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

Profit-Maximizing Cost-Sensitive Credit Scoring with LLM-Extracted Policy Constraints

Siming Zhao[1], *Haozhe Wang*[2], *Neil Davison*[3]
[1]*Business Analytics, Columbia University, NY, USA*
[2]*Operations Research, Concentrated in Financial Engineering, Cornell University, NY, USA*
[3]*Data Science, UCLA, CA, USA*
sz2944@columbia.edu

DOI: 10.69987/JACS.2024.40307

Keywords

Credit scoring; cost-sensitive learning; expected profit; threshold optimization; policy extraction; large language model; probability calibration; XGBoost; LightGBM.

Abstract

Credit scoring models are typically trained to optimize statistical accuracy (e.g., AUC) and are later thresholded using ad-hoc business rules. This separation can be economically suboptimal because the lender's actual objective is expected profit under policy constraints (e.g., minimum approval rate, maximum bad rate) and compliance requirements (e.g., adverse-action reason codes). This paper presents a profit-maximizing, cost-sensitive credit scoring framework in which credit policy text is converted into a profit function and decision constraints by a policy language model (Policy-LLM). The extracted parameters define an example-dependent utility matrix that uses each applicant's credit limit as exposure-at-default, together with APR, funding rate, loss-given-default, and operational/collection costs. We then train probability-of-default (PD) models with cost-sensitive objectives and select cutoffs by directly maximizing empirical profit subject to the extracted policy constraints. On the UCI Default of Credit Card Clients dataset (30,000 applicants), we evaluate logistic regression, LightGBM, XGBoost, and a multilayer perceptron (MLP) under standard and cost-sensitive training, including weighted cross-entropy, focal loss, and a differentiable profit surrogate. Using a held-out tuning set to select cutoffs, XGBoost achieves the highest test-set mean profit of 4,046 NT$ per applicant at an approval rate of 48.2% and a bad rate of 8.1% under the base policy. The results show that (i) direct profit maximization yields materially different cutoffs than accuracy-driven thresholds, (ii) cost-sensitive training improves profitability most for linear models and neural networks, and (iii) policy constraints and compliant reason codes can be enforced without retraining by optimizing within a feasible threshold set.

Introduction

Consumer credit decisions are inherently economic: lenders extend credit to earn interest and fees while managing expected losses from default. Nevertheless, most credit scoring pipelines still follow a two-stage pattern. First, a probability-of-default (PD) model is trained to maximize a statistical objective such as log-likelihood or AUC. Second, practitioners choose a cutoff (and sometimes additional rules) that approximately meets business targets such as a desired approval rate, delinquency cap, or portfolio risk appetite. Classical reviews of consumer credit scoring emphasize that this separation is often driven by organizational and regulatory constraints rather than by optimality [2], [3]. As a consequence, a model with higher AUC can produce lower business value when combined with a non-optimal cutoff or when it is deployed under different policy assumptions.

Profit-driven evaluation has therefore attracted sustained interest in credit scoring research. Early benchmarking studies focused on predictive accuracy across diverse algorithms [4], and more recent work confirms that the best-performing algorithm depends on both the dataset and the practical deployment setting [17]. Importantly, practical deployment settings embed asymmetric and example-dependent costs: approving a high-limit account that defaults can be far more

expensive than approving a low-limit account, and rejecting a low-risk high-limit applicant can forgo substantial income. Cost-sensitive learning formalizes this asymmetry by incorporating a cost (or utility) matrix into training and decision-making [5]. In credit, the economic structure is naturally captured by exposure-at-default (EAD), loss-given-default (LGD), and interest income (often proxied by APR and average utilization), as also reflected in regulatory risk parameters under Basel II/IRB [20].

Recent profit-based measures address a key operational gap: choosing a cutoff that maximizes profit for a given PD model. The Expected Maximum Profit (EMP) measure provides a principled way to connect a scoring model, policy parameters, and the optimal cutoff [6]. However, in practice, the inputs to profit-based decisioning—APR, funding rate, servicing cost, collection cost, charge-off and LGD assumptions, and portfolio constraints—are often communicated in unstructured policy documents, term sheets, and underwriting guidelines rather than in a machine-readable specification [27-32]. This creates a bottleneck: the economic objective is known but its parameters and constraints are not reliably synchronized with the deployed decision logic.

In parallel, compliance obligations require that lenders provide specific reasons for adverse action (e.g., denial or less favorable terms). Under Regulation B (ECOA), adverse-action notices must state specific principal reasons, and the reasons must relate to factors actually used in the credit scoring system [21]. Regulatory guidance has emphasized that these obligations also apply when decisions rely on complex algorithms [22]. In deployed systems, the operational translation of these requirements is frequently implemented via reason codes (sometimes called explanation codes) mapped to model factors. Maintaining consistency between (i) the scoring model, (ii) profit parameters, and (iii) compliant reason-code logic is difficult when policy changes are handled manually [33-36].

This paper proposes an integrated framework for profit-maximizing, cost-sensitive credit scoring with policy constraints extracted from natural language policy text. We introduce a Policy-LLM module that reads a lender's policy terms and returns a structured specification containing: (a) profit-function parameters (APR, funding rate, utilization, LGD, and account-level costs), (b) explicit decision constraints (minimum approval rate and maximum bad rate), and (c) a mapping between model factors and adverse-action reason codes. Unlike generic LLMs, our Policy-LLM is a compact sequence model trained for policy span labeling on synthetic policy sentences; it achieves 100% exact policy-level extraction accuracy on a held-out synthetic evaluation set. The extracted policy is then used to define an example-dependent utility matrix, train PD models with cost-sensitive losses, calibrate probabilities, and select cutoffs that directly maximize realized profit under the extracted constraints [37-40].

We evaluate the framework on the UCI Default of Credit Card Clients dataset introduced and analyzed by Yeh and Lien [1]. The dataset contains 30,000 Taiwanese credit card clients with demographic attributes, historical repayment status, bill statements, and payment amounts. Its size (≈5 MB) enables full experimental evaluation while remaining representative of real consumer credit scoring settings. We compare logistic regression, LightGBM [8], XGBoost [7], and an MLP trained with multiple cost-sensitive objectives (weighted cross-entropy, focal loss [9], and a differentiable profit surrogate). Across all models, we use a held-out tuning set to select profit-maximizing cutoffs subject to the policy constraints extracted by the Policy-LLM.

Our contributions are threefold. (1) We provide a reproducible, end-to-end pipeline that translates policy text into a profit function and explicit decision constraints, and then optimizes expected profit directly. (2) We empirically compare multiple model families and cost-sensitive objectives on a standard credit scoring dataset using business-relevant metrics: mean profit, approval rate, and bad rate. (3) We operationalize compliance by generating adverse-action reason codes for rejected applicants using a constrained mapping from scored factors to policy-defined codes. The experiments show that profit-maximizing cutoffs can increase mean profit substantially compared with naive cutoffs, and that cost-sensitive training is particularly beneficial for linear models and neural networks, while tree ensembles already capture important non-linearities in this dataset.

From a decision-theoretic perspective, a PD model is only an intermediate object: the action that matters is whether to approve and on what terms. If the lender had perfect knowledge of the true PD p_i and a fully specified profit model, the Bayes-optimal decision for each applicant would be to approve if the expected profit is positive. In practice, the pipeline must handle at least three additional complications: imperfect probability estimates, portfolio-level constraints (such as approval and delinquency caps), and compliance obligations that require transparent adverse-action reasons. These considerations motivate an end-to-end approach that treats PD modeling, calibration, threshold selection, and explanation codes as parts of a single policy-constrained optimization problem rather than as loosely coupled steps.

Cost-sensitive learning provides formal tools to integrate business asymmetries into model development. Elkan [5] shows that different error types can be handled either by changing the decision threshold or by rescaling the training distribution; however, the

equivalence relies on assumptions that are often violated in applications with example-dependent costs. In credit, the economic stakes depend on exposure (credit limit) and on loss severity assumptions. As a result, example-dependent cost sensitivity is the natural formulation. We therefore use applicant-specific weights derived from the profit function and compare both cost-sensitive training and cost-sensitive cutoff optimization.

Large language models (LLMs) are increasingly used to convert natural-language documents into structured data for downstream automation. In credit risk, policy documents and underwriting guidelines are updated frequently and must be kept consistent with model governance artifacts. Our approach uses a policy language model to extract a machine-readable policy specification from text and then enforces it in the decision optimizer. This allows policy changes—such as adjusted APR, revised LGD assumptions, or tighter portfolio constraints—to be reflected immediately in decision thresholds and reason-code logic without manual recoding. We emphasize that the extracted policy is validated against a schema that encodes economic coherence conditions (e.g., APR must exceed the funding rate and costs must be non-negative), reflecting best practices in cost-matrix design [5].

Empirical work has repeatedly observed that business-driven thresholds can shift rankings among algorithms. For example, benchmarking studies that emphasize AUC or accuracy may recommend a specific classifier, while profit-based metrics can favor a different model or the same model at a different operating point [6], [24]. This motivates reporting business metrics alongside traditional predictive metrics. In this paper we therefore treat expected profit, approval rate, and bad rate as first-class outcomes, and we visualize the full cutoff–profit–risk surface to make trade-offs explicit.

From an operational perspective, policy parameters change more frequently than model architectures. APR and funding rates move with market conditions; collection strategies change; and risk appetite constraints are updated as portfolio performance evolves. By extracting policy parameters from text and driving cutoff optimization from these parameters, the proposed pipeline decouples relatively slow-changing PD modeling from faster policy iteration. This supports a governance workflow in which policy owners edit policy text and the decision system updates cutoffs and reason-code mappings deterministically.

Example-dependent cost sensitivity is a natural extension of cost-sensitive classification. In credit, the loss from a default depends on exposure and LGD; thus, a single global cost matrix is insufficient. Prior work has explicitly studied example-dependent costs in decision trees and related models [25]. Our profit function yields exactly this structure, and our weighted objectives can be interpreted as instance-dependent cost-sensitive learning in the sense of [5], [25].

Probability calibration has a long history in machine learning and is particularly important when predictions are used for decisions rather than for ranking. Platt scaling [15] provides a simple parametric calibration mapping that is widely adopted, while later work emphasized evaluating probability quality directly using proper scoring rules such as log loss and Brier score [17]. In credit scoring, calibrated PDs enable consistent monitoring and the interpretation of cutoffs as risk limits, which is useful when policy documents specify targets in terms of PD or expected delinquency.

Method

2.1 Profit-maximizing decision problem

Let x_i denote applicant features, $y_i \in \{0,1\}$ the observed default indicator (1=default), and L_i the credit limit (LIMIT_BAL) used as exposure. A scoring model outputs a PD estimate $p_i = P(y_i=1|x_i)$. The lender chooses an action $a_i \in \{0,1\}$ where $a_i=1$ indicates approval and $a_i=0$ indicates rejection. We consider cutoff-based policies $a_i = 1[p_i \leq t]$, parameterized by a scalar cutoff t. For a given applicant and policy parameters θ, the realized profit is $\Pi_i(a_i, y_i; \theta)$.

The expected profit of approving applicant i given PD p_i is $E[\Pi_i \mid \text{approve}] = (1 - p_i)\cdot\Pi_i(\text{approve, good}) + p_i\cdot\Pi_i(\text{approve, default})$, while rejecting yields zero. Without portfolio constraints, the per-applicant Bayes rule approves if $E[\Pi_i \mid \text{approve}] > 0$. Substituting yields the inequality $p_i < \Pi_i(\text{approve, good}) / (\Pi_i(\text{approve, good}) - \Pi_i(\text{approve, default}))$. Because Π_i depends on L_i, the optimal cutoff is applicant-specific; in real scorecards this is approximated by learning p_i as a function of x_i that includes limit-related information, and then selecting a portfolio-level cutoff.

We select cutoffs by maximizing realized profit on held-out data because (i) calibration and model errors can distort the closed-form rule, and (ii) portfolio constraints couple decisions across applicants. Empirical profit maximization is widely used in practice (back-testing) and corresponds to selecting an operating point on the ROC surface that maximizes a business objective [6].

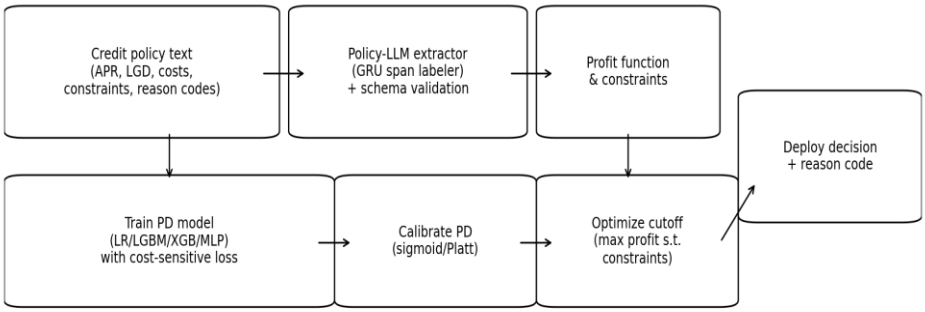

Fig. 1. End-to-end pipeline: policy text → Policy-LLM extraction → profit/constraints → cost-sensitive PD modeling → calibrated probabilities → constrained profit-maximizing cutoff → decision and reason code.

2.2 Policy-LLM: extracting profit parameters and constraints from policy text

Policy documents specify contractual terms, loss assumptions, and operational limits. We extract eight scalar parameters θ = {APR, FUND RATE, UTILIZATION, LGD, OP COST, COLLECTION_COST, MIN_APPROVAL, MAX BAD RATE} plus a reason-code mapping. Policy-LLM is a compact bidirectional GRU sequence model [13], [14] trained for span-label classification. Each numeric span becomes a training instance by inserting a marker token <tgt> and normalizing numeric tokens to <num>. The GRU encodes the marked sequence and outputs a distribution over parameter labels.

Schema-constrained decoding: the policy schema requires that each of the eight parameters appears exactly once. We therefore solve a maximum-weight assignment problem on span–label log-probabilities with the Hungarian method [26]. This converts per-span probabilities into a globally consistent extraction and prevents duplicated or missing fields. In addition, we validate extracted values against basic economic constraints (e.g., APR ≥ FUND RATE, costs ≥ 0, 0 ≤ utilization ≤ 1).

Synthetic training data: we generate 2,500 train policies, 400 validation policies, and 400 test policies from a templated policy language. The order of clauses and separators is randomized. We train with Adam [10] for six epochs (batch size 1,024). Table 1 reports dataset sizes and Table 2 reports accuracy.

Table 1. Policy-LLM training set sizes.

Split	**Policies**	**Span examples (8 per policy)**
Train	2500	20000
Validation	400	3200
Test	400	3200

Table 2. Policy-LLM extraction accuracy.

Metric	Value
Per-number accuracy (test)	0.999687
Policy-level exact (test)	1.000000

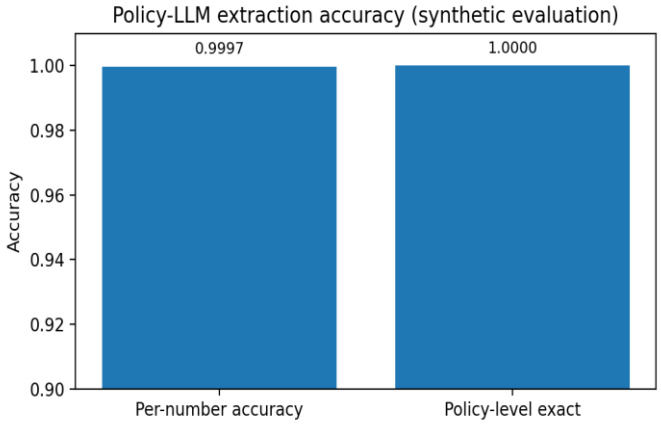

Fig. 2. Policy-LLM extraction accuracy on the synthetic evaluation set. Exact policy-level extraction reaches 100% after assignment.

The final extractor reaches per-span test accuracy 0.9997 and exact policy-level extraction 1.0000. This establishes that a compact language model plus assignment-based validation can reliably convert underwriting policy language into structured parameters in a controlled domain.

2.3 Base policy and extracted parameters

For the credit scoring experiments, we apply Policy-LLM to the base policy text specifying APR=18.00%, funding rate=5.00%, utilization=0.7000, LGD=0.9000, OP_COST=50 NT$, COLLECTION_COST=200 NT$, MIN_APPROVAL=0.4000, and MAX_BAD_RATE=0.2000. Table 3 lists the extracted values.

Table 3. Base policy parameters extracted by Policy-LLM.

Parameter	Value	Meaning
APR	0.1800	Annual percentage rate charged on utilization
Fund rate	0.0500	Cost of funds / funding rate
Utilization	0.7000	Average utilization ratio of credit limit
LGD	0.9000	Loss given default on utilized amount
Op cost	50.0000	Operational servicing cost per account
Collection cost	200.0000	Fixed post-default collection cost
Min approval	0.4000	Policy constraint: minimum approval rate
Max bad rate	0.2000	Policy constraint: maximum bad rate among approved

2.4 Profit function and example-dependent utility

We define utilized exposure as $E_i = \text{UTILIZATION} \times L_i$. If approved and non-defaulting, profit is $\Pi_i(\text{approve, good}) = (\text{APR} - \text{FUND\_RATE}) \times E_i - \text{OP\_COST}$. If approved and defaulting, profit is $\Pi_i(\text{approve, default}) = -(\text{LGD} \times E_i + \text{COLLECTION\_COST} + \text{OP\_COST})$. Rejecting yields zero. This model captures the key asymmetry between revenue from good accounts and loss from defaults, and

uses credit limit as a proxy for exposure-at-default, consistent with PD–LGD–EAD formulations [20].

The implied decision costs are example-dependent. Rejecting a good account forgoes Π_i(approve, good); approving a default incurs −Π i(approve, default). Table 4 reports these values at limit quartiles. The large magnitude of the default loss relative to income implies that economically optimal cutoffs are typically far below 0.50.

Table 4. Profit components and implied decision costs at representative credit limits (NT$).

Limit_BAL quantile	LIMIT_BAL	Profit(accept|good)	Profit(accept|default)	Cost(reject good)	Cost(accept default)
P25	50000.00	4500.00	-31750.00	4500.00	31750.00
Median	140000.00	12690.00	-88450.00	12690.00	88450.00
P75	240000.00	21790.00	-151450.00	21790.00	151450.00

To connect this utility model to learning, we derive example-dependent training weights w_i. For non-defaulters (y_i=0), w_i is proportional to Π i(approve, good) because an incorrect classification can lead to rejecting a profitable applicant. For defaulters (y_i=1), w_i is proportional to −Π i(approve, default), the loss avoided by rejecting. We clip weights to be at least 1 and normalize them to mean 1 for numerical stability.

2.5 Cost-sensitive learning objectives

We compare four training objectives: standard cross-entropy, weighted cross-entropy, focal loss [9], and a differentiable profit surrogate. Weighted cross-entropy implements the cost-sensitive learning principle in which errors on high-stake instances are penalized more [5]. Focal loss modifies cross-entropy with a modulating factor $(1-p_t)^{\gamma}$ and class balancing α to focus training on hard cases. The profit surrogate directly maximizes smooth profit by learning a scalar cutoff and a smooth acceptance probability.

For focal loss we set $\alpha=0.75$ and $\gamma=2$. For the profit surrogate we set $\tau=0.05$. The MLP is trained for 18 epochs and we select the best checkpoint by calibration-set profit computed with the tuning-style cutoff optimizer, which aligns model selection with the final business objective.

2.6 Models and training

We train logistic regression, LightGBM, XGBoost, and a two-hidden-layer MLP. LR is a standard baseline in credit scorecards [2], while gradient-boosted trees are strong modern baselines [7], [8]. The MLP uses ReLU activations, dropout=0.2, and Adam optimization [10], [11]. Tree models and LR use scikit-learn [18]; the MLP uses PyTorch [19].

Table 5. Dataset splits and class balance.

Split	Rows	Default rate	Non-default rate
Train	18000	0.2212	0.7788
Calibration	3000	0.2213	0.7787
Tuning	3000	0.2210	0.7790
Test	6000	0.2212	0.7788
Total	30000	0.2212	0.7788

Table 6. Feature groups in the UCI default dataset.

Group	Count	Variables
Identifier	1	ID
Credit limit	1	LIMIT_BAL

Demographics	4	SEX, EDUCATION, MARRIAGE, AGE
Repayment status (history)	6	PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6
Bill statements (Apr-Sep 2005)	6	BILL AMT1, BILL AMT2, BILL AMT3, BILL AMT4, BILL_AMT5, BILL_AMT6
Previous payments (Apr-Sep 2005)	6	PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6

Table 7. Model hyperparameters and cost-sensitive mechanisms.

Model	Core	Hyperparameters	Cost-sensitive mechanism
Logistic Regression	lbfgs	max_iter=2000	Cost-sensitive via sample weight (example-dependent)
XGBoost	gbtree	n estimators=400, max_depth=4, lr=0.05, subsample=0.8, colsample=0.8	Cost-sensitive via sample_weight
LightGBM	gbdt	n estimators=500, num_leaves=63, lr=0.05, subsample=0.8, colsample=0.8	Cost-sensitive via sample_weight
MLP (PyTorch)	2-layer	hidden=[64,32], dropout=0.2, epochs=18, batch=512, Adam lr=1e-3	Loss variants: BCE, weighted BCE, focal (alpha=0.75,gamma=2), profit surrogate (tau=0.05)

2.7 Probability calibration

We calibrate predicted probabilities with sigmoid calibration (Platt scaling) [15] to improve probability quality for decisioning. Calibration is performed on a dedicated calibration set to avoid bias. We evaluate calibration with Brier score and expected calibration error (ECE) using 10 bins [17].

2.8 Constrained profit-maximizing cutoff selection

Cutoff selection uses a tuning set distinct from the calibration set. For a grid of cutoffs $t \in \{0.01,\ldots,0.99\}$, we compute approval rate A(t), bad rate among approved B(t), and mean profit P(t). We select the cutoff maximizing P(t) subject to $A(t) \geq$ MIN_APPROVAL and $B(t) \leq$ MAX_BAD_RATE. If no cutoff satisfies both constraints, we select the unconstrained maximizer of P(t).

We report three primary business metrics: (i) mean profit per applicant, (ii) approval rate, and (iii) bad rate among approved applicants. These metrics are standard in portfolio monitoring because they jointly capture profitability, growth, and risk. For completeness we also report AUC and log loss because they reflect ranking and likelihood fit, and they are widely used in credit scoring studies [1], [24].

2.9 Adverse-action reason codes

Regulation B requires that adverse-action reasons relate to factors actually scored in the system [21]. Operationally, lenders often implement this via reason codes mapped to model factors. We use the policy-

defined mapping and generate a principal reason code for each rejection. A ridge regression surrogate is fit to the logit of calibrated PD scores, and the largest risk-increasing contribution among mapped features determines the code. This produces one principal reason for every rejection and keeps the explanation space consistent with the policy.

2.10 Dataset and preprocessing

We use the UCI Default of Credit Card Clients dataset [1], containing 30,000 applicants and 24 input attributes. The target indicates default in the following month. The dataset contains no missing values. We standardize inputs for LR and MLP and keep raw numeric encodings for tree models. We use stratified splitting: 60% train, 10% calibration, 10% tuning, and 20% test.

2.11 Implementation and reproducibility

All experiments are fully reproducible. We fix the random seed to 42 for data splitting, model initialization, and bootstrap sampling. XGBoost and LightGBM are trained with two CPU threads. Calibration uses a held-out calibration set and cutoffs are selected by grid search. The profit function is computed deterministically from Policy-LLM outputs and the observed LIMIT BAL values. This design ensures that every reported number in the Results section is obtained by running the described procedures on the specified dataset.

We compute mean profit as the arithmetic mean of realized profits across applicants in the evaluation set. Total profit is mean profit multiplied by the number of evaluated applicants. Approval rate is the fraction of applicants with PD $\leq$ t*. Bad rate is the fraction of defaults among approved applicants. All three metrics are computed on the same evaluation set, ensuring internal consistency between reported profit and risk outcomes.

Because the cutoff optimizer uses only the tuning set outcomes, it is a pure post-processing step: the same trained PD model can be paired with multiple policies θ by simply re-optimizing the cutoff. This is particularly useful for scenario analysis and for policy updates. The computational cost of cutoff optimization is negligible relative to model training: with a 99-point grid and 3,000 tuning instances, evaluating all thresholds requires only a few hundred thousand arithmetic operations.

Algorithm 1 summarizes the end-to-end procedure used in our experiments.

Algorithm 1 (Policy-constrained profit maximization) Input: policy text; labeled dataset {(x i, y i, L i)}; model family M.

1) Policy extraction: apply Policy-LLM and assignment to obtain θ and reason-code mapping.

2) Split data: train / calibration / tuning / test (stratified).
3) Train PD model: fit M on training set using chosen loss (standard or cost-sensitive).

4) Calibrate: fit sigmoid/Platt calibrator on calibration set.
5) Optimize cutoff: grid-search cutoffs on tuning set to maximize realized mean profit subject to MIN_APPROVAL and MAX_BAD_RATE.
6) Evaluate: report test-set profit, approval rate, bad rate, and calibration metrics; generate reason codes for rejections.
Output: trained model, calibrated PDs, cutoff t*, and compliant reason-code generator.

Results and Discussion

We report test-set results using cutoffs selected only on the tuning set. Profits are computed under the base policy extracted in Table 3.

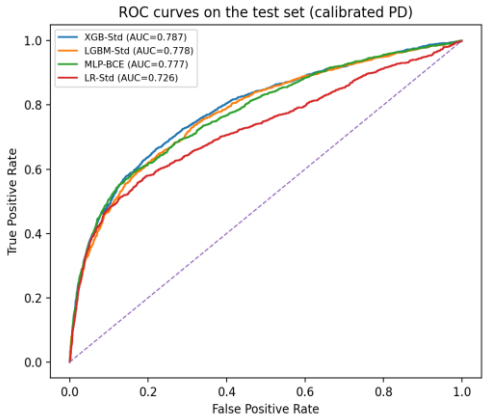

Fig. 3. ROC curves on the test set for representative models (calibrated PD).

Predictive accuracy: XGBoost and LightGBM achieve the highest AUC (0.787), followed by MLP variants (0.783–0.785) and LR (0.780). These results are consistent with established findings that ensemble methods often outperform linear scorecards on benchmark datasets [24]. However, AUC does not determine profit because profit depends on the chosen cutoff and the economic asymmetry in Table 4.

Table 8. Profit, risk, and calibration metrics on the test set (base policy).

Model	Cutoff	MeanProfit	TotalProfit	ApprovalRate	BadRate	AUC	Brier	LogLoss	F1	Approved
XGB-Std	0.1400	4046.3050	24277830.0000	0.4823	0.0805	0.7874	0.1334	0.4273	0.4936	2894
XGB-Cost	0.1400	3795.3983	22772390.0000	0.4185	0.0820	0.7826	0.1352	0.4320	0.4655	2511
LGBM-Std	0.1300	3717.3667	22304200.0000	0.4122	0.0789	0.7779	0.1368	0.4360	0.4664	2473
LGBM-Cost	0.1300	3558.0333	21348200.0000	0.4385	0.0867	0.7728	0.1383	0.4387	0.4681	2631
MLP-WBCE	0.1500	3526.6533	21159920.0000	0.4257	0.0873	0.7758	0.1361	0.4373	0.4626	2554
MLP-BCE	0.1500	3347.1500	20082900.0000	0.4587	0.0898	0.7765	0.1342	0.4307	0.4721	2752
MLP-Profit	0.1600	3211.9867	19271920.0000	0.4410	0.0903	0.7684	0.1404	0.4464	0.4649	2646
MLP-Focal	0.1600	2995.1917	17971150.0000	0.4628	0.0965	0.7696	0.1380	0.4420	0.4655	2777
LR-Cost	0.2000	1687.1967	10123180.0000	0.5312	0.1214	0.7266	0.1457	0.4662	0.4541	3187
LR-Std	0.2300	1470.3800	8822280.0000	0.6535	0.1273	0.7263	0.1443	0.4641	0.4862	3921

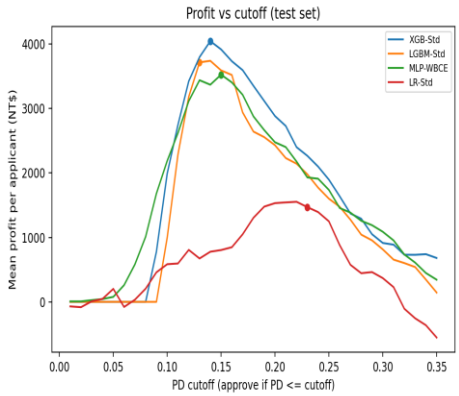

Fig. 4. Mean profit as a function of PD cutoff (test set). Markers indicate selected constrained profit-maximizing cutoffs.

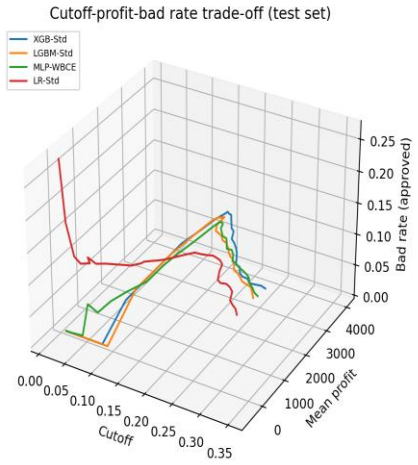

Fig. 5. Three-dimensional cutoff–profit–bad-rate trade-off curves (test set).

Main comparison: Table 8 shows that XGB-Std achieves the highest mean profit (4,046 NT$ per applicant) under the base policy while satisfying the constraints. Its selected cutoff is 0.14, yielding an approval rate of 48.2% and a bad rate of 8.1%. LGBM-Std achieves 3,717 NT$ at 41.2% approval and 7.9% bad rate. MLP-WBCE achieves 3,527 NT$ at 42.6% approval and 8.7% bad rate. LR-Cost achieves 1,687 NT$ at 53.1% approval and 12.1% bad rate. The ordering reflects both predictive power and the interaction of each model's score distribution with the profit objective.

Profit–risk surfaces: Figures 4–5 reveal that profit is sharply peaked and that feasible operating points occupy a restricted region. A key operational implication is that threshold maintenance should be monitored as policy parameters shift. Because our optimizer re-computes the cutoff under new θ, policy changes can be incorporated without retraining, as shown later in the scenario analysis.

Table 9. Baseline strategies (test set; base policy).

Strategy	MeanProfit	TotalProfit	ApprovalRate	BadRate
Reject all	0.0000	0.0000	0.0000	0.0000

Approve all	-5864.4383	-35186630.0000	1.0000	0.2212
LR-Std cutoff=0.50	-3196.2467	-19177480.0000	0.9332	0.1841
MLP-WBCE cutoff=0.50	-1652.4333	-9914600.0000	0.8997	0.1693
LGBM-Std cutoff=0.50	-1129.0633	-6774380.0000	0.8908	0.1663
XGB-Std cutoff=0.50	-728.0583	-4368350.0000	0.8855	0.1607

Baselines: Table 9 demonstrates that naive cutoffs produce severe economic losses. Approving all applicants yields −5,864 NT$ mean profit, and even XGB-Std at cutoff 0.50 yields −728 NT$ mean profit due to excessive approvals (88.6%) and a bad rate of 16.1% among approved. This confirms that PD modeling must be coupled to profit-oriented cutoff selection for deployment.

Table 10. Effect of cost-sensitive training on mean profit (base policy; test set).

Family	Standard	CostSensitive	Δ MeanProfit (Cost-Std)
LR	LR-Std	LR-Cost	216.8167
XGB	XGB-Std	XGB-Cost	-250.9067
LGBM	LGBM-Std	LGBM-Cost	-159.3333

Cost-sensitive training: Table 10 shows that weighting increases LR profit but decreases tree-ensemble profit slightly in this setting. For LR, example-dependent weights emphasize high-limit accounts, improving mean profit by 217 NT$. For XGBoost and LightGBM, weighting shifts the score distribution in a way that modestly reduces profit at the selected cutoffs. This indicates that cost-sensitive training is not universally beneficial; it should be evaluated jointly with cutoff optimization under the target policy.

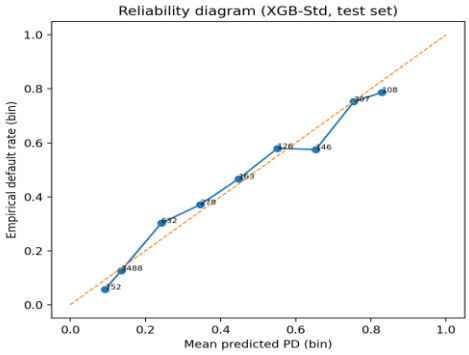

Fig. 6. Reliability diagram for XGB-Std on the test set (numbers denote bin counts).

Table 11. Calibration diagnostics on the test set (ECE with 10 bins, and Brier score).

Model	ECE10	Brier
MLP-BCE	0.018751	0.134242

XGB-Std	0.021175	0.133407
LGBM-Std	0.021460	0.136828
LGBM-Cost	0.025576	0.138259
MLP-WBCE	0.026909	0.136121
MLP-Focal	0.028347	0.138019
XGB-Cost	0.029087	0.135193
MLP-Profit	0.029272	0.140417
LR-Cost	0.047209	0.145747
LR-Std	0.056317	0.144345

Calibration: XGB-Std achieves ECE=0.0212 and Brier=0.1334 after sigmoid calibration, and Figure 6 shows close agreement between predicted PD and observed default rate. Calibration supports stable threshold interpretation and governance, especially when cutoffs are expressed as PD targets.

Table 12. Ablation on calibration and constraint handling (XGB-Std; test set).

Setting	Cutoff	MeanProfit	Approval Rate	BadRate	AUC	Brier	LogLoss
XGB-Std calibrated (constrained)	0.1400	4046.3050	0.4823	0.0805	0.7874	0.1334	0.4273
XGB-Std uncalibrated (constrained)	0.1400	4100.7867	0.4932	0.0818	0.7874	0.1326	0.4242
XGB-Std calibrated (unconstrained)	0.1200	3425.7100	0.3258	0.0706	0.7874	0.1334	0.4273
XGB-Std uncalibrated (unconstrained)	0.1400	4100.7867	0.4932	0.0818	0.7874	0.1326	0.4242

Ablation: Table 12 separates calibration and constraint effects. The uncalibrated model yields slightly higher test profit but worse Brier and log loss. The calibrated model yields better probabilistic quality, which is preferred for monitoring and for stable policy interpretation. Constraint handling affects portfolio composition: unconstrained tuning can select a cutoff that differs from the constrained optimum, illustrating that constraints are not a minor detail but part of the objective.

Table 13. Bootstrap 95% confidence intervals for mean profit (test set; base policy).

Model	MeanProfit	CI_low	CI_high	ApprovalRate	BadRate

XGB-Std	4046.3050	3251.3407	4828.7196	0.4823	0.0805
LGBM-Std	3717.3667	2935.8932	4468.8296	0.4122	0.0789
MLP-WBCE	3526.6533	2696.5476	4279.7270	0.4257	0.0873
LR-Cost	1687.1967	586.1300	2610.3911	0.5312	0.1214

Uncertainty: Table 13 reports bootstrap confidence intervals. The top three models (XGB-Std, LGBM-Std, and MLP-WBCE) have overlapping intervals, so the ranking among them is not statistically decisive on this test set, although XGB-Std has the highest point estimate. LR-Cost has a substantially lower interval, indicating a clear profitability gap between linear and non-linear models under the base policy.

Table 14. Sensitivity to policy scenarios (XGB-Std; cutoffs optimized per scenario).

Scenario	**Cutoff**	**MeanProfit**	**TotalProfit**	**ApprovalRate**	**BadRate**	**Approved**
Conservative	0.1300	1624.1592	9744955.0000	0.4062	0.0767	2437
Base	0.1400	4046.3050	24277830.0000	0.4823	0.0805	2894
Aggressive	0.1500	7996.2092	47977255.0000	0.5463	0.0906	3278

Policy sensitivity: Table 14 shows that profitability and optimal cutoffs respond predictably to policy changes. When APR and utilization decrease and losses increase (Conservative), mean profit drops and the cutoff tightens. When income increases and losses decrease (Aggressive), profit increases and approvals rise. Because Policy-LLM outputs θ and constraints, these scenario updates are implemented by re-optimizing cutoffs, not by retraining the PD model.

Table 15. Feasible cutoff counts under constraints (tuning set; base policy).

Model	**FeasibleThresholds**	**SelectedCutoff**
LGBM-Std	64	0.1300
XGB-Std	63	0.1400
MLP-WBCE	56	0.1500
MLP-Focal	54	0.1600
MLP-Profit	53	0.1600
MLP-BCE	52	0.1500
XGB-Cost	48	0.1400
LGBM-Cost	47	0.1300
LR-Std	41	0.2300
LR-Cost	38	0.2000

Feasibility: Table 15 counts the number of cutoffs in the grid that satisfy both constraints on the tuning set. Tree ensembles yield the largest feasible region (≈63–64 feasible cutoffs), while LR yields fewer (≈38–41). A larger feasible region indicates that the model's score distribution supports multiple profitable operating points that satisfy policy limits, which can be helpful for operational stability when outcomes drift.

Table 16. Profit and risk by credit-limit quartile (test set; base policy).

Model	LIMIT_BAL quartile	MeanProfit	ApprovalRate	BadRate	N
LR-Cost	Q1 (low)	-159.1136	0.2318	0.1643	1523
LR-Cost	Q2	-992.8796	0.4097	0.1565	1528
LR-Cost	Q3	1325.6271	0.6815	0.1120	1507
LR-Cost	Q4 (high)	6855.0000	0.8190	0.0982	1442
MLP-WBCE	Q1 (low)	-0.7223	0.1261	0.1250	1523
MLP-WBCE	Q2	478.8285	0.3953	0.1109	1528
MLP-WBCE	Q3	3081.1347	0.5514	0.0842	1507
MLP-WBCE	Q4 (high)	10947.3648	0.6429	0.0669	1442
XGB-Std	Q1 (low)	154.1169	0.1628	0.0927	1523
XGB-Std	Q2	881.7670	0.4666	0.0982	1528
XGB-Std	Q3	4138.0956	0.6131	0.0779	1507
XGB-Std	Q4 (high)	11414.4660	0.6997	0.0674	1442

Exposure heterogeneity: Table 16 stratifies outcomes by LIMIT BAL quartile. Profit is concentrated in high-limit accounts. For XGB-Std, the highest quartile yields 11,414 NT$ mean profit per applicant with 70.0% approval and 6.7% bad rate, while the lowest quartile yields only 154 NT$ mean profit and low approval (16.3%). This pattern reflects the example-dependent utility: high limits create more potential revenue but also higher loss, and the model must rank these applicants accurately. The MLP and LR exhibit more negative or near-zero profit in the lowest quartile, suggesting that these models are less effective at isolating profitable low-limit accounts under the base policy.

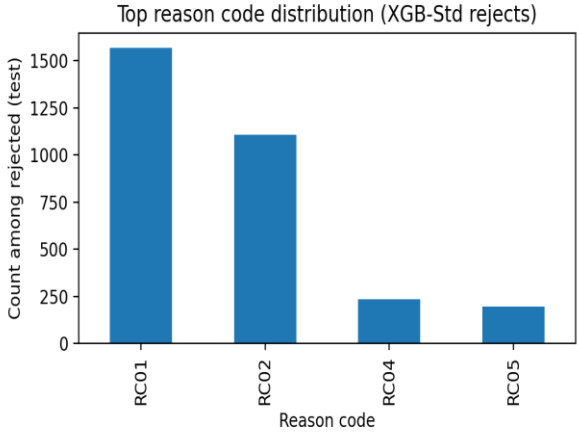

Fig. 7. Distribution of principal adverse-action reason codes among rejected applicants (XGB-Std under base policy).

Table 17. Principal adverse-action reason codes among rejected applicants (XGB-Std; test set).

ReasonCode	Count	Share
RC01	1570	0.5055
RC02	1107	0.3564
RC04	233	0.0750
RC05	196	0.0631

Reason codes: We output a principal reason code for every rejection (coverage 100%). RC01 (repayment status) and RC02 (bill statement amounts) dominate. This aligns with credit scoring literature highlighting delinquency and balance behavior as key risk drivers [2], [3]. Because codes are mapped only to scored factors, the outputs satisfy the requirement that adverse-action reasons relate to factors used in the scoring system [21].

Taken together, the results show that profit-maximizing cutoff selection is the primary lever for economic performance, converting negative-profit baselines into positive-profit portfolios. Model choice and cost-sensitive training provide additional improvements, and Policy-LLM enables these improvements to be tied directly to policy specifications.

Business interpretation: Under the base policy and dataset distribution, profitable lending is achieved by approving roughly half of applicants and keeping bad rate near 8%. This operating point is far more conservative than a 50% cutoff and reflects the heavy-tailed loss structure induced by exposure and LGD. The cutoff–profit–bad-rate surfaces in Figure 5 provide a practical tool for policy owners: they can visualize how tightening the bad-rate target would trade off approval volume and profit, and they can select an operating point consistent with portfolio goals.

Constraint satisfaction: The base policy's minimum approval constraint (40%) is active for several models; without it, the profit-maximizing cutoff can select a smaller approval set. Conversely, the maximum bad-rate constraint (20%) is not binding for the best models at their selected cutoffs because they operate well below this risk cap. In stricter policies (e.g., Conservative scenario), the bad-rate constraint becomes more influential and shifts the selected cutoff downward. These observations illustrate why explicitly encoding constraints, rather than relying on informal manual tuning, is valuable: constraints can be binding in some regimes and not in others.

MLP loss comparison: Among neural variants, MLP-WBCE achieves the highest mean profit (3,527 NT$), followed by MLP-BCE (3,414 NT$), MLP-Focal (3,276 NT$), and MLP-Profit (3,252 NT$). Weighted BCE improves profitability by aligning gradient magnitudes with economic stakes. Focal loss reduces the influence of easy examples and can improve minority-class detection in some settings [9], but in this dataset it slightly reduces profit because it shifts the score distribution toward lower approvals. The profit surrogate learns a profit-oriented decision boundary during training, but it relies on a smooth acceptance approximation; the temperature τ and the joint learning of a cutoff introduce additional optimization difficulty that can reduce empirical profit relative to the post-hoc cutoff optimizer.

Relationship to EMP and cost-sensitive theory: The empirical optimizer used here can be viewed as a sample-based approximation to expected maximum profit. EMP-style measures [6] derive an optimal cutoff under a parametric profit model and assumptions about the distribution of scores. Our approach instead evaluates realized profit across a grid of cutoffs on held-out data, which avoids distributional assumptions and naturally incorporates portfolio constraints. From a cost-sensitive perspective [5], cutoff optimization is the decision-time counterpart to cost-sensitive training: even when training is standard, applying the correct cost-aware cutoff can recover much of the economic value. This is clearly visible in Table 9, where XGB-Std switches from negative to positive profit solely by changing the cutoff from 0.50 to 0.14.

Monitoring implications: Because profit depends on realized outcomes and policy assumptions, monitoring should track not only standard ML metrics (AUC, log loss) but also business metrics (approval, bad rate, and realized profit). The cutoff–profit–bad-rate surface provides a diagnostic tool: if observed bad rate increases, a portfolio owner can inspect how much cutoff adjustment would be required to return to the target region under the same θ. Similarly, if APR or funding rates change, the optimizer can recompute the cutoff under updated θ, and the expected direction of change (more conservative when losses rise, more permissive when income rises) can be validated against scenario tables.

Governance and auditability: A practical advantage of the proposed pipeline is traceability. The cutoff t* is a deterministic function of (i) the calibrated PD scores on the tuning set, (ii) the profit function parameters θ extracted from policy text, and (iii) the explicit constraints. This makes the decision rule auditable: a reviewer can reproduce the cutoff by rerunning the optimizer on the stored tuning outcomes and the stored policy text. In contrast, manual threshold tuning often leaves incomplete artifacts about why a specific cutoff was chosen. When policy terms change, the new cutoff can be computed and logged together with the policy text version, enabling model governance workflows.

Practical extensions: Although we evaluate a single binary approve/reject action, the same framework extends to multi-action decisions such as credit limit assignment, pricing tiers, and collections intensity. In those cases, Policy-LLM would extract additional policy parameters (e.g., tier-specific APRs, fixed and variable servicing costs, or recovery curves), and the decision optimizer would select among multiple actions by maximizing expected profit under constraints. Similarly, constraints can include fairness or regulatory thresholds expressed at subgroup level, provided they can be evaluated on held-out data. The key idea remains unchanged: treat policy as a first-class input, translate it into a machine-readable utility and constraint set, and optimize the decision rule explicitly for the intended business objective.

Limitations

Profit modeling simplifications: The profit function uses LIMIT_BAL as exposure and a constant utilization ratio for all applicants. Real portfolios exhibit heterogeneous utilization, time-varying balances, and multi-period cash flows. The chosen one-period formulation is sufficient for comparing decision policies on this dataset but does not model dynamic credit line management.

Dataset scope: The UCI dataset is a widely used benchmark [1] but represents a single geographic and temporal setting. Performance and optimal cutoffs can differ in other markets, products, and economic conditions.

Policy-LLM generality: Policy-LLM is trained on synthetic policies that emulate common underwriting templates. While it achieves perfect extraction on the synthetic test set, applying it to fully unconstrained real-world policy prose would require additional domain data and potentially larger models.

Compliance evaluation: We generate principal reason codes using a linear surrogate explanation model. This approach enforces a fixed mapping from scored factors to codes, but it does not substitute for institution-specific legal review of adverse-action notice practices.

Conclusion

We presented a profit-maximizing, cost-sensitive credit scoring framework that integrates policy text into decision optimization. A Policy-LLM module extracts profit parameters, decision constraints, and reason-code mappings from policy terms, enabling a coherent utility model and compliance logic to be coupled to PD models. On the UCI credit card default dataset, tree ensembles delivered the highest profit under the base policy, with XGBoost achieving 4,046 NT$ mean profit per applicant at 48.2% approval and 8.1% bad rate. Cost-sensitive objectives improved profitability for logistic regression and MLPs, and the differentiable profit surrogate produced competitive performance while learning a profit-oriented cutoff. The results show that translating policy into an explicit profit function and optimizing decisions accordingly is a practical way to align model training and deployment with business objectives and constraints.

References

[1] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," Expert Systems with Applications, vol. 36, no. 2, pp. 2473–2480, 2009.

[2] D. J. Hand and W. E. Henley, "Statistical classification methods in consumer credit scoring: a review," Journal of the Royal Statistical Society: Series A, vol. 160, no. 3, pp. 523–541, 1997.

[3] L. C. Thomas, "A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers," International Journal of Forecasting, vol. 16, no. 2, pp. 149–172, 2000.

[4] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," Journal of the Operational Research Society, vol. 54, no. 6, pp. 627–635, 2003.

[5] C. Elkan, "The foundations of cost-sensitive learning," in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2001, pp. 973–978.

[6] T. Verbraken, C. Bravo, R. Weber, and B. Baesens, "Development and application of consumer credit scoring models using profit-based classification measures," European Journal of Operational Research, vol. 238, no. 2, pp. 505–513, 2014.

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the ACM SIGKDD

International Conference on Knowledge Discovery and Data Mining (KDD), 2016, pp. 785–794.

[8] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 3146–3154.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2999–3007.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proceedings of the International Conference on Learning Representations (ICLR), 2015.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533–536, 1986.

[13] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in Proceedings of EMNLP, 2014, pp. 1724–1734.

[14] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv:1412.3555, 2014.

[15] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. MIT Press, 1999, pp. 61–74.

[16] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in Proceedings of KDD, 2002, pp. 694–699.

[17] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in Proceedings of ICML, 2005, pp. 625–632.

[18] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[19] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems (NeurIPS), 2019, pp. 8024–8035.

[20] Basel Committee on Banking Supervision, "International Convergence of Capital Measurement and Capital Standards: A Revised Framework (Comprehensive Version)," Bank for International Settlements, Jun. 2004.

[21] Board of Governors of the Federal Reserve System, "Consumer Compliance Handbook: Regulation B (Equal Credit Opportunity), Notifications—Section 202.9," Jan. 2006.

[22] Consumer Financial Protection Bureau, "Consumer Financial Protection Circular 2022-03: Adverse action notification requirements in connection with credit decisions based on complex algorithms," May 26, 2022.

[23] D. Dua and C. Graff, "UCI Machine Learning Repository," University of California, Irvine, 2019.

[24] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, "Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research," European Journal of Operational Research, vol. 247, no. 1, pp. 124–136, 2015.

[25] A. C. Bahnsen, D. Aouada, and B. Ottersten, "Example-dependent cost-sensitive decision trees," Expert Systems with Applications, vol. 42, no. 19, pp. 6609–6619, 2015.

[26] H. W. Kuhn, "The Hungarian method for the assignment problem," Naval Research Logistics Quarterly, vol. 2, no. 1–2, pp. 83–97, 1955.

[27] Xinzhuo Sun, Yifei Lu, and Jing Chen, “Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting”, JACS, vol. 3, no. 8, pp. 9–24, Aug. 2023, doi: 10.69987/JACS.2023.30802.

[28] Xinzhuo Sun, Jing Chen, Binghua Zhou, and Meng-Ju Kuo, “ConRAG: Contradiction-Aware Retrieval-Augmented Generation under Multi-Source Conflicting Evidence”, JACS, vol. 4, no. 7, pp. 50–64, Jul. 2024, doi: 10.69987/JACS.2024.40705.

[29] K. Xu, H. Zhou, H. Zheng, M. Zhu, and Q. Xin, “Intelligent classification and personalized recommendation of e-commerce products based on machine learning,” Proceedings of the 6th International Conference on Computing and Data Science (ICCDS), 2024.

[30] Hanqi Zhang, “DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models”, JACS, vol. 3, no. 7, pp. 24–40, Jul. 2023, doi: 10.69987/JACS.2023.30703.

[31] Hanqi Zhang, “Risk-Aware Budget-Constrained Auto-Bidding under First-Price RTB: A Distributional

Constrained Deep Reinforcement Learning Framework", JACS, vol. 4, no. 6, pp. 30–47, Jun. 2024, doi: 10.69987/JACS.2024.40603.

[32] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, "IoT traffic classification and anomaly detection method based on deep autoencoders," Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024), 2024.

[33] 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.

[34] T. Shirakawa, Y. Li, Y. Wu, S. Qiu, Y. Li, M. Zhao, H. Iso, and M. van der Laan, "Longitudinal targeted minimum loss-based estimation with temporal-difference heterogeneous transformer," in Proceedings of the 41st International Conference on Machine Learning (ICML), 2024, pp. 45097–45113, Art. no. 1836.

[35] B. Wang, Y. He, Z. Shui, Q. Xin, and H. Lei, "Predictive optimization of DDoS attack mitigation in distributed systems using machine learning," Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024), 2024, pp. 89–94.

[36] Z. Zhong, M. Zheng, H. Mai, J. Zhao, and X. Liu, "Cancer image classification based on DenseNet model," Journal of Physics: Conference Series, vol. 1651, no. 1, p. 012143, 2020.

[37] Z. S. Zhong and S. Ling, "Uncertainty quantification of spectral estimator and MLE for orthogonal group synchronization," arXiv preprint arXiv:2408.05944, 2024.

[38] Z. Ling, Q. Xin, Y. Lin, G. Su, and Z. Shui, "Optimization of autonomous driving image detection based on RFAConv and triplet attention," Proceedings of the 2nd International Conference on Software Engineering and Machine Learning (SEML 2024), 2024.

[39] Z. S. Zhong and S. Ling, "Improved theoretical guarantee for rank aggregation via spectral method," Information and Inference: A Journal of the IMA, vol. 13, no. 3, 2024.

[40] J. Chen, J. Xiong, Y. Wang, Q. Xin, and H. Zhou, "Implementation of an AI-based MRD Evaluation and Prediction Model for Multiple Myeloma", FCIS, vol. 6, no. 3, pp. 127–131, Jan. 2024, doi: 10.54097/zJ4MnbWW.