Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Cost-Sensitive Learning, Simulated PU Learning, and One-Class Autoencoding for Extreme-Imbalance Credit Card Fraud Detection

Jiaying Jin[1], *Tina Huang*[2], *Sam Lu*[3]
[1]*Applied Analytics, Columbia University, NY, USA*
[2]*Computer Engineering, Columbia University, NY, USA*
[3]*Computer Science, Columbia University, NY, USA*
jj3373@columbia.edu

DOI: 10.69987/JACS.2024.40605

Keywords

fraud detection; extreme class imbalance; cost-sensitive learning; positive-unlabeled learning; focal loss; one-class autoencoder; threshold optimization; average precision

Abstract

Extreme class imbalance makes fraud detection evaluation sensitive to both the ranking metric and the chosen operating point. This revised study presents a single-split benchmark on the Credit Card Fraud Detection dataset (284,807 transactions; 492 frauds) comparing cost-sensitive gradient boosting, cost-aware neural networks, simulated positive-unlabeled (PU) training, and a one-class autoencoder baseline. Two methodological corrections are explicit: the benchmark itself is fully labeled, so PU learning is implemented by hiding negative labels during training; and the autoencoder is treated as a one-class reconstruction baseline rather than as strictly self-supervised learning. Models are evaluated with average precision (AP), ROC-AUC, recall at fixed false-positive rates, and a simple monetary cost model in which each reviewed alert costs one unit and each missed fraud costs the transaction amount. LightGBM attains the best ranking performance (AP 0.824), while XGBoost attains the greatest monetary savings after validation-based threshold selection, reducing test cost from 8483.36 to 3891.01 cost units (54.1% savings). Logistic regression remains competitive by ROC-AUC, which highlights why ROC-based comparisons can be misleading in this setting. Among neural models, nnPU gives the best ranking performance, whereas focal loss yields the best savings among the deep baselines. The one-class autoencoder is materially weaker than supervised models but remains useful as a label-light reference detector. The central conclusion is that ranking quality alone does not determine the preferred fraud-screening policy; business utility and threshold choice must be specified jointly.

1. Introduction

Payment fraud detection is a canonical rare-event problem: the positive class is extremely small, labels are often delayed, and operational actions have asymmetric consequences [2]-[4]. Under such imbalance, a classifier can achieve impressive accuracy simply by predicting every transaction as legitimate. That makes accuracy a poor guide to deployment quality and shifts attention toward ranking metrics, alert thresholds, and the downstream economics of review.

Precision-recall analysis is generally more informative than ROC analysis in highly imbalanced binary classification [11], [12]. Even average precision, however, is still only a ranking summary. A fraud team does not deploy a ranked list in the abstract; it deploys a thresholded screening policy under review-capacity and cost constraints [7]-[10]. Resampling strategies such as SMOTE [13] are another common response to imbalance, but resampling alone does not specify the operational objective. The present study therefore emphasizes loss-aware learning and validation-based threshold selection.

Deep models introduce additional choices. Focal loss reduces the relative contribution of easy negatives [16]; positive-unlabeled (PU) learning supports training when only positive labels are reliable [17]-[19]; and one-class reconstruction methods provide a label-light anomaly baseline [5], [6], [20]-[22]. These approaches address different practical problems, so a fair comparison

requires both clear protocol definitions and deployment-relevant reporting.

This revision retains the original comparative benchmark but corrects two conceptual issues. First, the Credit Card Fraud Detection benchmark is fully labeled [1], [23], so the PU experiment must be interpreted as a simulated protocol rather than as a naturally unlabeled production stream. Second, an autoencoder trained only on known normal transactions is best described as a one-class or semi-supervised anomaly detector, not as strictly self-supervised learning. The revised manuscript also removes a dubious unweighted MLP result from the main comparison, streamlines duplicate metrics, and replaces weak figures with plots directly supported by the reported summary results.

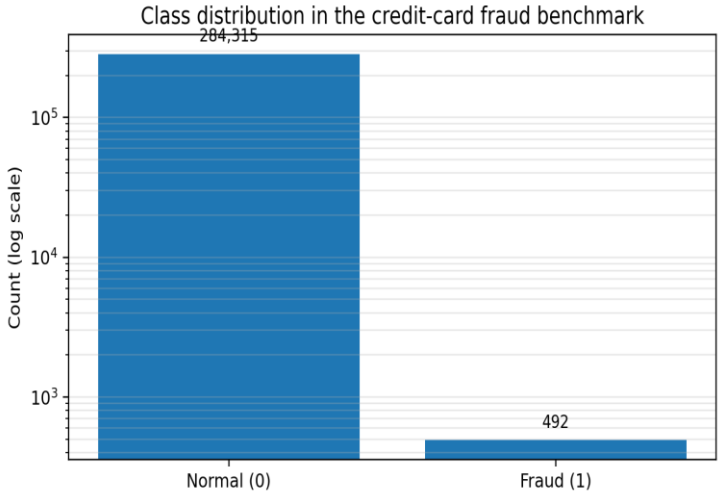

Fig. 1. Class distribution in the Credit Card Fraud Detection benchmark (log scale).

Methodological clarifications reflected in this revision

• PU learning is evaluated as a simulated protocol: positive labels remain visible during training, while all other training examples are treated as unlabeled.

• The autoencoder baseline is reported as a one-class reconstruction model trained on known normal transactions from the training split.

• The main comparative tables focus on operationally meaningful models; an unstable unweighted MLP baseline is not retained in the final benchmark summary.

2. Method

2.1 Dataset and preprocessing

We use the public credit-card benchmark described by Dal Pozzolo et al. [1] and archived on Zenodo [23]. The dataset contains 284,807 transactions collected over two days, with 492 frauds (0.173 percent) and 30 numerical features: Time, Amount, and anonymized components V1-V28. A stratified 70/15/15 split yields 199,364 training examples (344 frauds), 42,721 validation examples (74 frauds), and 42,722 test examples (74 frauds).

Linear and neural models use z-score normalization fitted on the training split. Tree-based models are trained on the raw numerical features. The unscaled Amount field is retained separately for all cost calculations so that threshold selection can depend on transaction value rather than on standardized units.

To make the PU comparison methodologically coherent on a fully labeled benchmark, we hide the negative labels during training and treat all non-positive training samples as unlabeled. Validation and test sets remain fully labeled. Likewise, the autoencoder is trained only on training-set normals selected by label; it is therefore a one-class baseline and should not be interpreted as fully unsupervised.

Table I. Dataset statistics and stratified split (70/15/15).

Split	Samples	Fraud	Fraud rate
Total	284807	492	0.00172749

Train (70%)	199364	344	0.00172549
Validation (15%)	42721	74	0.00173217
Test (15%)	42722	74	0.00173213

2.2 Operational cost model and threshold selection

We model a screening workflow in which every flagged transaction incurs a fixed review cost c r = 1, while every missed fraud incurs a loss equal to the transaction amount. Predicting a legitimate transaction as legitimate has zero cost. Predicting a fraud as fraud still incurs the review cost, but it avoids the missed-fraud loss.

$$\text{Cost}(\tau) = c_r \cdot N_{\text{flagged}}(\tau) + \sum_{i:v_i=1, s_i<\tau} \text{Amount}_i$$

$$\text{Savings}(\tau) = \text{Cost}_{\text{baseline}} - \text{Cost}(\tau)$$

Here, s i denotes the model score for transaction i and tau is the deployed decision threshold. Cost baseline corresponds to the no-screening policy that never flags any transaction. On the test split, Cost baseline = 8483.36. For each model, thresholds are selected once on the validation set by maximizing savings and are then frozen for the test set.

Table II. Business cost matrix used for cost-savings evaluation.

Actual \ Predicted	**Predicted normal (0)**	**Predicted fraud (1)**
Actual normal (0)	0	1.0 review cost
Actual fraud (1)	Amount loss	1.0 review cost; loss prevented

2.3 Evaluation metrics

Average precision (AP) is the primary ranking metric because it is sensitive to rare positives [11], [12]. ROC-AUC is also reported for reference, but it is not treated as the main selection criterion because it can remain optimistic when the negative class dominates. To approximate investigator-capacity constraints, we additionally report Recall at false-positive rates (FPR) of 0.1 percent, 0.5 percent, and 1.0 percent.

For deployment utility, each model receives a single operating point selected on the validation split. The main comparison therefore distinguishes between ranking quality and thresholded business utility rather than conflating them.

2.4 Model families and configurations

Table III summarizes the final model families. The benchmark emphasizes cost-aware and label-light baselines rather than exhaustive hyperparameter search.

Table III. Model families and key hyperparameters used in the final comparison.

Model	**Family**	**Key settings**	**Training features**	**Notes**
LogReg (balanced)	Linear	class weight=balanced	standardized	reference supervised baseline
XGBoost (cs)	Boosted trees	200 trees; depth 3; lr 0.1; subsample 0.8; colsample 0.8; scale_pos_weight 578.5	raw	cost-sensitive tree baseline
LightGBM (cs)	Boosted trees	800 trees; num_leaves 64; lr 0.03; subsample 0.8; colsample 0.8; is_unbalance=True	raw	best AP model

MLP-BCE (cs)	MLP	hidden (64,32); dropout 0.2; batch 32768; lr 1e-3; positive weight 578.5	standardized	weighted BCE objective
MLP-Focal (cs)	MLP	hidden (64,32); dropout 0.2; batch 65536; lr 1e-3; alpha 0.95; gamma 2.0	standardized	focal loss objective
MLP-nnPU	MLP	hidden (64,32); dropout 0.2; unlabeled batch 8192; lr 1e-3; class prior pi=0.001725	standardized	simulated PU protocol
AutoEncoder (one-class)	Autoencoder	30-16-8-16-30; batch 32768; lr 1e-3; MSE reconstruction loss	standardized	trained on normals only

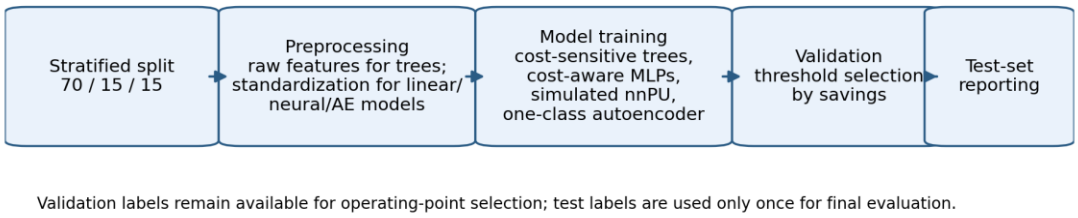

Fig. 2. End-to-end experimental pipeline used in the revised benchmark.

2.5 Reporting scope

This is a single-split benchmark rather than a repeated-run study. The reported values are therefore point estimates on one stratified partition, not confidence intervals over multiple resamples or temporal slices.

A plain unweighted MLP was explored during drafting but showed degenerate validation behavior under the present class ratio and is not retained in the final tables. The deep-learning comparison focuses on the cost-aware objectives that are operationally meaningful in this setting.

3. Results and Discussion

3.1 Ranking performance

Table IV and Fig. 3 show that LightGBM attains the highest AP (0.824), with XGBoost close behind (0.819). Logistic regression records the highest ROC-AUC (0.968) despite lower AP and lower savings than XGBoost. That mismatch is precisely why ROC-AUC should not be treated as the sole model-selection criterion under extreme imbalance [11], [12].

Among neural baselines, nnPU provides the strongest ranking performance (AP 0.784), followed by weighted BCE (0.754) and focal loss (0.736). The one-class autoencoder is much weaker by AP (0.276), although its ROC-AUC remains deceptively high because the negative class is so large.

Table IV. Test-set ranking performance (higher is better).

Model	AP	ROC-AUC
LightGBM (cs)	0.824154	0.940650
XGBoost (cs)	0.818874	0.964928
LogReg (balanced)	0.792092	0.967730
MLP-nnPU	0.784433	0.960914
MLP-BCE (cs)	0.753871	0.959289
MLP-Focal (cs)	0.735722	0.939697
AutoEncoder (one-class)	0.275648	0.947247

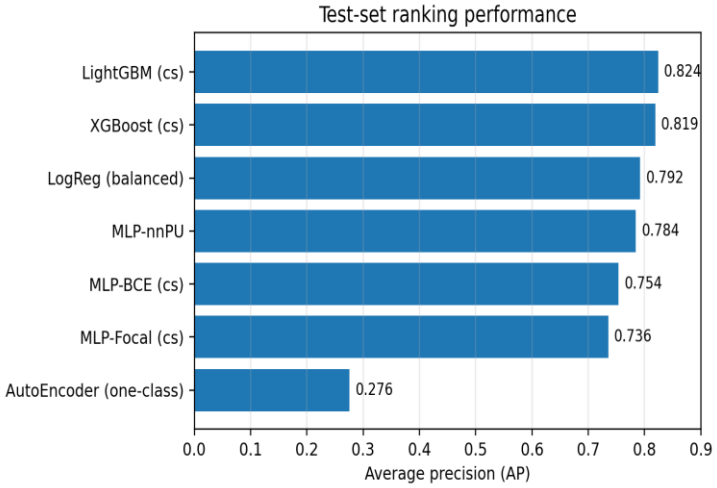

Fig. 3. Average precision on the test set for each evaluated model.

3.2 Recall at fixed false-positive rates

At the strictest operating point, FPR = 0.1 percent, XGBoost reaches 0.851 recall and LightGBM reaches 0.824 recall. Logistic regression and nnPU remain competitive in this regime, whereas the autoencoder captures fewer than one-third of frauds. These results reinforce the strength of tree ensembles and cost-aware neural models on this tabular benchmark.

Table V. Recall at fixed false-positive rates on the test set.

Model	Recall@FPR=0.001	Recall@FPR=0.005	Recall@FPR=0.010
LightGBM (cs)	0.824324	0.837838	0.851351
XGBoost (cs)	0.851351	0.851351	0.851351
LogReg (balanced)	0.837838	0.851351	0.864865
MLP-nnPU	0.824324	0.864865	0.878378
MLP-BCE (cs)	0.810811	0.837838	0.851351
MLP-Focal (cs)	0.797297	0.810811	0.810811

AutoEncoder (one-class)	0.297297	0.554054	0.621622

3.3 Business utility and model selection

Table VI and Fig. 4 make the central point of the study: the model with the best AP is not the model with the best business utility. XGBoost yields the greatest test savings (4592.35 cost units), while LightGBM, despite the best AP, yields lower savings (3867.35). Logistic regression is also strong by savings, reaching 4463.38. The reason is that thresholded deployment depends not only on ranking but also on where the score distribution places high-value fraud and low-cost review volume. LightGBM chooses a much stricter alert stream, which reduces false positives but misses more costly fraud under the stated cost model.

Table VI. Cost-savings results using validation-optimized thresholds.

Model	Optimal threshold (val)	Test cost	Test savings	Savings rate
LogReg (balanced)	0.737195	4019.98	4463.38	52.61%
XGBoost (cs)	0.026470	3891.01	4592.35	54.13%
LightGBM (cs)	0.002802	4616.01	3867.35	45.59%
MLP-BCE (cs)	0.507668	4878.91	3604.45	42.49%
MLP-Focal (cs)	0.338976	4270.07	4213.29	49.67%
MLP-nnPU	0.001218	4284.51	4198.85	49.50%
AutoEncoder (one-class)	2.634090	5092.08	3391.28	39.98%

Baseline cost on the test split (no screening) = 8483.36.

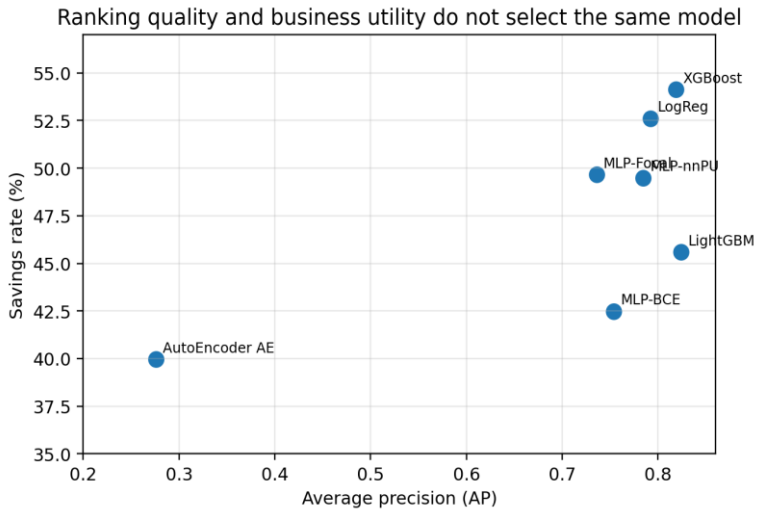

Fig. 4. Ranking quality and business utility do not select the same model: AP versus savings rate on the test split.

3.4 Cost-optimal operating points

At the validation-selected threshold, XGBoost flags 1,287 transactions and captures 66 of the 74 frauds in the test set. That corresponds to recall 66/74 = 0.892 and precision 66/1287 = 0.051. Although this precision is

low in absolute terms, it remains utility-improving when the review cost is only one unit per alert.

LightGBM chooses a far stricter operating point: only 139 alerts, of which 61 are true frauds. Its reviewed queue is much cleaner, but the additional missed fraud carries enough monetary loss to reduce overall savings under the present assumptions.

Table VII. Confusion-matrix components at the cost-optimal threshold.

Model	TP	FP	FN	TN
LogReg (balanced)	64	355	10	42293
XGBoost (cs)	66	1221	8	41427
LightGBM (cs)	61	78	13	42570
MLP-BCE (cs)	64	855	10	41793
MLP-Focal (cs)	59	69	15	42579
MLP-nnPU	66	1528	8	41120
AutoEncoder (one-class)	60	794	14	41854

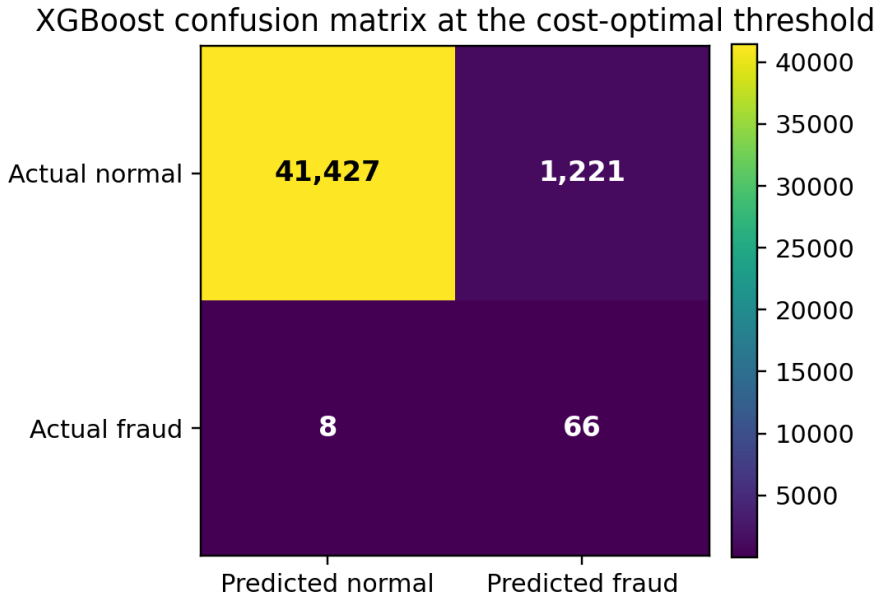

Fig. 5. Confusion matrix for XGBoost at the cost-optimal threshold.

3.5 Deep-objective comparison

Within the neural family, the preferred objective depends on the metric. nnPU produces the best ranking performance among deep models, whereas focal loss produces the largest savings after threshold selection. Weighted BCE remains competitive and slightly cheaper to train than nnPU.

The practical implication is that objective choice affects not only AP but also the calibration and tail behavior of the score distribution, which in turn affects the downstream decision threshold.

Table VIII. Cost-aware deep-learning comparison.

Variant	AP	ROC-AUC	Test savings	Train time (s)
MLP-BCE (cs)	0.753871	0.959289	3604.45	32.6
MLP-Focal (cs)	0.735722	0.939697	4213.29	39.3
MLP-nnPU	0.784433	0.960914	4198.85	55.4

3.6 One-class autoencoder baseline

The one-class autoencoder reduces cost relative to the no-screening baseline, but it remains materially behind all supervised models by both AP and savings. This is consistent with prior anomaly-detection literature [5], [20]-[22]: reconstruction error can provide a useful label-light signal, but it is not a substitute for supervised learning when reliable labels are available.

3.7 Feature importance and training cost

XGBoost assigns the highest gain to V14 and V12, followed by V4, V10, and V20. Amount also appears among the top-10 features, which is operationally plausible because large transactions contribute disproportionately to the cost objective. Training times remain modest for all models on this benchmark: each method fits in under a minute on the reported setup.

Table IX. Training time and model size.

Model	Train time (s)	Complexity unit	Complexity value
LogReg (balanced)	1.9	coefficients	31
XGBoost (cs)	13.2	trees	200
LightGBM (cs)	52.0	trees	800
MLP-BCE (cs)	32.6	trainable params	4097
MLP-Focal (cs)	39.3	trainable params	4097
MLP-nnPU	55.4	trainable params	4097
AutoEncoder (one-class)	39.5	trainable params	1286

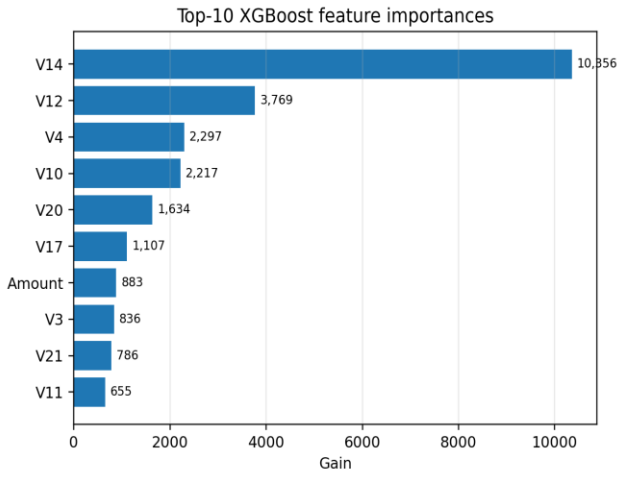

Fig. 6. Top-10 XGBoost feature importances by gain.

4. Limitations

The cost model is deliberately simple. A fixed review cost of one unit and a missed-fraud loss equal to transaction amount do not capture recovery rates, chargebacks, customer attrition, investigator capacity, or organized fraud rings [4]. Different assumptions would change the deployed threshold and may also change the preferred model.

The benchmark itself is also limited. Features are anonymized PCA components plus Time and Amount; merchant context, customer history, device features, graph structure, and delayed labels are absent. The conclusions therefore apply most directly to transaction-level tabular screening, not to full production fraud stacks.

Finally, this is a single-split comparative study. It does not estimate uncertainty across repeated resamples or temporal splits, and the PU experiment is simulated rather than natural. The results are therefore best interpreted as a controlled benchmark demonstration of how ranking metrics and business utility can diverge.

5. Conclusion

On this benchmark, cost-sensitive learning and validation-based threshold selection matter as much as model family. LightGBM achieves the best AP, logistic regression achieves the best ROC-AUC, and XGBoost achieves the best monetary savings. That ordering is the main substantive result: under extreme imbalance, ranking metrics and deployment utility do not necessarily agree.

Among the neural baselines, nnPU is strongest for ranking and focal loss is strongest for savings, while the one-class autoencoder remains a useful but clearly inferior label-light baseline. For deployment-oriented fraud research, the most defensible reporting protocol is to pair rare-event ranking metrics with an explicit cost model and a validation-selected operating point.

References

[1] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating Probability with Undersampling for Unbalanced Classification," in Proc. IEEE Symp. Series Comput. Intell., 2015, pp. 159-166.

[2] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," Statist. Sci., vol. 17, no. 3, pp. 235-255, 2002.

[3] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," Decis. Support Syst., vol. 50, no. 3, pp. 559-569, 2011.

[4] B. Baesens, V. Van Vlasselaer, and W. Verbeke, Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques. Hoboken, NJ, USA: Wiley, 2015.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1-58, 2009.

[6] C. C. Aggarwal, Outlier Analysis, 2nd ed. Cham, Switzerland: Springer, 2013.

[7] Yunhe Li, “Risk-Sensitive Offline Reinforcement Learning for Stable ABR QoE Improvements on Real HSDPA and LTE Traces”, JACS, vol. 3, no. 4, pp. 1–11, Apr. 2023, doi: 10.69987/JACS.2023.30401.

[8] K. M. Ting, "An instance-weighting method to induce cost-sensitive trees," IEEE Trans. Knowl. Data Eng., vol. 14, no. 3, pp. 659-665, 2002.

[9] C. Drummond and R. C. Holte, "Cost curves: An improved method for visualizing classifier performance," Mach. Learn., vol. 65, no. 1, pp. 95-130, 2006.

[10] D. J. Hand, "Measuring classifier performance: a coherent alternative to the area under the ROC curve," Mach. Learn., vol. 77, no. 1, pp. 103-123, 2009.

[11] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in Proc. Int. Conf. Mach. Learn. (ICML), 2006, pp. 233-240.

[12] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," PLoS ONE, vol. 10, no. 3, p. e0118432, 2015.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," J. Artif. Intell. Res., vol. 16, pp. 321-357, 2002.

[14] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 785-794.

[15] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 3146-3154.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2999-3007.

[17] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2008, pp. 213-220.

[18] M. C. du Plessis, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2014, pp. 703-711.

[19] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 1674-1684.

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504-507, 2006.

[21] Yifei Lu, Jinyi Mu, and Thao Tran, “Uncertainty-Aware Uplift Modeling for Safer Marketing Targeting: Conformal Prediction and Bayesian Calibration with LCB Policies”, JACS, vol. 4, no. 5, pp. 84–101, May 2024, doi: 10.69987/JACS.2024.40507.

[22] Meng-Ju Kuo, Boning Zhang, and Haozhe Wang, “Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s)”, JACS, vol. 3, no. 8, pp. 39–53, Aug. 2023, doi: 10.69987/JACS.2023.30804.

[23] Credit Card Fraud Detection Dataset, Zenodo, record 7395559, 2022.