LightMIRM: Light Meta-learned Invariant Risk Minimization for Trustworthy Loan Default Prediction

Meng Jiang†
University of Science and Technology of China
Hefei, China
jiangm@mail.ustc.edu.cn

Yang Zhang
University of Science and Technology of China
Hefei, China
zy2015@mail.ustc.edu.cn

Yuan Gao
University of Science and Technology of China
Hefei, China
yuanga@mail.ustc.edu.cn

Yansong Wang
Chery FS
Wuhu, China
wangyansong@cheryfs.cn

Fuli Feng
University of Science and Technology of China
Hefei, China
fulifeng93@gmail.com

Xiangnan He
University of Science and Technology of China
Hefei, China
xiangnanhe@gmail.com

***Abstract*—Machine learning models are increasingly applied to loan default prediction to reduce the labor cost of financial institutions and the waiting time of lenders. We find that existing loan default prediction models remain lack minimax fairness, *i.e.,* encountering significant performance drops on underrepresented subpopulations. The main cause of this trustworthy issue is pursuing Empirical Risk Minimization over the whole population, which will overlook the underrepresented subpopulations. To tackle this issue, we split the training data into subpopulations (*a.k.a.* environments) and conduct Invariant Risk Minimization (IRM) to learn the optimal prediction model across environments. A technical challenge is the computation cost of directly using existing IRM methods suitable for loan default prediction, such as meta-IRM, which quadratically increases as the number of environments. To reduce the complexity in training, we propose a light meta-IRM method which reduces time complexity to be linear through environment sampling and loss replaying strategies. We apply the light meta-IRM to train a representative loan default prediction model and conduct both online and offline evaluations on a large auto loan platform. Extensive experiment results validate the advantage of the proposed light meta-IRM *w.r.t.* the overall accuracy, minimax fairness, and training cost.**

***Index Terms*—Loan default prediction, Invariant risk minimization, Fairness, Trustworthy**

I. Introduction

Loan default prediction [1, 2] plays an important role in the financial system, which predicts loan defaults for financial institutions and the banking industry. In the current financial system, human approvers are overwhelmed by the massive loan applications, lengthening the average waiting time [1]. To accelerate the reviewing procedure, machine learning techniques [3–6] are increasingly adopted to share the workload, which predict loan defaults from the profile of the lender such as occupation, income, and credit records. Explainable machine learning models such as GBDT [4] and Logistic Regression [3] are the standard choices for loan default prediction due to the requirement of trustworthiness in practice as the increase of relevant regulations on financial algorithms launched by different countries [7].

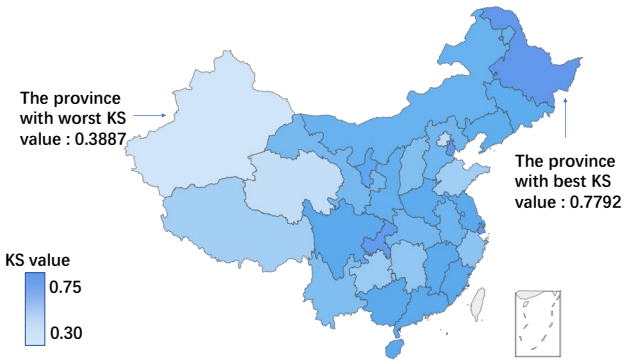


Fig. 1: The province-wise performance of a loan default prediction model trained by the ERM method. Darker color indicates better performance. "KS" denotes the Kolmogorov–Smirnov statistic.

In this work, we reveal that existing methods still face trustworthy issues due to lacking the minimax fairness [8] among different subpopulations, *i.e.,* the model performance on the underrepresented subpopulations are largely worse than the others. Figure 1 presents evidence on the data from one of the largest auto loan platforms in China[1], where the prediction model is trained over the aggregated data

† Work done at Chery FS. This work is supported by the Chery HuiYin Motor Finance Service Company Ltd..

[1]http://www.cheryfs.cn/english/.

from different provinces. As shown in the figure, the model performance changes significantly across different provinces, *e.g.,* the model performs 39.05% worse on the applications from Xinjiang province than Heilongjiang province, which means applications from such underrepresented provinces have a much higher risk to be faultily rejected. To avoid such a trustworthy issue, it is essential to further pursue the minimax fairness in the loan default prediction task.

Toward the goal, we apply the Invariant Risk Minimization (IRM) [9] to the learning of loan default prediction models instead of pursuing the Empirical Risk Minimization (ERM) as the existing methods. Under the IRM paradigm, training data is split into different environments (*e.g.,* provinces) that represent different subpopulations. IRM pursues the optimal loan default prediction model in all environments. Undoubtedly, IRM will enhance the performance of the underrepresented subpopulations and facilitate the minimax fairness across environments compared to the ERM. Moreover, IRM will capture the invariant relationship between lender features and the occurrence of defaults across different environments, *i.e.,* pushing the model to focus on features which causally affect the final status of a loan [10]. As such, the model will get rid of the notorious spurious correlation and generalize better.

Meta-IRM [11] is an effective implementation of IRM[2] with broader applicability[3] than other implementations such as IRMv1 [9], which seems to be a promising choice for learning trustworthy loan default prediction models. However, the direct application of meta-IRM faces high overhead due to the quadratic increase of computation and memory costs *w.r.t.* the number of environments, which can be large in the loan default prediction task. For instance, the cost of training a model with meta-IRM over M environments is roughly M times higher than ERM. Moreover, loan default prediction models have to be updated periodically at a relatively high frequency. Therefore, it is essential to improve the efficiency of meta-IRM for usage in loan default prediction.

To tackle this challenge, we propose a light meta-IRM (LightMIRM) method for efficiently training a fair loan default prediction model. The key to accelerating meta-IRM lies in avoiding the calculation of meta-losses from all environments in every iteration. In this light, we devise two techniques to equip meta-IRM: **1) environment sampling.** We sample some environments (K) instead of computing meta-losses from all environments (M) to reduce the number of calculations by $\frac{K}{M}$; and **2) meta-loss replaying.** We recycle the history losses from previous iterations to further reduce the number of calculations by $\frac{1}{K}$. We apply the proposed LightMIRM on a representative loan default prediction model (*cf.* Figure 2) and conduct both online and offline evaluations on the Chery FS auto loan platform, where LightMIRM achieves better prediction accuracy and minimax fairness than ERM with comparable training cost.

In summary, the contributions are as follows:

- We reveal the unfairness issue of existing loan default prediction methods and propose to optimize the minimax fairness through meta-IRM.
- We propose LightMIRM with environment sampling and meta-loss replaying, which reduces the computation cost of meta-IRM to be comparable with ERM.
- We conduct online and offline experiments on a large auto loan platform, validating the effectiveness and efficiency of the proposed method.

II. Related works

In this section, we will introduce some related works in loan default prediction and fairness models for trustworthiness.

A. Loan Default Prediction

Loan default prediction is quite important in the financial industry. When the platforms invest to the borrowers, they take the risk of loan default which could lead to the loss of their entire investment. Therefore, it is important for lending platforms to accurately predict the probabilities of loan default [12]. Nazeeh [13] proposes to use Random Forest Trees to predict default samples, which is demonstrated to outperform logistic regression [3], Support Vector Machine (SVM) [14], and some other machine learning algorithms. Zhu *et al.* further equip the Random Forest algorithm with SMOTE [15] to deal with the imbalanced nature of the dataset. Xu *et al.* [16] apply four machine learning methods including random forest, extreme gradient boosting tree, gradient boosting model, and neural networks to figure out implicit factors affecting repayment failure. Li [17] adopts back propagation neural network (BPNN) to simulate the loan default assessment process.

Above mentioned methods have achieved decent performance, but tend to fall short in trustworthiness. They blindly capture correlations between features and labels in training data, hence ignoring some specific subpopulations. In the industrial scenario, where have strict demands for trustworthiness, they are infeasible solutions. Next, we will elaborate more on trustworthy machine learning.

B. Trustworthy machine learning

In recent years, more and more research has been conducted on technologies for trustworthy machine learning. They aim to acquire AI-based services which have fairness, human rights, privacy, and other properties together [18]. Toreini *et al.* [19] summarize four key properties of trustworthy machine learning systems: Fairness, Explainability, Auditability, and Security (FEAS). When building loan default prediction systems, the Explainability and the Auditability are determined by the characteristics of the methods, *e.g.,* deep learning methods are known to be black-box which may have less explainability, while logistic regression and tree-based methods can provide reasons for their predictions. From this perspective, to achieve trustworthy loan default prediction, we use "LightGBM + Logistic Regression" to ensure that our method is explainable and auditable. Moreover, the unfairness for different subpopulations (Fairness) should be addressed properly.

[2]IRM corresponds to a hard bi-level optimization problem and meta-IRM could more flexibly and effectively solve the problem.

[3]Do not assume the linearity of the prediction model.

The measurement of fairness falls under several categories [20–22], while in our paper, we mainly focus on **Calibration** [23, 24] which means that false positive rates across groups should be similar. To ensure this condition, researchers focus on the model performance in the worst cases. Sagawa *et al.* [25] train models to minimize the worst-case loss over groups in the training data to avoid learning models that rely on spurious correlations and suffer a high loss on some groups of data. Krueger *et al.* [26] minimize the average as well as the variance of empirical risks on different groups to make the model fair for different groups. However, Geoff *et al.* [24] point out that the goals of low error and calibration are less likely to be satisfied simultaneously for a model. As an alternative, Johnson *et al.* [27] use a multi-calibration approach to achieve approximate calibration with a high probability. Bae *et al.* [11] propose meta-IRM which calculates the standard deviation of meta-losses to learn invariant features existing across different environments. Inspired by this idea, we also use the variance across different provinces. However, such a stream of methods requires a large time in data sampling which leads to low model efficiency. To fit in the industry scenario better, a more time-friendly approach is of great necessity.

III. Method

In this section, we first briefly introduce the preliminary knowledge of this work. We then present the overview of the proposed method for the loan default prediction task. Next, we elaborate on each module of the model. Lastly, we give some discussions about the time complexity and our designs.

A. Preliminaries

We first give the problem formulation and then briefly introduce the invariant risk minimization.

1) Problem Formulation: Loan default prediction aims at predicting loan default, *i.e.,* the probability that a customer will fail to repay the loan. This work studies the task in a practical scenario where the data is collected from different environments (*e.g.,* provinces) that represent different sub-populations. The environments are naturally heterogeneous for various reasons, *e.g.,* the differences in economies and cultures. We aim to learn a model that can perform well and fairly across these heterogeneous environments, improving the trust to use the model[4]. Especially, we emphasize that the model performance in the underrepresented environments (*e.g.,* Xinjiang province in Figure 1) should not drop largely compared to others, *i.e.,* pursuing minimax fairness.

Let $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_M\}$ denote the data, where $\mathcal{D}_m$ denotes the data collected from the m-th environment, and M denotes the number of environments. And we denote each sample in $\mathcal{D}_m$ as (x, y), where $y \in \{0, 1\}$ denotes whether or not a user fails to repay the loan, and x denotes the raw features of the instance, including basic applicant information (*e.g.,* age), information from banks (*e.g.,* the count of defaults), and other information (*e.g.,* car). Let $\mathcal{X}$ denote the space of the raw feature, *i.e.,* $x \in \mathcal{X}$, similarly $y \in \mathcal{Y}$. We aim to learn a predictor f: $\mathcal{X} \rightarrow \mathcal{Y}$ based on $\mathcal{D}$, which could generate accurate and fair predictions among different environments.

2) Invariant Risk Minimization: Recently, *Invariant Learning* is proposed to pursue optimal performances over all possible environments [9, 28, 29]. Invariant risk minimization (IRM) is the most representative method, which exploits correlations invariant across all training environments to learn an invariant predictor $\Phi \circ w$ as follows:

$$\begin{aligned} &\min_{\Phi, w} \sum_{m=1}^{M} R^m(\Phi \circ w) \\ &s.t.\ w \in \arg\min_{\bar{w}} R^m(\Phi \circ \bar{w}),\ for\ all\ m, \end{aligned} \tag{1}$$

where $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ is a representation encoder that transfers the raw feature x in the raw space $\mathcal{X}$ to a representation space $\mathcal{H}$; and $w : \mathcal{H} \rightarrow \mathcal{Y}$ is a classifier which generates predictions based on the representation generated by Φ. Φ and w are contacted together to form the overall predictor $\Phi \circ w$. This equation indicates IRM tries to find a representation encoder Φ such that the optimal w on top of the representation matches for all environments, forming an invariant predictor $\Phi \circ w$. Meanwhile, when utilizing patterns in one environment data, IRM takes into account whether the patterns can enable better overall performance across training environments, *i.e.,* whether the patterns are shared across environments, to capture invariant correlations. That is significantly different from ERM.

IRM is obviously a bi-level optimization problem that is hard to be directly solved. A more practical version of IRM is IRMv1 [9], which approximately converts the constraints in Equation (1) into an invariance penalty term by assuming w is a fixed linear module. However, IRMv1 is just an approximation for IRM and fails to capture invariant correlations in many cases [30, 31]. To overcome the issue, Bae et al. [11] propose to directly solve the IRM optimization problem to find an invariant predictor $f = \Phi \circ w$ with the meta-learning method MAML [32, 33], which is widely used to solve bi-level optimization problems.

B. Model Overview

Figure 2 illustrates the overall framework of our loan default prediction model. The framework generally follows a "GBDT+LR" architecture proposed by [34] with several considerations: 1) the architecture is powerful but lightweight, making it easy to be deployed [34, 35]; 2) a model built in such an architecture is explainable compared to other machine learning methods, which is critical for practical deployment; 3) such a framework could achieve automatic feature engineering, reducing human costs. As shown in Figure 2, the framework is comprised of two key modules:

- **Feature extraction module** (the blue box of Figure 2), which automatically selects and develops more effective features from raw features;
- **Logistic regression (LR) module** (the yellow box of Figure 2), which is the core prediction module and learned with IRM.

[4]Both on the applicant and the platform side.

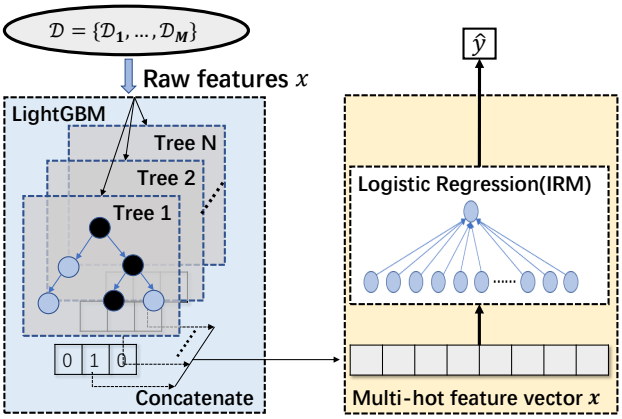


Fig. 2: Model overview. The blue box is the feature extraction module, and the yellow box is the logistic regression module.

Although our framework follows the "GBDT+LR" architecture, there are also significant differences. Compared with normal "GBDT+LR" methods, we take the invariant risk minimization equipped with a fast meta-learning learning algorithm to learn the LR model, instead of taking the ERM learning paradigm. And we aim at achieving generalizable and fair loan default prediction among different environments. Next, we present the details of the two parts of our framework, especially focusing on how to learn the LR model with IRM.

C. Feature Extraction

Feature extraction (also known as feature engineering), which aims to select effective features and develop new features from raw features, is vitally important for improving the performance of machine learning-based industrial systems [36]. Feature engineering usually relies on experts' knowledge, which could be expensive. To avoid paying huge human efforts to feature extraction, we take the gradient boost decision tree (GBDT) to achieve automatic feature extraction for loan default prediction in a way similar to [34]. In short, we first train a LightGBM[5] model to predict the loan default by optimizing the cross-entropy loss on $\mathcal{D}$ [34]. Then we treat the trained decision trees of the model as a tuple of non-linear transformations to select and combine features.

Specially, as shown in the blue box of Figure 2, for each instance, we input its raw features into the well-trained LightGBM. Then, each tree in LightGBM is seen as a transformation function and will generate a new category of cross-feature, *i.e.,* combining feature, according to the inputted raw features. The value of the generated categorical feature is the index of the leaf that the inputted instance falls in, and the value will be further one-hot encoded. For example, assuming the first tree has three leaves and an instance falls in the second leaf of the tree, the one-hot encoded value of the categorical feature generated by the first tree is $[0, 1, 0]$. Lastly, we concatenate the categorical features generated by different trees, formulating a multi-hot feature vector (denoted as $\boldsymbol{x} \in \mathcal{R}^N$, where N is the dimension of the vector) for the instance (with raw feature x).

[5]LightGBM is an effective and efficient GBDT-based algorithm.

D. Invariant Risk Minimization-based Logistic Regression

There are two demands for a predictor in our loan default prediction task: 1) it should generate fair predictions for different environments, *i.e.,* not show favoritism or prejudice towards the applicants (users) in certain environments; 2) it should achieve good overall performance over different environments. However, building a predictor with the ERM learning paradigm is easily dominated by some environments and suffers from spurious correlation issues, resulting in poor and unfair cross-environment performance. For example, a predictor learned with ERM possibly provides worse predictions for the environments with fewer data, and blindly predicts higher scores for the applicants belonging to the environments with higher loan default rates. To address this dilemma, we propose to learn the LR predictor with invariant risk minimization (IRM) to satisfy the two demands. IRM optimizes both the specific performance on each environment and the overall performance over environments, capturing the invariant correlations across different environments. That means IRM does not bias toward some special environments with the overall performance kept. Therefore IRM is suitable for achieving good and fair performance across environments. We next present how to learn the LR model with IRM from two aspects: learning objective and learning algorithm.

1) Learning Objective: We take f to represent the LR model. For instance $(x, y) \in \mathcal{D}$, f generates the prediction as follows:

$$\hat{y} = f(x; \boldsymbol{\theta}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}, \tag{2}$$

where $\boldsymbol{x} \in \mathcal{R}^N$ represents the one-hot vector gotten by the feature extraction module for the instance and $\boldsymbol{\theta} \in \mathcal{R}^N$ denotes the model parameters of the LR model.

To pursue fair and good overall performances over different environments, we apply the IRM objective in Equation (1) to learn the LR model. Formally,

$$\min_{\boldsymbol{\theta}} \quad \sum_{m} R^m(\mathcal{D}_m; \boldsymbol{\theta}) \tag{3a}$$

$$s.t. \quad \boldsymbol{\theta} \in \arg\min_{\bar{\boldsymbol{\theta}}} R^m(\mathcal{D}_m; \bar{\boldsymbol{\theta}}), \ for\ all\ m, \tag{3b}$$

where m denotes the m-th environment, $R^m(\mathcal{D}_m; \boldsymbol{\theta})$ denotes the loss on the data $\mathcal{D}_m$ and is computed with the binary cross entropy function. Formally,

$$R^m(\mathcal{D}_m; \boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_m|} \sum_{(x,y)\in\mathcal{D}_m} -y \log(f(x; \boldsymbol{\theta})) - (1-y)\log(1 - f(x; \boldsymbol{\theta})), \tag{4}$$

where $|\mathcal{D}_m|$ denotes the size of $\mathcal{D}_m$ and $f(x; \boldsymbol{\theta})$ is the prediction of LR model as shown in Equation (2).

Remark. The original IRM in Equation (1) splits a predictor into two parts — w and Φ, and learns an invariant predictor by finding a suitable w and Φ, respectively. Differently, we try to directly find a whole predictor, which could lead to good

Algorithm 1: The learning algorithm of Meta-IRM

Require: $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_M\}$; Inner/outer loop learning rate α/β; hyper-parameter λ.

1: **Initialize:** Randomly initialize model parameters $\boldsymbol{\theta}$
2: // outer loop
3: **while** Stop condition is not reached **do**
4: // inner loop
5: **for** all $m \in \{1, \ldots, M\}$ **do**
6: Compute $R^m(\mathcal{D}_m; \boldsymbol{\theta})$.
7: Compute $\bar{\boldsymbol{\theta}}_m = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} R^m(D_m; \boldsymbol{\theta})$.
8: Compute $R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{m' \neq m} R^{m'}(D_{m'}; \bar{\boldsymbol{\theta}}_m)$.
9: **end for**
10: Compute the standard deviation σ of $\{R_{meta}(\bar{\boldsymbol{\theta}}_m)\}_{m=1}^M$.
11: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} (\sum_m R_{meta}(\bar{\boldsymbol{\theta}}_m) + \lambda \sigma)$
12: **end while**

Algorithm 2: LightMIRM

Require: $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_M\}$; Inner/outer loop learning rate α/β; hyper-parameter λ

1: **Initialize:** Randomly initialize model parameters $\boldsymbol{\theta}$; initialize the elements of $\boldsymbol{H}_m$ as zeros for each m.
2: // outer loop
3: **while** Stop condition is not reached **do**
4: // inner loop
5: **for** all $m \in \{1, \ldots, M\}$ **do**
6: Compute $R^m(\mathcal{D}_m; \boldsymbol{\theta})$.
7: Compute $\bar{\boldsymbol{\theta}}_m = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} R^m(D_m; \boldsymbol{\theta})$.
8: Randomly sample an environment s_m ($\neq m$).
9: Compute $R^{s_m}(\mathcal{D}_{s_m}; \bar{\boldsymbol{\theta}}_m)$ and put it into $\boldsymbol{H}_m$.
10: Compute $R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{i=1}^{L} (\gamma)^{L-i} \boldsymbol{H}_m^i$.
11: **end for**
12: Compute the standard deviation σ of $\{R_{meta}(\bar{\boldsymbol{\theta}}_m)\}_{m=1}^M$.
13: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} (\sum_m R_{meta}(\bar{\boldsymbol{\theta}}_m) + \lambda \sigma)$
14: **end while**

overall performances across environments (Equation (3a)) and match for all environments (Equation (3b)), to capture invariant correlations. The main consideration is that the LR model is very shallow, and thus is not easy to split. Meanwhile, such a manner has been taken by previous work, and its effectiveness in capturing invariant correlations has been verified [11].

2) Meta-based Learning Algorithm: Our IRM learning objective in Equation (3) is a hard bi-level optimization problem. Instead of directly solving it, we follow previous work [11] to solve it with a famous meta-learning method MAML [32]. In short, we convert the challenging bi-level optimization of IRM into the two-level optimization scheme of the meta-learning framework by constructing a task for each environment. Algorithm 1 shows the MAML-based learning process. In the algorithm, we optimize the objective in Equation (3b) in the inner loop of MAML (lines 6-7), making the predictor perform well in a certain environment; and we optimize the objective in Equation (3b) in the outer loop of MAML (lines 10-11), making the predictor perform well across environments with the consideration of satisfying constraints in Equation (3b). We next describe the main steps in the inner loop and outer loop, respectively.

Inner loop: As shown in Equation (3b), we need to find optimal predictors for different environments. Therefore, for each environment m, we update the model parameters $\boldsymbol{\theta}$ of the LR predictor to minimize the loss $R^m(D_m; \boldsymbol{\theta})$, getting a temporary $\bar{\boldsymbol{\theta}}_m$ for the environment. Formally,

$$\bar{\boldsymbol{\theta}}_m = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} R^m(D_m; \boldsymbol{\theta}), \tag{5}$$

where α denotes the learning rate in the inner loop.

Outer loop: In the outer loop, we need to make the predictor perform well across all training environments (Equation (3b)) and satisfy the constraints in Equation (3b). Towards this goal, we update the model parameters $\boldsymbol{\theta}$ based on $\{\bar{\theta}_m\}_{m=1}^M$, as follows:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \left(\sum_m R_{meta}(\bar{\boldsymbol{\theta}}_m) + \lambda \sigma \right), \tag{6}$$

where $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ denotes the overall loss without considering the m-th environment, taking $\bar{\boldsymbol{\theta}}_m$ as the model parameters of the predictor, *i.e.,*

$$R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{m' \neq m} R^{m'}(D_{m'}; \bar{\boldsymbol{\theta}}_m),$$

and $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ is termed meta-loss; and $\sum_m R_{meta}(\bar{\boldsymbol{\theta}}_m)$ indicates that we finally update the predictor with the consideration of all $\{\bar{\boldsymbol{\theta}}_m\}_{m=1}^M$, *i.e.,* considering all the constraints in Equation (3b); β is the learning rate in the outer loop; λ is a hyper-parameter to control the strength of σ, and σ is an auxiliary loss proposed in [11], which is computed as follows:

$$\sigma = \sqrt{\frac{1}{M} \sum_m \left(R_{meta}(\bar{\boldsymbol{\theta}}_m) - \frac{1}{M} \sum_{m'} R_{meta}(\bar{\boldsymbol{\theta}}_{m'}) \right)^2}. \tag{7}$$

σ is indeed the standard deviation of $\{R_{meta}(\bar{\boldsymbol{\theta}}_m)\}_{m=1}^M$, and it could enable more stable training and faster convergence [11].

The updates in the inner loop and the outer loop are iterated until the stop condition is reached, as shown in line 3 of Algorithm 1. We denote the LR predictor learned with Algorithm 1 as meta-IRM.

E. Light Meta-based Invariant Risk Minimization

Although meta-IRM could easily solve the bi-level optimization problem in Equation (3), it is not directly applicable in the industry due to its high computational cost, which is at least M times than ERM-based methods. As shown in Algorithm 1, we need to compute a $R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{m' \neq m} R^{m'}(D_{m'}; \bar{\boldsymbol{\theta}}_m)$, which is nearly calculated on the

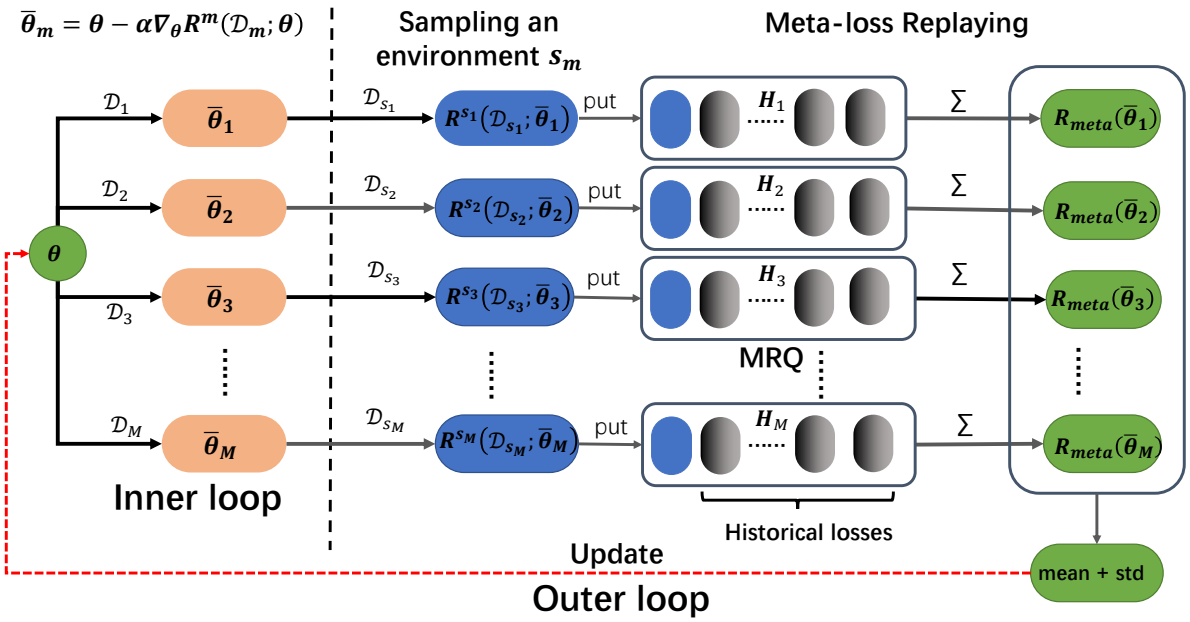


Fig. 3: Illustration of the LightMIRM, which contains two key parts for speeding up the training — environment sampling and meta-loss replaying. "MRQ" denotes our meta-loss replaying queue.

total data $\mathcal{D}$, for each environment m. That means meta-IRM needs to calculate M times the loss on the total data, while the ERM-based method only needs to calculate the loss once. Meanwhile, there is the additional cost of computing the second-order gradients in the outer loop of meta-IRM. Thus, meta-IRM has at least M times the computational cost.

To make the meta-IRM applicable in the industry, we propose a light version of meta-IRM named Light Meta-based Invariant Risk Minimization (LightMIRM), which speeds up the training process via effective and efficient sampling. LightMIRM has a similar learning algorithm to meta-IRM (Algorithm 1) but calculates $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ more rapidly. Especially, LightMIRM contains two main designs — environment sampling and meta-loss replaying— to rapidly calculate $R_{meta}(\bar{\boldsymbol{\theta}}_m)$, as shown in Figure 3.

Environment sampling. A direct way to rapidly compute the meta-loss $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ is sampling partial environments S_m to approximately compute it as

$$R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{m' \in S_m} R^{m'}(D_{m'}; \bar{\boldsymbol{\theta}}_m),$$

instead of utilizing all environments. In this way, although only partial environments are considered in $R_{meta}(\bar{\boldsymbol{\theta}}_m)$, the overall loss of training environments could also be approximately optimized via re-sampling environments at each iteration[6]. Let K denote the size of S_m. We can reduce the computation cost to K/M of meta-IRM. A smaller K obviously means greater acceleration. To minimize the computational cost, we only sample one environment. We denote the sampled environment as s_m $(\neq m)$, and then $S_m = \{s_m\}$.

[6]E.g., different batches when learning the model in a mini-batch manner.

Meta-loss replaying. Although only sampling one environment to calculate the meta-loss $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ could reduce the computation cost to the greatest degree, it could lead to bad performance. This is because a smaller sampling size means a worse approximation to the ideal $R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{m' \neq m} R^{m'}(D_{m'}; \bar{\boldsymbol{\theta}}_m)$ and thus leads to worse performance, as shown in Figure 6. To avoid sacrificing the performance when just sampling an environment, we propose a module named *Meta-loss Replaying*, in which the main component is the *Meta-loss Replaying Queue* (MRQ). MRQ is a fixed-length queue, which could store the losses of the environments sampled in previous iterations. We take the loss of the new sampled environment and the losses stored in the queue to better approximate the ideal $R_{meta}(\bar{\boldsymbol{\theta}}_m)$. Especially, as shown in Figure 3, for each environment m, we construct a MRQ denoted as $\boldsymbol{H}_m$. In each iteration, after getting the $\bar{\boldsymbol{\theta}}_m$ according to Equation (5) and sampling an environment s_m, we compute the loss $R^{s_m}(\mathcal{D}_{s_m}; \bar{\boldsymbol{\theta}}_m)$, and then put it into $\boldsymbol{H}_m$ as follows:

$$\begin{aligned} \boldsymbol{H}_m^i &= \boldsymbol{H}_m^{i+1} \quad 1 \leq i \leq L-1 \\ \boldsymbol{H}_m^L &= R^{s_m}(\mathcal{D}_{s_m}; \bar{\boldsymbol{\theta}}_m), \end{aligned} \tag{8}$$

where $\boldsymbol{H}_m^i$ denotes the i-th element in $\boldsymbol{H}_m$, L denotes the length of the queue. In this way, we move the elements in the queue forward one by one, and then put $R^{s_m}(\mathcal{D}_{s_m}; \bar{\boldsymbol{\theta}}_m)$ into the last position of the queue. Note that $\boldsymbol{H}_m^i$ $(i = 1, \ldots, L-1)$ stores the losses on the environments sampled in the previous iteration, which are computed with the previous $\bar{\boldsymbol{\theta}}_m$ gotten in the corresponding iterations. Next, we approximately compute

$R_{meta}(\bar{\boldsymbol{\theta}}_m)$ according to $\boldsymbol{H}_m$. Formally,

$$R_{meta}(\bar{\boldsymbol{\theta}}_m) = \sum_{i=1}^{L} (\gamma)^{L-i} \boldsymbol{H}_m^i, \tag{9}$$

where γ is a decay coefficient. $(\gamma)^{L-i}$ represents paying more attention to the losses of most recent iterations, considering that these losses are more reliant.

With environment sampling and meta-loss replaying, LightMIRM could achieve rapid calculation of $R_{meta}(\bar{\boldsymbol{\theta}}_m)$, reducing the time-consuming of training. Algorithm 2 summarizes the training process of LightMIRM. Compared with the learning algorithm shown in Algorithm 1, the only differences are that LightMIRM computes approximately the meta-loss $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ with environment sampling and the meta-loss replaying (lines 8 to 10), and the following steps utilize the approximated meta-loss $R_{meta}(\bar{\boldsymbol{\theta}}_m)$ (lines 12-13).

Discussion. How well does the meta-loss computed with the meta-replaying method approximate the true meta-loss? It lies in the differences between the replayed loss and the loss recomputed in each iteration for the same environment. The smaller the difference is, the better the approximation is. We could take a relatively small learning rate (1e-6 in this work) to keep the loss of an environment close between two consecutive iterations [37, 38] and take a length-fixed queue to filter far-distant history losses. The operations could make the differences relatively small, leading to a good approximation of the true meta-loss. Moreover, we pay more attention to the more recent losses memorized in the queue, making the approximated meta-loss more reliant.

F. Time Complexity Analyses

We next analyze the time complexity of meta-IRM and LightMIRM. To simplify, we treat each forward/backward propagation on an environment, e.g., computing $R^m(\mathcal{D}_m; \boldsymbol{\theta})$ and computing $\nabla_{\boldsymbol{\theta}} R^m(D_m; \boldsymbol{\theta})$, as an atomic operation and omit the time cost of others.

Meta-IRM. According to Algorithm 1, the cost of (the operation in) line 6 (line 7) is $O(1)$, and the cost of line 8 is $O(M-1)$. Considering the loop in line 5, the total cost of lines 5 to 9 is $O(M*(1+1+M-1)) = O(M^2+M)$. The cost of line 10 is $O(0)$, and the cost of line 11 is $O(M*(M-1))$. The cost of other lines is 0. Thus, in each iteration, the total cost is $O(M^2 + M + M*(M-1)) = O(2M^2)$.

LightMIRM. According to Algorithm 2, the cost of (the operation in) line 6 (line 7) is $O(1)$, the cost of line 8 is $O(0)$, the cost of line 9 is $O(1)$, and the cost of line 10 is $O(0)$. Considering the loop shown in line 5, the total cost of lines 5 to 10 is $O(M*(1+1+0+1+0)) = O(3M)$. The cost of line 12 is $O(0)$, and the cost of line 13 is $O(M*1)$ (only the last element in the queue has gradients). The cost of other lines is 0. Thus, in each iteration, the total cost is $O(3M + M) = O(4M)$.

IV. Experiment

In this section, we conduct experiments on the historical transaction data of Chery FS with the aim of answering the following research questions:

- **RQ1:** Why do we need trustworthy models?
- **RQ2:** Do our proposed light meta-IRM methods outperform the state-of-the-art methods?
- **RQ3:** How does our proposed method accelerate the training of Meta-IRM?
- **RQ4:** What are the impacts of each part of the proposed model?
- **RQ5:** Why does our method outperform other methods?

A. Experimental Setup

1) Baselines: We compared our proposed Light Meta-IRM methods with the following methods:

ERM [3]. The goal of ERM is to train a linear regression model which can get a minimum loss on the training set. This method assumes that the training set and the test set are independent and identical.

ERM + fine-tuning. In order to fit the differences between various environments, the ERM model is fine-tuned for each province respectively before the evaluation.

Up-sampling. This method adopts an up-sampling strategy in provinces with fewer samples. Note that we could adjust the rate of negative samples in loss function respectively.

Group DRO [25]. The group DRO aims to achieve substantially higher worst-group accuracy by coupling group DRO models with increased regularization.

V-REx [26]. This method not only uses the mean of losses calculated from different environments, but also minimizes the variance of losses which can decrease the discrepancy of performance in various environments.

2) Evaluation Metrics: Methods mentioned above are evaluated by the following metrics:

Area Under Curve(AUC): AUC is a performance measurement based on the ROC curve, which shows the performance of a classification model under different classification thresholds. The ROC curve plots the True Positive Rate (TPR) and False Positive Rate (FPR) at different classification thresholds whose area under the curve is AUC. Moreover, AUC represents the capacity of the model to distinguish different classes, especially in tasks like loan default prediction, where the classes are imbalanced.

Kolmogorov-Smirnov(KS): KS statistic [39] is widely used in binary classification problems. The KS statistic for two samples is simply the largest distance between their two cumulative distribution functions. Similar to AUC, KS statistic represents the discrimination of the model as well. A higher value in KS means that the model has a stronger risk-ranking ability.

However, figuring up the AUC and KS on all of the test sets can not express the performance of the model on a certain environment, which is not in line with our goals. In order to explicitly measure the model performance on different environments, we do not calculate the AUC and KS over all

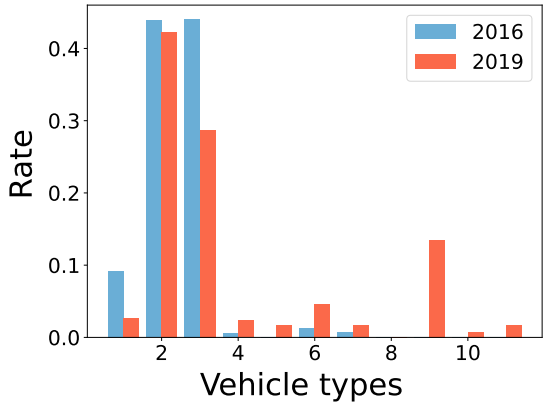


Fig. 4: The distribution of vehicle types in different years. Because it changes from year to year, we ignore the years between 2016 and 2019 for space.

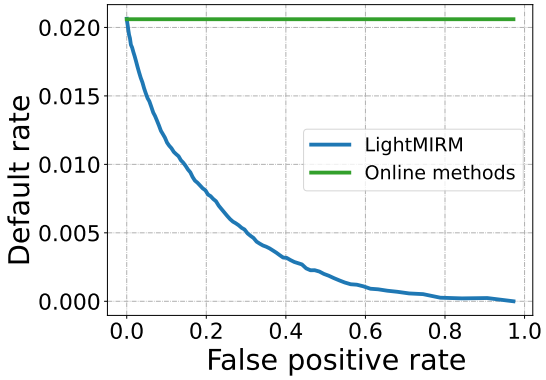


Fig. 5: The false positive rate and the default rate in the online test.

of the test sets. Note that our objective is to perform well on worst cases without harming the overall performance; we focus on two sets of metrics: (1) mean value (*i.e., mKS* for the mean KS value and *mAUC* for the mean AUC) to evaluate the overall performance; (2) the minimum (*i.e., wKS* for the worst KS value and *wAUC* for the worst AUC) to evaluate the fairness of the methods.

B. Data Analysis

We conduct experiments on Chery FS's historical transaction data. The dataset consists of 1.4 million records with 210-dimensional features. We observe that data collected from different provinces tend to have various patterns. Hence to evaluate the fairness of our methods in loan default tasks, we treat different provinces as different environments. To answer **RQ1**, the necessity of considering the trustworthiness issue can be summarized in two folds:

Firstly, the types of customers are different in various areas due to the shift of business centers. For instance, it is more likely to have more customers who buy trailer trucks in the area where trade is more developed. Whereas in economically backward areas, the proportion of customers who buy used cars may be higher. This can be illustrated in Figure 4. This issue will lead to varying patterns of users in different provinces which means the concept shift also exists across environments. So the conventional ERM methods could be unfair to these provinces, and we need more trustworthy models for prediction. Therefore, we will use the minimum metrics in various environments to assess the performance in the worst cases, thus inferring the fairness of the model.

Interestingly, we discover another kind of data drifts in the dataset. We extract data from 2016 to 2019 as the training set and leave 2020 as the test set. We suppose this setting is more in accordance with real-world applications. Specifically, the user percentage of various provinces can be changed as time goes by. For example, the Guangdong province has the highest percentage of users from 2016 to 2019. However, the number of transactions in Guangdong province begins to decline significantly in 2020. In light of this, we suppose that the distribution of provinces has covariate shift, which could be formulated as $P_{tr}(y|x) = P_{test}(y|x) \& \& P_{tr}(x) \neq P_{test}(x)$. Furthermore, there are a number of provinces suffering from COVID-19, which reduces the ability of most borrowers to repay their loans. This is called concept shift which is defined as $P_{tr}(y|x) \neq P_{test}(y|x)$. Therefore, the covariate shift and the concept shift co-occur in the default prediction task and the traditional ERM methods could suffer from this issue which may lead to models becoming unreliable. We will use the mean of metrics in different provinces to evaluate the model's overall performance.

C. Performance Comparison

1) Online Comparison: Firstly, we compare our model with Chery FS's existing online model, our model brings the following three enhancements:

- The model trained by our method is a plug-and-play model that can co-exist with existing online models. Therefore, our model can be adopted without applying any changes to the online architecture.
- Our method could reduce the bad debt rate by 63 percent while the threshold is 0.5. The online model has a 2.09 percent bad debt rate. After appending our model to the existing evaluation system as a companion runner, the bad debt rate could be reduced to 0.73 percent.
- As shown in Figure 5, the curve in the first half of the picture is very steep and becomes flat in the second half of the figure. Therefore, we can reduce the bad debt rate by only refusing a little number of loans. The domain experts could use their operation knowledge to find a trade-off between the two indicators.

2) Main Comparison Results: To answer **RQ2**, we evaluate the performance of mentioned baselines and the proposed method, and the comparison results are reported in Table I.

Firstly, as shown in Table I, our method outperforms ERM on most of the metrics, except for the mean AUC. Specially, we improve 0.0304 (*i.e.,* as much as 7.6%) on the minimum KS value, suggesting that ERM could obtain a high score on

TABLE I: Performance comparison. The best results of all methods are indicated in boldface. *mKS* stands for the mean KS value, while *wKS* stands for the worst KS value.

Methods	*mKS*	*wKS*	*mAUC*	*wAUC*
ERM	0.5784	0.3887	**0.8356**	0.7438
ERM + fine-tuning	0.5767	0.4144	0.8337	0.7483
Up Sampling	0.5781	0.3992	0.8330	0.7468
Group DRO	0.5615	0.3835	0.8253	0.7406
V-REx	0.5762	0.4000	0.8329	0.7471
meta-IRM	0.5781	0.4069	0.8332	0.7460
LightMIRM**(our)**	**0.5794**	**0.4183**	0.8351	**0.7518**

TABLE II: Performance comparison between meta-IRM and LightMIRM, the best results are in boldface.

Methods	*mKS*	*wKS*	*mAUC*	*wAUC*
meta-IRM	0.5781	0.4069	0.8332	0.7460
meta-IRM(20)[a]	0.5762	0.4079	0.8334	0.7335
meta-IRM(10)	0.5728	0.367	0.8335	0.7304
meta-IRM(5)	0.5736	0.3630	0.8342	0.7333
LightMIRM[b]	**0.5794**	**0.4183**	**0.8351**	**0.7518**

[a]The number of sampled provinces. [b]The length of MRQ is 5.

mean metrics (*i.e.,* overall performance) but it is unfair for some provinces. To tackle this issue, the conventional solution is fine-tuning the parameters of each province. However, fine-tuning could highly depend on the data quality and data distribution which is not a stable method. From the empirical observation, although fine-tuning can improve the worst score, it may behave worse in some provinces compared to the original ERM method which leads to lower mean performance. This interesting finding further validates the superiority of the proposed methods since we can perform better than the fine-tuning method on both the worst KS and the worst AUC without hindering the overall performance.

Secondly, we compare our method with more equitable methods which have already been illustrated in Section IV-A1. These methods have quite varying performances on CheryFS's dataset. For example, Up Sampling could get a higher mean performance, but can not perform well on the worst score. On the contrary, the V-REx could perform well on the worst metrics, but poorly on mean performance because of the limitation of variance. Differently, our method optimizes the parameters to align the goal of IRM, which endows our method with the ability to achieve both the highest mean and minimum scores. Specifically, our method achieves 0.0011 improvements over the Up Sampling on the mean KS value. Moreover, on the worst KS value, our method is 0.0191 higher than V-REx. More importantly, we compare our method with meta-IRM. As expected, the meta-IRM can outperform other methods except for our method.

3) Comparing with Meta-IRM: To validate the effectiveness of our design, we perform a detailed comparison between LightMIRM and meta-IRM under different sampling numbers (*i.e.,* complete, S=5, S=10, S=20). The percentage of relative improvement on each metric are reported in Table II. The main

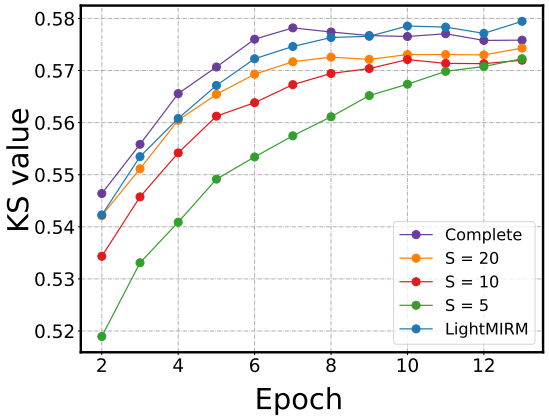


Fig. 6: The result of meta-IRM with different number of samples and LightMIRM where S denotes the number of samples.

TABLE III: Time Cost for Operation Steps

Step	Methods		
	meta-IRM	meta-IRM(5)	LightMIRM
loading data	0.0007s	0.0007s	0.0007s
transforming the format	0.0039s	0.0042	0.0043s
inner optimization	0.0058s	0.0057s	0.0063s
calculating the meta-losses	0.3067s	0.0054s	0.0113s
backward propagation	0.0536s	0.0320s	0.0314s
the whole epoch	6124s	1466s	520s

observations are as follows:

- LightMIRM outperforms all meta-IRM variants by a large margin. Firstly, on Cheryfs's dataset, the highest mean of KS value of complete meta-IRM is 0.5768, while our LightMIRM can reach 0.5792 under the same setting. Furthermore, for the worst KS value, our method achieves an even higher improvement, which is 0.0178 higher than the most competitive baseline.
- We further plot the evolution of testing KS value during training in Figure 6 to reveal the advantages of LightMIRM and to be clear of the training process. As shown in Figure 6, we observe that LightMIRM performs better than all of the sampling variants of the meta-IRM, and achieves similar scores with the complete meta-IRM. Note that LightMIRM requires much fewer operations than meta-IRM, which will be further discussed in Section IV-D. This suggests that we could reduce time complexity and computational complexity significantly, without sacrificing the performance of the model and even boosting it.
- For the converging efficiency, we see that the complete meta-IRM could converge more rapidly because of more computation. As a result, the curve of LightMIRM is below the complete meta-IRM at the beginning. However, more computation per epoch leads to a complex model which increases the probability of overfitting, as seen in Figure 6. Hence, the score of the LightMIRM begins to surpass the complete meta-IRM after 9 epochs and maintains this

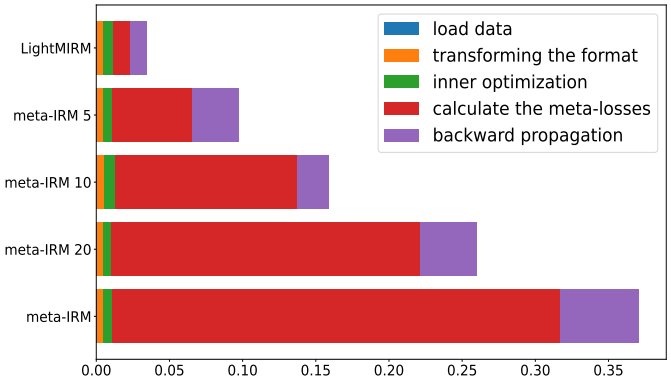


Fig. 7: The proportion of each step in the total time spent.

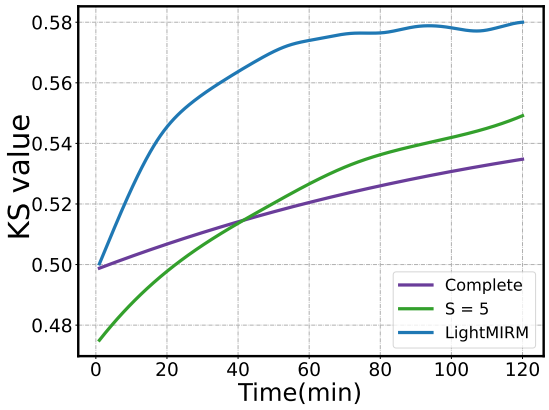


Fig. 8: The performance of different meta-IRM variants and LightMIRM during training.

advantage till the end.

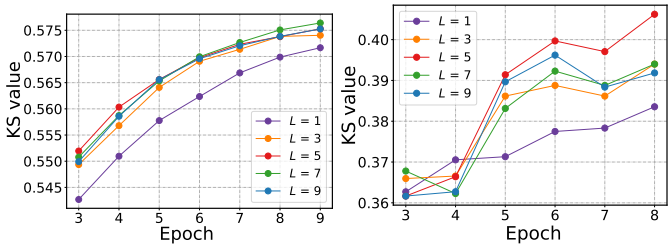
- In Figure 6, we can also see that the number of sampled provinces could largely affect the performance of meta-IRM. Moreover, compared with meta-IRM(20) which possesses similar converging epochs, the LightMIRM only contains 5 samples and thus is more computationally efficient but has a higher upper limit.

D. Time Complexity Analysis of Meta-IRM and LightMIRM

To answer **RQ3**, we compare the efficiency of the models and count the time of the different parts of the algorithms. In practice, we find that there are five operations that consume the most time, including loading data, transforming input into the one-hot format, inner optimization, calculating meta-losses, and final optimizing. Specially, we conduct experiments on CPU i7-11700 with 32GB RAM. Due to the trade secret requirements from Chery FS, we run the experiments on their workstations where GPU is not applicable. We leave the time complexity analysis on GPU for future work.

As shown in Figure 7, we observe that the loading data step consumes the least time which can be ignored. Moreover, the time cost for all the methods to transform the format and conduct inner optimization is nearly the same. However, complete meta-IRM spends lots of time to calculate the meta-losses, and LightMIRM is 30 times faster when conducting this operation.

What's more, as illustrated in Table III, the whole epoch time exhibits a great deal of variation. Compared with the complete meta-IRM, the LightMIRM takes only 8 minutes for an epoch while the complete meta-IRM needs 102 minutes. Our method can acquire a much stronger performance yet save 12 times as much time. Moreover, by combining 6 and Table III, we can see that our method has a higher score than meta-IRM(5), which takes a similar time to the proposed method.



(a) The Mean KS Value (b) The Minimum KS Value

Fig. 9: The impact of the length of MRQ. (a) represents the mean KS value of different lengths and (b) represents the minimum KS value of various lengths.

E. Ablation Analyses

To answer **RQ4**, we perform ablation studies on LightMIRM showing how the length of MRQ and weight of MRQ affect its performance.

1) Impact of the Length of the MRQ: Figure 9 shows the result of LightMIRM with different lengths *i.e.,* from 1 to 9. We have two main observations:

- When the LightMIRM has an MRQ of length 1, it degrades into meta-IRM sampling one province. In this case, the model has a poor performance on both the mean KS value and the minimum KS value compared to other settings. This finding indicates that meta-losses of other provinces are informative to the model of a specific province. Therefore, it is effective to utilize the MRQ structure to store meta-losses of different provinces.
- Focusing on the peak points of two images, we find that adopting MRQ of length 7 could get the highest mean score, and utilizing MRQ of length 5 can obtain the highest minimum KS value. The result verifies the effectiveness of our length-fixed queue on filtering far-distant history losses. Hence the length of MRQ should be chosen appropriately, neither too long nor too short. And generally, the performance of the proposed MRQ is stable around the optimal length.

2) Impact of the Weight γ of the MRQ: As shown in Table IV, our conclusion is two folds. Firstly, we observe that the MRQ with weight 1 has the worst performance in almost all of the metrics. Recall that γ controls the weight of meta-losses from other provinces, we suppose the relative weight

TABLE IV: The Impact of the Weight γ of the MRQ

γ	***mKS***	***wKS***	***mAUC***	***wAUC***
0.1	0.5784	0.4172	0.8343	**0.7548**
0.3	0.5779	0.415	0.8348	0.7521
0.5	0.5792	**0.4191**	0.8345	0.7523
0.7	0.5781	0.4144	0.8349	0.7526
0.9	**0.5794**	0.4183	**0.8351**	0.7518
1	0.5777	0.4170	0.8341	0.7489

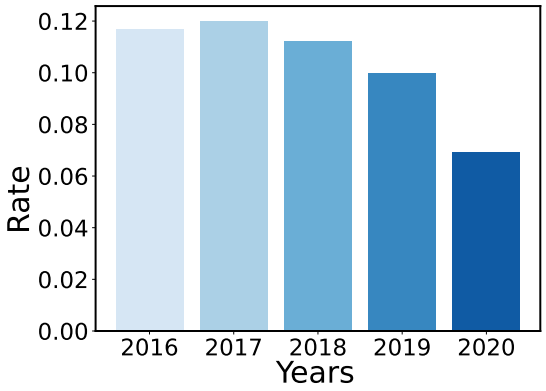


Fig. 10: The ratio of transactions in Guangdong to the all from 2016 to 2020.

TABLE V: The performance comparison on Guangdong

method	**KS**	**AUC**
ERM	0.6409	0.8818
Up Sampling	0.6475	0.8791
Group DRO	0.6365	0.8711
V-REx	0.6485	0.8794
meta-IRM	0.6489	0.8789
LightMIRM	**0.6539**	**0.8821**

of historical losses should be controlled since they are from the past iteration from other provinces. This finding supports our assumption that our meta-replaying method works the best when more attention is paid to the more recent losses. Furthermore, as we take a closer look at the result from weight 0.1 to weight 0.9, we find that none of the weights achieve the best performance constantly, suggesting that it is important to seek a balance between utilizing the meta-losses from other provinces and reducing the bad influence of past losses.

F. Effectiveness Analyses

To answer **RQ5**, we conduct an in-depth analysis of the effectiveness of our methods. As shown in Section IV-B, there are distribution shifts in data as time goes by. We guess that our methods can learn invariant features which are stable across different environments as well as at different times. In this subsection, the hypothesis is verified.

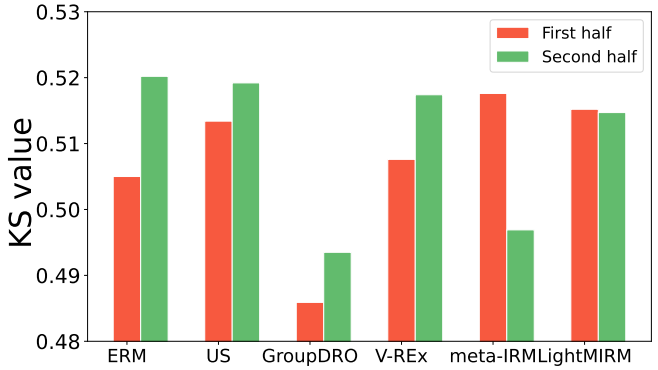


Fig. 11: The Performance of different methods on Hubei Province in 2020. And the "US" is the UpSampleing method.

1) The Performance on Special Provinces: In this section, we investigate the generalization of the model in some special provinces.

Firstly, because of the shift in focus of Chery FS's operations, the ratio of transactions in Guangdong Province decreases year by year. As illustrated in Figure 10, the share of Guangdong Province in 2020 is only half of what it was from 2016 to 2019. Therefore, we treat the data of Guangdong Province as out-of-distribution data which we utilize to evaluate the generalization ability of the methods.

As shown in Table V, We have the following observations:

- Among the methods, LightMIRM performs best with a KS value of 0.6539. The result indicates that the LightMIRM can learn invariant patterns which could resist the distribution shift induced by time. Therefore, we believe that the LightMIRM can perform well in industrial scenarios stably and sustainably.
- An interesting finding is that traditional ERM could get a decent AUC but a relatively low KS score. Given the knowledge that ERM may fail under data drift, we suppose the KS score is more reliable in this scenario where AUC may be misleadingly high.

Secondly, we pay attention to a special province. Recall that our train set consists of the data from 2016 to 2019, but the model is validated and tested on the data of 2020, there should be a large data drift on a special province that severely suffers from COVID-19. In the first half of 2020, the Hubei Province got hit by the epidemic and started to get on track in the second half of 2020. This period is such a special time that we assume that the patterns of the customers in Hubei Province changed greatly in the first half of 2020, but rolled back in the second half of 2020. Therefore, similarly to the change in Guangdong Province, we can evaluate the model trustworthiness in Hubei Province.

As shown in Figure 11, we focus on the generalization performance of the model in the first half of the year. And we have the following observations:

- Focusing on the first half of 2020, the ERM gets the lowest performance except for Group DRO. However, the ERM

TABLE VI: Performance comparison with random splitting (*i.e.,* i.i.d settings)

Methods	*mKS*	*wKS*	*mAUC*	*wAUC*
Up Sampling	0.6056	0.4983	0.8709	0.8093
Group DRO	0.5977	0.4944	0.8669	0.811
V-REx	0.6058	0.5019	0.8715	0.8147
meta-IRM (5)	0.6067	0.5216	0.8717	0.8208
meta-IRM(complete)	**0.6081**	0.5188	**0.8722**	**0.8235**
LightMIRM	0.6066	**0.5235**	0.8715	0.8223

method obtains the highest KS value in the second half of 2020. We attribute this phenomenon to two main reasons. Firstly, the result of the ERM verifies our assumption that the patterns of customers in Hubei Province change significantly in the first half of 2020 and roll back in the second half of 2020. Secondly, the ERM can perform well on data with similar distribution, however, it suffers from changes in the patterns of data.

- Our method obtains the highest KS value 0.5152. This means that our method could learn some invariant features less likely to be affected by COVID-19. And compare to other methods, our method could obtain a similar result in two periods which indicates that our method has higher security.
- We find that there is a balance between the performance of the model on the data in the first half of 2020 and the second half of 2020, indicating that the model which learns more invariant features could be more trustworthy. But the price is that it might lose some performance on data that does not change much. *e.g.,* the meta-IRM gets the best score in the first half of 2020, indicating its ability to capture more invariant features. But its performance sharply drops in the second half where the distribution is more similar to the original data.

2) The Performance on the i.i.d. Setting: We also conduct experiments on the i.i.d. setting where datasets are split randomly. In this case, the impacts of the time are eliminated so that we could evaluate the fairness of the method on the i.i.d. setting.

Table VI shows the result of LightMIRM and other OOD methods, which could represent the fairness of the methods. We have two main observations:

- The LightMIRM could obtain a similar mean score with meta-IRM when sampling number S=5. However, on the worst province, the LightMIRM could perform better than meta-IRM(5), even better than the complete meta-IRM which is the most competitive baseline. This finding suggests that the capacity of classification of the method could be affected by the number of the meta-losses and LightMIRM could obtain a fairer model in a similar setup.
- The best method is the complete meta-IRM, which obtains the 0.6081 mean KS value and 0.8722 mean AUC. This means that the methods which calculate more meta-losses could obtain a better score. However, as illustrated in Table III, the complete meta-IRM takes 12 times longer than LightMIRM while only gaining 0.0015 mean KS higher. Furthermore, compared with Table I, we have reason to believe that our method performs well on time-changing data because our method can capture more invariant features across time and thus have better robustness. To conclude, LightMIRM is more suitable for industrial scenarios.

V. Conclusion

In this work, we revealed that existing loan default prediction methods face trustworthy issues due to lacking the minimax fairness among different environments. To achieve fair predictions, we proposed to apply invariant risk minimization to learn the prediction models. Furthermore, we proposed a rapid meta-learning-based solution named LightMIRM, which speeds up the training with environment sampling and meta-loss replaying, to effectively and efficiently implement IRM in our industrial scenarios. We conducted extensive experiments on industrial data and provided insightful analyses of the experimental results, demonstrating the effectiveness of our proposal in improving training efficiency and solving trustworthy issues.

We believe solving the trustworthy issues is vitally important for the applications of machine-learning methods in industry, to meet legal regulations' requirements and increase user trust. This work mainly focuses on the minimax fairness problem in loan default prediction. In the future, we will extend our machine-learning-based load default prediction system to solve more trustworthy issues, such as individual fairness and transparency [40]. Besides, we will explore applying more advanced machine-learning methods in our industrial scenarios with the consideration of satisfying the explainability, efficiency, and trustworthiness demands.

References

[1] U. Aslam, H. I. Tariq Aziz, A. Sohail, and N. K. Batcha, "An empirical study on loan default prediction models," *Journal of Computational and Theoretical Nanoscience*, vol. 16, no. 8, pp. 3483–3488, 2019.

[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 (Statistics in Society)*, vol. 160, no. 3, pp. 523–541, 1997.

[3] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[4] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*. ACM, 2016, pp. 785–794.

[6] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *NIPS*, 2017, pp. 3146–3154.

[7] A. Mantelero, "The eu proposal for a general data protection regulation and the roots of the 'right to be forgotten'," *Computer Law & Security Review*, vol. 29, no. 3, pp. 229–235, 2013.

[8] S. Shekhar, G. Fields, M. Ghavamzadeh, and T. Javidi, "Adaptive sampling for minimax fair classification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 535–24 544, 2021.

[9] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *CoRR*, vol. abs/1907.02893, 2019.

[10] J. Peters, P. Bühlmann, and N. Meinshausen, "Causal inference by using invariant prediction: identification and confidence intervals," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 78, no. 5, pp. 947–1012, 2016.

[11] J. Bae, I. Choi, and M. Lee, "Meta-learned invariant risk minimization," *CoRR*, vol. abs/2103.12947, 2021.

[12] Y.-Q. Chen, J. Zhang, and W. W. Ng, "Loan default prediction using diversified sensitivity undersampling," in *International Conference on Machine Learning and Cybernetics*, 2018, pp. 240–245.

[13] N. Ghatasheh, "Business analytics using random forest trees for credit risk prediction: a comparison study," *International Journal of Advanced Science and Technology*, vol. 72, pp. 19–30, 2014.

[14] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, pp. 1565–1567, 2006.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[16] J. Xu, Z. Lu, and Y. Xie, "Loan default prediction of chinese p2p market: a machine learning methodology," *Scientific Reports*, vol. 11, pp. 1–19, 2021.

[17] B. Li, "Online loan default prediction model based on deep learning neural network," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[18] M. Aitken, E. Toreini, P. Carmichael, K. Coopamootoo, K. Elliott, and A. van Moorsel, "Establishing a social licence for financial technology: Reflections on the role of the private sector in pursuing ethical data practices," *Big Data & Society*, vol. 7, no. 1, p. 2053951720908892, 2020.

[19] E. Toreini, M. Aitken, K. P. L. Coopamootoo, K. Elliott, V. González-Zelaya, P. Missier, M. Ng, and A. van Moorsel, "Technologies for trustworthy machine learning: A survey in a socio-technical context," *CoRR*, vol. abs/2007.08911, 2020.

[20] S. Caton and C. Haas, "Fairness in machine learning: A survey," *arXiv preprint arXiv:2010.04053*, 2020.

[21] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[22] E. Pitoura, K. Stefanidis, and G. Koutrika, "Fairness in rankings and recommendations: an overview," *The VLDB Journal*, pp. 1–28, 2021.

[23] J. M. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores," in *8th Innovations in Theoretical Computer Science Conference*, vol. 67, 2017, pp. 43:1–43:23.

[24] G. Pleiss, M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger, "On fairness and calibration," in *NIPS*, 2017, pp. 5680–5689.

[25] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," *CoRR*, vol. abs/1911.08731, 2019.

[26] D. Krueger, E. Caballero, J. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. L. Priol, and A. C. Courville, "Out-of-distribution generalization via risk extrapolation (rex)," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 5815–5826.

[27] U. Hébert-Johnson, M. Kim, O. Reingold, and G. Rothblum, "Multicalibration: Calibration for the (computationally-identifiable) masses," in *ICML*, 2018, pp. 1939–1948.

[28] E. Creager, J.-H. Jacobsen, and R. Zemel, "Environment inference for invariant learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2189–2200.

[29] J. Liu, Z. Hu, P. Cui, B. Li, and Z. Shen, "Heterogeneous risk minimization," in *International Conference on Machine Learning*, 2021, pp. 6804–6814.

[30] P. Kamath, A. Tangella, D. Sutherland, and N. Srebro, "Does invariant risk minimization capture invariance?" in *International Conference on Artificial Intelligence and Statistics*, 2021, pp. 4069–4077.

[31] Y. Dranker, H. He, and Y. Belinkov, "Irm—when it works and when it doesn't: A test case of natural language inference," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 212–18 224, 2021.

[32] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1126–1135.

[33] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[34] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela, "Practical lessons from predicting clicks on ads at facebook," in *ADKDD@KDD*. ACM, 2014, pp. 5:1–5:9.

[35] C. Cheng, F. Xia, T. Zhang, I. King, and M. R. Lyu, "Gradient boosting factorization machines," in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014, pp. 265–272.

[36] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. S. Turaga, "Learning feature engineering for classification." in *Ijcai*, 2017, pp. 2529–2535.

[37] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade (2nd ed.)*, ser. Lecture Notes in Computer Science. Springer, 2012, vol. 7700, pp. 437–478.

[38] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[39] F. J. M. Jr., "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[40] D. Kaur, S. Uslu, K. J. Rittichier, and A. Durresi, "Trustworthy artificial intelligence: a review," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–38, 2022.