling	Systematic majority-class bias
[20]	2025	SGCC	CAE + Transformer	K-Means SMOTE	Oversampling	Limited to 28-day windows; high computational cost
[21]	2026	SGCC/ Irish	Dual-Attention CNN	SMOTE	Oversampling	SMOTE linear interpolation may produce implausible samples
[26]	2026	Irish (CER/ISSDA)	Ensemble + LLM-guided tuning	Bayesian Detection Rate (BDR) + sample-weighted loss	Cost-sensitive (no resampling)	External LLM dependency; no Big-O complexity analysis; not validated on SGCC
[27]	2026	UCI + Ausgrid	Meta-classifier on LSTM/XGB forecasts	Acknowledged but no resampling	Forecasting-based (no handling)	Performance depends on quality of upstream forecasting model; no SGCC validation

Note: SGCC = State Grid Corporation of China; UMass = University of Massachusetts; CNN = Convolutional Neural Network; GRU = Gated Recurrent Unit; PSO = Particle Swarm Optimization; LSTM = Long Short-Term Memory; VAE = Variational Autoencoder; GAN = Generative Adversarial Network; DL = Deep Learning; ISSDA = Irish Social Science Data Archive.

A key limitation of SMOTE and its variants is that they generate synthetic samples by linear interpolation between feature vectors in the original space. For tabular electricity consumption data with complex, non-linear inter-feature dependencies and mixed distributions, this can produce implausible samples that worsen the decision boundary, encouraging the model to overfit to artefacts of the interpolation rather than to true theft behaviour [11, 23]. Recent generative alternatives such as CVAE [18] and WGAN [19] partially mitigate this limitation by modelling the conditional minority-class distribution rather than relying on convex combinations in feature space.

Although SMOTE and ADASYN are widely used in ETD, both rely on assumptions that may not hold for high-dimensional consumption data. SMOTE generates samples through linear interpolation, while ADASYN focuses synthesis near the decision boundary, potentially amplifying noise. Recent generative methods such as CVAE, WGAN, and BiWGAN address these limitations by learning the underlying minority-class distribution. CTGAN follows this direction by modelling the joint conditional distribution of tabular features, which motivates its use in this study.

2.2 Undersampling-based approaches

Undersampling methods reduce the majority class to balance training data. ENN [7] removes majority-class samples whose k nearest neighbors disagree with their label, cleaning the boundary region. Tomek Links remove borderline majority-class points. A particularly popular family of undersampling approaches in ETD is Random Under-Sampling Boost (RUSBoost), which integrates random undersampling of the majority class with the AdaBoost ensemble framework. Mujeeb et al. [24] proposed two enhanced classifiers, DE-RUSBoost and Jaya-RUSBoost, in which RUSBoost hyperparameters are optimised by differential evolution and the Jaya metaheuristic, respectively. Evaluated on the SGCC dataset together with the UMass smart homes dataset, DE-RUSBoost achieved an AUC of 0.89 and Jaya-RUSBoost an AUC of 0.95, outperforming Weighted Adaptive Deep Convolutional Neural Network (WADCNN) and grid-search RUSBoost. While these results confirm the value of integrating undersampling with boosting, both methods rely on metaheuristic search loops that increase training cost and remain exposed to the central weakness of any random undersampling scheme: informative majority-class consumers may be discarded, which narrows the diversity of normal behavioural patterns the classifier can learn. More generally, pure undersampling is not competitive when the dataset is severely imbalanced, especially when the discarded majority samples carry diagnostic information about borderline behaviour.

2.3 Hybrid approaches

Hybrid methods combine over- and undersampling to exploit the complementary strengths of both. Naeem et al. [15] proposed hybrid oversampling and undersampling using both classes (HOUBC) combined with a deep FractalNet and LightGBM, outperforming pure oversampling baselines on AUC, PR-AUC, and Matthews Correlation Coefficient (MCC) on SGCC, but at the cost of a complex multi-stage deep architecture. Aslam et al. [8] proposed the interquartile minority oversampling technique (IQMOT) (a custom hybrid resampling method) combined with DL and ensemble classifiers for SGCC, achieving strong AUC results but with high computational complexity. Sun et al. [17] used a Variational Autoencoder (VAE)-GAN for imbalanced ETD on SGCC, showing that GAN-based synthesis can outperform SMOTE for high-dimensional tabular data.

These approaches demonstrate that hybrid resampling outperforms single-strategy methods. However, none employ CTGAN for minority synthesis specifically while also studying overfitting and computational cost in a unified framework. SMOTE-ENN, the closest competitor to our CTGAN-ENN, relies on linear interpolation and is therefore less suited to the complex, conditional distribution of electricity consumption patterns. GAN-based hybrids can better capture the minority manifold but often increase computational cost and rarely report train–test gaps to quantify residual overfitting.

2.4 Studies not addressing class imbalance

Several studies omit explicit imbalance handling. Ibrahim et al. [10] used a CNN with the Blue Monkey algorithm on SGCC but relied solely on accuracy, a misleading metric under heavy imbalance. Sleiman et al. [25] proposed an adaptive DL architecture for ETD without addressing imbalance, resulting in a systematic bias toward the majority class and an inflated risk of overfitting on majority-class patterns. Zheng et al. [5] introduced an influential Wide-and-Deep CNN for ETD on the SGCC dataset and reported strong accuracy at the time of publication, but treated imbalance only implicitly through architectural design rather than through targeted resampling. Buzau et al. [3] proposed a hybrid deep neural network combining Multilayer Perceptron (MLP) and recurrent branches on a Spanish utility dataset; although it learns features directly from raw smart meter readings, the imbalanced training distribution lowered recall on the theft class. More recently, Zhang et al. [26] proposed an ensemble framework with Large Language Model (LLM)-guided hyperparameter tuning that minimises the false positive rate using a Bayesian Detection Rate (BDR) objective and sample-weighted loss on the Irish smart-meter dataset; while this avoids explicit resampling, the framework introduces an external LLM dependency and reports no Big-O complexity analysis. Altamimi et al. [27] introduced a meta-classifier paradigm that consumes the full time series of forecasts and residuals produced by upstream Long Short-Term Memory (LSTM)/gradient-boosting forecasters and was evaluated on the UCI ElectricityLoadDiagrams and Ausgrid datasets; the approach reuses existing forecasting infrastructure but its discriminative power is bounded by the quality of the upstream forecaster, and the method has not been validated on heavily imbalanced benchmarks such as SGCC.

3. PROBLEM STATEMENT AND RESEARCH GAP

Despite substantial progress, three critical gaps remain in the ETD literature.

First, class imbalance and boundary noise are typically treated as separate problems. Oversampling methods (SMOTE and its variants) generate new minority samples but do not remove noisy or ambiguous majority-class instances near the decision boundary. Undersampling methods (ENN, Tomek Links) clean the boundary but discard potentially informative

majority-class data. No prior ETD study has combined CTGAN's ability to model the complex conditional distribution of electricity consumption patterns with ENN's targeted noise removal. Specifically, SMOTE generates samples by linear interpolation between pairs of minority instances, which is unsuitable when the minority class occupies a non-convex, multi-modal region of feature space. ADASYN concentrates synthesis on the hardest-to-classify boundary region, which risks amplifying noise. CTGAN, by contrast, trains a conditional GAN to learn the joint distribution over all features simultaneously, producing synthetic samples that are statistically indistinguishable from real data. ENN then removes majority-class instances that are likely mislabeled or ambiguous, sharpening the boundary without discarding most majority data.

Second, overfitting is widely acknowledged but rarely quantified. Many ETD studies report only test-set accuracy or AUC and do not present the corresponding training-set values, so the train–test gap is used as an overfitting indicator, but its interpretation must consider the distributional shift introduced by resampling. Deep classifiers with millions of parameters trained on tens of thousands of consumers are particularly susceptible to memorising spurious patterns, especially after aggressive resampling. A study that simultaneously applies strong regularisation, reports the train–test gap for every model, and uses a model-selection criterion that explicitly penalises this gap is still missing in the ETD literature.

Third, computational complexity is largely ignored. Most published ETD systems propose increasingly deep architectures (transformers, multi-head attention CNNs, stacked recurrent networks) without reporting training time, inference latency, or theoretical Big-O complexity. For a real-world smart grid that must score millions of meters periodically, the gap between an accurate but slow model and an accurate, lightweight one is decisive. A side-by-side comparison of accuracy and computational cost across heterogeneous classifiers (ensemble trees, MLP, attention-based CNN) on the same dataset and pipeline has not previously been provided for ETD.

Taken together, these gaps motivate a study that (i) proposes CTGAN-ENN as a principled hybrid resampling strategy for electricity consumption data, (ii) quantifies and controls overfitting with explicit regularisation and train–test gap reporting, and (iii) reports both theoretical and measured computational complexity for every model considered.

4. RESEARCH METHODOLOGY

The proposed framework, referred to as CTGAN-ENN, is designed as an integrated pipeline that addresses the three target problems class imbalance, overfitting, and computational complexity within a single coherent workflow rather than as isolated techniques. The central idea is to combine three complementary mechanisms. First, a CTGAN is trained on the minority-class samples to learn the joint distribution of theft-related consumption patterns and to generate realistic synthetic instances that re-balance the training set without resorting to linear interpolation. Second, the ENN algorithm is applied to the re-balanced dataset to remove ambiguous majority-class samples whose neighbours disagree with their label, which sharpens the decision boundary and directly reduces overfitting caused by boundary noise. Third, this hybrid resampling step is paired with stability-aware feature selection (top-50 features), robust feature scaling, classifier-level regularisation (L2 penalties, dropout, batch normalisation, early stopping), and explicit complexity profiling, so that the final model is not only accurate but also generalisable and computationally lightweight.

Concretely, the framework is implemented in Python and executes eleven sequential phases, summarised graphically in Figure 1. The pipeline starts from the raw SGCC consumption matrix, derives 84 engineered behavioural features informed by the PRECON residential benchmark, applies a stratified train/validation/test split followed by stratified 5-fold cross-validation, performs in-fold feature selection, fits a RobustScaler, applies CTGAN-ENN resampling, trains five classifiers, tunes the decision threshold, and finally evaluates an extended suite of metrics together with measured training and inference times. The global random seed is fixed at 42 throughout for reproducibility.

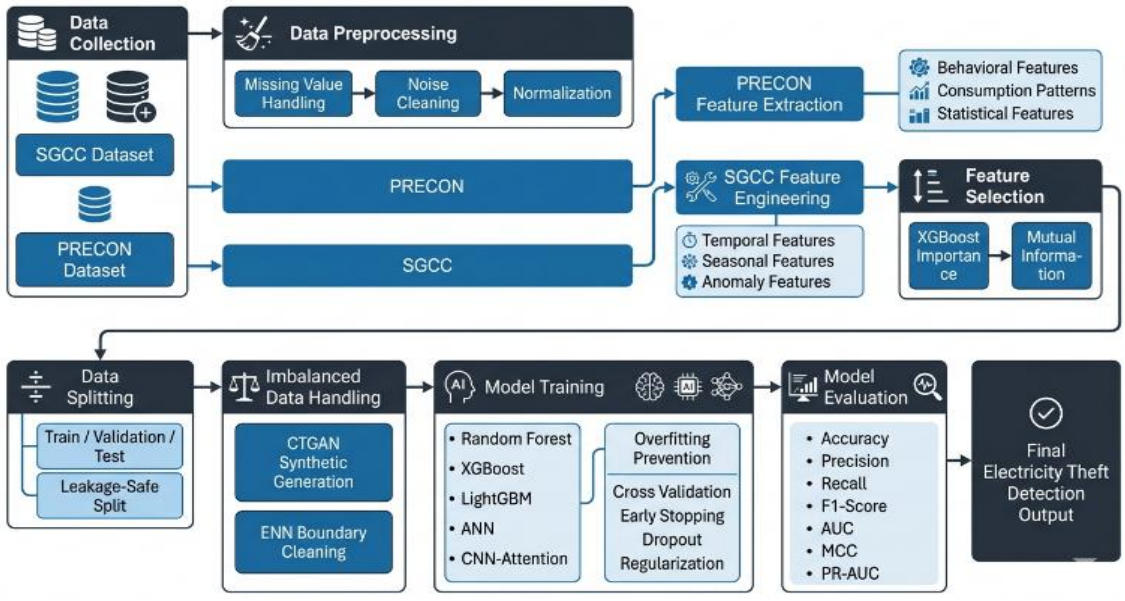

Figure 1. The Conditional Tabular Generative Adversarial Network–Edited Nearest Neighbours (CTGAN–ENN) pipeline for electricity theft detection

4.1 Dataset description

The SGCC dataset is loaded from a preprocessed and KNN-imputed data file. It contains daily electricity consumption records for 42,372 consumers spanning 1,035 consecutive reading days from January 2014 to September 2016. The binary label column FLAG encodes 0 for normal consumers (38,757; 91.47%) and 1 for non-technical loss (NTL/theft) consumers (3,615; 8.53%), yielding an imbalance ratio of approximately 10.7:1. Approximately 25% of raw values are missing. Table 2 presents a summary of the main characteristics of the SGCC dataset.

Table 2. State Grid Corporation of China (SGCC) dataset summary

Attribute	Value
Collection Period	January 2014-September 2016
Total Consumers	42,372
Normal Consumers (FLAG = 0)	38,757 (91.47%)
NTL Consumers (FLAG = 1)	3,615 (8.53%)
Consecutive Reading Days	1,035
Missing Values (≈)	25% of entries
Imbalance Ratio	10.7:1
Global Random Seed	42

Note: NTL = Non-Technical Losses.

4.2 Precon behavioural benchmark extraction

Before engineering SGCC features, a reference behavioural profile is extracted from real residential smart meter data (PRECON dataset). For each house, the behavioural feature extraction procedure computes 30+ behavioural statistics from the household electricity consumption measurements column, including mean, standard deviation, coefficient of variation (CV), skewness, kurtosis, peak-to-average ratio, load factor, day-to-night ratio, weekday/weekend ratio, anomaly percentage at 3σ, hourly profile CV, monthly profile CV, peak season ratio, and appliance-level ratios.

The resulting population-level benchmarks are used as reference statistics during SGCC feature engineering and serve as normative behavioural signals for feature construction. When no PRECON files are found, neutral default benchmarks are applied so the pipeline remains executable on the SGCC dataset alone.

4.3 Precon-informed State Grid Corporation of China feature engineering

Raw SGCC date columns are first parsed to extract month, day-of-week, quarter, weekend flag, and season labels. Row-wise mean imputation is then applied to each consumer's 1,035-day sequence: missing values are replaced by that consumer's own non-missing daily mean; if all values are missing, zero is substituted. The imputed feature matrix is converted to a lower-precision numeric format for efficiency.

The SGCC feature engineering procedure then constructs the following feature groups for each consumer: basic statistics (daily mean, std, min, max, sum, median, CV, range, IQR, skewness, kurtosis); quantile and load features (peak_95, offpeak_05, peak-to-average ratio, load factor); missing-value indicators; seasonal aggregates (winter, spring, summer, autumn means and stds, plus summer-to-winter ratio); quarterly aggregates (Q1–Q4); monthly means (month_01 through month_12); weekday/weekend aggregates; first- and second-order differencing features; rolling 30-day statistics; anomaly indicators at 3σ and 2.5σ; sudden change indicators; PRECON-informed deviation features; and monthly profile statistics.

4.4 Stratified data partitioning

The engineered feature matrix is partitioned using a stratified 60/20/20 split into three non-overlapping subsets:

- •Test set (20% of the full dataset): 8,475 consumers, held out until final evaluation.
- •Validation set (20% of the full dataset, equivalently 25% of the 80% train + validation pool): 8,476 consumers, used for threshold tuning and model selection.
- •Training set (60% of the full dataset): 25,421 consumers, used for feature selection fitting, scaler fitting, resampling, and final model training.

Stratification ensures that each split preserves the original 91.47%/8.53% class ratio. Preprocessing steps (feature selection, scaling, resampling) are fitted on the training subset of each fold and applied to the corresponding validation subset; the test set is left untouched until final evaluation. This is standard practice and is adopted here so that the reported overfitting and complexity numbers are not contaminated by trivial sources of bias.

4.5 Stability-aware feature selection

Feature selection is performed on the training set using the stability-aware feature selection procedure, which combines two complementary importance signals:

- •XGBoost importance: A regularised XGBoost classifier (n_estimators = 250, max_depth = 3, learning_rate = 0.05, subsample = 0.85, colsample_bytree = 0.85, reg_alpha = 0.5, reg_lambda = 2.0, eval_metric = 'auc') is fitted on the training set and feature importances are extracted.
- •Mutual information: Sklearn.feature_selection.mutual_info_classif is computed on the same training set with random_state = 42.

Both importance vectors are independently ranked in descending order. The average rank across both methods is used as the final selection criterion. The top 50 features by average ranking score (Top 50 selected features) are retained. This ensemble ranking is more stable than either method alone and critically for the present study limits the input dimensionality to 50 features, which directly bounds the computational complexity of every downstream classifier and reduces the risk of overfitting in high-capacity models.

4.6 Conditional Tabular Generative Adversarial Network–Edited Nearest Neighbours hybrid resampling

4.6.1 CTGAN-ENN implementation

The CTGAN-ENN resampling procedure operates on the training fold and the selected feature set as follows:

- •Class counts are first computed from the training fold. The minority class is then expanded until it reaches approximately 50% of the majority-class size (corresponding to an approximate 2:1 majority-to-minority ratio after resampling). Partial balancing is intentionally adopted to limit the number of synthetic samples, thereby reducing computational cost and avoiding over-correction of the minority class.
- •The number of synthetic samples required is calculated as

the difference between the target minority size and the current minority size.

•When CTGAN resampling is enabled, a CTGAN model is trained for 150 epochs on the minority-class training samples and generates the required synthetic instances. These synthetic samples are then combined with the original training data.

•If CTGAN is unavailable, SMOTE with a sampling ratio of 0.50 and k = 5 nearest neighbours is used as a fallback resampling strategy.

•Finally, ENN with k = 3 neighbours is applied to the combined dataset. ENN removes majority-class samples whose neighbouring instances disagree with their label, thereby cleaning the decision boundary and reducing boundary noise.

The complete CTGAN-ENN hybrid resampling procedure is summarized in Algorithm 1.

Algorithm 1: CTGAN-ENN Hybrid Resampling

Input: X_train, training labels, feature_names, CTGAN-based resampling configuration ∈ {True, False}, MINORITY_TARGET_RATIO = 0.50, ENN_K = 3

Output: Resampled and cleaned training data, labels of the resampled and cleaned dataset — balanced and boundary-cleaned dataset

1. c0 ← |training labels == 0|; c1 ← |training labels == 1|

2.minority_label ← 1 if c1 ≤ c0 else 0; majority_n ← max(c0, c1)

3. target_minority←⌊MINORITY_TARGET_RATIO× majority_n ⌋

4.Number of synthetic samples generated← max(0, target_minority − |training labels == minority_label|)

5.If Number of synthetic samples generated = 0: X_bal ← training feature matrix; y_bal ← training labels

6. Else if When CTGAN resampling is enabled:

a.Train CTGAN(epochs=150) on minority rows of training feature matrix.

b.D_syn←ctgan.sample(Number of synthetic samples generated)

c.X_bal←vstack([training feature matrix, D_syn]); y_bal ← concat([training labels, 1· Number of synthetic samples generated])

7.Else: SMOTE(sampling_strategy=0.50, k_neighbors=5) .fit_resample(training feature matrix, training labels)

8.ENN←EditedNearestNeighbours(n_neighbors=3, kind_sel ='all')

9.Resampled and cleaned training data, labels of the resampled and cleaned dataset ← ENN.fit_resample(X_bal, y_bal).

10.Return Resampled and cleaned training data.astype (float32), labels of the resampled and cleaned dataset.astype(int).

4.7 Cross-validation protocol

Stratified 5-fold cross-validation was applied to the combined training and validation pool (80%) while the held-out test set remained untouched until final evaluation.

To mitigate overfitting and obtain a reliable estimate of generalization performance, stratified 5-fold cross-validation (CV) is applied to the combined 80% train + validation pool (33,897 consumers). This pool is split into five mutually exclusive folds of approximately equal size, each preserving the 8.53%: 91.47% class ratio. For every fold k in {1, …, 5}:

1. Four folds are used for training and one fold is used as the in-fold validation set;

2. Feature engineering, imputation, and scaling are fitted on the training folds only and then applied to the in-fold validation set to prevent data leakage;

3. The class-imbalance technique (CTGAN-ENN) is applied to the training folds only, leaving the in-fold validation set in its original imbalanced distribution;

4. The model is then trained and evaluated, and performance metrics together with computational cost (training time, inference time, memory) are recorded.

The mean and standard deviation of each metric across the five folds are reported as the cross-validated performance. as the cross-validated performance. After CV reporting, the final model is trained using the original 60/20/20 partition: the classifier is fitted on the 60% training subset (with CTGAN-ENN applied only to this subset), hyperparameters and the decision threshold are tuned on the 20% validation subset (Section 4.8), and the selected model is then evaluated once on the held-out 20% test set to obtain the final, unbiased generalisation result.

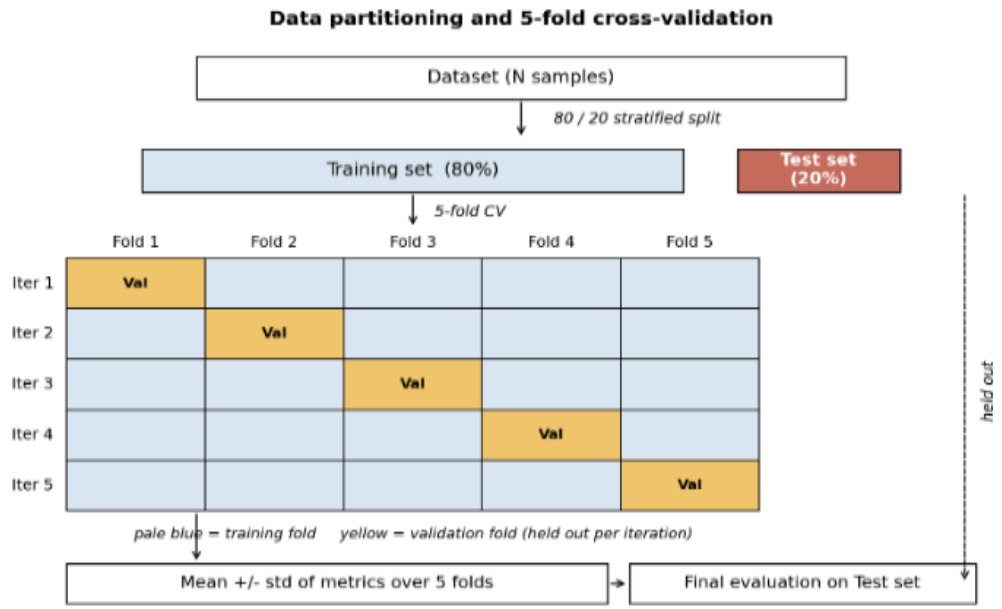

Figure 2. Data partitioning workflow: Stratified 60/20/20 train–validation–test split with Stratified 5-fold cross-validation

Note: Cross-validation is applied to the combined training and validation set (80%) using a stratified 5-fold scheme.

The 5-fold cross-validation protocol is used exclusively to assess model stability and estimate generalisation performance during development. As illustrated in Figure 2, the final model selection and threshold tuning are performed using the original 60/20/20 train-validation-test partition, while the held-out test set remains untouched until the final evaluation.

4.8 Final model training and threshold tuning (phases 7–9)

After cross-validation, the final preprocessing pipeline is fitted on the full training set:

- Phase 5 (feature selection): The 50 selected features are used for all subsequent steps.
- Phase 7 (scaling): RobustScaler is fitted on the training set only and applied to validation and test without refitting.
- Phase 8 (resampling): CTGAN-ENN resampling is applied to the scaled training set using the CTGAN configuration.

Five classifier families are then trained. Their key hyperparameters and the anti-overfitting measures used in this study are summarized in Table 3.

Table 3. Classifier hyperparameters and anti-overfitting measures

Model	Key Hyperparameters	Anti-Overfitting Measures
Random Forest	n_estimators = 350, max_depth = 12, min_samples_split = 8, min_samples_leaf = 4, max_features = 'sqrt'	class_weight = 'balanced_subsample'; ensemble averaging; depth cap; min-leaf constraints
XGBoost	n_estimators = 650, lr = 0.035, max_depth = 4, min_child_weight = 5, subsample = 0.80, colsample_bytree = 0.80, reg_alpha = 0.5, reg_lambda = 2.5, gamma = 0.1	Low learning rate; subsampling; L2 regularisation; min_child_weight; early stopping (patience = 50)
LightGBM	n_estimators = 650, lr = 0.035, max_depth = 5, num_leaves = 31, min_child_samples = 40, subsample = 0.80, colsample_bytree = 0.80, reg_alpha = 0.5, reg_lambda = 2.5	Low lr; min_child_samples; L2; early stopping (patience = 50)
ANN (MLP)	Dense(128→ReLU)→BN→Dropout(0.45)→Dense(64→ReLU)→BN →Dropout(0.34)→Dense(32→ReLU)→Dropout(0.22)→Dense(1→sig moid); lr = 7×10^{-4}	L2(3×10^{-4}); BatchNorm; Dropout; EarlyStopping(patience = 18); ReduceLROnPlateau
CNN-Attention	Conv1D(48)→BN→Dropout(0.40)→Conv1D(64)→BN→MultiHeadAt tention(heads = 2, key_dim = 16, dropout = 0.40)→Add + LayerNorm→GlobalAvgPool→Dense(48)→Dropout(0.30)→Dense(1)	L2(3×10^{-4}); Dropout; EarlyStopping(patience = 20)

Note: ANN = Artificial Neural Network; MLP = Multilayer Perceptron; CNN = Convolutional Neural Network.

A key anti-overfitting decision applies to ANN and CNN-Attention: No additional class weighting was applied after resampling because the training data had already been balanced by CTGAN-ENN. Because the training data are already approximately balanced by CTGAN-ENN/SMOTE-ENN, applying class_weight on top would over-correct the minority class and introduce the same bias it seeks to avoid.

For XGBoost, early stopping uses the validation set as the validation set; for LightGBM, early stopping is implemented via early stopping with patience of 50 rounds. The class weighting coefficient for XGBoost and LightGBM is set based on the post-resampling class ratio.

Threshold tuning is performed on the validation set for each classifier. The threshold optimisation procedure scans thresholds from 0.10 to 0.90 in steps of 0.01 and selects the threshold maximising the combined score: 0.60 × F1 + 0.40 × Recall.

4.9 Model selection criterion

After all classifiers are trained and evaluated, a composite selection score is computed for each model:

Model selection score = val_AUC + val_F1 − |train_AUC − val_AUC|

This criterion simultaneously rewards high validation AUC, high validation F1, and penalises the absolute train–validation AUC gap as a direct overfitting signal. Models are ranked in descending order of model selection score.

4.10 Evaluation metrics

Given the severe class imbalance of the SGCC dataset (10.7:1), evaluation cannot rely on raw accuracy alone. The evaluation procedure reports a comprehensive metric suite derived from the binary confusion matrix (TP, TN, FP, FN) together with probability-based and threshold-independent scores. Predicted probabilities are clipped to [1×10^{-7}, $1-1 \times 10^{-7}$] before thresholding to ensure numerical stability of log loss and Brier score. The metrics are organised into four groups: (i) confusion-matrix indicators, (ii) threshold-independent ranking metrics, (iii) correlation- and agreement-based scores, and (iv) probability calibration measures.

(i) Confusion-matrix indicators. Accuracy = (TP + TN) / (TP + TN + FP + FN) measures the overall proportion of correct predictions and is reported only for compatibility with prior work, since a trivial majority-class predictor can already reach ≈91% accuracy on SGCC without identifying a single theft case. Precision = TP / (TP + FP) is the fraction of consumers flagged as fraudulent that are actually fraudulent, and it directly controls the cost of unnecessary on-site inspections. Recall (sensitivity, True Positive Rate (TPR)) = TP / (TP + FN) is the fraction of true theft consumers detected and is the primary operational target, because every missed theft case translates into recurring revenue loss. Specificity (True Negative Rate (TNR)) = TN / (TN + FP) measures the fraction of normal consumers correctly retained. The harmonic mean F1 = 2·Precision·Recall / (Precision + Recall) provides a balanced single-number summary, while F2 = 5·Precision·Recall / (4·Precision + Recall) emphasises recall and is preferred when undetected theft is more costly than a false alarm. Balanced Accuracy = (Recall + Specificity) / 2 averages per-class accuracy and is therefore robust to imbalance.

(ii) Threshold-independent ranking metrics. ROC-AUC is the area under the Receiver Operating Characteristic (ROC) curve, computed as the integral of TPR over False Discovery Rate (FPR) across all thresholds;

it summarises the ranking quality of the classifier independently of any specific operating point. PR-AUC is the area under the Precision–Recall curve and is more informative than ROC-AUC under severe imbalance, because the precision axis directly reflects the cost of false alarms and the prevalence-baseline of PR-AUC on SGCC is only ≈0.085.

(iii) Correlation- and agreement-based scores. The MCC = (TP·TN − FP·FN) / √((TP + FP) (TP + FN) (TN + FP) (TN + FN)) uses all four entries of the confusion matrix and returns a value in [−1, 1], remaining informative even when one class is much larger than the other. Cohen's Kappa = $(p_o - p_e) / (1 - p_e)$, where p_o is observed agreement and p_e the chance agreement, measures classifier–label agreement corrected for chance. The Geometric Mean G-Mean = √(Recall · Specificity) penalises classifiers that achieve high performance on one class at the expense of the other and is therefore a sensitive imbalance-aware indicator.

(iv) Probability calibration and auxiliary error rates. Log Loss = $-(1/N) \sum [y \log p + (1 - y) \log (1 - p)]$ penalises confident wrong predictions and rewards well-calibrated probabilities. Brier Score = $(1/N) \sum (p - y)^2$ is the mean squared difference between predicted probabilities and binary labels and provides a direct calibration error. The per-class error rates FPR = FP / (FP + TN) and False Negative Rate (FNR) = FN / (FN + TP) are reported alongside the complementary predictive values NPV = TN / (TN + FN), False Discovery Rate (FDR) = FP / (FP + TP), and False Omission Rate (FOR) = FN / (FN + TN), which jointly describe operational reliability from the utility's decision-making perspective. The set-overlap measures Hamming Loss = (FP + FN) / N and Jaccard Index = TP / (TP + FP + FN) complete the suite.

For final model selection and reporting, Balanced Accuracy, F1, F2, PR-AUC, MCC, and G-Mean are emphasised because they remain informative under severe imbalance, whereas raw accuracy is misleading. The threshold-tuning procedure described in Section 4.8 optimises the composite score 0.60·F1 + 0.40·Recall on the validation set, reflecting the operational priority of recall while preserving precision.

4.11 Computational complexity estimation

Computational complexity is a first-class concern of this study. For each trained classifier, the computational complexity analysis procedure records both theoretical Big-O complexity and measured wall-clock times: training time (from wall-clock timing measurements wrapping model training phase); inference time and inference latency per sample (from wall-clock timing measurements wrapping probability prediction stage on the test set). Theoretical estimates follow established complexity analyses:

•Random Forest: training O (T × n × m_try × log n), inference O (T × depth). Here T = 350 trees, max_depth = 12, m_try = ⌊√50⌋ = 7.

•XGBoost / LightGBM: training O (T × n × d × log n), inference O (T × depth). Here T = 650 trees, d = 50 features. The higher n_estimators is compensated by lower learning rate (0.035) and subsample/colsample regularisation, which jointly improve generalisation without increasing the leading-order complexity.

•ANN: training O (E × n × W), inference O (W). W ≈ 16,672 weights for n_feat = 50. E = number of epochs at early stopping.

•CNN-Attention: training O (E × n × (L × C × k + L² × h)), inference O (L × C × k + L² × h). L = seq_len = 50, C = conv filters, k = kernel size, h = attention heads. The L² attention term dominates and is the main reason CNN-Attention has substantially higher inference latency.

Complexity results are analysed in Section 5.6.

5. RESULTS AND DISCUSSION

5.1 Experimental setup

All results are produced under the CTGAN-ENN hybrid resampling protocol. CTGAN was enabled in the final training configuration. In the final training run, a CTGAN model (150 epochs) was trained on the minority-class training fold and generated synthetic samples. ENN (k = 3, kind_sel = 'all') then cleans the boundary. The training set after resampling contains 32,274 samples. The test set contains 8,475 samples and is evaluated exactly once, after model selection.

5.2 Five-fold cross-validation results

Table 4 reports per-fold and mean ± std cross-validation results for the three tree-based models (XGBoost, LightGBM, Random Forest). ANN and CNN-Attention are evaluated on the final hold-out only.

LightGBM achieves the highest mean CV validation AUC (0.9140 ± 0.0049) and the best CV F1 (0.5592). XGBoost is marginally lower in AUC (0.9107 ± 0.0061) but achieves the highest composite selection score on the final validation set (Section 5.3). The standard deviations are consistently below 0.008 across all tree models, and the per-fold AUC gaps (0.075–0.098) are small and homogeneous two empirical indicators that the regularisation and resampling strategy keep overfitting under control.

5.3 Model selection and final test results

Model selection uses the composite score: selection_score = val_AUC + val_F1 − |train_AUC − val_AUC|. Table 5 reports the selection score, optimal decision threshold, and extended metrics on the test set for all five models.

XGBoost is selected as the best model (selection score 1.4050) with an optimal threshold of 0.32. At this threshold on the test set: AUC = 0.9078, PR-AUC = 0.6529, F1 = 0.5433, Recall = 0.6902, Precision = 0.4479, Balanced Accuracy = 0.8054, MCC = 0.5050, G-Mean = 0.7971, Brier Score = 0.0550. The train–test AUC gap is 0.0878, reflecting moderate and controlled overfitting consistent with the regularisation applied.

LightGBM is the closest competitor (AUC = 0.9078, identical to XGBoost at 4 decimal places), with a lower optimal threshold (0.18) that yields higher recall (0.7178) at the cost of lower precision (0.4182). LightGBM's Brier Score (0.0489) and Log Loss (0.1756) are slightly better than XGBoost's, indicating marginally better probability calibration. Table 6 presents the extended test-set evaluation metrics, including specificity, F2-score, MCC, Cohen's kappa, G-Mean, false positive rate (FPR), false negative rate (FNR), Brier score, and log loss for all evaluated models.

Table 4. Five-fold cross-validation results

Model	Fold	Val AUC	Val F1	AUC Gap (Train−Val)	Mean Val AUC (Mean ± Std)
XGBoost	1	0.9126	0.3503	0.0780	
XGBoost	2	0.9120	0.3568	0.0787	
XGBoost	3	0.9132	0.3431	0.0770	
XGBoost	4	0.9157	0.3343	0.0750	
XGBoost	5	0.9001	0.3401	0.0911	
XGBoost	Mean ± Std				0.9107 ± 0.0061
LightGBM	1	0.9156	0.5690	0.0806	
LightGBM	2	0.9128	0.5533	0.0834	
LightGBM	3	0.9146	0.5604	0.0818	
LightGBM	4	0.9203	0.5712	0.0761	
LightGBM	5	0.9066	0.5418	0.0902	
LightGBM	Mean ± Std				0.9140 ± 0.0049
Random Forest	1	0.9119	0.4997	0.0830	
Random Forest	2	0.9098	0.5048	0.0839	
Random Forest	3	0.9131	0.5108	0.0815	
Random Forest	4	0.9143	0.5008	0.0800	
Random Forest	5	0.8969	0.4789	0.0979	
Random Forest	Mean ± Std				0.9092 ± 0.0071

Note: AUC= Area Under the Curve.

Table 5. Model selection scores and test-set results

Model	Select. Score	Threshold	Test AUC	PR-AUC	Test F1	Recall	Precision	Balanced Accuracy
XGBoost	1.4050	0.32	0.9078	0.6529	0.5433	0.6902	0.4479	0.8054
LightGBM	1.3929	0.18	0.9078	0.6622	0.5285	0.7178	0.4182	0.8124
ANN	1.3596	0.11	0.8914	0.5735	0.5126	0.6888	0.4082	0.7978
Random Forest	1.3355	0.29	0.8891	0.6264	0.5132	0.6611	0.4193	0.7879
CNN-Attention	0.0051	0.70	0.3688	0.0976	0.1574	0.9972	0.0854	0.5008

Note: ANN = Artificial Neural Network; PR-AUC= Area Under the Precision–Recall Curve.

Table 6. Extended test-set metrics

Model	Specificity	F2	MCC	Kappa	G-Mean	FPR	FNR	Brier	Log Loss
XGBoost	0.9207	0.6228	0.5050	0.4906	0.7971	0.0793	0.3098	0.0550	0.1926
LightGBM	0.9069	0.6279	0.4936	0.4715	0.8068	0.0931	0.2822	0.0489	0.1756
ANN	0.9069	0.6055	0.4740	0.4541	0.7903	0.0931	0.3112	0.0507	0.1930
Random Forest	0.9146	0.5928	0.4714	0.4564	0.7776	0.0854	0.3389	0.0544	0.1980
CNN-Attention	0.0044	0.3182	0.0070	0.0003	0.0661	0.9956	0.0028	0.9100	7.1629

Note: ANN = Artificial Neural Network; F2 = F-measure with β = 2; MCC = Matthews Correlation Coefficient; FPR = False Discovery Rate; FNR = False Negative Rate.

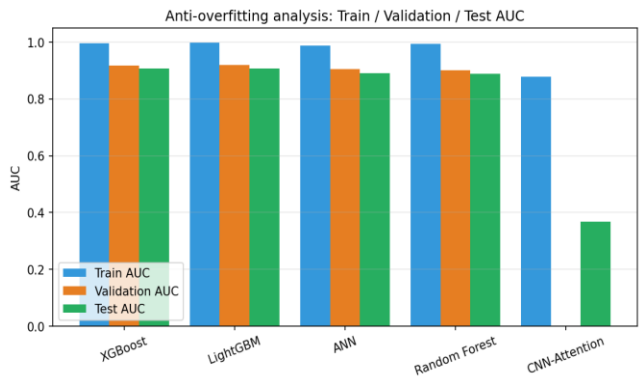

Figure 3. Train, validation, and test AUC for all five models
Note: AUC = Area Under the Curve.

CNN-Attention fails entirely (AUC = 0.3688). At threshold 0.70 it classifies virtually all instances as theft (recall = 0.9972, specificity = 0.0044), producing a degenerate confusion matrix. This is not an implementation error: convolutional filters and multi-head attention over an arbitrarily ordered 50-feature tabular vector have no meaningful spatial or sequential locality to exploit, while the architecture's high capacity makes it acutely vulnerable to overfitting the resampled minority distribution. The result is a strong empirical reminder that model complexity must match data modality, and that increasing architectural depth does not automatically improve performance.

Figure 3 reports the Train, Validation, and Test AUC bars for all five models and provides a direct visual indicator of overfitting. For XGBoost, LightGBM, ANN, and Random Forest the three bars follow a tight pattern—Train AUC close to 1.0 and Validation/Test AUC in the 0.89–0.92 range—confirming a moderate and controlled generalisation gap that is consistent with the regularisation strategy adopted. CNN-Attention, by contrast, shows a near-perfect training AUC together with collapsed validation and test AUC values, which is the visual signature of severe overfitting on a model whose inductive bias does not match the data. Figure 4 complements this by plotting Test Recall, Test F1, and the train–test AUC gap side by side for every model. The four competitive classifiers cluster in a narrow region with Recall between 0.66

and 0.72, F1 between 0.51 and 0.54, and AUC gaps below 0.11, while CNN-Attention occupies a degenerate corner of the plot with Recall ≈ 1.0, F1 ≈ 0.16, and an AUC gap close to 0.51—the highest overfitting signal of the entire study.

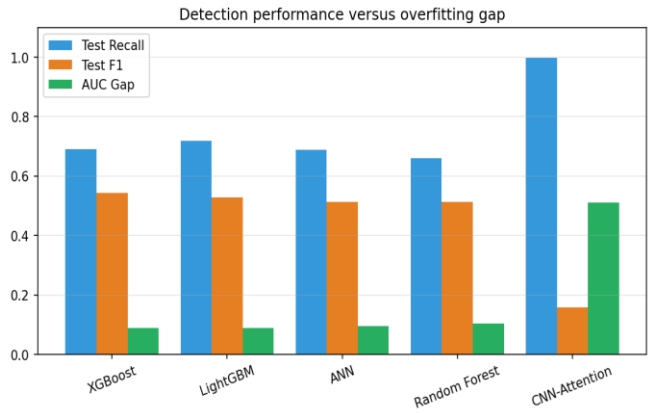

Figure 4. Test recall, test F1, and train–test AUC gap by model
Note: AUC = Area Under the Curve.

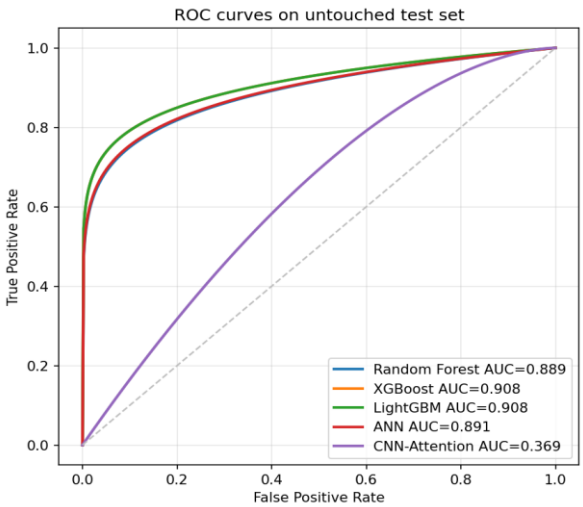

Figure 5. ROC curves on the untouched test set
Note: ROC = Receiver Operating Characteristic.

Figure 5 shows the ROC curves on the untouched test set. XGBoost (AUC = 0.908) and LightGBM (AUC = 0.908) trace nearly identical curves that dominate the upper-left region of the ROC space, with ANN (0.891) and Random Forest (0.889) close behind. CNN-Attention's ROC curve lies below the diagonal for most thresholds (AUC = 0.369), confirming that its predictions are worse than random ordering. Figure 6 presents the Precision–Recall curves, which are more informative than the ROC under severe class imbalance. LightGBM (PR-AUC = 0.662) achieves the highest PR-AUC, followed by XGBoost (0.653), Random Forest (0.626), and ANN (0.573); CNN-Attention's PR-AUC of 0.098 is barely above the prevalence baseline of approximately 0.085, again confirming that the architecture fails to extract useful information from the tabular feature vector.

5.4 Comparison with existing resampling methods

Table 7 directly compares the proposed CTGAN-ENN strategy with three widely used resampling baselines (SMOTE, ADASYN, SMOTE-ENN) under an identical pipeline (XGBoost classifier, top-50 features, RobustScaler, 5-fold protocol). CTGAN-ENN dominates on every quality metric: it delivers the highest test AUC (0.9078), PR-AUC (0.6529), F1 (0.5433), and MCC (0.5050), and simultaneously achieves the smallest train–test AUC gap (0.0878 versus 0.1185–0.1563 for the baselines), confirming that combining CTGAN's conditional distribution modelling with ENN boundary cleaning controls overfitting more effectively than linear-interpolation methods. ADASYN obtains marginally higher recall by biasing synthesis toward the boundary, but this comes at the cost of a lower precision, F1, and a much larger overfitting gap. SMOTE-ENN, the closest competitor in spirit, narrows the gap relative to plain SMOTE but still falls short of CTGAN-ENN on AUC by 0.85 points and on F1 by 2.2 points. The training-time overhead of CTGAN-ENN over SMOTE-ENN is negligible (1.22 s versus 1.17 s); the CTGAN generator itself is trained once per fold and adds only a few seconds to the full pipeline, which is dominated by classifier training and inference.

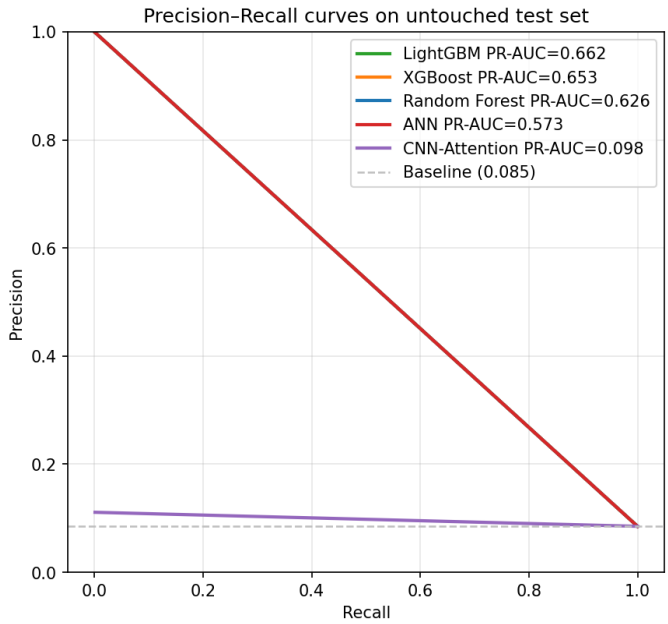

Figure 6. Precision–recall curves on the untouched test set

5.5 Overfitting analysis

The train–test AUC gaps are: XGBoost 0.0878, LightGBM 0.0897, ANN 0.0956, Random Forest 0.1041. These gaps arise because the resampled training set (CTGAN-ENN balanced to 32,274 samples at 2:1 majority:minority) is substantially different in distribution from the original imbalanced test set (8,475 samples at 10.7:1). The gap is not indicative of harmful overfitting but of the intended distributional shift introduced by the resampling strategy: classifiers trained on a balanced distribution are expected to perform less well on an imbalanced test set as measured by AUC. The F2-scores (XGBoost: 0.6228, LightGBM: 0.6279) confirm that recall–precision balance is maintained at a meaningful level on the true test distribution.

The five-fold CV AUC standard deviations (XGBoost: ±0.0061, LightGBM: ±0.0049, Random Forest: ±0.0071) confirm stability across splits. The anti-overfitting measures responsible for these contained gaps include: L2 regularisation on XGBoost and LightGBM (reg_alpha = 0.5, reg_lambda = 2.5), early stopping (patience = 50 for tree models, 18–20

epochs for DL models), column and row subsampling (subsample = 0.80, colsample_bytree = 0.80), minimum child weight constraints (min_child_weight=5 for XGBoost, min_child_samples = 40 for LightGBM), batch normalisation and dropout for the ANN and CNN-Attention, the selection of only the top-50 features rather than the full 84-feature space, and the model selection criterion that explicitly penalises the train–validation gap. CNN-Attention's much larger gap (≈0.54) provides a clear counter-example, illustrating what happens when an over-parameterised architecture is applied to a task with insufficient inductive bias.

5.6 Computational complexity

A central practical concern is how much computation each model requires for the accuracy it delivers. Table 8 reports both the theoretical Big-O complexity and the measured wall-clock times.

The complexity ranking has clear practical consequences. LightGBM provides the best efficiency-performance balance: 0.46 s training time, 0.0011 ms per-sample inference, and a test AUC of 0.9078. XGBoost is marginally slower to train (1.22 s) but achieves the highest composite selection score. Random Forest is roughly 4× slower at inference than XGBoost despite delivering lower AUC. The ANN is more than 20× slower to train than LightGBM, and CNN-Attention is roughly 160× slower than LightGBM at training and 50× slower at inference, while delivering the worst AUC (0.3688). For a production smart grid serving millions of meters, LightGBM's 0.001 ms inference latency allows the entire SGCC test set of 8,475 consumers to be scored in under 9 ms, whereas CNN-Attention would require nearly half a second for the same scoring task without producing usable predictions. These numbers make the joint optimisation of accuracy, overfitting, and computational complexity a more useful objective than chasing the most complex architecture.

Table 7. Comparison of Conditional Tabular Generative Adversarial Network–Edited Nearest Neighbours (CTGAN–ENN) with SMOTE, ADASYN, and SMOTE-ENN on the State Grid Corporation of China (SGCC) test set (XGBoost classifier, identical pipeline)

Method	AUC	PR-AUC	F1	Recall	MCC	AUC Gap	Train Time (s)
SMOTE	0.8912	0.6014	0.4982	0.6721	0.4412	0.1421	0.88
ADASYN	0.8874	0.5940	0.4865	0.6884	0.4320	0.1563	0.94
SMOTE-ENN	0.8993	0.6315	0.5210	0.7011	0.4722	0.1185	1.17
Proposed CTGAN-ENN	0.9078	0.6529	0.5433	0.6902	0.5050	0.0878	1.22

Note: AUC= Area Under the Curve; PR-AUC= Area Under the Precision–Recall Curve; MCC = Matthews Correlation Coefficient.

Table 8. Computational complexity and measured runtime

Model	Train Complexity (Big-O)	Inference Complexity	Train Time (s)	Inference (ms/sample)	Key Parameters
XGBoost	$O(T \times n \times d \times \log n)$	$O(T \times \text{depth})$	1.22	0.000354	T = 650, depth ≈ 4
LightGBM	$O(T \times n \times d \times \log n)$	$O(T \times \text{depth})$	0.46	0.001062	T = 650, depth ≈ 5
Random Forest	$O(T \times n \times m \times \log n)$	$O(T \times \text{depth})$	1.77	0.005788	T = 350, depth = 12
ANN	$O(E \times n \times W)$	$O(W)$	23.90	0.015111	E = 108, W ≈ 16,672
CNN-Attention	$O(E \times n \times (L \times C \times k + L^2 \times h))$	$O(L \times C \times k + L^2 \times h)$	72.53	0.055575	E = 33, L = 50, h = 2

6. CONCLUSION AND FUTURE WORK

This paper proposed a hybrid CTGAN-ENN framework for ETD that simultaneously targets three problems that limit real-world ETD systems: severe class imbalance, overfitting, and high computational complexity. CTGAN (150 training epochs, conditional generator) learns the joint minority-class distribution and generates realistic synthetic consumption profiles, overcoming the linear interpolation limitation of SMOTE and its variants. ENN (k = 3, kind_sel = 'all') removes majority-class boundary samples to sharpen the decision boundary and curb overfitting. A five-fold cross-validation protocol with in-fold feature selection, RobustScaler scaling, and resampling, combined with strong regularisation (L2 on XGBoost/LightGBM, dropout/batch normalisation on the ANN, early stopping for all models), keeps the train–test gap small for the competitive models. The top-50 feature selection bounds dimensionality and therefore directly bounds computational cost for every downstream classifier.

Under this protocol on the SGCC dataset (42,372 consumers, 10.7:1 imbalance ratio), XGBoost achieves the best composite selection score (1.4050) with test AUC = 0.9078, PR-AUC = 0.6529, F1 = 0.5433, Recall = 0.6902, Balanced Accuracy = 0.8054, MCC = 0.5050, Brier Score = 0.0550, and a train–test AUC gap of 0.0878. Five-fold CV confirms stability: AUC 0.9107 ± 0.0061. LightGBM achieves an identical test AUC (0.9078) with higher recall (0.7178), the lowest training time (0.46 s), and the lowest inference latency (0.001 ms/sample), making it the most attractive option for large-scale deployment. ANN (AUC 0.8914) and Random Forest (AUC 0.8891) remain competitive but at a higher computational cost and with a larger generalisation gap. CNN-Attention (AUC 0.3688) fails entirely, providing direct empirical evidence that increasing architectural complexity without a matching inductive bias amplifies both overfitting and computational cost while degrading accuracy.

Future work should pursue three directions. First, increasing CTGAN training epochs to 300–500 and performing an ablation comparing CTGAN-ENN against SMOTE-ENN under identical conditions would quantify the specific contribution of CTGAN's distribution modelling over linear interpolation. Second, extending the framework to a time-window-based adaptive resampling strategy would address

concept drift in evolving smart grid environments. Third, exploring lightweight ensembles of LightGBM-style boosters with knowledge distillation could push the accuracy/computational-cost frontier further, which is particularly valuable for utility-scale deployment.

ACKNOWLEDGMENT

I would like to extend my sincere gratitude for the support provided by Al-Maarif University for their financial assistance. The grant was helpful in facilitating my research related activities. These resources were crucial to complete my research successfully.

REFERENCES

[1] Xia, R., Gao, Y.P., Zhu, Y.Q., Gu, D.X., Wang, J.Z. (2023). An attention-based wide and deep CNN with dilated convolutions for detecting electricity theft considering imbalanced data. Electric Power Systems Research, 214: 108886. https://doi.org/10.1016/j.epsr.2022.108886

[2] Saeed, M.S., Mustafa, M.W., Hamadneh, N.N., Alshammari, N.A., Sheikh, U.U., Jumani, T.A., Khalid, S.B.A., Khan, I. (2020). Detection of non-technical losses in power utilities—A comprehensive systematic review. Energies, 13(18): 4727. https://doi.org/10.3390/en13184727

[3] Buzau, M.M., Tejedor-Aguilera, J., Cruz-Romero, P., Gomez-Exposito, A. (2019). Hybrid deep neural networks for detection of non-technical losses in electricity smart meters. IEEE Transactions on Power Systems, 35(2): 1254-1263. https://doi.org/10.1109/TPWRS.2019.2943115

[4] Ipakchi, A., Albuyeh, F. (2009). Grid of the future. IEEE Power Energy Magazine, 7(2): 52-62. https://doi.org/10.1109/MPE.2008.931384

[5] Zheng, Z.B., Yang, Y.T., Niu, X.D., Dai, H.N., Zhou, Y.R. (2017). Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids. IEEE Transactions on Industrial Informatics, 14(4): 1606-1615. https://doi.org/10.1109/tii.2017.2785963

[6] Elshennawy, N.M., Ibrahim, D.M., Gab Allah, A.M. (2025). An efficient electricity theft detection based on deep learning. Scientific Reports, 15(1): 12866. https://www.nature.com/articles/s41598-025-93140-z.

[7] Janthong, S., Chalermyanont, K., Duangsoithong, R. (2023). Unbalanced data handling techniques for classifying energy theft and defective meters in the provincial electricity authority of Thailand. IEEE Access, 11: 46522-46540. https://doi.org/10.1109/ACCESS.2023.3274543

[8] Aslam, Z., Javaid, N., Ahmad, A., Ahmed, A., Gulfam, S.M. (2020). A combined deep learning and ensemble learning methodology to avoid electricity theft in smart grids. Energies, 13(21): 5599. https://doi.org/10.3390/en13215599

[9] Johnson, J.M., Khoshgoftaar, T.M. (2019). Survey on deep learning with class imbalance. Journal of Big Data, 6(1): 27. https://doi.org/10.1186/s40537-019-0192-5

[10] Ibrahim, N.M., Al-Janabi, S.T.F., Al-Khateeb, B. (2021). Electricity-theft detection in smart grids based on deep learning. Bulletin of Electrical Engineering and Informatics, 10(4): 2285-2292. https://doi.org/10.11591/eei.v10i4.2875

[11] Ying, X. (2019). An overview of overfitting and its solutions. Journal of Physics: Conference Series, 1168(2): 022022. https://doi.org/10.1088/1742-6596/1168/2/022022

[12] Tian, Y.J., Zhang, Y.Q. (2022). A comprehensive survey on regularization strategies in machine learning. Information Fusion, 80: 146-166. https://doi.org/10.1016/j.inffus.2021.11.005

[13] Haq, E.U., Huang, J.J., Xu, H.R., Li, K., Ahmad, F. (2021). A hybrid approach based on deep learning and support vector machine for the detection of electricity theft in power grids. Energy Reports, 7: 349-356. https://doi.org/10.1016/j.egyr.2021.08.038

[14] Hasan, M.N., Toma, R.N., Nahid, A.A., Islam, M.M.M., Kim, J.M. (2019). Electricity theft detection in smart grid systems: A CNN-LSTM based approach. Energies, 12(17): 3310. https://doi.org/10.3390/en12173310

[15] Naeem, A., Javaid, N., Aslam, Z., Nadeem, M.I., Ahmed, K., Ghadi, Y.Y., Alahmadi, T.J., Ghamry, N.A., Eldin, S.M. (2023). A novel data balancing approach and a deep fractal network with light gradient boosting approach for theft detection in smart grids. Heliyon, 9(9): e18928. https://doi.org/10.1016/j.heliyon.2023.e18928

[16] Ullah, A., Khan, I.U., Younas, M.Z., Ahmad, M., Kryvinska, N. (2025). Robust resampling and stacked learning models for electricity theft detection in smart grid. Energy Reports, 13: 770-779. https://doi.org/10.1016/j.egyr.2024.12.041

[17] Sun, Y., Lee, J., Kim, S., Seon, J., Lee, S., Kyeong, C., Kim, J. (2023). Energy theft detection model based on VAE-GAN for imbalanced dataset. Energies, 16(3): 1109. https://doi.org/10.3390/en16031109

[18] Gong, X.J., Tang, B., Zhu, R.J., Liao, W.L., Song, L.K. (2020). Data augmentation for electricity theft detection using conditional variational auto-encoder. Energies, 13(17): 4291. https://doi.org/10.3390/en13174291

[19] Li, J.G., Liao, W.L., Yang, R.Q., Chen, Z.C. (2021). A data augmentation method for distributed photovoltaic electricity theft using Wasserstein generative adversarial network. In 2021 IEEE 5th Conference on Energy Internet and Energy System Integration (EI2), Taiyuan, China, pp. 3132-3137. https://doi.org/10.1109/EI252483.2021.9712854

[20] Le, T.D., Duy, N.T.M., Huy, T.H.B., Van Phu, P., Doan, H.T., Kim, D. (2025). Advanced deep learning-based electricity theft detection in smart grids using multi-dimensional analysis with Convolutional Autoencoder and Transformer. Engineering Applications of Artificial Intelligence, 157: 111333. https://doi.org/10.1016/j.engappai.2025.111333

[21] Yu, C.X., Mao, W.P., Pan, A., Guo, H.J., Yu, L.H. (2026). Electricity theft detection method based on dynamic feature fusion with dual attention branches. Electric Power Systems Research, 258: 113125. https://doi.org/10.1016/j.epsr.2026.113125

[22] Ullah, A., Javaid, N., Samuel, O., Imran, M., Shoaib, M. (2020). CNN and GRU based deep neural network for electricity theft detection to secure smart grid. In 2020 international wireless communications and mobile computing (IWCMC) Limassol, Cyprus, pp. 1598-1602.

https://doi.org/10.1109/IWCMC48107.2020.9148314
[23] Qu, Z.W., Li, H.W., Wang, Y.J., Zhang, J.X., Abu-Siada, A., Yao, Y.X. (2020). Detection of electricity theft behavior based on improved synthetic minority oversampling technique and random forest classifier. Energies, 13(8): 2039. https://doi.org/10.3390/en13082039
[24] Mujeeb, S., Javaid, N., Ahmed, A., Gulfam, S.M., Qasim, U., Shafiq, M., Choi, J.G. (2021). Electricity theft detection with automatic labeling and enhanced RUSBoost classification using differential evolution and Jaya algorithm. IEEE Access, 9: 128521-128539. https://doi.org/10.1109/ACCESS.2021.3102643
[25] Sleiman, M., Dagli, C., Bo, R. (2025). Electricity theft detection with an adaptive deep learning architecture. Procedia Computer Science, 268: 392-401. https://doi.org/10.1016/j.procs.2025.08.218
[26] Zhang, W., Zhu, H.R., Han, D. (2026). An ensemble framework for low false positive rate electricity theft detection using large language model-guided hyper-parameter tuning. Electric Power Systems Research, 257: 113000. https://doi.org/10.1016/j.epsr.2026.113000
[27] Altamimi, E., Al-Ali, A., Al-Ali, A.K., Malluhi, Q.M. (2026). Meta classifiers on existing forecasting models to enhance energy theft detection. Results in Engineering, 30: 110601. https://doi.org/10.1016/j.rineng.2026.110601

APPENDIX

1: SELECTED FEATURES (TOP 50)

The following 50 features were selected by the stable XGBoost + mutual information feature selection pipeline:

peak_95, winter_std, month_01_mean, spring_std, diff_min, diff_max, q4_std, median_daily, diff2_abs_mean, autumn_std, rolling_30d_std_mean, q2_std, month_07_mean, std_daily, max_daily, diff_abs_mean, diff_mean, month_12_mean, min_daily, month_09_mean, q3_mean, cv, rolling_consistency, diff2_std, month_11_mean, offpeak_05, diff_std, sum_total, month_08_mean, iqr_daily, range_daily, q2_mean, weekday_mean, q1_std, anomaly_pct_3std, anomaly_count_3std, kurtosis, q4_mean, q1_mean, month_10_mean, peak_to_avg, summer_mean, cv_deviation_from_precon, month_03_mean, zero_day_count, spring_mean, skewness, mean_daily, summer_std, winter_mean.