Shallow Self-Learning for Reject Inference in Credit Scoring

Nikita Kozodoi[1,2] (✉), Panagiotis Katsas[2],
Stefan Lessmann[1], Luis Moreira-Matias[2], and Konstantinos Papakonstantinou[2]

[1] Humboldt University of Berlin, Berlin, Germany
[2] Kreditech, Hamburg, Germany
nikita.kozodoi@external.kreditech.com

Abstract. Credit scoring models support loan approval decisions in the financial services industry. Lenders train these models on data from previously granted credit applications, where the borrowers' repayment behavior has been observed. This approach creates sample bias. The scoring model is trained on accepted cases only. Applying the model to screen applications from the population of all borrowers degrades its performance. Reject inference comprises techniques to overcome sampling bias through assigning labels to rejected cases. This paper makes two contributions. First, we propose a self-learning framework for reject inference. The framework is geared toward real-world credit scoring requirements through considering distinct training regimes for labeling and model training. Second, we introduce a new measure to assess the effectiveness of reject inference strategies. Our measure leverages domain knowledge to avoid artificial labeling of rejected cases during evaluation. We demonstrate this approach to offer a robust and operational assessment of reject inference. Experiments on a real-world credit scoring data set confirm the superiority of the suggested self-learning framework over previous reject inference strategies. We also find strong evidence in favor of the proposed evaluation measure assessing reject inference strategies more reliably, raising the performance of the eventual scoring model.

Keywords: Credit scoring · Reject inference · Self-learning · Evaluation

1 Introduction

Financial institutions use supervised learning to guide lending decisions. The resulting credit scoring models, also called scorecards, predict the probability of default (PD) – an applicant's willingness and ability to repay debt [31]. Loan approval decisions are made based on whether the scorecard predicts an applicant to be a repaying borrower (*good* risks) or a likely defaulter (*bad* risks).

Scoring models are trained on data of accepted applicants. Their repayment behavior has been observed, which provides the labels for supervised learning. Inevitably, the sample of accepted clients (accepts) differs from the overall population of credit applicants. Accepts have passed the screening of the lender's

scorecard, whereas the population also includes clients who have been denied credit by that scorecard (rejects) as well as customers who have not applied for credit. As a result, scoring models suffer from sample bias. Training a classifier only on data from accepts deteriorates the accuracy of PD predictions when the scorecard is out into production for screening incoming credit applications [28].

Reject inference refers to techniques that remedy sampling bias through inferring labels for rejects. Previous research has suggested several approaches including naive strategies (e.g., label all rejects as *bad*) and model-based techniques [28]. However, empirical evidence concerning the value of reject inference and the efficacy of labeling strategies is scarce. Several studies use incomplete data, which only contain accepted cases (e.g. [5, 11]), do not have a labeled unbiased sample with both accepts and rejects (e.g., [7]) or use synthetic data (e.g., [16]). In addition, the data sets employed in prior studies are usually low-dimensional (e.g., [21]), which is not representative of the real-world credit scoring data used today [33]. Previous work is also geared toward linear models and support vector machines (SVM) [1, 19, 21]. Yet, there is much evidence that other algorithms (e.g., tree-based ensembles) outperform these methods in credit scoring [18, 34].

The contribution of this paper is two-fold. First, we introduce a novel self-learning framework for reject inference in credit scoring. Our framework includes two different probabilistic classifiers for the training and labeling stages. The training stage benefits from using a strong learner such as gradient boosting. However, we suggest using a shallow (i.e. weaker) learner for the labeling stage and show that it achieves higher calibration with respect to the true PD [23]. As a result, we maximize the precision of our model on the extreme quantiles of its output and minimize the noise introduced on newly labeled rejects.

Second, we introduce a novel measure (denoted as *kickout*) to assess reject inference methods in a reliable and operational manner. Aiming at labeling rejects to raise the scorecard performance, the acid test of a reject inference strategy involves comparing a scorecard without correction for sample bias to a model that has undergone reject-inference based correction on data from an unbiased sample of clients including both accepts and rejects with actual labels for both groups of clients. Such a sample would represent the operating conditions of a scorecard and thus uncover the true merit of reject inference [11]. However, obtaining such a sample is very costly as it requires a financial institution to lend money to a random sample of applicants including high-risk cases that would normally be denied credit. Drawing on domain knowledge, the proposed *kickout* measure avoids dependence on the actual labels of rejects and, as we establish through empirical experimentation, assesses the merit of a reject inference method more accurately than previous evaluation approaches. The data set used in this paper includes an unbiased sample containing both accepts and rejects, giving us a unique opportunity to evaluate a scorecard in its operating conditions.

The paper is organized as follows. Section 2 reviews related literature on reject inference. Section 3 revisits the reject inference problem, presents our self-learning framework and introduces the kickout measure. Section 4 describes our experimental setup and reports empirical results. Section 5 concludes the paper.

2 Literature Review

The credit scoring literature has suggested different model-based and model-free approaches to infer labels of rejected cases. Some model-free techniques rely on external information such as expert knowledge to manually label rejects [22]. Another approach is to label all rejected cases as *bad* risks [28], assuming that the default ratio among the rejects is sufficiently high. One other strategy is to obtain labels by relying on external performance indicators such as credit bureau scores or an applicant's outcome on a previous loan [2, 28].

Model-based reject inference techniques rely on a scoring model to infer labels for rejects. Table 1 depicts corresponding techniques, where we sketch the labeling strategy used in a study together with the base classifier that was used for scorecard development. Table 1 reveals that most reject inference techniques have been tested with linear models such as logistic and probit regression.

The literature distinguishes several approaches toward model-based reject inference such as augmentation, extrapolation, bivariate models and others [19]. Extrapolation refers to a set of techniques that use the initial scoring model trained on the accepts to label the rejected cases. For instance, hard cutoff augmentation labels rejects by comparing their model-estimated PDs to a predefined threshold [28]. Parceling introduces a random component, separating the rejected cases into segments based on the range (e.g., percentile) of PDs. Instead of assigning labels based on the individual scores of rejects, they are labeled randomly within the identified segments based on the expected default rate for each score range. A drawback of such techniques is their reliance on the performance of the initial scoring model when applied to rejects.

Augmentation (or re-weighting) is based on the fact that applicants with a certain distribution of features appear in the training data disproportionately due to a non-random sample selection [11]. Re-weighting refers to the techniques that train an additional model that separates accepts and rejects and predicts the probability of acceptance. These probabilities are then used to compute sampling weights for a scoring model.

Some scholars suggest using a two-stage bivariate probit model or two-stage logistic regression to perform reject inference [6]. A bivariate model incorporates the Heckman's correction to account for a sample bias within the model, estimating both acceptance and default probability. These models assume linear effects within the logistic or probit regression framework.

Empirical studies have shown little evidence that reject inference techniques described above improve scorecard's performance [3, 9, 11, 32]. Recently suggested alternatives rely on semi-supervised learning. For example, Maldonado et al have shown that self-learning with SVM outperforms well-known reject inference techniques such as ignoring rejects or labeling all rejects as *bad* risks [21]. Their work is continued by [19], who propose a semi-supervised SVM that uses a non-linear kernel to train a scoring model.

We follow recent studies and cast the reject inference problem in a semi-supervised learning framework. Our approach to solve the problem is a variation of self-learning adapted to a credit scoring context by extending the work of [21].

Table 1. Model-Based Reject Inference Methods

Reference	Inference technique	Base model
Reichert et al (1983) [24]	LDA-based	LDA
Joanes (1993) [16]	Reclassification	LR
Hand et al (1993) [15]	Ratio prediction	-
Hand et al (1993) [15]	Rebalancing model	-
Feelders (1999) [12]	Mixture modeling	LR, QDA
Banasik et al (2003) [6]	Augmentation	LR, Probit
Smith et al (2004) [29]	Bayesian network	Bayesian
Crook et al (2004) [11]	Reweigthing	LR
Verstraeten et al (2005) [32]	Augmentation	LR
Banasik et al (2005) [3]	Augmentation	LR
Fogarty (2006) [13]	Multiple imputation	LR
Montrichard (2007) [22]	Fuzzy augmentation	LR
Banasik et al (2007) [4]	Augmentation	LR, Probit
Banasik et al(2007) [4]	Bivariate probit	Probit
Kim et al (2007) [17]	Bivariate probit	-
Banasik et al (2010) [5]	Augmentation	Survival
Maldonado et al (2010) [21]	Self-training	SVM
Maldonado et al (2010) [21]	Co-training	SVM
Maldonado et al (2010) [21]	Semi-supervised SVM	SVM
Chen et al (2012) [10]	Bound and collapse	Bayesian
Bücker et al (2012) [7]	Reweighting	LR
Siddiqi (2012) [28]	Define as bad	-
Siddiqi (2012) [28]	Soft cutoff augmentation	-
Siddiqi (2012) [28]	Hard cutoff augmentation	-
Siddiqi (2012) [28]	Parceling	-
Siddiqi (2012) [28]	Nearest neighbors	-
Anderson et al (2013) [1]	Mixture modeling	LR
Li et al (2017) [19]	Semi-supervised SVM	SVM

3 Methodology

3.1 Self-Learning for Reject Inference

In reject inference, we are given a set of n examples $x_1, ..., x_n \in \mathbb{R}^k$, where k is the number of features. Set X consists of l accepted clients $x_1^a, ..., x_l^a \in X^a$ with corresponding labels $y_1^a, ..., y_l^a \in \{good, bad\}$ and m rejected examples $x_1^r, ..., x_m^r \in X^r$, whose labels are unknown. To overcome sampling bias and eventually raise scorecard accuracy, reject inference aims at assigning labels $y_1^r, ..., y_m^r$ to the rejected examples, which allows using the combined data for training a scoring model.

Standard self-learning starts with training a classifier $f(x)$ on the labeled examples $x_1^a, ..., x_l^a$ and using it to predict the unlabeled examples $x_1^r, ..., x_m^r$. Next, the subset of unlabeled examples $X^* \subset X^r$ with the most confident predictions is selected such that $f(x_i^* \in X^*) > \alpha$ or $f(x_i^* \in X^*) < 1 - \alpha$, where α is a probability threshold corresponding to a specified percentile of $f(x_i^* \in X^r)$. The selected rejects are labeled in accordance with the classifier's predictions. Cases

obtained within this process are removed from X^r and appended to X^a to form a new labeled sample X_1^a. Finally, the classifier is retrained on X_1^a and used to score the remaining cases in X^r. The procedure is repeated until all cases from X^r are assigned labels or until certain stopping criteria are fulfilled [25].

Self-learning assumes that labeled and unlabeled examples in X follow the same distribution [25]. In a credit scoring context, X^a and X^r come from two different distributions because the scoring model employed by the financial institution separates accepts and rejects based on their feature values. The difference in distributions has negative consequences for self-learning: since the initial model is trained on a sample that is not fully representative of the unlabeled data, predictions of this model for the rejects are less reliable. The error is propagated through the iterative self-learning framework, which deteriorates the performance of the final model due to the incorrectly assigned labels.

In this section, we describe a novel shallow self-training framework for reject inference that is geared toward reducing the negative effects of sample bias. The proposed framework consists of three stages: filtering, labeling and model training. We summarize the algorithm steps in Algorithm 2.

Within the proposed framework, we suggest to filter and drop some rejected cases before assigning them with labels. The goal of the filtering stage is two-fold. Firstly, we strive to remove rejected cases that come from the most different part of distribution compared to the accepts. Removing these cases would reduce the risk of error propagation, since predictions of the model trained on the accepts become less reliable as the distribution of cases to be predicted becomes more different from the one observed on the training data. Secondly, we remove rejects that are most similar to the accepted cases. Labeling such cases would potentially provide little new information for a scorecard and might even harm performance due to introducing noise. Therefore, the filtering stage is aimed at removing the cases that could have a negative impact of the scorecard performance.

The filtering is performed with isolation forest, which is a novelty detection method that estimates the normality of a specific observation by computing the number of splits required to isolate it from the rest of the data [20]. We train isolation forest on all accepts in X^a and use it to evaluate the similarity of the rejects in X^r. Next, we remove rejects that are found to be the most and least similar to the accepts by dropping cases within the top β_t and bottom β_b percentiles of the similarity scores. Algorithm 1 describes the filtering stage.

After filtering, we use self-learning with distinct labeling and training regimes to perform reject inference. While the scoring model is based on a tree-based algorithm (gradient boosting), we propose using a weak learner for labeling rejects because of its ability to produce better-calibrated predictions [23]. In this paper, we rely on L1-regularized logistic regression (L1) to label rejects.

1 train isolation forest classifier $g(x)$ using all data in X^a;
2 use $g(x)$ to evaluate similarity scores of all unlabeled examples in X^r;
3 select a subset $X^* \subset X^r$ such that $g(x_i^* \in X^*) \in [\beta_b, \beta_t]$, where β_b and β_t are values of pre-defined percentiles of $g(x_j^r \in X^r)$, $j = 1, ..., m$.

Algorithm 1: Isolation Forest for Filtering Rejected Examples

Logistic regression is a parametric learner which assumes a Gaussian distribution of the data. Because of this assumption, predicted probabilities can be output directly by the sigmoid function. In contrast, XG is a non-parametric learner which has more degrees of freedom and a higher potential for inductive bias reduction. Predicted scores produced by XG are not well calibrated [23]. Consider the example score distributions of L1 and extreme gradient boosting (XG) depicted in Figure 1. Here, adding regularization to logistic regression is important as we are dealing with high-dimensional data with noisy features. Compared to L1, the range of the output probabilities of XG is wider.

Within the proposed framework, we require the labeling model to produce well-calibrated probabilities as we limit the number of selected rejects based on the predicted PD values. Furthermore, by using different base models for application scoring and reject inference, we strive to reduce bias and error propagation. Hence, using a weak learner for reject inference is more promising.

An important aspect of our framework is to account for a higher default rate among the rejects [21]. Recall that X is partitioned into accepts and rejects based on a scoring model that is currently employed by a financial institution. Assuming that the scoring model in place performs better than a random loan allocation, we expect that the default rate among rejects is higher than among accepts. To address that difference, we introduce the imbalance parameter θ into our self-learning framework. On each labeling iteration, we only select the top $\alpha\%$ of the *good* loans and top $\alpha\theta\%$ of the *bad* loans among rejects for labeling. Keeping only the top-ranked instances ensures that we append rejects with high confidence in the assigned labels, reducing the potential amount of noise. By setting $\theta > 1$ we append more *bad* cases to the training data, accounting for the imbalance. Parameter θ can be optimized at the parameter tuning stage.

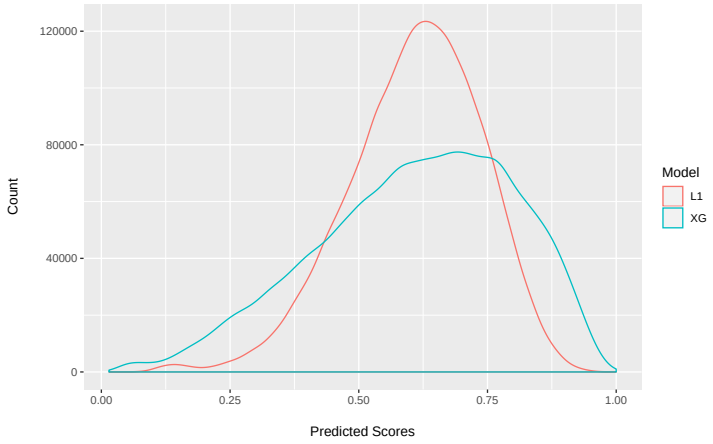

Fig. 1. *Score Densities.* The figure compares the distributions of the scores predicted by two scoring models: L1-regularized logistic regression (red) and extreme gradient boosting (blue). Both models use the same data.

filter rejected cases in X^r with isolation forest (see Algorithm 1);
set $X^* = X^r$;
while $X^* \neq \emptyset$ **do**
    train L1 classifier $f(x)$ with penalty parameter λ on all data in X^a;
    use $f(x)$ to predict PD for all unlabeled examples in X^*;
    if $c_b = \{\}$ *and* $c_g = \{\}$ **then**
        derive c_g: $P(f(x_i^* \in X^*) < c_g) = \alpha$, α is a percentile threshold;
        derive c_b: $P(f(x_i^* \in X^*) > c_b) = \alpha\theta$, θ is the imbalance parameter;
    end
    select $X^* \subset X^r$ such that $f(x_i^* \in X^*) < c_g$ or $f(x_i^* \in X^*) > c_b$;
    remove examples in X^* from X^r and append them to X^a;
end
train a scoring model $s(x)$ using XG classifier on all cases in X^a.

Algorithm 2: Shallow Self-Learning for Reject Inference

Different variants of self-learning consider different ways to choose the most confident cases for labeling: either selecting top and bottom percentiles of the probability distribution or selecting cases based on a pre-defined probability threshold [8]. We suggest using the combined approach: on the first iteration, we compute the corresponding score values c_g and c_b for the selected $\alpha\%$ and $\alpha\theta\%$ probability percentiles. Since the labeling model is geared toward providing well-calibrated probabilities, we fix the absolute values c_g and c_b as thresholds for the subsequent iterations. By doing that, we reduce the risk of error propagation on further iterations. The absence of rejected cases with predicted scores above the fixed thresholds serves as a stopping criterion.

3.2 Proposed Evaluation Measure

Performance evaluation is an important part of selecting a suitable reject inference technique. In practice, accurate evaluation of reject inference is challenging. The true labels of rejects are unknown, which prohibits estimating the accuracy directly. Therefore, prior research evaluates the performance of a given technique by comparing the performance of the scorecard before and after appending the labeled rejects to the training data [3, 6, 19]. The major downside of this approach is that the performance of a scorecard is not evaluated on a representative sample, which should include both accepts and rejects. Since labels of rejects are unknown, the literature suggests to evaluate models on a holdout sample drawn from the accepts which exhibits sample bias (e.g., [21]). Very few empirical studies have access to the data on both accepts and rejects for evaluation [11].

Model selection based on the performance on accepts might lead to selecting a sub-optimal model. Let us illustrate that by comparing the performance of different scoring models validated on the accepts (4-fold stratified cross-validation) and on the unbiased sample consisting of both accepts and rejects. We train a set of scoring models with different meta-parameter values and evaluate their per-

formance in terms of the area under the receiver operating characteristic curve (AUC) [26]. Here, XG is used as a base classifier. Figure 2 depicts the results.

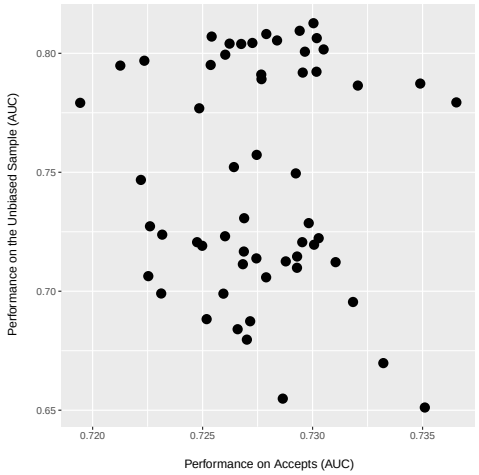

Fig. 2. Comparing AUC on the accepted cases (4-fold stratified cross-validation) and the unbiased sample. The dots indicate scoring models with different meta-parameters.

The rank correlation between AUC values is just 0.0132. Due to the distribution differences between the accepted and rejected cases, the model's performance on the accepted applicants becomes a poor criterion for model selection. This result suggests that there is a need to develop an alternative measure for comparing and evaluating the scoring models in the presence of sample bias.

Without access to an unbiased sample that contains data on a representative set of applicants, the literature suggests performing the evaluation by using synthetic data [16], emulating rejected cases by artificially moving the acceptance threshold [21] or using other criteria based on the applicants' feature values [9]. In this paper, we suggest using *kickout* – a novel evaluation measure based on the known data. We argue that developing such a measure is a valuable contribution since obtaining an unbiased data sample for performance evaluation is costly.

The key idea of *kickout* is to compare a set of applications accepted by a scoring model before and after reject inference. Recall that we have data on the previously accepted X^a and rejected applicants X^r. Here, we partition X^a into two subsets: X^a_{train} and $X^a_{holdout}$. Let $s_1(x)$ be a scoring model trained on X^a_{train}. We use $s_1(x)$ to score cases from $X^a_{holdout}$ and select a pool of customers $A_1 \subset X^a_{holdout}$ that would be accepted by the model using the acceptance rate μ. Thus, A_1 contains the (simulated) accepted applications before reject inference.

The rejected cases in X^r are also split into two subsets: X^r_{train} and $X^r_{holdout}$. The former is labeled with a reject inference technique and appended to the X^a_{train}. Rejected cases in $X^r_{holdout}$ are appended to $X^a_{holdout}$, which now contains

labeled accepts and unlabeled rejects, simulating the production-stage environment. Next, we train a new scoring model $s_2(x)$ on the expanded training sample X^a_{train} and use it to score and select customers in $X^a_{holdout}$ using the same acceptance rate μ. Since both training and holdout samples have changed, model $s_2(x)$ would accept a different pool of customers A_2. Analyzing the differences between A_1 and A_2, we can identify the kicked-out cases – applications that were included in A_1 but do not appear in A_2.

We define the *kickout* metric as follows:

$$kickout = \frac{\frac{\mathrm{K}_B}{\mathrm{p}(B)} - \frac{\mathrm{K}_G}{1-\mathrm{p}(B)}}{\frac{\mathrm{S}_B}{\mathrm{p}(B)}}, \qquad kickout \in [-1, 1] \tag{1}$$

where K_B is the number of *bad* cases kicked out from the set of accepted cases after performing reject inference, K_G is the number of kicked-out *good* cases, S_B is the number of *bad* cases selected by the original model, and $\mathrm{P}(B)$ is the share of *bad* cases in A_1. The *kickout* metric ranges from -1 (all *good* cases and no *bad* cases are kicked out) to 1 (all *bad* cases and no *good* cases are kicked out). We normalize the metric by the share of *bad* cases to reflect the difficulty of kicking out a *bad* customer. Positive values of *kickout* signal a positive impact of reject inference, with higher values indicating a better performance.

It is important to note that *kickout* does not require knowing the actual labels of the rejected cases that replace previously accepted cases. Instead, the metric focuses on the kicked-out applications. Replacing a *bad* loan with a rejected case may have two possible outcomes. If the newly selected rejected case is also *bad*, we are indifferent between the old and the new scoring model. If the rejected case is *good*, the scoring model improves. Therefore, kicking out a *bad* case has a positive expected value. In contrast, kicking out a *good* case has a negative expected value: we are indifferent between the old and the new scoring model if the new rejected case is *good*, whereas scorecard performance deteriorates if the rejected case is *bad*. Hence, a good reject inference technique should change a scorecard such that it starts to kick out more *bad* and less *good* customers.

The proposed measure relies on two assumptions. First, we assume that all *bad* loans and all *good* loans have the same expected value: that is, replacing one *bad* case with another *bad* case does not have any effect on the model's performance. Given the stable interest rates that determine the return on investment at fixed terms [31] and an uncertain relationship between a loan amount and its PD, we argue that this assumption is reasonable in a credit scoring context. Second, we assume that the *bad* ratio among rejected cases is higher compared to the accepted applications. As we detailed above, this assumption holds if the employed scoring model performs better than random.

4 Experimental Results

4.1 Data Description

The empirical experiments are based on a real-world credit scoring data set on consumer micro-loans provided by Kreditech, a Germany-based financial institution. Although the data are not available publicly, it provides a unique opportunity to study reject inference on a high-dimensional data set which includes an unbiased sample with customers who have been granted a loan without scoring.

Table 2. Data Summary

Characteristic	Accepts	Rejects	Unbiased
Number of cases	39,579	18,047	1,967
Number of features	2,410	2,410	2,410
Default rate	0.39	unknown	0.66

The data set contains 2,410 features describing the applicants, their behavior and loan characteristics. The target variable is a binary indicator of whether the customer has repaid the loan. The data consist of 59,593 loan applications, out of which 39,579 were accepted and 18,047 were rejected. The target variable is only observed for the accepts, whereas the repayment status of rejects is unknown. Table 2 summarizes the main characteristics of the data set.

The unbiased sample contains 1,967 customers accepted without scoring. The sample, therefore, includes cases that would normally be rejected by a scorecard. This makes it representative of the through-the-door population of customers who apply for a loan. As noted in Table 2, the default rate in the unbiased sample is 1.7 times higher than on the accepted cases. The unbiased sample allows us to evaluate the performance gains from reject inference on the sample representative of the production environment.

4.2 Experimental Setup

To evaluate the effectiveness of our propositions, we perform two experiments. Experiment 1 benchmarks the proposed self-learning framework against conventional reject inference techniques and standard self-learning. In the second experiment, we illustrate the effectiveness of the new *kickout* measure for model selection. Below, we describe the modeling pipeline for these experiments.

We partition the data into three subsets: accepts, rejects and the unbiased holdout sample. Next, we use 4-fold stratified cross-validation on accepts to perform reject inference. On each iteration, the training folds are used to develop a reject inference technique that is used to infer labels of the rejects. Next, labeled rejects are appended to the training folds, providing a new sample to train a scoring model. Finally, a scoring model after reject inference is evaluated on the remaining fold and on the holdout sample. To ensure robustness, we evaluate

performance on 50 bootstraped samples of the holdout set. Performance metrics of the reject inference techniques are then averaged over 4×50 obtained values.

We use XG classifier as a scoring model in both experiments. Meta-parameters of XG are tuned once on a small subset of training data using grid search. Within the experiments, we employ early stopping with 100 rounds while setting the maximum number of trees to 10,000 to fine-tune the model for each fold.

In Experiment I, we compare the suggested self-learning framework to the following benchmarks: ignore rejects, label all rejects as *bad* risks, hard cutoff augmentation, parceling, cross-validation-based voting and standard self-learning. Here, cross-validation-based voting is an adaption of a label noise correction method suggested by [30]. It refers to an extension of hard cutoff augmentation that employs a homogeneous ensemble of classifiers based on different training folds instead of a single scoring model to label the rejects. The labels are only assigned to the cases for which all individual models agree on the label.

We test multiple versions of each reject inference technique with different meta-parameter values using grid search. For shallow self-learning, penalty λ of the labeling model is tuned and optimized once on the first labeling iteration. Table 3 provides the candidate values of meta-parameters.

For performance evaluation, we use three metrics that capture different dimensions of the predictive performance: AUC, Brier Score (BS) and R-Precision (RP). We use AUC as a well-known indicator of the discriminating ability of a model. In contrast, BS measures the calibration of the predicted default probabilities. Last, we use RP as it better reflects the business context. The financial institution that provided data for this study decides on a loan allocation by approving a certain percentage of the least risky customers. RP measures performance only for cases which will indeed be accepted. In our experiments, we compute RP in the top 30% of the applications with the lowest predicted PDs.

In Experiment II, we compare different variants of self-learning using grid search within the cross-validation framework described above. Apart from the three selected performance measures, we also evaluate reject inference in terms of the proposed *kickout* measure. The goal of this experiment is to compare model rankings based on three evaluation strategies: performance on the accepts, performance on the unbiased sample and performance in terms of *kickout*.

4.3 Empirical Results

Experiment I: Assessing the Shallow Self-Learning

Table 4 summarizes the performance of the reject inference techniques on the accepted cases and on the unbiased sample. Recall that the latter serves as a proxy for the production-stage environment for a scoring model, whereas performance on accepts refers to a conventional approach toward evaluation in credit scoring. According to the results, not all methods improve on the benchmark of ignoring rejects: only three out of six techniques achieve higher AUC and lower BS on the unbiased sample, and only one has a higher RP.

Labeling rejects as *bad* performs better than disregarding reject inference on the accepts but does substantially worse on the unbiased sample. In contrast,

parceling is outperformed by all other techniques on the accepts but has higher AUC on the unbiased sample. These results support the argument that performance on accepts might be a poor indicator of the production-stage performance.

Regular self-learning outperforms ignoring rejects in terms of AUC and BS but does not improve in terms of RP. The proposed self-learning framework performs best in all three measures on the unbiased sample as well as on the accepted applicants. The best performance is achieved by a self-learning model that includes filtering of rejects ($\beta_b = 1 - \beta_t = 0.02$). Therefore, the suggested modifications help to adjust self-learning for the reject inference problem.

Table 3. Reject Inference Techniques: Parameter Grid

Technique	Parameter	Candidate values
Label all as *bad*	–	–
Hard cutoff augmentation	probability threshold	0.3, 0.4, 0.5
Parceling	multiplier	1, 2, 3
	no. batches	10
CV-based voting	probability threshold	0.3
	no. folds	2, 5, 10
Regular self-learning	labeled percentage α	0.01, 0.02, 0.03
	max no. iterations	5
Shallow self-learning	filtered percentage β_b	0, 0.02
	filtered percentage β_t	1, 0.98
	penalty parameter λ	$2^{-8}, 2^{-7.5}, ..., 2^{8}$
	labeled percentage α	0.01, 0.02, 0.03
	imbalance parameter θ	1, 2
	max no. iterations	5

Performance gains appear to be modest, supporting the prior findings [15]. We check statistical significance of the differences using Friedman's rank sum test and Nemenyi pairwise test [14]. According to Friedman's test, we reject the null hypothesis that all reject inference techniques perform the same at 5% level for AUC ($\chi^2 = 419.82$), RP ($\chi^2 = 326.99$) and BS ($\chi^2 = 485.59$). Nemenyi test indicates that shallow self-learning performs significantly better than all competitors in terms of AUC and RP, whereas differences in BS between standard and shallow self-learning are not statistically significant at 5% level.

Even small differences might have a considerable effect on the costs of the financial institution [27]. Comparing shallow self-learning to ignoring rejects, 0.006 increase in RP translates to 60 less defaulted loans for every 10,000 accepted clients. Considering the average personal loan size of $\$17,100$ and interest rate of 10.36% observed in the US in Q1 2019[1], potential gains from reject inference could amount for up to $\$1.13$ million depending on the recovery rates.

[1] Source: https://www.supermoney.com/studies/personal-loans-industry-study/

Table 4. Comparing Performance of Reject Inference Techniques

Method	Accepted cases			Unbiased sample		
	AUC	BS	RP	AUC	BS	RP
Ignore rejects	0.7297	0.1829	0.8436	0.8007	0.2092	0.7936
Label all as bad	0.7332	0.1816	0.8474	0.6797	0.2284	0.7253
Hard cutoff augmentation	0.7295	0.1770	0.8430	0.7994	0.2212	0.7751
Parceling	0.7277	0.1842	0.8430	0.8041	0.1941	0.7851
CV-based voting	0.7293	0.1804	0.8430	0.7167	0.2160	0.7510
Regular self-learning	0.7302	0.1758	0.8434	0.8063	0.1838	0.7929
Shallow self-learning	**0.7362**	**0.1736**	**0.8492**	**0.8070**	**0.1799**	**0.7996**

Table 5. Correlation between Evaluation Strategies

Evaluation strategy	(1)	(2)	(3)
(1) AUC on the accepted cases	1		
(2) AUC on the unbiased sample	−0.0009	1	
(3) The kickout metric	0.0336	0.4069	1

Experiment II: Evaluation Strategy for Model Selection
In the second experiment, we perform model selection on 28 variants of self-learning with different meta-parameter values. Table 5 displays the correlation between model ranks in terms of three evaluation measures: AUC on the accepts, AUC on the unbiased sample and the *kickout* measure.

The absolute value of rank correlations between the performance on the accepts and performance on the unbiased data does not exceed 0.01. In contrast, the rankings based on *kickout* are positively correlated with those on the unbiased sample ($r = 0.41$). Therefore, the common practice to assess reject inference strategies using the model's performance on the accepted cases provides misleading results as there is a very small correlation with the performance on the production stage. In contrast, comparing reject inference techniques using the proposed *kickout* measure is more promising.

Figure 3 illustrates the advantages of using *kickout* instead of the performance on the accepts for model selection. Red points indicate the predictive performance of a scoring model selected by the *kickout* measure, while green dots refer to the best-performing model on the accepts in terms of AUC, BS and RP. As before, we evaluate the selected scoring models on the unbiased sample.

As shown in Figure 3, using the *kickout* measure results in selecting a better model in terms of all three performance indicators. By relying on *kickout* instead of the performance on the accepts, we are able to identify a scorecard that has a better performance on the unbiased sample.

These results emphasize the importance of using a suitable evaluation strategy to assess the value of reject inference. Relying on conventional evaluation measures such as AUC that are estimated on the accepted cases would result

in selecting a suboptimal scoring model in terms of its production-stage performance. Our experiments show that *kickout* proves to be a suitable measure for doing model selection. According to the results, the *kickout* measure identifies a better scoring model in the absence of an unbiased sample, which is particularly useful for practitioners.

5 Conclusion

This paper suggests a self-learning framework with distinct training and labeling regimes for reject inference in credit scoring and develops a novel evaluation measure for model selection. We evaluate the effectiveness of our approach by running empirical experiments on a high-dimensional real-world credit scoring data set with unique properties.
Empirical results indicate that the proposed self-learning framework outperforms regular self-learning and conventional reject inference techniques in terms of three performance measures. These results indicate that the modifications suggested here help to adjust self-learning to the reject inference problem.

We also develop a novel evaluation measure to perform model selection for reject inference techniques. We show that the standard practice of selecting models (or meta-parameters) based on their performance on the accepted cases may lead to choosing a model with a suboptimal predictive performance at the production stage. Compared to the standard approach, the proposed *kickout* measure exhibits a higher correlation with the performance on the unbiased sample and allows to identify a scoring model with better performance.

Our results imply that future research on reject inference should not rely on the model's performance on the accepted cases to judge the value of a certain reject inference technique. The *kickout* measure proves to be a good alternative for practitioners who often do not have access to an unbiased sample that contains both accepted and rejected applications.

References

1. Anderson, B., Hardin, J.M.: Modified logistic regression using the EM algorithm for reject inference. IJDATS **5**(4), 359–373 (2013)
2. Ash, D., Meester, S.: Best practices in reject inference. Presentation at Credit Risk Modeling and Decision Conference. Wharton Financial Institutions Center, Philadelphia, May (2002)
3. Banasik, J., Crook, J.: Credit scoring, augmentation and lean models. Journal of the Operational Research Society **56**(9), 1072–1081 (2005)
4. Banasik, J., Crook, J.: Reject inference, augmentation, and sample selection. European Journal of Operational Research **183**(3), 1582–1594 (2007)
5. Banasik, J., Crook, J.: Reject inference in survival analysis by augmentation. Journal of the Operational Research Society **61**(3), 473–485 (2010)
6. Banasik, J., Crook, J., Thomas, L.: Sample selection bias in credit scoring models. Journal of the Operational Research Society **54**(8), 822–832 (2003)

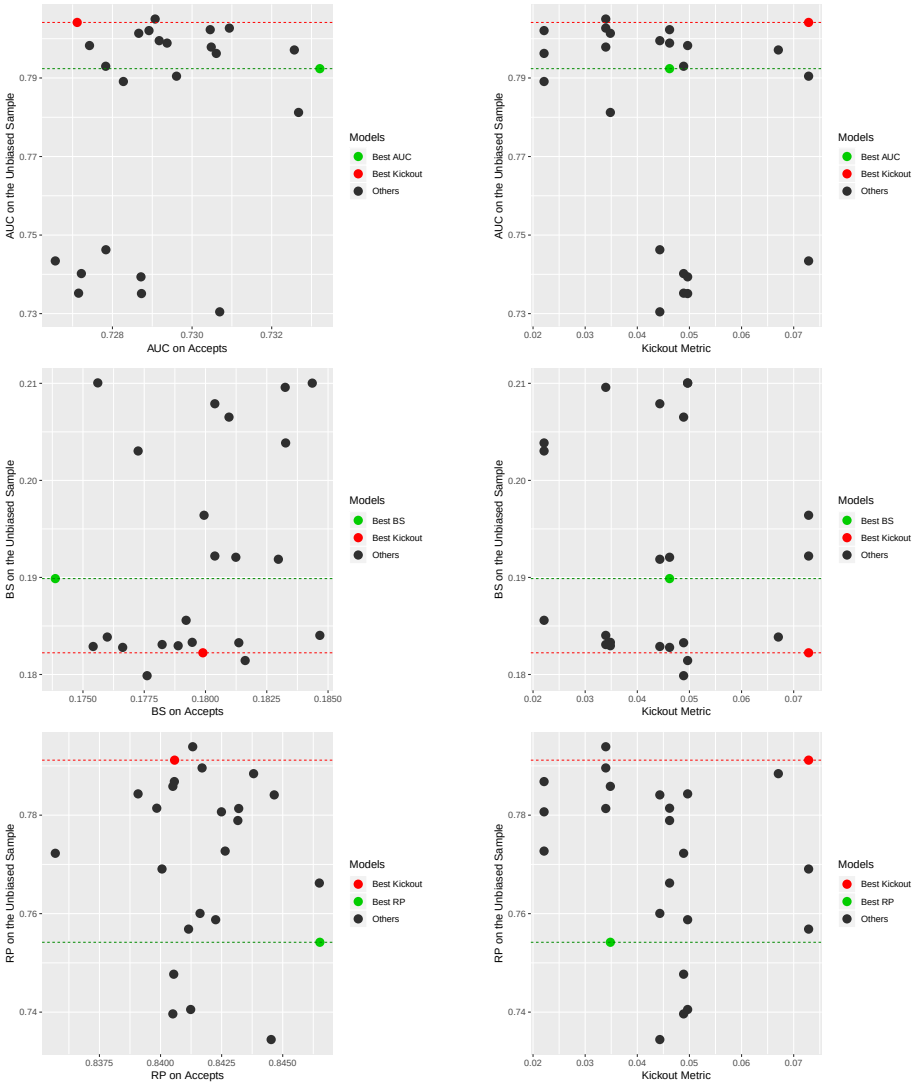

Fig. 3. *Model Selection Results.* Two upper diagrams compare results based on AUC on the accepts (green) and on the kickout metric (red). Two diagrams in the center compare results based on Brier Score on the accepts (green) and kickout (red). Two lower diagrams refer to R-Precision on the accepts (green) and kickout (red).

7. Bücker, M., van Kampen, M., Krämer, W.: Reject inference in consumer credit scoring with nonignorable missing data. Journal of Banking & Finance **37**(3), 1040–1045 (2013)
8. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press (2006)

9. Chen, G.G., Astebro, T.: The economic value of reject inference in credit scoring. Department of Management Science, University of Waterloo (2001)
10. Chen, G.G., Åstebro, T.: Bound and collapse bayesian reject inference for credit scoring. Journal of the Operational Research Society **63**(10), 1374–1387 (2012)
11. Crook, J., Banasik, J.: Does reject inference really improve the performance of application scoring models? Journal of Banking & Finance **28**(4), 857–874 (2004)
12. Feelders, A.: Credit scoring and reject inference with mixture models. Intelligent Systems in Accounting, Finance & Management **9**(1), 1–8 (2000)
13. Fogarty, D.J.: Multiple imputation as a missing data approach to reject inference on consumer credit scoring. Interstat **41**, 1–41 (2006)
14. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences **180**(10), 2044–2064 (2010)
15. Hand, D.J., Henley, W.E.: Can reject inference ever work? IMA Journal of Management Mathematics **5**(1), 45–55 (1993)
16. Joanes, D.N.: Reject inference applied to logistic regression for credit scoring. IMA Journal of Management Mathematics **5**(1), 35–43 (1993)
17. Kim, Y., Sohn, S.: Technology scoring model considering rejected applicants and effect of reject inference. Journal of the Operational Research Society **58**(10), 1341–1347 (2007)
18. Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C.: Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research **247**(1), 124–136 (2015)
19. Li, Z., Tian, Y., Li, K., Zhou, F., Yang, W.: Reject inference in credit scoring using semi-supervised support vector machines. Expert Systems with Applications **74**, 105–114 (2017)
20. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422. IEEE (2008)
21. Maldonado, S., Paredes, G.: A semi-supervised approach for reject inference in credit scoring using svms. In: Industrial Conference on Data Mining. pp. 558–571. Springer (2010)
22. Montrichard, D.: Reject inference methodologies in credit risk modeling. In: The Proceedings of the South-East SAS Users Group (2007)
23. Niculescu-Mizil, A., Caruana, R.: Obtaining calibrated probabilities from boosting. In: UAI. p. 413 (2005)
24. Reichert, A.K., Cho, C.C., Wagner, G.M.: An examination of the conceptual issues involved in developing credit-scoring models. Journal of Business & Economic Statistics **1**(2), 101–114 (1983)
25. Rosenberg, C., Hebert, M., Schneiderman, H.: Semi-supervised self-training of object detection models. In: 2005 Seventh IEEE Workshops on Applications of Computer Vision. vol. 1, pp. 29–36. IEEE (2005)
26. Rosset, S.: Model selection via the auc. In: Proceedings of the 21st International Conference on Machine Learning. p. 89. ACM (2004)
27. Schebesch, K.B., Stecking, R.: Using multiple svm models for unbalanced credit scoring data sets. In: Data Analysis, Machine Learning and Applications, pp. 515–522. Springer (2008)
28. Siddiqi, N.: Credit risk scorecards: developing and implementing intelligent credit scoring, vol. 3. John Wiley & Sons (2012)

29. Smith, A., Elkan, C.: A bayesian network framework for reject inference. In: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 286–295. ACM (2004)
30. Verbaeten, S., Van Assche, A.: Ensemble methods for noise elimination in classification problems. In: International Workshop on Multiple Classifier Systems. pp. 317–325. Springer (2003)
31. Verbraken, T., Bravo, C., Weber, R., Baesens, B.: Development and application of consumer credit scoring models using profit-based classification measures. European Journal of Operational Research **238**(2), 505–513 (2014)
32. Verstraeten, G., Van den Poel, D.: The impact of sample bias on consumer credit scoring performance and profitability. Journal of the Operational Research Society **56**(8), 981–992 (2005)
33. Wang, D., Zhang, Z., Bai, R., Mao, Y.: A hybrid system with filter approach and multiple population genetic algorithm for feature selection in credit scoring. Journal of Computational and Applied Mathematics **329**, 307–321 (2018)
34. Wang, G., Hao, J.x., Ma, J., Huang, L.h.: Empirical evaluation of ensemble learning for credit scoring. In: Machine Learning: Concepts, Methodologies, Tools and Applications, pp. 1108–1127. IGI Global (2012)